Digital Forensik

Dr. Ir. Agus Wibowo, M.Kom., M.Si., MM.

YAYASAN PRIMA AGUS TEKNIK

DIGITAL FORENSIK

Penulis :
Dr. Ir. Agus Wibowo, M.Kom., M.Si., MM.

ISBN :

Editor :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

Penyunting :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom.

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Puji Syukur penulis panjatkan atas terselesaikan nya buku yang berjudul ***"Digital Forensik"*** dengan baik. Peran dan tanggung jawab penyelidik forensik dunia maya adalah untuk secara akurat melaporkan tindakan yang diambil untuk mengidentifikasi, mengekstrak, dan menganalisis data tersebut secara ahli yang pada akhirnya akan mewakili materi bukti sebagai bagian dari penyelidikan terhadap seseorang yang diduga terlibat dalam kegiatan yang tidak sah. Sebagai seorang ahli, penyelidik forensik dunia maya yang sangat bergantung pada hasil perangkat lunak forensik yang dihasilkan secara otomatis, tanpa pengetahuan yang mendalam tentang bagaimana hasil telah dicapai, tidak hanya mempertaruhkan reputasi profesionalnya tetapi juga potensi kerugian.

Buku ini akan memberi Anda pengetahuan khusus tentang validitas data yang diidentifikasi, diakses, dan dianalisis sebagai bagian dari penyelidikan forensik dunia maya yang komprehensif. Buku ini juga akan dimulai dari yang kecil, bahkan sangat kecil seperti bit dan byte, menjelaskan asal-usul data dan kemajuan selanjutnya, membahas konsep yang terkait dengan penyimpanan data, catatan boot, partisi, volume, dan sistem file, dan bagaimana masing-masing hal ini saling terkait dan penting dalam penyelidikan forensik dunia maya. Peran yang dimainkan masing-masing dalam investigasi dan jenis data bukti apa yang dapat diidentifikasi dalam masing-masing bidang ini.

Dalam buku ini terbagi menjadi 12 bab, bab pertama menjelaskan tentang dasar-dasar pengetahuan dalam dunia digital. Melanjutkan ke bab 2 akan berfokus pada binary data yang diolah kedalam bentuk digital. Bab 3 membahas tentang bentuk heksadesimal dalam menemukan potongan data. Beb 4 buku ini tentang File, bertumpu pada pemformatan data yang akan diolah dalam digital forensik, dalam bab ini juga akan memberikan pengayaan tentang exstensi file pada umumnya. Melangkah ke bab selanjutnya yang menjelaskan tentang dasar pemahaman tentang proses boot, BIOS, dan keterkaitannya dengan proses forensik dunia maya yang dijabarkan dalam bab 5. Bab 6 akan menjelaskan tentang Endianness dan Tabel Partisi, endianness merupakan bagaimana data multibyte direpresentasikan oleh sistem komputer dan ditentukan oleh arsitektur CPU dari sistem tersebut, untuk lebih jelasnya akan diuraikan pentingnya endianness dan partisi dalam penyelidikan digital.

Bab 7 memperkenalkan dan membahas lebih lanjut konsep alamat blok logis dan sistem file, penyimpanan dan representasi data dan pentingnya proses investigasi forensik cyber dan penyelidik forensik cyber. Bab 8 yang mencakup sistem file dan konsep yang terkait dengan "pemasangan" data ke dalam informasi yang dapat diidentifikasi. BAB 9 berfokus pada sistem file di luar FAT, yang paling mungkin ditemui oleh penyelidik forensik dunia maya. Bab 10 akan memberikan praktik cerdas investigasi dan langkah-langkahyang diambil selama proses investigasi forensik dunia maya. Lanjut ke Bab 11, membahasan tentang pentingnya waktu, baik untuk penyelidikan maupun penyelidik forensik dunia maya, di Bab 12 kita akan bergabung kembali dengan penyelidikan Ronelle terhadap aktivitas Jose, pada langkah

penutup dari proses Praktik Cerdas Investigasi yang dimulai di Bab 10 hingga mengkomunikasikan temuan dalam penyelidikan digital. Bab akhir buku ini membahas proses di mana data berasal, disimpan, dipindahkan, dimanipulasi, dan dianalisis untuk menilai relevansinya sebagai bukti. Buku ini akan memberi Anda pemeriksaan dan diskusi komprehensif tentang ilmu investigasi forensik dunia maya, apa yang terjadi di balik layar hingga data dan mengapa, apa yang harus dicari dan di mana menemukannya secara logis, dari data hingga bukti digital. Akhir kata semoga buku ini berguna bagi para pembaca.

Semarang, Maret 2023
Penulis

Dr. Ir. Agus Wibowo, M.Kom., M.Si., MM.

DAFTAR ISI

BAB 1
DASAR-DASAR DATA

Sebelum seseorang dapat mengaku memiliki pengetahuan dan sepenuhnya menyadari keluasan yang mencakup disiplin profesional forensik dunia maya, sebuah yayasan, yang berakar pada dasar-dasar teknologi informasi, penyimpanan, penanganan, dan pemrosesan data, serta bagaimana data dipindahkan dan dimanipulasi, sangat penting. Bagi penyelidik forensik dunia maya, data adalah bukti. Memahami bagaimana bukti muncul dari data sangatlah penting; namun, yang lebih penting adalah mampu mengartikulasikan dengan percaya diri bagaimana data bukti diidentifikasi, dikumpulkan, dan diproses.

Sebagai penyelidik forensik dunia maya, hanya dengan menekan tombol atau mencentang opsi dalam rangkaian perangkat lunak forensik, tanpa mengetahui apa yang terjadi di balik layar, menimbulkan potensi pertanggungjawaban. Memahami "siklus hidup" data sangat penting, dari awalnya yang sederhana sebagai bit elektronik, berkembang menjadi byte, karakter, lalu kata, akhirnya muncul sebagai bahasa, sebagai informasi, dan mungkin akhirnya sebagai bukti.

Buku ini akan menyediakan platform untuk memperluas sekaligus meningkatkan keterampilan Anda dalam elemen dasar teknologi informasi karena teknologi tersebut mendukung dan tertanam dalam ilmu investigasi forensik dunia maya. Saat Anda membaca buku ini, Anda akan menemukan kata-kata yang dicetak miring.

Kata-kata ini mewakili konsep-konsep kunci dan lebih lengkap didefinisikan oleh definisi kerja, yang termasuk dalam glosarium di akhir buku. Jika Anda menginginkan penjelasan tentang kata yang dicetak miring, silakan lihat glosarium ini. Seperti kebanyakan tugas, seseorang harus merangkak sebelum berjalan dan tentu saja sebelum berlari dengan kecepatan penuh. Oleh karena itu, bab pertama kami dimulai secara alami, di awal, dengan diskusi tentang blok bangunan utama data dan bagaimana sebagai masyarakat kita manusia berbasis karbon telah belajar berkomunikasi dengan teknologi berbasis silikon — komputer.

1.1 SISTEM PANGGANGAN DASAR 2: ENCODING BINER DAN KARAKTER

Manusia modern menggunakan set karakter (atau huruf) untuk mewakili suara dan kata-kata tertulis. Dalam banyak abjad, termasuk abjad berbasis Latin, setiap simbol atau huruf memiliki bunyi fonetiknya sendiri. Huruf (atau kombinasi huruf, seperti "ph") dipasangkan dengan bunyi yang sesuai, membentuk kode karakter. Melalui kombinasi simbol atau huruf inilah manusia menghasilkan kata-kata, kemudian frasa, dan akhirnya komunikasi yang kompleks.

Karakter simbolik, seperti simbol alfanumerik yang ditemukan dalam bahasa berbasis Latin, bekerja cukup baik untuk daya komputasi kompleks otak manusia. Komputer, bagaimanapun, belum berkembang ke tingkat yang mampu secara tepat menduplikasi pemrosesan kompleks—secara konsisten, lancar, dan andal—dari otak manusia. Saat ini, komputer dapat berkomunikasi paling baik dengan komputer lain, dengan cara yang

didasarkan pada prinsip matematika dasar. Komunikasi komputer-ke-manusia, meski telah berevolusi ke tingkat replikasi suara tertentu, masih didasarkan, sekali lagi, pada prinsip-prinsip matematika dasar. Metodologi saat ini untuk transfer data digital disebut biner, dan itu adalah dasar untuk semua teknologi komputasi. Untuk memahami bagaimana komputer menangani, memindahkan, menyimpan, mengakses, menyimpan, menghapus, atau memanipulasi data, penting untuk terlebih dahulu memahami konsep sistem biner.

Biner adalah nama yang diberikan untuk sistem penomoran Basis 2 atau skema pengkodean. Seperti yang tersirat dari nama Basis 2, ada dua dan hanya dua kemungkinan keadaan. Faktanya, skema pengkodean Basis 2 adalah satu-satunya pilihan komunikasi ketika hanya ada dua kemungkinan keadaan. Skema pengkodean seperti itu bekerja dengan baik dengan komunikasi elektronik.

Pertimbangkan listrik, di mana hanya ada dua keadaan. Listrik hidup atau mati; tidak ada pilihan atau keadaan lain yang memungkinkan. Sirkuit terbuka atau tertutup. Jadi, jika kita memasang bola lampu ke sirkuit listrik, kita dapat melihat secara visual ketika sirkuit terbuka atau tertutup, karena lampu masing-masing akan mati atau menyala (ingat, sirkuit tertutup berarti loop tertutup, dan karenanya pada).

1.2 KOMUNIKASI DUA NEGARA

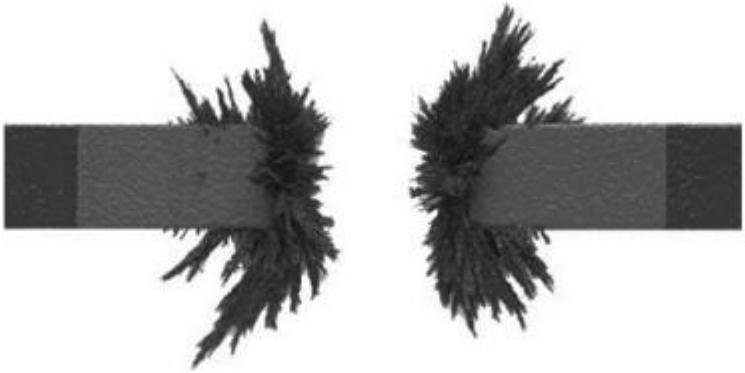
Komunikasi dalam lingkungan dua negara kini dimungkinkan; lampu menyala, sama dengan "ya", atau mati, sama dengan "tidak". Potensi untuk menjawab pertanyaan yang belum sempurna dan tertutup hanya dengan menunjukkan jawaban sebagai "ya" (1) atau "tidak" (0) sepenuhnya dapat dilakukan.

Ini penting, mengingat saat ini, komputer pada dasarnya dapat meneruskan atau menyimpan informasi sebagai keadaan listrik atau magnet. Ingat bola lampu kita hanya bisa "on" atau "off".

Tanpa membahas secara mendetail tentang dasar-dasar kelistrikan atau magnetisme, mungkin perlu mempelajari dengan sangat hati-hati konsep dasar kemagnetan dan kelistrikan, dan hubungannya dengan konstruksi, penyimpanan, mobilitas, dan pemrosesan data.

1.3 LISTRIK DAN MAGNETISME

Magnetisme adalah gaya di mana benda-benda tertarik atau ditolak satu sama lain. Biasanya benda-benda tersebut adalah logam seperti besi. (Lihat Gambar 1.1.) Magnet dapat menyimpan listrik, seperti dalam baterai, misalnya. Magnet juga dapat menghasilkan listrik (misalnya generator). Magnetisme, seperti listrik, hanya memiliki dua keadaan atau kutub yang berlawanan, positif dan negatif. Keadaan magnetis juga dapat ditampung atau dipertahankan; misalnya, arah serutan oksida besi dapat dimanipulasi oleh magnet. Ini disebut domain magnetik, yang merupakan rangkaian atom yang mengarahkan kutubnya ke arah yang sama. Sebuah magnet batang terdiri dari sekelompok domain.



Gambar 1.1 Gaya Magnet

Sumber medan magnet yang paling umum adalah loop arus listrik. Listrik adalah suatu jenis kegiatan yang timbul karena adanya muatan. Satuan dasar muatan adalah pada proton atau elektron. Muatan proton disebut positif sedangkan muatan elektron negatif.

Listrik cenderung bergerak atau mengalir dalam keadaan aktifnya. Karena itu, listrik bagus untuk merepresentasikan data bergerak dan magnet bagus untuk merepresentasikan data diam. Keduanya, bagaimanapun, memiliki dua keadaan yang terpisah dan berlawanan, dan seperti yang telah dibahas, memiliki dua keadaan terpisah memungkinkan komunikasi digital Base 2.

Dengan komputer, pergerakan data digital dengan mudah diwakili oleh dua keadaan listrik atau magnet, dan dengan mudah disajikan masing-masing dengan 1 dan 0. Oleh karena itu, karena teknologi yang digunakan untuk berkomunikasi dan merepresentasikan data saat ini sudah ada, representasi ini dilakukan melalui sistem penomoran dua-status atau biner.

1.4 BUILDING BLOK: ASAL USUL DATA

Satu nol (0) atau satu (1) sama dengan apa yang disebut bit. Representasi dari dua kemungkinan keadaan data digital ini adalah unit terkecil dari data yang dikenali atau diproses oleh komputer.

Oleh karena itu, dalam satu-bit, skema pengkodean Basis 2 (atau seperti yang biasa disebut, biner), kami memiliki blok bangunan dasar dari sistem komunikasi: kemampuan untuk berkomunikasi melalui dan di antara teknologi berbasis silikon. Untuk mengomunikasikan, misalnya, status lampu menyala, kita dapat menetapkan nilai satu (1). Untuk mengomunikasikan bahwa lampu dimatikan, kita dapat dengan mudah mengatur nilainya menjadi nol (0). Untuk situasi yang lebih kompleks, untuk menunjukkan hidup atau mati (ya atau tidak), kita dapat menetapkan nilai yang sama: ya/hidup sama dengan satu (1), atau tidak/mati sama dengan nol (0). Saat komunikasi tumbuh dalam keluasan dan kompleksitas, kami dibatasi oleh skema pengkodean Base 2 bit tunggal. Pada dasarnya, kami memiliki dua dan hanya dua kemungkinan hasil komunikasi ketika dibatasi pada satu bit (misalnya, ya atau tidak, hidup atau mati, 1 atau 0).

Komunikasi dimungkinkan, kemudian, ketika hanya dua keadaan atau kondisi yang diperlukan. Begitu kita ingin memperluas kemungkinan opsi komunikasi ke leksikon yang lebih luas di luar status dua opsi, satu bit gagal, sangat membatasi kemungkinan komunikasi.

1.5 MENUMBUHKAN BUILDING BLOK DATA

Namun, saat Anda menghubungkan 0 dan 1 (atau bit) berurutan, kemampuan untuk merepresentasikan serangkaian karakter, kata, komunikasi, dan kemungkinan pengiriman pesan yang semakin besar meningkat secara geometris. Hanya dengan menambahkan sedikit lagi, kita menggandakan hasil atau status potensial (dari dua menjadi empat). Ada dua kemungkinan status dengan satu bit: satu (1) atau nol (0). Tambahkan bit lain dan sekarang jumlah status yang mungkin menjadi dua kali lipat: 00, 01, 10, dan 11. Berbekal sistem seperti itu, kita sekarang dapat mewakili ide yang lebih kompleks atau kondisi yang membutuhkan lebih dari sekadar penyederhanaan, hidup/mati, ya/tidak, deskripsi dua negara. Misalnya, empat musim sekarang dapat digambarkan dengan dua bit, misalnya 00 = musim dingin, 01 = musim semi, 10 = musim panas, dan 11 = musim gugur.

Untuk lebih memahami pertumbuhan geometris dari kemungkinan hasil yang dicapai dengan menggabungkan bit, mari kita lihat beberapa contoh. Diskusi berikut mungkin membuat banyak pembaca merinding, mengingat kembali masa-masa muda dan pemikiran bahwa tantangan matematika sudah berlalu; namun, pemahaman tentang prinsip matematika dasar ini sangat penting dalam memahami detail penyimpanan data yang berfungsi lebih baik dan pada akhirnya ekstraksi data menggunakan perangkat lunak forensik.

Apa 2 pangkat 0?

Penjelasan singkat, yang mengharuskan kita menggunakan hukum eksponen, mungkin berguna untuk menghidupkan sinapsis matematika tersebut. Salah satu hukum eksponen adalah:

$$\frac{n\hat{}x}{n\hat{}y} = n\hat{}(x-y)$$

untuk semua n, x, dan y. Jadi, misalnya,

$$\frac{2\hat{}4}{2\hat{}2} = 2\hat{}(4-2)$$

$$\frac{2\hat{}4}{2\hat{}3} = 2\hat{}(4-3)$$

Sekarang misalkan kita memiliki pecahan:

$$\frac{2\hat{}4}{2\hat{}4} = 1$$

Pecahan ini sama dengan 1, karena pembilang dan penyebutnya sama. Jika kita menerapkan hukum eksponen, kita mendapatkan:

$$1 = \frac{2\text{^}4}{2\text{^}4} = 2\text{^}(4-4) = 2\text{^}0$$

$$\text{So, } 2\text{^}0 = 1$$

Kita dapat memasukkan angka apa saja sebagai pengganti 2, dan angka yang dipangkatkan nol itu akan tetap menjadi 1. Nyatanya, seluruh pembuktian berhasil jika kita hanya memasukkan x untuk 2:

$$x\text{^}0 = x\text{^}(4-4) = \frac{x\text{^}4}{x\text{^}4} = 1$$

Wow, kilas balik matematika—kami membuktikan bahwa 2 ^ 0 sama dengan 1, lalu bagaimana dengan yang berikut ini:

Berapa 2 pangkat satu? Kekuatan kedua? Kekuatan ketiga?

Nah, tentunya kita akan menjawab 2 × 1 = 2, 2 × 2 = 4, dan 2 × 2 × 2 = 8!

Mengapa ini penting? Ini memberi kita cara yang lebih baik untuk memahami pertumbuhan geometris dari kemungkinan keadaan atau hasil yang dicapai dengan menggabungkan bit.

Berikut ini adalah contoh kecil dari kekuatan 2 dan pertumbuhan eksponensial meningkatkan kemungkinan kombinasi bit:

2^0 = 1
2^1 = 2
2^2 = 4
2^3 = 8
2^4 = 16
2^5 = 32
2^6 = 64
2^7 = 128
2^8 = 256
2^9 = 512
2^10 = 1,024

Dari pertanyaan kita sebelumnya, “Berapa pangkat 2 pangkat tiga?” kami menemukan jawabannya dalam skema pengkodean kami. Angka 2 mewakili skema pengkodean kami, Basis 2 atau biner; kekuatan (ketiga) menunjukkan berapa banyak bit yang akan dirangkai. Jawaban 8 adalah berapa banyak hasil atau kombinasi yang mungkin jika kita dapat merangkai tiga bit: 000, 111, 001, 010, 100, 110, 101, 011. Itu dia! Ada 8 hasil yang mungkin, jadi 2 ke yang ketiga = 8.

1.6 BERGERAK DI LUAR DASAR 2

Delapan hasil atau kombinasi yang mungkin masih cukup membatasi untuk komunikasi manusia yang kompleks, sebagaimana diperlukan dalam ekonomi bisnis global saat ini. Saat kami terus menambahkan 0 dan 1, potensi pensinyalan atau komunikasi digital yang sangat kompleks meningkat, sebagaimana dinyatakan secara eksponensial. Faktanya, jika kita merangkai cukup banyak bit, kita akan dapat merepresentasikan abjad lengkap, abjad dari lebih dari satu bahasa, dan abjad bahkan untuk mewakili konsep dan ekspresi grafis.

Untuk merepresentasikan alfabet bahasa Inggris (A–Z) dan angka 1 hingga 10, kita memerlukan 26 representasi unik untuk huruf dan 10 untuk angka (0–9). Jadi, kami membutuhkan 36 pengidentifikasi unik. Berapa banyak bit yang diperlukan untuk mewakili 36 pengidentifikasi atau hasil unik?

Nah, dari pelajaran matematika kita sebelumnya, 6 bit akan dengan mudah memenuhi kebutuhan kita, menghasilkan 2 ke enam atau 2 ^ 6, direpresentasikan sebagai hasil 2 × 2 = 4 × 2 = 8 × 2 = 16 × 2 = 32 × 2 = 64.

Kombinasi enam-bit ini tidak hanya menghasilkan 36 pengenal unik yang diperlukan, tetapi juga memberi kita beberapa pengenal unik untuk disimpan, 28 lebih spesifik, yang dapat kita gunakan untuk mewakili simbol khusus seperti (!, @, #, $, %, ^, &,*) dan seterusnya. Nyatanya, 64 karakter unik, meski signifikan dalam jumlah kombinasi yang mungkin, tidaklah cukup. Dalam merepresentasikan sebagian besar karakter dasar bahasa Inggris, kami menggunakan 7 bit atau 2 ^ 7, menghasilkan 128 karakter unik untuk diidentifikasi atau dipetakan.

1.7 KODE STANDAR AMERIKA UNTUK PERTUKARAN INFORMASI

Sejarah Kode Standar Amerika untuk Pertukaran Informasi (ASCII) dan perkembangannya merupakan cerita panjang dan hanya akan disinggung secara singkat dalam bab ini. Karakter yang diidentifikasi oleh 2 ^ 7, atau karakter unik 128-bit untuk diidentifikasi atau dipetakan, dikenal sebagai Kode Standar Amerika untuk Pertukaran Informasi/ASCII yang diperluas atau hanya ASCII.

Komputer pribadi berbahasa Inggris yang digunakan di Amerika menggunakan kode karakter tujuh bit yang disebut Kode Standar Amerika untuk Pertukaran Informasi (ASCII), yang memungkinkan kumpulan karakter berisi 128 item huruf Latin besar dan kecil, angka Arab, tanda, dan mengontrol karakter (yaitu, 2 ^ 7 = 128 poin kode). ASCII juga berfungsi sebagai dasar dari Universal Character Set (UCS), berisi 0–9, A–Z, a–z, dan karakter khusus).

Ketika bit kedelapan digunakan sebagai "bit paritas", dengan nilainya digunakan untuk memeriksa apakah data telah ditransmisikan dengan benar atau tidak, maka ASCII menjadi delapan bit, atau satu byte (delapan bit = satu byte), kode karakter. Kode karakter delapan bit yang sebenarnya memungkinkan hingga 256 item untuk dikodekan (2^8 = 256 poin kode). ASCII adalah skema pengkodean karakter berdasarkan urutan abjad Inggris. Kode ASCII mewakili teks di komputer, peralatan komunikasi, dan perangkat lain yang menggunakan teks. Sebagian besar skema pengkodean karakter modern, yang mendukung lebih banyak karakter daripada aslinya, didasarkan pada ASCII.

Tabel 1.1 dan 1.2 menyoroti skema pengkodean ASCII dan ekuivalen biner terkait. Tabel 1.1 menyajikan angka 0–9 dan Tabel 1.2 menggambarkan daftar karakter khusus. Karakter abjad dari bahasa Inggris memiliki representasi serupa dalam skema pengkodean ASCII, seperti yang ditunjukkan pada Tabel 1.3.

Tabel 1.1 Bilangan Diwakili oleh 0–9

Karakter	ASCII	Biner
0	chr(48)	110000
1	chr(49)	110001
2	chr(50)	110010
3	chr(51)	110011
4	chr(52)	110100
5	chr(53)	110101
6	chr(54)	110110
7	chr(55)	110111
8	chr(56)	111000
9	chr(57)	111001

Tabel 1.2 Representasi Karakter Khusus

Karakter	ASCII	Biner
!	chr(33)	100001
“	chr(34)	100010
#	chr(35)	100011
$	chr(36)	100100
%	chr(37)	100101
&	chr(38)	100110
'	chr(39)	100111
(	chr(40)	101000
)	chr(41)	101001
*	chr(42)	101010
+	chr(43)	101011

Tabel 1.3 Representasi Abjad dalam Skema Pengkodean ASCII

Karakter	ASCII	Biner
A	chr(65)	1000001
B	chr(66)	1000010
C	chr(67)	1000011
D	chr(68)	1000100
E	chr(69)	1000101
F	chr(70)	1000110
G	chr(71)	1000111

H	chr(72)	1001000
I	chr(73)	1001001
J	chr(74)	1001010
K	chr(75)	1001011
L	chr(76)	1001100
M	chr(77)	1001101

1.8 KODE KARAKTER: DASAR PENGOLAHAN DATA TEKSUAL

Banyak orang tidak menyadari fakta bahwa bagi komputer, data tekstual juga merupakan data numerik. Dalam sistem komputer modern, karakter individu dari skrip yang digunakan manusia untuk merekam dan mengirimkan bahasa mereka dikodekan dalam bentuk kode numerik biner, seperti halnya angka Arab yang digunakan dalam program perhitungan. (Lihat Tabel 1.1, 1.2, dan 1.3.) Ini karena sirkuit mikroprosesor yang terletak di jantung sistem komputer modern hanya dapat melakukan dua hal—menghitung operasi aritmatika biner dan menjalankan logika Boolean (yaitu, benar atau salah) operasi.

Kode karakter memasangkan kumpulan karakter, seperti alfabet, dengan sesuatu yang lain, dalam hal ini dengan sistem desimal dan/atau biner. Contoh yang paling familiar adalah Sistem Pengkodean Braille. Sementara beberapa dari kita mungkin tidak tahu apa arti titik-titik yang disandikan Braille, kita telah melihatnya, karena banyak lift menampilkan nomor lantai bersama dengan mitra Braille-nya. Kombinasi informasi ini akan dianggap sebagai kode karakter. Karakter maksimum yang mungkin dalam kode karakter bergantung pada sistem penomoran (Basis 2 untuk Biner) dan jumlah bit. Seperti yang ditunjukkan sebelumnya, semakin banyak bit dalam kode karakter, semakin besar rangkaian karakternya. Berkenaan dengan kode karakter, perlu juga dicatat bahwa komputer beroperasi paling efisien saat memproses data dalam byte. Ini karena sirkuit internal mereka biasanya dirancang dengan jalur data selebar 8, 16, 32, atau 64 bit. Oleh karena itu, kode karakter 10-bit atau 15-bit adalah kikuk dan tidak efisien untuk ditangani di dalam komputer pribadi.

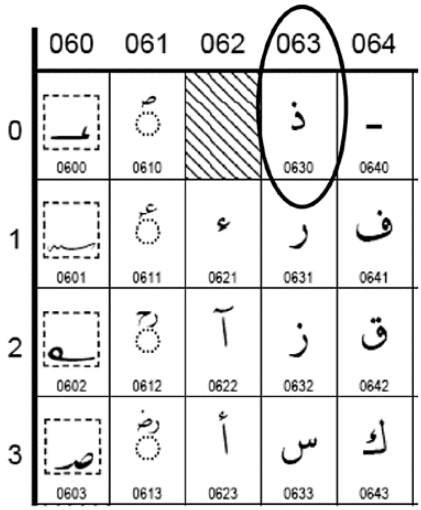
Di sisi lain, jika terlalu banyak byte digunakan untuk mengkodekan karakter, komputer akan cenderung memproses data secara tidak efisien. Misalnya, kode karakter tiga byte dapat menyandikan hampir 17 juta karakter (2 ^ 24 = 16.777.216 titik kode), yang akan mencakup semua set karakter historis dan yang saat ini digunakan di seluruh dunia. Tetapi sebagian besar bahasa di dunia hanya memerlukan kode satu byte (delapan bit) untuk pengkodean karakter, karena merupakan skrip abjad.

1.9 ASCII DAN UNICODE YANG DIPERLUAS

Karena orang secara bertahap membutuhkan komputer untuk memahami karakter tambahan, set ASCII menjadi terbatas. Extended ASCII adalah skema pengkodean delapan bit yang mencakup karakter ASCII tujuh bit standar serta yang lainnya.

Unicode adalah standar industri yang dikembangkan oleh Unicode Consortium. Standar Unicode adalah sistem pengkodean karakter yang dirancang untuk mendukung pertukaran, pemrosesan, dan tampilan teks tertulis di seluruh dunia dari berbagai bahasa dan

disiplin teknis dunia modern. Selain itu, mendukung teks klasik dan sejarah dari banyak bahasa tertulis. Unicode (atau Universal Character Set) adalah skema pemetaan biner lain yang dimaksudkan untuk menjadi lebih universal, dan menyertakan rangkaian karakter yang lebih luas, yang membantu mengakomodasi rangkaian karakter yang benar-benar global. UCS menggabungkan skema pemetaan karakter ASCII awal, memungkinkan kompatibilitas mundur. Unicode secara kasar dapat digambarkan sebagai "ASCII berbadan lebar" yang telah direntangkan dari 8 bit menjadi 16 bit. Unicode juga memungkinkan format biner 8-, 16-, atau 32-bit. Skema pengkodean 16-bit akan memungkinkan 65.536 hasil potensial. 65.536 hasil bit potensial inilah yang memungkinkan Unicode mencakup sebagian besar karakter dari semua bahasa yang hidup di dunia. Gambar 1.2 menunjukkan 20 nilai pertama dari set karakter Arab Unicode, dengan huruf Arab THAL yang disorot.



Gambar 1.2 Rentang Bahasa Arab 0600–06FF—Unicode Arabic Range 0600–06FF, Unicode Standard 5.2,

Gambar 1.3 Huruf Arab THAL: Nilai Unicode 0630

Jadi, jika pengguna ingin menghasilkan huruf Arab THAL, individu tersebut akan menggunakan nilai Unicode 0630. Nilai numerik spesifik ini (0630) memiliki arti yang unik dan khusus ketika digunakan dalam komunikasi elektronik, komputasional, dan secara efektif mewakili bahasa Arab. karakter THAL (lihat Gambar 1.3) ke perangkat komputasi, yang saat ini hanya dapat menginterpretasikan dan menghitung nilai numerik.

1.10 RINGKASAN

Bab pertama ini dimulai dengan pengenalan singkat dan diskusi tentang bagaimana sistem komunikasi komputasional telah berevolusi dan bagaimana kita mencoba

mengkodifikasikan kemampuan kita untuk berkomunikasi di dunia dengan hanya dua kemungkinan keadaan, sebuah keberadaan biner.

Kami beralih ke diskusi lebih lanjut, tidak hanya tentang peran yang dimainkan oleh sistem penomoran biner dalam kemampuan kami untuk mewakili pola komunikasi manusia yang paling dasar, tetapi juga bagaimana sistem biner ini memungkinkan kami berkembang untuk menghasilkan pola abjad yang kompleks, karakter. set, dan akhirnya metode yang memungkinkan kita untuk mewakili seluruh bahasa.

Dari keadaan utama keberadaan hingga representasi pola bahasa yang kompleks, komunikasi berbasis karbon, seperti komunikasi berbasis silikon, dimulai dengan blok bangunan utama. Untuk komputer, itu adalah bit yang diwakili oleh satu (1) atau nol (0).

Kombinasi dan pasangan 1 dan 0 ini memungkinkan mesin komputasi untuk berkomunikasi dan pada akhirnya, untuk melakukan manipulasi data yang kompleks. Mewakili kata-kata atau grafik tekstual yang kompleks ke perangkat mekanis sekarang dimungkinkan hanya dengan mengatur dan mengatur ulang pasangan dan pengelompokan 1 dan 0.

Menetapkan metode untuk memasangkan karakter alfabet dengan persamaan biner karakter menghasilkan kode karakter (yang telah berevolusi menjadi kumpulan karakter yang lebih kompleks), memungkinkan kita untuk memperluas kemampuan kita untuk mewakili rentang karakter yang lebih luas dan juga mengontrol bagaimana komputer menyimpan, memanipulasi, dan mengirimkan data.

Representasi biner dari angka dan karakter diperlukan saat bekerja di dunia yang terbatas hanya pada dua keadaan deskripsi atau keberadaan (misalnya listrik atau magnet). Untungnya bagi kita, dunia manusia kita lebih kuat, lebih berwarna, dan ada di banyak negara bagian, jauh melampaui kehidupan biner. Ini juga akan lebih sulit dan memakan waktu jika kita diminta untuk melakukan semua fungsi menghitung, berkomunikasi, dan serupa dengan angka atau huruf yang diwakili oleh kelompok dan pasangan 1 dan 0 (misalnya, pernyataan, "Hai, Nama saya Tom ”akan diwakili oleh serangkaian satu dan nol 144 karakter panjang, 01001000 01101001 00101100 00100000 01101101 01111001 00100000 01101110 01100001 01101101 01100101 00100000 01101001 01110011 00100000 01010100 01101111 01101101).

Untungnya, hanya perangkat komputasi berbasis silikon yang harus memproses data dengan cara biner ini. Manusia, di sisi lain, memiliki metode yang lebih nyaman. Manusia bekerja lebih efektif dan lebih efisien merepresentasikan angka dalam Basis 10 atau desimal yang setara dengan representasi biner komputer, membuat hidup sedikit lebih mudah.

Dalam Bab 2 kita membahas bagaimana mengubah bilangan biner menjadi padanan desimalnya, dan mengapa pengetahuan ini juga penting untuk mendapatkan pemahaman yang lebih mendalam tentang bagaimana data disimpan, dipindahkan, dimanipulasi, dan diproses dan bagaimana perlakuan data ini sangat penting. untuk pemahaman yang lebih baik tentang forensik cyber.

BAB 2
BINER KE DESIMAL

Di bab 1 kita memahami konsep dasar sistem penomoran dan bagaimana data dipindahkan dan dimanipulasi. Siklus hidup data, dari permulaannya yang sederhana sebagai bit dan byte elektronik, berkembang menjadi karakter, kemudian kata-kata, akhirnya muncul sebagai bahasa, kemudian sebagai informasi dan akhirnya menjadi bukti potensial. Memahami bagaimana bukti muncul dari data, sangat penting dalam penyelidikan forensik yang berhasil.

Kami sekarang melanjutkan langkah berikutnya dari proses pembelajaran forensik siber kami, bergerak dari permulaan biner kami yang sederhana dari dunia dua negara kami, berkembang sekarang melampaui biner ke desimal dan kembali lagi, mendapatkan pemahaman yang lebih dalam tentang matematika di balik forensik dan bagaimana pengetahuan tentang matematika sangat penting dalam memahami penyelidikan forensik dunia maya yang paling dasar sekalipun.

Jadi, saat kita menyelidiki lebih dalam untuk memahami apa yang terjadi di balik flash dan desis forensik, mari kita mulai dari bagian terakhir yang kita tinggalkan.

2.1 KODE STANDAR AMERIKA UNTUK PERTUKARAN INFORMASI

Sejarah ASCII dan perkembangannya telah dibahas panjang lebar sebelumnya, dan sekarang kita tahu bahwa karakter yang diidentifikasi oleh 2 ^ 7 atau karakter unik 128-bit dikenal sebagai Kode Standar Amerika untuk Pertukaran Informasi / ASCII yang diperluas atau hanya ASCII.

Karakter ASCII diberi nilai desimal karena biner tidak dapat langsung dikonversi ke ASCII, dan perangkat komputasi berbasis silikon hanya dapat dihitung dalam matematika biner. Namun, biner dapat diubah menjadi nilai desimal, dan nilai desimal ini diberi karakter ASCII, sehingga menyelesaikan siklus. 32 karakter pertama dalam tabel ASCII adalah kode kontrol yang tidak dapat dicetak dan digunakan untuk mengontrol periferal seperti printer. Kode 32-127 adalah umum untuk semua variasi tabel ASCII yang berbeda; mereka disebut karakter yang dapat dicetak, mewakili huruf, angka, tanda baca, dan beberapa simbol lainnya. Anda akan menemukan hampir setiap karakter di keyboard Anda. Karakter 127 mewakili perintah DEL.

Pada Tabel 2.1 kita melihat contoh nilai biner, ekuivalen desimalnya, dan karakter ASCII yang ditetapkan untuk nilai biner tersebut.

Tabel 2.1 Nilai Biner, Setara Desimalnya, dan Kode ASCII

Biner	Desimal	Simbol ASCII	Deskripsi
110000	48	0	Nol
110001	49	1	Satu
110010	50	2	Dua

110011	51	3	Tiga
110100	52	4	Empat
110101	53	5	Lima
110110	54	6	Enam
110111	55	7	Tujuh
111000	56	8	Delapan
111001	57	9	Sembilan
111111	63	?	Tanda tanya
1000000	64	@	Di simbol
1000001	65	A	Huruf besar A
1000010	66	B	Huruf besar B
1000011	67	C	Huruf besar C
1111011	123	{	Kurung bukaan
1111100	124	|	Bilah vertikal
1111101	125	}	Kurung penutup
1111110	126	~	Tanda kesetaraan
1111111	127		Menghapus

2.2 KOMPUTER SEBAGAI KALKULATOR

Komputer mendasarkan fungsinya pada matematika, jadi pada kenyataannya, Mikroprosesor komputer (otaknya) pada dasarnya adalah kalkulator yang dimuliakan. Ini adalah mikroprosesor komputer yang melakukan kalkulasi matematis dari biner ke desimal (melakukannya dengan kecepatan jutaan kalkulasi per detik), dan ini dilakukan “di belakang layar”, artinya kita tidak benar-benar melihat fungsi ini terjadi saat kita menggunakan komputer. Kecepatan instruksi komputer jatuh ke dalam berbagai kisaran, seperti yang ditunjukkan pada Tabel 2.2.

Nilai dan angka biner (desimal) mampu melakukan operasi matematika pada mereka. Karena mereka memiliki kesamaan ini, yang satu dapat diturunkan dari yang lain. Biner tidak dapat dihitung secara matematis menjadi huruf "A" atau karakter Arab, misalnya. Karakter ASCII atau unicode adalah simbol yang dipahami oleh manusia; mereka tidak lebih dari piktogram sejauh menyangkut mikroprosesor.

Tabel 2.2 Kecepatan Mikroprosesor

Mili detik	seperseribu (10^–3) detik
Mikrodetik	sepersejuta (10^–6) detik
Nanodetik	sepersejuta (10^–9) detik
Pikodetik	satu triliun (10^–12) detik
Femtodetik	seperempat triliun (10^–15) detik

Nilai biner dapat dihitung secara matematis menjadi nilai desimal. Dan nilai desimal dapat diberikan ke nilai ASCII seperti yang terlihat pada Tabel 2.1. Nilai desimal direferensikan oleh nilai yang sesuai dalam bagan karakter (ASCII atau UniCode) oleh Sistem Operasi (OS) dan/atau perangkat lunak yang digunakan. Pada akhirnya, perangkat lunaklah yang menerjemahkan informasi menjadi sesuatu yang bermanfaat: gambar, kata-kata, video, dan sebagainya.

Mereferensikan bagan untuk mengonversi nilai desimal menjadi kode karakter ASCII adalah konsep sederhana untuk dipahami; namun, kita perlu menjelaskan lebih rinci untuk menjelaskan kompleksitas yang terkait dengan konversi biner ke desimal.

2.3 MENGAPA INI PENTING DALAM FORENSIK?

Data tidak selalu lengkap. Sebagian besar waktu, faktanya, data tidak lengkap atau tidak ada sama sekali. Bukti ditemukan dalam bit data yang tidak berada dalam format aslinya atau tidak terlihat dalam kode karakter ASCII.

Data tidak mudah dilihat ketika tidak dapat disusun kembali ke dalam “format aslinya” oleh perangkat lunak yang dirancang untuk membaca data. Ini terjadi ketika header atau bagian lain dari dokumen asli ditimpa atau dihapus. Bayangkan menghapus ".doc" dari dokumen kata, lalu mencoba membuka dokumen itu. Apa yang terjadi? Banyak pesan kesalahan untuk satu. Komputer mengalami kesulitan besar untuk membuka (memproses atau menindaklanjuti) sesuatu (file, folder, instruksi, dll.) yang tidak dikenalnya.

Secara forensik, untuk mengekstrak hanya bit dan potongan data penting yang diperlukan (mungkin mewakili dokumen) relatif terhadap penyelidikan, kita harus dapat melihat data yang terkandung dalam dokumen, terlepas dari perangkat lunak yang digunakan untuk menghasilkan dokumen. atau "jenis file asli" dokumen.

2.4 REPRESENTASI DATA

Seperti yang dijelaskan pada Bab 1, satuan data terkecil adalah bit. Delapan bit membentuk satu byte. Delapan bit adalah representasi biner dari sebuah byte yang telah diberi kode karakter atau simbol yang sesuai, baik karakter atau simbol itu dalam bahasa Inggris, Urdu, Cina, atau Sanskerta. Jadi, delapan bit yang mewakili satu byte harus secara matematis "diterjemahkan" menjadi setara desimal yang representatif untuk dipahami oleh kita manusia. Nilai desimal dari delapan bit ditunjukkan pada Tabel 2.3.

Tabel 2.3 Nilai Desimal Delapan Bit

Kekuatan	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Setara Desimal	128	64	32	16	8	4	2	1

Tabel ini menunjukkan Basis 2 pangkat ke-n. Hasil matematis (atau nilai desimal) dari setiap pangkat disajikan di baris kedua. Nilai desimal ini mewakili total kemungkinan hasil (atau keadaan) dari Basis 2 pangkat n. Ini adalah konstanta matematika, 27 akan selalu sama dengan 128.

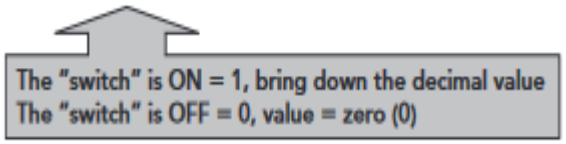
2.5 KONVERSI BINER KE DESIMAL

Karena komputer tidak dapat mengenali atau memproses karakter "&" dalam bentuk aslinya, dan hanya memproses bit yang disimpan dalam biner, dan karena kita manusia tidak memproses informasi biner, bagaimana kita mengubah nilai biner menjadi nilai desimal?

Tabel 2.4 Konversi Biner ke Desimal

Kekuatan	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Setara Desimal	128	64	32	16	8	4	2	1
Nilai Biner	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**

Kekuatan	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
Setara Desimal	128	64	32	16	8	4	2	1	
		↓		↓	↓				
Nilai Biner	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**	
		↓		↓	↓				**Menambahkan nilai desimal**
Nilai Desimal	0	**64**	0	**16**	**8**	0	0	0	0+**64**+0+**16**+**8**+0+0+0 = **88**



Gambar 2.1 Konversi Biner ke Desimal

Mari kita ambil, misalnya, mengubah nilai biner 01011000 menjadi padanan desimalnya. Dengan menggunakan informasi pada Tabel 2.3, kita menambahkan baris lain ke tabel untuk nilai biner kita, 01011000, yang menghasilkan Tabel 2.4. Kunci untuk mengonversi nilai biner menjadi padanan desimalnya adalah keberadaan (atau ketiadaan) "arus" yang diwakili oleh nilai biner dari sakelar "0" atau "1" atau karakter biner. Jika nilai biner ada di tempat penampung, nilai diaktifkan, diwakili oleh nilai satu (1). Jika tidak ada nilai biner yang menempati placeholder, maka nilai tersebut dimatikan, yang diwakili oleh nilai nol (0). Jika sakelar biner (atau nilai) AKTIF ("1"), maka nilai desimal AKTIF, artinya ditambahkan atau dihitung saat menentukan nilai desimal total. Jika sakelar biner MATI ("0"), maka nilai desimal tidak dihitung atau ditambahkan saat menentukan total ekuivalen desimal. Untuk menyelesaikan proses konversi kami, kami menambahkan baris terakhir ke tabel kami untuk mewakili nilai desimal dari nilai biner yang dikonversi. (Lihat Gambar 2.1.)

Satu-satunya Nilai biner yang "diaktifkan" dan diwakili oleh nilai satu, adalah bit keempat, kelima, dan ketujuh, dengan ekuivalen desimal masing-masing 8, 16, dan 64. Cukup menjumlahkan nilai desimal ini (8 + 16 + 64) memberi kita nilai desimal 88, dan ekuivalen desimal dari nilai biner 01011000.

2.6 ANALISIS KONVERSI

Hitung jumlah digit dalam bilangan biner. Untuk setiap digit, tuliskan pangkat 2 dari kanan ke kiri secara berurutan, dimulai dengan 1 sampai Anda memiliki satu pangkat 2 untuk setiap digit. Dalam contoh kita, untuk bilangan biner delapan digit 01011000, Anda akan mencantumkan 128, 64, 32, 16, 8, 4, 2, dan 1.

Hubungkan digit biner dengan pangkat 2 yang sesuai dengan garis lurus. Telusuri bilangan biner dan jika bilangan binernya adalah 1, turunkan pangkat 2 dan tuliskan di kotak yang sesuai pada garis nilai desimal. Jika bilangan binernya adalah 0, masukkan 0 ke dalam kotak. Ubah bilangan biner menjadi desimal dengan menjumlahkan nilai desimal yang Anda masukkan ke dalam setiap kotak. Jumlah angka adalah ekuivalen desimal dari angka biner. Biner 01011000 sama dengan nilai desimal 88.

Contoh Kasus Forensik: Aplikasi Matematika

Nyonya Ronelle Sawyer, penyelidik forensik dunia maya sedang mencari bukti komunikasi antara Tuan Jose McCarthy, seorang ilmuwan riset di ABC Inc., dan Ibu Janice Witcome, direktur pelaksana Perusahaan XYZ, pesaing, dalam kasus yang melibatkan potensi pencurian kekayaan intelektual (IP).

Pertanyaannya kemudian adalah, "Bagaimana Ronelle dapat mengidentifikasi kemunculan nama perusahaan 'XYZ' saat memeriksa isi hard drive Jose?" Jawaban awal, “pandangan manusia” dari data yang dicari Ronelle, sudah diketahui. Ronelle akan mulai mencari kejadian atau referensi apa pun ke "Witcome", "Janice", atau "XYZ" yang mungkin ada di hard drive Jose. Dari tabel ASCII mana pun, seperti yang direproduksi sebagian di Tabel 2.5, Ronelle akan mengidentifikasi karakter ASCII "X", "Y", dan "Z" dan dengan melakukan itu akan dapat memperoleh ekuivalen desimal dari karakter ini: 88 , 89, dan 90, masing-masing. Ronelle sekarang dapat menentukan ekuivalen biner dari nilai desimal 88, 89, dan 90, berdasarkan diskusi kita sebelumnya dengan mengaktifkan bit-bit yang akan memberi Ronelle nilai masing-masing 88, 89, dan 90. (Lihat Tabel 2.6, 2.7, dan 2.8.)

Tabel 2.5 Nilai Desimal dan ASCII

Desimal	Simbol ASCII	Deskripsi
83	S	Huruf besar S
84	T	Huruf besar T
85	U	Huruf besar U
86	V	Huruf besar V
87	W	Huruf besar W
88	X	Huruf besar X
89	Y	Huruf besar Y
90	Z	Huruf besar Z

Tabel 2.6 88 Desimal ke Biner

Kekuatan	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Setara Desimal	128	64	32	16	8	4	2	1
Kode biner	0	**1**	0	**1**	**1**	0	0	0
Nilai Desimal	0	**64**	0	**16**	**8**	0	0	0

Tabel 2.7 89 Desimal ke Biner

Kekuatan	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Setara Desimal	128	64	32	16	8	4	2	1
Kode biner	0	**1**	0	**1**	**1**	0	0	**1**
Nilai Desimal	0	**64**	0	**16**	**8**	0	0	**1**

Tabel 2.8 90 Desimal ke Biner

Kekuatan	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Setara Desimal	128	64	32	16	8	4	2	1
Kode biner	0	**1**	0	**1**	**1**	0	**1**	0
Nilai Desimal	0	**64**	0	**16**	**8**	0	**2**	0

Tabel 2.9 90 Desimal ke Biner

Biner	**Desimal**	**Simbol ASCII**	**Keterangan**
1011000	88	X	Huruf besar X
1011001	89	Y	Huruf besar Y
1011010	90	Z	Huruf besar Z

Tabel 2.9 merangkum konversi nilai desimal 88, 89, dan 90 menjadi padanan binernya. Karakter ASCII yang dikenali "X," "Y," dan "Z" disimpan dan "dilihat" sebagai nilai biner oleh komputer. Untuk komputer, gabungan karakter "X", "Y", dan "Z" hanya akan "terlihat" seperti ini: 010110000101100101011010.

Sangat mudah bagi komputer untuk mengenali dan memahami apa arti semua angka 1 dan 0 itu. Namun, mustahil bagi manusia untuk menguraikan, terutama ketika melihat melalui ribuan 1 dan 0 yang semuanya dirangkai! Bagaimana Ronelle menentukan apakah Jose benar-benar berkorespondensi dengan Janice? Bagaimana Ronelle dapat memeriksa jutaan angka 1 dan 0 yang mengisi hard drive Jose, untuk mengidentifikasi kemunculan 010110000101100101011010? Alat khusus yang digunakan oleh penyelidik forensik dunia maya dapat mengubah kode biner 010110000101100101011010 menjadi format yang lebih mudah dipahami oleh manusia. Format ini disebut Heksadesimal (HEX), yang benar-benar merupakan representasi ramah manusia dari nilai-nilai biner, dan subjek Bab 3.

2.7 DESIMAL KE BINER: REKAP UNTUK TINJAUAN

Nilai desimal adalah perhitungan matematis dari biner, bukan representasi visual dari biner. Ada 10 karakter desimal unik– 0, 1, 2, 3, 4, 5, 6, 7, 8, dan 9. Sepuluh karakter unik itu sendiri tidak dapat mewakili keseluruhan rangkaian karakter ASCII atau Unicode. Mengonversi biner ke desimal, seperti yang telah dijelaskan sebelumnya, mudah jika nilai biner yang akan dikonversi kecil, tetapi seiring bertambahnya ukuran nilai biner, angkanya bisa menjadi agak besar dan membosankan. Misalnya, asumsikan nilai biner 010110000101100101011010. Nilai ini mungkin tampak menakutkan, tetapi hanya setara dengan tiga byte atau 24 bit. Jika kita mengonversi string biner ini ke nilai desimal yang setara dengan mengaktifkan nilai posisi "on" yang diwakili oleh 1s dan meninggalkan "off" nilai posisi yang diwakili oleh 0s, string angka kita akan terlihat seperti ini:

010110000101100101011010

mati + 4194304 + mati + 1048576 + 524288 + mati + mati + mati
+ mati + 16384 + mati + 4096 + 2048 + mati + mati + 256
+ mati + 64 + mati + 16 + 8 + mati + 2 + mati

Ketika akhirnya dijumlahkan, rangkaian nilai biner ini akan menghasilkan 5.787.994.Proses penguraian nilai biner menjadi padanan desimalnya bisa menjadi sangat membosankan, memakan waktu, dan sangat mahal, terutama jika string nilai biner lebih dari tiga byte. Bayangkan mengonversi seluruh kalimat, atau bagaimana dengan gambar? Bagaimana kita manusia dapat merepresentasikan biner dengan lebih baik tanpa kebosanan komputasi desimal?
Solusi: notasi heksadesimal dan representasi numerik.

2.8 RINGKASAN

Matematika mungkin satu-satunya bahasa universal, dan prinsip-prinsipnya didasarkan pada kebenaran yang melekat. Terlepas dari representasi bahasa atau karakter tertulis, 1 + 1 akan selalu sama dengan 2. Umat manusia mungkin telah menciptakan angka atau simbol yang digunakan untuk menyatakan konsep matematika; namun, manusia tidak menciptakan matematika. Umat manusia mengembangkan metode pengkodean karakter berdasarkan representasi simbolik termasuk: huruf, simbol, skrip, tanda baca, angka, piktograf, gambar gua, dan sebagainya. Bahasa tertulis terus berkembang dan berkembang; namun, sehalus ini, tidak ada yang mampu melakukan transmisi digital "langsung".

Sifat universal sejati matematika diungkapkan oleh seberapa cocoknya untuk representasi digital dari bahasa tertulis. Untuk berkomunikasi secara elektronik, atau digital, harus ada metode yang dapat mengubah skrip berbasis karakter manusia menjadi skrip matematis. Biner adalah metode pengkodean matematis dimana data dikirim secara elektronik. Manusia lebih menyukai paradigma penyandian simbol atau karakter. Oleh karena itu perlu untuk menghubungkan titik-titik dan mengubah set karakter biner ke manusia.

BAB 3
KEKUATAN HEX
MENEMUKAN KEPINGAN DATA

Ronelle dihadapkan dengan memeriksa jutaan keping data bukti potensial yang berada di hard drive Jose, mencari jarum pepatah di tumpukan jerami, kejadian berurutan dari string karakter "X," "Y," "Z." Ronelle telah menguraikan ekuivalen desimal karakter menjadi nilai biner individual, dan sekarang harus memahami proses ini pada string 1 dan 0 dalam upaya untuk mengidentifikasi string serupa 1 dan 0 pada hard drive yang disita dari kantor Jose. Kasing kami digunakan sebagai ilustrasi dasar dari skenario yang kompleks, untuk mengkomunikasikan kepada pembaca kompleksitas di balik blok bangunan penting forensik cyber. Penting bagi pembaca untuk mencatat bahwa dalam penyelidikan sebenarnya, mencari string karakter dari nama perusahaan, “XYZ,” seperti dalam contoh kasus kami, mungkin tidak ideal karena akan menghasilkan banyak positif palsu. Pembaca harus ingat bahwa contoh-contoh yang digunakan dalam buku ini digunakan untuk mengungkapkan konsep atau ide forensik, dan bukan bantuan investigasi.

Proses penguraian nilai biner menjadi padanan desimalnya bisa menjadi sangat membosankan, memakan waktu, dan sangat mahal, terutama jika rangkaian nilai binernya besar. Untungnya bagi Ronelle, string biner yang mewakili karakter "X", "Y", dan "Z" relatif kecil. Bagaimana Ronelle dapat merepresentasikan string biner ini dengan lebih baik tanpa kebosanan perhitungan desimal seperti yang telah dibahas sebelumnya? Solusinya, ubah string biner 1s dan 0s menjadi notasi ekuivalen heksadesimalnya.

3.1 APA HEX?

Hexadecimal (disingkat HEX) benar-benar merupakan representasi ramah manusia dari nilai-nilai biner. Karakter HEX sering diawali dengan 0x (nol, sub x) untuk menunjukkannya dari sistem pengkodean lain. Jadi, 0x3F memberi tahu pembaca bahwa 3F adalah HEX, dan bukan ASCII.

Komputer atau prosesor tidak menghitung dalam HEX. Perangkat lunak dapat mengubah biner menjadi HEX, tetapi mikroprosesor itu sendiri tidak melakukan perhitungan matematisnya dalam HEX. HEX adalah kode karakter berbasis 16, yang, seperti yang akan kita lihat, bekerja dengan baik untuk merepresentasikan biner.

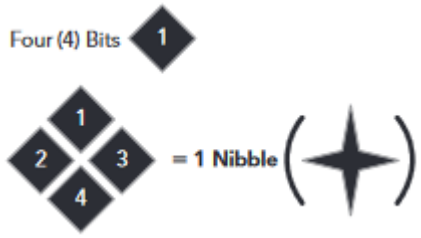
Mengapa? 16 karakter yang mewakili HEX adalah simbol 0–9, yang mewakili nilai 0 hingga 9, dan A, B, C, D, E, dan F, yang mewakili nilai 10 hingga 15. Jadi ada 16 karakter unik, yang masing-masingnya sesuai dengan pola empat bit. 16 nilai HEX beserta konversi desimal dan binernya dapat dilihat pada Tabel 3.1. Seperti yang terlihat pada Tabel 3.1, 16 karakter HEX dapat mengasumsikan semua kemungkinan nilai biner 4 bit (24 = 16). Untuk memahami bagaimana biner diubah menjadi HEX, kita perlu memeriksa konsep byte "bit" lebih dekat.

TABEL 3.1 Persamaan HEX, Biner, dan Desimal

Hex	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1 0
Biner	1111	1110	1101	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001 0000
Desimal	15	14	13	12	11	10	9	8	7	6	5	4	3	2	10

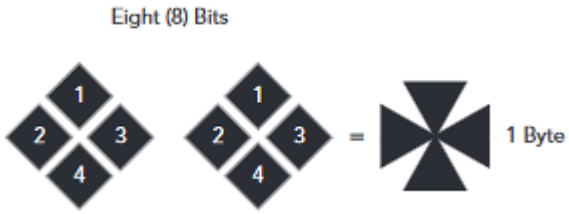
3.2 BIT DAN BYTE DAN NIBBLES

Sebuah byte dibagi (untuk pemahaman manusia, sekali lagi ini bukan sesuatu yang dilakukan prosesor saat menghitung) menjadi dua bagian yang sama yang disebut nibbles. Satu karakter HEX sesuai dengan nilai satu nibble data, 4 bit. Namun, karakter pengkodean standar membutuhkan byte penuh atau 8 bit untuk representasi. Jadi, dua nibble perlu dipasangkan untuk mencapai representasi ini.



Gambar 3.1 Empat Bit Sama Dengan Satu Nibble

Gambar 3.2 Dua Nibble Sama Dengan Satu Byte



Gambar 3.3 Delapan Bit Sama Dengan Satu Byte

Gambar 3.4 Setara Bit, Nibble, dan Byte

Jadi, kita memiliki empat (4) bit per nibble (lihat Gambar 3.1), dua (2) nibble per byte (lihat Gambar 3.2), dan delapan (8) bit per byte (lihat Gambar 3.3). Jadi pada Gambar 3.3, kita melihat bahwa delapan (8) bit sama dengan satu (1) byte. Dapat juga dikatakan, bahwa delapan bit, atau dua nibble, sama dengan satu byte (lihat Gambar 3.4).

Penting untuk diingat bahwa byte berdiri sendiri saat merepresentasikan karakter atau simbol, sedangkan nibble tidak bisa; ia membutuhkan separuh lainnya (atau dengan kata lain — karakter penyandian standar membutuhkan byte penuh atau delapan bit untuk representasi). Oleh karena itu, dua Nibble harus dipasangkan untuk mencapai representasi ini. Ingat, kode karakter HEX memiliki 16 nilai unik—A hingga F dan 0 hingga 9. Berapa banyak nilai biner (bit) yang diperlukan untuk mewakili 16 nilai unik? Jawabannya: 2^4 (2 . . . 4 . . . 8 . . 16).

Penting untuk diingat bahwa kita berurusan dengan nilai biner, jadi matematika apa pun harus berbasis dua. Jadi jumlah perhitungan "basis" adalah 2:

$$2 \times 2 \times 2 \times 2 = 16$$

2 pangkat empat.

Oleh karena itu, nilai HEX dapat persis sesuai dengan semua 16 nilai, masing-masing panjangnya empat bit.

Untuk membantu memahami hal ini lebih lanjut, mari kita tinjau kembali pertanyaan yang diajukan sebelumnya, "berapa banyak bit yang diperlukan untuk mewakili empat musim (musim dingin, musim semi, musim panas, dan musim gugur)?" Satu bit hanya memberi kita dua nilai yang mungkin, 0 atau 1. Ini akan cukup untuk mewakili keadaan lampu Nyala atau Mati tetapi tidak untuk empat musim. Ingat, kita sedang mencari nilai biner, Basis 2, sehingga setiap nilai hanya memiliki dua kemungkinan status. Dua bit sebenarnya akan memberi kita empat kemungkinan nilai yang diperlukan untuk mewakili empat musim: 00, 01, 10, dan 11, seperti yang ditunjukkan pada Tabel 3.2. Ini bisa dilakukan secara matematis juga: $2 \times 2 = 4$.

Tabel 3.2 Dua Bit Mewakili Empat Kemungkinan Nilai

Bit 1	Bit 2
0	0
0	1
1	0
1	1

Oleh karena itu, nilai HEX dapat sama persis dengan 16 kemungkinan nilai empat-bit.

16 karakter HEX (A–F, 0–9) dapat diwakili oleh empat nilai biner (misalnya, 1101, 0110, 0010) per karakter, sehingga karakter HEX "A", misalnya, diwakili oleh nilai biner 1010. (Lihat Tabel 3.3.)

Tabel 3.3 Karakter HEX dan Setaraan Biner

Hex	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Biner	1111	1110	1101	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001	0000
Desimal	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Ingat, empat bit sama dengan satu Byte; oleh karena itu, satu karakter HEX mewakili satu Byte.

Kesimpulan:

1 bit = nilai biner dari 0 atau 1
4 bit = Byte = satu karakter HEX
2 Nibble = 1 byte
8 bit = 1 byte
1 byte = 2 karakter HEX

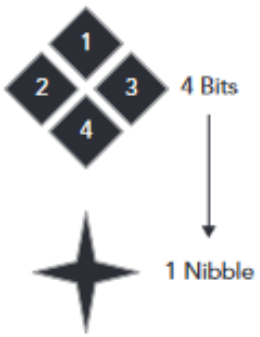
Jadi, untuk menjawab pertanyaan yang mengawali diskusi ini, mengapa menggunakan HEX untuk merepresentasikan biner? Karena satu byte dan 2 karakter HEX mewakili delapan bit, dengan total 256 nilai potensial.

3.3 NIBBLES DAN BIT

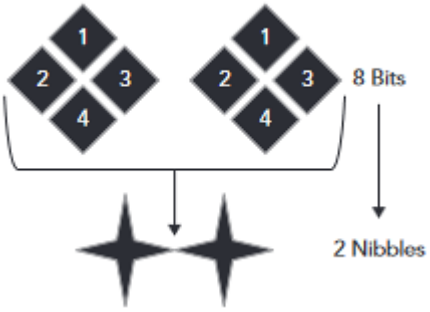
Untuk membedakan antara dua bit, yang satu disebut sebagai bit kiri dan yang lainnya sebagai bit kanan.

HEX, seperti yang telah kami tunjukkan, menggunakan basis 16. Kita dapat membagi sebuah byte menjadi dua kelompok yang masing-masing terdiri dari empat bit (mis., nnnn nnnn). Setiap kelompok individu kemudian disebut sebagai Nibbles.

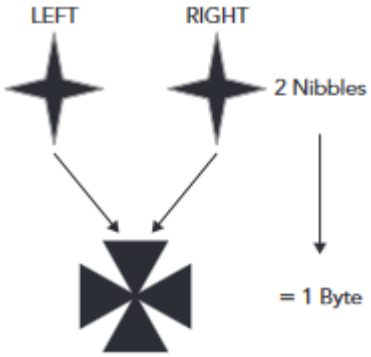
Bit kanan dengan semua bitnya dimatikan, atau 0000, sama dengan nilai 0; dengan semua bitnya dihidupkan, 1111, nibble kanan memiliki nilai 15 (8 + 4 + 2 + 1). Mari meringkas (lihat Gambar 3.5, 3.6, dan 3.7). Ingat c Nibbles bekerja berpasangan, kiri dan kanan. Lihat Tabel 3.4.



Gambar 3.5 Empat Bit Sama Dengan Satu Nibble



Gambar 3.6 Delapan Bit Sama Dengan Dua Nibbles



Gambar 3.7 Dua Nibbles Sama Dengan Satu Byte

Tabel 3.4 Byte Kiri dan Kanan

	Byte Kiri					**Byte Benar**			
Kekuatan	2^7	2^6	2^5	2^4		2^3	2^2	2^1	2^0
Nilai Desimal	128	64	32	16		8	4	2	1
Nilai Biner	0	1	0	1		1	0	0	0
Nilai Desimal	0	64	0	16		8	0	0	0

0+64+0+16+8+0+0+0=88 88 = 0101 1000
KIRI KANAN

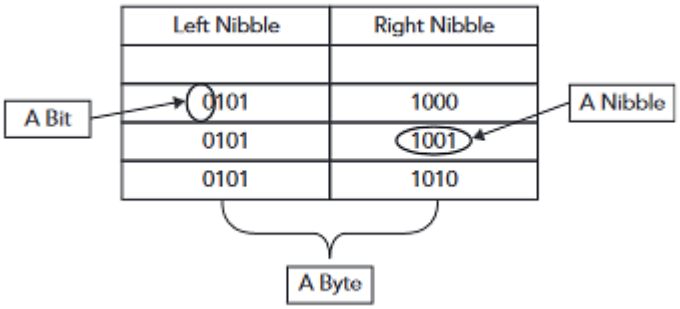
3.4 KONVERSI BINARI KE HEX

Mari bandingkan konversi desimal dengan konversi ke HEX, menggunakan string biner Ronelle yang relatif kecil yang mewakili karakter "X", "Y", dan "Z". Nilai biner dari karakter “X”, “Y”, dan “Z” sama dengan 010110000101100101011010.

Jika kita membagi nilai-nilai ini menjadi potongan-potongan empat-bit yang mewakili Nibbles, dan memasangkannya untuk membentuk sebuah byte, hasilnya ditunjukkan pada Tabel 3.5. Gambar 3.8 menunjukkan dekomposisi sebuah byte, dimulai dengan nibble, kanan dan kiri, dan masing-masing bit. Kita membutuhkan empat (4) bit untuk membuat satu byte, dan dua (2) bit untuk membuat satu byte.

Tabel 3.5 Nibbles Kiri dan Kanan untuk Nilai Biner 010110000101100101011010

Byte Kiri	Byte Kanan
0101	1000
0101	1001
0101	1010



Gambar 3.8 Dasar Data

Tabel 3.6 menunjukkan betapa mudahnya kita mengubah biner menjadi desimal. Tabel 3.7 menunjukkan konversi nilai biner menjadi nilai HEX yang setara. Menggabungkan Tabel 3.6 dan 3.7 menghasilkan hasil yang ditunjukkan pada Tabel 3.8.

Tabel 3.6 Biner ke Desimal

Biner	1111	1110	1101	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001	0000
Desimal	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Tabel 3.7 Biner ke HEX

Hex	**F**	**E**	**D**	**C**	**B**	**A**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Biner	1111	1110	1101	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001	0000

Tabel 3.8 HEX, Biner, dan Setara Desimal

Hex	**F**	**E**	**D**	**C**	**B**	**A**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Biner	1111	1110	1101	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001	0000
Desimal	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Tabel 3.9 ASCII "X" Dikonversi ke Setara Biner 01011000

Byte Kiri					**Byte Kanan**				
Kekuatan	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
Nilai Desimal	128	64	32	16	8	4	2	1	
Kode biner	0	1	0	1	1	0	0	0	
Nilai Desimal	0	64	0	16	8	0	0	0	0 + 64 + 0 + 16 + 8 + 0 + 0 + 0 = 88

Untuk terus menggunakan string biner Ronelle yang mewakili karakter "X," "Y," dan "Z" sebagai bagian dari strategi pencariannya untuk menentukan kesalahan Jose, pertama-

tama Ronelle mengonversi nilai ASCII ke ekivalen binernya seperti yang ditunjukkan pada Tabel 3.9, 3.10, dan 3.11.

Tabel 3.10 ASCII "Y" Dikonversi ke Setaraan Biner 01011001

	Byte Kiri				**Byte Kanan**				
Kekuatan	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
Nilai Desimal	128	64	32	16	8	4	2	1	
Kode biner	0	1	0	1	1	0	0	1	
Nilai Desimal	0	64	0	16	8	0	0	1	0 + 64 + 0 + 16 + 8 + 0 + 0 + 1 = 89

Tabel 3.11 ASCII "Z" Dikonversi ke Setaraan Biner 01011010

	Byte Kiri				**Byte Kanan**				
Kekuatan	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
Nilai Desimal	128	64	32	16	8	4	2	1	
Kode biner	0	1	0	1	1	0	1	0	
Nilai Desimal	0	64	0	16	8	0	2	0	0 + 64 + 0 + 16 + 8 + 0 + 2 + 0 = 90

Ringkasnya, data Ronelle ditunjukkan pada Tabel 3.12.

Tabel 3.12 Konversi Desimal ke Biner

ASCII	**Nilai Desimal**	**Nilai Biner**
X	88	0101 1000
Y	89	0101 1001
Z	90	0101 1010

Selanjutnya, Ronelle mengonversi nilai biner menjadi persamaan HEX-nya, seperti yang ditunjukkan pada Tabel 3.13, 3.14, dan 3.15.

Tabel 3.13 Biner 01011000 Dikonversi ke HEX

Hex	**F**	**E**	**D**	**C**	**B**	**A**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Biner	1111	1110	1101	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001	0000

Tabel 3.14 Biner 01011001 Dikonversi ke HEX

Hex	**F**	**E**	**D**	**C**	**B**	**A**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Biner	1111	1110	1101	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001	0000

Tabel 3.15 Biner 01011010 Dikonversi ke HEX

Hex	**F**	**E**	**D**	**C**	**B**	**A**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Biner	1111	1110	1101	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001	0000

Tabel 3.16 merangkum upaya konversi Ronelle untuk memberinya nilai HEX dari karakter ASCII yang harus dicarinya di hard drive Jose. Jadi, Ronelle harus mencari string HEX 58595A, yang mewakili karakter ASCII "X", "Y", dan "Z".

Tabel 3.16 Setara HEX dari Karakter ASCII "X," "Y," dan "Z"

ASCII	Nilai Desimal	Nilai Biner	Setara HEX
X	88	0101 1000	58
Y	89	0101 1001	59
Z	90	0101 1010	5A

Dengan menggunakan editor HEX, Ronelle mencari representasi nilai HEX 58595A yang sangat dapat dibedakan untuk mengungkap adanya referensi ke Perusahaan XYZ di hard drive Jose.

3.5 EDITOR BINER (HEX).

Editor HEX adalah program yang memungkinkan Anda untuk melihat dan atau mengedit program yang dikompilasi dan file data biner. Editor ini disebut editor HEX karena paling sering menampilkan data dalam format heksadesimal. Heksadesimal digunakan karena lebih mudah bagi kebanyakan manusia daripada bekerja dalam biner. Selain itu, heksadesimal sering berguna karena komputer cenderung bekerja dengan informasi delapan-bit byte dan karena ASCII adalah kode 8-bit.

Anda tidak dapat melihat semua byte yang disimpan dalam file menggunakan aplikasi biasa untuk membuka file, dan tidak ada aplikasi yang tersedia untuk melihat item yang dihapus. Terkadang, sebagian file hilang, termasuk bagian yang berisi kode yang dapat dijalankan yang meluncurkan aplikasi yang diperlukan untuk membukanya terlebih dahulu. Penyelidik forensik dunia maya memerlukan editor biner/HEX untuk menganalisis struktur file. Melihat HEX memungkinkan Ronelle melampaui aplikasi atau file, dan itu akan memungkinkan untuk melihat semua data yang terkandung dalam file, termasuk sisa-sisa file lama atau bahkan file yang dihapus, yang mungkin ada di hard drive Jose. Gambar 3.9 adalah contoh representatif dari tampilan keluaran editor HEX.

Editor HEX terdiri dari empat area tampilan yang berbeda, atau panel, masing-masing memiliki signifikansinya sendiri dan menyampaikan informasi spesifik relatif terhadap editor itu sendiri, serta memberikan informasi penting kepada penyelidik.

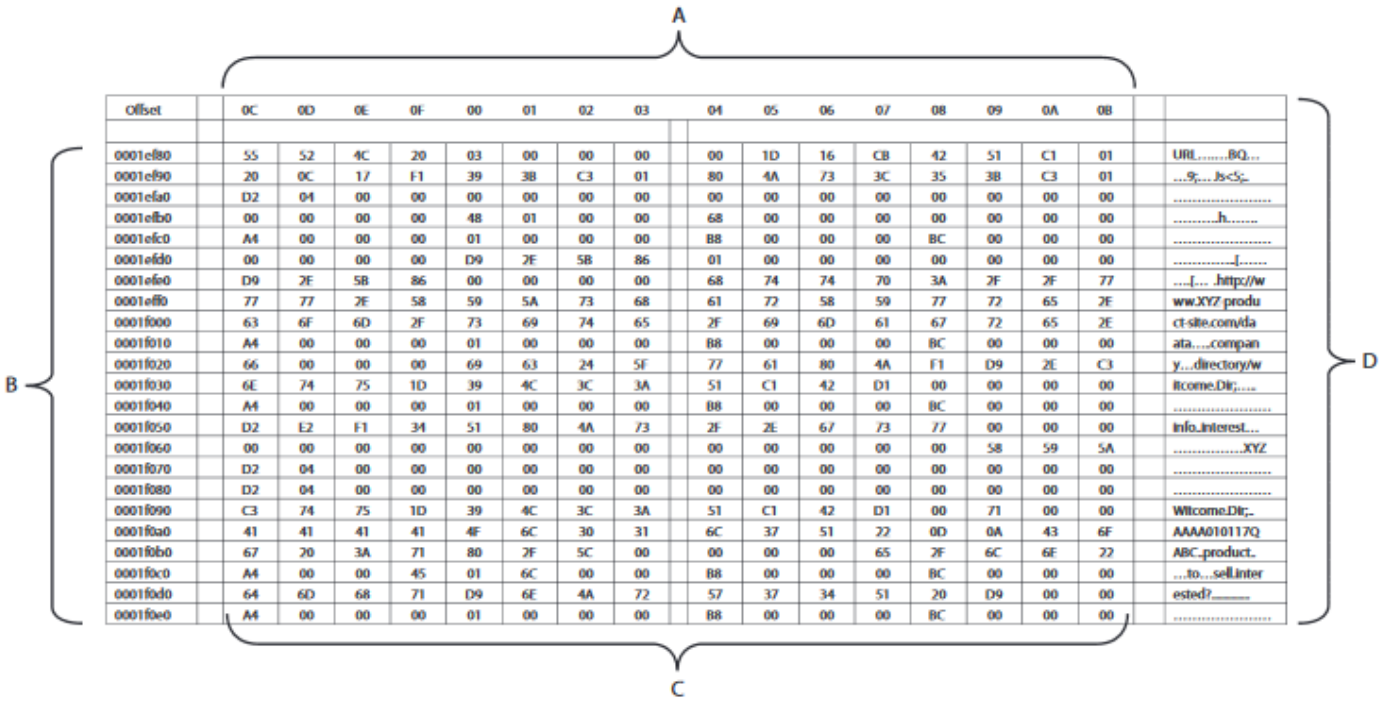
Empat area tampilan yang berbeda adalah:

1. Panel Tajuk (A)
2. Panel Alamat (B)
3. Panel data HEX (C)
4. Panel Karakter (D)

Keempat area panel editor HEX ini diidentifikasi pada Gambar 3.10.

Offset		0C	0D	0E	0F	00	01	02	03		04	05	06	07	08	09	0A	0B		
0001ef80		55	52	4C	20	03	00	00	00		00	1D	16	CB	42	51	C1	01		URL.....BQ...
0001ef90		20	0C	17	F1	39	3B	C3	01		80	4A	73	3C	35	3B	C3	01		...9;... Js<5;..
0001efa0		D2	04	00	00	00	00	00	00		00	00	00	00	00	00	00	00		.....
0001efb0		00	00	00	00	48	01	00	00		68	00	00	00	00	00	00	00		.....h.....
0001efc0		A4	00	00	00	01	00	00	00		B8	00	00	00	BC	00	00	00		.....
0001efd0		00	00	00	00	D9	2E	5B	86		01	00	00	00	00	00	00	00		.....[.....
0001efe0		D9	2E	5B	86	00	00	00	00		68	74	74	70	3A	2F	2F	77		....[.... .http://w
0001eff0		77	77	2E	58	59	5A	73	68		61	72	58	59	77	72	65	2E		ww.XYZ.produ
0001f000		63	6F	6D	2F	73	69	74	65		2F	69	6D	61	67	72	65	2E		ct-site.com/da
0001f010		A4	00	00	00	01	00	00	00		B8	00	00	00	BC	00	00	00		ata.....compan
0001f020		66	00	00	00	69	63	24	5F		77	61	80	4A	F1	D9	2E	C3		y...directory/w
0001f030		6E	74	75	1D	39	4C	3C	3A		51	C1	42	D1	00	00	00	00		itcome.Dir;.....
0001f040		A4	00	00	00	01	00	00	00		B8	00	00	00	BC	00	00	00		.....
0001f050		D2	E2	F1	34	51	80	4A	73		2F	2E	67	73	77	00	00	00		info..interest...
0001f060		00	00	00	00	00	00	00	00		00	00	00	00	00	58	59	5A		.....XYZ
0001f070		D2	04	00	00	00	00	00	00		00	00	00	00	00	00	00	00		.....
0001f080		D2	04	00	00	00	00	00	00		00	00	00	00	00	00	00	00		.....
0001f090		C3	74	75	1D	39	4C	3C	3A		51	C1	42	D1	00	71	00	00		Witcome.Dir;..
0001f0a0		41	41	41	41	4F	6C	30	31		6C	37	51	22	0D	0A	43	6F		AAAA010117Q
0001f0b0		67	20	3A	71	80	2F	5C	00		00	00	00	65	2F	6C	6E	22		ABC..product..
0001f0c0		A4	00	00	45	01	6C	00	00		B8	00	00	00	BC	00	00	00		...to...sell.inter
0001f0d0		64	6D	68	71	D9	6E	4A	72		57	37	34	51	20	D9	00	00		ested?.....
0001f0e0		A4	00	00	00	01	00	00	00		B8	00	00	00	BC	00	00	00		.....

Gambar 3.9 Editor HEX (tampilan sampel)



Gambar 3.10 Area Empat Panel Editor HEX

Panel Tajuk (A)

Panel header menampilkan header dari salah satu dari tiga panel lainnya (lihat Gambar 3.11). Diantaranya, panel data HEX (C) menggunakan offset byte relatif terhadap awal baris sebagai header.

Offset		0C	0D	0E	0F	00	01	02	03		04	05	06	07	08	09	0A	0B		

Gambar 3.11 Panel Tajuk

Panel Alamat (B)

Setiap byte dalam file diberi nomor, disebut alamatnya, mulai dari 0 untuk byte pertama file, 1 untuk byte kedua, dan seterusnya. Lihat Gambar 3.12. Panel alamat menampilkan alamat byte di awal baris.

Offset	
0001ef80	
0001ef90	
0001efa0	
0001efb0	
0001efc0	
0001efd0	
0001efe0	
0001eff0	
0001f000	
0001f010	
0001f020	
0001f030	
0001f040	
0001f050	
0001f060	
0001f070	
0001f080	
0001f090	
0001f0a0	
0001f0b0	
0001f0c0	
0001f0d0	
0001f0e0	

Gambar 3.12 Panel Alamat

Panel HEX-Data (C)

Area heksadesimal tengah adalah area editor HEX yang paling umum digunakan. Ini mencantumkan setiap byte file dalam sebuah tabel, biasanya 16 byte per baris. Pada Gambar 3.13, delapan byte pertama dari file adalah "55-52-4C-20-03-00 00 00." Panel HEX-data mencantumkan nilai heksadesimal dari setiap byte file; standarnya adalah 24 byte per baris. Ia bekerja dalam mode pengeditan HEX, dan menampilkan nilai heksadesimal setiap byte sebagai bidang dua karakter.

Panel Karakter (D)

Gambar 3.14 menunjukkan detail panel karakter. Panel karakter menampilkan nilai ASCII dari setiap byte file. Ini hanya memberikan tampilan karakter yang dapat dicetak; karakter yang tidak dapat dicetak ditampilkan sebagai karakter titik (".") atau dengan karakter lain yang ditentukan pengguna. Ronelle sekarang akan mencari editor HEX untuk mencari kemunculan nilai HEX 58595A, karena dia tahu bahwa nilai ini mewakili karakter "X, Y, dan Z."

	55	52	4C	20	03	00	00	00		00	1D	16	CB	42	51	C1	01
	20	0C	17	F1	39	3B	C3	01		80	4A	73	3C	35	3B	C3	01
	D2	04	00	00	00	00	00	00		00	00	00	00	00	00	00	00
	00	00	00	00	48	01	00	00		68	00	00	00	00	00	00	00
	A4	00	00	00	01	00	00	00		B8	00	00	00	BC	00	00	00
	00	00	00	00	D9	2E	5B	86		01	00	00	00	00	00	00	00
	D9	2E	5B	86	00	00	00	00		68	74	74	70	3A	2F	2F	77
	77	77	2E	58	59	5A	73	68		61	72	58	59	77	72	65	2E
	63	6F	6D	2F	73	69	74	65		2F	69	6D	61	67	72	65	2E
	A4	00	00	00	01	00	00	00		B8	00	00	00	BC	00	00	00
	66	00	00	00	69	63	24	5F		77	61	80	4A	F1	D9	2E	C3
	6E	74	75	1D	39	4C	3C	3A		51	C1	42	D1	00	00	00	00
	A4	00	00	00	01	00	00	00		B8	00	00	00	BC	00	00	00
	D2	E2	F1	34	51	80	4A	73		2F	2E	67	73	77	00	00	00
	00	00	00	00	00	00	00	00		00	00	00	00	00	58	59	5A
	D2	04	00	00	00	00	00	00		00	00	00	00	00	00	00	00
	D2	04	00	00	00	00	00	00		00	00	00	00	00	00	00	00
	C3	74	75	1D	39	4C	3C	3A		51	C1	42	D1	00	71	00	00
	41	41	41	41	4F	6C	30	31		6C	37	51	22	0D	0A	43	6F
	67	20	3A	71	80	2F	5C	00		00	00	00	65	2F	6C	6E	22
	A4	00	00	45	01	6C	00	00		B8	00	00	00	BC	00	00	00
	64	6D	68	71	D9	6E	4A	72		57	37	34	51	20	D9	00	00
	A4	00	00	00	01	00	00	00		B8	00	00	00	BC	00	00	00

Gambar 3.13 Panel HEX-Data

URL.....BQ...
...9;... Js<5;..
.....
.....h.....
.....
.....[.....
....[... .http://w
ww.XYZ-produ
ct-site.com/da
ata.....compan
y...directory/w
itcome.Dir;.....
.....
info..interest...
.....XYZ
.....
.....
Witcome.Dir;..
AAAA010117Q
ABC..product..
...to...sell.inter
ested?.....
.....

Gambar 3.14 Panel Karakter

Jika Ronelle menemukan urutan HEX 58595A di dalam data yang diambil dari hard drive Jose, kemungkinan besar ini adalah bukti yang lebih kuat, yang selanjutnya dapat melibatkan Jose dalam potensi pencurian kekayaan intelektual dan keterlibatan antara Jose dan Perusahaan XYZ.

Nilai HEX 58595A

Pada Gambar 3.15, kita melihat contoh representatif tentang bagaimana nilai HEX 58595A mungkin tampak bagi Ronelle saat dia memeriksa panel data editor HEX.

0001ef80	55	52	4C	20	03	00	00	00	00	1D	16	CB	42	51	C1	01	URL.....BQ...
0001ef90	20	0C	17	F1	39	3B	C3	01	80	4A	73	3C	35	3B	C3	01	...9;... Js<5;..
0001efa0	D2	04	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0001efb0	00	00	00	00	48	01	00	00	68	00	00	00	00	00	00	00	.....h.....
0001efc0	A4	00	00	00	01	00	00	00	B8	00	00	00	BC	00	00	00	.....
0001efd0	00	00	00	00	D9	2E	5B	86	01	00	00	00	00	00	00	00	.....[.....
0001efe0	D9	2E	5B	86	00	00	00	00	68	74	74	70	3A	2F	2F	77	....[... .http://w
0001eff0	77	77	2E	58	59	5A	73	68	61	72	58	59	77	72	65	2E	ww.XYZ-produ
0001f000	63	6F	6D	2F	73	69	74	65	2F	69	6D	61	67	72	65	2E	ct-site.com/da
0001f010	A4	00	00	00	01	00	00	00	B8	00	00	00	BC	00	00	00	ata.....compan
0001f020	66	00	00	00	69	63	24	5F	77	61	80	4A	F1	D9	2E	C3	y...directory/w
0001f030	6E	74	75	1D	39	4C	3C	3A	51	C1	42	D1	00	00	00	00	itcome.Dir;.....
0001f040	A4	00	00	00	01	00	00	00	B8	00	00	00	BC	00	00	00	.....
0001f050	D2	E2	F1	34	51	80	4A	73	2F	2E	67	73	77	00	00	00	info..interest...
0001f060	00	00	00	00	00	00	00	00	00	00	00	00	00	58	59	5A	.....XYZ
0001f070	D2	04	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0001f080	D2	04	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0001f090	C3	74	75	1D	39	4C	3C	3A	51	C1	42	D1	00	71	00	00	Witcome.Dir;..
0001f0a0	41	41	41	41	4F	6C	30	31	6C	37	51	22	0D	0A	43	6F	AAAA010117Q
0001f0b0	67	20	3A	71	80	2F	5C	00	00	00	00	65	2F	6C	6E	22	ABC..product..
0001f0c0	A4	00	00	45	01	6C	00	00	B8	00	00	00	BC	00	00	00	...to...sell.inter
0001f0d0	64	6D	68	71	D9	6E	4A	72	57	37	34	51	20	D9	00	00	ested?.....
0001f0e0	A4	00	00	00	01	00	00	00	B8	00	00	00	BC	00	00	00	.....

Gambar 3.15 Nilai HEX 58595A seperti yang Muncul di Editor HEX

3.6 JARUM DI DALAM TUPUKAN JERAMI

Mengapa manusia membutuhkan cara yang ramah untuk melihat biner? Kebanyakan manusia akan memiliki waktu yang sangat sulit untuk mencoba memahami string panjang 1 dan 0, ujung ke ujung, untuk lebih dari, katakanlah, delapan posisi. Cepat, ini isinya apa: 01101000011001010110110001101100011011111?

Penyelidik forensik dunia maya HARUS memiliki pemahaman menyeluruh tentang proses di mana bit data mentah diubah menjadi informasi, dan bagaimana perangkat silikon kompleks menafsirkan pulsa energi, menetapkan pulsa ini nilai yang dapat dihitung dari satu dan nol, dan kemudian menjadi nilai numerik urutan yang lebih tinggi. - ues, dan akhirnya menjadi karakter yang dipahami oleh penangan berbasis karbon mereka. Ini memberi Ronelle dan penyelidik forensik dunia maya sarana untuk menemukan potongan data yang mereka cari, di tengah potensi miliaran byte data asing.

3.7 RINGKASAN

Dokumen atau file memiliki apa yang terkadang disebut header atau data tambahan, ditempatkan di awal blok data. Dalam transmisi data, data yang mengikuti header disebut body. Header yang berlaku mengikat tubuh ke perangkat lunak yang diperlukan untuk membukanya atau mengaksesnya.

Misalnya, dokumen yang dibuat dengan Microsoft (MS) Word mungkin mengalami kesulitan untuk dibuka menggunakan/aplikasi Adobe Acrobat Reader. Ini karena ada kode yang disematkan di dalam dokumen apa pun yang dibuat menggunakan MS Word, yang memberi tahu sistem operasi bahwa MS Word (atau perangkat lunak lain yang kompatibel) diperlukan untuk membuka dokumen tersebut.

Jika kode yang mengikat dokumen ke perangkat lunak asli entah bagaimana ditimpa atau "dihapus", perangkat lunak tidak akan dapat menyusun kembali dokumen ke dalam format aslinya atau ke dalam format yang dapat dibaca oleh pengguna, sehingga menyebabkan dokumen menjadi tidak dapat diakses dan dibaca oleh pengguna.

Beberapa dari data ini, seperti teks yang memberatkan (kemunculan "XYZ" misalnya), mungkin masih berada di dokumen, di disk, di dalam hard drive. Ronelle tidak ingin melewatkan bukti yang berpotensi memberatkan ini dan membutuhkan cara yang dapat diverifikasi untuk menyisihkan jutaan byte data di hard drive Jose. Menemukan bukti semacam itu dapat mendukung atau menyangkal klaim yang diajukan oleh ABC Inc. terhadap karyawannya Jose, dalam pencurian kekayaan intelektual ABC.

Untuk penyelidik forensik dunia maya untuk mencari kata kunci yang terkandung dalam data yang disita dari seluruh hard drive (atau bahkan dari data yang dipersempit ke folder tertentu atau gambar tertentu di dalam hard drive pengguna), sebaiknya gunakan HEX untuk membantu menyelesaikan tugas yang sangat besar ini.

Melihat HEX memungkinkan penyelidik untuk melampaui aplikasi atau file. Ini memungkinkan untuk melihat semua data yang terkandung dalam file termasuk sisa-sisa file lama atau bahkan dihapus. Dengan pengenalan dasar-dasar "matematika komputer" ini, kami siap untuk meningkatkannya satu level di Bab 4 untuk bekerja di dunia HEX, sahabat penyelidik forensik dunia maya.

BAB 4
FILE

Saat Kita Bergerak Lebih Jauh, saat kita menyelami lebih dalam untuk mengeksplorasi dan mendapatkan pemahaman tentang sains di balik forensik dunia maya, tujuan kami adalah menyediakan materi yang dapat digunakan untuk pembaca, materi yang akan menopang perjalanan evolusi teknologi, materi yang tidak akan ketinggalan zaman atau usang sebelum diterbitkan.

Untuk mencapai tujuan kami, perlu untuk mengeksplorasi konsep umum atau luas (walaupun mungkin rumit) sambil menahan diri dari menangani perangkat lunak, program, atau bahkan alat forensik umum tertentu, yang dapat dengan cepat menjadi kuno dan usang seiring waktu.

Saat kami memeriksa lebih lanjut blok bangunan forensik dunia maya, perhatian khusus telah dibuat untuk fokus pada alat yang tidak akan cepat ketinggalan zaman, kedaluwarsa, atau tidak lagi didukung oleh vendor. Kami menghabiskan lebih banyak waktu, misalnya, membahas alat seperti editor HEX, dibandingkan membahas sistem operasi Windows NT. Editor HEX juga telah ada sejak lama—sejak HEX itu sendiri—sedangkan Windows NT dengan cepat menjadi semakin tidak relevan.

Baca terus selagi kami melanjutkan eksplorasi kami tentang ilmu di balik forensik dunia maya, dengan fokus di sini pada file, tanda tangan file, dan peran serta relevansinya dalam investigasi forensik dunia maya.

4.1 PENGANTAR

Dalam Bab 3 topik-topik berikut dibahas:

1. Membahas HEX dan langkah-langkah yang terkait dengan konversi representasi biner ini ke ASCII.
2. Mencakup proses konversi aktual dalam upaya untuk lebih memahami representasi karakter HEX.
3. Membahas detail tentang mur dan baut HEX.
4. Merujuk ke editor HEX dan fungsinya, tetapi berhenti menyelidiki secara mendalam fungsi dan kegunaan editor HEX saat melihat file (atau potongan file). Diskusi ini disimpan untuk bab selanjutnya.

HEX, seperti yang telah dibahas, berguna saat mencoba melihat file yang terhapus sebagian. Ini menimbulkan pertanyaan:

1. Mengapa file yang terhapus sebagian sulit dibuka atau dilihat secara normal?
2. Bagian mana dari file yang dapat dilihat oleh editor HEX, yang jika tidak, tidak akan terlihat?

4.2 FILE, STRUKTUR FILE, DAN FORMAT FILE

Untuk menjawab pertanyaan yang diajukan di atas, kita perlu menyelidiki lebih lanjut dasar-dasar file, struktur file, dan format file. File yang terhapus sebagian dalam banyak kasus mungkin kehilangan sebagian dari data pemformatannya, data yang mengidentifikasi file tersebut.

Informasi pemformatan inilah yang mengidentifikasi file ke perangkat lunak induk atau aslinya. Jika file tidak berisi informasi pemformatan ini, perangkat lunak atau sistem operasi (OS) kemungkinan besar tidak akan dapat mengakses atau mengeksekusi file tersebut. Informasi pemformatan inilah yang secara unik mengidentifikasi file. Ada ratusan format data yang berbeda (database, pengolah kata, spreadsheet, gambar, video, dll.). Ada juga format untuk program yang dapat dieksekusi pada platform yang berbeda (Windows, Mac, Linux, Unix, dll.). Setiap format menentukan bagaimana urutan bit dan byte ditata, dengan file teks berbasis ASCII menjadi salah satu format paling sederhana untuk diuraikan oleh manusia.

Beberapa format file dirancang untuk menyimpan jenis data yang sangat khusus: format JPEG, misalnya, dirancang hanya untuk menyimpan gambar fotografi statis. Format file lain, bagaimanapun, dirancang untuk penyimpanan beberapa jenis data yang berbeda: format GIF mendukung penyimpanan gambar diam dan animasi sederhana, dan format QuickTime dapat bertindak sebagai wadah untuk berbagai jenis multimedia.

File teks hanyalah file yang menyimpan teks apa pun, dalam format seperti ASCII atau UTF-8, dengan sedikit karakter kontrol jika ada. Beberapa format file, seperti HTML, atau kode sumber dari beberapa bahasa pemrograman tertentu, sebenarnya juga merupakan file teks, tetapi mematuhi aturan yang lebih spesifik yang memungkinkannya digunakan untuk tujuan tertentu. Ada berbagai macam file digital jenis di alam semesta elektronik kita yang terus berkembang. Berbagai jenis file ini berisi informasi pemformatan khusus yang memungkinkan akses, penyimpanan, atau "manipulasi" file. “Manipulasi” ini dapat terjadi melalui sistem operasi itu sendiri, atau dapat terjadi melalui program “induk” yang diinstal pada sistem operasi.

Program induk, artinya program dan mungkin perangkat lunak berpemilik, digunakan untuk membuat, menjalankan, atau mengakses file. Dalam banyak kasus, file akan berisi data, tanda tangan file, yang darinya perangkat lunak induknya (atau sistem operasi) akan dapat mengidentifikasi dan menangani operasinya. Informasi tanda tangan file ini terkandung dalam apa yang terkadang disebut sebagai header file. Data yang terkandung dalam header file tidak terlihat oleh pengguna biasa, namun sangat penting agar file berfungsi seperti yang dirancang. Data inilah yang terkandung dalam header file yang digunakan untuk mengidentifikasi format file. Header file juga dapat berisi data mengenai integritas file serta informasi tentang file itu sendiri dan isinya. Data ini sering disebut sebagai Metadata. Tidak ada satu struktur format file khusus yang cocok untuk semua jenis file. Format file akan bervariasi seperti halnya konten file. Isi gambar, serta formatnya, misalnya, akan berbeda dengan isi dan format dokumen pengolah kata.

4.3 EKSTENSI FILE

Dalam lingkungan Sistem Operasi Windows, format file mudah diidentifikasi dengan ekstensi file. Sistem Operasi Windows menggunakan ekstensi file untuk "mengikat" aplikasi ke jenis file tertentu. Misalnya, Windows akan mengikat perangkat lunak Adobe Reader ke ekstensi file .PDF, atau MS Word ke ekstensi file .DOC (atau .DOCX). Ekstensi file khusus untuk Sistem Operasi Windows dan tanpa ekstensi, Sistem Operasi Windows tidak akan tahu cara membuka, memproses, atau menangani file. Sistem Operasi Windows melihat ekstensi saat mengikat file ke aplikasi.

Pertanyaan: Apa yang akan terjadi jika ekstensi file dari file yang dapat dieksekusi (.EXE) diubah menjadi ekstensi file Adobe (.PDF)?
Jawaban: Windows akan melihat ekstensi file dan melihat bahwa itu adalah .PDF; karena itu akan menyerahkan file itu ke Adobe untuk dibuka. Adobe akan mencoba meluncurkan atau membuka file dan melaporkan kesalahan karena file tersebut, apa pun namanya, sebenarnya bukan file Adobe.

Windows menyimpan informasi pengikatan aplikasi ini di bagian Operating System (OS) disebut registry. Setiap jenis file berisi ekstensi file yang sesuai; korelasi yang disimpan dalam registri ini memberi tahu OS jenis program apa yang diperlukan untuk mengakses jenis file tertentu. Ini adalah cara Window mengatur berbagai jenis file ke perangkat lunak yang sesuai. Ketika OS mengidentifikasi ekstensi, katakanlah .CSV (Comma Separated Values), OS melihat ke registri dan menemukan aplikasi mana yang terikat dengan ekstensi ini. Dalam kebanyakan kasus, MS Excel terikat ke CSV, jadi Windows akan menyerahkan file itu ke Excel untuk dibuka dan diproses. Ekstensi file dan/atau informasi registri terkait dapat dimanipulasi oleh pengguna yang cerdas. Misalnya, perubahan dibuat pada registri sehingga ekstensi file .CSV dikaitkan dan dibuka dengan penampil gambar seperti Windows Picture Viewer. Jika pengguna mengklik file CSV yang sebenarnya, Windows akan menyerahkan file itu ke Windows Viewer alih-alih aplikasi logis (mis., Excel); penampil gambar kemudian akan mencoba membuka file. Jika file tersebut adalah file CSV yang sebenarnya, pesan "pratinjau tidak tersedia" atau kesalahan akan ditampilkan, karena aplikasi Windows Viewer tidak akan dapat memproses file dengan ekstensi format file .CSV.

Katakanlah file itu adalah gambar, yang telah diganti namanya dengan ekstensi .CSV. Windows akan menyerahkan file CSV itu ke perangkat lunak penampil gambar dan gambar akan ditampilkan. File dengan ekstensi file yang salah akan terbuka selama Registri Windows memiliki ekstensi file yang "salah" yang terkait dengan perangkat lunak yang benar. Ingat, mengubah atau mengganti nama ekstensi file tidak mengubah konten file; itu hanya mengubah cara OS Windows menangani file (yaitu, ke aplikasi mana file tersebut dikirim).

Jadi mengapa cara OS menangani interpretasi ekstensi file penting bagi penyelidik forensik dunia maya? Bagaimana jika penyelidik forensik dunia maya menerima gambar forensik dari hard drive tersangka penganiaya anak, dan mencari konten drive untuk file gambar (mis., JPG). Katakanlah penyelidik tidak dapat menemukan keberadaan file dengan

ekstensi gambar seperti .JPG. Apakah kasus ini ditutup? Apakah tersangka penganiaya anak tidak bersalah dan bebas untuk pergi?

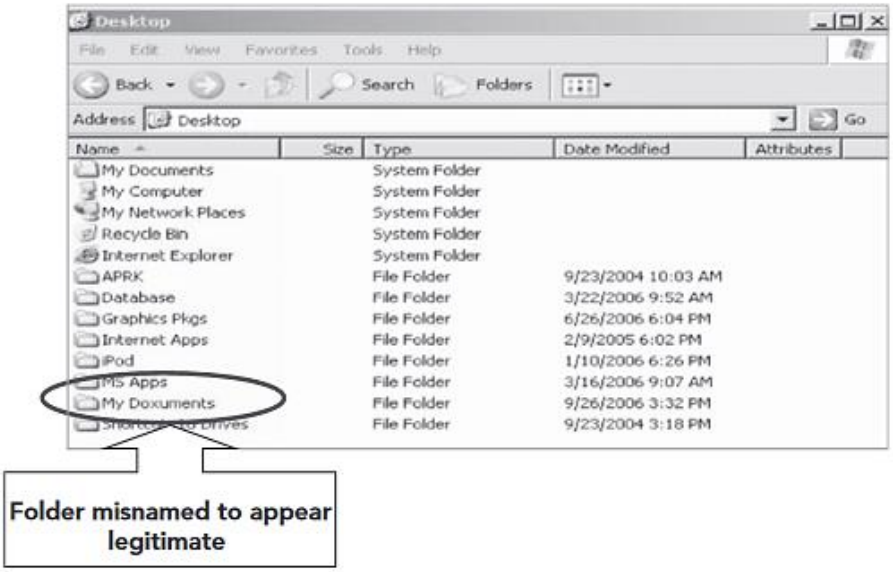
Hampir tidak; mungkin ada banyak gambar di hard drive ini yang baru saja diganti namanya. Fakta bahwa Windows menggunakan ekstensi file menciptakan sarana yang dapat digunakan pengguna untuk menyembunyikan informasi dengan mengganti nama ekstensi file.

4.4 MENGUBAH EKSTENSI FILE UNTUK MENGHINDARI DETEKSI

Proses untuk mengubah ekstensi file agar tidak terdeteksi cukup sederhana, seperti yang ditunjukkan pada langkah-langkah berikut.

Langkah 1

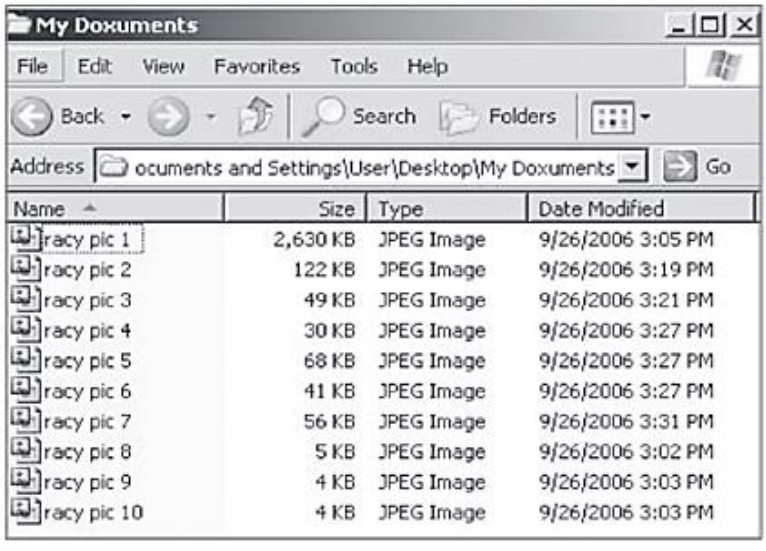
Buatlah folder yang tampak sah di mana Anda ingin menempatkan file Anda (lihat Gambar 4.1).



Gambar 4.1 Membuat Folder yang Salah Nama tetapi Terlihat Sah

Langkah 2

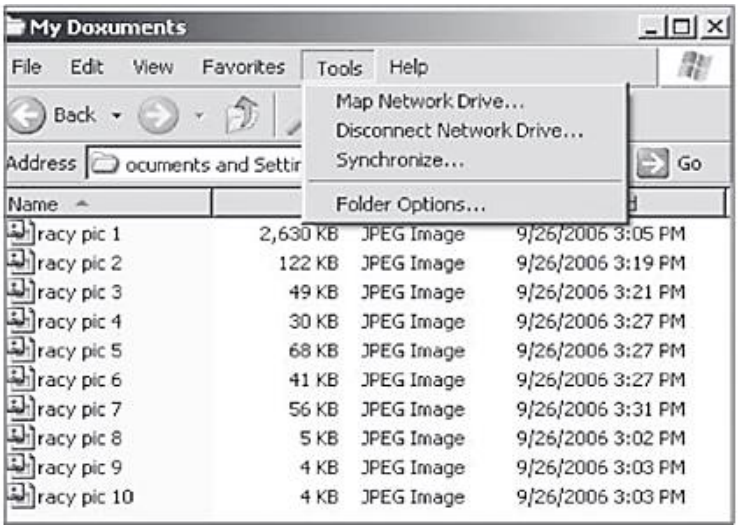
Buka folder yang tampak salah nama tetapi sah bernama My Doxuments (lihat Gambar 4.2).



Gambar 4.2 Isi Folder Dokumen Saya

Langkah 3

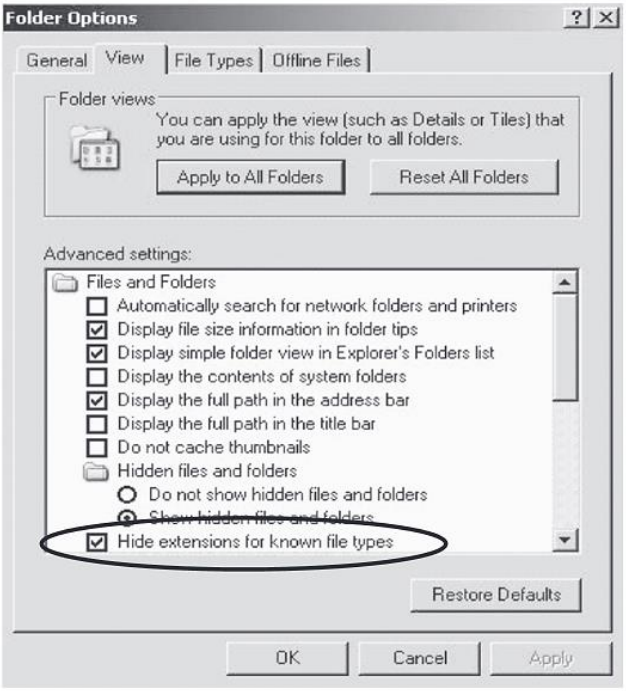
Buka tab Tools dan pilih Folder Options (lihat Gambar 4.3).



Gambar 4.3 Bersiap untuk Mengubah Ekstensi File

Langkah 4

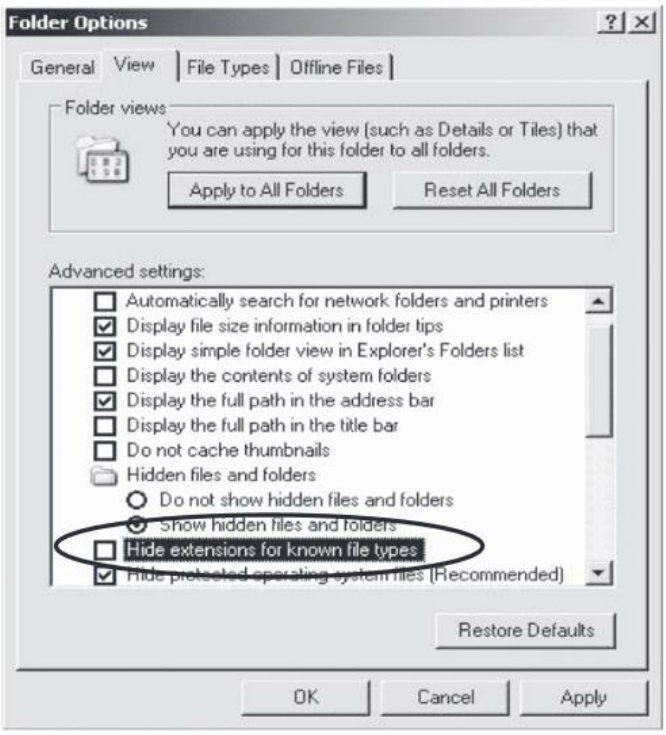
Buka tab View (lihat Gambar 4.4).



Gambar 4.4 Opsi Manajemen File untuk Menyembunyikan Ekstensi File

Langkah 5

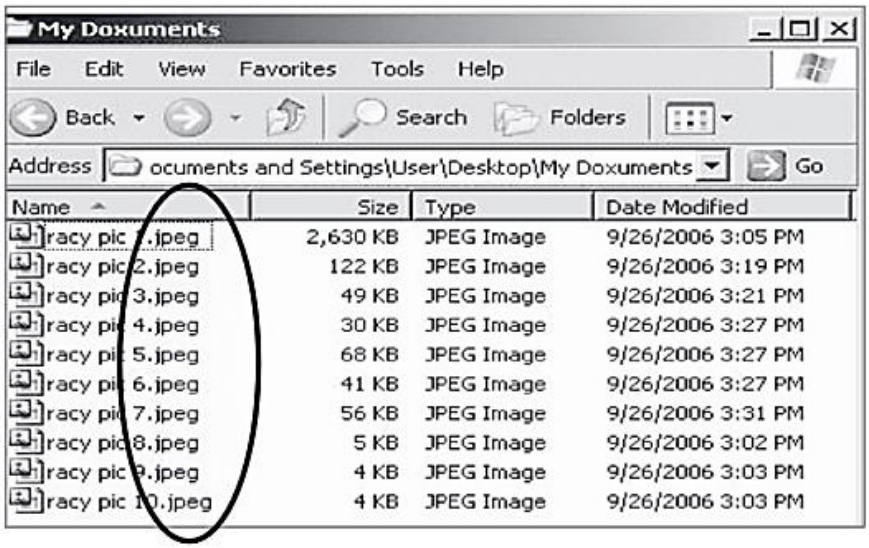
Hilangkan centang pada "Hide extensions for known file types" (lihat Gambar 4.5).



Gambar 4.5 Opsi untuk Menyembunyikan Ekstensi File yang Tidak Dipilih

Langkah 6

Jenis ekstensi file muncul (lihat Gambar 4.6).



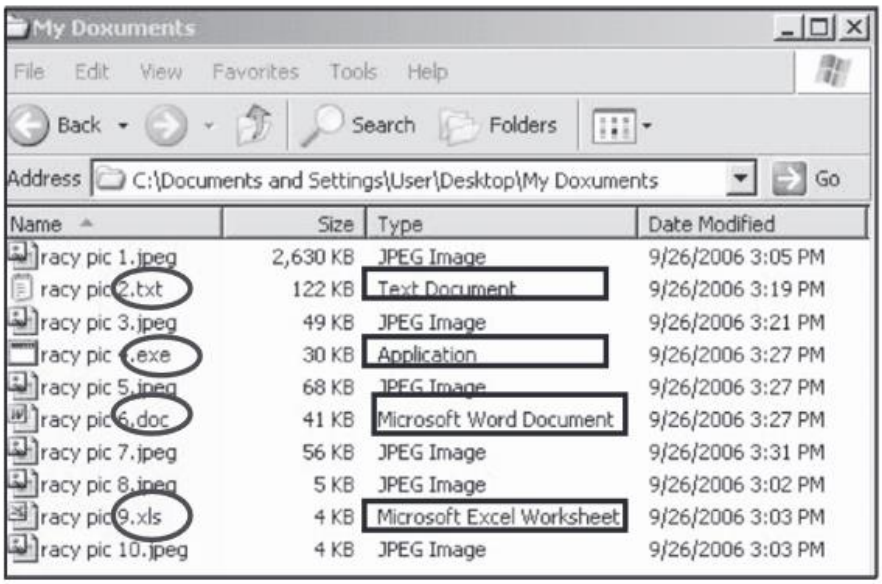
Gambar 4.6 Jenis Ekstensi File Muncul

Langkah 7

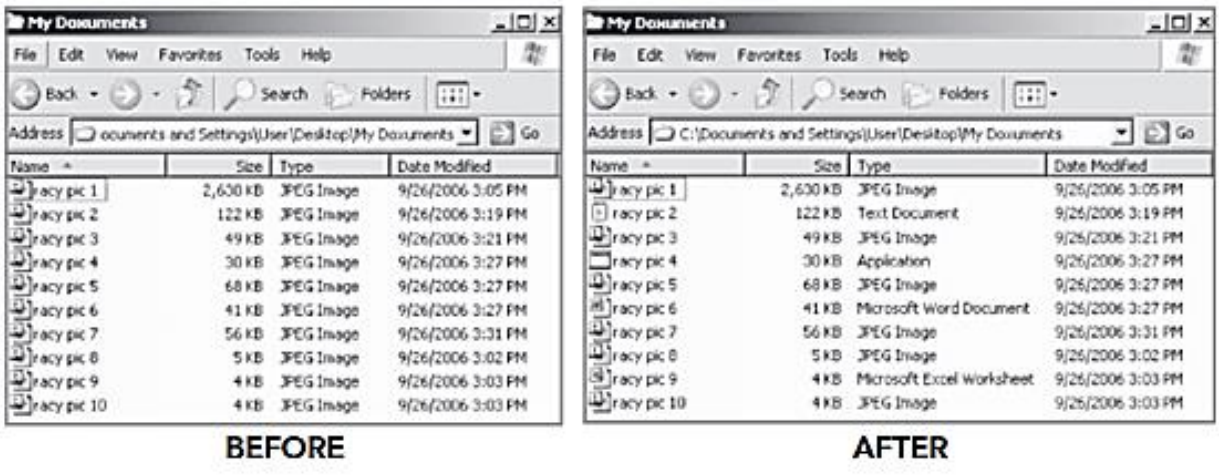
Klik kanan pada nama file untuk mengganti nama file, termasuk memberikan jenis ekstensi file yang valid. Jenis file diubah berdasarkan ekstensi yang disediakan (lihat Gambar 4.7).

Langkah 8

Klik Hide extensions for known file types, untuk menyembunyikan ekstensi file baru (lihat Gambar 4.8).

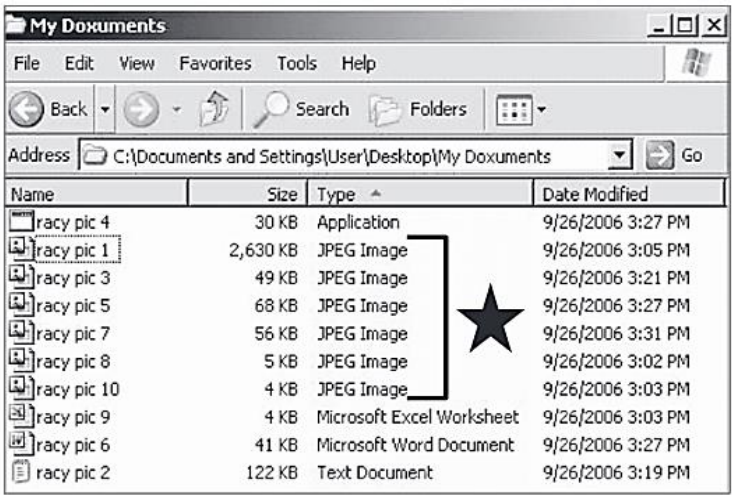


Gambar 4.7 Mengganti Nama File dan Ekstensi File



Gambar 4.8 File dan Ekstensi Asli dan File setelah Mengubah Ekstensinya

Dulu ada 10 file gambar JPEG, sekarang hanya ada enam (lihat Gambar 4.9). Memindai hanya untuk file gambar akan mengakibatkan hilangnya empat file dengan ekstensi yang dimodifikasi!



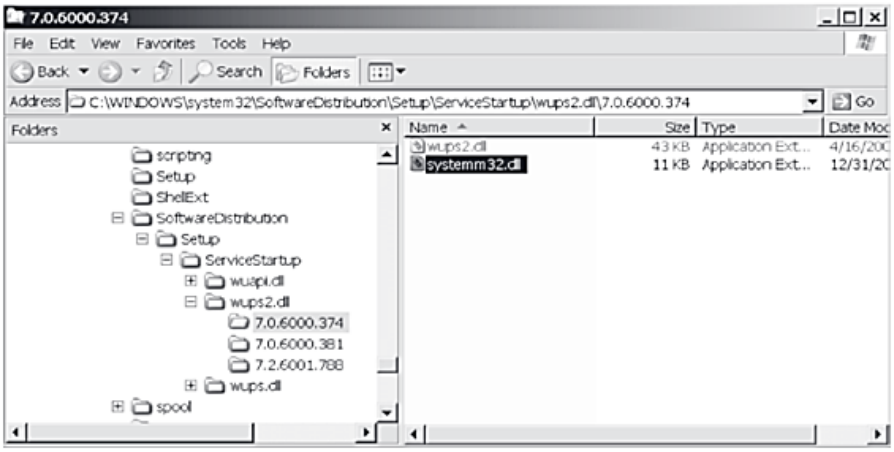
Gambar 4.9 Hasil Perubahan Ekstensi File

Nasihat untuk penjahat potensial: Mungkin bijaksana untuk mengganti nama file dari "gambar cabul" menjadi sesuatu yang lebih tidak mencolok! Juga, menggunakan atau mengganti nama folder yang kurang terkenal yang terkubur lebih jauh di dalam pohon direktori mungkin menguntungkan.

Ingat Windows melihat ekstensi file terlebih dahulu, dan menyerahkan file itu ke aplikasi yang sesuai untuk dibuka. Aplikasi Microsoft Word yang mencoba membuka file .JPEG atau .TIF akan mencoba meluncurkan atau membuka file dan melaporkan kesalahan karena file tersebut, apa pun namanya, sebenarnya bukan file Microsoft Word.

4.5 FILE DAN EDITOR HEX

Dalam kasus pencurian kekayaan intelektual kami, Ronelle Sawyer sedang menyelidiki apakah Jose McCarthy berpotensi terlibat dalam distribusi kekayaan intelektual organisasinya secara tidak sah kepada pesaing, Janice Witcome, Direktur Pelaksana Perusahaan XYZ. Ronelle dihadapkan pada pemeriksaan jutaan keping data bukti potensial yang berada di hard drive Jose, seperti kemunculan string karakter "X", "Y", dan "Z". Untuk menambah kerumitan tugas Ronelle, file-file ini dapat dengan mudah diubah namanya dan dipindahkan ke lokasi yang terkubur jauh di dalam struktur folder logis komputer.

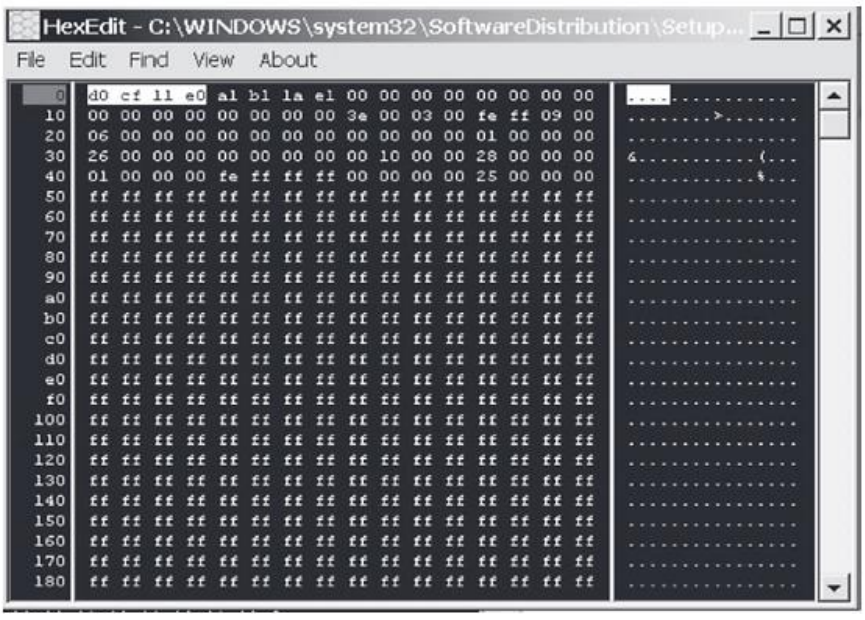


Gambar 4.10 Folder Windows 7.0.6000.374

Gambar 4.10 menampilkan isi folder (7.0.6000.374) yang terkubur di dalam struktur folder Windows, lokasi yang biasanya berisi file sistem seperti .DLL. Nama foldernya adalah [C:\WINDOWS\system32\SoftwareDistribution\Setup\ServiceStartup\wups2.dll\7.0.6000.374] Folder Windows 7.0.6000.374 berisi dua file: file #1, wups2.dll, dan file #2, systemm32.dll. Ingat, bisa ada ratusan bahkan ribuan folder dan bahkan lebih banyak file, yang semuanya mungkin tampak tidak penting karena tersebar dan disimpan di seluruh hard drive individu., Jadi jenis file ini (mis., .DLL) tampak cukup normal, bukan?

Mari kita lihat file dengan editor HEX. Ada banyak editor HEX yang tersedia, sebagian besar gratis untuk diunduh. Google adalah teman Anda; pastikan Anda mengunduh editor HEX dan bukan Trojan.

Gambar 4.11 Tampilan Editor HEX dari File wups2.dll



Gambar 4.12 Tampilan Editor HEX dari File systemm32.dll

Gambar 4.11 memperlihatkan File #1 “wups2.dll” yang dilihat di editor HEX. Gambar 4.12 memperlihatkan File #2, “systemm32.dll” yang dilihat di editor HEX.

Terkandung dalam informasi format file adalah tanda tangan file, terkadang disebut sebagai “Angka Ajaib”. Angka ajaib disebut sebagai angka ajaib karena tujuan dan signifikansi nilainya tidak terlihat tanpa pengetahuan tambahan. Istilah Angka Ajaib juga digunakan dalam pemrograman untuk merujuk pada konstanta yang digunakan untuk beberapa tujuan tertentu tetapi keberadaan atau nilainya tidak dapat dijelaskan tanpa informasi tambahan.[2]

4.6 TANDA TANGAN FILE

Tanda tangan file adalah biner yang mengidentifikasi file tertentu: data yang akan membantu identifikasi file ke perangkat lunak asli atau induknya. Untuk format file umum, tanda tangan file dengan mudah mewakili nama jenis file. Misalnya, file gambar yang sesuai dengan format GIF87a yang banyak digunakan dalam HEX sama dengan 0x474946383761; ketika diubah menjadi ASCII itu sama dengan GIF87a. ASCII adalah standar de facto yang digunakan oleh komputer dan peralatan komunikasi untuk pengkodean karakter (yaitu, menghubungkan karakter abjad dan karakter lainnya dengan angka).

Ingat, “0x - zerox” mengacu pada nilai notasi HEX(adecimal). Oleh karena itu, nilai 0x474946383761 tidak dan tidak boleh ditafsirkan sebagai nilai desimal, melainkan sebagai representasi heksadesimal dari persamaan desimal.

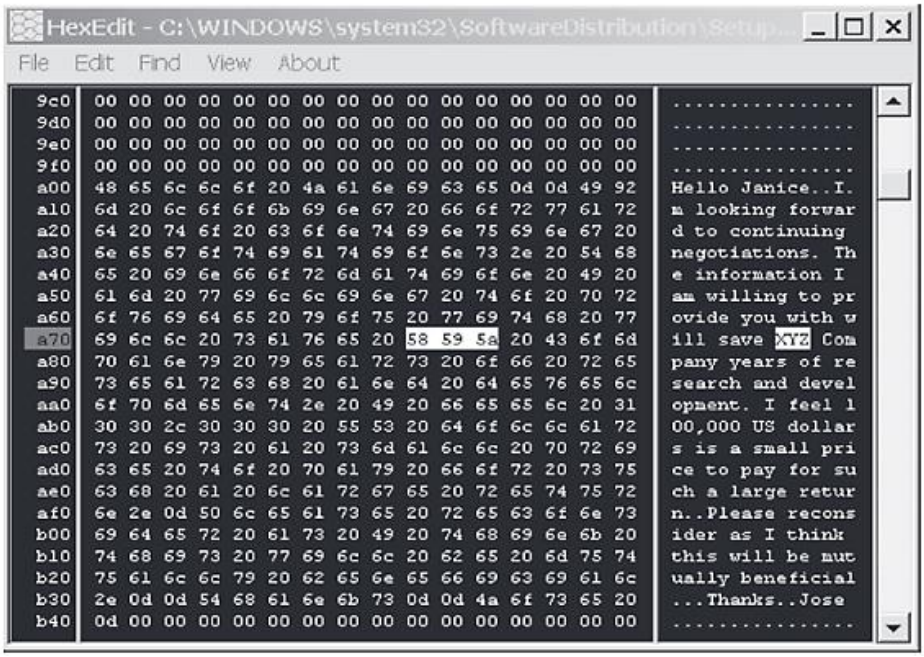
Demikian juga, file signature untuk file gambar yang memiliki format GIF89a yang diperkenalkan berikutnya adalah 0x3474946383961. Untuk kedua jenis file GIF (Graphic Interchange Format), tanda tangan file menempati enam byte pertama file. Mereka kemudian diikuti oleh informasi umum tambahan (yaitu, metadata) tentang file tersebut.

Demikian pula, tanda tangan file yang umum digunakan untuk file gambar JPEG (Joint Photographic Experts Group) adalah 0x4A464946, yang setara dengan ASCII dari JFIF (JPEG File Interchange Format). Namun, tanda tangan file JPEG bukanlah byte pertama dalam file; sebaliknya, mereka mulai dengan byte ketujuh. Contoh tambahan termasuk 0x34D546864 untuk file MIDI (Musical Instrument Digital Interface) dan 0x425a6831415925 untuk file terkompresi bzip2.

Perhatikan di editor HEX file "systemm32.dll" (Gambar 4.12) kita melihat tanda tangan file "d0 cf 11 e0." Ini adalah tanda tangan file yang dikenal untuk MS Word. Bahkan, Microsoft memilih kode biner untuk mengidentifikasi file-nya dengan beberapa pemikiran sebelumnya, karena representasi HEX dari biner (yaitu d0 cf 11 e0) hampir menjelaskan (jika Anda melihat lebih dekat dan menggunakan imajinasi Anda) kata "docfile" (d0c f11e 0). Mungkin itu adalah contoh humor teknologi atau desainer aplikasi yang pintar, meskipun mungkin bosan?

Penasaran? Mengapa file dengan ekstensi file .dll berisi tanda tangan file "docfile"? Jika kita menggulir ke bawah melalui editor HEX lagi, kita juga akan melihat teks aktual yang terdapat di dalam file (Gambar 4.13). Perhatikan nilai HEX 58 59 5a dan persamaan ASCII-nya, “XYZ” yang terdapat dalam Panel Karakter ASCII. Editor HEX, sebagai bagian dari "perangkat

alat" mereka, akan secara otomatis mengonversi HEX ke ASCII, sehingga proses konversi HEX ke ASCII yang ketat yang kami lakukan di diskusi sebelumnya tidak diperlukan di sini.



Gambar 4.13 Teks yang Terkandung dalam File .dll “systemm32.dll”

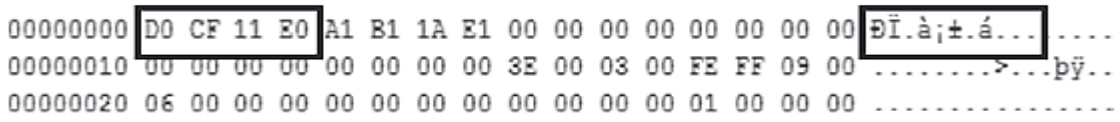
Nilai biner yang mewakili teks "XYZ" terkandung dalam file "systemm32.dll." Ketika kita mengganti nama "systemm32.dll" menjadi "systemm32.doc" dan mengklik dua kali file tersebut, kita akan melihat bahwa itu bukan file sistem (file .dll) tetapi dokumen Word (.doc). Contoh ini menunjukkan pentingnya HEX saat melihat file atau mencoba melihat file. Karena kita sudah mengetahui tanda tangan file untuk dokumen MS Word, "d0 cf 11 e0", kita sekarang dapat mencari keseluruhan drive Jose McCarthy untuk karakter HEX spesifik tersebut, mengungkapkan keberadaan dokumen MS Word. Perhatikan kami tidak dapat mencari drive untuk persamaan ASCII. Persamaan ASCII dari biner yang diwakili oleh HEX adalah “.....”. “.....” terkadang akan ditampilkan saat tidak ada ASCII yang setara dengan kode biner, seperti dengan tanda tangan file. Lihat Lampiran 4C untuk ulasan dan diskusi lebih lanjut tentang tanda tangan file.

4.7 ASCII BUKAN TEKS ATAU HEX

Ingat, ASCII setara dengan HEX d0 cf 11 e0, bukan ekstensi file ".doc." Mungkin tidak selalu ada ASCII yang setara dengan tipe file; ini adalah salah satu alasan untuk menggunakan HEX, atau pentingnya HEX. Mungkin tidak selalu ada ASCII yang setara dengan, katakanlah, header file (seperti dalam kasus ini), ergo HEX.

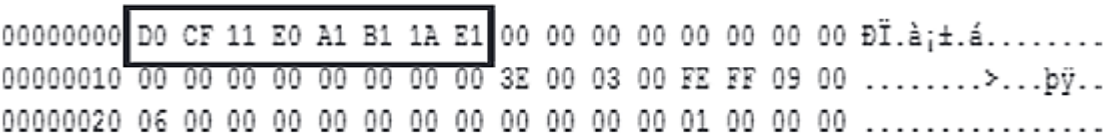
Ingat, ASCII memiliki batasan dan diperluas dengan Unicode (setara dengan Unicode dari HEX D0 CF 11 E0 adalah ÐÏ.à¡±.á (dilihat dan diucapkan sebagai DIATA). (Lihat Gambar 4.14.) Namun, ini bukanlah sesuatu mudah dicari karena karakter tidak semuanya berbasis teks.

Buka file .doc (sebelum Office 2007) di editor HEX dan Anda akan melihat titik-titik “.” untuk banyak karakter HEX yang bervariasi. Tanda tangan file untuk nilai HEX ini mungkin tidak memiliki persamaan ASCII serta beberapa kode dan data header file lainnya.



Gambar 4.14 ASCII Setara dengan HEX D0 CF 11 E0 (MS Word .doc file)

Delapan (8) byte pertama terkandung dalam pengidentifikasi file dokumen majemuk tetap, sehingga pengidentifikasi file akan menjadi delapan (8) byte: D0 CF 11 E0 A1 B1 1A E1. Namun, ketika mencari melalui HEX empat (4) byte pertama tentu saja sudah cukup, tetapi yang terbaik adalah yang paling akurat, jadi tanda tangan file lengkap untuk HEX D0 CF 11 E0 (file .doc MS Word) adalah D0 CF 11 E0 A1 B1 1A E1 (Gambar 4.15).



Gambar 4.15 ASCII Setara dengan HEX D0 CF 11 E0 (MS Word .doc file)

Untuk perincian yang lebih rinci tentang header file majemuk, silakan lihat Lampiran 4D.

4.8 NILAI TANDA TANGAN FILE

Kami melihat bahwa meskipun sebuah file diganti namanya, kami masih dapat melihat isinya. Jika kami mencari representasi biner dari "XYZ" di seluruh drive, kami akan menemukan nilai ini terlepas dari ekstensi file atau tanda tangan file yang dimodifikasi. Seperti yang telah dibahas sebelumnya, berkali-kali dalam operasi normal sehari-hari dan pemrosesan file, file yang dihapus dan metadata yang terkait akan ditimpa sebagian, mungkin seluruh tanda tangan file hilang, atau informasi pemformatan penting lainnya dan bahkan beberapa teks . Namun, jika nilai biner yang mewakili sepotong bukti (seperti dalam kasus kami, "XYZ") tetap berada di dalam sisa, maka file tersebut dapat ditemukan.

Penting untuk dicatat bahwa pemeriksa forensik tidak dapat selalu bergantung pada memiliki file utuh atau file dengan ekstensi file resmi (benar) (yaitu, tipe file), tersedia dalam format aslinya, untuk melakukan analisis. Office 2007 telah secara drastis mengubah format file MS Word. Contoh sebelumnya berlaku untuk Office 2003 dan versi sebelumnya. Dokumen Microsoft Office 2007 sekarang disimpan dalam apa yang disebut sebagai Format File Office Open XML. Ini pada dasarnya adalah file ZIP dari berbagai dokumen XML yang menjelaskan seluruh dokumen. Inti dari contoh sebelumnya bukan untuk membahas cara kerja bagian dalam format file MS Word, tetapi untuk menunjukkan bagaimana format file dan tanda tangan bekerja secara umum.

Semua tanda tangan file berbeda dan akan terus berkembang. Tujuannya di sini bukan untuk mencakup semua tanda tangan file, tetapi untuk memberi pembaca contoh yang sangat praktis tentang relevansi tanda tangan file dalam menemukan dan mengidentifikasi informasi yang berpotensi memberatkan sebagai bagian dari penyelidikan forensik dunia maya. Ada database dan tabel tanda tangan file yang tersedia di Internet. Sebagian besar alat forensik dapat mengidentifikasi tanda tangan file dan informasi header, dan akan memverifikasi jenis file dengan cara ini. Alat forensik akan mengonversi biner ke ASCII, memverifikasi tanda tangan file, dan mencari string biner (atau kata kunci, seperti "XYZ") tanpa banyak usaha dari pemeriksa forensik, dan sekarang Anda tahu BAGAIMANA perangkat lunak menyelesaikannya.

4.9 FILE KOMPLEKS: FILE GABUNGAN, TERKOMPRESI, DAN TERENKRIPSI

Sebelum mengakhiri bagian ini, ada file lain yang lebih kompleks yang perlu didiskusikan: file majemuk, terkompresi, dan terenkripsi. Kompleksitas penuh dari file-file ini tidak tercakup di sini, karena ada buku yang ditulis tentang masing-masingnya. Namun, kami menjelaskan beberapa dasar dan kepentingannya dalam forensik. File gabungan adalah format file yang terdiri dari banyak file. File majemuk itu sendiri tidak lebih dari wadah untuk file-file itu. Struktur dalam file gabungan mirip dengan sistem file nyata yang terdiri dari hierarki penyimpanan dengan satu direktori induk. Ada folder direktori root, anak-anak yang terkandung di dalamnya, dan file (aliran data) yang terkandung di dalamnya. File gabungan terkadang dikaitkan dengan file Format File Biner (CFBF) Microsoft.

Semua alokasi ruang dalam File Gabungan dilakukan dalam "potongan" atau unit yang disebut sektor. Ukuran sektor ditentukan pada waktu pembuatan File Gabungan, dan sektor tersebut biasanya berukuran 512 byte. Aliran virtual terdiri dari urutan sektor. Paling sederhana, Compound File Binary Format adalah wadah, dengan sedikit batasan pada apa yang dapat disimpan di dalamnya. Namun, dalam forensik, istilah file majemuk kadang-kadang digunakan lebih longgar, mewakili file apa pun yang mungkin berisi struktur direktori. Sekali lagi, tujuan kami bukan untuk membahas jenis file atau perangkat lunak tertentu, tetapi konsep secara umum. Seperti file lainnya, header file dari file gabungan akan berisi tanda tangan file, yang mengidentifikasi file; itu juga akan berisi informasi yang diperlukan untuk menginterpretasikan sisa file seperti ukuran file dan lokasi penyimpanan.

Metadata inilah yang memungkinkan perangkat lunak merekonstruksi file ke dalam format file yang sesuai yang akan menampilkan informasi spesifik file (yaitu, ukuran, tanggal pembuatan, tanggal perubahan, dll.). Oleh karena itu file perlu “direkonstruksi” oleh perangkat lunak induknya agar data dapat dibaca atau diakses. Untuk penjelasan lebih lanjut, kami biasanya menganggap penyimpanan data sebagai linier. Misalnya, pertimbangkan informasi dalam aliran data berikut, "XYZ Corp." Data ditampilkan dalam pola bersebelahan linier, X sebelum Y dan Y sebelum Z. Jika data itu ditampilkan dalam pola nonlinier, kita mungkin akan melihat, "oZ pYCrX." Jika data yang sama sekarang tidak bersebelahan, data lain dari file gabungan yang sama juga dapat terjalin (mis., …?>>o….Z^qL p….77Ymn….C@ qwerbsbdX……,,,,). Aliran data asli "XYZ Corp" tidak mudah dibedakan sekarang. Bahkan mencari persamaan HEX tidak akan membantu kami mengungkap data dalam contoh ini. Kami membutuhkan set instruksi untuk merekonstruksi data ini.

4.10 MENGAPA FILE GABUNGAN ADA?

File menjadi lebih kompleks dan perlu memuat banyak informasi. Banyak file berisi teknologi Penautan Objek dan Penyematan (OLE), di mana satu file dapat berisi banyak file. OLE (object linking and embedding) memungkinkan pengguna untuk mengintegrasikan data dari berbagai aplikasi. Penautan objek memungkinkan pengguna untuk berbagi satu sumber data untuk objek tertentu. Dokumen berisi nama file yang berisi data, beserta gambar datanya. Saat sumber diperbarui, semua dokumen yang menggunakan data juga diperbarui. Dengan penyematan objek, satu aplikasi (disebut sebagai "sumber") menyediakan data atau gambar yang akan dimuat dalam dokumen aplikasi lain (disebut sebagai "tujuan"). Aplikasi tujuan berisi data atau gambar grafik, tetapi tidak memahaminya atau memiliki kemampuan untuk mengeditnya. Ini hanya menampilkan, mencetak, dan/atau memutar item yang disematkan. Untuk mengedit atau memperbarui objek yang disematkan, objek tersebut harus dibuka di aplikasi sumber yang membuatnya. Ini terjadi secara otomatis saat Anda mengklik dua kali item atau memilih perintah edit yang sesuai saat objek disorot.

Meskipun penyematan tidak memungkinkan pengguna untuk memiliki satu sumber data, penyematan membuatnya lebih mudah untuk mengintegrasikan aplikasi. Objek yang disematkan berisi data sebenarnya untuk objek tersebut, nama aplikasi yang membuatnya, dan gambar datanya.

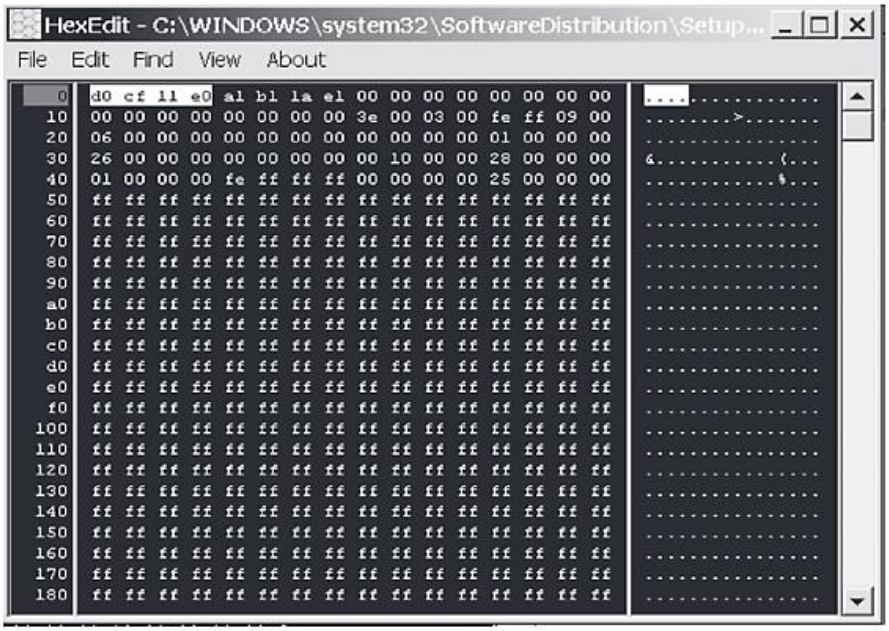
Misalnya, dokumen MS Word mungkin berisi gambar JPG; file di dalam file. File majemuk memungkinkan akses inkremental, memungkinkan komponen individu untuk diakses tanpa perlu seluruh file. Ini dapat menghemat waktu dan sumber daya dengan tidak harus memuat seluruh file, hanya bagian atau bagian yang diinginkan.

4.11 FILE TERKOMPRES

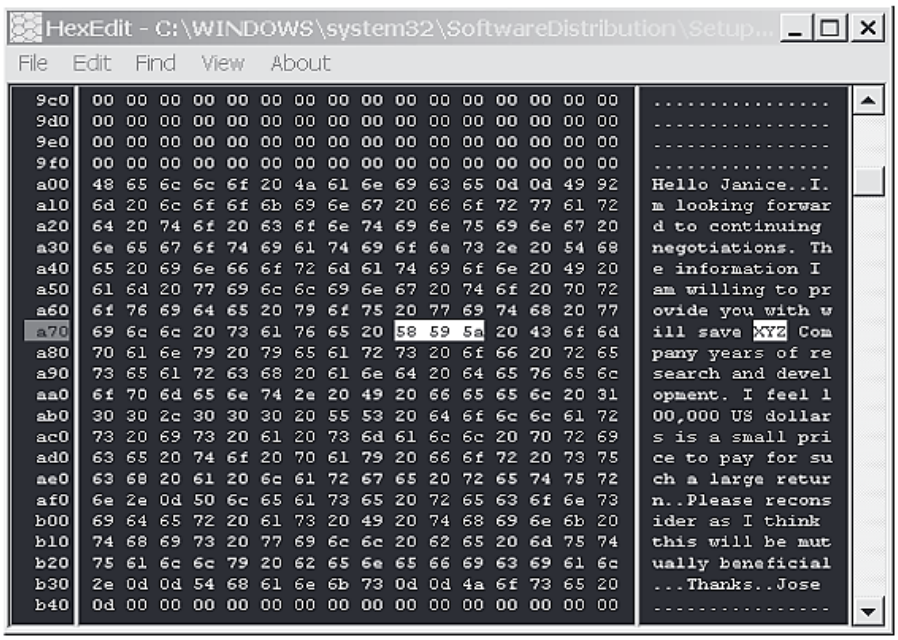
Saat kami melanjutkan diskusi kami, file yang dikompresi pada dasarnya adalah file gabungan (dan kadang-kadang disebut demikian dalam komunitas forensik) yang dikompresi. Mereka bekerja dengan cara yang sama; namun, juga terkandung di dalam file gabungan adalah instruksi kompresi.

Ekstensi file umum yang terkait dengan file terkompresi adalah .ZIP. Format file ini telah menjadi arus utama dan didukung oleh banyak utilitas perangkat lunak selain perangkat lunak induknya, PKZIP. Format file .ZIP dirilis untuk umum, menjadikannya format terbuka yang digunakan oleh program lain termasuk format Microsoft Open Office XML. Nama ekstensi file ZIP sering digunakan untuk menggambarkan format file arsip apa pun. Ada format file ZIP lainnya termasuk WINZIP, 7-Zip, GZip, dan RZip. Format file dari file terkompresi (atau file .ZIP) berubah tergantung pada algoritme kompresinya. Algoritma adalah operasi atau instruksi matematika untuk menyelesaikan tugas, dalam hal ini mengompresi data. Ini adalah metode pengkodean data menggunakan bit lebih sedikit daripada yang digunakan dalam pengkodean asli. Algoritma rumit untuk sedikitnya dan buku telah ditulis mengenai topik ini. Untuk mencontohkan perbedaan antara file biasa dan file yang lebih kompleks (gabungan atau terkompresi), sebaiknya periksa file serupa dalam kedua format.

Mari kita periksa surat dari Jose McCarthy, yang disita sebagai bagian dari penyelidikan Ronelle Sawyer. Surat yang diperiksa adalah format file MS Word (yaitu,.doc), seperti yang diperjelas saat dilihat melalui editor HEX, ditampilkan lagi di sini. Kita dapat dengan mudah melihat signature file doc, d0 cf 11 e0, ditampilkan di editor HEX pada Gambar 4.16. Dengan mengetahui tanda tangan file untuk dokumen MS Word, kita dapat dengan mudah mengidentifikasi dan/atau mencari teks yang terdapat dalam file .doc ini, dan dengan demikian, menemukan referensi ke perusahaan "XYZ", seperti yang ditunjukkan pada Gambar 4.17.



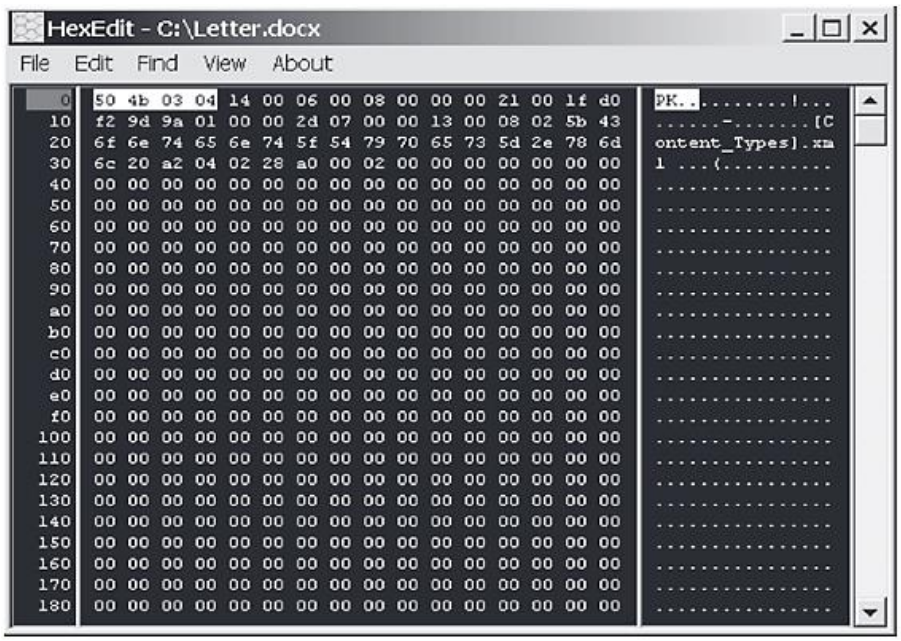
Gambar 4.16 Tanda Tangan File Dokumen MS Word Office 2003



Gambar 4.17 Identifikasi HEX dan ASCII Perusahaan "XYZ".

Namun, apa yang terjadi ketika aplikasi ditingkatkan? Bagaimana pengaruhnya terhadap tanda tangan file aplikasi? Untuk melihat hasil dari perubahan dalam pemformatan file aplikasi perangkat lunak, mari kita lihat file dokumen yang sama dari Jose McCarthy dengan editor HEX, ketika Microsoft Office 2007 daripada Office 2003 digunakan untuk membuat dokumen.

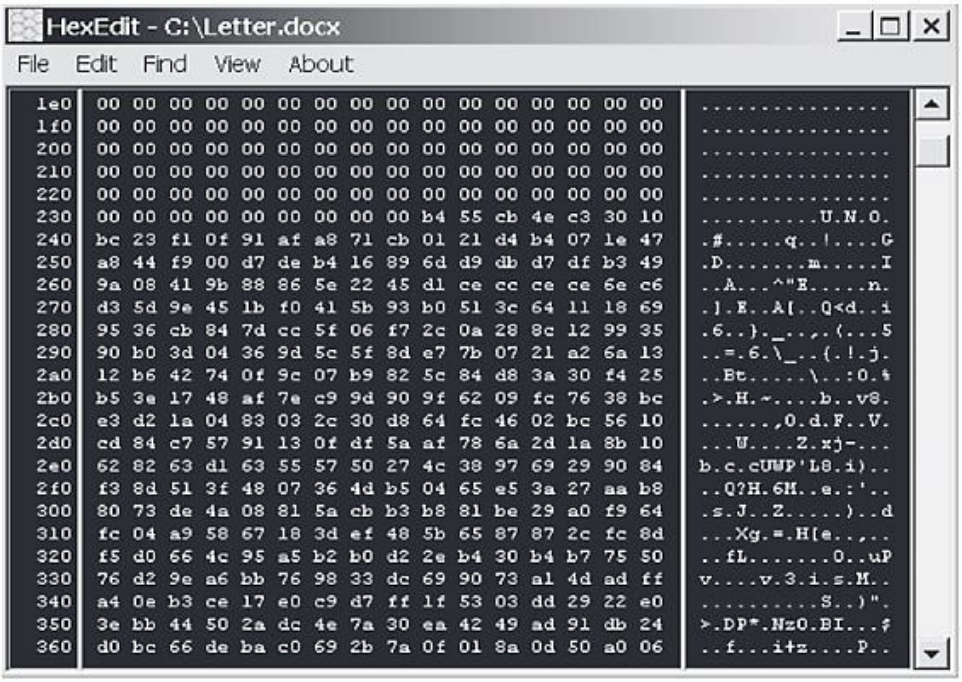
Kita lihat pada Gambar 4.18 bahwa file signature telah berubah. Jika Anda mencari tanda tangan file yang cocok dengan 50 4B 03 04, Anda akan melihatnya sesuai dengan tanda tangan file .ZIP (panel ASCII menampilkan format PK). Dengan rilis Office '07,



Gambar 4.18 Tanda Tangan File Dokumen Office 2007 MS Word

Dokumen Microsoft Word sekarang menggunakan tanda tangan formatfile yang sama dengan file .ZIP. Apa pentingnya penyelidikan forensik dunia maya dan apa artinya ini? Sebagai permulaan, artinya file tersebut adalah file gabungan yang terdiri dari file lain. Jika kami melihat keseluruhan file dengan editor HEX kami, kami tidak akan menemukan karakter ASCII yang dapat dibaca (lihat panel ASCII pada Gambar 4.17).

Mengapa? Struktur file dan instruksi perakitan terdapat di dalam file; dengan demikian, file tersebut perlu Dipasang oleh perangkat lunak aslinya agar konten dapat dilihat. Seperti dapat dilihat pada Gambar 4.19, representasi ASCII tidak dapat diidentifikasi. Melihat dan, yang lebih penting, mencari isi dari file “kompleks” ini dapat dilakukan setelah file terpasang. Alat forensik menggabungkan perangkat lunak untuk memasang ini sehingga pencarian dapat dilakukan. Jika file kompleks ini tidak dipasang maka tidak ada hasil pencarian yang akan diperoleh.



Gambar 4.19 Representasi ASCII yang Dapat Diidentifikasi untuk File .doc

4.12 FORENSIK DAN FILE TERENKRIPSI

File terenkripsi juga rumit tetapi berbeda karena kunci enkripsi diperlukan untuk mendekripsi file terenkripsi. Enkripsi menggunakan algoritme (sandi) untuk mengubah atau mengubah data dalam upaya mencegah rekonstruksi oleh mereka yang tidak memiliki set instruksi, alias Kunci Enkripsi. Dekripsi mengacu pada proses kebalikan dari membuat data dapat dibaca atau diakses. Enkripsi adalah metode dimana kerahasiaan data dapat dilindungi. Sebagian besar, file terenkripsi tidak dapat didekripsi tanpa kunci enkripsi (alias kata sandi). Proses enkripsi menggunakan algoritme atau cipher untuk mengubah teks biasa secara matematis bersama dengan kunci enkripsi (kata sandi), sehingga menyandikannya sedemikian rupa sehingga tidak terbaca atau tidak dapat diuraikan. Dengan kunci dekripsi (kata sandi) yang benar, data kemudian dijalankan melalui teks sandi terkait (algoritma) dan diubah kembali menjadi teks bersih, yang secara default didekripsi. Ingat, seluruh proses ini terjadi dalam biner, sebagai 0 atau 1. Sandi itulah yang benar-benar mengubah file; kata sandi hanyalah sekumpulan data yang digunakan untuk "mencampur secara matematis" dan menggerakkan proses, mengubah data teks biasa menjadi produk akhir yang tidak dapat dibaca.

4.13 STRUKTUR CIPHER

Struktur sandi bergantung pada jenis sandi. Jenis sandi bervariasi tetapi umumnya dapat dikategorikan sebagai berikut:

- ▪ Blokir atau streaming. Block cipher umumnya bekerja pada bit data dengan panjang tetap yang disebut blok. Cipher dapat mengambil 256-bit blok data plaintext dan mengenkripsinya, yang menghasilkan blok data terenkripsi 256-bit. Dalam sebuah stream cipher, bit-bit plainteks dienkripsi satu per satu bersama dengan kunci enkripsi.

- Simetris atau asimetris. Dalam enkripsi simetris, kunci enkripsi atau kata sandi yang sama digunakan untuk enkripsi dan dekripsi, sedangkan dengan enkripsi asimetris, kunci yang berbeda digunakan. Enkripsi kunci syMMetric intuitif karena kata sandi yang sama digunakan untuk mengenkripsi atau mendekripsi data. Enkripsi kunci AsyMMetric, atau kriptografi kunci publik, menggunakan dua kunci enkripsi yang berbeda, kunci publik dan privat. Data dienkripsi menggunakan kunci publik seseorang, di mana setiap orang dapat memiliki akses atau bahkan didistribusikan. Namun, data hanya dapat didekripsi menggunakan kunci pribadi orang tersebut, yang dirahasiakan oleh individu tersebut.

Ada berbagai metode enkripsi yang tersedia, seperti Advanced Encryption Standard (AES), yang saat ini merupakan standar yang diadopsi oleh pemerintah Amerika Serikat dan salah satu metode enkripsi terpopuler yang tersedia dan digunakan saat ini. Ada algoritme (atau format) enkripsi lain yang tersedia dan banyak buku telah ditulis mengenai masing-masing algoritme tersebut. Teks ini tidak mencakup berbagai standar enkripsi. Atribut serupa yang dimiliki oleh semua jenis file "kompleks" yang dibahas ini adalah bahwa mereka berisi beberapa level atau bentuk instruksi yang diperlukan untuk merekonstruksi file. Jika informasi itu ditimpa atau hilang, kemampuan untuk mengambil data yang terkandung dalam file akan sangat terganggu.

Jika data instruksional yang diperlukan untuk merekonstruksi file gabungan hilang, ditimpa, dihancurkan, disusupi, dll., file tersebut mungkin tidak dapat dipulihkan, meskipun data yang berisi bukti (mis., XYZ Corp) mungkin masih terdapat di dalam file itu sendiri.

Namun, dengan demikian, dimungkinkan untuk merekonstruksi file kompleks yang telah ditimpa sebagian. Analis forensik kreatif, terdepan, inovatif, dan sangat cerdas; mereka telah mengembangkan solusi untuk beberapa masalah yang paling kompleks. Namun, memulihkan data dengan metode "tunjuk dan klik" yang normal mungkin tidak selalu dapat dilakukan.

4.14 RINGKASAN DAN PENGAYAAN

Penting untuk dipahami bahwa tidak semua nilai biner dapat diubah menjadi ASCII yang dapat dibaca. ASCII adalah kode, berdasarkan urutan abjad Inggris, dan tidak semua data yang ada di dalam komputer harus berbasis teks (ASCII). Banyak program atau aplikasi perangkat lunak yang ditulis dalam kode pemrograman yang tidak berbasis ASCII. Kode pemrograman ini tidak dimaksudkan untuk dilihat di ASCII, ini dimaksudkan untuk menjalankan suatu fungsi. Ingat dari diskusi kita sebelumnya, semua fungsi komputer didasarkan pada matematika, bukan bahasa Inggris (atau Prancis, Cina, Slavia, Yunani, atau Arab); Oleh karena itu, kode harus didasarkan pada prinsip matematika, bukan prinsip tata bahasa.

Jenis atau format file didasarkan pada tanda tangan file, bukan ekstensi Microsoft Windows. Header file, termasuk tanda tangan file paling baik dilihat di HEX karena tidak ada representasi ASCII terkait yang terbaca atau dapat diidentifikasi. Seperti yang telah kita diskusikan, tanda tangan/header file adalah bagian dari file yang mengidentifikasi file tersebut ke perangkat lunak "induknya", bukan ke pengguna. Jadi, saat kita melihat editor HEX dan melihat nilai HEX muncul sebagai “ ” di Panel Karakter ASCII, ini bisa berarti bahwa mungkin

tidak ada menjadi representasi ASCII untuk nilai-nilai HEX tersebut. Nilai HEX, jika memang ada, bersifat unik dan karenanya dapat dicari. Sangat mudah (dan berpotensi berbahaya) untuk menjadi tergantung pada alat forensik dan melupakan mur dan baut dari proses teknologi, dan melupakan atau bahkan tidak menyadari BAGAIMANA jawabannya diperoleh.

Ketergantungan pada "alat" apa pun tanpa memiliki pemahaman yang kuat tentang cara kerja alat tersebut dapat menyebabkan bencana pribadi dan profesional bagi penyelidik forensik dunia maya.

Ini mirip dengan berhasil memberikan jawaban yang benar untuk semua pertanyaan pada ujian matematika, dan masih menerima nilai gagal karena Anda gagal menunjukkan pekerjaan Anda.

Jika diminta untuk menjelaskan bagaimana suatu jawaban diperoleh atau pada analisis data apa suatu kesimpulan dicapai, jika seseorang menjawab, "Saya menggunakan alat 'ABC' dan itu memberikan jawabannya," dan jika Anda tidak dapat menjelaskan bagaimana alat itu diperoleh jawaban atau bagaimana Anda dapat memvalidasi dan membuktikan bahwa jawaban yang Anda berikan benar, validitas dan kepercayaan jawaban Anda dapat dipertanyakan dan dicurigai. Gunakan alat, tetapi pastikan Anda tahu cara kerja alat dan cara mereplikasi hasilnya jika Anda harus melakukannya, tanpa alat.

PENGAYAAN

EKSTENSI FILE UMUM*

Ekstensi file umum yang baik untuk diketahui, diatur menurut format file.

File Teks

.doc -Dokumen Microsoft Word
.docx -Microsoft Word Buka Dokumen XML
.log -Berkas Log
.msg -Pesan Email Outlook
.pages -Dokumen Halaman
.rtf - File Format Teks Kaya
.txt - File Teks Biasa
.wpd -Dokumen WordPerfect
.wps -Dokumen Pengolah Kata Microsoft Works

File Data

.123 -Lotus 1-2-3 Spreadsheet
.accdb - Mengakses File Database 2007
.csv - File Nilai yang Dipisahkan Koma
.dat -Berkas Data
.db - File Basis Data
.dll - Perpustakaan Tautan Dinamis
.mdb -Database Microsoft Access
.pps -Peragaan Slide PowerPoint
.ppt - Presentasi PowerPoint
.pptx -Microsoft PowerPoint Buka Dokumen XML
.sdb - File Basis Data Dasar OpenOffice.org
.sdf - File Data Standar
.sql -Data Bahasa Kueri Terstruktur
.vcfv -Kartu Berkas
.wks - Lembar Kerja Microsoft Works
.xls - Lembar Bentang Microsoft Excel
.xlsx -Microsoft Excel Buka Dokumen XML
.xml - Berkas XML

File Gambar

.pct -Berkas Gambar
File Gambar Raster
.bmp -File Gambar Bitmap
.gif - File Format Interchange Grafis
.jpg -File Gambar JPEG
.png - Grafik Jaringan Portabel
.psd -Dokumen Photoshop
.psp - File Gambar Pro Toko Cat
.thm -File Gambar Thumbnail
.tif -Tag File Gambar

File Gambar Vektor

.ai -Adobe Illustrator File
.drw -File Gambar
.dxf -Menggambar File Format Pertukaran
.eps - Berkas PostScript Terenkapsulasi
.ps -Berkas PostScript
.svg - Berkas Grafik Vektor yang Dapat Diskalakan

File Gambar 3D

.3dm - Model 3D Badak
.dwg -File Database Gambar AutoCAD
.pln - File Proyek ArchiCAD

File Tata Letak Halaman

.indd -Adobe InDesign File
.pdf -File Format Dokumen Portabel
.qxd -Dokumen QuarkXPress
.qxp -Berkas Proyek QuarkXPress

File audio

.aac - File Pengkodean Audio Tingkat Lanjut
.aif - Format File Pertukaran Audio
.iff - Tukar Format File
.m3u -File Daftar Putar Media
.mid -Berkas MIDI
.midi -Berkas MIDI
.mp3 - Berkas Audio MP3
.mpa -MPEG-2 File Audio
.ra -File Audio Nyata
.wav - Berkas Audio WAVE
.wma -Berkas Audio Windows Media

File Video

.3g2 -3GPP2 Berkas Multimedia
.3gp -Berkas Multimedia 3GPP
.asf - File Format Sistem Lanjutan
.asx -File Pengalih Microsoft ASF
.avi -Audio Video Interleave File
.flv - Berkas Video Flash
.mov -Apple QuickTime Movie
.mp4 -MPEG-4 File Video
.mpg - Berkas Video MPEG
.rm -File Media Nyata
.swf -Film Flash
.vob -File Objek Video DVD
.wmv -Berkas Video Media Windows

File Web

.asp -Halaman Server Aktif
.css -Cascading Style Sheet
.htm - File Bahasa Markup Hiperteks
.html - File Bahasa Markup Hiperteks
.js - Berkas JavaScript
.jsp -Halaman Server Java
.php - File Praprosesor Hiperteks
.rss -Ringkasan Situs Kaya
.xhtml - Berkas Bahasa Markup Hiperteks yang Dapat Diperluas

File Huruf

.fnt -File Font Windows
.fon - File Font Umum
.otf -Font Jenis Terbuka
.ttf -TrueType Font

File Pengaya

.8bi -Plugin Photoshop
.plugin - Pengaya Mac OSX

.xll -File Tambahan Excel

File Sistem

.cab -Berkas Kabinet Windows
.cpl -Panel Kontrol Windows
.cur - Kursor Windows
.dmp -Dump Memori Windows
.drv -Pengemudi Perangkat
.key -Kunci Keamanan
.lnk -Pintasan File
.sys -Berkas Sistem Windows

File Pengaturan

.cfg -File Konfigurasi
.ini -File Inisialisasi Windows
.prf -File Profil Outlook

File yang Dapat Dijalankan

.app -Aplikasi Mac OS X
.bat - Berkas Kumpulan DOS
.cgi -Skrip Antarmuka Gateway Umum
.com - File Perintah DOS
.exe - File yang Dapat Dijalankan Windows
.pif - File Informasi Program
.vb - Berkas VBScript
.ws -Skrip Windows

File Terkompresi

.7z -7-Zip File Terkompresi
.deb -Paket Perangkat Lunak Debian
.gz - File Zip Gnu
.pkg -Paket Pemasang Mac OS X
.rar - Arsip Terkompresi WinRAR
.sit -Stuffit Arsip
.sitx -Stuffit X Arsip
.zip - Berkas Zip
.zipx - File Zip yang Diperpanjang

File yang Dikodekan

.bin -File Enkode Macbinary II
.hqx -BinHex 4.0 File yang Disandikan
.mim -Pesan Surat Internet Serbaguna
.uue -Uuencoded File

File Pengembang

.c -C/C++ File Kode Sumber
.cpp -C++ File Kode Sumber
.java - File Kode Sumber Java
.pl -Skrip Perl

File Cadangan

.bak - File Cadangan
.bup - File Cadangan
.gho - File Cadangan Hantu Norton
.ori - File Asli
.tmp - File Sementara

File Disk

.dmg -Gambar Disk Mac OS X
.iso -File Gambar Disk
.toast -Toast Disk Gambar
.vcd -Virtual CD

File Permainan

.gam - File Game Tersimpan
.nes - File ROM Nintendo (NES).
.rom -N64 Permainan File ROM
.sav - Permainan Tersimpan

File Lain-Lain

.msi -Paket Pemasang Windows
.part - File yang Diunduh Sebagian
.torrent - Berkas BitTorrent
.yps -Yahoo! File Data Kurir

DATABASE TANDA TANGAN FILE

File Format	File Signature/ Magic Number	ASCII Equivalent	Windows OS File Extension
DeskMate document file	0D 44 4F 43	.DOC	DOC
Perfect Office document	CF 11 E0 A1 B1 1A E1 00	Ï.à¡±.á.	DOC
MS Word document	D0 CF 11 E0 A1 B1 1A E1	ÐÏ.à¡±.á	DOC
Office 2007 documents	50 4B 03 04 14 00 06 00	PK..	DOCX
Rich Text Format file	7B 5C 72 74 66 31	{\rtf1	RTF
Lotus 1-2-3 spreadsheet (v9)	00 00 1A 00 05 10 04	•••••••	123
Dynamic link library	4D 5A	MZ	DLL
PowerPoint slide show	D0 CF 11 E0 A1 B1 1A E1	ÐÏ.à¡±.á	PPS
PowerPoint presentation	D0 CF 11 E0 A1 B1 1A E1	ÐÏ.à¡±.á	PPT
PowerPoint presentation subheader (MS Office)	[512 byte offset] FD FF FF FF nn 00 00 00 (where nn has been seen with values 0x0E, 0x1C, and 0x43)	ýÿÿÿ....	PPT
PowerPoint presentation subheader (MS Office)	[512 byte offset] 00 6E 1E F0	.n.ð	PPT
PowerPoint presentation subheader (MS Office)	[512 byte offset] 0F 00 E8 03	..è.	PPT
Microsoft PowerPoint Office 2007 documents	50 4B 03 04 14 00 06 00	PK.....	PPTX

File Format	File Signature/ Magic Number	ASCII Equivalent	Windows OS File Extension
Microsoft Excel spreadsheet	D0 CF 11 E0 A1 B1 1A E1	ÐÏ.à¡±.á	XLS
Microsoft Excel Open XML document	50 4B 03 04 14 00 06 00	PK.....	XLSX
XML file	FF FE 3C 00 3F 00 78 00 6D 00 6C 00 20 00 76 00 65 00 72 00 73 00 69 00 6F 00 6E 00 3D 0	? x m l v e r s i o n =	XML
JPEG image file	FF D8 FF E0 xx xx 4A 46	ÿØÿà..JF	JPG/JPEG
	49 46 00		
Digital camera JPG using Exchangeable Image File Format (EXIF)	FF D8 FF E1 xx xx 45 78 69 66 00	ÿØÿá..Ex	JPG
Still Picture Interchange File Format (SPIFF)	FF D8 FF E8 xx xx 53 50 49 46 46 00	ÿØÿè..SP	JPG
	Samsung D807		JPEG
	0xFF-D8-FF-DB		
	Samsung D500		JPEG
	0xFF-D8-FF-E3		
	Canon EOS-1D		JPEG
	0xFF-D8-FF-E2		
Tagged Image File	49 20 49	I I	TIF/TIFF
Tagged Image File Format file (little endian, i.e., LSB first in the byte; Intel)	49 49 2A 00	II*.	TIF/TIFF
Tagged Image File Format file (big endian, i.e., LSB last in the byte; Motorola)	4D 4D 00 2A	MM.*	TIF/TIFF

File Format	File Signature/ Magic Number	ASCII Equivalent	Windows OS File Extension
Big TIFF files; Tagged Image File Format files >4 GB	4D 4D 00 2B	MM.+	TIF/TIFF
Portable Document Format file	25 50 44 46	%PDF	PDF
MP3 audio file	49 44 33	ID3	MP3
WAVE audio file	57 41 56 45 66 6D 74 20	RIFF....	WAV
Audio Video Interleave file	41 56 49 20 4C 49 53 54	AVI LIST	AVI
MPEG-4 video file	33 67 70 35	3gp5	MP4
MPEG video file	00 00 01 Bx	....	MPG
Windows Media Video file	30 26 B2 75 8E 66 CF 11 A6 D9 00 AA 00 62 CE 6C	0&²u.fÍ.ÌÙ.ª.bÌl	WMV/ASF/WMA
Windows Cabinet file	4D 53 43 46	MSCF	CAB
Windows minidump file	4D 44 4D 50 93 A7	MDMP“§	DMP
Windows 64-bit memory dump	50 41 47 45 44 55 36 34	PAGEDU64	DMP
Windows memory dump	50 41 47 45 44 55 4D 50	PAGEDUMP	DMP
File shortcut	4C 00 00 00 01 14 02 00	L.....	LNK
Windows System file	FF FF FF FF	..ÿÿÿÿ	SYS
Windows Executable file	E8 or E9 or EB	è	SYS
Configuration file	19 0D B7 4D 64 CE 0D D1	·MdÎÑ	CFG
DOS Command file	4D 5A	MZ	COM
Windows Executable file	E8 or E9 or EB	è	COM

File Format	File Signature/ Magic Number	ASCII Equivalent	Windows OS File Extension
Windows Executable file	4D 5A	MZ	EXE
7-Zip compressed file	37 7A BC AF 27 1C	7z¼'.	7Z
PKZIP archive file	50 4B 03 04	PK..	ZIP
ZLock Pro encrypted ZIP	50 4B 03 04 14 00 01 00 63 00 00 00 00 00	PK..... c.....	ZIP
Mac OS X disk image	78	x	DMG
Disc image file	43 44 30 30 31	CD001	ISO
VideoVCD (GNU VCD Imager) file	45 4E 54 52 59 56 43 44 02 00 00 01 02 00 18 58	ENTRYVCD X	VCD
Virtual CD	45 4E 54 52 59 56 43 44 02 00 00 01 02 00 18 58	ENTRYVCD.....X	VCD
BitTorrent file	64 38 3A 61 6E 6E 6F 75 6E 63 65	d8:announce	TORRENT

DEFINISI ANGKA AJAIB*

Angka Ajaib adalah angka yang disematkan pada atau di dekat awal file yang menunjukkan format filenya (yaitu, jenis file itu). Kadang-kadang juga disebut sebagai tanda tangan file. Angka ajaib umumnya tidak terlihat oleh pengguna. Namun, mereka dapat dengan mudah dilihat dengan menggunakan editor HEX, yang merupakan program khusus yang menampilkan dan memungkinkan modifikasi setiap byte dalam file. Untuk format file umum, angka dengan mudah mewakili nama jenis file. Jadi, misalnya, angka ajaib untuk file gambar yang sesuai dengan format GIF87a yang banyak digunakan dalam heksadesimal (yaitu, basis 16) adalah 0x474946383761, yang bila diubah menjadi ASCII adalah GIF87a. ASCII adalah standar de facto yang digunakan oleh komputer dan peralatan komunikasi untuk pengkodean karakter (yaitu, menghubungkan karakter abjad dan lainnya dengan angka).

Demikian pula, angka ajaib untuk file gambar yang memiliki format GIF89a yang diperkenalkan selanjutnya adalah 0x474946383961. Untuk kedua jenis file GIF (Graphic Interchange Format), angka ajaib menempati enam byte pertama file. Mereka kemudian diikuti oleh informasi umum tambahan (yaitu, Metadata) tentang file tersebut. Demikian pula, angka ajaib yang umum digunakan untuk file gambar JPEG (Joint Photographic Experts Group) adalah 0x4A464946, yang setara dengan ASCII dari JFIF (JPEG File Interchange Format). Namun, angka ajaib JPEG bukanlah byte pertama dalam file; sebaliknya, mereka mulai dengan byte ketujuh. Contoh tambahan termasuk 0x4D546864 untuk file MIDI (Musical Instrument Digital Interface) dan 0x425a6831415925 untuk file terkompresi bzip2.

Angka ajaib tidak selalu setara ASCII dengan nama format file, atau bahkan yang serupa. Misalnya, dalam beberapa jenis file, mereka mewakili nama atau inisial pengembang format file tersebut. Selain itu, setidaknya dalam satu jenis file, angka ajaib mewakili hari lahir

pengembang format tersebut. Berbagai program menggunakan angka ajaib untuk menentukan jenis file. Diantaranya adalah program baris perintah (yaitu, mode semua teks) bernama file, yang tujuan utamanya adalah menentukan jenis file. Meskipun bisa berguna, angka ajaib tidak selalu cukup untuk menentukan jenis file. Alasan utamanya adalah beberapa jenis file tidak memiliki angka ajaib, terutama file teks biasa, yang mencakup file HTML (hypertext markup language), XHTML (extensible HTML), dan XML (extensible markup language) serta kode sumber.

Untungnya, ada juga cara lain yang bisa digunakan program untuk menentukan tipe file. Salah satunya adalah dengan melihat kumpulan karakter file (mis., ASCII) untuk melihat apakah itu file teks biasa. Jika ditentukan bahwa file adalah file teks biasa, maka sering kali dimungkinkan untuk mengkategorikannya lebih lanjut berdasarkan awal teks, seperti <htMl> untuk file HTML dan #! (yang disebut shebang) untuk file skrip (yaitu, program singkat). Cara lain untuk menentukan tipe file adalah melalui penggunaan ekstensi nama file (mis., .exe, .htMl, dan .jpg), yang diperlukan di berbagai sistem operasi Microsoft tetapi hanya sebagian kecil di Linux dan Unix lainnya. seperti sistem operasi. Namun, pendekatan ini memiliki kelemahan yaitu relatif mudah bagi pengguna untuk mengubah atau menghapus ekstensi secara tidak sengaja, sehingga sulit untuk menentukan jenis file dan menggunakan file tersebut.

Cara lain yang mungkin dalam kasus beberapa sistem file yang umum digunakan adalah melalui penggunaan informasi tipe file yang disematkan di setiap metadata file. Dalam sistem operasi mirip-Unix, metadata seperti itu terkandung dalam inode, yang merupakan struktur data (yaitu, cara efisien untuk menyimpan informasi) yang menyimpan semua informasi tentang file kecuali nama dan data aktualnya. Angka ajaib disebut sebagai Sihir karena tujuan dan signifikansi nilainya tidak terlihat tanpa pengetahuan tambahan. Istilah Angka Ajaib juga digunakan dalam pemrograman untuk merujuk pada konstanta yang digunakan untuk beberapa tujuan tertentu tetapi keberadaan atau nilainya tidak dapat dijelaskan tanpa informasi tambahan.

HEADER DOKUMEN Gabungan*

512 byte pertama dari file mungkin terlihat seperti dibawah ini:

00000000_H	D0 CF 11 E0 A1 B1 1A E1 00 00 00 00 00 00 00 00
00000010_H	00 00 00 00 00 00 00 00 3B 00 03 00 FE FF 09 00
00000020_H	06 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00
00000030_H	0A 00 00 00 00 00 00 00 00 10 00 00 02 00 00 00
00000040_H	01 00 00 00 FE FF FF FF 00 00 00 00 00 00 00 00
00000050_H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000060_H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000070_H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000080_H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000090_H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
$000000A0_H$	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

$000000B0_H$	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
$000000C0_H$	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
$000000D0_H$	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
$000000E0_H$	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
$000000F0_H$	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000100_H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000110_H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000120_H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000130_H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000140_H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000150_H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000160_H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000170_H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000180_H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000190_H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
$000001A0_H$	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

Meneliti detail dari Compound Document Header ini mengungkapkan hal berikut: delapan (8) byte yang berisi pengenal file dokumen compound tetap

00000000_H	D0 CF 11 E0 A1 B1 1A E1 00 00 00 00 00 00 00 00

Enam belas (16) byte berisi pengenal unik, diikuti oleh empat (4) byte berisi nomor revisi dan nomor versi. Pengidentifikasi Unik, Nomor Revisi, dan Nomor Versi

00000000_H	D0 CF 11 E0 A1 B1 1A E1 00 00 00 00 00 00 00 00
00000010_H	00 00 00 00 00 00 00 00 3B 00 03 00 FE FF 09 00

Dua (2) byte berisi pengidentifikasi urutan byte. Itu harus selalu terdiri dari urutan byte FEH FFH

00000010_H	00 00 00 00 00 00 00 00 3B 00 03 00 FE FF 09 00

Dua (2) byte berisi ukuran sektor, dua (2) byte berisi ukuran sektor pendek. Ukuran sektor adalah 512 byte, dan ukuran sektor pendek di sini adalah 64 byte

00000010_H	00 00 00 00 00 00 00 00 3B 00 03 00 FE FF 09 00
00000020_H	06 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00

Sepuluh (10) byte tanpa data yang valid dapat diabaikan (Tabel 4D.6).

00000020_H	06 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00

Empat (4) byte berisi jumlah sektor yang digunakan oleh tabel alokasi sektor (SAT). SAT hanya menggunakan satu sektor di sini.

00000020_H	06 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00

Empat (4) byte berisi SecID dari sektor pertama yang digunakan oleh direktori.

Direktori dimulai pada sektor 10 di sini. SecID dari Sektor Pertama yang Digunakan oleh Direktori

00000030_H 0A 00 00 00 00 00 00 00 10 00 00 02 00 00 00

Empat (4) byte tanpa data yang valid dapat. Byte tanpa Data Valid

00000030_H 0A 00 00 00 00 00 00 00 10 00 00 02 00 00 00

Empat (4) byte berisi ukuran minimum aliran standar. Ukuran ini adalah 00001000H = 4096 byte di sini

00000030_H 0A 00 00 00 00 00 00 00 10 00 00 02 00 00 00

Empat (4) byte berisi SecID dari sektor pertama dari tabel alokasi sektor pendek

00000030_H 0A 00 00 00 00 00 00 00 10 00 00 02 00 00 00

Empat (4) byte berisi jumlah sektor yang digunakan oleh SSAT. Dalam contoh ini, SSAT dimulai pada sektor 2 dan menggunakan satu sektor

00000040_H 01 00 00 00 FE FF FF FF 00 00 00 00 00 00 00 00

Empat (4) byte berisi SecID sektor pertama dari tabel alokasi sektor master, diikuti oleh empat (4) byte berisi jumlah sektor yang digunakan oleh MSAT. SecID di sini adalah –2, yang menyatakan bahwa tidak ada MSAT yang diperluas dalam file ini. SecID dari Sektor Pertama dari Tabel Alokasi Sektor Master, Diikuti oleh Empat (4) Byte yang Berisi Jumlah Sektor yang Digunakan oleh MSAT

00000040_H 01 00 00 00 FE FF FF FF 00 00 00 00 00 00 00 00

436 byte berisi 109 SecID pertama dari MSAT. Hanya SecID pertama yang valid, karena SAT hanya menggunakan satu sektor (lihat sebelumnya).

Oleh karena itu, semua SecID yang tersisa diatur ke SecID Gratis khusus dengan nilai –1.
Satu-satunya sektor yang digunakan oleh SAT adalah sektor 0. 436 Byte Berisi 109 Detik Pertama dari MSAT

00000040_H 01 00 00 00 FE FF FF FF 00 00 00 00 00 00 00 00

00000050_H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

00000060_H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

00000070H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

00000080H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

00000090H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

000000A0H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

000000B0H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

000000C0H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000D0H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000E0H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000F0H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000100H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000110H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000120H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000130H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000140H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000150H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000160H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000170H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000180H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000190H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000001A0H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000001B0H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000001C0H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000001D0H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000001E0H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000001F0H	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

BAB 5
PROSES BOOT DAN MASTER BOOT RECORD (MBR)

Ini menjadi lebih jelas saat kami menjelajahi file kompleks termasuk file gabungan dan terkompresi, file yang berisi lebih banyak data yang tidak terbaca dalam kode berbasis teks, khususnya petunjuk tentang cara menyusun file kompleks. Dalam menjelaskan struktur file ini, kami mencatat bahwa file ini perlu "dipasang" agar data dapat "diekstraksi" atau "dirakit". Seperti yang telah kita bahas sebelumnya, mount file adalah proses membuat file siap digunakan oleh software yang kompatibel. Proses yang dijelaskan dengan kata "mount" mungkin mendapatkan namanya dari proses serupa yang terjadi pada skala yang lebih besar dan menyeluruh dengan sistem operasi komputer. Proses pemasangan file "di seluruh sistem" ini harus terjadi agar setiap data yang terkandung dalam hard drive dapat diakses, dan ditindaklanjuti.

Mounting adalah proses mengambil data mentah yang terdapat pada hard drive atau media penyimpanan lain dan membuatnya dapat diakses, dibaca, dan informasi yang dapat digunakan. Intinya, ini adalah proses mengambil 0 dan 1 yang disimpan secara magnetis yang dapat dimengerti oleh mesin dan mengubahnya menjadi folder dan file yang dapat dimengerti oleh operator manusia mesin. Proses pemasangan inilah yang mengidentifikasi atau menentukan batasan data komputer atau sistem file.

Komputer harus beroperasi (dinyalakan dan dijalankan) agar dapat memasang sistem file. Terlepas dari apakah itu sistem file dari hard drive utama, sistem file yang terdapat pada CD, atau sistem file dari hard drive USB eksternal, komputer harus hidup dan berjalan agar dapat memasang sistem file. Komputer saat pertama kali dihidupkan perlu memasang sistem file "primer" terlebih dahulu, sistem file yang berisi sistem operasi yang dengannya komputer dapat dioperasikan. Karena itu, dan untuk lebih memahami sistem file dan strukturnya, sebaiknya mulai dari awal.

Sebelum komputer dapat memasang sistem filenya sendiri, komputer harus dihidupkan. Proses menyalakan komputer seringkali disebut sebagai "boot up". Proses “booting-up” ini dan Master Boot Record (MBR) yang terkait, serta kepentingannya dalam memahami esensi proses forensik dunia maya, adalah pokok bahasan bab ini. Kadang-kadang dalam bab ini kami menjelajah secara khusus ke Basic Input Output System (BIOS) karena berkaitan dengan sistem berbasis PC Windows/IBM. Faktanya, istilah BIOS awalnya digunakan untuk menjelaskan program startup sistem berbasis PC IBM. Sejak itu mengadopsi konotasi yang lebih luas, nonvendor-spesifik. Semua model komputer, dari yang kompatibel dengan IBM hingga Unix hingga MAC, memiliki fungsi tipe BIOS atau BIOS. Semua sistem harus boot dan semua sistem memiliki sistem file. Proses boot seperti yang kami uraikan dalam bab ini terkadang menargetkan mesin Windows yang kompatibel dengan IBM, tetapi konsep proses boot sama di sebagian besar sistem operasi. Tujuan di Bab 5 adalah untuk mengeksplorasi proses booting, BIOS sebagai elemen dalam proses booting tersebut, dan keseluruhan hubungan serta relevansinya dengan forensik dunia maya.

Pembaca, menyadari bahwa ada banyak sistem operasi dan dengan demikian berbagai proses booting, didorong untuk mencari dokumen vendor tertentu atau khusus, teks fokus tunggal yang menangani sistem operasi yang terkait dengan komputer yang diperiksa, dan menggunakan materi tersebut. disajikan dalam bab ini untuk menetapkan dasar bagi pemahaman menyeluruh tentang proses boot, BIOS, dan keterkaitannya dengan proses forensik dunia maya.

5.1 BOOTING

Proses menghidupkan sistem, terlepas dari membalik sakelar daya ke posisi “on” dan mengirimkan daya listrik ke mesin, sering disebut sebagai “booting”. "Mem-boot komputer" adalah frasa umum yang digunakan untuk menggambarkan proses di mana kode yang diperlukan untuk menghidupkan komputer dimulai. Karena komputer secara logis dalam mode "mati" sebelum dihidupkan, proses start-up pada awalnya dibandingkan dengan menarik diri sendiri dengan tali sepatu sendiri, ergo istilah "boot" —menyalakan mesin dari keadaan mati . Dengan demikian, kode yang memungkinkan komputer untuk menarik dirinya sendiri (yaitu, untuk memulai sendiri), sejak itu mempertahankan nama ikonik "kode bootstrap." Kode bootstrap pada dasarnya adalah katalisator, sebuah program kecil yang digunakan untuk memulai atau memulai program yang lebih besar. Konsep dan detail sistem file dan hubungannya dengan proses forensik dunia maya dibahas di bab selanjutnya. Untuk saat ini, mari kita kembali ke awal dan pemeriksaan proses booting kita.

Fungsi Utama Proses Boot

Power On Self Test atau POST, singkatnya, adalah program diagnostik mandiri yang digunakan untuk melakukan tes tunggal CPU, RAM, dan berbagai perangkat input/output (I/O). Pembaca harus menyadari bahwa proses booting tidak harus langsung mendahului pemasangan sistem file.

Sistem file dapat dipasang secara terpisah dari proses boot dan sistem file dapat berada di tempat lain selain hard drive yang dapat di-boot. Sistem file dapat ada di drive USB eksternal, area lain dari hard drive, dan CD. Sistem file dapat dipasang dengan baik setelah sistem "di-boot." Ini dapat dicapai melalui berbagai perangkat lunak, termasuk namun tidak terbatas pada perangkat lunak forensik tertentu.

Jika sistem file tidak tergantung pada proses booting, mengapa menutup proses booting di sini? Untuk menjelaskan sistem file dengan baik, yang terbaik adalah menjelaskan di mana sistem file berada: dalam partisi dan volume. Untuk menjelaskan partisi dan volume, pertama-tama kita perlu menjelaskan bagaimana partisi dan volume "dipasang" atau "di-boot".

Perangkat keras komputer diuji untuk memastikan perangkat keras berfungsi dengan baik sebelum proses loading sistem operasi diinisialisasi. POST dilakukan saat startup saat komputer pertama kali dihidupkan dan disimpan di ROM BIOS.

BIOS

BIOS menyiapkan sistem untuk "keadaan yang diketahui", sehingga perangkat lunak yang disimpan pada media yang kompatibel dapat dimuat, dijalankan, dan diberikan kendali

atas PC. Seperti yang dijelaskan sebelumnya, istilah BIOS awalnya diciptakan untuk menggambarkan firmware pemuatan boot pada komputer yang kompatibel dengan IBM PC. Namun, akronim tersebut memiliki arti yang lebih luas, dan sekarang digunakan untuk menjelaskan firmware pemuatan boot dari Sistem Operasi lain yang tidak kompatibel dengan IBM. Fungsi utamanya meliputi:

- Menghitung, menguji, dan menginisialisasi perangkat periferal (keyboard, mouse, disk drive, printer, kartu video, dll.).
- Memuat sistem operasi (primary file system) ke dalam memori utama.

Setelah start-up, program BIOS dapat mengelola aliran data antara sistem operasi (OS) dan periferal, sehingga baik OS maupun program aplikasi tidak perlu mengetahui rincian periferal (seperti alamat perangkat keras). Memuat firmware ini, sebagian, diperlukan untuk menyiapkan mesin ke status yang diketahui, status kesiapan operasional, sehingga perangkat lunak yang disimpan pada media yang kompatibel dapat dimuat, dijalankan, dan diberi kendali atas PC. Setelah sistem operasi dimuat ke dalam memori, ada potensi data untuk disimpan, disimpan, diubah, atau diubah. Pada titik ini dalam proses boot, jika disk tidak dilindungi dari penulisan, maka file yang sedang diakses pada sistem file tersebut dapat dan akan diubah. Sangatlah penting untuk memahami bahwa ketika file diakses, perubahan terjadi; Metadata seperti waktu diakses dan waktu yang dimodifikasi diubah.

Menu Pengaturan BIOS/CMOS

Pengaturan BIOS adalah program yang digunakan untuk menampilkan dan mengedit pengaturan yang dapat dikonfigurasi pengguna di BIOS PC. Pada PC sebelumnya, pengguna harus mengubah pengaturan saat drive baru ditambahkan, tetapi fitur deteksi otomatis ditambahkan kemudian. Meskipun banyak pengaturan yang cukup misterius dan hanya diubah oleh teknisi berpengalaman, pengguna mungkin ingin mengubah urutan boot PC mereka. Setup BIOS juga disebut "CMOS setup" atau "CMOS RAM," karena pengaturan pengguna pada awalnya diadakan di bank MeMory CMOS kecil yang didukung baterai yang merupakan bagian dari sirkuit jam real-time PC. Selanjutnya, lebih banyak pengaturan konfigurasi pengguna disimpan di memori flash BIOS.

Sebuah komputer dapat dibuat untuk boot dari hampir semua media yang berlaku seperti floppy, zip disk, CD, DVD, perangkat USB, hard drive eksternal, atau hard drive internal mesin itu sendiri. BIOS menghitung drive ini, dan urutan/urutan boot masing-masing. Dengan kata lain, BIOS menentukan perangkat boot mana yang akan dimuat; ini disebut sebagai "urutan boot pertama".

"Urutan boot pertama" adalah urutan periferal yang digunakan komputer untuk mencari sistem operasi. Jika tidak menemukan OS yang dapat di-boot di perangkat pertama, ia mencari di perangkat kedua dan seterusnya. Meskipun komputer biasanya melakukan booting dari hard disk primer, urutan booting pertama memungkinkannya memuat sistem operasi yang berbeda jika diperlukan dengan menempatkan disk yang dapat di-boot ke dalam drive. Urutan boot pertama dikonfigurasi di BIOS dan dapat diubah sesuai kebutuhan.

Pada masa awal komputer pribadi, floppy disk dipilih sebagai perangkat pertama dan hard disk kedua. Selanjutnya, CD-ROM biasanya dipilih menjadi urutan pertama, diikuti oleh floppy dan hard disk. Hanya satu perangkat yang akan diakses dan dimuat dalam proses boot-

up ini. Namun, BIOS dapat diakses dan perubahan pada BIOS dapat dilakukan. Perubahan ini terjadi pada chip ROM dan sebagai hasilnya, tidak menyebabkan apa pun ditulis ke hard drive, tempat bukti kami berada. Pemahaman tentang BIOS penting dalam dunia forensik, karena mungkin ada kebutuhan untuk memasuki BIOS sebagai bagian dari penyelidikan forensik dunia maya. Beberapa alasan termasuk:

- Verifikasi dan validasi jam sistem. Jam sistem di sebagian besar mesin tidak 100 persen akurat; jam sistem berpotensi kehilangan beberapa menit setiap hari, atau berhenti menambah waktu saat sistem dimatikan.

  Penyebab paling umum dari masalah ini adalah baterai CMOS, yang juga mencadangkan tanggal dan waktu sehingga tidak hilang saat mesin dimatikan. Baterai CMOS yang lemah dapat menyebabkan masalah dengan jam waktu nyata, meskipun baterai tidak cukup lemah untuk menyebabkan hilangnya pengaturan BIOS. Beberapa motherboard tampaknya menonaktifkan jam sebagai tindakan penghematan daya saat voltase baterai rendah. Tentu saja, terkadang masalah dengan jam hanya karena tidak akurat.
- Seperti yang dijelaskan di bagian menu pengaturan BIOS, komputer mungkin perlu di-boot ke disk boot yang berdiri sendiri. Masalah teknis dengan integritas hard drive komputer mungkin memerlukan perangkat boot alternatif, seperti disk perbaikan darurat. Mungkin alasan paling penting untuk mengubah urutan booting adalah untuk mencegah komputer mem-boot drive bukti. Mengapa tidak boot ke drive bukti? Baca terus!

5.2 PENCITRAAN FORENSIK DAN PENGUMPULAN BUKTI

Penyelidik forensik digital, seperti penyelidik mana pun, kadang-kadang akan bertanggung jawab untuk mengumpulkan dan menangkap bukti. Memahami bahwa data dapat ditulis ke hard drive (bukti) selama proses boot sangat penting karena proses ini mengubah bukti. Mengetahui kapan dan bagaimana data diubah pada bukti (hard drive atau lainnya) tidak hanya penting saat menyelidiki bukti, tetapi juga penting saat memperoleh bukti.

Seperti yang telah kita diskusikan, selama proses boot data sistem file utama (atau partisi), dalam banyak kasus, ditulis ke hard drive, sehingga tanggal diubah dan file ditulis dan diubah. Sangat penting untuk penyelidikan yang baik untuk tidak mengubah bukti yang telah dipercayakan kepada Anda untuk dicitrakan dengan cara yang sehat secara forensik. Mem-boot komputer dapat mencemari integritas data yang terkandung dalam bukti (hard drive). Ini akan dianalogikan dengan seorang detektif pembunuhan yang menginjak-injak cipratan darah di TKP. Kalaupun sang detektif bisa menghilangkan jejak kakinya, paling tidak kualitas dan kompetensi pekerjaannya akan dipertanyakan.

Pengacara pembela yang baik dapat dengan cepat dan mudah mendiskreditkan penyelidikan. Pertanyaan mungkin muncul, seperti, "apa lagi yang Anda ubah di TKP?" dan "dengan pekerjaan ceroboh seperti itu, apakah Anda tahu bukti lain apa yang mungkin telah Anda ubah?" Seperti yang dapat Anda bayangkan, meskipun kesalahan tersebut dapat dijelaskan oleh penyelidik, kesalahan tersebut dapat menimbulkan efek negatif yang drastis

dan tidak hanya mencegah bukti kritis untuk diakui sebagai bukti, tetapi juga berpotensi merusak karier penyelidik.

Jadi dalam upaya untuk menghindari penulisan data ke suatu barang bukti, seorang penyelidik biasanya akan menghubungkan pemblokir penulisan langsung ke hard drive barang bukti. Pemblokir tulis akan memungkinkan drive untuk dihidupkan dan disalin, tetapi akan memblokir upaya penulisan apa pun (secara tidak sengaja atau sengaja) langsung ke drive bukti. Contoh dari beberapa teknik pemblokiran tulis, meskipun mungkin agak kuno untuk status teknologi saat ini meliputi:

- Kaset memiliki tab plastik yang dapat dipatahkan untuk memblokir fitur tulis atau rekam pada kaset tersebut.
- Floppy disk berukuran 3½ inci dengan tab plastik kecil yang dapat diposisikan untuk menulis blok.
- Pemutar CD tanpa kemampuan membakar adalah pemblokir tulis, karena tidak dapat menulis atau membakar.
- LP dan pemutar rekaman.

Penyelidik forensik dunia maya pada dasarnya melakukan hal yang sama ketika berhadapan dengan hard drive. Satu-satunya perbedaan adalah peralatan pemblokiran tulis tidak ditemukan di toko elektronik sudut dan bisa jadi mahal. Penyelidik pada dasarnya menghubungkan salah satu ujung pemblokir tulis ke Sumber (hard drive bukti dengan asumsi penghapusan dari sistem) dan ujung lainnya ke Target (area penyimpanan — mis., hard drive, CD, DVD, perangkat USB, dll.).

Write blocker kemudian memungkinkan hard drive untuk dinyalakan dan dibaca, tetapi tidak ditulis. Data, biasanya pada tingkat bit sebagai biner, kemudian "disalin" dari drive bukti ke area penyimpanan. Dalam lingkungan yang sempurna secara teoritis, hard drive dihapus begitu saja dari komputer dan bukti data laten potensialnya diperoleh secara independen, seperti yang dijelaskan.

Namun, terkadang hal ini tidak mungkin atau tidak praktis, seperti dalam kasus di mana hard drive tertanam begitu dalam di laptop sehingga melepasnya akan memerlukan pembongkaran laptop, seperti pada beberapa MAC atau Sony VAIO. Kasus lain yang membuat penghapusan hard drive menjadi alternatif yang kurang layak meliputi:

1. Komputer “canggih” yang berisi teknologi hard drive baru dengan adaptor bus canggih (port yang menerima kabel data) yang tidak memiliki adaptor pemblokir tulis.
2. Komputer lama dengan hard drive yang adaptornya mungkin tidak ada.
3. Server mungkin memiliki beberapa hard disk drive (HDD) yang dikonfigurasi sebagai satu hard drive logis (seperti dalam larik RAID). Akan lebih mudah untuk mendapatkan ini sebagai satu drive (secara logis) dibandingkan dengan menghapus dan memperoleh setiap drive dalam array satu per satu, terutama jika adaptor untuk HDD khusus tidak tersedia.

Apa pun alasannya, melepas hard drive dan menyambungkannya ke pemblokir tulis untuk pencitraan mungkin bukan pilihan. Jika demikian, komputer harus dihidupkan, tanpa mem-boot ke sistem operasi hard drive sehingga menghindari penghancuran atau pencemaran

bukti potensial. Memahami BIOS mungkin sangat bermanfaat jika tidak penting saat mengumpulkan bukti ini.

Urutan boot diubah sehingga BIOS menyerahkan kendali ke sistem operasi penyelidik, yang ditemukan di floppy disk, USB, atau compact disc, dan bukan ke sistem operasi yang ditemukan di hard disk, yaitu buktinya. Dengan kata lain, ini memungkinkan sistem untuk melanjutkan booting melalui sistem operasi atau utilitas yang ada di dalam boot disk, tidak bergantung pada penggunaan sistem operasi internal yang disimpan di hard drive internal mesin. Ini juga memungkinkan penyelidik forensik dunia maya mengakses data yang ada di sistem tanpa mengubah bukti yang ada di dalam hard drive.

Ringkasan Bios

Beberapa fungsi atau komponen BIOS yang ditemukan di banyak sistem antara lain:

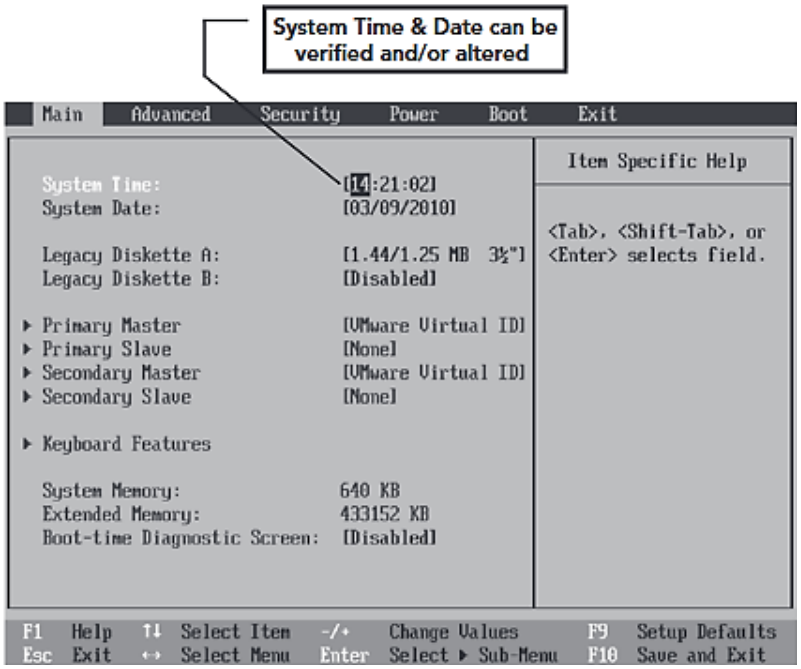
- POS. Test perangkat keras komputer, memastikan perangkat keras berfungsi dengan baik sebelum memulai proses loading sistem operasi.
- Pemuat Bootstrap. Proses menemukan sistem operasi. Jika sistem operasi yang mampu ditemukan, BIOS akan memberikan kontrol padanya.
- BIOS. Perangkat Lunak/Driver yang menghubungkan antara sistem operasi dan perangkat keras Anda. Saat menjalankan DOS atau Windows yang Anda gunakan lengkap dukungan BIOS.
- Pengaturan BIOS/CMOS. Program konfigurasi yang memungkinkan Anda mengonfigurasi pengaturan perangkat keras termasuk pengaturan sistem seperti kata sandi komputer, waktu, dan tanggal.[2]

5.3 UTILITAS PENGATURAN BIOS

BIOS diakses saat startup. Segera setelah PC dihidupkan, pesan teks singkat biasanya lewat dengan sangat cepat di layar yang menunjukkan tombol mana yang harus ditekan (biasanya tombol DEL, F1, atau F2). Sebuah prompt akan muncul di layar yang menyatakan sesuatu yang mirip dengan yang ditunjukkan pada Gambar 5.1.

Gambar 5.1 Menyiapkan BIOS Phoenix

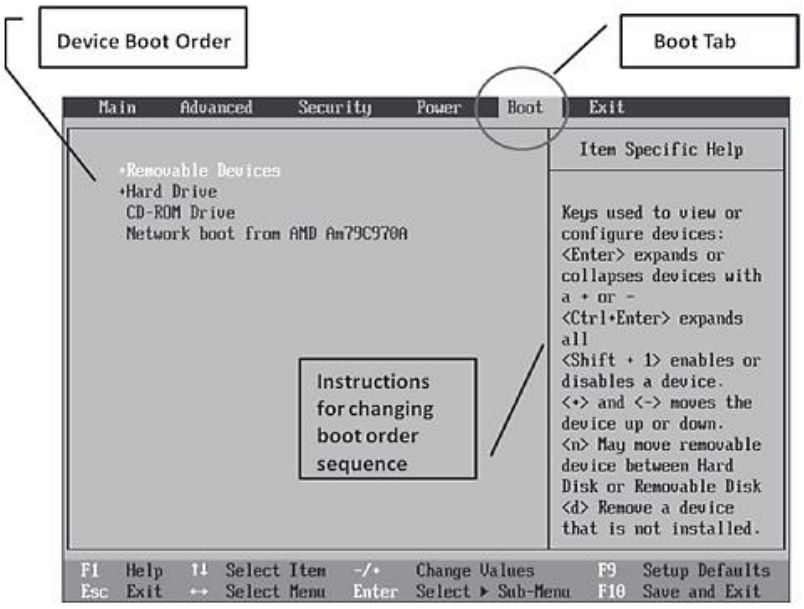
Pada titik ini, menekan F2 untuk sistem khusus yang kompatibel dengan IBM ini akan menghentikan proses booting dan memasukkan "BIOS". Sistem yang berbeda akan memiliki tombol atau penekanan tombol yang berbeda untuk masuk ke "BIOS" masing-masing. Setelah menekan F2 (atau tombol fungsi yang sesuai (misalnya, F12, F8, F10, ESC, atau DEL), Anda akan diperlihatkan layar utama utilitas pengaturan BIOS, yang memungkinkan Anda untuk mengakses setiap bagian lain dari BIOS Gambar 5.2 menunjukkan layar pengaturan BIOS yang khas—Tab Utama, untuk produk BIOS yang populer, PhoenixBIOS Seperti yang dapat kita lihat, waktu dan tanggal sistem dapat diverifikasi dan/atau diubah.



Gambar 5.2 Menu Pengaturan BIOS—Utama

Pada Gambar 5.3, kita melihat tab Boot. Di sinilah urutan perangkat boot dapat diubah. Dalam contoh ini, kita melihat "removable devices" sebagai perangkat booting pertama; hard drive adalah yang kedua dan CD-ROM adalah yang ketiga.

Petunjuk untuk mengubah urutan biasanya terdapat pada tab. Di sini kita dapat melihat bahwa tanda positif atau plus (+) akan menggerakkan perangkat ke atas secara berurutan dan tanda negatif atau minus (-) akan menggerakkan perangkat ke bawah secara berurutan. Jadi, untuk mem-boot sistem ini dari boot disk (CD), CD-ROM perlu dipindahkan ke bagian atas urutan boot. Gambar 5.4 dan 5.5 adalah contoh menu pengaturan BIOS lainnya yang mungkin Anda temui.



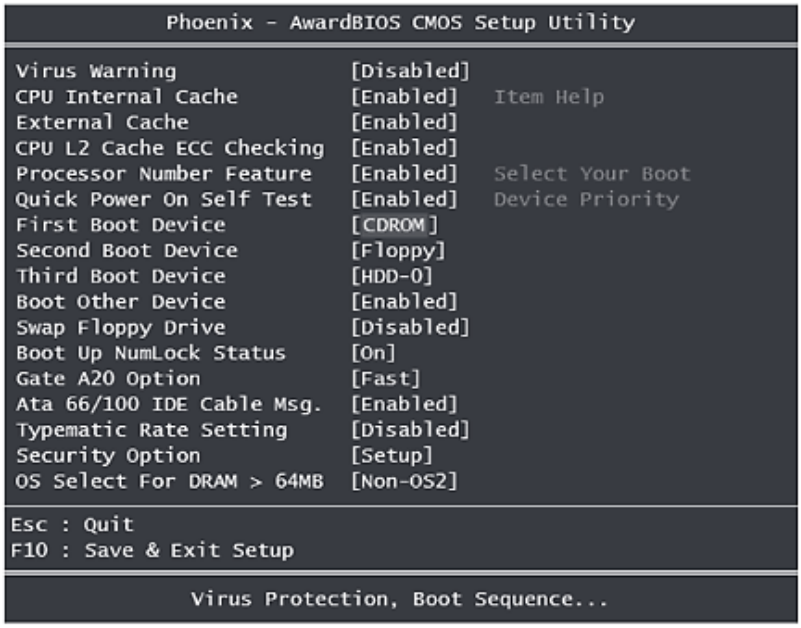
Gambar 5.3 Menu Pengaturan BIOS—Boot

Ada banyak versi dan varietas BIOS, sehingga tangkapan layar BIOS yang ditampilkan di sini mungkin tidak terlihat mirip dengan layar BIOS pada sistem Anda. Selain itu, ada banyak cara untuk masuk ke BIOS—biasanya F2, tetapi tidak selalu! Lakukan penelitian Anda! Berhati-hatilah! Anda tidak ingin melewatkan BIOS dan membiarkan Sistem Operasi melakukan boot, mengakibatkan file diakses dan ditulis dan akhirnya diubah!

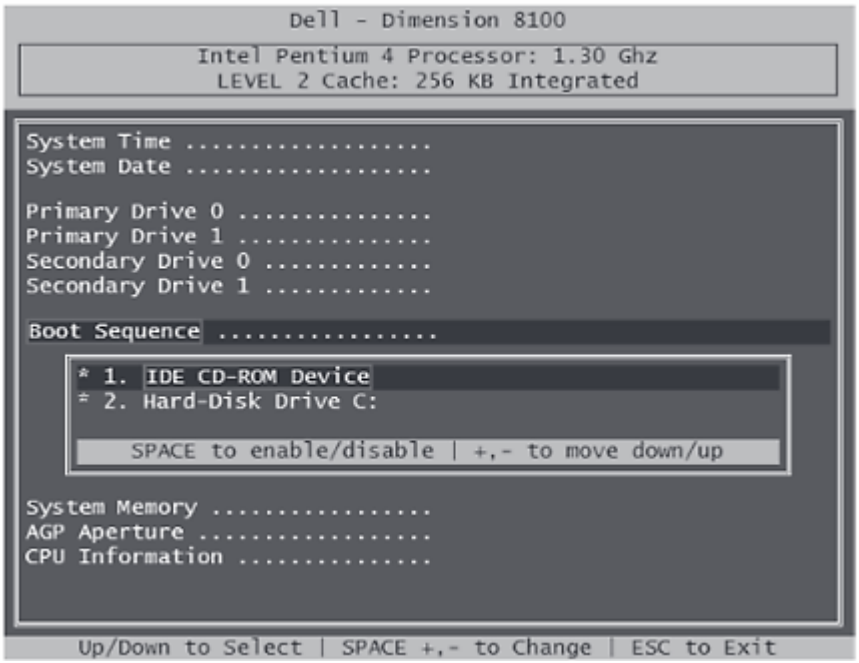
Fase Dua dari Proses Boot

Setelah semua pengujian memastikan bahwa perangkat keras berfungsi dengan baik dan sebelum memulai proses memuat sistem operasi, drive fisik disebutkan satu per satu dan kode boot mencoba menemukan dan memuat utilitas sistem operasi atau perangkat lunak. Setelah ini terjadi, proses booting berhenti dan drive utama aktif. Penulisan dan perubahan pada drive utama mulai terjadi! Ingat, mencegat dan mengakses BIOS akan menghentikan proses boot dan mencegah kode boot mengakses drive aktif (atau utama).

Jika "fase kedua" proses boot diizinkan untuk melanjutkan dengan HDD sebagai perangkat boot utama, perubahan akan dilakukan pada hard drive tersebut, dan hasilnya adalah manipulasi data yang tidak disengaja pada hard drive tersebut, sehingga potensi untuk penghancuran atau pengubahan data, yang menyebabkan hilangnya integritas data sehubungan dengan nilai pembuktian data yang terdapat pada hard drive tersebut. Jika HDD pertama kali dalam urutan booting, maka kode booting akan terlihat terlebih dahulu di sana. Mengetahui bagaimana BIOS dikonfigurasi, dan identifikasi perangkat boot pertama, merupakan informasi penting bagi penyelidik forensik dunia maya, dalam upaya penyelidik untuk mengakses hard drive yang akan diperiksa sambil selalu memastikan integritas data yang dikumpulkan.



Gambar 5.4 Layar Pengaturan BIOS Menampilkan Perangkat Booting Pertama (Perangkat Lunak Phoenix)



Gambar 5.5 Layar Dell Dimension BIOS Setup

5.4 MASTER BOOT RECORD (MBR)

Pada fase kedua booting inilah BIOS yang ada di dalam komputer berbasis Intel ini (atau komputer yang kompatibel dengan IBM) akan memuat sektor pertama hard drive ke dalam memori. Sektor pertama ini disebut Master Boot Record (MBR). Kode boot terlihat pada sektor pertama dari drive default (drive pertama pada daftar di urutan boot di BIOS. Sektor pertama ini berisi MBR atau Master Boot Record. MBR berisi tiga komponen, yang akan kami bahas secara rinci:

1. Sejumlah kecil kode yang dapat dieksekusi disebut kode boot master.
2. Tanda tangan disk.
3. Tabel partisi untuk disk.

Boot loader bekerja dengan mencari partisi aktif di tabel partisi dan memuat sektor pertama di partisi tersebut. Sektor itu dikenal sebagai Partition Boot Record. Partition Boot Record kemudian akan memulai proses memuat kernel sistem operasi. Proses boot (kode) mencari drive yang tersedia (sudah diidentifikasi di BIOS) untuk sistem operasi. Setelah ditemukan, sistem operasi menguji tanda tangan disk (nomor unik pada offset 0x01B8, yang mengidentifikasi disk ke sistem operasi). Dua sektor terakhir dari MBR berisi struktur dua byte yang disebut kata tanda tangan atau penanda akhir sektor, yang selalu disetel ke 0x55AA (HEX 55AA). Kata tanda tangan juga menandai akhir dari catatan boot yang diperpanjang (EBR) dan sektor boot. Untuk drive yang dikonfigurasi/dapat di-boot secara valid, HEX 55AA harus ditemukan di dua byte terakhir dari sektor ini. Kode boot mencari drive yang dapat di-boot, yang diidentifikasi dengan nilai 0x55AA.

Kode boot master melakukan aktivitas berikut:

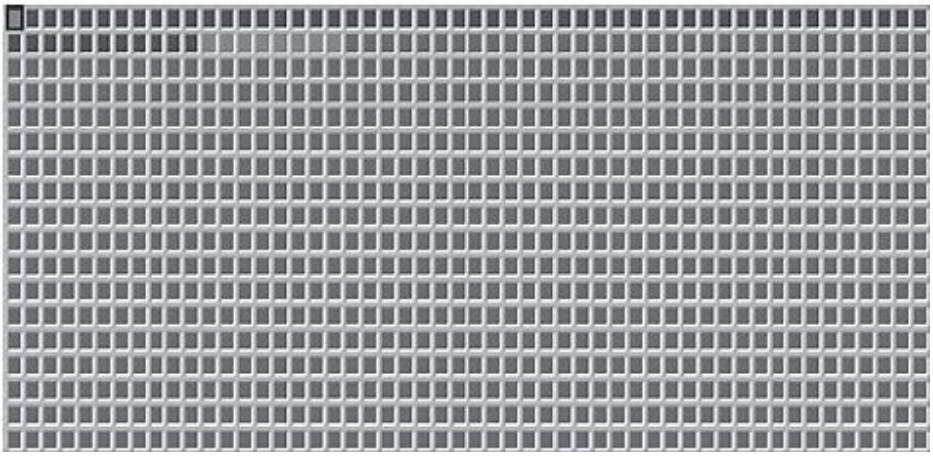
1. Memindai tabel partisi untuk partisi yang aktif.
2. Menemukan sektor awal dari partisi aktif.
3. Memuat salinan sektor boot dari partisi aktif ke dalam memori.

4. Mentransfer kontrol ke kode yang dapat dieksekusi di sektor boot.

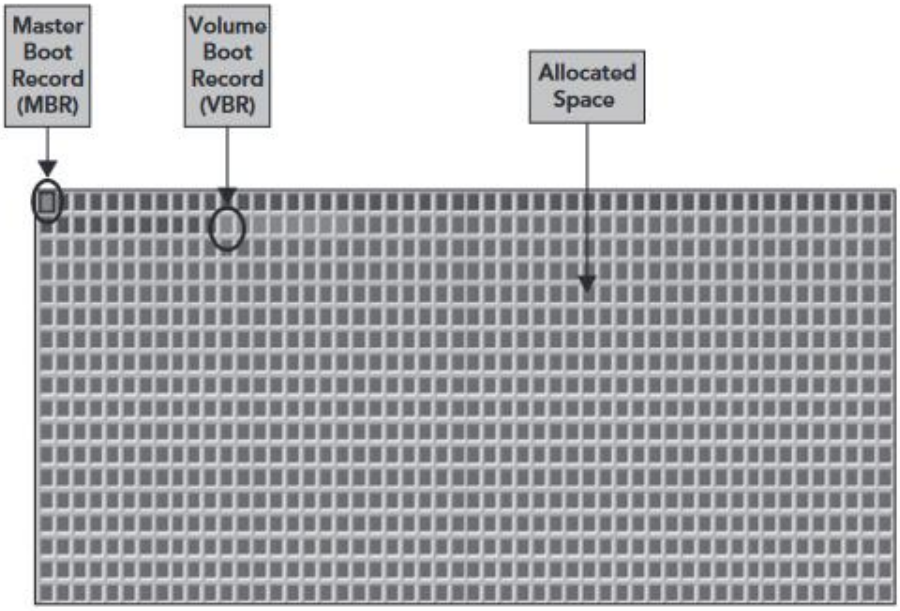
Jika kode boot master tidak dapat menyelesaikan fungsi ini, sistem mungkin menampilkan salah satu pesan kesalahan berikut:

- Tabel partisi tidak valid
- Kesalahan memuat sistem operasi
- Sistem operasi hilang

Gambar 5.6 menunjukkan representasi visual/grafis dari First Sectors dari hard drive fisik seperti yang digambarkan oleh perangkat lunak forensik Encase. Maklum ini hanya representasi visual dari konstruksi fisik hard drive. Setiap kotak mewakili satu sektor. Berapa banyak byte per sektor? Ingat dari diskusi kita di Bab 4, sebuah sektor berisi 512 byte. Gambar 5.7 menunjukkan rincian lebih lanjut dari sektor pertama hard drive fisik, menunjukkan posisi MBR, VoluMe Boot Record (VBR), dan sisa ruang yang dialokasikan pada hard drive.



Gambar 5.6 Representasi Grafis dari Sektor Pertama Hard Drive Fisik



Gambar 5.7 Lokasi MBR dan VBR di Sektor Pertama Hard Drive Fisik

Penting untuk dicatat bahwa setiap kotak yang ditampilkan di sini mewakili satu sektor (512 byte). Ingat, byte tanda tangan MBR adalah dua byte terakhir dari sektor pertama, dan digunakan sebagai validasi sederhana dari konten MBR.

Apa itu Editor HEX?

Konsep tertentu perlu dipahami saat membaca editor HEX:

1. ACSII (dibahas pada Bab 2)
2. HEX (dibahas di Bab 3)
3. Offset (dibahas selanjutnya)

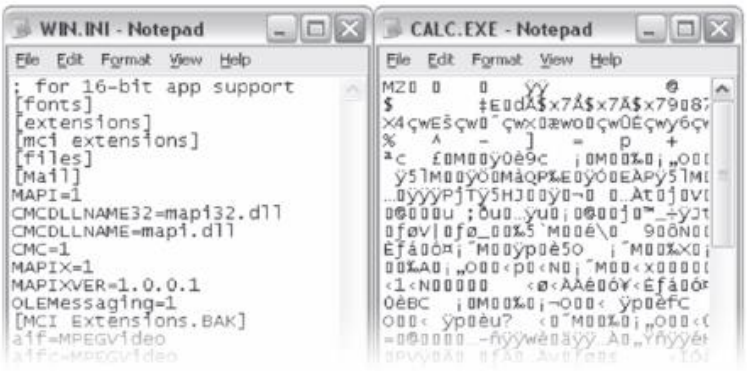
Apa itu Offset?

Dalam ilmu komputer, offset dalam array atau objek struktur data lainnya adalah bilangan bulat yang menunjukkan jarak (perpindahan) dari awal objek ke atas (alamat basis) hingga elemen atau titik tertentu, mungkin dalam objek yang sama. Konsep jarak hanya berlaku jika semua elemen objek memiliki ukuran yang sama (biasanya diberikan dalam byte atau kata). Dalam arti offset (asli) ini, hanya unit alamat dasar, biasanya byte 8-bit, yang digunakan untuk menentukan ukuran offset. Dalam konteks ini offset terkadang disebut alamat relatif. Alamat absolut diturunkan dengan menambahkannya (alamat relatif) ke alamat dasar. Misalnya, pada Gambar 5.8, diberikan array karakter A yang berisi konten abcdef, dapat dikatakan bahwa elemen yang berisi huruf "c" memiliki offset 2 dari elemen yang berisi "a". Editor HEX adalah program komputer yang digunakan untuk melihat dan mengedit file biner. File biner adalah file yang berisi data dalam bentuk yang dapat dibaca mesin (berlawanan dengan file teks yang dapat dibaca oleh manusia; lihat Gambar 5.9).

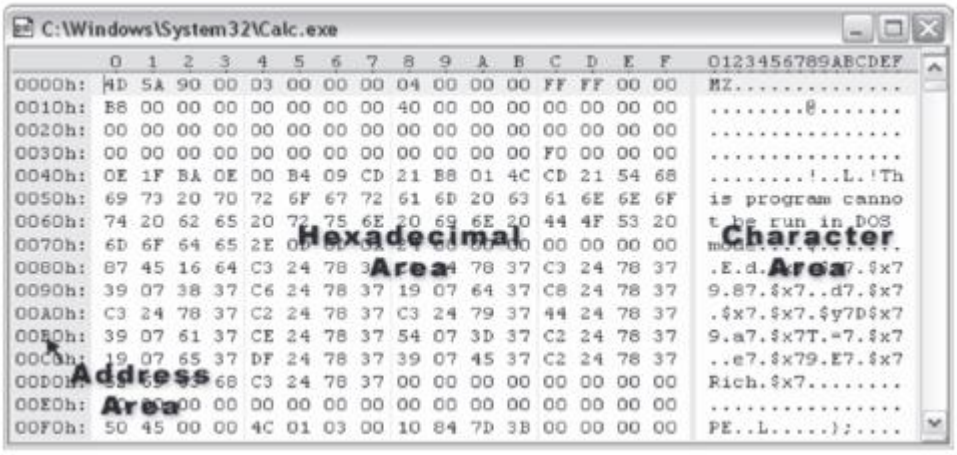
Editor HEX memungkinkan pengeditan konten data mentah suatu file, alih-alih program lain yang mencoba menginterpretasikan data untuk Anda. Karena editor HEX digunakan untuk mengedit file biner, terkadang disebut editor biner atau editor file biner. Jika Anda mengedit file dengan editor HEX, Anda dikatakan mengedit file HEX, dan proses menggunakan editor HEX disebut pengeditan HEX. Tipikal editor HEX memiliki tiga area: area alamat di kiri, area heksadesimal di tengah, dan area karakter di kanan (lihat Gambar 5.10).

a	b	c	d	e	f
0	1	2	3	4	5

Gambar 5.8 Susunan Karakter A Berisi "abcdef"



Gambar 5.9 File Teks yang Dimuat di Notepad di Sebelah Kiri (Dapat Dibaca Manusia) versus File Biner (Dapat Dibaca Mesin) di Sebelah Kanan

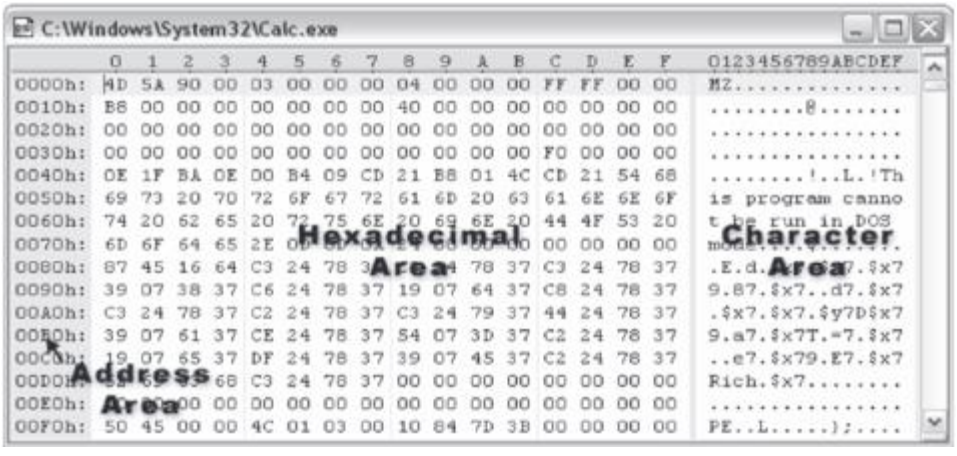


Gambar 5.10 Editor HEX dengan Alamat di Kiri, Area Heksadesimal di Tengah, dan Area Karakter di Kanan

Panel kiri (area alamat) menampilkan offset byte (16 byte/baris), panel tengah (area heksadesimal) menampilkan nilai dua digit yang terdiri dari setiap byte, dan panel kanan (area karakter) menampilkan persamaan ASCII dari setiap byte data.

Editor HEX digunakan terutama karena dua alasan khusus:

1. Menganalisa struktur file. Anda tidak dapat melihat byte yang disimpan dalam file menggunakan aplikasi biasa untuk membukanya. Anda mungkin memerlukan pengetahuan ini untuk menulis aplikasi yang akan menginterpretasikan isi file. (Tujuan penyelidik forensik dunia maya)
2. Mengedit isi file. Ini juga membutuhkan pengetahuan tentang struktur file yang tepat. Jika Anda tidak tahu bagaimana tanda air disimpan dalam file MPEG, misalnya, Anda tidak dapat berbuat apa-apa. (Tujuan programmer)

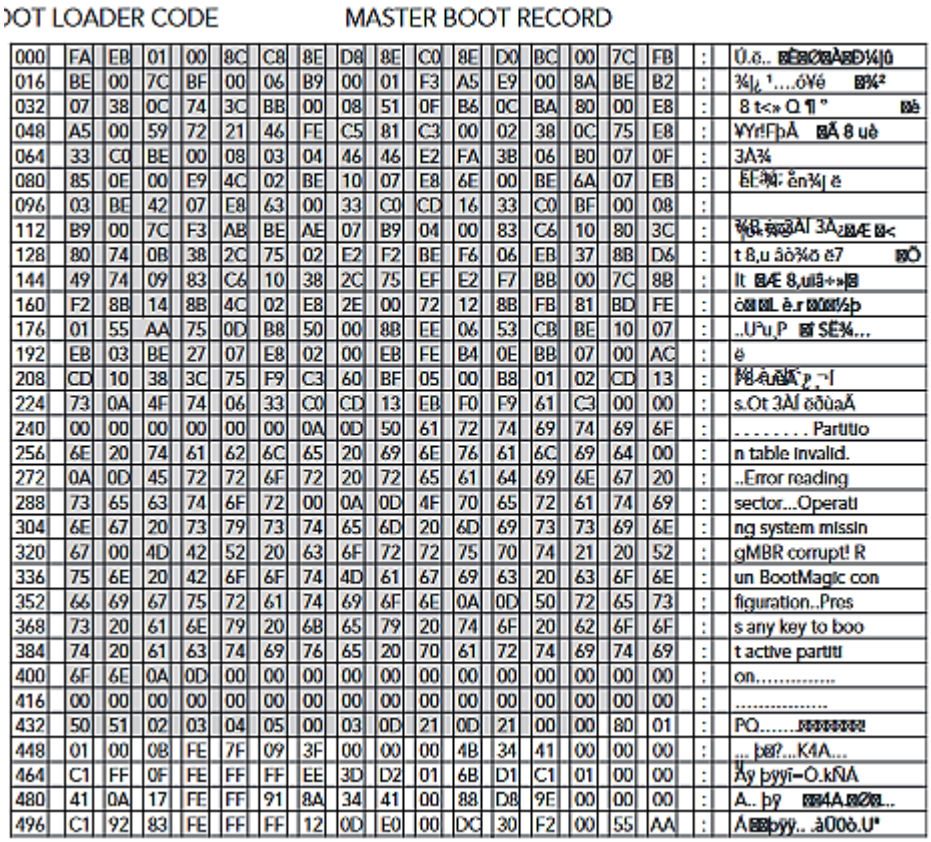


Gambar 5.11 Tata Letak Editor HEX Umum

Pada Gambar 5.11, contoh kita menunjukkan bahwa ada 16 byte di setiap baris, baris pertama berisi byte 0 sampai 15; offset baris kedua adalah 16. Ingat kami sedang mendiskusikan MBR sebelum kami mengambil deviasi singkat untuk melihat lebih dekat pada editor HEX dan

konsep "offset". Mari kita lanjutkan sekarang dengan pembahasan lebih lanjut tentang MBR. Merangkum, MBR berisi tiga komponen utama:

1. Pemuat boot
2. Tabel partisi
3. Byte tanda tangan



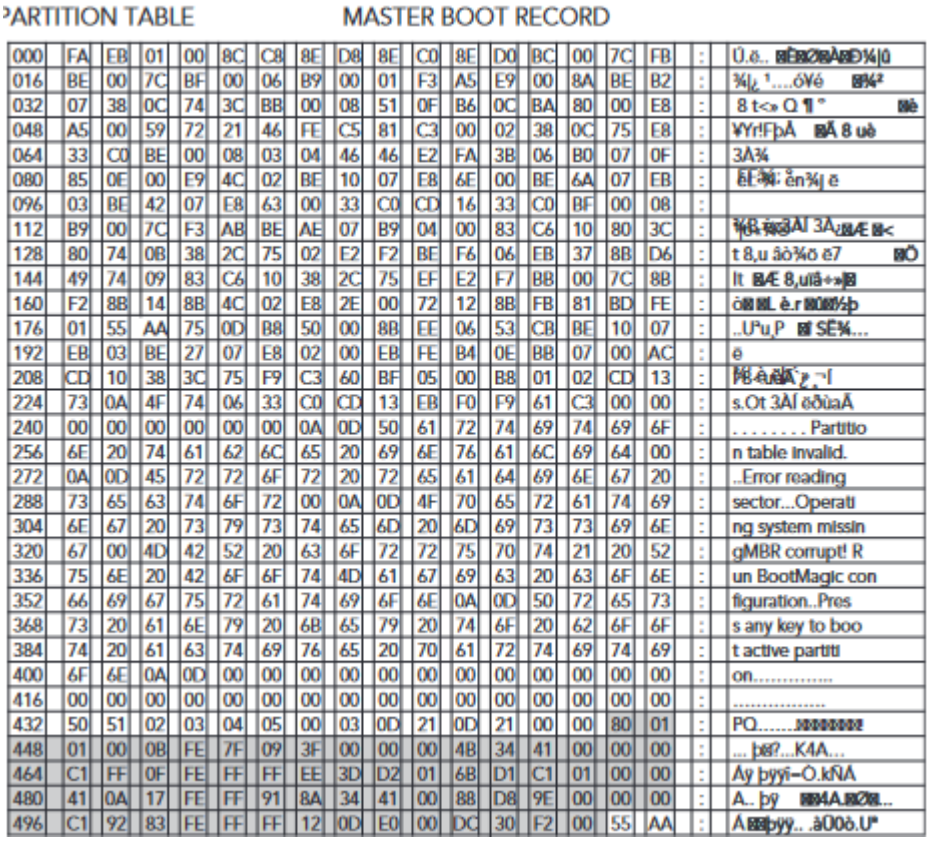
000	FA	EB	01	00	8C	C8	8E	D8	8E	C0	8E	D0	BC	00	7C	FB	:	Ú.ð.. ëÈØÀÐ¼|û
016	BE	00	7C	BF	00	06	B9	00	01	F3	A5	E9	00	8A	BE	B2	:	¾|¿ ¹....ó¥é ¾²
032	07	38	0C	74	3C	BB	00	08	51	0F	B6	0C	BA	80	00	E8	:	8 t<» Q ¶ ° è
048	A5	00	59	72	21	46	FE	C5	81	C3	00	02	38	0C	75	E8	:	¥Yr!FþÅ Ã 8 uè
064	33	C0	BE	00	08	03	04	46	46	E2	FA	3B	06	B0	07	0F	:	3À¾
080	85	0E	00	E9	4C	02	BE	10	07	E8	6E	00	BE	6A	07	EB	:	…é.é L.¾ èn¾j ë
096	03	BE	42	07	E8	63	00	33	C0	CD	16	33	C0	BF	00	08	:	
112	B9	00	7C	F3	AB	BE	AE	07	B9	04	00	83	C6	10	80	3C	:	¹ ó« ®¹ ƃÆ <
128	80	74	0B	38	2C	75	02	E2	F2	BE	F6	06	EB	37	8B	D6	:	t 8,u âò¾ö ë7 ÖÖ
144	49	74	09	83	C6	10	38	2C	75	EF	E2	F7	BB	00	7C	8B	:	It ƃÆ 8,uïâ÷» |ƃ
160	F2	8B	14	8B	4C	02	E8	2E	00	72	12	8B	FB	81	BD	FE	:	ò‹ ‹L è.r ‹û½þ
176	01	55	AA	75	0D	B8	50	00	8B	EE	06	53	CB	BE	10	07	:	..Uªu¸P ‹î SË¾...
192	EB	03	BE	27	07	E8	02	00	EB	FE	B4	0E	BB	07	00	AC	:	ë ¾'.è..ëþ´.» ..¬
208	CD	10	38	3C	75	F9	C3	60	BF	05	00	B8	01	02	CD	13	:	Í.8<uùÃ`¿ ¸ ..Í.
224	73	0A	4F	74	06	33	C0	CD	13	EB	F0	F9	61	C3	00	00	:	s.Ot 3ÀÍ ëðùaÃ..
240	00	00	00	00	00	00	0A	0D	50	61	72	74	69	74	69	6F	:	..... Partitio
256	6E	20	74	61	62	6C	65	20	69	6E	76	61	6C	69	64	00	:	n table invalid.
272	0A	0D	45	72	72	6F	72	20	72	65	61	64	69	6E	67	20	:	..Error reading
288	73	65	63	74	6F	72	00	0A	0D	4F	70	65	72	61	74	69	:	sector...Operati
304	6E	67	20	73	79	73	74	65	6D	20	6D	69	73	73	69	6E	:	ng system missin
320	67	00	4D	42	52	20	63	6F	72	72	75	70	74	21	20	52	:	g.MBR corrupt! R
336	75	6E	20	42	6F	6F	74	4D	61	67	69	63	20	63	6F	6E	:	un BootMagic con
352	66	69	67	75	72	61	74	69	6F	6E	0A	0D	50	72	65	73	:	figuration..Pres
368	73	20	61	6E	79	20	6B	65	79	20	74	6F	20	62	6F	6F	:	s any key to boo
384	74	20	61	63	74	69	76	65	20	70	61	72	74	69	74	69	:	t active partiti
400	6F	6E	0A	0D	00	00	00	00	00	00	00	00	00	00	00	00	:	on..............
416	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	.....
432	50	51	02	03	04	05	00	03	0D	21	0D	21	00	00	80	01	:	PQ.....!.!.....
448	01	00	0B	FE	7F	09	3F	00	00	00	4B	34	41	00	00	00	:	... þ?...K4A....
464	C1	FF	0F	FE	FF	FF	EE	3D	D2	01	6B	D1	C1	01	00	00	:	Áÿ þÿÿî=Ò.kÑÁ..
480	41	0A	17	FE	FF	91	8A	34	41	00	88	D8	9E	00	00	00	:	A.. þÿ ‘Š4A.ˆØž...
496	C1	92	83	FE	FF	FF	12	0D	E0	00	DC	30	F2	00	55	AA	:	Á’ƒþÿÿ.. .à.Ü0ò.Uª

Gambar 5.12 Isi MBR dengan Kode BootMagic PowerQuest di Bagian Boot Loader-nya (berwarna abu-abu, 000–432)

Pemuat boot (yaitu, bootstrap) biasanya memuat sistem operasi utama untuk komputer. Pemuat boot MBR terdiri dari kode yang dimuat BIOS untuk mem-boot sistem operasi. Boot loader bekerja dengan mencari partisi aktif di tabel partisi dan memuat sektor pertama di partisi tersebut. Sektor itu dikenal sebagai Catatan Boot Partisi dan biasanya (tetapi tidak selalu) adalah catatan boot OS. Partition Boot Record kemudian akan memulai proses pemuatan kernel sistem operasi. Gambar 5.12 menampilkan representasi grafis dari MBR, dengan kode boot loader yang disorot dengan warna abu-abu.

Sementara area boot loader selalu 446 byte (Byte Offset 0–445), jumlah byte yang benar-benar digunakan untuk kode boot loader bervariasi menurut program yang diinstal di area ini. MBR yang dibuat dengan FDISK DOS menggunakan program yang lebih kecil di area boot loader sehingga Anda akan melihat lebih banyak 00h byte. Tabel partisi, yang dimulai segera setelah area boot loader (ditampilkan dengan warna abu-abu pada Gambar 5.13) dimulai dengan nilai 0x80 yang mewakili partisi aktif (dapat di-boot). Ini berisi empat

deskriptor yang panjangnya masing-masing 16 byte. Deskriptor mewakili informasi logis yang diperlukan untuk mengakses partisi di drive.



PARTITION TABLE MASTER BOOT RECORD

000	FA	EB	01	00	8C	C8	8E	D8	8E	C0	8E	D0	BC	00	7C	FB	:	Ú.ë..
016	BE	00	7C	BF	00	06	B9	00	01	F3	A5	E9	00	8A	BE	B2	:	¾¿ ¹....óÝé
032	07	38	0C	74	3C	BB	00	08	51	0F	B6	0C	BA	80	00	E8	:	8 t<» Q ¶ °
048	A5	00	59	72	21	46	FE	C5	81	C3	00	02	38	0C	75	E8	:	¥Yr!FþÅ 8 uè
064	33	C0	BE	00	08	03	04	46	46	E2	FA	3B	06	B0	07	0F	:	3À¾
080	85	0E	00	E9	4C	02	BE	10	07	E8	6E	00	BE	6A	07	EB	:	…é L ¾ è n ¾ j ë
096	03	BE	42	07	E8	63	00	33	C0	CD	16	33	C0	BF	00	08	:	
112	B9	00	7C	F3	AB	BE	AE	07	B9	04	00	83	C6	10	80	3C	:	¹ ó« ¾ ® ¹ ƃÆ <
128	80	74	0B	38	2C	75	02	E2	F2	BE	F6	06	EB	37	8B	D6	:	t 8,u âò¾ö ë7 Ö
144	49	74	09	83	C6	10	38	2C	75	EF	E2	F7	BB	00	7C	8B	:	It ƃÆ 8,uïâ÷» |
160	F2	8B	14	8B	4C	02	E8	2E	00	72	12	8B	FB	81	BD	FE	:	ò ‹ û L è . r ‹ û ½ þ
176	01	55	AA	75	0D	B8	50	00	8B	EE	06	53	CB	BE	10	07	:	..UªuP ¸ P ‹ î S Ë ¾ ....
192	EB	03	BE	27	07	E8	02	00	EB	FE	B4	0E	BB	07	00	AC	:	ë ¾ ' è ë þ ´ » ¬
208	CD	10	38	3C	75	F9	C3	60	BF	05	00	B8	01	02	CD	13	:	Í 8<uùÃ`¿ ¸ Í
224	73	0A	4F	74	06	33	C0	CD	13	EB	F0	F9	61	C3	00	00	:	s.Ot 3ÀÍ ëðùaÃ
240	00	00	00	00	00	00	0A	0D	50	61	72	74	69	74	69	6F	:	. . . . . Partitio
256	6E	20	74	61	62	6C	65	20	69	6E	76	61	6C	69	64	00	:	n table invalid.
272	0A	0D	45	72	72	6F	72	20	72	65	61	64	69	6E	67	20	:	..Error reading
288	73	65	63	74	6F	72	00	0A	0D	4F	70	65	72	61	74	69	:	sector...Operati
304	6E	67	20	73	79	73	74	65	6D	20	6D	69	73	73	69	6E	:	ng system missin
320	67	00	4D	42	52	20	63	6F	72	72	75	70	74	21	20	52	:	gMBR corrupt! R
336	75	6E	20	42	6F	6F	74	4D	61	67	69	63	20	63	6F	6E	:	un BootMagic con
352	66	69	67	75	72	61	74	69	6F	6E	0A	0D	50	72	65	73	:	figuration..Pres
368	73	20	61	6E	79	20	6B	65	79	20	74	6F	20	62	6F	6F	:	s any key to boo
384	74	20	61	63	74	69	76	65	20	70	61	72	74	69	74	69	:	t active partiti
400	6F	6E	0A	0D	00	00	00	00	00	00	00	00	00	00	00	00	:	on..............
416	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	.....
432	50	51	02	03	04	05	00	03	0D	21	0D	21	00	00	80	01	:	PQ.....!.!.....
448	01	00	0B	FE	7F	09	3F	00	00	00	4B	34	41	00	00	00	:	... þ ? ... K4A...
464	C1	FF	0F	FE	FF	FF	EE	3D	D2	01	6B	D1	C1	01	00	00	:	Áÿ þÿÿî=Ò.kÑÁ
480	41	0A	17	FE	FF	91	8A	34	41	00	88	D8	9E	00	00	00	:	A.. þÿ ‘ Š 4 A . ˆ Ø ž ...
496	C1	92	83	FE	FF	FF	12	0D	E0	00	DC	30	F2	00	55	AA	:	Á ’ ƒ þ ÿ ÿ .. . à . Ü 0 ò . U ª

Gambar 5.13 Tabel Partisi MBR (berwarna abu-abu 432–496)

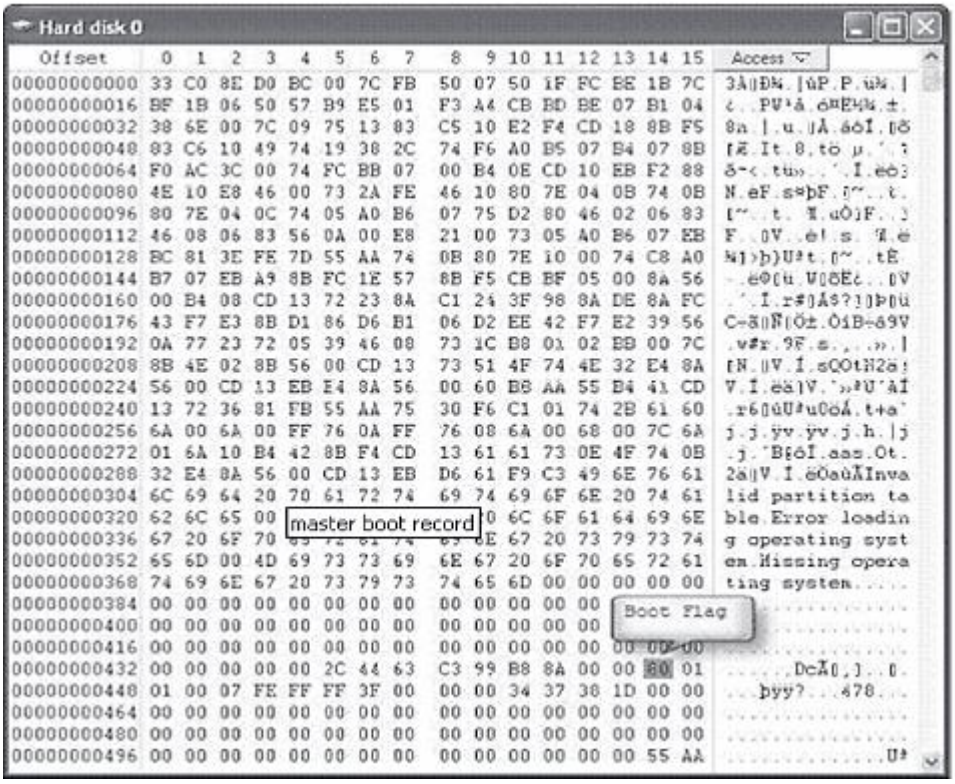
Byte tanda tangan harus selalu 0x55AA dalam MBR yang valid (ditampilkan dengan warna abu-abu pada Gambar 5.13). Tidak mungkin byte tanda tangan saja akan berubah tanpa bagian lain dari MBR juga berubah. Jika byte tanda tangan bukan 0x55AA, hard drive Anda tidak akan boot hingga diubah ke angka heksadesimal ini.

5.5 TABEL PARTISI

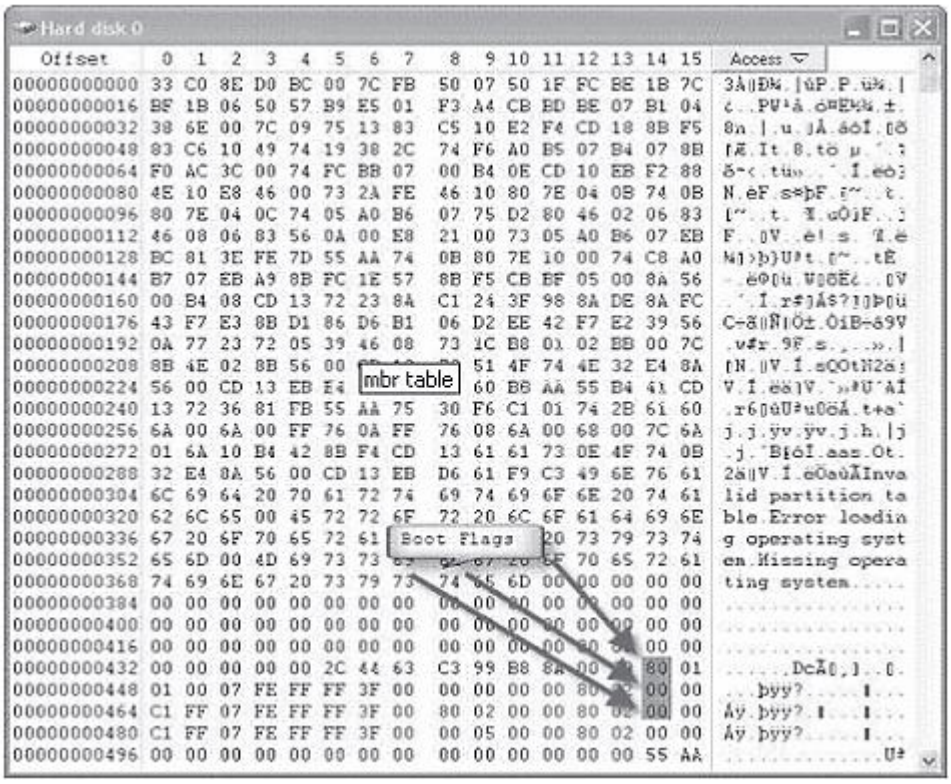
Tabel partisi yang ditunjukkan pada Gambar 5.13 dimulai dengan nilai 0x80 yang mewakili partisi aktif (dapat di-boot).

Tabel Partisi atau Tabel-P berisi empat deskriptor yang masing-masing panjangnya 16 byte dengan panjang total 64 byte (offset 446–509). Deskriptor mewakili informasi logis yang diperlukan untuk mengakses partisi di drive.[7] Tabel Partisi adalah bagian dari catatan boot master yang menjelaskan bagaimana disk dipartisi. MBR membaca tabel partisi untuk menentukan partisi mana yang aktif (berisi sistem operasi) dan di mana sektor bootnya berada.

Lihat Gambar 5.14 untuk melihat boot flag. Bendera boot diatur ke 0x80 (HEX 80), yang terletak di offset 446. Ini adalah catatan partisi normal, dalam satu hard drive yang dipartisi. Gambar 5.15 menunjukkan MBR untuk drive Multi-partisi.



Gambar 5.14 Boot Flag, di dalam MBR, Yang Terletak di Offset 446

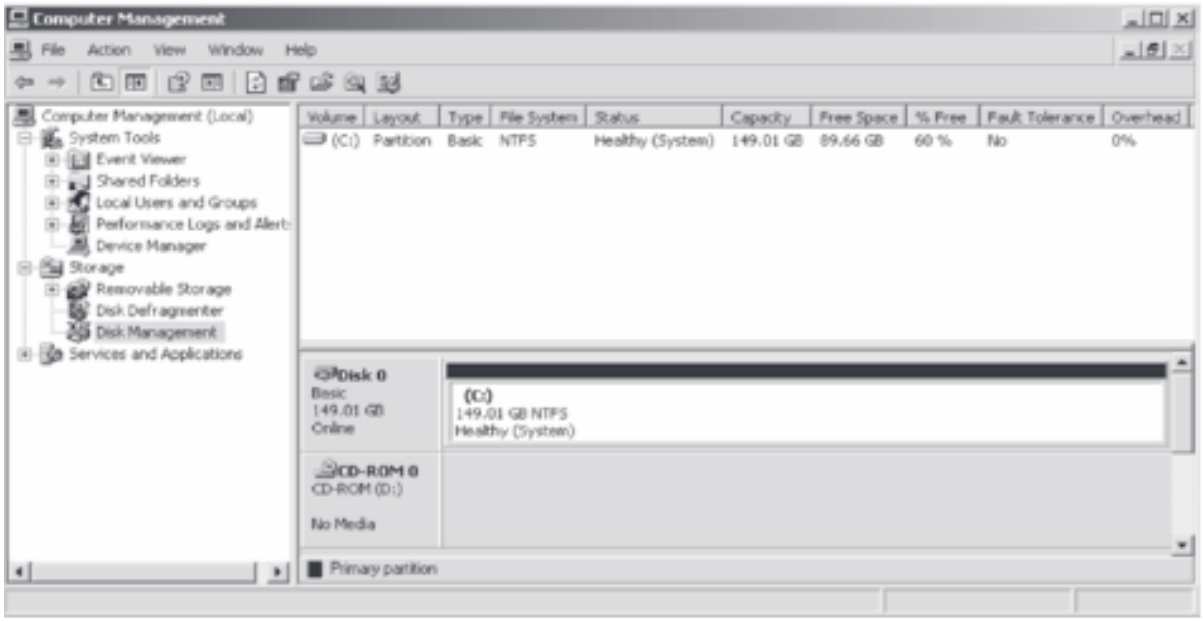


Gambar 5.15 Tampilan Drive Multi-Partisi

5.6 PARTISI HARDDISK

Partisi hard disk adalah ruang penyimpanan yang ditentukan pada hard drive. Hard drive dimulai dengan satu "partisi" yang menampung sistem operasi, aplikasi, game, musik, foto, video, dan semua data penting Anda.

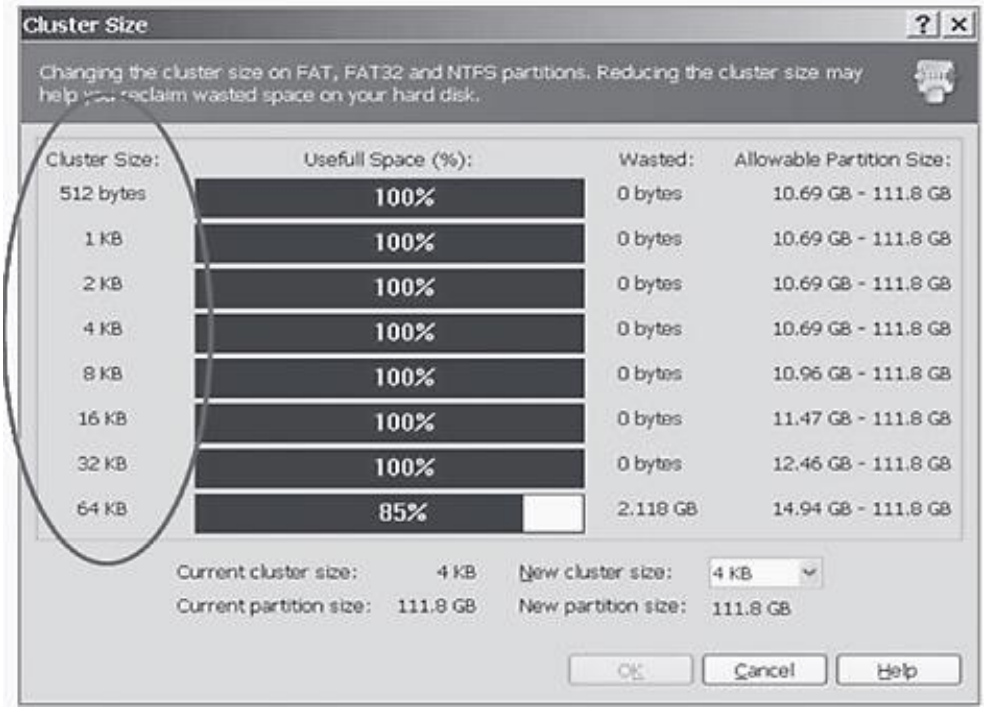
Seiring waktu, hard drive Anda menjadi sangat berantakan dan berantakan; Anda dapat secara signifikan meningkatkan kecepatan dan organisasi hard drive Anda dengan memisahkan sistem operasi, aplikasi, dan data penting ke dalam partisi terpisah di drive yang sama. Hal ini memungkinkan hard drive Anda untuk menemukan file lebih cepat dan lebih mudah.[8] Partisi dibuat saat Anda memformat hard disk. Biasanya, hard disk satu partisi diberi label drive "C:" ("A:" dan "B:" biasanya disediakan untuk drive disket), lihat Gambar 5.16. Hard drive dua partisi biasanya berisi drive "C:" dan "D:". (Drive CD-ROM biasanya diberi huruf terakhir dalam urutan huruf apa pun yang telah digunakan sebagai hasil dari pemformatan hard disk, atau biasanya dengan dua partisi, drive "E:".) Pengguna dapat memutuskan untuk membagi hard disk menjadi beberapa partisi karena partisi yang lebih kecil seringkali memiliki ukuran cluster yang lebih kecil.



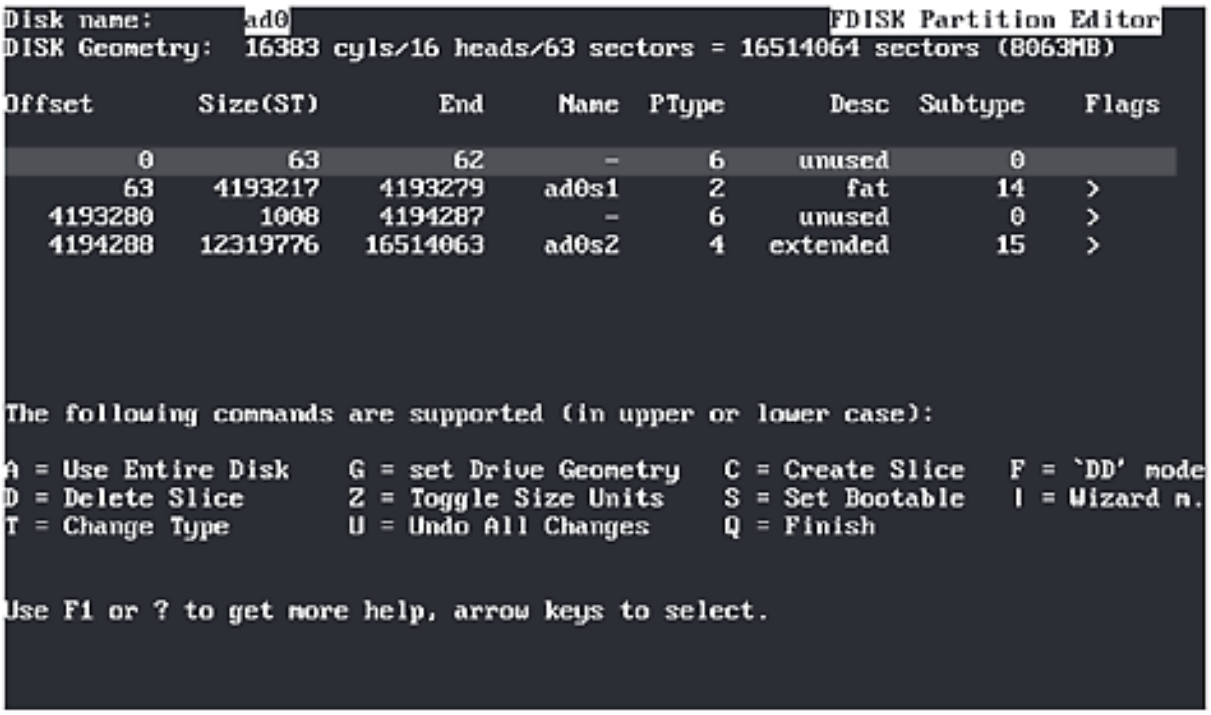
Gambar 5.16 Hard Drive Berpartisi Tunggal

Cluster adalah unit alokasi ruang disk untuk file dan direktori. Untuk mengurangi beban pengelolaan struktur data pada disk, sistem file tidak mengalokasikan sektor disk individual, tetapi kelompok sektor yang berdekatan, yang disebut kluster. Pada disk yang menggunakan sektor 512-byte, cluster 512-byte berisi satu sektor, sedangkan cluster 4-kibibyte (KB) berisi delapan sektor. (Lihat Gambar 5.17.) Cluster adalah jumlah logis terkecil dari ruang disk yang dapat dialokasikan untuk menyimpan file. Jika Anda memiliki banyak file kecil, ukuran kluster menjadi masalah, karena terlepas dari ukurannya, setiap file disimpan dalam setidaknya satu kluster. Ini berarti file dengan satu karakter (berukuran lima byte) dapat menempati ruang 16KB pada disk. Di partisi yang lebih kecil, sebuah cluster hanya dapat dialokasikan 4KB. Ini adalah strategi yang efisien jika Anda menyimpan sejumlah besar file kecil.

Sebagian besar sistem operasi menggunakan perintah “fdisk” untuk membuat partisi hard disk. (Lihat Gambar 5.18.) Banyak sistem operasi juga memiliki alat grafis yang menyelesaikan tugas yang sama.



Gambar 5.17 Ukuran Cluster Hard Disk



Gambar 5.18 Editor Partisi FDISK

Apa itu Partisi Logis (alias Volume)?

1. Volume adalah cara sistem operasi (Anda, saya, kami, kami, dll.) "melihat" ruang disk kosong Anda.
2. Volume (juga disebut drive logis) di Windows diwakili oleh huruf drive seperti C:, K:, X:, Y:, Z:, dan seterusnya.
3. Volume harus diformat dengan sistem file sebelum data dapat disimpan di dalamnya.

Kata Tanda Tangan

Byte tanda tangan harus selalu 0x55AA dalam MBR yang valid (ditampilkan dengan warna abu-abu pada Gambar 5.19). Jika byte tanda tangan bukan 0x55AA, hard drive Anda tidak akan boot sampai diubah ke angka heksadesimal ini.

Merangkum Proses Boot

Setelah menyalakan komputer Anda, hal pertama yang terjadi adalah BIOS (Basic Input Output System) mengambil kendali, menginisialisasi layar dan keyboard, dan menguji memori utama. Pada titik ini, tidak ada media penyimpanan atau perangkat eksternal yang diketahui oleh sistem. Setelah itu, sistem membaca tanggal dan waktu saat ini serta informasi tentang perangkat periferal terpenting dari pengaturan CMOS. Setelah membaca CMOS, BIOS akan mengenali hard disk pertama, termasuk detail seperti geometrinya. Itu kemudian dapat mulai memuat sistem operasi (OS) dari sana.

END OF THE MBR MASTER BOOT RECORD

000	FA	EB	01	00	8C	C8	8E	D8	8E	C0	8E	D0	BC	00	7C	FB	:	Ú.ë..
016	BE	00	7C	BF	00	06	B9	00	01	F3	A5	E9	00	8A	BE	B2	:	¾¿ ¹....óÝé ²¾²
032	07	38	0C	74	3C	BB	00	08	51	0F	B6	0C	BA	80	00	E8	:	8 t<» Q ¶ ° è
048	A5	00	59	72	21	46	FE	C5	81	C3	00	02	38	0C	75	E8	:	¥Yr!FþÅ Å 8 uè
064	33	C0	BE	00	08	03	04	46	46	E2	FA	3B	06	B0	07	0F	:	3À¾
080	85	0E	00	E9	4C	02	BE	10	07	E8	6E	00	BE	6A	07	EB	:	…é.éL.¾..èn.¾j.ë
096	03	BE	42	07	E8	63	00	33	C0	CD	16	33	C0	BF	00	08	:	.¾B.èc.3ÀÍ.3À¿..
112	B9	00	7C	F3	AB	BE	AE	07	B9	04	00	83	C6	10	80	3C	:	¹..ó«¾®.¹...ƃÆ.€<
128	80	74	0B	38	2C	75	02	E2	F2	BE	F6	06	EB	37	8B	D6	:	€t.8,u.âò¾ö.ë7.ÖÖ
144	49	74	09	83	C6	10	38	2C	75	EF	E2	F7	BB	00	7C	8B	:	It.ƃÆ.8,uïâ÷».|
160	F2	8B	14	8B	4C	02	E8	2E	00	72	12	8B	FB	81	BD	FE	:	ò‹.‹L.è..r.‹û.½þ
176	01	55	AA	75	0D	B8	50	00	8B	EE	06	53	CB	BE	10	07	:	.Uªu.¸P.‹î.SË¾...
192	EB	03	BE	27	07	E8	02	00	EB	FE	B4	0E	BB	07	00	AC	:	ë.¾'.è...ëþ´.».. ¬
208	CD	10	38	3C	75	F9	C3	60	BF	05	00	B8	01	02	CD	13	:	Í.8<uùÃ`¿...¸...Í.
224	73	0A	4F	74	06	33	C0	CD	13	EB	F0	F9	61	C3	00	00	:	s.Ot.3ÀÍ.ëðùaÃ..
240	00	00	00	00	00	00	0A	0D	50	61	72	74	69	74	69	6F	:	........Partitio
256	6E	20	74	61	62	6C	65	20	69	6E	76	61	6C	69	64	00	:	n table invalid.
272	0A	0D	45	72	72	6F	72	20	72	65	61	64	69	6E	67	20	:	..Error reading
288	73	65	63	74	6F	72	00	0A	0D	4F	70	65	72	61	74	69	:	sector...Operati
304	6E	67	20	73	79	73	74	65	6D	20	6D	69	73	73	69	6E	:	ng system missin
320	67	00	4D	42	52	20	63	6F	72	72	75	70	74	21	20	52	:	gMBR corrupt! R
336	75	6E	20	42	6F	6F	74	4D	61	67	69	63	20	63	6F	6E	:	un BootMagic con
352	66	69	67	75	72	61	74	69	6F	6E	0A	0D	50	72	65	73	:	figuration..Pres
368	73	20	61	6E	79	20	6B	65	79	20	74	6F	20	62	6F	6F	:	s any key to boo
384	74	20	61	63	74	69	76	65	20	70	61	72	74	69	74	69	:	t active partiti
400	6F	6E	0A	0D	00	00	00	00	00	00	00	00	00	00	00	00	:	on..............
416	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	.....
432	50	51	02	03	04	05	00	03	0D	21	0D	21	00	00	80	01	:	PQ.....!.!.....
448	01	00	0B	FE	7F	09	3F	00	00	00	4B	34	41	00	00	00	:	...þ.?...K4A...
464	C1	FF	0F	FE	FF	FF	EE	3D	D2	01	6B	D1	C1	01	00	00	:	Áy.þÿÿî=Ò.kÑÁ..
480	41	0A	17	FE	FF	91	8A	34	41	00	88	D8	9E	00	00	00	:	A..þÿ.Š4A.ˆØž...
496	C1	92	83	FE	FF	FF	12	0D	E0	00	DC	30	F2	00	55	AA	:	Á.ƒþÿÿ.. .à.Ü0ò.Uª

Gambar 5.19 Akhir dari MBR—Kata Tanda Tangan Dua-Byte (berwarna abu-abu)

Untuk memuat OS, sistem memuat segmen data 512-byte dari hard disk pertama ke dalam memori utama dan mengeksekusi kode yang disimpan di awal segmen ini. Instruksi

yang terkandung di dalamnya menentukan sisa proses booting. Inilah mengapa 512 byte pertama dari hard disk sering disebut Master Boot Record (MBR). Hingga saat ini (memuat MBR), urutan boot tidak bergantung pada sistem operasi yang diinstal dan identik di semua PC. Juga, semua PC harus mengakses perangkat keras periferal adalah rutinitas (driver) yang disimpan di BIOS. MBR terdiri dari tiga komponen utama (lihat Tabel 5.1):

1. Pemuat boot (Byte Offset 0–445)
2. Tabel partisi (Byte Offset 446–509)
3. Byte tanda tangan (Byte Offset 510–511)

Tabel 5.1 Struktur Master Boot Record (MBR) Komputer Berbasis Intel

Offset Byte	Keterangan	Ukuran
0–445	Pemuat Boot	446 Byte
446–509	Tabel Partisi	64 Byte
510–511	Tanda Tangan MBR	2 Byte
Ukuran MBR		512

MASTER BOOT RECORD

000	1A	1B	01	00	8C	C8	8E	D8	8E	C0	8E	D0	BC	00	7C	FB	:	Ú.ë..
016	BE	00	7C	BF	00	06	B9	00	01	F3	A5	E9	00	8A	BE	B2	:	¾¿ ¹....óÝé ²%
032	07	38	0C	74	3C	BB	00	08	51	0F	B6	0C	8A	80	00	E8	:	8 t<» Q ¶ °
048	A5	00	59	72	21	46	FE	C5	81	C3	00	02	38	0C	75	E8	:	¥YrlFþÅ Á 8 uè
064	33	C0	BE	00	08	03	04	46	46	E2	FA	3B	06	B0	07	0F	:	3A¾
080	85	0E	00	L9	4C	02	8E	10	07	8E	6E	00	8E	6A	07	1B	:	ÊÈ¾: ºên¾j ë
096	03	BE	42	07	E8	63	00	33	C0	CD	16	33	C0	BF	00	08	:	
112	B9	00	7C	F3	A8	BE	AE	07	B9	04	00	83	C6	10	80	3C	:	¹Ó¨¾®Á¹ 3Æ Á<
128	80	74	0B	38	2C	75	02	E2	F2	BE	F6	06	EB	37	8B	D6	:	t 8,u âò¾ö ë7 Ö
144	49	74	09	83	C6	10	38	2C	75	1F	L2	17	BB	00	7C	8B	:	It Æ 8,uïã+»Ø
160	F2	8B	14	8B	4C	02	E8	2E	00	72	12	8B	FB	81	BD	FE	:	ò Ø ò.r Ø½þ
176	01	55	AA	75	0D	8B	50	00	8B	EF	06	53	CB	BF	10	07	:	..UªuP Ø SË¿...
192	EB	03	BE	27	07	E8	02	00	EB	FE	B4	0E	BB	07	00	AC	:	ë
208	CD	10	38	3C	75	F9	C3	60	BF	05	00	B8	01	02	CD	13	:	Í¹-âuùÃ`¿ ¸ Í
224	73	0A	4I	74	06	33	CD	CD	13	LB	I0	I9	61	C3	00	00	:	s.Ot 3Íl ëðúaÃ
240	00	00	00	00	00	00	0A	0D	50	61	72	74	69	74	69	6F	:	. . . . . Partitio
256	6F	20	74	61	62	6C	65	20	69	6F	76	61	6C	69	64	00	:	n table invalid.
272	0A	0D	45	72	72	6F	72	20	72	65	61	64	69	6E	67	20	:	..Error reading
288	73	65	63	74	6F	72	00	0A	0D	4F	70	65	72	61	74	69	:	sector...Operati
304	6L	67	20	73	79	73	74	65	6D	20	6D	69	73	73	69	6L	:	ng system missin
320	67	00	4D	42	52	20	63	6F	72	72	75	70	74	21	20	52	:	gMBR corrupt! R
336	75	6E	20	42	6F	6F	74	4D	61	67	69	63	20	63	6F	6E	:	un BootMagic con
352	66	69	67	75	72	61	74	69	6F	6E	0A	0D	50	72	65	73	:	figuration..Pres
368	73	20	61	6L	79	20	6B	65	79	20	74	6I	20	62	6I	6I	:	s any key to boo
384	74	20	61	63	74	69	76	65	20	70	61	72	74	69	74	69	:	t active partiti
400	6F	6F	0A	0D	00	00	00	00	00	00	00	00	00	00	00	00	:	on.....
416	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	:	.....
432	50	51	02	03	04	05	00	03	0D	21	0D	21	00	00	80	01	:	PQ.....!.!.....
448	01	00	0B	I L	7I	09	3I	00	00	00	4B	34	41	00	00	00	:	... Þ8?...K4A...
464	C1	FF	0F	FE	FF	FF	EE	3D	D2	01	6B	D1	C1	01	00	00	:	Áÿ þÿÿî=Ò.kÑÁ
480	41	0A	17	FF	FF	91	8A	34	41	00	88	D8	9F	00	00	00	:	A.. þÿ 4A.Ø..
496	C1	92	83	FE	FF	FF	12	0D	E0	00	DC	30	F2	00	55	AA	:	Á.þÿÿ.. à.Ü0ò.Uª

Gambar 5.20 Representasi Grafis dari MBR

Gambar 5.20 menunjukkan representasi grafis dari MBR.

1. Boot loader (Byte offset 0–445). Pemuat boot MBR terdiri dari kode yang dimuat BIOS untuk mem-boot sistem operasi. Boot loader bekerja dengan mencari partisi aktif di tabel partisi dan memuat sektor pertama di partisi tersebut. Sektor itu dikenal sebagai Partition

Boot Record. Partition Boot Record kemudian akan memulai proses pemuatan kernel sistem operasi.

2. Tabel partisi (Byte offset 446–509). Panjangnya 64 byte, terdiri dari empat entri 16 byte (4 × 16 = 64). Tabel partisi berisi empat deskriptor yang panjangnya masing-masing 16 byte. Tabel mendefinisikan atau menjelaskan ruang penyimpanan atau partisi. Deskriptor mewakili informasi logis yang diperlukan untuk mengakses partisi di drive. Tabel partisi dimulai dengan nilai 80 (HEX), yang mewakili partisi aktif (dapat di-boot).
3. Byte tanda tangan (Byte offset 510–511). Byte tanda tangan MBR adalah dua byte terakhir dari sektor pertama, dan digunakan sebagai validasi sederhana dari konten MBR. Ketika MBR dimuat, BIOS memeriksa dua byte terakhir dari sektor tersebut. Dua sektor terakhir harus berisi nilai HEX 55AA. Jika tanda tangan catatan boot ini tidak ada, pesan kesalahan seperti "masukkan disk boot" atau "boot nonsistem" akan muncul.

5.7 RINGKASAN

Memahami bahwa nilai HEX 55AA adalah tanda tangan rekaman boot mungkin tidak membuat atau menghancurkan kasus. Pada kenyataannya, penyelidik forensik dunia maya bahkan mungkin tidak pernah menemukan kebutuhan untuk mencari atau memverifikasi data tersebut. Lalu mengapa kita harus mempelajari detail yang begitu teliti?

Ini adalah bagian dari teka-teki yang perlu dipahami untuk mendapatkan pemahaman yang lengkap tentang proses booting. Seperti yang dibahas sepanjang bab ini, pemahaman yang kuat tentang proses boot diperlukan jika, untuk satu hal, mengetahui kapan bukti dapat, mungkin, atau diubah dan dengan demikian menghindari kontaminasi bukti melalui proses pencitraan adalah penting.

Memahami proses boot juga penting karena akan membawa kita ke langkah logis berikutnya untuk mendapatkan pemahaman yang lebih baik tentang forensik dunia maya: sistem file. Bab 7 akan membahas langkah selanjutnya dalam perkembangan alami dalam mengakses informasi pada sistem, memasang sistem file. Bab 6 berfokus pada endianness, atribut sistem yang menunjukkan apakah bilangan bulat direpresentasikan dari kiri ke kanan atau dari kanan ke kiri. Mengetahui bagaimana informasi ditulis ke disk sangat penting bagi penyelidik forensik dunia maya.

BAB 6
ENDIANNESS DAN TABEL PARTISI

Ketika Anda mencapai 900 tahun, terlihat bagus, Anda tidak akan melakukannya.

—Yoda

Berbicara dalam Big Endian, kebanyakan dari kita melakukannya.

—Penulis

BEBERAPA BAHASA MANUSIA dibaca dan ditulis dari kiri ke kanan; lainnya dari kanan ke kiri; beberapa dari bawah ke atas dan yang lain dari atas ke bawah. Urutan pengumpulan data dapat bervariasi secara dramatis dari budaya ke budaya, wilayah ke wilayah, dan negara ke negara.

Di Amerika Serikat dan banyak negara Amerika Utara, misalnya, tanggal berikut 12/09/12 akan mewakili 9 Desember 2012; di Inggris Raya (UK), dan banyak negara Eropa, bagaimanapun, ini akan mewakili tanggal 12 September 2012. Apa yang terjadi? Mengapa kita mengalami kebingungan ini? Hal ini karena urutan dan interpretasi nilai numerik dipandang berbeda oleh kelompok masyarakat yang berbeda. Di Amerika Serikat tanggal telah dan terus ditulis secara tradisional sebagai MM/DD/YYYY, sedangkan di banyak masyarakat/negara lain di seluruh dunia, representasi tanggal saat ini ditulis dan dipandang sebagai DD/MM/YYYY. Meskipun demikian, bahkan di Amerika Serikat, pencatatan dan tampilan tanggal terkadang ditampilkan sebagai DD/MM/YYYY (mis., formulir masuk imigrasi bea cukai AS untuk semua orang yang datang ke Amerika Serikat dari negara asing menggunakan DD/MM /format tanggal YYYY). Dengan demikian, representasi data tidak konsisten, bahkan di dalam negara yang sama. Untuk menginterpretasikan data ini dengan benar dalam format tanggal tertentu yang digunakan, urutan penyimpanan dan interpretasi informasi perlu diketahui, jika tidak, hasilnya akan sangat tidak akurat. Masalah serupa muncul di bidang komputer yang melibatkan representasi angka dan interpretasinya.

Seperti yang telah kita bahas, data elektronik disimpan pada level terendah dalam bit, bit dirangkai menjadi byte, byte menjadi kata, kata menjadi dwords, dan seterusnya. Endianness data elektronik melibatkan pengurutan unit-unit fundamental ini. Endianness adalah atribut sistem yang menunjukkan apakah bilangan bulat direpresentasikan dari kiri ke kanan atau kanan ke kiri. Lalu mengapa di dunia mesin virtual dan prosesor gigahertz saat ini, seorang programmer atau penyelidik forensik dunia maya peduli dengan spesifikasi teknis tingkat dasar seperti itu? Alasannya adalah bahwa keteguhan harus dipilih setiap kali arsitektur perangkat keras atau perangkat lunak dirancang, dan tidak banyak cara hukum alam untuk membantu memutuskan atau mendikte. Jadi, penerapan endianness berbeda-beda di antara produsen perangkat keras dan pengembang perangkat lunak.

6.1 JENIS ENDIANNESS

Secara umum, dalam komputasi, endianness hadir dalam dua Jenis: big endian dan little endian.

Di big endian, unit (atau byte) paling signifikan dari bidang data diurutkan terlebih dahulu, atau dibiarkan rata. Namun, dengan little endian, unit (atau byte) yang paling tidak signifikan dari bidang data diurutkan terlebih dahulu dengan byte paling signifikan di sebelah kanan, yaitu pembenaran kanan.

Mungkin timbul pertanyaan, apa yang menentukan byte paling signifikan? Dengan byte, bilangan bulat, dan angka, secara umum byte pertama biasanya yang paling signifikan. Digit pertama biasanya memiliki nilai terbesar. Sebagai contoh, dalam nilai Rupiah sebesar Rp.123.456.789,00, digit mana dari angka yang agak besar tersebut yang paling penting? Yang satu (1) tentu saja, karena mewakili 100 juta. Secara terpisah, sembilan (9) mungkin lebih besar dari 1, tetapi 9 hanya mewakili sembilan dolar.

Pada contoh tanggal sebelumnya, jika kita meletakkan tanggal dalam format big endian maka akan ditulis sebagai YYYY-MM-DD atau 2012-12-09. Dalam hal ini, satu tahun adalah periode waktu yang lebih signifikan daripada satu hari, jadi karena big endian, kami mengurutkan data dari yang paling signifikan hingga yang paling kecil. Oleh karena itu, tanggal yang sama yang ditulis dalam little endian akan ditulis sebagai DD-MM-YYYY, atau 12-09-2012. Format tanggal yang biasanya digunakan di Amerika Serikat (yaitu, MM-DD-YY), bagaimanapun, biasanya bukan keduanya; susunan ini terkadang disebut mixed endian atau middle endian.

Definisi berikut ini lebih tepat:

- Big endian ada ketika byte paling signifikan dari setiap bidang data multibyte disimpan di alamat memori terendah, yang juga merupakan alamat bidang yang lebih besar.
- Little endian berarti byte terkecil dari bidang data multibyte apa pun disimpan di alamat memori terendah, yang juga merupakan alamat bidang yang lebih besar.[1]

Contoh Big Endian

Di big endian, Anda menyimpan byte paling signifikan di alamat terkecil. Tabel 6.1 menunjukkan tampilannya.

Tabel 6.1 Contoh Penyimpanan Angka Menggunakan Big Endianness

Alamat	Nilai
1000	90
1001	AB
1002	12
1003	CD

Contoh Little Endian

Di little endian, Anda menyimpan byte paling tidak signifikan di alamat terkecil. Tabel 6.2 menunjukkan tampilannya.

Tabel 6.2 Contoh Penyimpanan Angka Menggunakan Little Endianness

Alamat	Nilai
1000	CD
1001	12
1002	AB
1003	90

Perhatikan bahwa urutannya terbalik dibandingkan dengan big endian. Untuk mengingat yang mana, ingat apakah byte paling signifikan disimpan terlebih dahulu (dengan demikian, little endian) atau byte paling signifikan disimpan terlebih dahulu (dengan demikian, big endian).

Semua prosesor harus ditetapkan sebagai big endian atau little endian. Prosesor Intel 80 × 86 dan tiruannya adalah little endian. SPARC Sun, Motorola 68K, dan keluarga PowerPC semuanya adalah big endian. Java Virtual Machine juga merupakan big endian. Beberapa prosesor bahkan memiliki sedikit register yang memungkinkan pemrogram untuk memilih endianness yang diinginkan. Perbedaan endianness dapat menyebabkan masalah jika komputer tanpa sadar mencoba membaca data biner yang ditulis dalam format yang berlawanan dari lokasi atau file memori bersama.

6.2 ENDIANNESS

Endianness menjelaskan bagaimana data multibyte direpresentasikan oleh sistem komputer dan ditentukan oleh arsitektur CPU dari sistem tersebut. Sayangnya, tidak semua sistem komputer didesain dengan arsitektur Endian yang sama. Perbedaan dalam arsitektur endian adalah masalah ketika perangkat lunak atau data dibagi antara sistem komputer. Analisis sistem komputer dan antarmukanya akan menentukan persyaratan implementasi perangkat lunak endian. Endianness hanya masuk akal jika Anda ingin membagi nilai besar (seperti kata) menjadi beberapa nilai kecil. Anda harus memutuskan pesanan untuk menempatkannya di memori.

Namun, jika Anda memiliki register 32-bit yang menyimpan nilai 32-bit, tidak masuk akal untuk membicarakan endianness. Registernya bukan big endian atau little endian. Itu hanya register yang menyimpan nilai 32-bit. Bit paling kanan adalah bit paling tidak signifikan, dan bit paling kiri adalah bit paling signifikan. Tidak ada alasan untuk mengatur ulang byte dalam register dengan cara lain. Endianness hanya masuk akal saat Anda memisahkan bidang data multibyte, dan mencoba menyimpan byte di lokasi memori yang berurutan. Dalam register, itu tidak masuk akal. Register hanyalah kuantitas 32-bit, b31 . . . b0, dan endianness tidak berlaku untuk itu.

Berkenaan dengan endianness, Anda mungkin berpendapat bahwa ada cara yang sangat alami untuk menyimpan empat byte dalam empat alamat berturut-turut, dan cara lainnya terlihat aneh. Secara khusus, ini terlihat "mundur". Namun, apa yang alami bagi Anda mungkin tidak alami bagi orang lain. Faktanya adalah bahwa kata tersebut dibagi menjadi empat byte, dan kebanyakan orang akan setuju bahwa Anda memerlukan urutan tertentu untuk menempatkannya di memori. Sebagian besar komputer berbasis Intel (x86, AMD, dll.)

menggunakan little endian. Komputer Apple berbasis non-Intel dan prosesor berbasis RISC lainnya misalnya, menggunakan big endian. Penting juga untuk dicatat bahwa lalu lintas jaringan menggunakan pengurutan big endian.

6.3 ASAL ASAL ENDIAN

Asal usul istilah aneh big endian dan little endian dapat ditelusuri ke buku Gulliver's Travels tahun 1726, oleh Jonathan Swift. Di salah satu bagian cerita, penolakan terhadap dekrit kekaisaran untuk memecahkan telur setengah matang di "ujung kecil" meningkat menjadi perang saudara. (Plotnya adalah sindiran tentang perpisahan Raja Henry VIII dari Inggris dengan Gereja Katolik.)

Pada tahun 1981, Danny Cohen, dalam makalahnya "On Holy Wars and a Plea for Peace," menggunakan Gulliver's Travels karya Jonathan Swift sebagai latar belakang kontroversi yang berkecamuk di Lilliput, menerapkan istilah dan sindiran untuk pertanyaan "What is the proper byte memesan dalam pesan?" Secara lebih khusus, pertanyaan yang diperdebatkan adalah, "Bagian mana yang harus berjalan lebih dulu — potongan dari ujung kata yang kecil atau potongan dari ujung kata yang besar? Cohen menyimpulkan bahwa "Kesepakatan atas suatu perintah lebih penting daripada perintah yang disepakati."

6.4 TABEL PARTISI DALAM MASTER BOOT RECORD

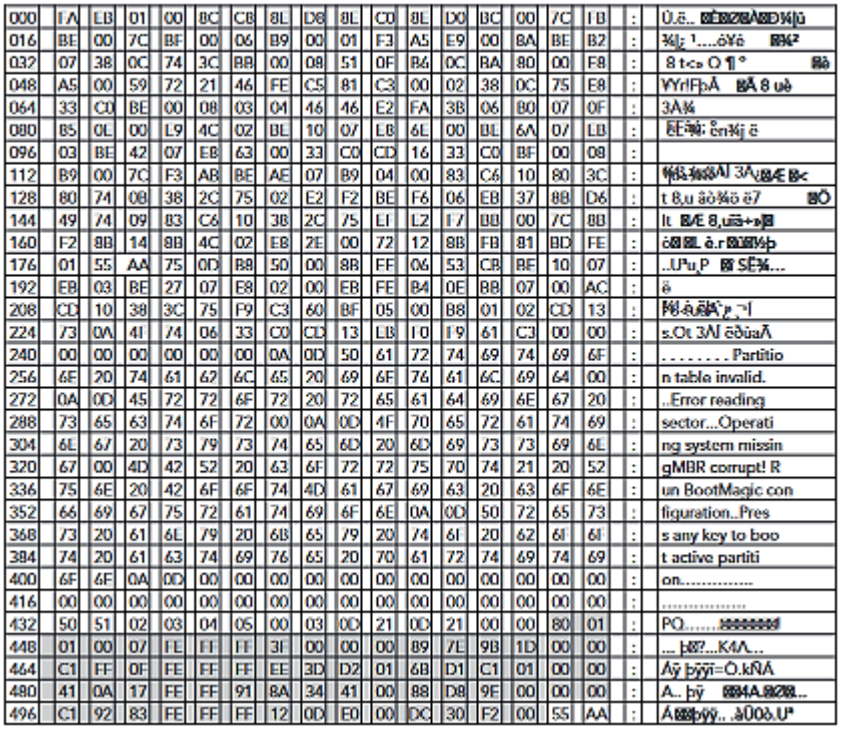
Di Bab 5 kita membahas Master Boot Record (MBR) dan data yang ada di dalamnya. Dalam forensik dunia maya, bagaimana data disimpan pada drive merupakan informasi yang sangat penting, karena seringkali, analis forensik dunia maya harus melihat data mentah di editor HEX untuk kemungkinan bukti; jadi, mengetahui bagaimana informasi ditulis ke disk, big endian atau little endian, sangatlah penting. Mari kita lihat MBR di editor HEX sekali lagi (lihat Gambar 6.1).

Untuk meringkas, mari kita lihat tabel partisi (disorot dalam warna abu-abu) pada offset byte 446–509. Dimulai dengan nilai 80 (HEX) yang mewakili partisi aktif (dapat di-boot). Ini berisi empat deskriptor yang panjangnya masing-masing 16 byte. Deskriptor mewakili informasi logis yang diperlukan untuk mengakses partisi pada drive. Tabel partisi dibagi menjadi empat bagian atau empat partisi primer. Partisi primer adalah partisi pada hard drive yang hanya dapat berisi satu drive logis (atau bagian). Setiap bagian dapat menyimpan informasi yang diperlukan untuk mendefinisikan satu partisi, artinya tabel partisi dapat mendefinisikan tidak lebih dari empat partisi.

Di dunia DOS/Windows, partisi diberi nama menggunakan metode berikut:

- Setiap tipe partisi diperiksa untuk menentukan apakah dapat dibaca oleh DOS/Windows.
- Jika tipe partisi kompatibel, maka akan diberi "huruf drive". Drive
- huruf dimulai dengan "C" dan beralih ke huruf berikutnya, tergantung pada jumlah partisi yang akan diberi label.
- ▪ Huruf drive kemudian dapat digunakan untuk merujuk ke partisi tersebut serta sistem file yang ada di partisi tersebut.

MASTER BOOT RECORD



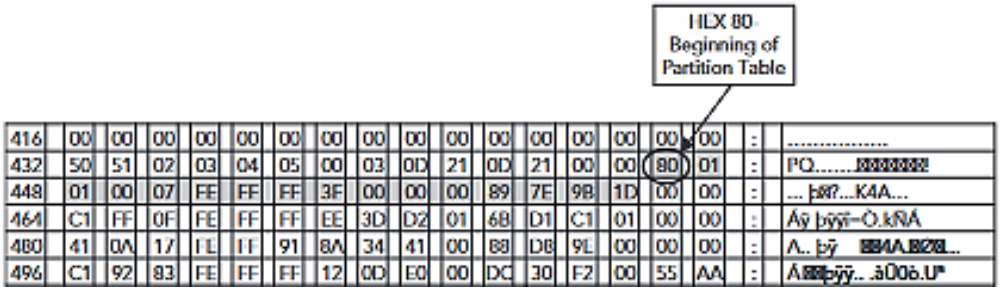
Gambar 6.1 Master Boot Record Ditampilkan di HEX Editor

Setiap entri tabel partisi berisi beberapa karakteristik penting dari partisi:

1. Apakah partisi tersebut "aktif". (Lihat Gambar 6.2.)
2. Lokasi pada disk tempat partisi dimulai. (Lihat Gambar 6.3.)
3. Jumlah total sektor yang terdapat dalam partisi. (Lihat Gambar 6.4.)
4. Jenis partisi. (Lihat Gambar 6.5.)

Partisi "Aktif".

Tabel partisi berisi entri (deskriptor) yang berfungsi sebagai penunjuk ke setiap partisi drive (volume) dan berisi informasi penting seperti jenis partisi, apakah partisi tersebut aktif atau tidak (dapat di-boot), di mana partisi dimulai dan diakhiri, dan ukuran partisi. Ingat, tabel partisi dapat menunjukkan maksimal empat partisi. (Sebuah teknik yang disebut partisi "diperpanjang" digunakan untuk memungkinkan lebih dari empat, dan sering kali digunakan ketika ada lebih dari dua partisi.) HEX 80 menunjukkan partisi aktif, di awal Tabel Partisi, ditunjukkan pada (Gambar 6.2). Bendera "aktif" digunakan oleh beberapa pemuat boot sistem operasi. Dengan kata lain, sistem operasi di partisi yang ditandai "aktif" di-boot. Perhatikan byte offset 08–11 dari partisi pertama (Gambar 6.2). Di MBR sistem operasi berbasis Windows, di sinilah kita akan menemukan lokasi untuk sektor awal partisi ini.



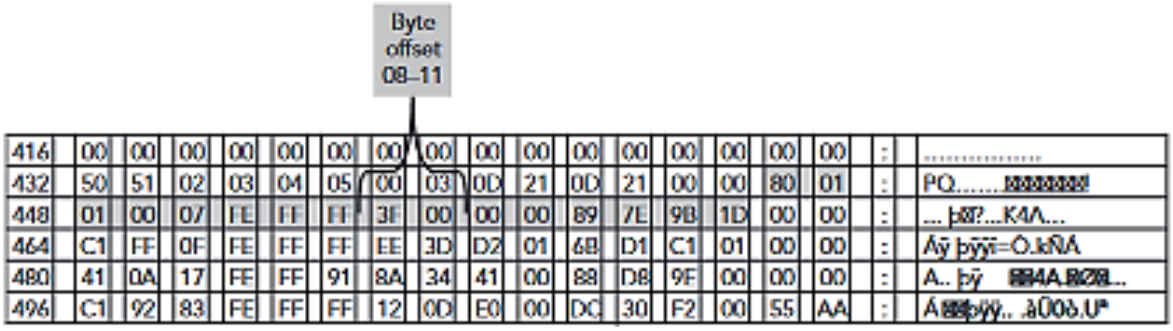
Gambar 6.2 Partisi Aktif dan Awal Tabel Partisi

Awal dari Partisi

HEX 3F 00 00 00, seperti yang ditunjukkan pada Gambar 6.3, menggambarkan awal dari partisi pertama, dan ditampilkan sebagai (atau dalam) format BIG ENDIAN. Membalik nilai ini ke dalam format LITTLE ENDIAN menghasilkan nilai 00 00 00 3F, atau setelah menghilangkan angka nol di depan cukup HEX 3F. Kami membalikkan nilai ini (HEX 3F 00 00 00) sehingga kami dapat memperoleh nilai sebenarnya dari HEX 3F, yaitu 63, nilai untuk partisi awal. Orang tahu kapan atau mengapa nilai ini perlu dibalik sama seperti orang tahu di mana mendapatkan nilai ini di tempat pertama. Jadi orang mungkin bertanya, "Bagaimana kita tahu bahwa HEX 0x80 di sektor pertama tabel partisi mengidentifikasi partisi yang aktif?"

"Bagaimana kita tahu dari mana tabel partisi dimulai? Bagaimana kita tahu nilai-nilai itu mengidentifikasi titik awal partisi, apalagi urutannya? Jawaban yang sangat tidak teknis untuk pertanyaan yang sangat logis dan penting ini adalah kita melakukan penelitian dan belajar. Kami mempelajari sistem operasi mana yang menyimpan data dalam format endianness mana dan memasukkannya ke memori. Pada akhirnya, beberapa hal hanya perlu dipelajari.

Jadi, seperti yang telah kita bahas sebelumnya tentang endianness dan bagaimana urutan nilai numerik dapat dibalik, nilai HEX ini perlu dibalik untuk mendapatkan nilai yang benar. Jika kami tidak membalikkan endianness, kami tidak akan mendapatkan data yang benar.



Gambar 6.3 HEX 3F 00 00 00—Sektor Awal Partisi

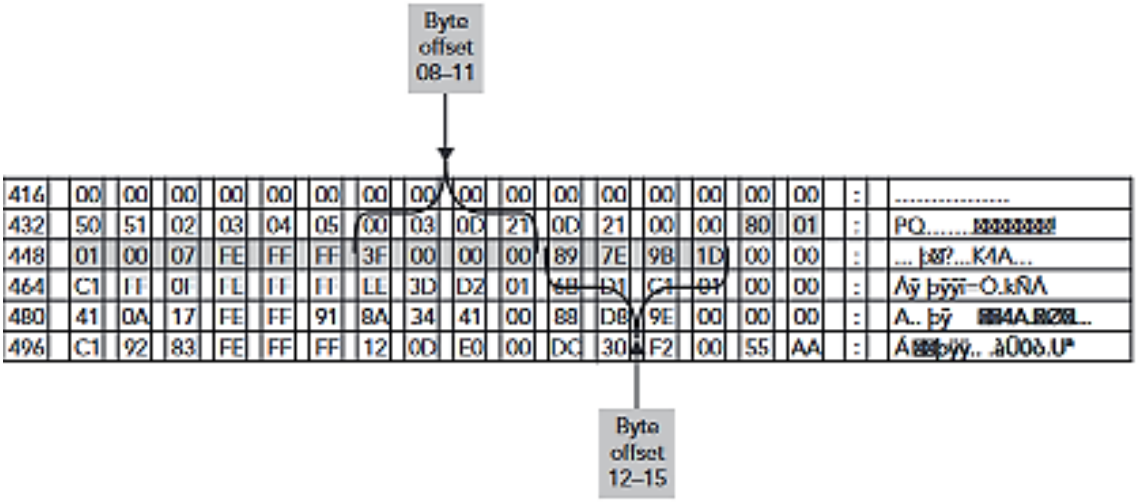
Menguraikan Nilai Awal Hex Partisi

Nilai HEX 3F diubah menjadi nilai desimal 63, karena begitulah sistem mengubahnya. Hanya saja bagaimana sistem menangani referensi data ini. Saat sistem melakukan booting, ia mencari byte di sektor ini dan menarik data yang dibutuhkannya. Sistem mengetahui apa yang harus dilakukan dengan data ini karena sistem secara khusus mencari nilai HEX 80.

Sistem melihat ke MBR, dalam offset yang telah ditentukan sebelumnya (byte 08–11) untuk HEX 80. Ketika sistem menemukan nilai yang dicari, HEX 80, sistem mengetahui bahwa partisi aktif telah ditemukan (lihat Gambar 6.2). Hal yang sama berlaku untuk sektor start (lihat Gambar 6.3); sistem mencari offset byte tersebut untuk mengekstrak data, membalikkan endian, dan mengonversi ke desimal. "Proses" ini termasuk dalam set instruksi dari proses booting itu sendiri. HEX 3F memiliki desimal yang setara dengan 63. Nilai desimal ini memberi kita alamat sektor relatif untuk awal partisi.

Ukuran Partisi

Bidang data multi-byte yang terdapat dalam byte offset 12–15 dari tabel partisi menentukan jumlah sektor yang terdapat dalam partisi, dengan kata lain, ukurannya (lihat Gambar 6.4). Nilai HEX ini (89 7E 9B 1D dalam contoh kita) juga "diukur" dalam little endian dan diubah menjadi nilai desimal. Nilai ini juga menentukan sektor terakhir dari partisi. Saat kami memperoleh sektor awal dari tabel partisi byte offset 8-11, kami cukup menambahkan jumlah sektor ke sektor awal dan mendapatkan sektor terakhir. Jadi dalam contoh ini, 89 7E 9B 1D adalah bidang data multi-byte. Kami pertama-tama membalikkan urutan byte dalam bidang data ini (little endian) ke 1D 9B 7E 89. Karena ini adalah bidang data multi-byte, semua byte diperiksa sebagai satu. Jadi data diubah menjadi nilai desimal. 1D 9B 7E 89 HEX, dikonversi ke persamaan desimalnya, sama dengan 496.729.737. Ini mendefinisikan jumlah sektor, 496.729.737. Karena kita tahu ada 512 byte per sektor, kita ambil nilainya 496.729.737 dan kalikan dengan 512, dan dapatkan nilainya 254.325.625.344 byte. Jadi sekarang kami dapat menyimpulkan bahwa kami memiliki 236 GB (drive 250 GB).



Gambar 6.4 Total Jumlah Sektor yang Terkandung dalam Partisi

Jenis Partisi

Jenis partisi bisa sedikit membingungkan. Jenisnya adalah angka yang mengidentifikasi penggunaan partisi yang diantisipasi. Jika pernyataan itu terdengar agak kabur, itu karena arti dari tipe partisi agak kabur. Beberapa sistem operasi menggunakan tipe partisi untuk menunjukkan tipe sistem file tertentu, untuk menandai partisi yang terkait dengan sistem operasi tertentu, untuk menunjukkan bahwa partisi tersebut berisi sistem operasi yang dapat di-boot, atau beberapa kombinasi dari ketiganya.

Jenis partisi mengacu pada hubungan partisi dengan partisi lain di drive disk. Ada tiga jenis partisi yang berbeda:

1. Partisi primer (partisi yang menempati salah satu dari empat slot partisi primer di tabel partisi disk drive).
2. Partisi yang diperluas (dikembangkan sebagai respons terhadap kebutuhan lebih dari empat partisi per disk drive. Sebuah partisi yang diperluas dapat berisi banyak partisi, sangat memperluas jumlah partisi yang dimungkinkan).
3. Partisi logis (partisi yang terdapat di dalam partisi yang diperluas).

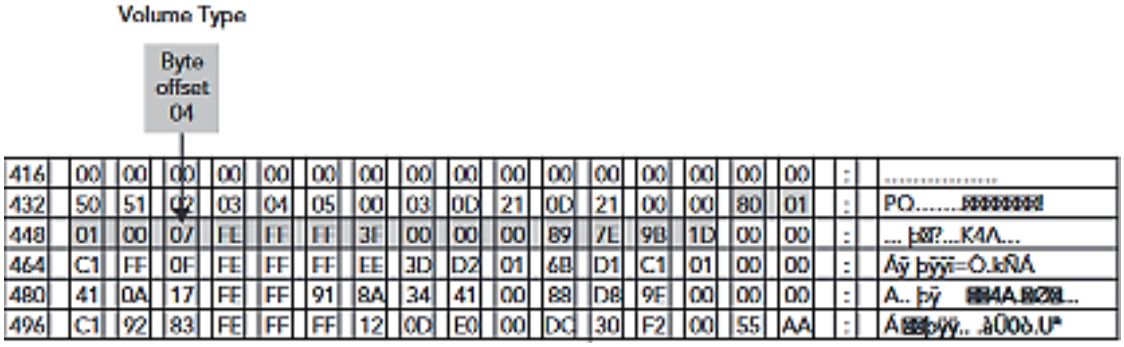
Setiap partisi memiliki bidang tipe yang berisi kode yang menunjukkan penggunaan partisi yang diantisipasi. Dengan kata lain, jika partisi akan digunakan sebagai Windows NT, tipe partisi harus diatur ke 07 (yang merupakan kode yang mewakili Windows NTFS). Tabel 6.3 menunjukkan beberapa tipe partisi yang diasosiasikan dengan sistem operasi tertentu dan nilainya yang ditetapkan.

Tabel 6.3 Jenis Partisi

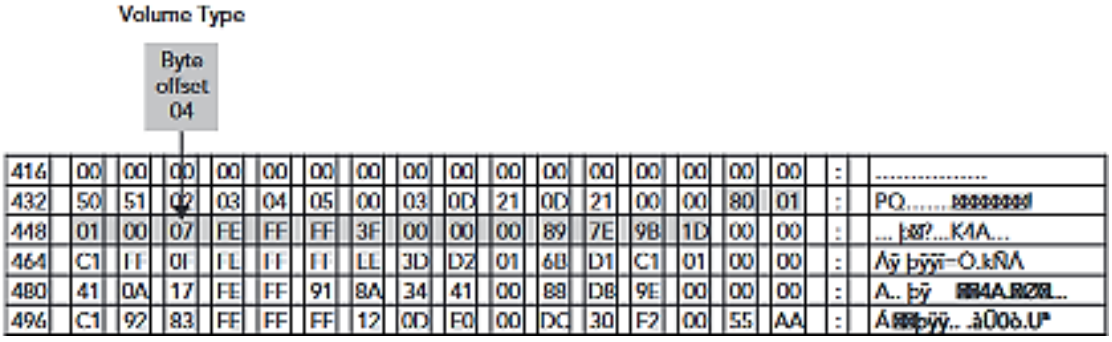
Tipe Partisi	Nilai
Kosong	00
DOS 12-bit FAT	01
DOS 16-bit <=32M	04
DOS 3.3+ Partisi yang Diperpanjang	05
DOS 3.31+ 16-bit FAT > 32M	06
Windows NT NTFS	07
Pengelola Boot OS/2	0a
Win95 FAT32	0b
Win95 FAT32 (LBA)	0c
Win95 FAT16 (LBA)	0e
Win95 Diperpanjang (LBA)	0f
Novell	51
Novell Netware 286	64
Novell Netware 386	65
Linux asli	83
Linux diperpanjang	85
Partisi pemulihan Partisi Magic	3c
Tabel Blok Buruk Xenix	ff

Untuk menentukan tipe partisi yang digunakan, kita lihat kolom System ID (byte offset 04) di dalam partisi (lihat Gambar 6.5). Untuk partisi utama dan drive logis, kolom ID Sistem menjelaskan sistem file yang digunakan untuk memformat volume. Bab 7 membahas sistem file secara lebih mendalam; untuk saat ini penting untuk hanya mengetahui bahwa sistem file (kadang-kadang ditulis sistem file) adalah cara di mana file diberi nama dan di mana mereka ditempatkan secara logis untuk penyimpanan dan pengambilan. Sistem operasi menggunakan

kolom ID Sistem (lihat Tabel 6.4) untuk menentukan driver perangkat sistem file apa yang akan dimuat selama pengaktifan. Dari Tabel 6.5, kita dapat menentukan sistem file yang tepat yang digunakan untuk memformat volume. Tinjauan Tabel Partisi yang ditunjukkan pada Gambar 6.6 memberi tahu kita bahwa tipe volume adalah partisi NTFS. Saat hard disk diformat (diinisialisasi), hard disk dibagi menjadi beberapa partisi atau divisi utama dari total ruang hard disk fisik. Di dalam setiap partisi, sistem operasi melacak semua file yang disimpan oleh sistem operasi tersebut.



Gambar 6.5 Kolom ID Sistem



Gambar 6.6 Byte Offset 04 Nilai HEX 7 (Partisi NTFS)

Tabel 6.4 Bidang Tabel Partisi

Offset Byte	Panjang Bidang	Nilai	Arti
00	BYTE	0 × 80	Indikator Boot. Menunjukkan apakah partisi adalah partisi sistem. Nilai legalnya adalah: 00 = Jangan gunakan untuk booting. 80 = Partisi sistem.
01	BYTE	0 × 01	Mulai Kepala.
02	6 bits	0 × 01	Sektor Mulai. Hanya bit 0–5 yang digunakan. Bit 6–7
03	10 bits	0 × 00	Silinder Mulai. Bidang ini berisi 8 bit yang lebih rendah dari nilai silinder. Silinder awal dengan demikian merupakan angka 10-bit, dengan nilai maksimum 1023.
04	BYTE	0 × 07	ID sistem. Byte ini mendefinisikan tipe volume.
05	BYTE	0 × 0F	Kepala Akhir.
06	6 bits	0 × 3F	Sektor Akhir. Hanya bit 0–5 yang digunakan. Bit 6–7 adalah dua bit atas untuk bidang silinder akhir.

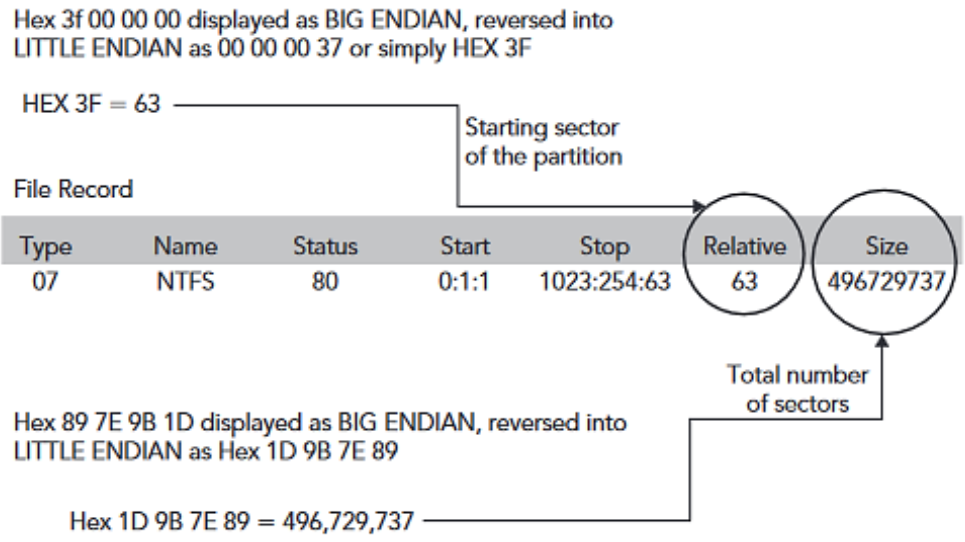
07	10 bits	0 × 196	Silinder Akhir. Bidang ini berisi 8 bit yang lebih rendah dari nilai silinder. Silinder akhir dengan demikian adalah angka 10-bit, dengan nilai maksimum 1.023.
08	DWORD	3F 00 00 00	Sektor Relatif.
12	DWORD	51 42 06 00	Total Sektor.

Tabel 6.5 Nilai untuk Bidang ID Sistem (byte offset 04)

Nilai	Arti
0 × 01	Partisi FAT 12-bit atau drive logis. Jumlah sektor dalam volume kurang dari 32.680.
0 × 04	Partisi FAT 16-bit atau drive logis. Jumlah sektor antara 32.680 dan 65.535.
0 × 05	Partisi yang diperluas.
0 × 06	Partisi FAT BIGDOS atau drive logis.
0 × 07	Partisi NTFS atau drive logis.

Setiap file sebenarnya disimpan di hard disk dalam satu atau lebih cluster atau ruang disk dengan ukuran seragam yang telah ditentukan sebelumnya. Menggunakan NTFS, ukuran cluster berkisar dari 512 byte hingga 64 kilobyte. Baca lebih lanjut tentang sistem file dan kepentingannya bagi penyelidik forensik dunia maya di Bab 7. Pada tampilan properti partisi Windows (lihat Gambar 6.7), semua karakteristik partisi yang telah dijelaskan ditampilkan oleh Sistem Operasi Windows. Perhitungan kami sekarang diverifikasi, dalam hal itu:

1. Status = 80—partisi aktif
2. Relatif = 63—lokasi awal
3. Ukuran = 496.729.737—dalam sektor
4. Ketik = 07—NTFS



Gambar 6.7 Tampilan Properti Windows dari Partisi

6.5 RINGKASAN

Dalam forensik dunia maya, bagaimana data disimpan pada drive merupakan informasi penting, karena sering kali, penyelidik forensik dunia maya harus melihat data mentah (melalui editor HEX) untuk kemungkinan bukti. Jadi, mengetahui bagaimana informasi ditulis ke disk, dan bagaimana data direpresentasikan dan disajikan secara fisik dan logis, sangatlah penting.

Setelah menyentuh konsep kunci penting di bab-bab sebelumnya, seperti HEX dan biner, masuk akal untuk memulai dari awal, dengan boot awal sistem. Kami membahas pentingnya Master Boot Record (MBR) dan isinya seperti tabel partisi, dan bagaimana sistem mengidentifikasi partisi aktif (HEX 80), sektor awalnya, dan ukurannya. Kami menyimpang dari proses boot dan mempelajari endianness untuk memahami bagaimana sistem menangani atau menginterpretasikan data ini. Karena endianness atau urutan data yang terkandung dalam MBR tunduk pada ukuran tersebut, sangat penting untuk memperluas konsep penting ini, seperti yang kita miliki di bagian ini. Ingat endianness tidak eksklusif untuk data yang terkandung dalam MBR. Data lain yang terkandung dalam hard drive juga tunduk pada tindakan tersebut.

Mungkin timbul pertanyaan, bagaimana komputer tahu untuk melihat rentang byte tertentu di sektor tertentu? Dan, bagaimana cara mengetahui untuk mengganti urutan? Mengapa itu tidak menangani semua biner saat menemukannya, seperti berapa banyak yang dibaca manusia, dari kiri ke kanan?

Penting untuk dipahami bahwa tidak semua data biner diperlakukan sama. Cara menangani biner (HEX, dalam pandangan kami) semuanya tergantung pada arsitektur sistem, atau kodenya. Saat sistem mem-boot, ia akan menemukan kode yang akan memerintahkannya untuk pergi ke sini dan melakukan ini atau itu atau yang lainnya.

Logika yang ditafsirkan secara longgar mungkin lebih masuk akal untuk dilihat dengan cara ini:

Langkah 1—buka byte offset 8–11 di tabel partisi.

Langkah 2—lihat empat byte data ini sebagai satu nilai (bidang data)—Dword.

Langkah 3—sebelum menerapkan matematika, balik urutan byte—yaitu, little endian.

Langkah 4—ubah Dword menjadi nilai desimal. Jawaban = sektor awal dari partisi.

Bagaimana suatu sistem mengetahui ke mana harus mencari atau bagaimana ia mengetahui cara menghitung nilai adalah apa yang dikenal sebagai arsitektur sistem. Penyelidik forensik dunia maya tidak perlu memahami semua logika di balik desain dan arsitektur setiap sistem. Yang penting adalah bahwa kita dapat memperoleh informasi ini melalui sumber apa pun yang memungkinkan. Faktor penting adalah untuk mengkonfirmasi, menguji, dan memverifikasi data yang diambil. Sumber dapat menyatakan bahwa ukuran partisi dapat ditentukan dengan memperoleh nilai desimal byte offset 12–15 dari tabel partisi di little endian. Adalah bijaksana untuk mengonfirmasi hal ini dengan sumber lain, mengujinya, dan memverifikasi pengujiannya. Bahkan jika Anda 100 persen yakin, itu layak untuk diuji sehingga Anda dapat sepenuhnya memahami kerumitan yang terlibat. Bab 7 memperkenalkan dan membahas lebih lanjut konsep alamat blok logis dan sistem file, dan menyelidiki lebih lanjut penyimpanan dan representasi data dan pentingnya proses investigasi forensik cyber dan penyelidik forensik cyber.

BAB 7
VOLUME VERSUS PARTISI

Di Bab 5 Kita Membahas proses boot-up setelah menyentuh konsep kunci penting, file di Bab 4, dan di Bab 2 dan 3, HEX dan biner. Kami juga telah membahas pentingnya Master Boot Record (MBR) dan isinya seperti tabel partisi, dan bagaimana sistem mengidentifikasi partisi aktif (HEX 80) dan sektor awalnya serta ukurannya. Untuk menjelaskan bagaimana beberapa data yang terkandung dalam tabel partisi diinterpretasikan, kami menyimpang dan mempelajari endianness. Karena urutan beberapa data yang terkandung dalam MBR tunduk pada endianness ini, sangat penting untuk menguraikan bagaimana data diurutkan dan pentingnya endianness, karena hal itu memengaruhi data dalam MBR. Untuk melanjutkan ke langkah berikutnya dalam urutan kami, proses data yang dikumpulkan menjadi informasi yang dapat ditafsirkan manusia, data tersebut perlu ditempatkan dan diidentifikasi oleh sistem.

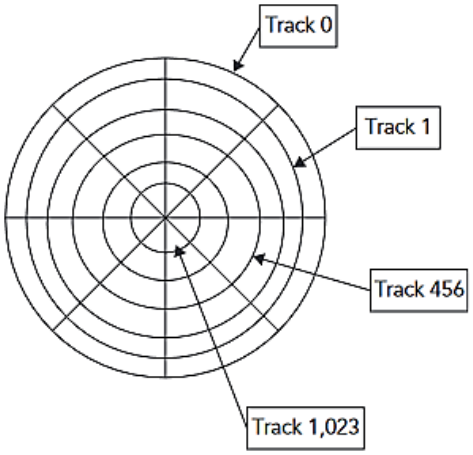
Kita sudah tahu bagaimana sektor awal partisi aktif dan ukurannya berasal dari tabel partisi; namun, ada beberapa offset byte tambahan di dalam tabel partisi yang belum kita bahas. Offset ini, disebut Cylinder, Head, and Sector (CHS) atau, saat ini, Logical Block Address (LBA), membantu menghitung lokasi dan ukuran partisi. Bab ini menjelaskan bagaimana partisi ditempatkan dan diidentifikasi dan peran penting volume dalam membuat data ini "dapat dipasang" atau dapat diakses dan dibaca oleh sistem Anda saat startup.

7.1 TINJAUAN TEKNOLOGI

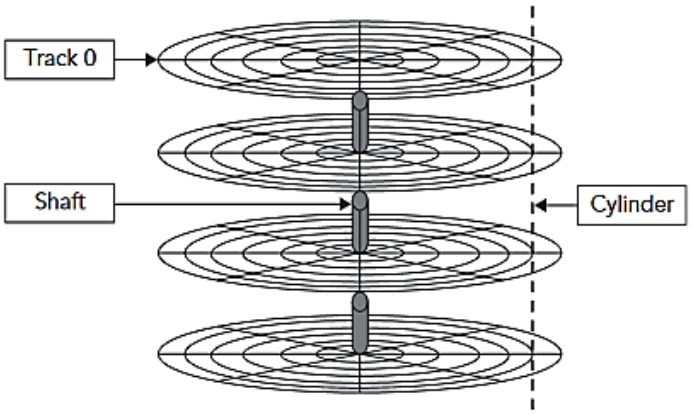
Sebelum memulai pemeriksaan volume dan partisi secara langsung, mungkin sebaiknya kita berhenti sejenak untuk tinjauan singkat tentang teknologi yang menggerakkan perangkat keras, tempat data berada. Hard disk atau fixed disk menyimpan informasi pada piringan logam atau kaca yang berputar yang dilapisi dengan bahan magnetik. Disk biasanya terdiri dari beberapa piringan fisik pada poros yang sama. Setiap disk terdiri dari piring-piring, cincin di setiap sisi setiap piring disebut track, dan bagian dalam setiap track disebut sektor. Sektor adalah unit penyimpanan fisik terkecil pada disk, hampir selalu berukuran 512 byte.

Track dan Silinder

Di hard disk, data disimpan di disk dalam pita konsentris tipis yang disebut track. Bisa ada lebih dari 1.000 track pada hard disk 3½ inci. Trek lebih logis daripada struktur fisik, dan dibuat saat disk diformat level rendah. Nomor track dimulai dari 0, dan track 0 adalah track terluar dari disk. Trek bernomor tertinggi ada di sebelah spindel. Jika geometri disk sedang diterjemahkan, trek bernomor tertinggi biasanya adalah 1.023. Lihat Gambar 7.1 untuk contoh ilustratif trek pada hard disk tipikal. Silinder terdiri dari kumpulan trek yang berada pada posisi kepala yang sama pada disk. (Lihat Gambar 7.2.)



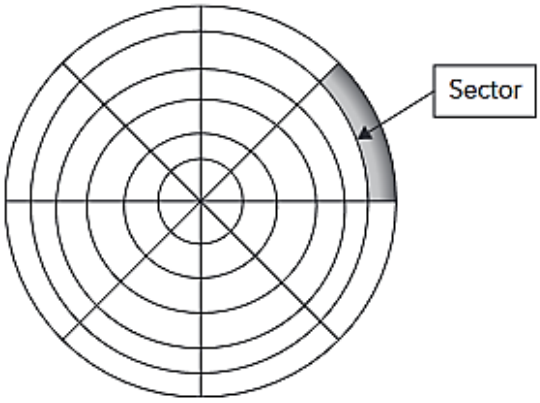
Gambar 7.1 Trek pada Hard Disk Khas



Gambar 7.2 Silinder

Sektor dan Cluster

Setiap trek dibagi menjadi beberapa bagian yang disebut sektor. Sektor adalah unit penyimpanan fisik terkecil pada disk. Ukuran data suatu sektor selalu berkekuatan dua, dan hampir selalu 512 byte. (Lihat Gambar 7.3.)



Gambar 7.3 Sektor

Setiap trek memiliki jumlah sektor yang sama, yang berarti sektor-sektor tersebut dikemas lebih dekat satu sama lain pada trek di dekat bagian tengah disk. Pengontrol disk menggunakan informasi identifikasi sektor yang disimpan di area tepat sebelum data di sektor untuk menentukan di mana sektor itu sendiri dimulai. Saat file ditulis ke disk, sistem file mengalokasikan jumlah cluster yang sesuai untuk menyimpan data file. Misalnya, jika setiap cluster berukuran 512 byte dan file berukuran 800 byte, dua cluster dialokasikan untuk file tersebut. Kemudian, jika Anda memperbarui file, misalnya, dua kali ukurannya (1.600 byte), dua klaster lagi akan dialokasikan. Jika cluster yang berdekatan (cluster yang bersebelahan pada disk) tidak tersedia, data ditulis di tempat lain pada disk dan file dianggap terfragmentasi. Fragmentasi adalah masalah ketika sistem file harus mencari beberapa lokasi berbeda untuk menemukan semua bagian dari file yang ingin Anda baca. Pencarian menyebabkan penundaan sebelum file diambil. Ukuran cluster yang lebih besar mengurangi potensi fragmentasi, tetapi meningkatkan kemungkinan bahwa cluster akan memiliki ruang yang tidak terpakai.

7.2 CYLINDER, HEAD, SECTOR, DAN LOGIC BLOK ADDRESSING

Pada sistem berbasis x86 yang lebih lama, ketika hard drive tidak melebihi 8 gigabyte, bidang awal dan akhir Cylinder, Head, dan Sector (CHS) dari partisi aktif (HEX 80) sangat penting selama booting sistem. Nilai-nilai ini, yang terkandung dalam Tabel Partisi MBR digunakan untuk menemukan dan memuat partisi ini. Pengalamatan Blok Logis (LBA) secara virtual menghilangkan hal ini dan melibatkan cara yang lebih baru untuk menangani sektor menggunakan lokasi sektor yang tepat. Alih-alih mengacu pada silinder, kepala, dan nomor sektor, masing-masing sektor diberi "nomor sektor" yang unik. Jadi alih-alih menggunakan semacam triangulasi dalam mengidentifikasi lokasi pada disk, LBA cukup menomori setiap sektor dengan nomor unik. Intinya, sektor diberi nomor 0, 1, 2, dst. hingga (N–1), di mana N adalah jumlah sektor pada disk.

Analoginya adalah sebagai berikut.

Alamat Anda (dengan asumsi Anda tinggal di Amerika Serikat dan memiliki alamat yang khas) terdiri dari nomor jalan, nama jalan, nama kota, dan nama negara bagian. Ini mirip dengan cara kerja pengalamatan CHS konvensional. Namun sebaliknya, katakanlah setiap rumah di Amerika Serikat diberi nomor pengenal unik. Ini akan menjadi lebih bagaimana LBA bekerja.

Silinder, head, dan Sektor

Tabel 7.1 menunjukkan tupel CHS yang sesuai untuk grup nilai LBA terpilih, dan bagaimana seseorang menghitung secara berurutan melalui sejumlah sektor yang diwakili oleh nilai-nilai ini. Data yang tercantum dalam Tabel 7.1 hanya berlaku untuk hard disk yang memiliki 63 sektor per track dan 255 head per silinder.

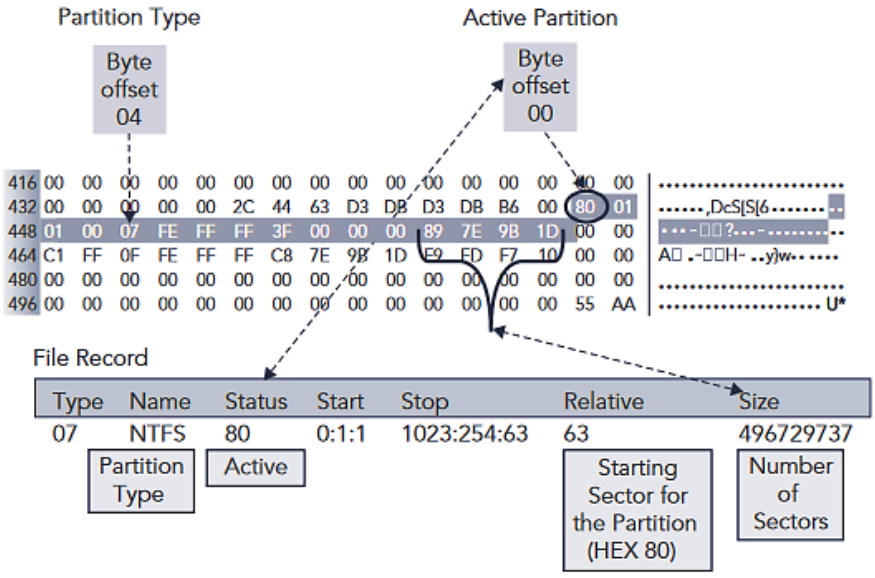
Tabel 7.1 Blok 512-Byte Diidentifikasi Menggunakan LBA dan CHS

Nilai LBA	Tupel CHS
0	0, 0, 1
1	0, 0, 2
2	0, 0, 3

62	0, 0, 63
63	0, 1, 1
64	0, 1, 2
65	0, 1, 3
125	0, 1, 63
126	0, 2, 1
127	0, 2, 2
188	0, 2, 63
189	0, 3, 1
190	0, 3, 2
16,063	0, 254, 62
16,064	0, 254, 63
16,065	1, 0, 1
16,606	1, 0, 2
16,127	1, 0, 63
16,128	1, 1, 1
16,450,497	1,023, 254, 1
16,450,558	1,023, 254, 62
16,450,559	1,023, 254, 63

Entri Entri Tabel Partisi Pertama

Mari kita periksa kembali tabel partisi yang ada di dalam MBR (lihat Gambar 7.4) dan tinjau kembali nilai-nilai tersebut, yang telah kita bahas (lihat Tabel 7.2).



Gambar 7.4 Tabel Partisi yang Terkandung dalam MBR

Tabel 7.2 Entri Byte dari Entri Tabel Partisi Pertama

Entri 16-Byte Pertama dari Tabel Partisi			
Nilai HEX	**Offset Byte**	**Keterangan**	**Hasil**
80	0	Dapat Di-boot/Aktif—80 = Ya; 00 = Tidak	yA
01 01 00	1, 2, 3	Memulai Sektor di CHS	0:01:01
07	4	Tipe Partisi	NTFS
FE FF FF	5, 6, 7	Sektor Terakhir di CHS	1023:254:63
3F 00 00 00	8, 9, 10, 11	Sektor Relatif—Sektor Awal—LBA	63
89 7E 9B 1D	12, 13, 14, 15	Total Sektor/Panjang	496729737

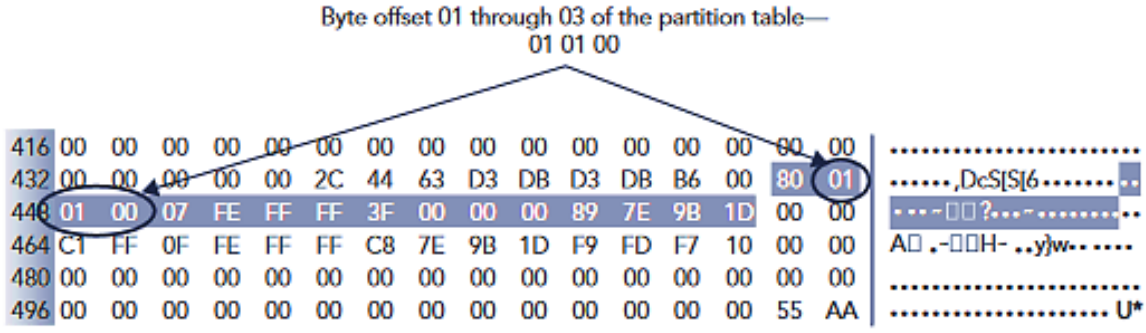
1. Nilainya, HEX 80, menunjukkan bahwa partisi tersebut adalah partisi aktif yang dapat di-boot.
2. HEX 07 menunjukkan sistem file, NTFS.
3. HEX 3F 00 00 00 menunjukkan sektor relatif atau awal, 63.
4. HEX 89 7E 9B 1D menunjukkan total sektor atau panjang (496, 729, 737).

Entri Byte Tersisa dari Entri Tabel Partisi Pertama

Nilai HEX yang tersisa adalah nilai untuk memulai dan mengakhiri nilai CHS. Kami tidak membahas ini sebelumnya karena konsep CHS dan LBA belum diperkenalkan. Mari kita tinjau ini sekarang.

Mulai Nilai CHS

Byte mengimbangi 01 hingga 03 dari tabel partisi—01 01 00 (lihat Gambar 7.5). Byte pertama dicadangkan untuk nilai Head—01, nilai HEX sisanya adalah 01 00. Byte berikutnya diberi nilai Sektor dan byte ketiga diberi Nilai Silinder.



Gambar 7.5 Nilai Awal CHS

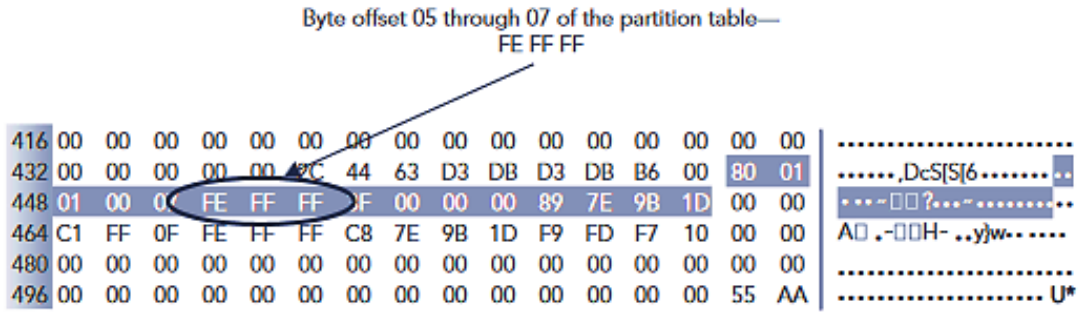
Dengan demikian nilai CHS 00:01:01 (0, 1, 1) (lihat Tabel 7.1):

- Nilai Kepala = 01
- Nilai Sektor = 01
- Nilai Silinder = 00

Ini mungkin terlihat mundur, tapi ingat, begitu juga dengan Little Endian. Mengapa byte nilai Sektor datang sebelum byte nilai Cylinder? Semuanya ada dalam kode. Mengapa menetapkan nilai HEX 80 ke partisi aktif? Ini semua ada hubungannya dengan pengkodean. Anda dapat melihat ini sebagai penataan ulang endian kecil dari dua byte terakhir.

Mengakhiri Nilai CHS

Byte offset 05 sampai 07 dari tabel partisi—FE FF FF (lihat Gambar 7.6). Nilai pertama dicadangkan untuk nilai Head (H = FE); ketika HEX FE diubah menjadi desimal, nilai yang dihasilkan sama dengan 254. Perhitungan yang tersisa menjadi sedikit rumit. Nilainya tidak dibalik (little endian), tetapi dikelompokkan ulang. Ingat, HEX adalah representasi biner kita. Ingat juga bahwa kode dapat ditulis untuk menjalankan fungsi matematika apa pun pada nilai biner apa pun. Seperti yang dinyatakan, byte pertama, dalam hal ini FE, dicadangkan untuk nilai HEAD, jadi kami menghapus nilai tersebut. Kami ditinggalkan dengan HEX FF FF.



Gambar 7.6 Byte Offset 05 sampai 07 dari Tabel Partisi

Langkah selanjutnya ini adalah tempat kita berkumpul kembali, tetapi untuk melakukannya kita harus mengonversi nilai HEX menjadi ekuivalen binernya, yang menghasilkan:

FF = 11111111

FF = 11111111

Kami menyatukan nilai-nilai biner ini—11111111 11111111, dan kami mengelompokkan kembali nilai-nilai tersebut, 6 bit pertama, dan 10 bit berikutnya.

Mengapa pengelompokan ulang ini?

Itu semua berkaitan dengan bagaimana kode menangani biner. Penting untuk diingat bahwa biner, pada dasarnya, hanyalah 0 dan 1. Ini adalah kode yang menggabungkannya menjadi byte sama seperti kode yang menyusun ulang dan membaginya. Pada akhirnya, jika kita melihat representasi data numerik yang disematkan di dalam permukaan disk (yaitu, piring), semua yang akan kita lihat adalah rangkaian panjang 0 dan 1 yang digabungkan secara berurutan.

Apa yang mendefinisikan tabel partisi? Apa yang mendefinisikan sektor? Apa yang mendefinisikan byte? Mengapa kata-kata dan Dwords? Mengapa endianisme? Mengapa pengelompokan ulang byte offset 05 hingga 07 dari tabel partisi ini?

Semuanya ada dalam kode (misalnya, firmware, ROM, kode boot, kode OS, pemrogram dan pengembang, dll.—tergantung pada "bagaimana" kode ditulis). Kode memberi tahu sistem "bagaimana" menangani bit biner "yang mana" dan bagaimana mengatur bit-bit ini dalam pengelompokan untuk interpretasi. Ini memberi kita pengelompokan bilangan biner seperti itu:

(11 1111) (11 1111 1111)

Sekarang mengubah biner 111111 menjadi padanan desimalnya, memberi kita nilai 63 (alamat sektor). Mengubah biner 1111111111 menjadi padanan desimalnya memberi kita nilai 1.023 (alamat silinder). Sektor diberi pengelompokan pertama dan silinder diberi pengelompokan kedua:

63 = Sektor

1.023 = Silinder

Tuple CHS terakhir adalah 1023:254:63. Menggunakan sistem CHS lama, titik awal untuk partisi adalah 0:1:1, dan titik akhir adalah 1023:254:63.

Ini adalah penjelasan panjang untuk mengekstraksi nilai CHS dari tabel partisi, tetapi untuk alasan apa informasi ini berharga atau bahkan diperlukan? Informasi yang lebih bermakna mengikuti pengalamatan CHS, yaitu sektor awal (dalam kolom relatif) dan ukuran dalam sektor. Ukuran partisi adalah 496.729.737 sektor, relatif terhadap sektor 63. Jadi, hal ini menyiratkan bahwa titik awal jika menggunakan referensi CHS adalah 0:1:1, sedangkan jika menggunakan referensi LBA, akan menjadi 63. Oleh karena itu, lokasi sektor awal untuk partisi di CHS adalah 0:1:1 dan di LBA, 63.

Mengapa ini penting?

Sebagai penyelidik forensik dunia maya, penting untuk memahami di mana data dan bagaimana data berada di hard drive. Alat forensik saat ini melakukan ini di belakang layar. Penyelidik forensik dunia maya mungkin tidak perlu menyibukkan diri dengan informasi ini untuk melakukan pemeriksaan forensik. Namun, sangat penting untuk memahami bagaimana dan mengapa hal ini terjadi. Jika dipanggil ke kursi saksi dan pembela bertanya, "Bisakah Anda memberi tahu pengadilan secara tepat di mana Anda menemukan bukti di hard drive dan bagaimana Anda membuat keputusan itu?" Anda perlu memahami konsep-konsep dasar ini.

Ringkasan CHS

Ini adalah sistem lawas, CHS, yang pada masanya sangat membantu dalam mengidentifikasi ukuran di hard disk yang lebih kecil. Itu dibatasi oleh jumlah bit yang tersedia di tabel partisi, yang digunakan untuk mewakili nilai-nilai ini.

Ingat dari Bab 1 kita membahas kemungkinan hasil atau kombinasi di sekitar skema pengkodean Basis dua; dengan satu (1) bit kami hanya memiliki dua hasil — hidup atau mati, dan dengan dua (2) bit kami memiliki empat (4) hasil atau kombinasi. Seperti yang baru saja kita diskusikan, satu byte penuh digunakan untuk menghitung nilai Head. Jika satu byte berisi delapan (8) bit, berapa banyak kemungkinan hasil yang dapat kita miliki dalam skema pengkodean Basis dua dengan delapan (8) bit? 256! Jadi, kita hanya dapat memiliki 256 nilai untuk variabel Kepala.

Catatan:
sebenarnya hanya ada 255 kemungkinan nilai untuk Head karena bug Microsoft.

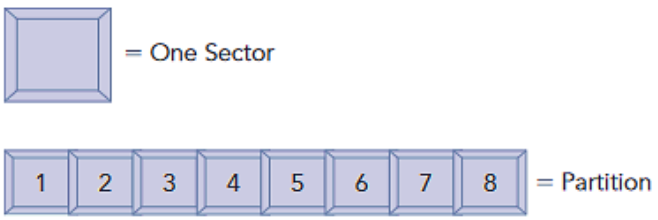
Karena nilai Head mulai dihitung dari nol (0), maka kemungkinan nilai terbesar untuk Head adalah 254. Ingat juga bahwa untuk menghitung nilai sektor, kami menghapus dua bit terdepan dari byte kedua (atau tengah) dan menggesernya ke atas ke nilai Silinder, sehingga menyisakan enam bit untuk nilai sektor. Oleh karena itu, nilai terbesar yang dapat Anda miliki untuk suatu Sektor adalah [11 1111] (enam bit), atau 63, dan demikian pula nilai terbesar yang dapat diperoleh untuk Silinder adalah [11 1111 1111] atau 1.023.

Jadi ukuran terbesar untuk penggerak yang menggunakan metode pelabelan CHS pembatas ditentukan dengan mengalikan nilai silinder, kepala, dan sektor (C * H * S) menjadi 1.023 * 254 * 63, yang sama dengan 16.370.046 sektor. Mengalikan jumlah sektor dengan jumlah byte per sektor (512), (16.370.046 * 512), menghasilkan jumlah total byte untuk drive sebesar 8.381.463.552 byte, atau kira-kira 8 GB drive. Parameter CHS tidak lagi menjadi pengidentifikasi sektor akhir setelah drive melebihi 8 GB.

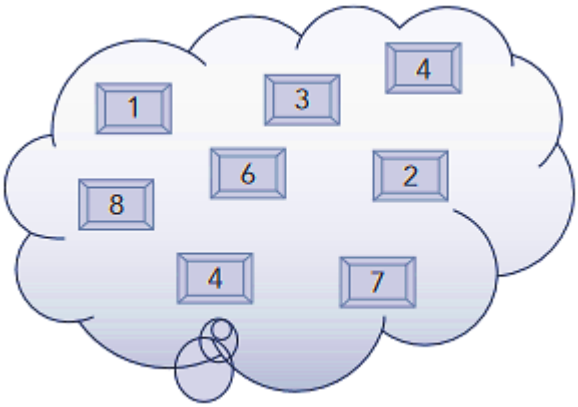
7.3 VOLUME DAN PARTISI

Dalam dunia teknologi informasi yang lebih besar, volume dan partisi sering kali digunakan oleh para praktisi karena mengacu pada hal yang sama; namun, ada perbedaan halus antara keduanya dan terkadang garis di antara keduanya bisa kabur. Singkatnya, volume ada di tingkat OS logis, dan partisi ada di tingkat fisik, khusus media. Terkadang ada korespondensi satu-ke-satu, tetapi tidak dijamin benar.

Partisi adalah kumpulan (secara fisik) sektor berurutan (lihat Gambar 7.7), di mana volume adalah kumpulan (secara logis) sektor yang dapat dialamatkan (lihat Gambar 7.8). Di sinilah letak perbedaannya; data yang terkandung dalam volume mungkin muncul berurutan, tetapi hanya secara logis.



Gambar 7.7 Sebuah Partisi Adalah Kumpulan Sektor Berturut-turut



Gambar 7.8 Sebuah Volume Adalah Kumpulan (LOGICALLY CONSECUTIF) Addressable Sector

Partisi, seperti yang dijelaskan di bab sebelumnya, adalah area hard disk drive yang ditentukan oleh entri di tabel partisi MBR, dan dikenali di seluruh sistem. Partisi ditafsirkan oleh kode yang terkandung dalam sektor yang sama, MBR, dan partisi biasanya merupakan subdivisi. Seperti namanya, itu adalah proses membelah sesuatu yang lebih besar menjadi bagian-bagian yang lebih kecil. Volume adalah area yang ditentukan atau ditafsirkan oleh sistem operasi. Volume dikenali oleh sistem operasi dan akan memiliki huruf drive yang terkait dengannya. Ini sering digunakan secara sinonim dengan istilah drive atau disk. (Lihat Tabel 7.3.)

Tabel 7.3 Volume dan Partisi

Disk Fisik	Partisi	Berkas sistem	Dalam Komputer
Hardisk 1	Partisi 1	NTFS	C:
	Partisi 2	FAT32	D:
Harddisk 2	Partisi 1	FAT32	E:

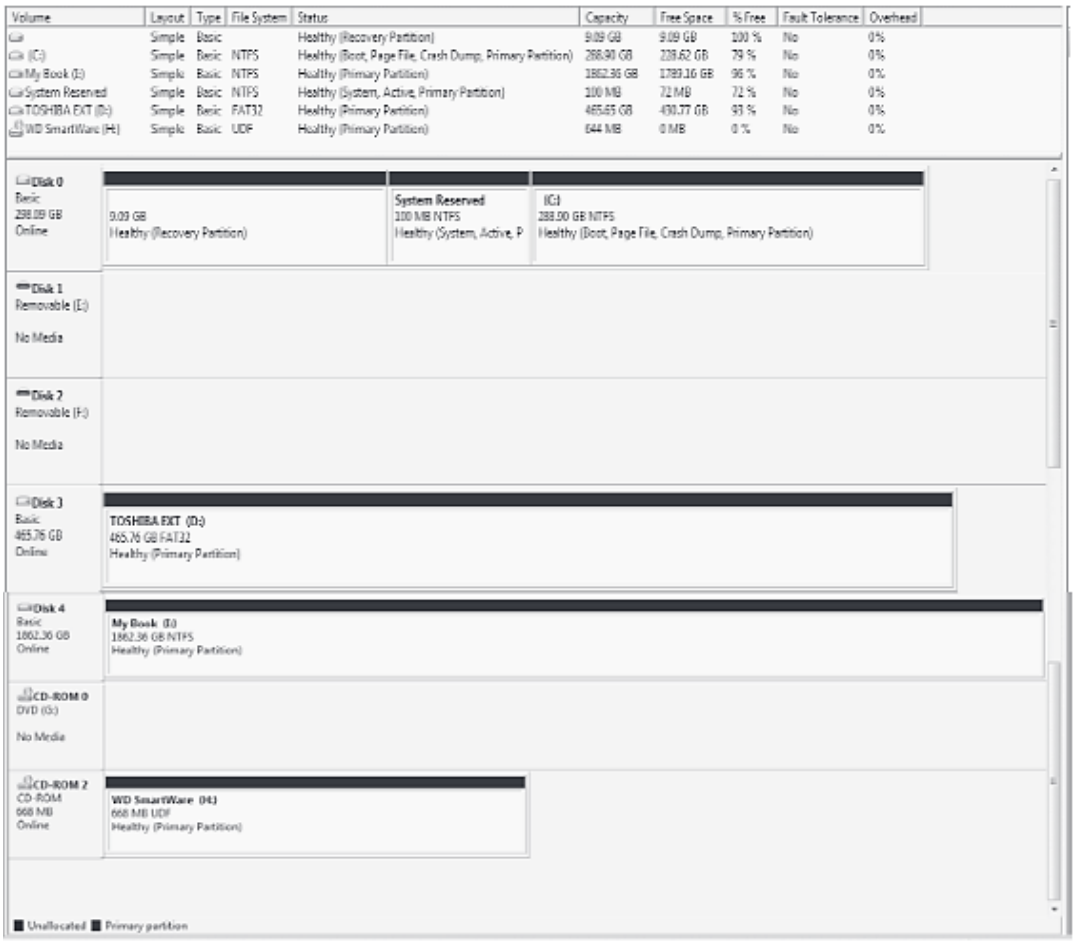
Dalam contoh ini:

1. "C:," "D:," dan "E:" adalah volume.
2. Hard Disk 1 dan Hard Disk 2 adalah disk fisik.
3. Semua ini bisa disebut "drive".

Mungkin yang paling penting, sebuah volume berisi sistem file, yang unik untuk sistem operasi dan hanya dipahami oleh sistem operasi tertentu. Sistem file akan dibahas secara mendalam di Bab 8. Namun, sifat logis ayat fisik dari partisi dan volume tidak selalu eksklusif satu sama lain. Perbedaan atau kesamaan terkadang menjadi kabur karena tidak diciptakan dengan gagasan tentang orang lain. Faktanya, berkali-kali mereka adalah hal yang sama. Misalnya, jika sebuah sistem berisi satu hard disk drive maka itu akan berisi satu volume, sehingga jika hard disk dibagi menjadi partisi (sebagaimana ditetapkan oleh tabel partisi di MBR), maka setiap partisi akan menjadi volume . Dalam hal ini, partisi juga merupakan volume dan pada dasarnya sama, meskipun definisinya tetap unik. Jadi volume bisa berupa seluruh hard disk atau partisi. Volume dapat berupa partisi, flash drive seperti iPod, floppy disk, dipasang dari server jaringan, atau bahkan larik RAID.

Alasan Mengapa Kami Memiliki Partisi dan Volume

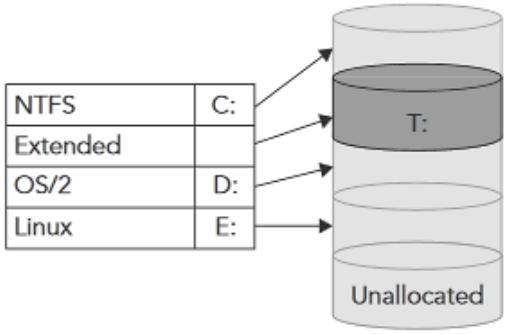
Mempartisi drive dapat membantu meningkatkan efisiensi HDD dengan membuat ukuran cluster lebih kecil. Volume, di sisi lain, membuat area penyimpanan yang intuitif secara logis (mis., "Simpan ke drive K"). Volume diidentifikasi oleh sistem file, dan Sistem file adalah cara file diakses dan disimpan oleh sistem operasi. Jadi, seseorang mungkin memiliki dua volume terpisah untuk menjalankan dua sistem operasi terpisah pada mesin yang sama. Volume menyediakan opsi konfigurasi pemulihan, ketersediaan data, kinerja, dan penyimpanan yang ditingkatkan. (Lihat Gambar 7.9.)



Gambar 7.9 Tampilan Disk Manager—Partisi

Partisi yang Diperpanjang

Kita sudah tahu bahwa satu hard drive dapat memiliki hingga empat partisi. Alasannya adalah hanya tersedia cukup byte di tabel partisi MBR untuk menampung empat entri. Kami juga menyadari bahwa biasanya ketika sebuah partisi dibuat pada satu hard drive, itu secara otomatis menjadi volume dan diberi huruf drive (mis., C:, D:, dll.). Volume dan partisi sama pada saat ini, meskipun dengan definisi yang berbeda. Seperti yang lainnya selalu ada pengecualian. Salah satu dari empat partisi yang ditentukan oleh tabel partisi hard drive dapat dibagi menjadi beberapa partisi logis (volume), sehingga memungkinkan dua volume atau lebih ada dalam satu partisi. Partisi yang dibagi ini disebut sebagai partisi yang diperluas. (Lihat Gambar 7.10.)



Gambar 7.10 Disk Drive dengan Extended Partition

Partisi yang diperluas dibuat untuk memungkinkan partisi atau volume logis tambahan. Karena partisi pada dasarnya menjadi volume saat dibuat, maka hard disk dibatasi hingga empat volume. Untuk mengatasi kekurangan ini partisi extended dibuat. Ini memungkinkan dirinya untuk dibagi lagi menjadi volume logis. Ingat, tabel partisi yang terdapat di dalam MBR masih hanya mendefinisikan empat partisi dan sejauh ini hanya ada empat partisi. Saat partisi extended dibuat, tabel partisi extended juga dibuat. Intinya, partisi yang diperluas mirip dengan disk drive dengan sendirinya — ia memiliki tabel partisi sendiri yang menunjuk ke satu atau lebih partisi (sekarang disebut partisi logis, berlawanan dengan empat partisi utama) yang seluruhnya terdapat di dalam partisi yang diperluas itu sendiri.

Tidak ada batasan jumlah volume logis selain fakta bahwa hanya ada 26 huruf dalam alfabet dan A, B, dan C telah diambil. Gambar 7.10 menunjukkan adanya lima jilid. Agar ada lima volume pada satu disk, partisi yang diperluas perlu dibuat. Seperti yang bisa kita lihat, ini ada sebagai partisi logis dan semuanya merupakan bagian dari partisi yang diperluas.

7.4 RINGKASAN

Penulis mengakhiri bab ini dengan diskusi tentang perluasan partisi untuk menunjukkan keberadaan data penentu partisi yang terdapat di dalam area hard drive selain tabel partisi, yang terdapat di dalam MBR. Partisi yang diperluas berisi "sub" atau catatan boot partisi yang diperluas untuk mengidentifikasi subdivisinya. Konsep mengidentifikasi dan mendefinisikan parameter lebih lanjut ini berlanjut dalam volume juga. Volume juga berisi sektor boot yang mengidentifikasi sistem file yang terkandung di dalamnya dan parameternya, sama seperti MBR berisi data yang mengidentifikasi dirinya sendiri untuk melakukan volume. Mengapa tinjauan dan pembahasan volume dan partisi begitu penting bagi pemeriksa forensik siber? Pemisahan dalam beberapa bentuk telah dibahas dalam tiga bab sebelumnya; mengapa menghabiskan begitu banyak waktu untuk meliput, meninjau, dan mendiskusikan partisi? Partisi sangat penting karena mengidentifikasi tata letak hard drive. Sebagai pemeriksa forensik dunia maya, sangat penting untuk memahami di mana data berada dan bagaimana data itu berada di sana.

Alat forensik yang tersedia untuk penyelidik saat ini secara otomatis melakukan sebagian besar fungsi untuk mengidentifikasi partisi, dan pemeriksa tidak harus melalui langkah-langkah yang melelahkan untuk mengidentifikasi di mana partisi dimulai dan diakhiri. Penyelidik telah berkembang untuk mengembangkan tingkat ketergantungan tertentu pada alat-alat ini. Namun, hapus alat tersebut dan bagaimana Anda dapat mengidentifikasi jenis dan ukuran partisi? Bagaimana Anda memahami semua 0 dan 1 tanpa mengetahui di mana data dimulai dan di mana berhenti? Sebelum Anda dapat menyelidiki dengan benar, Anda harus "menghitung" dengan benar. Bagaimana jika Anda dipanggil ke kursi saksi dan kemudian diminta untuk menjelaskan konsep-konsep tersebut? Menyatakan, “alat melakukannya untuk saya” tidak akan cukup, terutama tidak cukup untuk membuktikan keahlian atau pengetahuan Anda tentang “bagaimana” data diidentifikasi, ditemukan, dan diperiksa.

Pentingnya mengidentifikasi atau menghitung data dapat dilihat dalam kasus Ronelle Sawyer yang sedang berlangsung, yang menyelidiki apakah Jose McCarthy berpotensi terlibat dalam distribusi kekayaan intelektual organisasinya yang melanggar hukum kepada pesaing,

Janice Witcome, direktur pelaksana Perusahaan XYZ. Jose McCarthy, dalam upaya untuk menyembunyikan tindakannya, membuat dua partisi pada hard drive 80 GB miliknya; satu di mana dia gunakan untuk aktivitas sehari-hari dan yang kedua dia gunakan untuk menyimpan aktivitas jahatnya. Ketika dihadapkan pada kemungkinan tuntutan pidana, dia dengan cepat menghapus partisi kedua, sehingga membuatnya "tidak terlihat" oleh sistem operasi. Ronelle dapat dengan jelas melihat bahwa hard drive adalah 80 GB, tetapi bagaimana dia dapat mengidentifikasi ukuran partisi? Bagaimana dia bisa memperhitungkan semua 80 GB data? Jika partisi kedua tidak teridentifikasi, dia mungkin kehilangan banyak bukti; dalam hal ini, semua "bukti kriminal yang penting".

Sebagai penyelidik forensik dunia maya, Ronelle akan, dengan protokol, memeriksa tabel partisi dan hanya melihat satu partisi berukuran 60 GB, dan titik awal dan akhir dari satu partisi tersebut dapat ditentukan dengan cepat. Tanpa menggunakan alat forensik, jika Ronelle tidak memiliki pengetahuan tentang spesifikasi dan fungsi tabel partisi, bagaimana penyelidik kami dapat menjelaskan 20 GB yang hilang?

Tentu, ada cara lain untuk mengidentifikasi ruang yang hilang tetapi validitas pemahaman fungsi dan peran tabel partisi tetap ada. Ingatlah bahwa saat menghapus partisi, hanya entri tabel partisi yang dihapus. Semua data yang ada di dalam partisi tetap ada sampai ditimpa. Karena partisi tidak lagi digunakan dan tidak aktif, data yang ada di dalamnya tidak akan ditimpa. Jadi kemungkinan besar data ini akan tetap utuh dan dapat diambil kembali sebagai bagian dari penyelidikan. Mengidentifikasi akhir dari partisi pertama akan membantu Ronelle dalam menemukan titik awal dari partisi yang dihapus. Terakhir, partisi yang dihapus dapat dipulihkan dan semua data yang terkait dengan partisi kedua diambil, diakses, dan dianalisis. Memulihkan partisi, dan informasi data pentingnya, mungkin sulit tanpa pemahaman yang kuat tentang fungsi partisi.

Bab 8 berlanjut dengan perkembangan alami tentang bagaimana komputer "memasang" data dan membahas lebih detail tentang topik penting volume dan sistem file, dan hubungannya dengan penyelidikan forensik dunia maya secara keseluruhan.

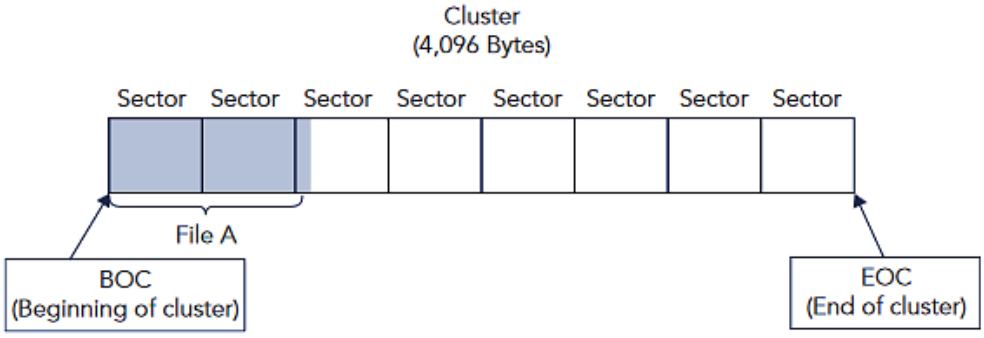
BAB 8
SISTEM FILE—FAT 12/16

Penulis melanjutkan bagaimana data disimpan secara elektronik. Kami membahas proses boot dan tabel partisi untuk mencontohkan konsep bagaimana data disimpan di hard drive. Kami mengikuti urutan boot komputer, karena ini adalah cara utama di mana data "dirangkai" menjadi informasi yang dapat kami gunakan, atau yang mungkin kami telusuri sebagai bagian dari penyelidikan. Bagian berikutnya dari "perakitan data" lebih logis dan benar-benar membawa data ke keadaan di mana kita benar-benar dapat mengakses dan menggunakannya. Bab ini mencakup sistem file dan konsep yang terkait dengan "pemasangan" data ke dalam informasi yang dapat diidentifikasi.

8.1 TINJAUAN TEKNOLOGI

Pada bab sebelumnya kita sudah mulai melewati batas antara penyimpanan fisik data dan penyimpanan logis data. Definisi physical adalah keadaan on/off sebenarnya dari bit dan lokasinya pada piringan hard drive, seperti yang didefinisikan oleh Cylinder, Head, and Sector (CHS) dan Logical Block Address (LBA), dengan partisi yang mendefinisikan lebih banyak fisik batas data karena mereka berada di disk itu sendiri. Kami membahas bagaimana bit fisik yang bersebelahan dirakit bersama menjadi konstruksi fisik yang disebut sektor. Sektor-sektor yang berurutan kemudian disusun secara logis menjadi apa yang disebut cluster. Seperti yang telah dibahas, cluster adalah unit penyimpanan logis yang terdiri dari setidaknya satu sektor.

Ketika sebuah file disimpan, secara otomatis menerima cluster penuh sebagai "wadah" untuk penyimpanannya. Jika file lebih kecil dari penampung, ia hanya menempati satu kluster. Jika file lebih besar dari cluster maka akan menempati cluster sebanyak yang diperlukan sampai isinya dialokasikan. Berikut adalah contoh singkat untuk memperkuat konsep penting ini. Misalkan sebuah cluster berisi delapan sektor; pada 512 byte per sektor ukuran total cluster akan sama dengan 4.096 byte (8 × 512). Mari ambil file dasar, File A, yang ukurannya total 1.040 byte. "File A" kami menempati satu cluster penuh. Menentukan berapa banyak sektor yang ditempati File A dalam cluster tunggal ini hanya melibatkan membagi ukuran File A (1.040 byte) dengan 512 byte per sektor. Kita melihat bahwa File A menggunakan dua sektor penuh dan 16 byte dari sektor ketiga dalam cluster (lihat Gambar 8.1).



Gambar 8.1 Pemanfaatan Sektor dari File yang Disimpan dalam Cluster

Cluster tidak perlu digunakan secara bersamaan, artinya file besar yang membutuhkan banyak cluster dapat menyimpan sebagian file ke satu cluster dan bagian lain ke cluster lain di tempat lain di hard drive. File besar seringkali perlu dipecah sedemikian rupa untuk menemukan tempat peristirahatan. Logikanya muncul sebagai satu file tetapi secara fisik terfragmentasi di seluruh drive. Kita sering mendengar ungkapan "defrag hard drive." Defragmentasi hard drive berarti benar-benar memindahkan data sehingga cluster secara fisik bersebelahan, sehingga tidak terfragmentasi. Ketika file sangat terfragmentasi, sistem perlu memasang kembali cluster tersebut sebelum menyajikan data. Hard drive harus berputar dan menarik komponen bersama-sama. Hal ini membutuhkan waktu dan, jika dikalikan, dapat menghambat kinerja dan efisiensi operasional secara keseluruhan.

8.2 SISTEM FILE

Sistem file adalah alat yang digunakan untuk menyimpan dan mengambil data di komputer. Ini adalah alat yang melacak alokasi cluster, dan memungkinkan hierarki direktori, folder, dan file. Sistem file menangani dan mengelola semua cluster yang terkandung dalam volume. Sistem file biasanya ditentukan selama pembuatan partisi; pada titik inilah partisi "menjadi" volume. Sistem file menentukan bagaimana dan di mana file ditempatkan pada hard drive, dengan tujuan mencoba mengoptimalkan kecepatan pengambilan data. Seperti yang telah dibahas, menyimpan file ke cluster yang berdekatan secara optimal lebih efisien untuk akses dan pengambilan data.

Saat mencari file, sistem operasi memerlukan sistem atau metode untuk menentukan di mana pada hard drive file tersebut ditempatkan. Bayangkan jika Anda ingin dokumen Microsoft Word yang disimpan di hard drive dianalogikan dengan buku di perpustakaan. Sangat menyenangkan dan nyaman bagi kami untuk mengklik ikon dan mengakses dokumen kami. Sebagai pengguna, kami tidak peduli di mana dokumen itu berada secara fisik di hard drive. Kami mungkin tahu di mana dokumen itu berada secara logis dalam struktur folder, tetapi kami tidak menyadari (dan memang benar), mengenai bit spesifik mana pada hard drive yang dialokasikan untuk dokumen individual ini. Ini bukanlah sesuatu yang perlu diperhatikan oleh pengguna akhir; namun, sangat penting bahwa sistem file mengetahui sebaliknya ketika kita mengklik ikon dokumen Word tidak akan terjadi apa-apa.

Untuk menemukan sebuah buku di dalam perpustakaan, seseorang cukup mencari nama buku (atau judul, penulis, dan/atau informasi terkait lainnya) di katalog kartu (lihat Gambar 8.2). Ini mengarahkan pembaca potensial ke lorong tertentu (lihat Gambar 8.3). Dan kemudian menunjuk ke lokasi persis buku tersebut di dalam perpustakaan (lihat Gambar 8.4).

Gambar 8.2 Sistem Katalog Kartu Perpustakaan (sekitar pertengahan abad kedua puluh)

Gambar 8.3 Lokasi Penyimpanan Fisik Buku di Perpustakaan

Gambar 8.4 Lokasi Tepat Buku yang Diinginkan di dalam Perpustakaan (lantai tertentu, lorong, rak, dan posisi di rak)

Bayangkan, misalnya, jika perpustakaan tidak memiliki sistem katalogisasi (seperti Sistem Desimal Dewey) untuk semua bukunya. Perpustakaan berisi ribuan dan ribuan buku. Siapa pun yang pernah menginjakkan kaki di perpustakaan dapat membayangkan betapa menakutkannya tugas menemukan buku tertentu tanpa mengetahui lokasi persisnya. Menemukan satu buku bisa memakan waktu berjam-jam, berhari-hari, atau bahkan berminggu-minggu, tergantung pada jumlah buku yang ada di perpustakaan. Sistem

katalogisasi yang digunakan dalam perpustakaan membuat pencarian lokasi yang tepat dari sebuah buku relatif tidak sulit (lihat Gambar 8.5).

Gambar 8.5 "Sistem File" Perpustakaan

Entri katalog akan berisi informasi tentang buku seperti penulis, judul, tanggal, dan lokasi di dalam perpustakaan. Sistem file di komputer harus menggunakan metode yang serupa, jika tidak, menemukan file tidak mungkin dilakukan. Sistem file komputer akan berisi informasi tentang file seperti nama file, ukuran, dan lokasi awal (alamat pada disk). Informasi yang terkandung dalam sistem pengarsipan ini sering disebut sebagai Metadata. Metadata dalam definisi paling sederhana adalah data tentang data. Ini pada dasarnya adalah data yang ada pada kartu fisik yang berada di dalam katalog kartu manual itu sendiri. Kartu tersebut tidak akan berisi seluruh isi buku (cerita), melainkan berisi judul, penulis, tanggal penerbitan, dan lokasi pasti buku tersebut di dalam perpustakaan.

8.4 METADATA

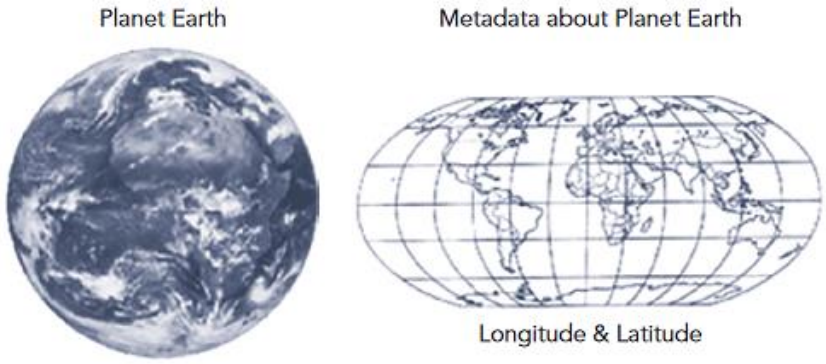
Kami telah membahas metadata dan akan membahasnya lagi karena relevan di sini dalam diskusi kami tentang sistem file. Metadata terdiri dari informasi yang mencirikan data. Intinya, metadata menjawab siapa, apa, kapan, di mana, mengapa, dan bagaimana tentang data/dokumen tersebut. Setiap kali dokumen elektronik dibuat, dibuka, atau disimpan, metadata diubah. Sistem operasi memerlukan sistem file agar dapat berfungsi, dan informasi tentang sistem file ini sebagian terkandung dalam metadata. Metadata digunakan untuk berbagai alasan, seperti untuk meningkatkan penerbitan, pengeditan, tampilan, pengarsipan, dan pengambilan. Beberapa metadata dapat diakses melalui perangkat lunak operasi atau antarmuka perangkat lunak induk. Misalnya, buka folder di sistem operasi Windows dan pilih tampilan "Detail". Informasi yang tersedia dan dapat dilihat meliputi nama, ukuran, jenis, dan

tanggal perubahan dokumen isinya. Buka tab "properti" pada dokumen MS Word dan informasi lebih lanjut tentang dokumen tersebut terungkap. Ini adalah metadata. Metadata lain hanya dapat diakses melalui editor HEX atau alat lain yang lebih khusus.

Berikut adalah beberapa contoh metadata yang dapat disimpan bersama dengan dokumen:

- Nama Anda
- Inisial Anda
- Nama perusahaan atau organisasi Anda
- Nama komputer Anda
- Nama server jaringan atau hard disk tempat Anda menyimpan dokumen
- Properti file lainnya dan informasi ringkasan
- Bagian objek OLE tersemat yang tidak terlihat
- Nama penulis dokumen sebelumnya
- Revisi dokumen
- Versi dokumen
- Informasi templat
- Teks tersembunyi
- Komentar

Konsep metadata dapat dilihat di banyak bidang sehari-hari selain bukti elektronik. Seperti yang telah kita bahas sebelumnya, data yang terkandung dalam katalog kartu adalah metadata. Perhatikan, misalnya, grafik planet Bumi yang sangat sederhana seperti yang ditunjukkan pada Gambar 8.6.



Gambar 8.6 Planet Bumi dan Metadata

Bujur dan lintang (peta pada umumnya) berisi metadata tentang planet kita, Bumi. Mereka buatan; lagipula, planet kita tidak memiliki garis kisi yang melewatinya, atau melapisinya. Metadata ini diperlukan untuk memungkinkan seseorang mengidentifikasi lokasi unik di planet ini, dan pada dasarnya dirancang oleh mereka yang perlu menavigasi lautan, yang terutama kekurangan fitur yang terlihat. Jika bukan karena metadata ini, kita mungkin tidak akan pernah memetakan Bumi secara keseluruhan dengan benar.

Metadata seringkali penting untuk mengakses dan mengidentifikasi objek yang didefinisikannya. Sama seperti peta yang membantu kita menemukan lokasi geografis yang tidak dikenal atau katalog kartu membantu menemukan satu buku dari ribuan, sistem file juga

membantu kita menemukan beberapa bit dari miliaran yang membentuk file. Ada sistem pengarsipan perpustakaan yang berbeda seperti Sistem Desimal Dewey atau sistem klasifikasi Perpustakaan Kongres. Hal yang sama berlaku untuk sistem file komputasi. Sistem operasi Microsoft Windows memiliki sistem file yang unik, seperti halnya sistem operasi berbasis Apple, Linux, dan Unix. Beberapa jenis sistem operasi komputer dengan sistem file yang sesuai ditunjukkan pada Tabel 8.1.

Tabel 8.1 Sistem Operasi Komputer dengan Sistem File yang Sesuai

Sistem operasi	Sistem File Digunakan
disket PC	FAT 12 saja
DOS	FAT 16 saja
Windows 3.x	FAT 16 saja
Windows 95	FAT 16 atau FAT32
Windows 98	FAT 16 atau FAT32
Windows ME	FAT 32
Windows 2000	FAT 16, FAT 32 atau NTFS
Windows XP	NTFS
Windows Server 2003	NTFS
Windows Vista	NTFS
Windows 7	NTFS
HFS	MAC
Ekst2, Ekst3	Linux
UFS	Unix

8.4 TABEL ALOKASI FILE (FAT) SISTEM FILE

Sistem pengarsipan FAT agak kuno dan digunakan di sistem operasi Microsoft sebelumnya. Ini dibahas di sini untuk menjelaskan dan mencontohkan konsep terbaik yang terlibat dalam sistem file. Sistem file tertentu apa pun kemungkinan besar akan menjadi usang, tetapi konsep umumnya layak untuk diperiksa. Yang penting di sini adalah memahami konsep cara kerja sistem file.

FAT digunakan untuk menempatkan file di cluster ruang kosong di hard drive. Ini tidak seperti kaset video, piringan hitam, atau bahkan CD, di mana file ditempatkan dalam urutan fisik dari awal sampai akhir. Setiap entri dalam Tabel Alokasi File berhubungan langsung dengan satu kluster, pada titik mana kluster dialokasikan ke data yang direferensikan dalam FAT.

Seperti yang telah dibahas, ada versi yang berbeda dari sistem pengarsipan FAT: FAT 12, 16, dan 32. Entri Tabel Alokasi File dapat sepanjang 12, 16, atau 32 bit sesuai dengan FAT 12, FAT 16, atau FAT 32. Entri tabel inilah yang digunakan untuk mengidentifikasi atau membatasi cluster. Satu file dialokasikan (atau disimpan) ke sebuah cluster, bahkan jika file tersebut tidak sepenuhnya memenuhi seluruh cluster. File besar tentu saja mungkin perlu dialokasikan ke beberapa cluster.

Sistem pengarsipan FAT terdiri dari tiga komponen utama:

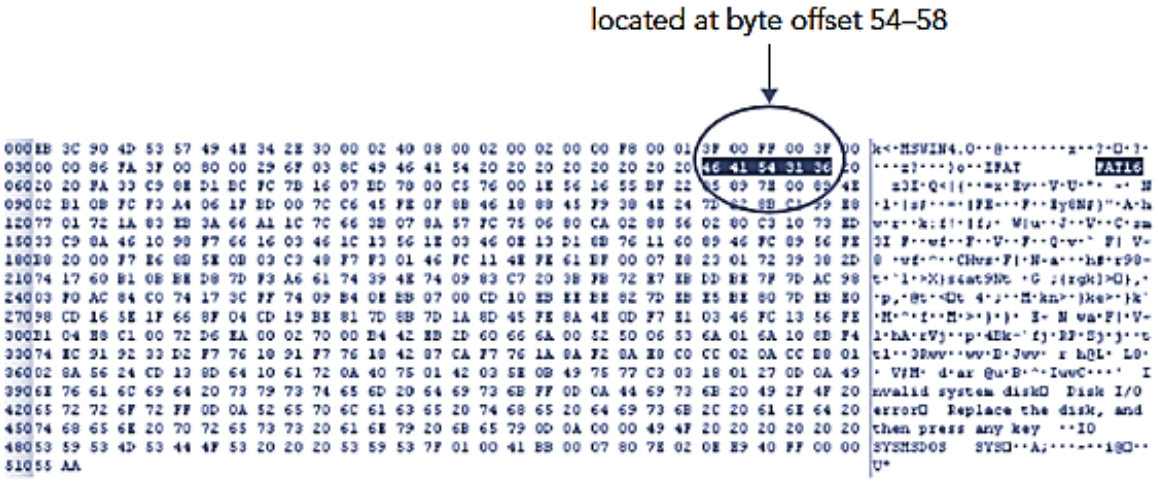
1. Rekam Boot Volume
2. Entri Direktori
3. Tabel Alokasi File

Rekam Boot Volume (VBR)

Catatan boot volume juga dikenal sebagai volume boot sector atau volume boot. Ini biasanya merupakan sektor pertama dari sebuah partisi. Karena Master Boot Record (MBR) mendefinisikan partisi disk fisik, VBR melakukan fungsi serupa untuk volume/partisi individual. Ini mendefinisikan tipe partisi (sistem file) dan menyimpan informasi parameter seperti byte per sektor dan sektor per cluster dalam volume.

Kami tidak membahas semua offset byte di dalam VBR, tetapi mendiskusikan yang membawa konsep utama ke cahaya. Untuk informasi tambahan tentang nilai offset byte tertentu dalam VBR dan diskusi lebih lanjut tentang VBR.

Kita lihat pada Gambar 8.7 bahwa byte offset 54–58 mengidentifikasi sistem file, dalam contoh ini, FAT 16. Jika Anda "mengubah" nilai HEX 46:41:54:31:36 (sebagai byte) ke ASCII seperti yang ditunjukkan pada Tabel 8.2, nilai HEX individu FAT 16 dikembalikan. Lihat nilai HEX individual yang ditandai pada Gambar 8.8.



Gambar 8.7 Jenis Sistem File Diidentifikasi pada Byte Offset 54–58

Tabel 8.2 Offset Byte VBR 54–58 Nilai HEX dan Setara ASCII

Offset Byte	Nilai HEX	Setara Karakter ASCII
54	46	F
55	41	A
56	54	T
57	31	1
58	36	6

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

Gambar 8.8 Bagan ASCII Standar/Tabel ASCII—Konversi HEX ke Kode Desimal

Byte per Sektor

Dalam Bab 4 dan 5 kita membahas fakta bahwa biasanya ada 512 byte per sektor. Ini adalah pengetahuan yang cukup standar; namun, sama saja, fakta ini tetapi harus didefinisikan di suatu tempat agar OS mengetahui bahwa ini adalah byte per ukuran sektor yang digunakan sistem.

Di FAT yang ada di Volume Boot Record, dengan offset byte 11–12. Dalam contoh kami, editor HEX yang ditunjukkan pada Gambar 8.9, kami melihat nilai HEX 00 02. Jika kita mengeluarkan kalkulator ilmiah lama untuk menentukan desimal yang setara dengan HEX 00 02 (atau hanya 2) kita menemukan jawabannya adalah 2. Tapi, tunggu sebentar! Dua byte per sektor? Bukankah cukup standar untuk memiliki 512 byte per sektor? Nilai HEX benar. Ingat, itu semua berkaitan dengan bagaimana sistem operasi menginstruksikan kode untuk memproses dan menangani data, dan dalam hal ini endianness penting karena dibalik, dari persepsi membaca nilai dari kiri ke kanan (big endian) untuk membaca nilai data dari kanan ke kiri atau dalam hal ini, little endian. Oleh karena itu, nilai HEX adalah 02 00, atau hanya 200. Sekali lagi, kembali ke kalkulator kita, dan kita melihat bahwa ekuivalen desimal dari HEX 200 memang 512. Jadi ada 512 byte per sektor!

Gambar 8.9 Menentukan Byte per Sektor dari VBR pada Byte Offset 11–12

Sektor per Cluster

Sektor per cluster juga didefinisikan di sini di VBR. Nilai ini ditentukan dalam offset sektor 13. (Lihat Gambar 8.10.) Nilai HEX offset sektor 13 adalah 40 (lihat Gambar 8.10). Namun, jika kita mengonversi HEX 40 terlebih dahulu ke persamaan desimalnya, kita mendapatkan 64, kemudian desimal 64 ke persamaan ASCII-nya; kita mendapatkan "@." Namun, nilai ini (HEX 40) sebenarnya digunakan untuk menjalankan fungsi matematika. Jadi, kami mengonversi nilai HEX 40 menjadi nilai desimal (sekali lagi kembali ke kalkulator ilmiah kami). Setelah melakukan perhitungan dengan kalkulator ilmiah, kita melihat bahwa nilai HEX 40 menghasilkan desimal yang setara dengan 64.

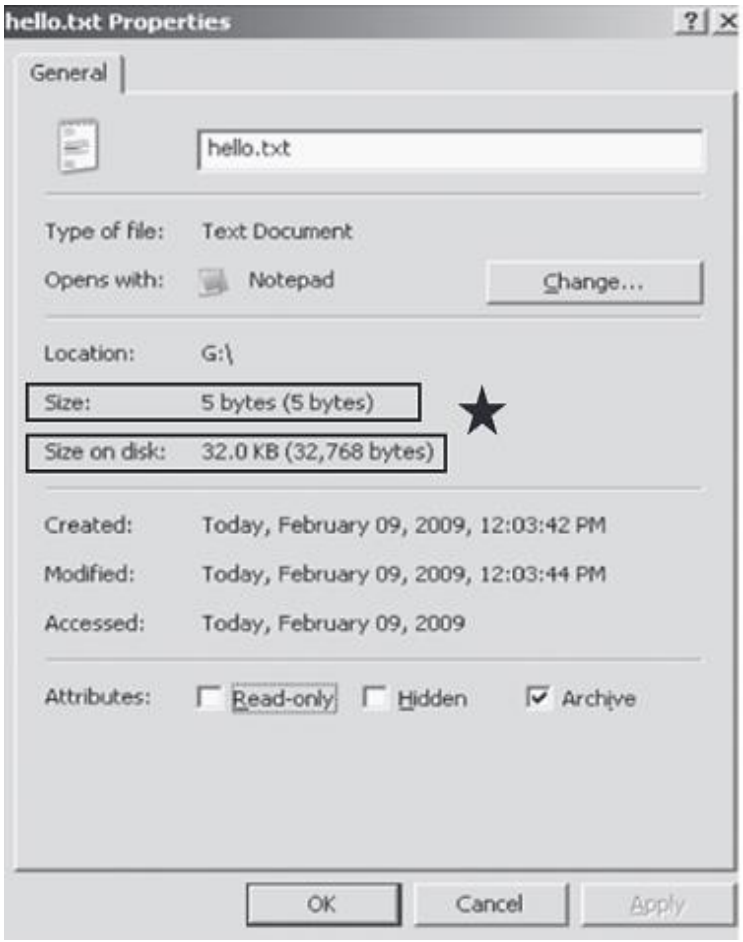
Gambar 8.10 Menentukan Cluster per Sektor dari VBR pada Byte Offset 13

Seperti yang telah kita singgung, biasanya ada 512 byte per sektor. Ini cukup umum dan dengan ukuran sektor VBR contoh kami pada Gambar 8.9, kami telah menentukannya menjadi 512 byte. Kami sekarang telah mengidentifikasi berapa banyak sektor yang ada per cluster, sekali lagi dalam contoh kami (Gambar 8.10), menjadi 64. Jika kita mengalikan jumlah byte per sektor dengan jumlah sektor per cluster, kita mendapatkan ukuran total cluster, dalam byte. Dengan demikian:

(Byte per sektor) × (Sektor per cluster) atau 512 × 64, yang menghasilkan nilai 32.768 byte per cluster.

Sebuah cluster penuh kemudian, dalam contoh kita, adalah 32K byte.

Untuk memverifikasi ini, yang perlu kita lakukan hanyalah melihat properti dokumen, dokumen yang telah disimpan ke sistem file FAT 16. Lihat Gambar 8.11, di mana kita telah membuat sebuah dokumen bernama “hello.txt.” Jika kita melihat properti dokumen, kita melihat bahwa ukuran dokumen adalah 5 byte, dan ukuran pada disk (atau alokasi cluster) adalah 32.768 byte.



Gambar 8.11 Jendela Properti Dokumen untuk Dokumen hello.txt

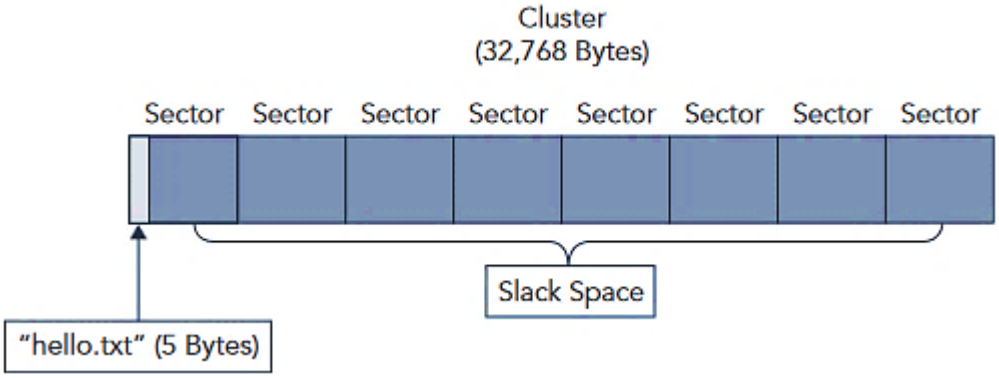
8.5 KENDUR

Dokumen pada Gambar 8.11 dialokasikan ke setidaknya satu cluster dan dalam hal ini hanya satu; namun, kami melihat penggunaan ruang yang sangat tidak efisien. Seperti yang telah kita bahas dalam tinjauan teknologi di awal bab ini, saat sebuah file ditulis ke drive, file tersebut secara otomatis menerima minimal satu cluster.

Dalam contoh ini, file "hello.txt" akan dialokasikan ruang disk sebesar 32KB, terlepas dari ukurannya. Jadi, dokumen teks kecil berukuran 5 byte akan mendapatkan seluruh cluster 32.768 byte. Seperti yang dapat Anda bayangkan, sistem file FAT 16 tidak efisien dalam penggunaan ruangnya. Apa yang terjadi pada ruang yang tersisa? Tidak ada apa-apa! Itu dialokasikan ruang, karena merupakan bagian dari cluster yang ditugaskan ke file "hello.txt", tetapi tidak diperlukan karena ukuran file "hello.txt" yang kecil.

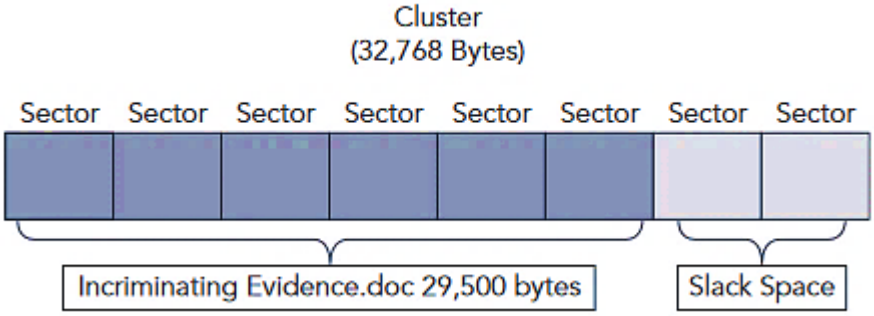
Ruang tak terpakai yang terdapat di dalam kluster yang dicadangkan untuk "hello.txt" disebut sebagai ruang kendur. Itu dialokasikan ruang tetapi tidak berisi konten dari file yang

disimpan "hello.txt." Ruang kendur akan berisi sisa-sisa data yang disimpan di cluster yang dialokasikan sebelumnya. (Lihat Gambar 8.12.)



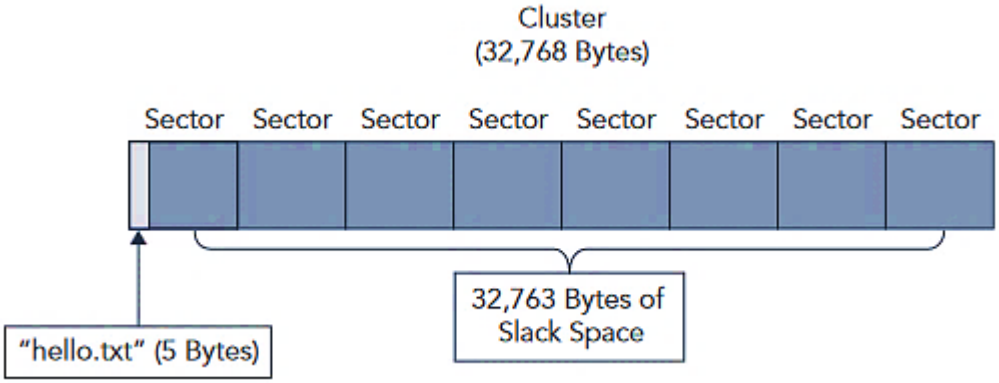
Gambar 8.12 Slack Space (alias File Slack)

Misalnya, dokumen berukuran 29.500 byte bernama "Bukti yang memberatkan. doc" telah dialokasikan ke cluster yang sama sebelumnya. (Lihat Gambar 8.13.) Dokumen "Bukti yang memberatkan.doc" itu dikirim ke tempat sampah dan kemudian tempat sampah dikosongkan, sehingga secara resmi membuat cluster itu tersedia atau tidak terisi.



Gambar 8.13 Bukti yang Memberatkan.doc

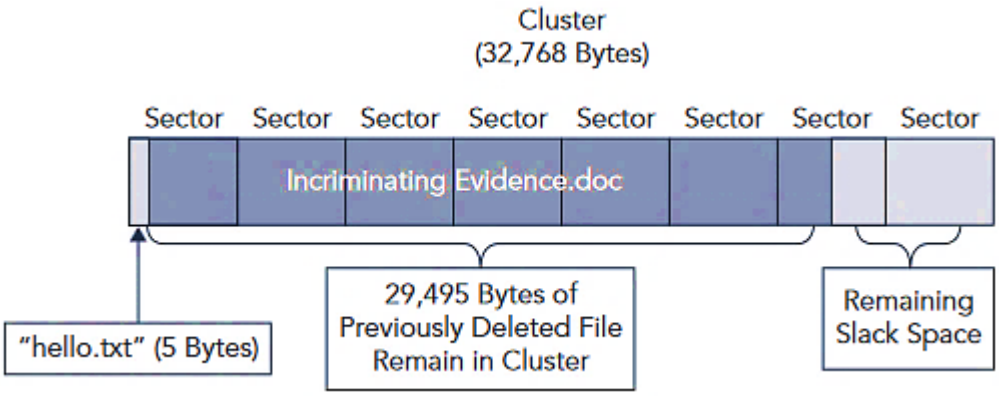
Kemudian, file "hello.txt" disimpan dan dialokasikan ke cluster yang sama. Karena "hello.txt" hanya berukuran 5 byte, ia hanya membutuhkan lima byte pertama dari cluster tersebut untuk memuat dirinya sendiri. Ini kemudian akan meninggalkan 32.763 byte file kendur. (Lihat Gambar 8.14.)



Gambar 8.14 32.763 Byte Ruang Slack

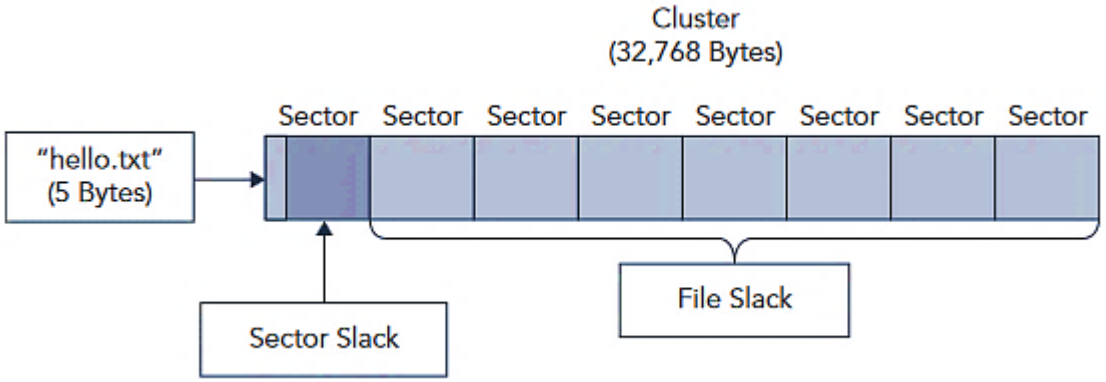
Kami menentukan ini dengan mengetahui jumlah total byte per cluster (32.768) (yaitu, ukuran cluster), dikurangi lima byte untuk file baru, "hello.txt", memberi kami sisa jumlah byte yang tersisa di cluster asli yang dialokasikan ke file "hello.txt", 32.763 byte, yang sekarang disebut sebagai ruang kendur. File sebelumnya yang disimpan ke cluster, "Incriminating evidence.doc" berukuran 29.500 byte.

Oleh karena itu, sebagian besar dari file ini juga akan tetap berada dalam file yang sekarang kendur. Kami menentukan ini dengan mengetahui jumlah total byte (29.500 byte) dari file lama, "Bukti yang memberatkan.doc" dikurangi lima byte dari file baru, "hello.txt," memberi kami 29.495 byte ruang kendur, yang akan berisi sisa-sisa file sebelumnya untuk menempati ruang ini: "Bukti yang memberatkan.doc." (Lihat Gambar 8.15.) Seperti yang dapat Anda bayangkan, mungkin ada beberapa data berharga yang terkandung dalam 29.495 byte ruang kosong, terutama dengan dokumen bernama “Bukti yang memberatkan.doc.” Slack space dapat dipecah lebih lanjut menjadi File Slack dan Slack Sektor. Ada perbedaan tetapi keduanya berkaitan dengan ruang kendur yang dijelaskan.



Gambar 8.15 29, 495 Byte File yang Dihapus Tersisa di Ruang Slack

Ketika sebuah file disimpan, itu dialokasikan ke sebuah cluster. Seperti yang telah dibahas, cluster terdiri dari sektor. Saat disimpan, file disimpan ke sektor pertama cluster. Jika ukuran file melebihi ukuran sektor (512 byte), maka disimpan ke sektor berikutnya, dan cluster berikutnya, dan selanjutnya, dan seterusnya, hingga seluruh file disimpan. Sektor yang tidak digunakan dalam sebuah cluster sering disebut sebagai file slack sedangkan byte yang tidak digunakan dalam suatu sektor sering disebut sebagai sector slack. (Lihat Gambar 8.16.) Untuk melanjutkan contoh kita: 32.768 byte adalah ukuran cluster kita, dikurangi lima byte dari file baru, "hello.txt", sama dengan 32.763 byte file slack di cluster ini. Dengan 512 byte sebagai ukuran sektor, kurang dari lima byte file baru, "hello.txt", kami sekarang memiliki 507 byte yang tidak terpakai di sektor tersebut, yang kami sebut sebagai kelonggaran sektor.



Gambar 8.16 Sektor dan File Slack

Ruang kendur akan dipertahankan (tidak ditimpa) kecuali:

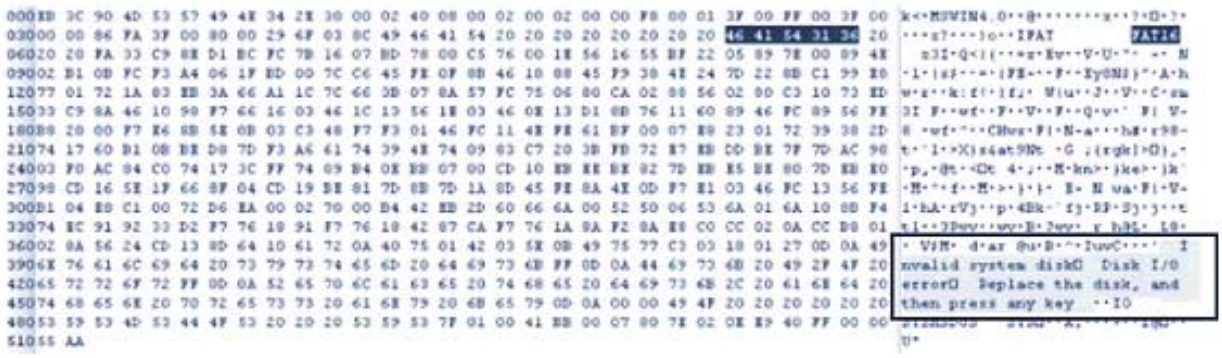
1. File yang ditetapkan ke ruang yang dialokasikan ("hello.txt") diubah dan/atau dibuat lebih besar ukurannya, sehingga menulis ke ruang kendur, (ruang yang dialokasikan).
2. File, "hello.txt" dihapus dari recycle bin, sehingga membuat seluruh cluster tersedia atau tidak terisi. Hal ini dapat menyebabkan file lain dialokasikan ke cluster tersebut. Catatan — jika file baru itu berukuran sama atau lebih kecil dari file sebelumnya ("hello.txt"), sebagian besar data yang ada di dalam ruang kendur dapat tetap dipertahankan.

8.6 CATATAN TINJAUAN HEX

HEX dapat dikonversi ke ASCII seperti yang terlihat di editor HEX. Namun, tidak semua HEX akan muncul sebagai teks atau ASCII yang dapat dibaca. Sebagian besar kode pemrograman tidak memiliki presentasi ASCII yang dapat dibaca manusia. Faktanya, satu-satunya HEX yang memiliki representasi yang dapat dibaca adalah nilai HEX yang dapat dipanggil oleh kode perangkat lunak untuk berbagai tujuan tampilan.

Misalnya, ketika masuk ke perangkat lunak Manajemen Disk dan melihat informasi partisi disk, kita akan melihat jenis sistem file adalah FAT 16. Bagaimana komputer mengetahui untuk menampilkan informasi ini dan, lebih tepatnya, bagaimana komputer mengetahui apa yang harus ditampilkan? Kode pemrograman Manajemen Disk akan memberi tahu sistem untuk beralih ke byte offset 54–58 di sektor pertama partisi dan menampilkan informasi tersebut dalam ASCII, "FAT16," seperti yang ditampilkan pada Gambar 8.17. Nilai HEX ini tidak memiliki fungsi matematika; mereka digunakan untuk tujuan tampilan. Informasi yang akan ditampilkan biasanya dapat dibaca di ASCII.

Perhatikan misalnya pesan di Volume Boot Record Gambar 8.17, "Invalid System Disk", "Disk I/O error", dan "Replace the disk, and press any key". Teks ini diambil dengan kode dan ditampilkan sebagai pesan kesalahan saat terjadi kesalahan saat boot. Karena alasan inilah maka dapat dibaca, karena dapat dipanggil dengan kode dan ditampilkan sebagai pesan. Biasanya, ASCII terbaca yang terlihat di editor HEX tidak melakukan perhitungan matematis atau berfungsi sebagai kode pemrograman; itu hanyalah teks yang dapat diekstraksi dan ditampilkan oleh sistem.



Gambar 8.17 Nilai ASCII Entri HEX di VBR

8.7 ENTRI DIREKTORI

Bagian kedua dari sistem Pengarsipan FAT, adalah entri direktori. Ingatlah bahwa sistem pengarsipan FAT terdiri dari tiga komponen utama:

1. Rekam Boot Volume
2. Entri Direktori
3. Tabel Alokasi File

Setiap file dan folder/direktori direferensikan dalam entri 32-byte terpisah yang disebut entri direktori. Entri direktori unik ada untuk setiap file dan direktori yang disimpan di disk. Setiap entri direktori berisi informasi seperti:

1. Nama file dan direktori. Misalnya: "hello.txt".
2. Metadata waktu dan data.
3. Lokasi—nama file harus ditautkan ke data aktual yang terdiri dari file dan oleh karena itu harus ada atribut yang menunjuk ke tempat asal data sebenarnya, yaitu cluster awal.
4. Ukuran file—panjangnya.

Entri ini terletak di cluster yang dialokasikan ke direktori induk file.

Gambar 8.18 adalah contoh entri untuk dokumen contoh kami, “Bukti yang memberatkan.doc,” yang direpresentasikan dalam ASCII. Perhatikan lagi setiap entri panjangnya 32 byte. Jumlah total entri direktori yang tersedia ditentukan saat volume diformat dan ditetapkan pada 32 sektor dalam FAT 12/16. Jadi, 32 sektor dialokasikan untuk entri direktori dan setiap entri direktori itu sendiri panjangnya 32 byte.

Byte Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ASCII	I	n	c	r	i	m	~	1	D	O	C		·	·	N	Q	F	:	I	:	·	·	O	Q	F	:	·	·	·	j	·	·
HEX	49	6E	63	72	69	6D	7E	31	44	4F	43	20	00	8C	4E	51	46	3A	49	3A	00	00	4F	51	46	3A	02	00	00	6A	00	00

Gambar 8.18 Entri Direktori untuk Dokumen “Bukti yang memberatkan.doc”

Berikut gambaran singkat tentang nilai offset byte entri direktori pada Gambar 8.18:

- Offset Byte 0. Karakter pertama dari nama file atau byte status. Saat file dihapus, karakter ini akan diganti dengan "~" yang memberi tahu sistem bahwa ruang yang sebelumnya

ditempati oleh file tersebut sekarang tersedia, atau tidak terisi. Ini adalah byte penting karena ini yang mengidentifikasi file sedang dihapus, atau lebih tepatnya, ini mengidentifikasi ruang sebagai tidak terisi. Catatan: sisa 31 byte dari entri direktori dan file sebenarnya tetap utuh sampai ruang akhirnya dialokasikan kembali dan berpotensi ditimpa.

- Offset Byte 1–7. Nama file dilanjutkan. Di sini kita melihat batasan sistem file FAT dan disebut sebagai Short File Names (SFN). Delapan karakter yang dicadangkan untuk nama file sangat terbatas. Nah, sistem file mempertimbangkan ini dan menambahkan nama file. The "~" di Byte Offset 6 menyiratkan nama singkat. Angka 1 dalam Byte Offset 7 menyiratkan bahwa ini adalah dokumen pertama yang berisi enam karakter pertama tersebut. Misalnya, dokumen lain dibuat dengan nama "Incrimination Study.doc". Itu akan berisi enam karakter pertama yang sama. Karena keterbatasan FAT dan untuk membedakan antara dokumen-dokumen ini, FAT akan menamai dokumen kedua, "Incrimination Study.doc," sebagai Incrim~2.
- Offset Byte 8–10. Ekstensi file tiga karakter. Ekstensi digunakan sebagian besar oleh Sistem Operasi Microsoft dan tidak diperlukan atau digunakan oleh sistem operasi lain. Untuk informasi lebih lanjut mengenai ekstensi dan jenis file, silakan merujuk kembali ke Bab 4.
- Offset Byte 11. Atribut—Hanya baca, tersembunyi, dan sebagainya.
- Offset Byte 12–13. Disimpan.
- Offset Byte 14–17. Buat waktu dan tanggal file yang disimpan sebagai stempel waktu MS_DOS 32-bit (dibahas di Bab 11).
- Offset Byte 18–19. Terakhir diakses, hanya tanggal, tanpa waktu.
- Offset Byte 20–21. Dua byte tinggi dari cluster awal FAT 32; FAT 12/16 akan memiliki nol.
- Offset Byte 22–25. Waktu dan tanggal file terakhir ditulis lagi sebagai stempel waktu MS-DOS 32-bit (dibahas dalam Bab 11).
- Offset Byte 26–27. Cluster awal untuk FAT 12/16; dua byte rendah dari cluster awal untuk FAT 32.
- Offset Byte 28–31. Ukuran dalam byte file. Akan menjadi 0 untuk direktori.

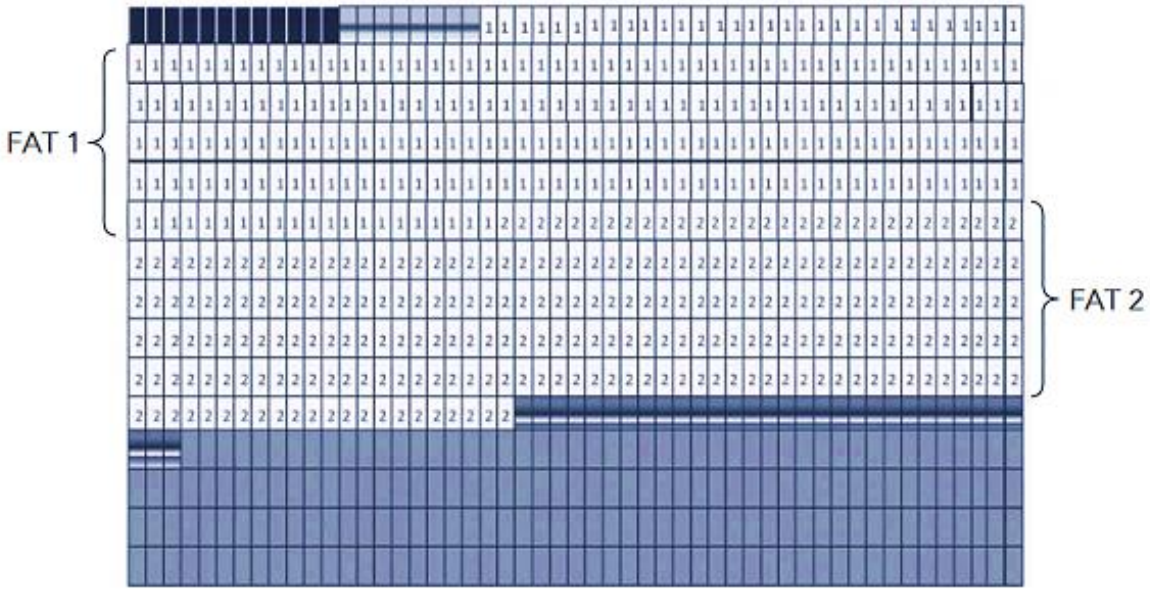
8.8 TABEL ALOKASI FILE (FAT)

Ingat, Sistem Pengajuan FAT terdiri dari tiga komponen utama:

1. Rekam Boot Volume
2. Entri Direktori
3. Tabel Alokasi File (FAT)

Berikut ini akan kita bahas dan telaah bagian ketiga dari sistem pengarsipan FAT ini. Nomenklatur untuk mendeskripsikan dan menulis berbagai jenis FAT dapat ditampilkan sebagai FAT12, misalnya, tanpa spasi, atau FAT 12, dengan spasi. Either way diterima dan digunakan sama. FAT dapat dianggap sebagai peta dari semua cluster (cluster, Anda ingat, adalah unit terkecil yang digunakan untuk menyimpan file) pada hard drive. FAT berisi entri untuk setiap cluster yang tersedia di disk. FAT melacak cluster mana yang dialokasikan atau tidak dialokasikan (tersedia atau tidak tersedia), melacak unit yang dialokasikan (atau cluster) yang berisi file atau data. Sering kali sebuah dokumen melebihi satu unit alokasi (satu cluster)

dan perlu disimpan dalam beberapa unit alokasi yang tidak bersebelahan, atau cluster; itu adalah FAT yang akan menghubungkan kelompok-kelompok ini. FAT menemukan cluster yang mengikuti cluster awal untuk setiap file atau direktori yang diberikan. Seperti yang telah kita diskusikan, file tidak harus disimpan secara fisik di hard drive. Oleh karena itu tugas FAT untuk melacak urutan cluster tersebut yang akan digabungkan (atau digabungkan) untuk membuat file yang disimpan. FAT juga melacak cluster yang buruk. Penggambaran visual dari FAT ditunjukkan pada Gambar 8.19. Dalam sistem pengarsipan FAT, FAT secara default diidentifikasi sebagai FAT1. FAT mempertahankan salinannya sendiri, diidentifikasi sebagai FAT2, dan disimpan segera setelah masuknya FAT1. FAT untuk FAT 12, 16, atau 32 dimulai di lokasi yang ditentukan oleh struktur drive tersebut, dan panjang tabel bergantung pada ukuran dan pemformatan disk. Ukuran entri FAT tergantung pada versi FAT. Faktanya, versi FAT (FAT 12, 16, atau 32) dinamai berdasarkan jumlah bit yang terkandung dalam setiap entri FAT:



Gambar 8.19 Tabel Alokasi File

FAT 12 memiliki entri 12-bit, FAT 16 memiliki entri 16-bit, dan FAT 32 memiliki entri 32-bit. Ini secara alami memberlakukan batasan berdasarkan jumlah bit dalam FAT, tetapi bagaimana caranya? Mari kita lihat lebih dekat setiap jenis FAT.

FAT 12

Setiap entri berukuran 12 bit, dan masing-masing entri 12-bit ini di FAT mewakili cluster sebenarnya, unit alokasi terkecil pada disk. Tabel 8.3 memberi tahu kita berapa banyak nilai atau kemungkinan hasil yang dapat dicapai dengan 12 bit. Saat kita menambah jumlah bit, akan lebih mudah untuk kesederhanaan dan kemudahan pemahaman untuk menggunakan kalkulator dan hanya menghitung 212. Ingat, biner adalah Basis 2! Hanya ada 4.096 nilai yang mungkin dengan 12 bit. Oleh karena itu, jika setiap entri mewakili sebuah cluster dan hanya ada cukup bit (12) untuk mewakili 4.096 nilai unik, maka maksimal hanya 4.096 cluster yang dapat dicapai dalam FAT 12.

Tinjauan Teknologi

Tinjauan singkat tentang pengkodean Basis 2 akan membantu menjernihkan poin ini. Seperti yang telah kita bahas sebelumnya, kita tidak bisa merepresentasikan empat musim hanya dengan satu bit.

Tabel 8.3 Nilai atau Kemungkinan Hasil yang Dapat Dicapai dengan 12 Bit

Jumlah Bit	Kemungkinan Jumlah Nilai atau Hasil
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1,024
11	2,048
12	4,096

Dengan satu bit kita dapat merepresentasikan dua nilai, seperti on/off, stop/go, atau hijau/merah (mis., 0 = merah, 1 = hijau). Untuk mewakili empat nilai, seperti musim, kita membutuhkan minimal dua bit sehingga setiap musim dapat diberi nilai unik (mis., 00 = musim semi, 01 = musim panas, 10 = musim gugur, 11 = musim dingin) .

Jika kami menambahkan musim kelima, kami tidak akan dapat merepresentasikannya dengan nilai dua bit yang unik. Kami telah menggunakan keempat nilai biner yang mungkin dengan dua bit. Kami pada dasarnya berurusan dengan masalah yang sama di sini. Sekalipun hard drive memiliki lebih banyak ruang, katakanlah satu terabyte, jika diformat dengan sistem pengarsipan FAT 12, hard drive hanya dapat memuat 4.096 cluster, tidak lebih.

Jika Anda tidak dibatasi oleh ukuran di mana Anda dapat membuat cluster maka Anda secara teoritis akan membagi satu drive terabyte menjadi 4.096 "potongan" (atau unit data misalnya), masing-masing "potongan" kemudian menjadi sebuah cluster. Setiap cluster akan berukuran sekitar 200MB. Bahkan jika ini mungkin, itu akan menjadi penggunaan ruang yang sangat tidak efisien. Seperti yang telah kita diskusikan, setiap file disimpan ke klusternya sendiri, sehingga setiap file akan diberi ruang 200MB. Pada akhirnya hanya 4.096 file yang dapat disimpan ke drive yang diformat sebagai FAT 12.

Sekali lagi, ada 4.096 kemungkinan hasil dengan bilangan bulat 12-bit. Hasilnya, dalam hal ini, mewakili jumlah cluster yang mungkin. Jika Anda mencoba menambahkan klaster ke-4.097, hal itu akan serupa dengan menetapkan nilai ketiga (misalnya lampu kuning) ke bilangan bulat 1 bit; hanya ada dua kemungkinan nilai dengan 1 bit—dalam contoh kita, merah dan hijau (yaitu, 0 = merah, 1 = hijau, ? = kuning. Tidak ada nilai tersisa untuk kuning).

Sebenarnya, hanya ada 4.084 klaster yang dimungkinkan dengan FAT 12. Bit tertentu dicadangkan, yang tidak dapat menyimpan data; oleh karena itu, pada kenyataannya, ada batasan hanya 4.084 file yang dapat disimpan dalam volume yang diformat dalam FAT 12. Rasa FAT lainnya juga memiliki batasan ukuran masing-masing karena batasan FAT: FAT 16, misalnya, pada 216 hanya dapat menampung maksimal 65.536 cluster. FAT 32, di 232 dapat menampung maksimal 4.294.967.296 cluster. Dengan FAT 12, beberapa bit yang digunakan dicadangkan, sehingga jumlah cluster maksimum yang mungkin sebenarnya akan lebih sedikit. Di dalam FAT itu sendiri, setiap nilai 16-bit dalam FAT dari sistem file FAT 16 akan mewakili satu kluster. Demikian pula, setiap nilai 32-bit dalam FAT dari sistem file FAT 32 akan mewakili satu kluster. Persamaan desimal yang sesuai dari nilai-nilai biner tersebut (12-, 16-, atau 32-bit), seperti yang direpresentasikan dalam FAT, adalah deskriptif dari cluster unik. Gambaran singkat beberapa nilai kunci dalam FAT disajikan pada Tabel 8.4.

Tabel 8.4 Nilai Kunci dalam FAT

Keterangan	**Nilai**
Tidak dialokasikan	Diwakili oleh 0.
Dialokasikan	Nilai akan diwakili oleh cluster berikutnya yang digunakan oleh file.
Alokasi Terakhir	Cluster terakhir yang digunakan oleh file. Biasanya terlihat dengan nilai HEX “End of File” dari Fs. Pengecualian: FAT 12—nilai HEX lebih besar dari nilai HEX FF8 (12-bit) FAT 16—nilai HEX lebih besar dari nilai HEX FF F8 (16-bit) FAT 32—nilai HEX lebih besar dari nilai HEX FF FF FF F8 (32-bit)
Kluster Buruk	Terlihat dengan nilai HEX yang diakhiri dengan F7: FAT 12—FF7 FAT 16—FF F8 FAT 32—FF FF FF F7

Bagaimana Cara Kerja FAT?

File dipanggil dengan nama dan jalur file, misalnya saat pengguna mengklik ikon dokumen yang disimpan di desktop. Jalur penyimpanan mengarah ke lokasi direktori induk di hard drive. Di sinilah entri direktori untuk file itu berada. Sistem operasi (OS) mencari di direktori induk ini dan membaca entri direktori. Entri direktori menyediakan cluster awal (memulai lokasi fisik pada hard drive) dan ukuran file. OS kemudian pergi ke cluster awal dan mulai membaca data. Itu hanya membaca data di dalam cluster hingga ukuran file, lalu OS berhenti membaca. Data lain apa pun dalam kluster itu (data ini yang sebelumnya kami sebut sebagai slack) diabaikan karena persyaratan panjang telah terpenuhi. Jika ukuran file melebihi

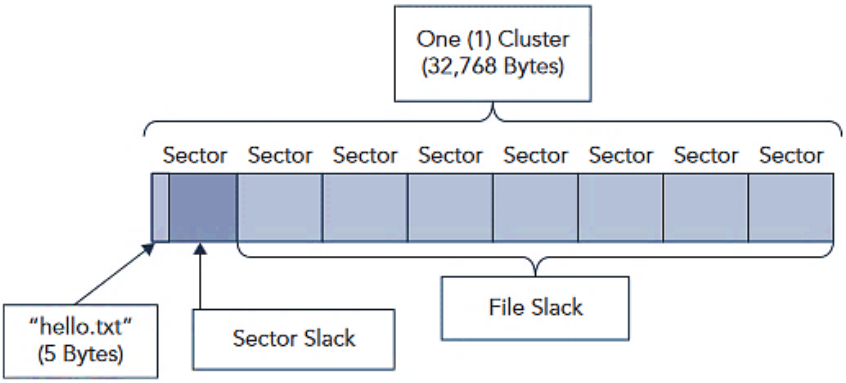
satu kluster, maka FAT diperlukan untuk menghubungkan kluster ini bersama-sama (ingat, FAT diperlukan setiap kali ukuran file melebihi satu kluster).

8.9 BAGAIMANA UKURAN CLUSTER DITENTUKAN?

Jika kita mengalikan jumlah byte per sektor dengan jumlah sektor per cluster, kita mendapatkan ukuran total cluster, dalam byte. Dengan demikian:

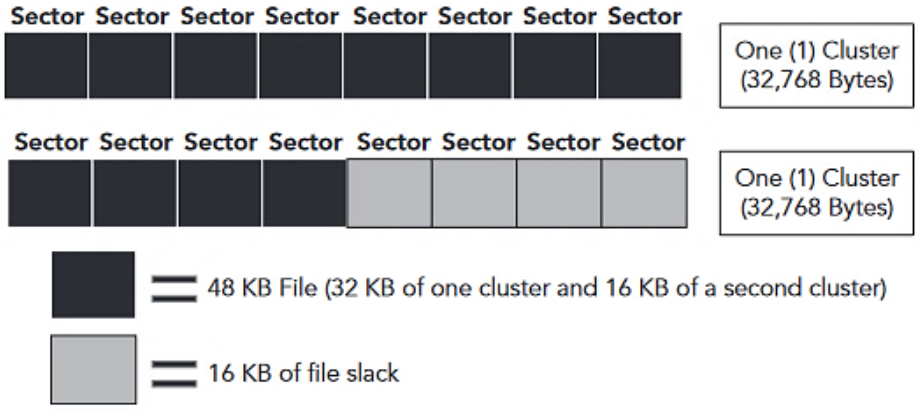
(Byte per sektor) × (Sektor per cluster) atau 512 × 64, yang menghasilkan nilai 32.768 byte per cluster.

Sebuah cluster penuh kemudian, dalam contoh kita, adalah 32 KB. Ingatlah bahwa dokumen kita bernama "hello.txt," yang telah dibahas sebelumnya (lihat Gambar 8.11), panjangnya hanya 5 byte. Ukuran cluster yang digunakan adalah 32 KB (sebagaimana didefinisikan oleh sistem file FAT 16 seperti yang terlihat di Volume Boot Record). Oleh karena itu, dokumen hanya membutuhkan (atau ditempati) satu cluster, yang ditentukan oleh satu dokumen lima byte yang menempati satu sektor 512 byte, dan slack sektor terkait untuk mengisi bagian yang tidak terpakai dari sektor pertama dan kemudian mengajukan slack ke isi bagian yang tidak terpakai dari sektor yang tersisa, membentuk cluster tunggal. (Lihat Gambar 8.20.)



Gambar 8.20 File 5 KB Menempati Satu Cluster

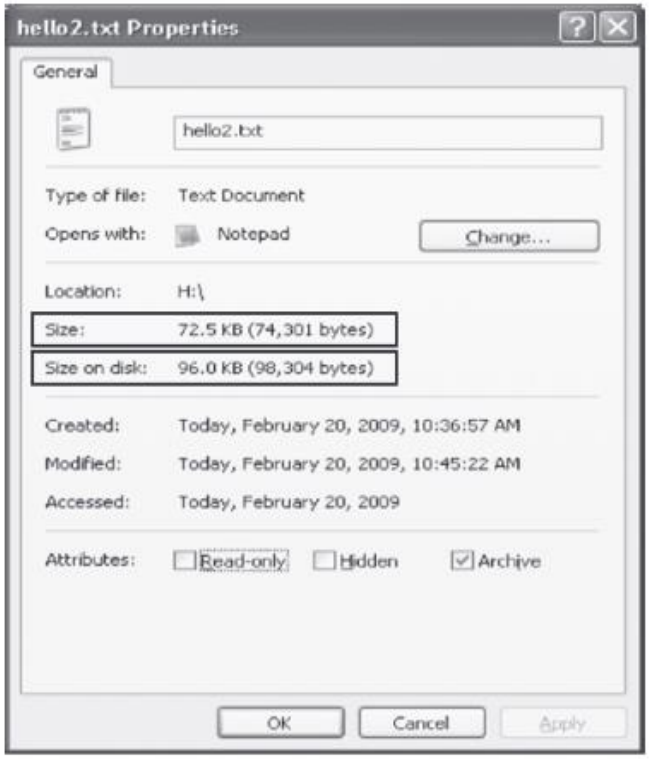
Jadi, file berukuran 48 KB akan membutuhkan dua cluster dan oleh karena itu penggunaan FAT. File 48 KB akan menempati 32 KB dari cluster pertama dan hanya 16 KB dari cluster kedua. (Lihat Gambar 8.21.)



Gambar 8.21 File 48 KB Memerlukan Lebih dari Satu Cluster

8.10 UKURAN CLUSTER YANG DIPERLUAS

Untuk tinjauan lanjutan kami tentang klaster, ukuran klaster, dan bagaimana ukuran klaster ditentukan, kami mengambil dokumen contoh awal kami "hello.txt" dan (a) menamainya menjadi "hello2.txt" dan (b) meningkatkan ukurannya secara substansial sehingga bahwa itu akan membutuhkan lebih banyak ruang yang dialokasikan (misalnya, cluster). Jika kita menelusuri file dan mengklik kanan pada dokumen untuk melihat propertinya, kita melihat bahwa ukuran fisiknya 72,5 KB sekarang lebih besar dari panjang dua cluster (64KB). (Lihat Gambar 8.22.)



Gambar 8.22 Dokumen "hello.txt" Menampilkan Ukuran File Lebih Besar

Berapa banyak cluster yang dibutuhkan dokumen 72,5 KB ini?
1 kluster = 32 KB—tidak cukup ruang
2 kluster = 64 KB—masih belum cukup ruang
3 kluster = 96 KB—ruang yang cukup untuk menampung dokumen berukuran 72,5 KB

Jadi, karena ukuran file melebihi satu kluster, FAT secara otomatis mengalokasikan kluster tambahan untuk ditempati file, hingga sejumlah kluster yang cukup untuk menampung seluruh file. Sangat jarang ukuran file menjadi ukuran yang tepat dari satu cluster; ini akan mengakibatkan kurang dimanfaatkannya seluruh klaster atau membutuhkan klaster tambahan.

Dalam situasi apa pun, cluster biasanya akan memiliki ruang yang tidak terpakai atau "sisa" yang dialokasikan, tidak ditempati oleh file, yang mengakibatkan kelonggaran sektor dan file, yang masing-masing berguna dan bernilai bagi penyelidik forensik dunia maya.

8.11 ENTRI DIREKTORI DAN FAT

Mari kita lihat bagaimana entri direktori dan FAT bekerja sama dengan memeriksa entri direktori FAT untuk "hello2.txt" (lihat Gambar 8.23).

Byte Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ASCII	H	E	L	L	O	2			T	X	T		·	/	·	T	T	:	T	:	·	·	+	U	T	:	·	·	–	"	·	·
HEX	48	45	4C	4C	4F	32	20	20	54	58	54	20	18	AF	9C	54	54	3A	54	3A	00	00	AB	55	54	3A	02	00	3D	22	01	00

Gambar 8.23 Entri Direktori FAT untuk “hello2.txt”

Langkah 1: Tentukan Dimana File Dimulai

T: Mengapa atau bagaimana OS mengetahui untuk beralih ke byte offset 26–27 untuk menentukan di mana file dimulai? Mengapa tidak byte offset 12–13?

A: Seperti yang telah kami jelaskan sebelumnya, keputusan terletak pada kode spesifik yang terkandung dalam OS.

Nilai HEX pada byte offset 26–27, dalam little endian, ingat, adalah HEX 00 02, yang sama dengan nilai desimal 2. Dengan demikian, file ini dimulai pada cluster 2. Dari informasi yang diberikan dalam entri direktori, sistem dapat menentukan bahwa file dimulai pada cluster 2. (Lihat Gambar 8.24.)

Byte Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ASCII	H	E	L	L	O	2			T	X	T		·	/	·	T	T	:	T	:	·	·	+	U	T	:	·	·	–	"	·	·
HEX	48	45	4C	4C	4F	32	20	20	54	58	54	20	18	AF	9C	54	54	3A	54	3A	00	00	AB	55	54	3A	02	00	3D	22	01	00

Gambar 8.24 Offset Byte 26–27—Mulai dari "hello2.txt" di Cluster Dua

Langkah 2: Tentukan Ukuran File

Byte offset 28–31 berisi ukuran file dalam byte. (Lihat Gambar 8.25.)

Byte Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ASCII	H	E	L	L	O	2			T	X	T		·	/	·	T	T	:	T	:	·	·	+	U	T	:	·	·	=	"	·	·
HEX	48	45	4C	4C	4F	32	20	20	54	58	54	20	18	AF	9C	54	54	3A	54	3A	00	00	AB	55	54	3A	02	00	3D	22	01	00

Gambar 8.25 Byte Offset 28–31 Berisi Ukuran File dalam Byte

Langkah 3: Tentukan Jumlah Cluster yang Dibutuhkan

Dari diskusi kami sebelumnya tentang cara menentukan ukuran cluster yang diperlukan, kami tahu bahwa:

1 kluster = 32 KB—tidak cukup ruang

2 kluster = 64 KB—masih belum cukup ruang

3 kluster = 96 KB—merupakan ruang yang cukup untuk menampung dokumen berukuran 74KB.

Jadi, entri direktori memberi tahu sistem informasi berikut tentang file contoh kami "hello.txt":

1. Di mana file dimulai.
2. Ukuran file dalam byte.
3. Berapa banyak cluster yang diharapkan (tiga).

Langkah 4: Tentukan Dimana File Berakhir

Seperti yang baru saja kita diskusikan, dengan sistem pengarsipan FAT 16, setiap cluster diwakili oleh nilai 16-bit di FAT. Lihat Gambar 8.26 untuk tampilan HEX dari FAT.

FFF8	FFFF	002A	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	000F	0010	0011	0012
0013	0014	0015	0016	0017	0018	0019	001A	FFFF	FFFF	FFFF	0029	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	FFFF	002B	FFFF	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000														

Gambar 8.26 Tampilan HEX dari FAT

Permulaan FAT berisi dua entri khusus, FFF8 dan FFFF, memesan cluster 0 dan 1. (Lihat Gambar 8.27.)

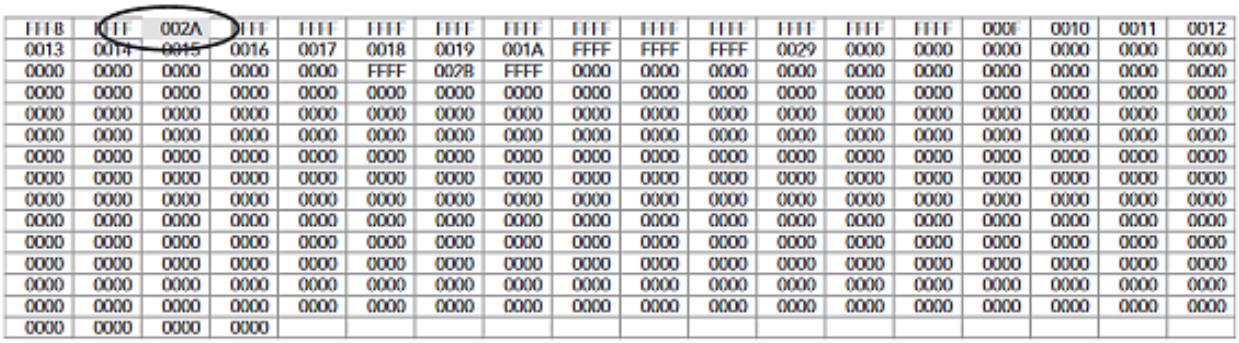
Reserved Clusters
Clusters "0" and "1"

FFF8	FFFF	002A	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	000F	0010	0011	0012
0013	0014	0015	0016	0017	0018	0019	001A	FFFF	FFFF	FFFF	0029	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	FFFF	002B	FFFF	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000														

Gambar 8.27 Awal dari FAT Reserved Cluster 0 dan 1

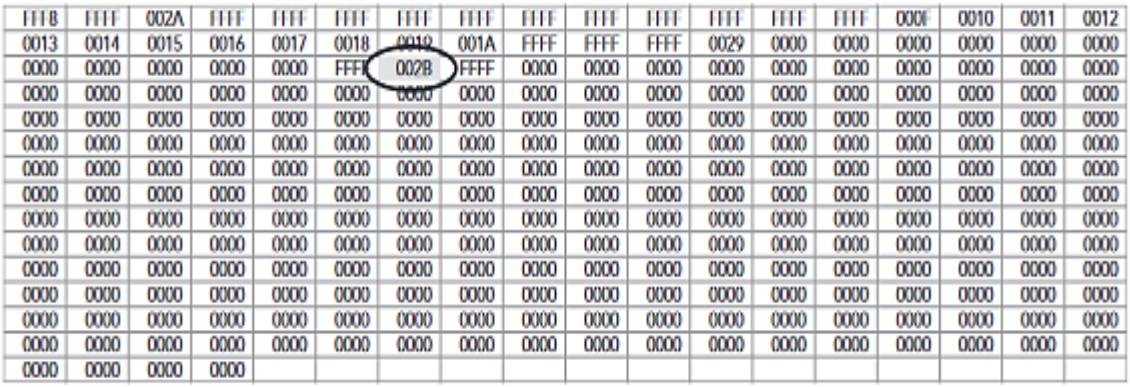
Cluster pertama yang tersedia untuk file adalah cluster 2. Ingat, byte offset 26 dan 27 dari entri direktori mengidentifikasi cluster awal dari sebuah file. Dalam contoh kita, "hello2.txt," cluster awal adalah 2, yang sebenarnya berada pada offset byte 3. (Lihat Gambar 8.28.)

Menggunakan editor HEX dan menggabungkan karakter HEX delapan bit menjadi pasangan, kita dapat lebih mudah memvisualisasikan setiap cluster. Sektor FAT pertama (FAT 1), dan offset karakter HEX 16-bit ketiga dipilih di sini, seperti yang ditunjukkan pada Gambar 8.28.

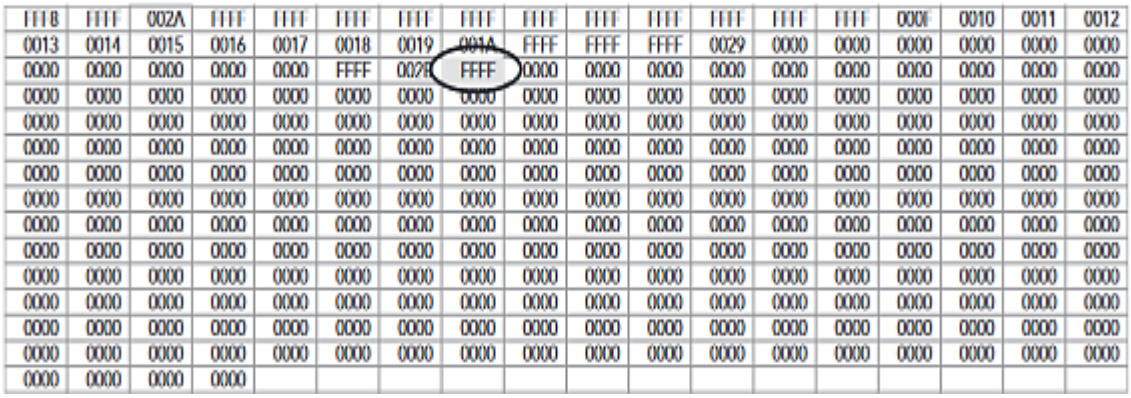


Gambar 8.28 Cluster 2 di Byte Offset 3 — Cluster Pertama Tersedia untuk File

Nilai HEX 002A (atau hanya HEX 2A) dikonversi ke desimal yang setara, yang sama dengan nilai 42. Bagian berikutnya dari file ini ("hello2.txt"), karena telah melebihi ukuran cluster 32 KB, adalah terletak di cluster 42. Pindah ke sektor 42 (offset 43), dan kami menemukan nilai HEX adalah 002B. (Lihat Gambar 8.29.) Pada offset byte 43, kami menemukan nilai HEX 002B, dan mengonversi nilai HEX ini ke hasil ekuivalen desimalnya dalam nilai 43. Ini memberi tahu kami bahwa file berlanjut ke cluster 43 (offset 44), satu dari cluster 42, di offset byte 43. (Lihat Gambar 8.30.)



Gambar 8.29 Kelanjutan Cluster di Cluster 42 (offset 43)



Gambar 8.30 Kelanjutan Cluster di Cluster 43 (offset 44)

Cluster 44 adalah one over dan kami melihatnya memiliki nilai HEX FFFF. Sekali lagi, mengonversi Hex FFFF menjadi padanan desimalnya menghasilkan nilai 65.535. Nilai apa pun yang lebih besar dari FFF8 (desimal 65.528) ditafsirkan sebagai akhir file. Mengingat nilai pada cluster 44 adalah 65.535, yang lebih besar dari 65.528, ini merupakan indikasi bahwa akhir file telah tercapai.

Perhatikan bagaimana dalam contoh kita file "hello2.txt" tidak diadakan (atau disimpan) secara bersamaan pada (di seberang) hard drive, melainkan file dimulai di cluster dua (2) dan melompati 40 cluster ke cluster 42, dan kemudian file berakhir di cluster 43 (lihat Tabel 8.5).

Bayangkan sejenak file yang sangat panjang seperti video. Penyimpanan file yang tidak bersebelahan atau terfragmentasi ini dapat menyebabkan inefisiensi, oleh karena itu diperlukan utilitas defragmentasi. Anda mungkin pernah mendengar beberapa orang mengatakan "defrag hard drive Anda." Pada dasarnya, proses defragmentasi ini berupaya menempatkan semua cluster ke dalam keadaan bersebelahan atau terdefragmentasi.

Tabel 8.5 Ringkasan Nilai Kluster FAT untuk “hello2.txt”

Nomor Klaster FAT	Nilai HEX	Setara Desimal
2	002A	42
43	002B	43
43	FFFF	65,535

Untuk mendekode empat byte ini, 3D 22 01 00, sebagai bilangan bulat endian kecil 32-bit, kita perlu membalikkan byte menjadi 00 01 22 3D, dan mengubah nilai HEX yang dihasilkan ini menjadi ekuivalen desimalnya. Jadi, HEX 00 01 22 3D diubah menjadi desimal sama dengan 74.301. Oleh karena itu, ukuran file untuk “hello2.txt” adalah 74.301 byte.

8.12 BATASAN SISTEM FILING FAT

Sistem file FAT memiliki batasan yang dikenakan padanya oleh berbagai strukturnya; FAT dan entri direktori menimbulkan batasan penyimpanan.

Kekurangan FAT

Sistem file FAT 16 memiliki entri FAT 16-bit. 16 bit ini memungkinkan 65.524 kemungkinan hasil

Berapa banyak byte per sektor?
Ada 512 byte per sektor.

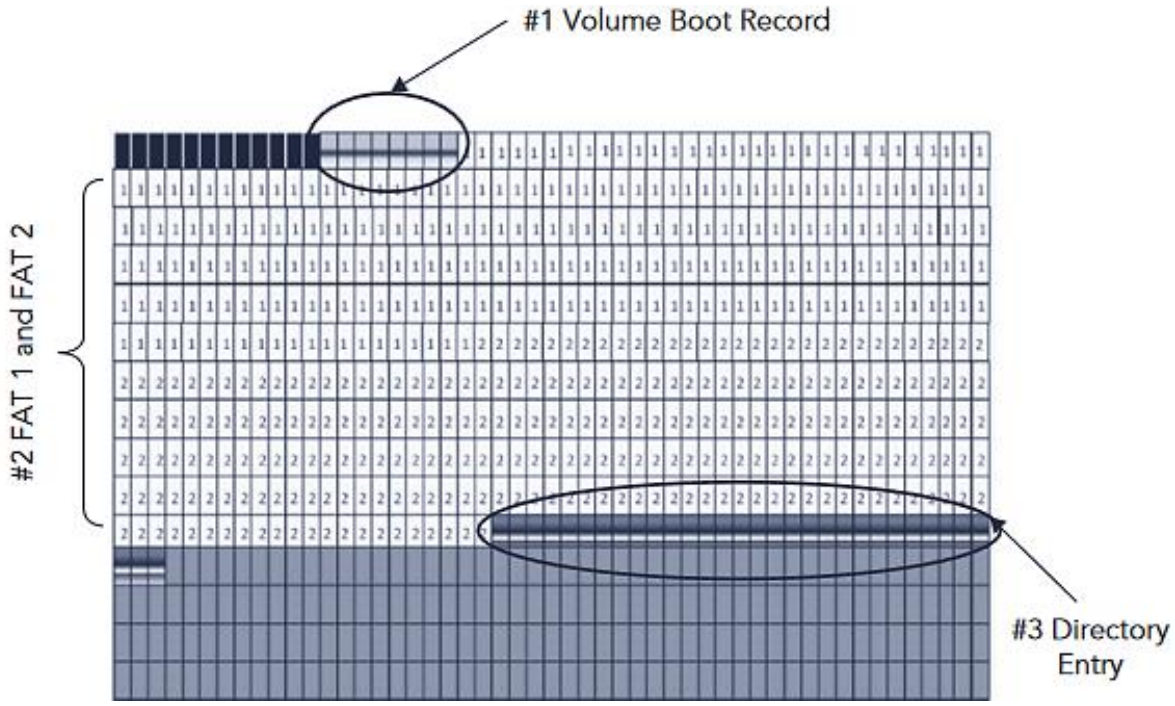
Berapa banyak byte yang diperlukan untuk mewakili satu cluster dari tabel FAT 16?
Dua byte atau 16 bit (FAT 16). Ingat setiap entri FAT mewakili satu cluster.

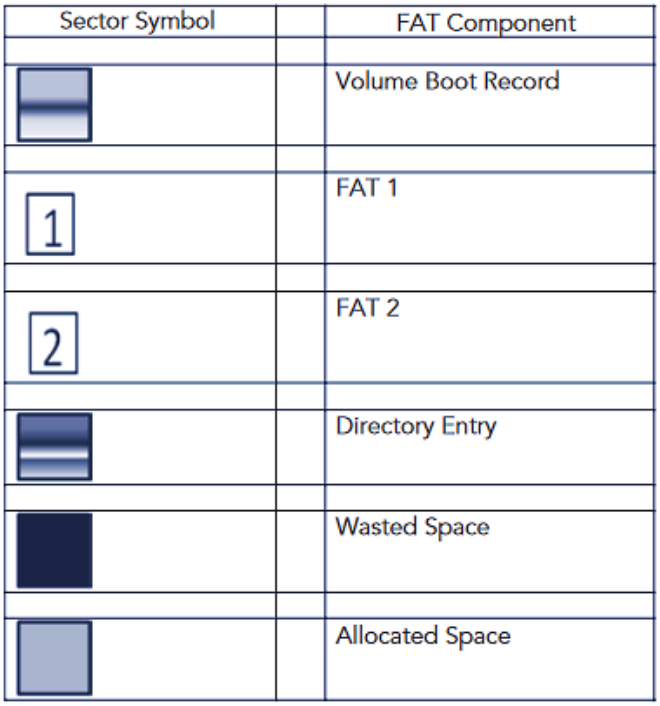
Berapa banyak cluster yang dapat diwakili oleh FAT dalam satu sektor?
Sebanyak 256 cluster, ditentukan dengan membagi dua 512 byte per sektor, karena dibutuhkan dua byte untuk mewakili satu cluster, sehingga 256 cluster dapat diwakili oleh FAT dalam satu sektor.

Berapa banyak sektor yang dibutuhkan FAT untuk melacak potensi ukuran maksimumnya?
Untuk menjawabnya, pertama-tama kita perlu mengetahui berapa ukuran maksimum untuk sistem pengarsipan FAT 16 dan bagaimana hal ini ditentukan? Seperti yang telah kita diskusikan, ukuran potensial maksimum dari sistem pengarsipan FAT 16 adalah 65.536 klaster (benar-benar lebih sedikit karena ruang yang disediakan, tetapi untuk tujuan kita di sini, kita akan tetap menggunakan 65.536 klaster). Ini adalah batasan ukurannya. Perhatikan ini adalah batasan cluster, bukan batasan byte.

Total cluster yang dimungkinkan dengan sistem file FAT 16 sama dengan 65.536. Dengan demikian, total ada 65.536 cluster, masing-masing memerlukan entri dua byte unik di FAT; oleh karena itu, hanya 256 klaster yang dapat diwakili per sektor, diperoleh dengan membagi jumlah total klaster (65.536 dengan 256), sehingga jumlah klaster per sektor adalah 256. Ringkasnya, 256 sektor diperlukan untuk dialokasikan ke FAT untuk memuat total kemungkinan cluster yang tersedia dengan sistem file 16-bit seperti FAT 16.

Gambar 8.31 menggambarkan tampilan grafis dari tiga komponen sistem file FAT, selain mengidentifikasi dua jenis "ruang" yang ditemukan di semua disk, ruang terbuang bersama dengan ruang yang dialokasikan.





Gambar 8.31 Tiga Komponen Sistem File FAT

Legenda yang terkait dengan Gambar 8.31 mengidentifikasi ketiga komponen FAT, serta area lain yang biasanya ditemukan di registri.

Catatan: Ini adalah representasi logis dari konstruksi fisik pada hard drive. Ini hanyalah interpretasi logis berdasarkan karakteristik fisik sebenarnya dari volume FAT 16, yang dibuat oleh perangkat lunak forensik populer, EnCase oleh Guidance Software.

Ringkasan FAT

1. Setiap kotak mewakili satu sektor atau 512 byte.
2. Ada dua FAT—untuk tujuan redundansi; mereka identik.
3. Setiap FAT, seperti yang kami hitung, memiliki panjang 256 sektor.
4. 256 sektor dikalikan 512 (byte per sektor) sama dengan FAT sebesar 131.072 byte.
5. Ingatlah bahwa dibutuhkan dua byte untuk mewakili satu cluster; dengan demikian, 131.072 byte/2 sama dengan 65.536 kemungkinan total cluster.
6. Benar-benar hanya ada 65.524 klaster yang memungkinkan, dengan memperhitungkan klaster yang dicadangkan sistem.

8.13 BATASAN ENTRI DIREKTORI

Seperti yang telah kita diskusikan sebelumnya, FAT memberlakukan batasan pada sistem pengarsipan FAT; dengan FAT 16 misalnya hanya 65.536 cluster yang dimungkinkan. Entri direktori menimbulkan batasannya sendiri. Ingat dari diskusi kita sebelumnya tentang entri direktori (FAT 12/16), setiap entri memiliki panjang 32 byte. Jumlah total entri direktori yang tersedia ditentukan saat volume diformat dan ditetapkan pada 32 sektor dalam FAT 12/16.

Ingat, satu sektor berisi 512 byte, satu entri direktori terdiri dari 32 byte, dan total sektor yang disediakan untuk entri direktori (dalam sistem file FAT 16) sama dengan 32 sektor. Jumlah total entri direktori ditentukan sebagai berikut:

1. Total sektor yang dicadangkan untuk semua entri direktori sama dengan 32, dikalikan dengan 512 (byte per sektor), yang sama dengan 16.384 byte.
2. Oleh karena itu, total entri direktori ditentukan dengan mengambil 16.384 (total byte yang dicadangkan untuk entri direktori) dan membagi nilai tersebut dengan 32 (byte per entri direktori), menghasilkan total 512 entri direktori.

Sebagai alternatif, total entri direktori dapat ditentukan sebagai berikut:

> ***Berapa banyak entri direktori yang ada per sektor?***
> 512 (byte per sektor) dibagi dengan 32 (byte per entri direktori tunggal) sama dengan 16 entri direktori per sektor.
>
> ***Berapa banyak total entri direktori yang mungkin?***
> 16 (entri direktori per sektor) dikalikan dengan 32 (total sektor yang ditetapkan untuk entri direktori) sama dengan 512 kemungkinan entri direktori.

Jadi, batasan FAT 16! Hanya 512 entri direktori atau file potensial yang dapat disimpan terlepas dari ukuran file. FAT 16 akan dengan cepat memenuhi keterbatasannya, jika tidak dengan ruang penyimpanan yang tersedia secara umum maka dalam jumlah file yang dapat disimpan/disimpan. Beberapa konsep kunci dari sistem pengarsipan FAT telah dicocokkan dengan pendamping analog yang sesuai pada Tabel 8.6.

Tabel 8.6 Perbandingan Pustaka dengan FAT

Alegori Perpustakaan	Tabel Alokasi File
Perpustakaan	Partisi
Katalog kartu	Sistem pengisian
Sistem Desimal Dewey	Jenis sistem pengarsipan (yaitu, FAT 12/16)
Data katalog kartu	Data entri direktori
Rak buku	FAT
Buku	File

8.14 RINGKASAN DAN PENGAYAAN

Kami telah membahas secara rinci dan panjang lebar tentang sistem pengarsipan FAT, tidak begitu banyak untuk mendapatkan pemahaman mendalam tentang sistem pengarsipan yang kurang lebih usang, tetapi untuk mendapatkan pemahaman yang lebih baik tentang sistem pengarsipan secara umum. Selain itu, menggunakan sistem pengarsipan FAT sebagai latar membantu menjelaskan banyak konsep penting seperti ruang yang dialokasikan, ruang kendur, endianness, dan penataan ulang bit. Konsep-konsep ini sangat penting dalam memahami bagaimana data ditimpa dalam penyimpanan normal, penghapusan, dan pemrosesan informasi secara umum.

Beberapa dari konsep ini sebelumnya telah disinggung, tetapi pembahasan di sini tentang sistem pengarsipan FAT menyatukan konsep-konsep kritis ini. Apakah memahami seluk-beluk FAT relevan dalam investigasi? Mungkin dan mungkin tidak, tetapi memahami ruang kendur dan ruang yang dialokasikan sangatlah penting.

Memahami bahwa sistem file ada dan memiliki gagasan tentang bagaimana sistem file beroperasi secara umum sangat penting untuk memahami bagaimana data diakses, disimpan, diubah, dihapus, dan/atau ditimpa. Tanpa pemahaman tentang sistem pengarsipan, bagaimana kita bisa memahami bagaimana sebuah file dihapus, atau mungkin lebih ringkasnya, tidak dihapus? Saat file dihapus, kita membahas bagaimana hanya byte pertama dalam entri direktori yang diubah menjadi "~", membuat ruang tersebut tersedia untuk penggunaan di masa mendatang, atau "tidak terisi".

Namun, sisa entri direktori dan keseluruhan file, sebagaimana dibatasi dalam FAT, tetap ada. Ruang tidak terisi dan siap untuk ditimpa, tetapi data yang tidak terisi dan berpotensi memberatkan tetap ada. Ini adalah konsep forensik dasar, tetapi beberapa yang paling kritis dan penting untuk dipahami!

PENGAYAAN

1. **Tabel bidang partisi**

Offset Byte	Panjang Bidang	Nilai Sampel	Keterangan
00	1 byte	80	Indikator Boot. Menunjukkan apakah partisi adalah partisi sistem. Nilai legalnya adalah: 00 = Jangan gunakan untuk booting. 80 = Partisi sistem.
01	1 byte	01	Mulai Header.
02–03	2 byte	01 00	Sektor Mulai. Hanya bit 0–5 yang digunakan. Bit 6–7 adalah dua bit atas untuk bidang Silinder Mulai. Silinder Mulai. Bidang ini berisi delapan bit yang lebih rendah dari nilai silinder. Silinder awal dengan demikian merupakan angka 10-bit, dengan nilai maksimum 1.023
04	1 byte	06	Byte ini mendefinisikan tipe volume.
05	1 byte	0F	Header Akhir.
06-07	2 byte	FF FF	Sektor Akhir. Hanya bit 0–5 yang digunakan. Bit 6–7 adalah dua bit atas untuk bidang silinder akhir. Silinder Akhir. Bidang ini berisi delapan bit yang lebih rendah dari nilai silinder. Silinder akhir dengan demikian adalah angka 10-bit, dengan nilai maksimum 1.023.
08-11	4 byte	3F 00 00 00	Sektor Relatif. Memulai sektor untuk partisi
12-15	4 byte	89 7E 9B 1D	modus LBA.

Tipe Partisi atau Volume (Offset 4)

01 partisi primer FAT 12-bit atau drive logis.
04 Partisi primer FAT 16-bit atau drive logis.
05 Partisi yang diperluas.
06 partisi primer FAT 32-bit atau drive logis.
07 Partisi primer NTFS atau drive logis.

2. Nilai Tabel Alokasi File

Setiap karakter 16-bit dalam tabel sistem file FAT 16 akan mewakili satu cluster.
Setiap karakter 3-bit dalam tabel sistem file FAT 32 akan mewakili satu cluster.

Nilai-nilai yang ada dalam tabel adalah sebagai berikut:
Tidak terisi—diwakili oleh 0.
Dialokasikan—nilai akan diwakili oleh cluster berikutnya yang digunakan oleh file.
Terakhir Dialokasikan—cluster terakhir yang digunakan oleh file.

Biasanya terlihat dengan nilai HEX "End Of File" dari Fs. Meskipun ini tidak selalu terjadi, aturannya adalah:
- FAT 12—nilai HEX lebih besar dari nilai HEX FF8 (12 bit)
- FAT 16—nilai HEX lebih besar dari nilai HEX FF F8 (16 bit)
- FAT 32—nilai HEX lebih besar dari nilai HEX FF FF FF F8 (32 bit)

Bad Sector—terlihat dengan nilai HEX yang diakhiri dengan F7:
- FAT12—FF7
- FAT16—FF F8
- FAT32—FF FF FF F7

3. Deskripsi Offset Byte Entri Direktori

0 Karakter pertama dari nama file atau byte status.
1–7 Nama file dilanjutkan.
8–10 Ekstensi file tiga karakter.
11 Atribut—hanya baca, tersembunyi, dan sebagainya.
12–13 Dicadangkan.
14–17 Buat waktu dan tanggal file yang disimpan sebagai stamp waktu MS-DOS 32-bit.
18–19 Terakhir diakses, tanpa waktu.
20–21 Dua byte tinggi dari klaster awal FAT 32; FAT 12/16 akan memiliki nol.
22–25 Waktu dan tanggal file terakhir ditulis lagi sebagai MS-DOS 32-bit.
26–27 Klaster awal untuk FAT 16/12; dua byte rendah dari cluster awal untuk FAT 32.
28–31 Ukuran dalam byte file. Akan menjadi 0 untuk direktori.

4. Nilai Offset Fat 12/16 Byte*

0–2 Pada offset ini kami memiliki instruksi lompatan perakitan untuk mem-bootstrap kode. Biasanya 0xEB3C90 atau 0xEB5890.

3–10 Offset dari 3–10 berisi id OEM. ID OEM adalah serangkaian karakter yang mengidentifikasi nama dan nomor versi sistem operasi yang memformat volume. Beberapa contoh id OEM untuk versi OS yang berbeda adalah: Win95 = MSWIN4.0 Win98 = MSWIN4.1 Win2K / XP = MSDOS5.0

11–12 Byte per sektor. Jumlah byte per sektor secara default adalah 512, tetapi bisa 1.024, 2.048, atau 4.096. Seperti yang dapat kita lihat pada gambar, nilai HEX dari 11–12 adalah 00 02. Perlu dilihat di little endian.

13 Sektor per klaster. Penting untuk memiliki nilai-nilai ini dalam pangkat 2 dan harus selalu lebih besar dari nol.

14–15 Jumlah sektor di area yang dicadangkan. FAT 12/16 biasanya satu.

16 Jumlah tabel FAT yang ada. Biasanya dua dengan FAT 1 dan FAT 2. FAT 2 adalah duplikat dari FAT 1 yang digunakan untuk redundansi saat FAT 1 rusak.

17–18 Jumlah maksimum entri direktori 32-byte di direktori akar. Biasanya 0 untuk FAT 32 dan 512 untuk FAT 12/FAT 16.

19–20 bilangan bulat 16-bit yang menjelaskan jumlah sektor dalam partisi. Jika 0, jumlah sektor melebihi 65.536. Jika demikian, nilainya kemudian tercermin dalam offset byte 32–35, bilangan bulat 32-bit yang menjelaskan jumlah sektor dalam sebuah partisi.

21 Deskriptor media yang memberi tahu apakah itu media yang dapat dilepas atau media yang tidak dapat dilepas. Jika nilainya 0xF8, maka itu adalah media yang tidak dapat dilepas. Jika 0xF0, maka itu adalah media yang dapat dipindahkan.

22–23 bilangan bulat 16-bit yang menjelaskan jumlah sektor yang digunakan oleh setiap FAT di FAT 12/FAT 16. Nilai nol untuk FAT 32.

24–25 Sektor per nilai trek. Biasanya 63 untuk hard disk. 26–27 Nilai jumlah kepala. Biasanya 255 untuk hard disk.

28–31 Jumlah sektor tersembunyi sebelum dimulainya partisi. Nilainya biasanya 63 untuk volume pertama di hard disk.

32–35 bilangan bulat 32-bit yang menjelaskan jumlah sektor dalam partisi. Jika 0, jumlahnya tidak melebihi 65.536 dan digambarkan sebagai bilangan bulat 16-bit dalam byte 19-20. Hanya satu dari keduanya (19–20 atau 32–35), dan bukan keduanya, harus disetel ke 0.

36 Nomor drive interupsi; HEX 00 untuk disket dan HEX 80 untuk hard drive.

37 Hanya digunakan oleh Windows NT; biasanya diatur ke 0.

38 Tanda tangan boot yang diperluas untuk menentukan validitas dari tiga bidang berikutnya. Jika HEX 29, tiga bidang berikutnya ada dan valid, jika tidak harap HEX 00.

39–42 Nomor seri volume.

43–53 Label volume dalam ASCII. Ini adalah label yang diberikan kepada pengguna secara opsional saat volume dibuat.

54–61 Jenis sistem file pada saat pemformatan. Yang ditampilkan adalah ASCII sebagai FAT 12, FAT 16.

62–509 Kode program bootstrap dan pesan kesalahan.

510–511 Nilai tanda tangan adalah dua byte dan harus berupa HEX 55AA.

5. Nilai Offset Fat 32 Byte*

0–2 Pada offset ini kami memiliki instruksi lompatan perakitan untuk mem-bootstrap kode. Biasanya 0xEB3C90 atau 0xEB5890.

3–10 Offset dari 3–10 berisi id OEM. ID OEM adalah serangkaian karakter yang mengidentifikasi nama dan nomor versi sistem operasi yang memformat volume. Beberapa contoh id OEM untuk versi OS yang berbeda adalah: Win95 = MSWIN4.0 Win98 = MSWIN4.1 Win2K / XP = MSDOS5.0

11–12 Byte per sektor. Jumlah byte per sektor secara default adalah 512, tetapi bisa 1.024, 2.048, atau 4.096. Seperti yang dapat kita lihat pada gambar, nilai HEX dari 11–12 adalah 00 02. Perlu dilihat di little endian.

13 Sektor per klaster. Penting untuk memiliki nilai-nilai ini dalam pangkat dua dan harus selalu lebih besar dari nol.

14–15 Jumlah sektor di area yang dicadangkan. FAT 12/16 biasanya satu.

16 Jumlah tabel FAT yang ada. Biasanya dua dengan FAT 1 dan FAT 2. FAT 2 adalah duplikat dari FAT 1 yang digunakan untuk redundansi saat FAT 1 rusak.

17–18 Jumlah maksimum entri direktori 32 byte di direktori akar. Biasanya 0 untuk FAT 32 dan 512 untuk FAT 12/FAT 16.

19–20 bilangan bulat 16 bit yang menjelaskan jumlah sektor dalam partisi. Jika 0, jumlah sektor melebihi 65.536. Jika demikian, nilainya kemudian tercermin dalam offset byte 32–35, bilangan bulat 32-bit yang menjelaskan jumlah sektor dalam sebuah partisi.

21 Deskriptor media yang memberi tahu apakah itu media yang dapat dilepas atau media yang tidak dapat dilepas. Jika nilainya 0xF8, maka itu adalah media yang tidak dapat dilepas. Jika 0xF0 maka itu adalah media yang dapat dipindahkan.

22–23 bilangan bulat 16-bit yang menjelaskan jumlah sektor yang digunakan oleh setiap FAT di FAT 12/FAT 16. Nilai nol untuk FAT 32.

24–25 Sektor per nilai trek. Biasanya 63 untuk hard disk.

26–27 Nilai jumlah kepala. Biasanya 255 untuk hard disk.

28–31 Jumlah sektor tersembunyi sebelum dimulainya partisi. Nilainya biasanya 63 untuk volume pertama di hard disk.

32–35 bilangan bulat 32-bit yang menjelaskan jumlah sektor dalam partisi. Jika 0, jumlahnya tidak melebihi 65.536 dan digambarkan sebagai Bilangan bulat 16-bit dalam byte 19–20. Hanya satu dari keduanya (19–20 atau 32–35), dan bukan keduanya, harus disetel ke 0.

36–39 bilangan bulat 32-bit yang menjelaskan jumlah sektor yang digunakan oleh satu FAT (FAT 1/FAT 2) pada partisi FAT 32. Byte 22–23 harus disetel ke 0 untuk FAT 32.

40–41 Serangkaian nilai yang digunakan untuk menjelaskan apakah entri FAT digandakan. Jika nilainya 0 maka FAT digandakan. Jika nilainya tidak nol maka duplikasi tidak terjadi.

42–43 Nomor versi mayor dan minor dari volume FAT 32. Biasanya nilainya 0×00 dan 0×00 artinya bilangan mayor dan minor FAT adalah 0.

44–47 Nomor cluster di mana direktori root dimulai, yang biasanya 2. Cluster 2 dimulai segera setelah salinan cermin dari FAT 1, FAT 2 berakhir.

48–49 Nomor sektor tempat FSINFO (Informasi Sistem File) ditemukan. Biasanya terletak di sektor 1 dan cadangan FSINFO ditemukan di sektor 7. Tujuan FSINFO adalah untuk memberikan informasi kepada Sistem Operasi tentang jumlah cluster gratis yang tersedia untuk sistem dan lokasi cluster gratis berikutnya.

50–51 Nomor sektor untuk lokasi sektor boot cadangan, biasanya sektor 6.

52–63 Dicadangkan, saat ini tidak digunakan.

64 Nomor driver interupsi. Ini adalah HEX 00 untuk disket dan HEX 80 untuk hard drive.

65 Tidak digunakan oleh semua versi windows kecuali WindowsNT. Biasanya diatur ke 0.

66 Tanda tangan boot yang diperluas untuk menentukan validitas dari tiga bidang berikutnya. Jika HEX 0×29, tiga bidang berikutnya (nomor seri, label volume, tipe sistem file) ada dan valid. Jika 0×00, maka tidak ada yang tersedia.

67–70 Nomor seri volume. Setiap volume memiliki serial number yang dapat dilihat dengan menggunakan perintah “vol” di DOS.

71–81 Label volume dalam ASCII. Jika tidak ada yang diberikan oleh pengguna maka itu akan menampilkan "NO NAME."

82–89 Jenis sistem file pada saat pemformatan. Ini tidak berguna setelah pemformatan dan dapat diubah ke nilai arbitrer apa pun.

90–509 Kode program boot strap dan pesan kesalahan.

510–511 Nilai tanda tangan; itu adalah dua byte dan harus 0×55AA.

6. Kelebihan

2^0	=	1	2^44	=	17,592,186,044,416
2^1	=	2	2^45	=	35,184,372,088,832
2^2	=	4	2^46	=	70,368,744,177,664
2^3	=	8	2^47	=	140,737,488,355,328
2^4	=	16	2^48	=	281,474,976,710,656
2^5	=	32	2^49	=	562,949,953,421,312
2^6	=	64	2^50	=	1,125,899,906,842,620
2^7	=	128	2^51	=	2,251,799,813,685,250
2^8	=	256	2^52	=	4,503,599,627,370,500
2^9	=	512	2^53	=	9,007,199,254,740,990
2^10	=	1,024	2^54	=	18,014,398,509,482,000
2^11	=	2,048	2^55	=	36,028,797,018,964,000
2^12	=	4,096	2^56	=	72,057,594,037,927,900
2^13	=	8,192	2^57	=	144,115,188,075,856,000
2^14	=	16,384	2^58	=	288,230,376,151,712,000
2^15	=	32,768	2^59	=	576,460,752,303,423,000
2^16	=	65,536	2^60	=	1,152,921,504,606,850,000
2^17	=	131,072	2^61	=	2,305,843,009,213,690,000

2^18	= 262,144	2^62	= 4,611,686,018,427,390,000
2^19	= 524,288	2^63	= 9,223,372,036,854,780,000
2^20	= 1,048,576	2^64	= 18,446,744,073,709,600,000
2^21	= 2,097,152	2^65	= 36,893,488,147,419,100,000
2^22	= 4,194,304	2^66	= 73,786,976,294,838,200,000
2^23	= 8,388,608	2^67	= 147,573,952,589,676,000,000
2^24	= 16,777,216	2^68	= 295,147,905,179,353,000,000
2^25	= 33,554,432	2^69	= 590,295,810,358,706,000,000
2^26	= 67,108,864	2^70	= 1,180,591,620,717,410,000,000
2^27	= 134,217,728	2^71	= 2,361,183,241,434,820,000,000
2^28	= 268,435,456	2^72	= 4,722,366,482,869,650,000,000
2^29	= 536,870,912	2^73	= 9,444,732,965,739,290,000,000
2^30	= 1,073,741,824	2^74	= 18,889,465,931,478,600,000,000
2^31	= 2,147,483,648	2^75	= 37,778,931,862,957,200,000,000
2^32	= 4,294,967,296	2^76	= 75,557,863,725,914,300,000,000
2^33	= 8,589,934,592	2^77	= 151,115,727,451,829,000,000,000
2^34	= 17,179,869,184	2^78	= 302,231,454,903,657,000,000,000
2^35	= 34,359,738,368	2^79	= 604,462,909,807,315,000,000,000
2^36	= 68,719,476,736	2^80	= 1,208,925,819,614,630,000,000,000
2^37	= 137,438,953,472	2^81	= 2,417,851,639,229,260,000,000,000
2^38	= 274,877,906,944	2^82	= 4,835,703,278,458,520,000,000,000
2^39	= 549,755,813,888	2^83	= 9,671,406,556,917,030,000,000,000
2^40	= 1,099,511,627,776	2^84	= 1.93E+25
2^41	= 2,199,023,255,552	2^85	= 3.87E+25
2^42	= 4,398,046,511,104	2^86	= 7.74E+25
2^43	= 8,796,093,022,208	2^87	= 1.55E+26

2^88	= 3.09E+26	2^109	= 6.49E+32
2^89	= 6.19E+26	2^110	= 1.30E+33
2^90	= 1.24E+27	2^111	= 2.60E+33
2^91	= 2.48E+27	2^112	= 5.19E+33
2^92	= 4.95E+27	2^113	= 1.04E+34
2^93	= 9.90E+27	2^114	= 2.08E+34
2^94	= 1.98E+28	2^115	= 4.15E+34
2^95	= 3.96E+28	2^116	= 8.31E+34
2^96	= 7.92E+28	2^117	= 1.66E+35
2^97	= 1.58E+29	2^118	= 3.32E+35
2^98	= 3.17E+29	2^119	= 6.65E+35
2^99	= 6.34E+29	2^120	= 1.33E+36
2^100	= 1.27E+30	2^121	= 2.66E+36
2^101	= 2.54E+30	2^122	= 5.32E+36
2^102	= 5.07E+30	2^123	= 1.06E+37
2^103	= 1.01E+31	2^124	= 2.13E+37
2^104	= 2.03E+31	2^125	= 4.25E+37
2^105	= 4.06E+31	2^126	= 8.51E+37
2^106	= 8.11E+31	2^127	= 1.70E+38
2^107	= 1.62E+32	2^128	= 3.40E+38
2^108	= 3.25E+32		

2^128 = 3.40E+38
340,282,366,920,938,000,000,000,000,000,000,000

BAB 9
SISTEM FILE—NTFS DAN SELANJUTNYA

Karena teknologi benar-benar mengikuti iramanya sendiri dan terus-menerus dalam keadaan berubah dan berubah, profesional forensik dunia maya harus terbiasa dengan perubahan yang diumumkan oleh vendor mengenai sistem operasi mereka dan variasi sistem file yang mungkin muncul dari kemajuan apa pun dalam operasi. desain sistem dan pembaruan rilis mendatang untuk sistem operasi ini.

9.1 SISTEM FILE TEKNOLOGI BARU

Selanjutnya, tinjauan sistem file Windows lainnya, Sistem File Teknologi Baru (NTFS), yang penggunaannya dimulai dengan Windows NT pada tahun 1993. Windows XP, 2000, Server 2003, 2008, dan Windows 7 juga semuanya menggunakan versi NTFS yang lebih baru. Sistem pengarsipan sangat kompleks dan yang lebih buruk lagi, sangat sedikit spesifikasi yang dipublikasikan dari Microsoft yang menjelaskan "tata letak di disk". Artinya, representasi logis dari struktur fisik sistem file ini bersifat spekulatif dan sangat sulit untuk divisualisasikan. Untung tujuan kami di sini bukan analisis mendalam sedikit demi sedikit dari setiap sistem file, melainkan pemahaman yang lebih konseptual tentang sistem file secara umum.

Konsep penting dalam memahami desain NTFS adalah bahwa semua data dialokasikan ke file, termasuk sistem file itu sendiri; file sistem file dapat ditempatkan di mana saja dalam volume, seperti halnya file biasa. Oleh karena itu, NTFS tidak memiliki Tata Letak Sistem File normal seperti FAT, seperti yang dibahas di Bab 8, di mana terdapat area di awal volume yang dicadangkan untuk data ini. Seluruh sistem file dianggap sebagai area data, dan sektor apa pun dapat dialokasikan ke file. Satu-satunya yang konstan dalam struktur file NTFS adalah bahwa sektor pertama berisi sektor boot, mirip dengan volume boot di FAT.

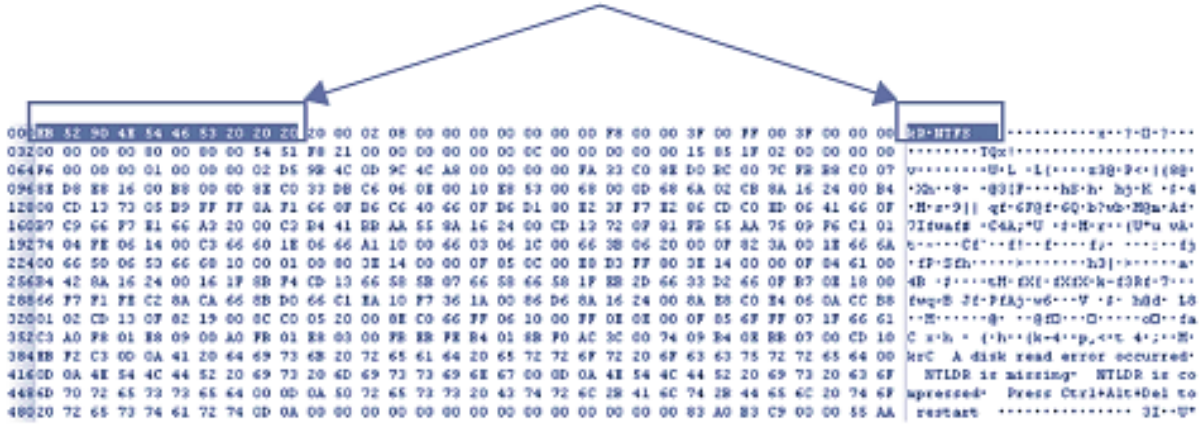
Sistem File Teknologi Baru (NTFS) berisi “komponen” berikut:
1. Partition Boot Sector (PBR)—mirip dengan VBR di FAT
2. Master File Table (MFT)—mirip dengan entri direktori di FAT
3. $bitmap—mirip dengan FAT

9.2 CATATAN BOOT PARTISI

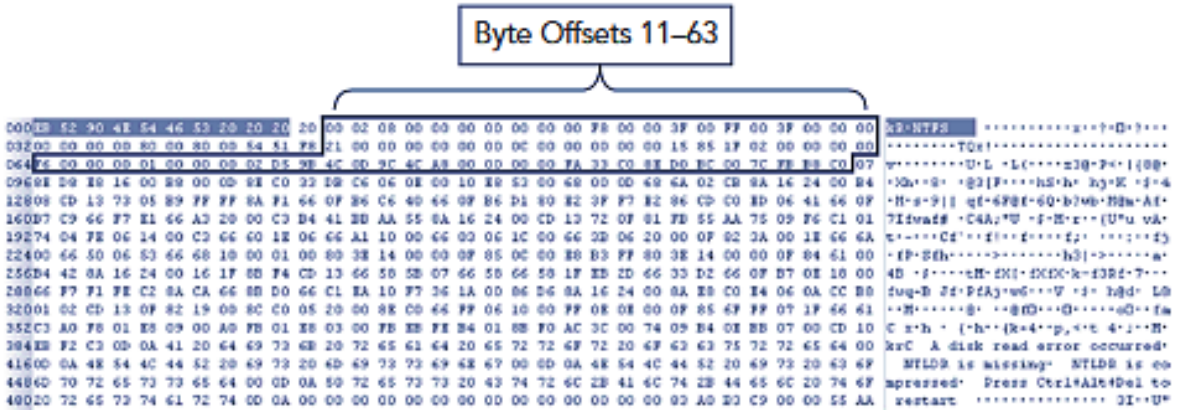
Partition Boot Record (PBR) terdiri dari 16 sektor, berbeda dengan satu sektor dengan FAT. Namun biasanya, hanya delapan sektor dari 16 sektor yang tersedia yang digunakan. Byte offset 0–10 berisi instruksi lompatan dan OEM ID (NTFS). OEM adalah singkatan dari Original Equipment Manufacturer, dan di NTFS, ini diwakili oleh serangkaian karakter yang mengidentifikasi nama dan nomor versi sistem operasi yang memformat volume. Gambar 9.1 adalah sektor pertama dari catatan boot seperti yang terlihat di editor HEX.

Ingat, NTFS mengalokasikan 16 sektor pertama untuk "sektor" boot. Offset byte 0–9 disorot pada Gambar 9.1, menunjukkan pengidentifikasi OEM “NTFS.” Offset byte 3–6 di

sektor pertama sektor boot akan berisi representasi ASCII “NTFS” dari HEX 4E 54 46 53 yang mendefinisikannya seperti itu. Gambar 9.2 menampilkan offset byte 11–63 yang berisi informasi parameter partisi.



Gambar 9.1 Sektor Pertama dari Boot Record (NTFS)



Gambar 9.2 Offset Byte 11 sampai 63 Berisi Informasi Parameter Partisi

Bidang Tabel Partisi, untuk melihat lebih mendalam tentang bidang data dan representasinya di PBR. Tabel 9.1 mencantumkan dan menjelaskan nama field dan nilai desimal untuk isi offset byte 11 sampai 63 dari PBR. Pandangan lebih lanjut pada PBR akan mengungkapkan bahwa byte offset 64–509 berisi kode boot strap, dan byte offset 510–511, secara default, dan sebagai kontrol, berisi akhir penanda file, dengan nilai HEX 55AA.

Tabel 9.1 Offset Byte 11 sampai 63 dari PBR

Offset Byte	Panjang Bidang	Contoh (HEX)	Nama Bidang	Nilai Desimal (little endian)
11 & 12	2 byte	00 02	Byte per sektor	512
13	1 byte	8	Sektor per cluster	8
28–31	4 byte	3F 00 00 00	Sektor tersembunyi	63
40–47	8 byte	54 51 F8 21 00 00 00 00	Jumlah sektor	569,921,876
48–55	8 byte	00 00 0C 00	Cluster logis # untuk	786,432

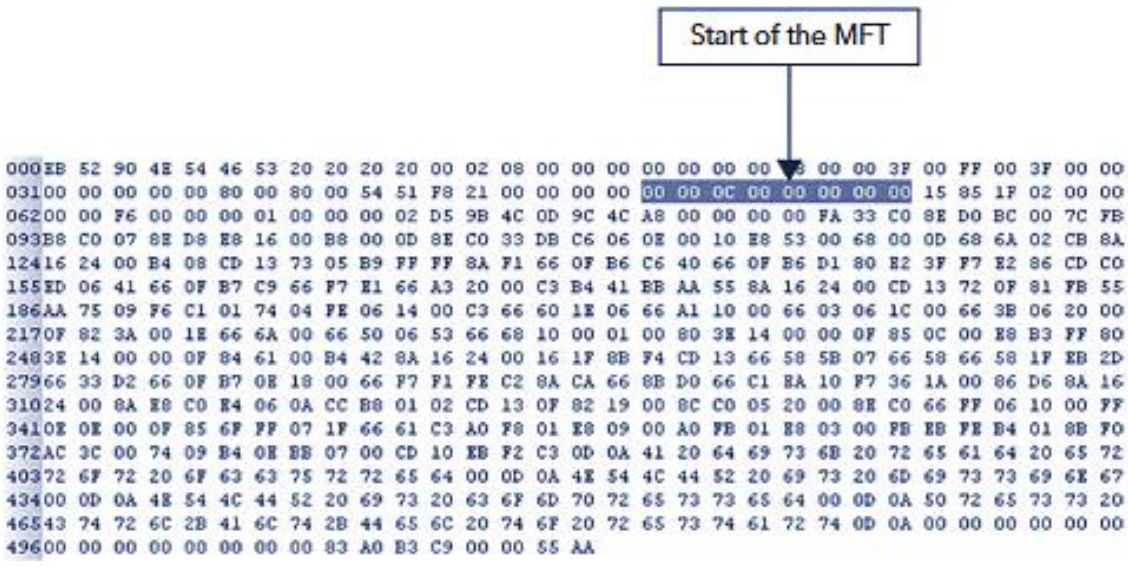
		00 00 00 00	file $MFT	
56–63	8 byte	15 85 1F 02 00 00 00 00	Cluster logis # untuk file $MFT Mirr	35,620,117

9.3 TABEL FILE MASTER

Master File Table (MFT) adalah jantung dari sistem file NTFS. Ini analog dengan entri direktori dalam sistem pengarsipan FAT; itu akan berisi banyak metadata, mirip dengan katalog kartu dalam analogi perpustakaan kami. MFT sangat mirip dengan database karena berisi entri untuk melacak semua data yang terkandung dalam sistem file, dan dalam hal ini bertindak sedikit seperti FAT. Dalam perbandingan kami dengan FAT, MFT sebenarnya memiliki fungsi yang mirip dengan entri direktori dan FAT. MFT berisi entri untuk setiap file dan direktori di partisi, termasuk dirinya sendiri, yang diberi nama $MFT. MFT tersebar di seluruh struktur disk; itu tidak terkandung atau dibatasi untuk sektor tertentu yang ditentukan seperti entri direktori FAT. Karena MFT tidak terbatas pada sektor yang telah ditentukan sebelumnya, ini memungkinkan sistem file menjadi dinamis dan berkembang sesuai kebutuhan. Itu tidak dibatasi atau terbatas pada sejumlah file tertentu seperti pada sistem pengarsipan FAT. Namun setiap entri (atau catatan) memiliki panjang tetap 1.024 byte. Karena ada 512 byte per sektor, ada dua sektor per entri MFT.

Menentukan Lokasi MFT

Untuk menentukan lokasi awal MFT, kami melihat byte offset 48–55 (delapan byte) di Boot Record. Nilai desimal dari nilai biner ini memberi kita nomor Cluster Logis untuk $MFT. Gambar 9.3 menunjukkan penempatan byte offset 48–55, di dalam PBR, di mana kita akan mencari awal dari MFT.



Gambar 9.3 Byte Offset 48 sampai 55 dan Awal dari MFT

Menentukan lokasi MFT ($MFT) memerlukan sedikit komputer matematika dan mengingat diskusi kita sebelumnya tentang endianness. Untuk memulai, kami mengidentifikasi nilai HEX untuk offset byte 48–55. Dengan melihat Gambar 9.3, kami

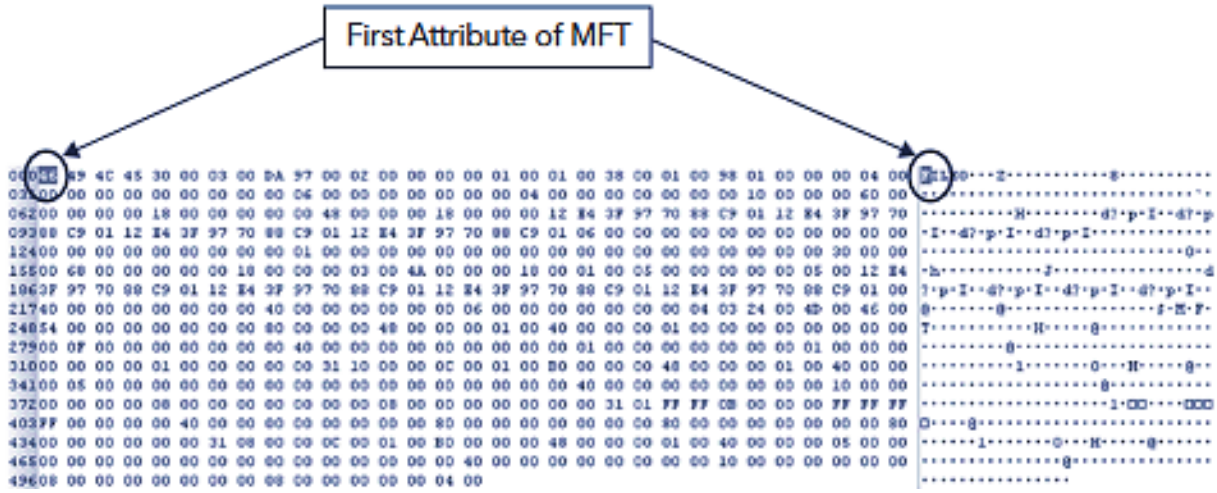
menentukan bahwa nilai HEX untuk byte offset 48–55 sama dengan HEX 00 00 0C 00 00 00 00 00, yang dalam format big endian.

Selanjutnya, atur ulang nilai HEX ini menjadi little endian, sehingga menghasilkan HEX 00 00 00 00 00 C0 00 00.

Asli dalam big endian = 00 00 0C 00 00 00 00 00
Ubah ke little endian = 00 00 00 00 00 0C 00 00
Jatuhkan nol di depan = 00 00 00 00 00 0C 00 00

Mengapa angka nol di depan dihilangkan? HEX 00 dalam desimal sama dengan nol, dan seperti angka desimal apa pun, nol yang mendahului angka selain nol biasanya diabaikan (mis., 098 biasanya ditulis hanya sebagai 98).

Terakhir, ubah nilai HEX yang dihasilkan 0C 00 00 menjadi padanan desimalnya, sehingga memperoleh nilai 786.432. Dengan demikian, konfirmasi MFT akan dimulai pada cluster offset 786.432. Gambar 9.4 menunjukkan entri atribut pertama di sektor pertama MFT seperti yang dilihat oleh editor HEX.



Gambar 9.4 Entri Atribut Pertama di Sektor Pertama MFT

MFT memandang segala sesuatu tentang file sebagai atribut, metadata, dan data. Byte pertama dari entri MFT adalah header record file standar (lihat Gambar 9.4). Empat byte pertama dari MFT digabungkan untuk membentuk Pengidentifikasi file, "FILE." Atribut inilah yang mendefinisikan sektor ini sebagai rekor. Faktanya, beberapa data aktual yang terkandung dalam file akan ada dalam entri MFT, karena dianggap hanya atribut lain. Jika sebuah file berukuran kecil, terkadang keseluruhan file tersebut disimpan dalam entri MFT; ini disebut data penduduk.

File dan Atributnya

Jika file terlalu besar untuk menampung semua datanya di dalam MFT, maka file tersebut dialokasikan ke cluster. Cluster berjalan kemudian disimpan di tempat data penduduk. Biasanya, 480 byte adalah panjang maksimum untuk file residen.

Atribut memiliki dua bagian:

1. Header—Mengidentifikasi atribut: jenis file, ukuran file, dan nama. Juga, ia memiliki tanda untuk mengidentifikasi apakah atribut dikompresi atau dienkripsi. Header bersifat umum dan standar untuk semua atribut.
2. Konten—Konten sebenarnya dari file untuk file residen. Lokasi cluster file untuk file bukan penduduk. Konten bersifat spesifik dan unik serta dapat berukuran berapa pun.

Semua atribut file disimpan dalam salah satu dari dua cara berbeda, tergantung pada karakteristik atribut, khususnya ukuran:

1. Atribut Residen—Atribut yang disimpan langsung di dalam rekaman MFT utama file itu sendiri. Banyak atribut file yang paling sederhana dan paling umum disimpan di dalam file MFT. Bahkan, beberapa diminta oleh NTFS untuk menjadi penduduk dalam catatan MFT untuk pengoperasian yang benar. Misalnya, nama file, dan pembuatannya, Modifikasi, dan stempel tanggal/waktu akses ada untuk setiap file.
2. Atribut Non-Residen—Jika atribut membutuhkan lebih banyak ruang daripada yang tersedia dalam catatan MFT, atribut tersebut tidak dapat disimpan dalam catatan tersebut. Sebagai gantinya, atribut ditempatkan di lokasi terpisah di disk. Pointer ditempatkan di dalam MFT yang mengarah ke lokasi atribut. Ini disebut penyimpanan atribut non-penduduk.

Sebuah surat atau paket yang diterima di kantor pos bisa dianalogikan. Atributnya adalah amplop, kotak, atau wadah penyimpanan. Bagian luar wadah mungkin memiliki nama, alamat, dan mungkin isinya tertulis di atasnya. Bagian dalam kotak sebenarnya berisi isinya.

Jadi, semua barang disimpan dalam wadah (kotak atau amplop). Wadah dapat memiliki berbagai ukuran dan bentuk, semuanya dengan isi yang berbeda; namun, data yang tertulis di bagian luar penampung semuanya berisi info yang sama—nama, alamat, dan konten. Dalam analogi ini header sama dengan bagian luar kotak dan isi sama dengan isi kotak. Pembaca yang menginginkan informasi lebih lanjut mengenai atribut didorong untuk meninjau Lampiran 9A, Atribut Ditetapkan Sistem NTFS Umum.

$Bitmap

Komponen lain dari NTFS, yang fungsinya agak mirip dengan FAT, adalah $Bitmap. Dalam membandingkan sistem pengarsipan NTFS dan FAT, jelas bahwa MFT di NTFS sebenarnya bertindak sebagai entri direktori dan melakukan beberapa fungsi FAT. Seperti yang telah dibahas sebelumnya, MFT berisi atribut non-penduduk yang menunjuk ke lokasi data. Ini pada dasarnya mirip dengan peran FAT.

Jadi, peran apa yang dilakukan $Bitmap dan bagaimana kemiripannya dengan FAT? $Bitmap adalah file yang merepresentasikan alokasi cluster di dalam sebuah partisi. Ini mengidentifikasi apakah sebuah cluster dialokasikan atau tidak dialokasikan. Setiap bit dalam $Bitmap mewakili sebuah cluster. Jika sebuah bit yang mewakili sebuah cluster memiliki nilai nol (dalam biner), maka cluster tersebut tersedia untuk digunakan atau tidak dialokasikan; jika bit memiliki nilai satu (1), maka cluster tersebut tidak tersedia atau dialokasikan.

$Bitmap tidak mengidentifikasi lokasi awal file, panjang file, ukuran cluster file, atau data apa pun terkait lokasi atau penyimpanan file. Ini hanya memberi tahu sistem jika cluster dialokasikan atau tidak dialokasikan.

9.4 RINGKASAN NTFS

Menjelaskan kompleksitas NTFS agak disederhanakan saat membangun pemahaman yang ada tentang FAT. Keduanya cukup mirip untuk dianalogikan. Tujuan menjelaskan salah satu dari sistem ini adalah untuk memahami bagaimana sistem file dapat bekerja secara umum. Dengan memahami konsep sistem file seperti itu, kami mulai memahami bagaimana satu sistem file (seperti NTFS) dapat jauh lebih dinamis dan adaptif terhadap pertumbuhan daripada yang lain. Kami juga, dan mungkin yang lebih penting bagi penyelidik forensik dunia maya, mulai memahami bagaimana data disimpan.

exFAT

Tabel Alokasi File Diperpanjang Microsoft (exFAT) dirilis dengan Windows Vista SP1 (Paket Layanan Satu). Sebuah sistem file yang dirancang untuk penyimpanan memori Flash dan perangkat eksternal lainnya, ExFAT memperluas ukuran file, ukuran drive, dan batasan direktori dari versi FAT yang lebih lama namun mempertahankan overhead FAT yang rendah. Sistem file yang kuat dan kompleks seperti Windows NTFS memungkinkan penyimpanan yang relatif efisien di drive yang sangat besar. Namun, overhead penyimpanan yang efisien adalah konsumsi sumber daya sistem, seperti memori dan daya pemrosesan. Dalam sistem di mana sumber daya terbatas, NTFS tidak efisien. Sistem file NTFS menghabiskan banyak sumber daya untuk mempertahankan dirinya sendiri. ExFAT dirancang untuk digunakan di area di mana NTFS terlalu banyak dan tidak efisien.

Ada sistem pengarsipan lain selain Microsoft NTFS, exFAT, dan FAT. Seperti yang dapat diharapkan, Sistem Operasi yang berbeda menggunakan sistem pengarsipan yang berbeda, dan seiring dengan kemajuan teknologi, begitu pula fungsionalitas yang terkandung dalam Sistem Operasi dan oleh karena itu dengan sistem File. Banyak dari konsep sistem pengarsipan ini tidak selaras dengan paradigma sistem pengarsipan Microsoft. Memang ada sedikit kesamaan, karena tujuan akhirnya sama, yaitu mengakses file, tetapi umumnya sistem pengarsipan Microsoft bukanlah model untuk semua sistem file.

9.5 KONSEP SISTEM FILING ALTERNATIF

Sistem file alternatif dan konsep yang terkait dengan sistem file alternatif ini akan dieksplorasi di sisa bab ini.

Sistem Pengarsipan Pohon Pencarian Biner

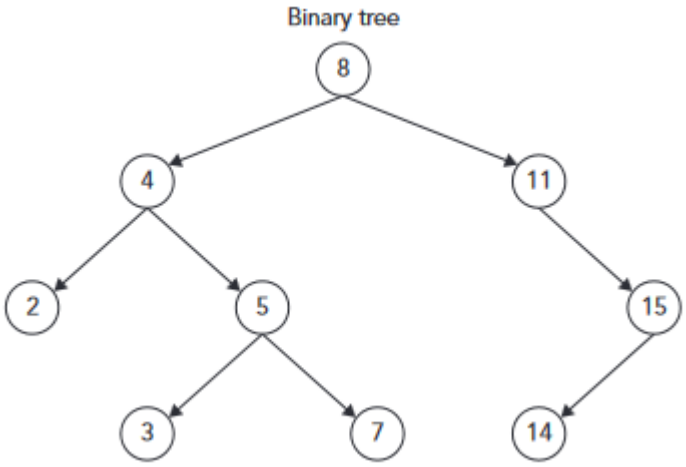
Pohon biner adalah konsep struktur data hierarkis yang digunakan untuk menempatkan dan menemukan file. Secara konseptual, dalam struktur data pohon hierarkis, hal-hal diurutkan di atas atau di bawah hal-hal lain. Pohon itu terdiri dari simpul-simpul yang dihubungkan bersama sebagai orang tua atau anak.

Node paling atas dari pohon disebut root. Node yang berada langsung di bawah node lain disebut sebagai anak. Simpul yang berada tepat di atas simpul lain disebut orang tua. Node tanpa anak terkadang disebut daun. Urutan pohon biner sama dengan jumlah maksimum anak

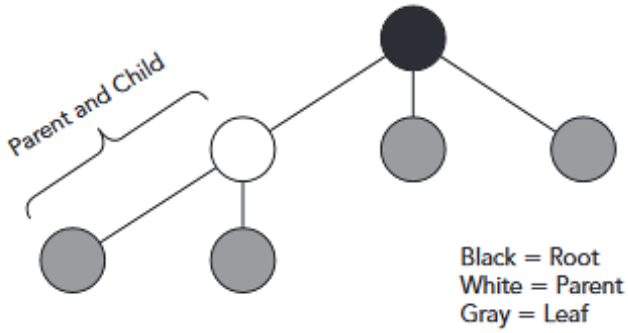
per node (dua dengan pohon biner). Kedalaman pohon sama dengan panjang lintasan (dalam jumlah simpul) dari simpul akar paling atas hingga daun paling bawah. Ukuran pohon pencarian sama dengan jumlah total node.

Dalam pohon biner dasar, setiap simpul hanya dapat memiliki maksimal dua anak. Dalam struktur pohon biner, setiap simpul (item dalam pohon) memiliki nilai yang berbeda. Subpohon kiri dan kanan juga harus mengikuti struktur pohon pencarian biner. Subpohon kiri dari sebuah simpul hanya berisi nilai yang lebih kecil dari nilai simpul induknya dan subpohon kanan dari sebuah simpul hanya berisi nilai yang lebih besar dari nilai simpul induknya. Sekali lagi, setiap simpul hanya memiliki maksimal dua anak. Gambar 9.5 menunjukkan tampilan grafis pada struktur pohon biner.

Sebagai contoh, pada gambar pada Gambar 9.5, simpul akar 8 adalah induk dari simpul 4 dan simpul 11. Demikian juga, simpul 4 dan simpul 11 adalah anak dari simpul akar 8. Ukuran pohon pencarian adalah 9 simpul; kedalamannya 3. Daunnya adalah simpul 2, 3, 7, dan 14. Simpul kiri adalah 4, 2, dan 3, dan simpul kanan adalah 5, 7, 11, dan 15. Semua simpul kecuali simpul 8 adalah anak. Gambar 9.6 menunjukkan hubungan antara akar, induk, dan daun.



Gambar 9.5 Struktur Pohon Pencarian Biner



Gambar 9.6 Hubungan Pohon Biner

Pohon biner (b-tree) adalah struktur data yang berguna untuk menyimpan data yang diurutkan dengan cepat dan mengambil data yang tersimpan dengan cepat. Ini adalah hubungan antara anak-anak dan simpul induk yang membuat pohon biner menjadi struktur data yang efisien. Daun di sebelah kiri memiliki nilai kunci yang lebih kecil, dan daun di sebelah

kanan memiliki nilai kunci yang sama atau lebih besar. Akibatnya, daun di bagian paling kiri pohon memiliki nilai terendah, sedangkan daun di sebelah kanan pohon memiliki nilai terbesar. Lebih penting lagi, karena setiap simpul induk bercabang menjadi dua daun lainnya (misalnya, anak), subset baru yang lebih kecil di dalam pohon biner dibuat. Karena proses pembagian ini, dimungkinkan untuk dengan mudah mengakses dan menyisipkan data dalam pohon biner menggunakan fungsi pencarian dan penyisipan yang dipanggil secara rekursif pada daun yang berurutan.

Keuntungan dari pohon biner (dan varian b-tree) berkaitan dengan efisiensi pembacaan disk, kecepatan menemukan dan menyajikan data. Saat data dipanggil, catatan perlu dicari untuk menemukan data. Pencarian catatan membutuhkan waktu. Dalam FAT dan NTFS, pencarian ini dapat memakan banyak waktu (relatif berbicara) karena OS mencari melalui keseluruhan sistem file.

Data dalam pohon pencarian biner disimpan dalam simpul pohon, dan harus dikaitkan dengan nilai atau kunci ordinal; kunci-kunci ini digunakan untuk menyusun pohon sedemikian rupa sehingga nilai simpul anak kiri lebih kecil dari simpul induk, dan nilai simpul anak kanan lebih besar dari simpul induk. Terkadang, kunci dan datum adalah satu dan sama. Nilai kunci tipikal mencakup bilangan bulat atau string sederhana, data sebenarnya untuk kunci akan bergantung pada aplikasi.

Untuk melakukan pencarian, pertama-tama mulai dari akar pohon, dan bandingkan nilai ordinal dari akar dengan nilai ordinal dari simpul yang akan dicari lokasinya. Jika nilai ordinal lebih kecil dari akar, ikuti cabang kiri akar, atau ikuti cabang kanan. Mulai perbandingan lagi, tetapi di cabang bandingkan nilai ordinal anak dengan node yang akan ditempatkan. Penjelajahan pohon berlanjut dengan cara ini hingga mengidentifikasi node.[4] Struktur indeks hirarkis dari pohon biner memungkinkan pencarian untuk melewati catatan subpohon kanan atau kiri pada setiap tingkat. Saat pencarian menuruni pohon, volume data yang perlu dibaca berkurang dengan cepat.

Demi argumen, katakanlah setiap divisi di dalam pohon dibagi 50/50; 50 persen catatan di sisi kiri dan 50 persen di sisi kanan. Karena pencarian akan menuruni pohon, jumlah rekaman potensial yang perlu dicari akan berkurang setengahnya di setiap lapisan. Indeks hierarkis meminimalkan jumlah pembacaan disk yang diperlukan untuk menemukan catatan tertentu. Untuk mengilustrasikan poin ini lebih lanjut, asumsikan bahwa nilai target pencarian terdapat dalam kumpulan data karakter fiksi, dengan nilai target yang diidentifikasi sebagai Holmes, Sherlock. Dalam contoh ini, kumpulan data akan dicari berdasarkan abjad, dengan tujuan untuk mengidentifikasi catatan individu untuk Holmes, Sherlock (lihat Gambar 9.7).

Gambar 9.7 Kumpulan Data Nama Karakter Fiksi

Pertama, mulailah dengan memilih elemen tengah kumpulan data (Gambar 9.8), terutama median, dan membandingkannya dengan nilai target. Jika nilainya cocok maka akan mengembalikan kesuksesan.

Gambar 9.8 Elemen Tengah dari Kumpulan Data

Dalam contoh ini nilai median "M" tidak sesuai dengan "H" dari Holmes. Jika nilai target (contoh kita "H") lebih tinggi dari nilai median ("M"), ia akan mengambil setengah bagian atas kumpulan data dan melakukan operasi yang sama terhadapnya. Demikian pula, jika nilai target lebih rendah dari nilai median ("M"), maka akan melakukan operasi terhadap setengah bagian bawah. (Lihat Gambar 9.9.)

Gambar 9.9 Nilai Target (“H”) Lebih Tinggi dari Nilai Median (“M”)

Mengingat bahwa nilai target (contoh kita "H") lebih tinggi dari nilai median ("M"), pencarian akan menyelidiki bagian atas kumpulan data. Kumpulan data dibagi lagi di median, dan nilai target ("H") sekali lagi dibandingkan dengan median yang baru dibuat (yang dalam iterasi kedua ini sekarang, "F"). (Lihat Gambar 9.10.)

Gambar 9.10 Iterasi Kedua Separuh Atas Set Data yang Akan Diselidiki

Mengingat bahwa nilai target (“H”) tidak lebih tinggi dari nilai median (“F”), pencarian tidak akan menyelidiki bagian atas kumpulan data (A, B, C, D, E,) tetapi bukan bagian bawah dari set tanggal yang tersisa (G, H, I, J, K, L). (Lihat Gambar 9.11.)

Gambar 9.11 Separuh Bagian Bawah Kumpulan Data yang Akan Diselidiki

Mengingat bahwa nilai target ("H") sekarang lebih tinggi dari nilai nilai median berikutnya ("I"), pencarian akan menyelidiki bagian atas kumpulan data (G, H, I) daripada bagian bawah dari set tanggal yang tersisa (J, K, L). (Lihat Gambar 9.12.)

Gambar 9.12 Iterasi Ketiga Binary Search

Proses pencarian biner akan terus membagi dua set data dengan setiap iterasi, hingga nilainya ditemukan atau hingga tidak dapat lagi membagi set data. Dengan iterasi keempat nilai target “H” ditemukan (lihat Gambar 9.13) dan proses pencarian awal berakhir.

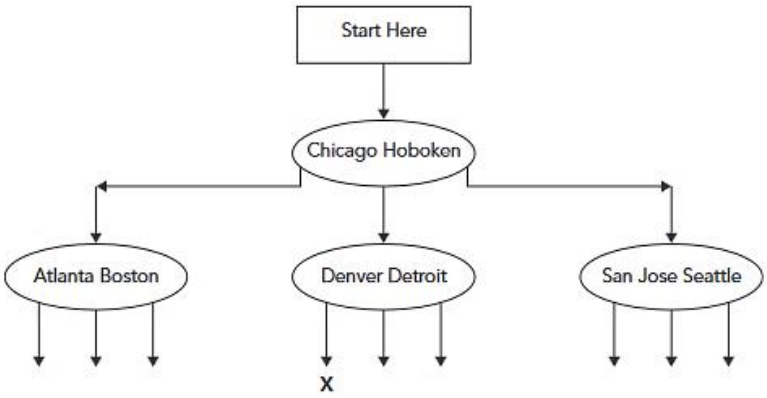
Gambar 9.13 Iterasi Keempat Binary Search

Metodologi pencarian yang sama akan dimulai lagi saat kumpulan data "H" diperiksa untuk kecocokan yang tepat dengan nilai target "Holmes".

B-tree

Sebuah b-tree secara konseptual mirip dengan pohon biner karena keduanya merupakan metode untuk menempatkan dan menemukan file, tetapi keduanya tidak analog. Perbedaan utamanya adalah b-tree berorde lebih tinggi, artinya node dapat memiliki lebih dari dua anak. Hal ini memungkinkan record ditemukan dengan melewati node yang lebih sedikit (kedalaman pohon akan lebih sedikit) daripada jika hanya ada dua anak per node. Urutan yang lebih tinggi dalam b-tree meminimalkan berapa kali media atau (node) harus diakses untuk menemukan record yang diinginkan. Dengan kata lain, lebih sedikit menulis ke disk mempercepat proses pencarian.

Setiap potongan data yang disimpan dalam b-tree disebut sebagai “key” karena setiap kunci bersifat unik dan dapat muncul di b-tree hanya di satu lokasi. Sebuah b-tree terdiri dari record “node” yang berisi kunci-kunci, dan pointer yang menghubungkan node-node b-tree secara bersamaan. Setiap b-tree terdiri dari beberapa “urutan n”, yang berarti node berisi dari n hingga 2n kunci, dan node dengan demikian selalu setidaknya setengah penuh dengan kunci. Kunci disimpan dalam urutan terurut dalam setiap node. Daftar pointer yang sesuai diselingi secara efektif di antara kunci untuk menunjukkan di mana mencari kunci jika tidak ada di node saat ini. Sebuah node yang berisi k kunci selalu berisi k + 1 pointer. Sebagai contoh, Gambar 9.14 memperlihatkan sebagian dari b-tree dengan orde 2 (node memiliki setidaknya dua kunci dan tiga pointer). Node dibatasi dengan elips. Kuncinya adalah nama kota, dan disimpan menurut abjad di setiap node. Di kedua sisi setiap kunci adalah penunjuk yang menghubungkan kunci ke node berikutnya. Untuk menemukan kunci "Des Moines", kami mulai mencari di simpul "akar" teratas. "Des Moines" tidak ada di node tetapi mengurutkan antara "Chicago" dan "Hoboken", jadi kami mengikuti penunjuk tengah ke node berikutnya. Sekali lagi, "Des Moines" mengurutkan antara "Denver" dan "Detroit" jadi kami mengikuti penunjuk pertama simpul itu ke bawah ke simpul berikutnya (ditandai dengan "X"). Akhirnya, kita akan menemukan kuncinya, atau menemukan simpul "daun" di tingkat bawah pohon-b tanpa penunjuk ke simpul yang lebih rendah dan tanpa kunci yang kita inginkan, yang menunjukkan bahwa kuncinya tidak ada di pohon-b.



Gambar 9.14 Bagian dari B-Tree dengan Urutan 2 (node memiliki setidaknya dua kunci dan tiga penunjuk)

Pencarian kunci pada b-tree selalu dimulai dari simpul akar dan mengikuti petunjuk dari simpul ke simpul hingga kunci ditemukan atau pencarian gagal karena simpul daun tercapai dan tidak ada lagi petunjuk yang harus diikuti. Bahkan b-tree yang sangat besar dapat menjamin hanya sejumlah kecil node yang harus diambil untuk menemukan kunci yang diberikan. Misalnya, b-tree dengan 10.000.000 kunci dengan 50 kunci per node tidak perlu mengambil lebih dari empat node untuk menemukan kunci apa pun.5 Jumlah maksimum anak per node adalah urutan pohon; b-tree juga biner dan karenanya dapat memiliki urutan 32, 64, 128, atau lebih (anak per node).

Di pohon, catatan (data) disimpan di lokasi yang disebut daun. Mungkin ada miliaran catatan (daun). Kedalaman berubah seiring bertambahnya record untuk memastikan bahwa b-tree berfungsi secara optimal untuk jumlah record yang dikandungnya.

Sistem File Hirarkis

Hierarchical File System (HFS) adalah sistem file yang digunakan dalam sistem Apple Macintosh. Itu diperkenalkan oleh Apple pada tahun 1995, menggantikan MFS warisan mereka (Sistem Pengarsipan Macintosh). File direferensikan dengan ID file unik daripada nama file; pengidentifikasi unik ini bisa sepanjang 255 karakter. Ini menggunakan struktur b-tree, yang merupakan varian dari sistem file b-tree. Seperti dijelaskan di atas, struktur data pohon inilah yang membuat data diurutkan dan memungkinkan pencarian cepat, penyisipan, dan penghapusan.

Sistem File UNIX

Sistem File UNIX (UFS) adalah sistem file yang digunakan oleh Sistem Operasi UNIX, juga disebut Komposisi Sistem File Berkeley. Ada yang disebut "Superblok", yang berisi "angka ajaib" yang mengidentifikasi ini sebagai sistem file UFS, serta informasi yang menjelaskan geometri, statistik, dan parameter drive; secara fungsional, ini mirip dengan Volume Boot Record, yang telah kami perincian dalam sistem pengarsipan FAT. UFS juga berisi apa yang disebut Inodes, yang berisi atribut file (mis. Metadata) yang secara fungsional mirip dengan entri direktori. Inode ini terkandung dalam grup yang disebut Grup Silinder bersama dengan Blok Data, yang berisi data aktual.

EXT2 dan EXT3

EXT adalah sistem file yang digunakan di banyak Sistem Operasi Linux. Linux mendukung banyak sistem file tetapi EXT adalah sistem file default dari sebagian besar distribusi. EXT3 adalah versi terbaru dari EXT2; EXT3 menambahkan penjurnalan sistem file; Namun, konstruksi dasarnya sama. EXT didasarkan pada Sistem File UFS tetapi, seperti halnya dengan kebanyakan Linux, banyak kerumitan UFS telah dihapus, menjaga sistem file ini relatif sederhana. Sistem pengarsipan ini membagi ruang disk menjadi Grup Blok (mirip dengan Grup Silinder UFS). Semua bagian grup blok berisi jumlah blok yang sama, kecuali yang terakhir, yang mungkin berisi lebih sedikit tergantung pada ukuran drive. Blok ini digunakan untuk menyimpan nama file, metadata, dan konten file. EXT menyimpan semua data yang berkaitan dengan file, tidak seperti FAT atau NTFS, yang menyimpan metadata di lokasi terpisah.

Analogi yang bisa diterapkan untuk sistem pengarsipan ini adalah teater. Kursi di teater dikelompokkan menjadi beberapa bagian, seperti blok pada hard drive yang dikelompokkan ke dalam Grup Blok. Setiap bagian teater (kelompok blok) diberi nomor dan setiap bagian berisi jumlah kursi (blok) yang sama. Hanya informasi tata letak dasar EXT yang disimpan dalam apa yang disebut struktur data Superblock, sekali lagi sebanding dengan Volume Boot di FAT. Konten file disimpan dalam blok, yang merupakan grup dari sektor berurutan. Blok yang digunakan di sini sebanding dengan cluster. Mirip dengan UFS, metadata untuk setiap file dan direktori juga disimpan dalam blok dalam struktur data yang disebut inode. Inode memiliki ukuran tetap dan terletak di dalam tabel inode. Ada satu tabel inode per grup blok.

9.6 RINGKASAN DAN PENGAYAAN

Berbagai sistem pengarsipan dan komponennya mungkin memiliki nama yang berbeda dan penempatan fisiknya pada drive dapat bervariasi, tetapi secara fungsional semua sistem file memerlukan bagian yang serupa: yang mengidentifikasinya, yang mengidentifikasi datanya, dan yang berisi data itu sendiri.

Kami membahas sistem file di sini untuk menjelaskan komponen ini dan bagaimana data diakses, disimpan, dihapus, dan seterusnya. Untuk keperluan teks ini, kami tidak membahas setiap sistem file yang mungkin ditemui oleh penyelidik forensik dunia maya, kami juga tidak memberikan perincian bit demi bit yang tepat dari setiap sistem file—untuk alasan yang bagus.

Buku telah ditulis dan terus ditulis tentang masing-masing sistem pengarsipan dan konsep sistem pengarsipan ini. Sistem file yang ada akan berkembang dan sistem file baru akan berkembang; kami akan terus melihat pertumbuhan ini di ponsel, iPhone, PDA, iPad, media penyimpanan, dan semua perangkat baru yang akan datang. Yang penting bagi pembaca adalah bahwa dengan membaca bab ini, Anda sekarang memiliki pemahaman yang lebih baik, lebih percaya diri, dan apresiasi yang lebih besar terhadap konsep sistem file dan fungsi yang dijalankannya.

Teknologi akan terus berkembang, begitu pula penyidik forensik dunia maya. Sebagai penyelidik, Anda akan, dalam beberapa kasus, menemukan diri Anda berpotensi tertarik untuk berspesialisasi dalam jenis sistem tertentu (mis., Windows, Unix, dll.). Saat ini terjadi, Anda akan menemukan diri Anda mahir dalam satu jenis sistem file di atas yang lain. Ini adalah

evolusi alami dari pembelajaran dan penguasaan. Dalam bidang forensik dunia maya, dan komputer pada umumnya, hampir mustahil bagi satu orang untuk mengetahui segalanya.

PENGAYAAN

Atribut Yang Ditetapkan Sistem Ntfs Umum*

Berikut adalah daftar atribut yang ditentukan sistem NTFS yang paling umum:

Informasi Standar (SI): Berisi "informasi standar" untuk semua file dan direktori. Ini termasuk properti dasar seperti cap tanggal/waktu ketika file dibuat, dimodifikasi, dan diakses. Ini juga berisi atribut seperti FAT "standar" yang biasanya dikaitkan dengan file (seperti jika file hanya-baca, disembunyikan, dan sebagainya).

Daftar Atribut: Ini adalah "meta-attribute": atribut yang menjelaskan atribut lainnya. Jika atribut harus dibuat non-resident, atribut ini ditempatkan di rekaman MFT asli untuk bertindak sebagai pointer ke atribut non-resident.

Nama File (FN): Atribut ini menyimpan nama yang terkait dengan file atau direktori. Perhatikan bahwa file atau direktori dapat memiliki beberapa atribut nama file, untuk memungkinkan penyimpanan nama file "biasa", bersama dengan alias pendek nama file MS-DOS. Menyimpan nama di Unicode.

Security Descriptor (SD): Kontrol akses dan properti keamanan file.

Nama Volume, Informasi Volume, dan Versi Volume: Ketiga atribut ini menyimpan nama kunci, versi, dan informasi lain tentang volume NTFS. Digunakan oleh file metadata $Volume.

Data: Berisi data file. Secara default, semua data dalam file disimpan dalam satu atribut data — walaupun atribut tersebut dipecah menjadi beberapa bagian karena ukuran, itu masih satu atribut.

Atribut Root Indeks: Atribut ini berisi indeks sebenarnya dari file yang terkandung dalam direktori, atau bagian dari indeks jika besar. Jika direktorinya kecil, seluruh indeks akan sesuai dengan atribut ini di MFT; jika terlalu besar, beberapa informasi ada di sini dan sisanya disimpan dalam atribut buffer indeks eksternal.

Atribut Alokasi Indeks: Jika indeks direktori terlalu besar untuk dimasukkan ke dalam atribut root indeks, catatan MFT untuk direktori tersebut akan berisi atribut alokasi indeks, yang berisi pointer ke entri buffer indeks yang berisi informasi indeks direktori lainnya.

Bitmap: Berisi bitmap alokasi cluster. Digunakan oleh file metadata $Bitmap.

- Fungsinya mirip dengan tabel FAT di FAT.
- Berisi satu bit untuk setiap cluster dalam partisi. Tidak seperti FAT, ini melacak alokasi cluster saja. Itu tidak melacak berjalan cluster. ika bit (mewakili cluster) memiliki 1, cluster dialokasikan ke file. Jika bit (mewakili cluster) memiliki 0, cluster tidak terisi, atau tersedia untuk digunakan. Sangat sederhana.

Extended Attribute (EA) dan Extended Attribute Information: Ini adalah atribut khusus yang diimplementasikan untuk kompatibilitas dengan penggunaan OS/2 dari partisi NTFS. Sebagian besar sistem file ada untuk membaca dan menulis konten file, tetapi NTFS ada untuk membaca dan menulis atribut, yang dapat berisi konten file atau konten

header (metadata). Seolah-olah NTFS tidak membedakan antara metadata dan data biasa (isi file).

Analogi Kotak

Pikirkan atribut sebagai kotak atau wadah penyimpanan. Bagian luar wadah mungkin memiliki nama, alamat, dan konten Anda tertulis di atasnya; dalamnya adalah isinya. Header bersifat umum dan standar untuk semua atribut, sedangkan konten adalah jenis atribut tertentu dan dapat berukuran berapa pun. Jadi, semua item disimpan dalam "kotak". Kotak boleh dari berbagai ukuran dan bentuk dengan isi yang berbeda, tetapi bagian luarnya berisi informasi yang sama: nama, alamat, dan isi. Jadi tajuk sama dengan bagian luar kotak dan isinya sama dengan isi kotak.

BAB 10
CYBER FORENSIK
PRAKTIK CERDAS INVESTIGATIF

Praktik "terbaik" atau bahkan "baik" biasanya didasarkan pada data, analisis analitis, garis dasar, persepsi, dan praktik yang didukung oleh tindakan atau hasil terutama dari satu perusahaan yang menarik perhatian suatu industri, yang kemudian diikuti oleh organisasi lain meniru atau mengadopsi praktik-praktik ini dari waktu ke waktu, yang pasti menghasilkan pembentukan "praktik terbaik". Tampaknya ada sesuatu yang melingkar atau bahkan incest tentang praktik terbaik.

Dalam Bab 10, konsep Praktik Cerdas Investigasi diperkenalkan. Praktik Cerdas Investigasi ini tertanam dalam "langkah" berurutan yang diambil selama proses investigasi forensik dunia maya.

Praktik cerdas menggarisbawahi fakta bahwa praktik apa pun yang layak mendapat perhatian khusus, sementara juga memanfaatkan data dan analisis analitis, biasanya harus menangkap daya cipta yang diperlukan untuk melampaui "mengikuti paket" untuk menerapkan pemikiran logis dan independen untuk mendekati dan memecahkan masalah. Ini berarti bahwa tindakan diambil karena itu adalah keputusan paling cerdas untuk dibuat, dan langkah untuk diambil, bukan hanya karena orang lain melakukannya. Praktik cerdas juga dapat dipertahankan dengan sendirinya dan tidak bergantung pada persetujuan atau penerimaan “kelompok”.

Praktik cerdas digunakan untuk membantu mengatasi hambatan inovasi. Seiring kemajuan dan perubahan teknologi hampir setiap hari, tantangan teknis dan skenario investigasi yang dihadapi oleh penyelidik forensik dunia maya akan memerlukan implementasi dan penggunaan praktik cerdas untuk menentukan tindakan terbaik untuk kasus yang dihadapi, yang mungkin sangat berbeda. pendekatan atau serangkaian tindakan untuk kasus berikutnya. Meskipun demikian, mengikuti proses merupakan bagian integral dari penyelidikan sukses kasus apapun.

Ingat dari kasus yang sedang berlangsung: Ronelle Sawyer sedang menyelidiki apakah Jose McCarthy berpotensi terlibat dalam distribusi kekayaan intelektual organisasinya secara tidak sah kepada pesaing, Janice Witcome, direktur pelaksana Perusahaan XYZ. Ronelle bekerja untuk perusahaan Fortune 500 bernama ABC Inc. Perusahaan Ronelle bersifat global, sangat besar, dan terdepartementalisasi. Departemen-departemen ini "terdiam" sedemikian rupa sehingga masing-masing melakukan satu tugas dan hanya tugas itu. Ada prosedur, proses, dan panduan untuk setiap aktivitas, termasuk forensik siber.

Penting bagi Ronelle, sebagai penyelidik forensik dunia maya, mengikuti proses forensik yang ditentukan. Jika ada, penyelidik harus mengikuti proses yang ditetapkan oleh praktik forensik standar yang diadopsi oleh organisasi, departemen, atau lembaga penyelidik.

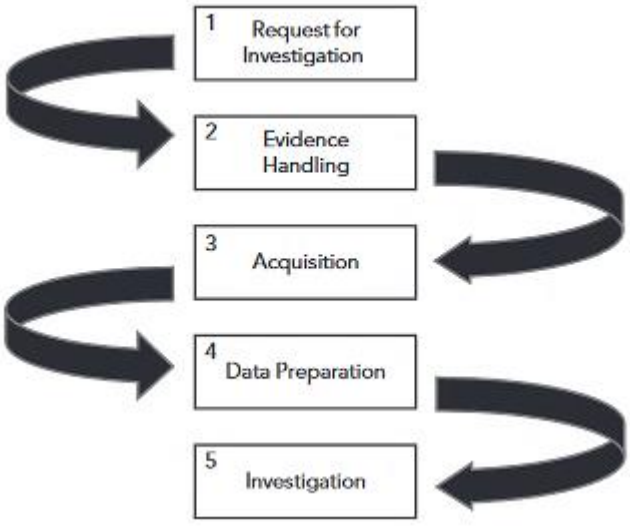
Apapun proses forensik yang diikuti, itu perlu diikuti dari awal penyelidikan, kasus, atau insiden sampai ke kesimpulan logis dari kasus, investigasi, atau insiden, yang bahkan mungkin mengharuskan penyelidik forensik untuk hadir sebagai saksi ahli di pengadilan.

10.1 PROSES FORENSIK

Proses ini, Investigative Smart Practices, bagaimanapun, tidak perlu merinci dengan tepat setiap prosedur langkah demi langkah dari investigasi forensik dunia maya, karena ini bisa menjadi latihan yang sia-sia; ada begitu banyak arah yang berbeda di mana investigasi dapat berkembang, dan upaya untuk mengikuti pola investigasi tertentu atau menyesuaikan setiap investigasi ke dalam pendekatan investigasi yang sama dapat menghasilkan bencana. Selain itu, prosedur seperti itu akan segera ketinggalan zaman karena teknologi investigasi baru pasti akan berkembang.

Selama liputan kami tentang konsep dasar cyber forensik di bab-bab sebelumnya, kami kadang-kadang bergabung dengan Ronelle selama penyelidikannya. Investigasi yang sebenarnya hanyalah satu bagian dari proses investigasi ini. Mungkin bagian yang membayar tagihan; namun, ada komponen lain dalam proses investigasi yang sama pentingnya. Potongan-potongan lain ini mungkin tidak memerlukan penemuan "jarum di tumpukan jerami" atau "pistol merokok", namun keduanya sama pentingnya. Mengapa? Salah satu langkah yang dilakukan secara tidak benar, sembarangan, atau dilewati dapat menggagalkan dan membatalkan seluruh penyelidikan, terlepas dari temuan investigasi pada akhirnya.

Secara umum, proses investigasi forensik siber kemungkinan menggabungkan kombinasi dari langkah-langkah awal berikut (lihat Gambar 10.1). Namun, Praktik Cerdas Investigasi ini tidak dimaksudkan untuk diikuti secara berurutan, juga tidak eksklusif satu sama lain. Sebagaimana dinyatakan, setiap penyelidikan harus dievaluasi dan diproses berdasarkan kemampuannya sendiri dan langkah-langkah yang diambil selama penyelidikan harus memastikan bahwa prosedur penyelidikan forensik dunia maya organisasi atau departemen diikuti dengan jelas, dan bahwa tidak ada pertanyaan mengenai kelengkapan proses yang diikuti atau keakuratan bukti yang dikumpulkan.



Gambar 10.1 Alur Proses Investigasi Umum

Setiap kasus forensik harus dianggap unik, karena masing-masing dapat menjangkau spektrum dan keragaman topik: perceraian, pembunuhan, penganiayaan anak, pencurian kekayaan intelektual, masalah majikan / karyawan, pelecehan seksual, pemerasan dunia maya, spionase industri, dan daftar lainnya. aktif (hampir tak terbatas).

Ini bisa menjadi kasus perceraian berdasarkan klaim pelecehan seksual oleh seorang karyawan. Atau mungkin seorang pekerja yang tidak puas mencuri salinan rencana pemasaran strategis majikannya dan menawarkannya untuk dijual kepada pesaing, atau seorang anak membuat teman online baru yang kemudian menjadi orang dewasa dengan keyakinan sebelumnya untuk ajakan anak di bawah umur. Beberapa kasus mungkin mencakup lebih dari satu topik; misalnya, kasus tersebut mungkin melibatkan majikan yang menyelidiki pencurian kekayaan intelektual dan spionase industri karyawan, seperti halnya penyelidikan Ronelle. Kemungkinannya tidak terbatas.

Untuk menambah berbagai jenis investigasi, ada juga berbagai praktik investigasi. Penyelidik forensik dunia maya bekerja di semua bidang dan di semua tingkatan dalam masyarakat: federal, negara bagian, kota, dalam penegakan hukum, di bisnis kecil independen, dan perusahaan global. Masing-masing bidang ini dibatasi oleh aturan, peraturan, hukum, dan prosedur yang berbeda. Penyelidik forensik dunia maya akan menghadapi rintangan yang berbeda tergantung pada organisasi, negara bagian, dan bahkan negara tempat mereka bekerja.

Penting untuk diingat bahwa penyelidik forensik terikat oleh prosedur. Jika organisasi telah mengembangkan prosedur tertulis langkah demi langkah, maka penyelidik organisasi diharuskan mengikuti langkah-langkah tersebut. Kesuksesan atau kegagalan suatu kasus dapat sangat bergantung pada kepastian bahwa penyelidik forensik telah mengikuti serangkaian prosedur tertulis tertentu. Pengacara yang cerdas dapat membuat penyelidik forensik pihak lawan terlihat bodoh. Seseorang dapat dengan mudah membayangkan dialog pemeriksaan silang: "Jika Anda bahkan tidak mengikuti prosedur Anda sendiri, bagaimana kami dapat memastikan bahwa Anda mengikuti prosedur Federal yang ada?" Dalam beberapa kasus, tidak mengikuti prosedur tertulis Anda sendiri dapat disamakan dengan tidak mengikuti hukum. Bersiaplah untuk mengikuti prosedur forensik tertulis.

Proses di mana prosedur dapat dirumuskan akan bergantung pada masing-masing organisasi, dengan mempertimbangkan undang-undang yang menggantikan dan memerintahkan organisasi untuk mengikuti undang-undang ini. Namun, ada beberapa langkah umum atau praktik cerdas, pedoman, yang harus dilakukan dan menjadi bagian dari penyelidikan forensik dunia maya yang terorganisir dengan baik. Maklum, langkah-langkah ini mungkin tidak terjadi secara berurutan; beberapa mungkin terjadi bersamaan dengan langkah lain, atau di seluruh proses.

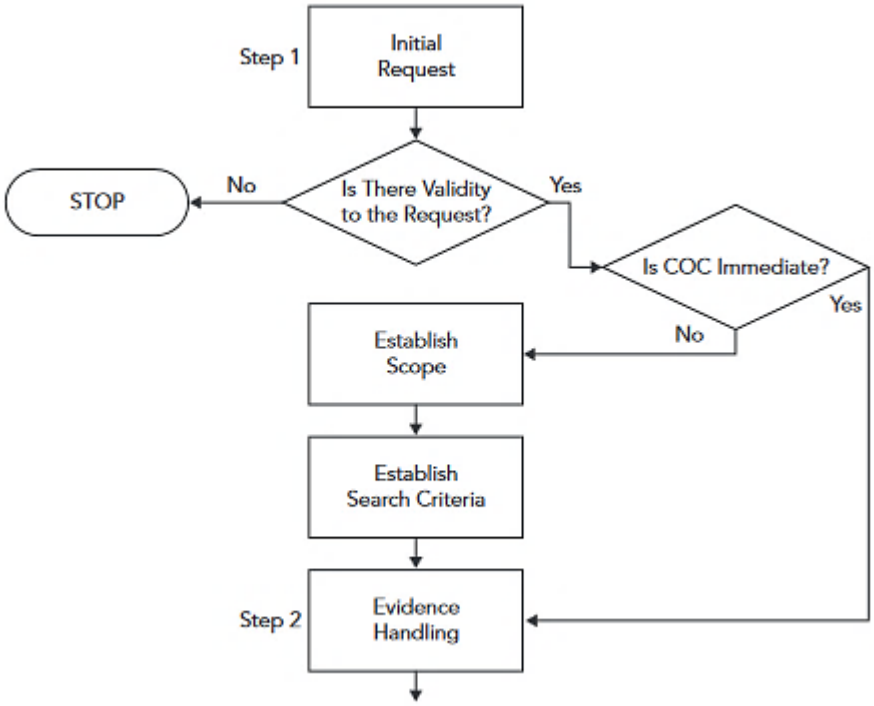
Praktik Cerdas Investigasi ini bukanlah aturannya melainkan praktik cerdas yang disarankan. Mungkin ada situasi atau organisasi yang memerlukan langkah-langkah tambahan, menggabungkan langkah-langkah, atau mengatur ulang langkah-langkah yang ditunjukkan pada Gambar 10.1 sampai tingkat tertentu; namun, melewatkan satu atau beberapa langkah atau tidak melakukan langkah tertentu dapat menyebabkan potensi bencana.

10.2 PRAKTIK CERDAS INVESTIGASI FORENSIK

Tujuan: Untuk menentukan validitas permintaan investigasi dan untuk menetapkan ruang lingkup investigasi.

Langkah 1: Kontak Awal, Permintaan

Agar ada penyelidikan, seseorang harus maju dan membuat permintaan (lihat Gambar 10.2). Biasanya terdapat dua pihak atau pihak yang berlawanan, yaitu pemohon dan sasaran atau subjek. Subjek adalah orang yang sedang diselidiki. Bergantung pada kasusnya, mungkin pasangan, rekan kerja, kontraktor, atau seseorang di negara asing yang telah menulis beberapa kode berbahaya. Subjek dapat diberi label "tersangka" tetapi menamainya seperti itu mungkin lancang atau mengarah pada subjektivitas.



Gambar 10.2 Langkah 1: Kontak Awal, Permintaan

Biasanya ada beberapa “dokumen permintaan”, yang menetapkan dasar atau pembenaran untuk melakukan penyelidikan. Contoh dokumen spesifik yang mungkin menjadi dasar untuk melakukan penyelidikan termasuk, namun tidak terbatas pada, hal-hal berikut:

- Surat Perikatan
- Kontrak
- Permintaan Resmi (Perusahaan).
- Panggilan pengadilan
- Surat Perintah Penggeledahan
- Perintah Pengadilan

Jika ada pembenaran untuk keluhan atau alasan tertentu untuk menyelidiki, harus ada juga aturan, undang-undang, kebijakan perusahaan, prosedur, atau garis dasar yang menjadi dasar pengajuan permintaan penyelidikan. Sekali lagi, jenis dokumen khusus yang diperlukan untuk

memulai penyelidikan forensik akan bergantung pada jenis organisasi dan mungkin jenis kasusnya.

Misalnya, sebelum praktik forensik di lingkungan perusahaan besar dapat memulai penyelidikan, formulir persetujuan yang diserahkan melalui departemen Hukum atau departemen Sumber Daya Manusia mungkin diperlukan. Penegakan hukum mungkin memerlukan panggilan pengadilan atau surat perintah penggeledahan sebelum penyelidikan dapat dilakukan. Praktik forensik independen kecil mungkin memerlukan surat perikatan atau kontrak, dari klien, sebelum memulai penyelidikan.

Di sisi lain, pemohon (misalnya, manajemen eksekutif, mitra bisnis, pasangan, atau pihak ketiga yang dirugikan) dapat meminta penyelidik forensik cyber independen untuk menandatangani perjanjian kerahasiaan (NDA), karena dia tidak akan diragukan lagi dibuat rahasia untuk informasi hak milik. Masing-masing skenario forensik ini sangat berbeda dan membutuhkan penanganan yang unik.

Di sinilah legitimasi dan ruang lingkup investigasi ditentukan. Seorang hakim dapat menentukan keabsahan penyelidikan penegakan hukum dengan mengeluarkan surat perintah penggeledahan, tetapi tidak demikian halnya dengan penyelidikan internal perusahaan.

Aturan atau pedoman untuk investigasi perusahaan mungkin tidak memerlukan kontrol ketat seperti itu; mungkin formulir permintaan sederhana yang disetujui oleh departemen Hukum. Namun, penyelidik forensik dunia maya harus mendapat saran dari penasihat hukum sebelum melanjutkan penyelidikan yang melibatkan pengaksesan barang-barang pribadi seseorang. Di Amerika Serikat, misalnya, pengumpulan dan pemeriksaan bukti tidak boleh melanggar (sejak penulisan teks ini) sebagai berikut:

- Amandemen Keempat
- Undang-Undang Perlindungan Privasi
- Undang-Undang Privasi Komunikasi Elektronik

Pencarian tempat kerja tanpa jaminan oleh majikan swasta jarang melanggar Amandemen Keempat. Selama pemberi kerja tidak bertindak sebagai instrumen atau agen Pemerintah AS pada saat penggeledahan, dan penggeledahan tersebut merupakan penggeledahan pribadi, Amandemen Keempat tidak akan berlaku.

Penyelidik forensik dunia maya juga akan menghadapi tantangan unik lainnya. Apakah pemohon memiliki sistem dan memiliki wewenang untuk meminta penyelidikan forensik? Dalam perceraian, apakah sistem yang memerlukan penyelidikan menjadi milik pasangan pemohon? Jika demikian, penyelidik forensik mungkin melanggar undang-undang privasi dengan melakukan penyelidikan semacam itu tanpa wewenang, otorisasi, perintah pengadilan, dll., dan dengan demikian melanggar hukum. Cakupan atau legitimasi mungkin tidak selalu berada di tangan penyelidik forensik dunia maya.

Beberapa item tambahan yang harus ditangani sebelum memulai penyelidikan forensik dunia maya meliputi:

1. Tentukan apakah layak untuk melanjutkan penyelidikan berdasarkan informasi yang dikumpulkan, legitimasi dan/atau kredibilitas pengaduan, paparan terhadap organisasi, dan seterusnya.
2. Manfaat dan/atau risiko yang terlibat dalam melakukan penyelidikan.

3. Kewajiban dan/atau risiko karena tidak melanjutkan penyelidikan.
4. Kewajiban untuk mengejar (etis/moral).
5. Apakah tersedia sumber daya yang cukup untuk melaksanakan investigasi dengan sukses?

Sangat penting bagi penyelidik forensik dunia maya untuk diberikan ruang lingkup investigasi dari manajemen, penegak hukum, dan sebagainya, atau untuk mengembangkan ruang lingkup investigasi berdasarkan bukti awal yang disajikan, pelanggaran hukum, dll. Hal ini dapat melibatkan pertemuan awal di mana pemohon yang mencari investigasi forensik dunia maya memberikan penyelidik forensik dengan parameter investigasi tertentu, luas dan dalamnya bidang investigasi yang dimaksud, kriteria pencarian, hukum yang berlaku, dll., sehingga menetapkan ruang lingkup investigasi .

Minimal kriteria pencarian dapat mencakup beberapa komponen:

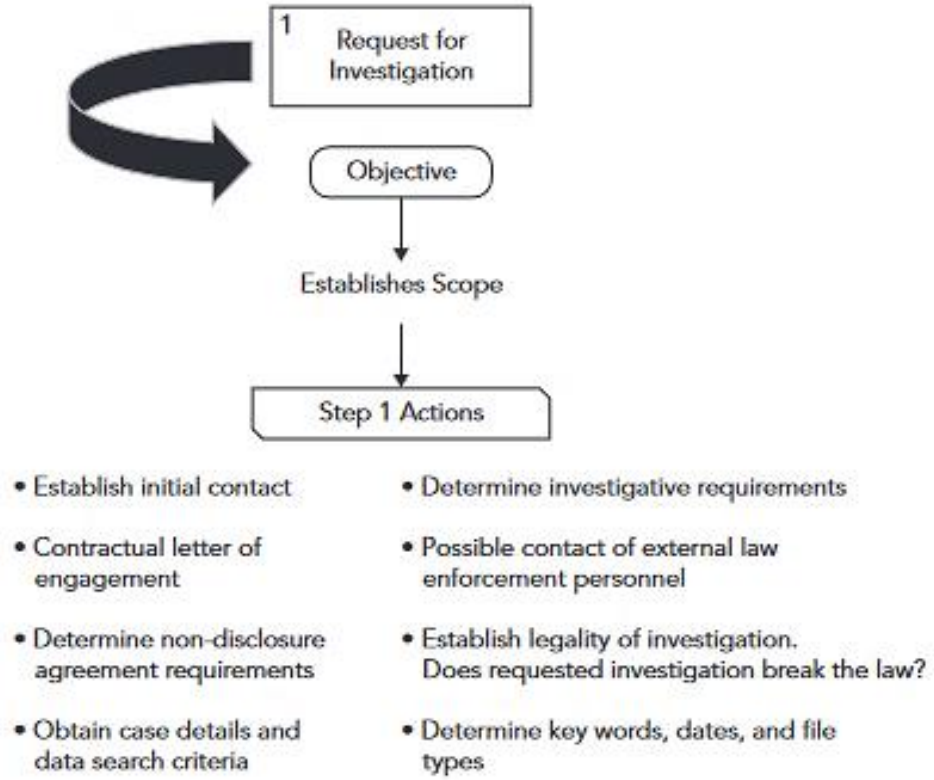
1. Kata kunci yang terkait dengan kasus yang sedang diselidiki (nama personel, nama kode proyek perusahaan, nama pesaing, frasa, dll.).
2. Tanggal atau rentang tanggal di mana tindakan yang sedang diselidiki dilaporkan telah terjadi.
3. Jenis file (.docx, .doc, .xls, .pdf, .nef, .tif, .png, .jpg, .wav, .mp4, dll.) terkait dengan kasus yang sedang diselidiki.

Beberapa penyelidik mungkin sangat terikat dengan kriteria ini dan yang lain mungkin tidak. Beberapa permintaan investigasi mungkin cukup umum sehingga hanya rentang tanggal yang luas dan jenis file umum yang mungkin diperlukan. Kasus penganiayaan anak yang dilakukan oleh penegak hukum tidak boleh meminta jenis file tertentu, karena penyelidik mungkin tertarik untuk meninjau file apa pun pada perangkat keras subjek yang disita.

Semua data, apa pun jenisnya, dapat dianggap adil, baik itu video, gambar, email, artefak Internet, atau transkrip obrolan. Di sisi lain, beberapa kasus, yang melibatkan panggilan pengadilan misalnya, mungkin sangat terikat oleh kriteria yang tepat, seperti rentang waktu dan tanggal. Melampaui periode waktu atau tanggal tertentu akan menjadi pelanggaran terhadap perintah pengadilan (yaitu panggilan pengadilan). Sekali lagi, setiap kasus atau situasi adalah unik. Dalam pengaturan perusahaan, di mana sistem, aplikasi pada sistem tersebut, dan data semuanya dimiliki secara eksklusif oleh organisasi, mungkin ada lebih banyak fleksibilitas mengenai proses investigasi.

Namun, organisasi perlu memberi tahu karyawannya bahwa sistem adalah milik organisasi seperti halnya semua konten dan datanya (undang-undang pemberitahuan berbeda-beda di Amerika Serikat menurut negara bagian; oleh karena itu, penting untuk memverifikasi undang-undang mana yang mungkin diperlukan organisasi Anda. untuk mematuhi). Sebuah organisasi biasanya akan menyajikan spanduk pemberitahuan ini kepada karyawan, sebagai layar pop-up tepat sebelum login sehingga karyawan tersebut harus mengonfirmasi pengetahuan tentang kepemilikan organisasi atas sistem dan semua data pada sistem tersebut, dan menerima kondisi ini, agar untuk berhasil masuk. Pengakuan dan penerimaan oleh karyawan ini memberi organisasi hak untuk melakukan apa pun yang diinginkannya dengan properti digital mereka.

Karena itu, permintaan perusahaan dapat (dan biasanya) seluas dan seumum mungkin sesuai kebutuhan (lihat Gambar 10.3).



Gambar 10.3 Langkah 1: Permintaan Investigasi

Dalam penyelidikan kami yang sedang berlangsung yang dilakukan oleh Ronelle terhadap aktivitas karyawan Jose McCarthy, prosedur perusahaan menyatakan bahwa semua permintaan penyelidikan diarahkan terlebih dahulu melalui departemen Hukum perusahaan. Departemen Hukum memutuskan keabsahan investigasi, dan apakah akan dilanjutkan atau tidak. Departemen Hukum juga memutuskan ruang lingkup penyelidikan tetapi, karena mereka tidak melakukan penyelidikan yang sebenarnya, kadang-kadang, kriteria pencarian khusus yang diperlukan untuk penyelidikan yang efisien mungkin tidak diperoleh dengan cara yang paling membantu Ronelle dalam penyelidikannya. - tions (misalnya, departemen Hukum kemungkinan akan kekurangan kemampuan teknis untuk mengajukan pertanyaan forensik cyber terbaik).

Manajemen senior dalam ABC Inc. memutuskan bahwa kekhawatiran awal mereka dibenarkan dan memutuskan untuk meminta penyelidikan forensik dunia maya penuh. Prosedur menentukan persetujuan SDM dan Hukum. Permintaan diajukan dan disetujui oleh kedua departemen. Kasus tersebut kemudian dikirim ke departemen forensik ABC Inc. dan ditugaskan ke penyelidik forensik dunia maya Ronelle Sawyer sebagai penyelidik utama.

Setelah diskusi lebih lanjut dengan manajemen senior, Ronelle memperoleh rincian lebih lanjut terkait kasus tersebut dan dapat menguraikan kriteria ruang lingkup investigasi berikut:

Lingkup Investigasi

Rentang Tanggal—Periode waktu: dari tanggal mulai karyawan Jose McCarthy hingga tanggal penghentiannya—10 Mei 2008 hingga 26 Maret 2009.

Kata kunci—"Janice", "Witcome", dan "XYZ". Catatan: Kata kunci sebenarnya tidak disertakan dengan permintaan, tetapi Ronelle dapat membuat daftar kata kunci dari detail yang diberikan oleh manajemen di ABC, manajer langsung Jose, dan dari penasihat umum ABC.

Jenis File—Dokumen: pengolah kata dan spreadsheet. Ronelle bertemu lebih lanjut dengan departemen Hukum untuk menanyakan apakah ada jenis file tertentu yang harus dia fokuskan, karena informasi ini pada awalnya tidak disediakan oleh manajemen ABC dan dalam banyak kasus dapat dihilangkan. Jenis file juga bisa APAPUN dan SEMUA; sekali lagi, setiap kasus berbeda.

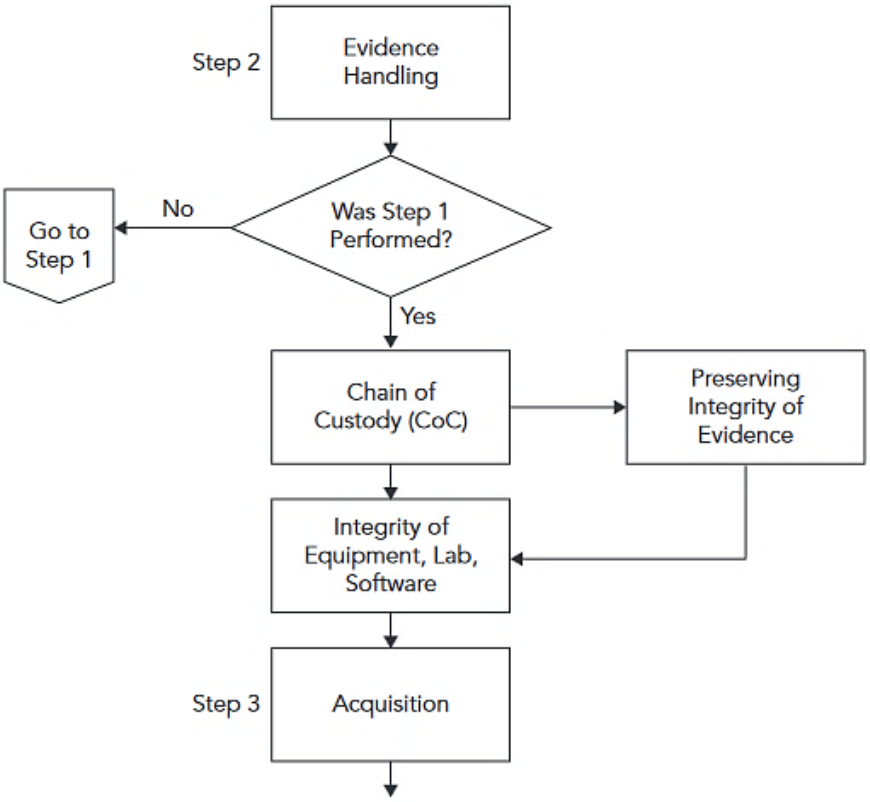
Tujuan dalam langkah ini adalah untuk menentukan ruang lingkup penyelidikan. Cakupan atau batasan dimulai dengan menentukan apakah kasus tersebut sah. Jika diputuskan bahwa kasus tersebut tidak sah, tidak melanggar hukum, tidak memiliki manfaat, atau tidak didukung oleh manajemen, maka kebutuhan atau alasan penyelidikan harus diperiksa kembali, karena tidak ada ruang lingkup penyelidikan yang telah ditentukan.

Langkah ini dapat, dan bergantung pada kasusnya, mungkin diperlukan, dibagikan dengan orang lain di seluruh organisasi (misalnya, Sumber Daya Manusia, Keamanan TI, Operasi) atau secara eksternal ke organisasi (penegakan hukum). Departemen Hukum Ronelle memutuskan, berdasarkan bukti awal yang diberikan, bahwa kasus awal terhadap Jose McCarthy pantas dan sah, sehingga membuka jalan untuk memulai penyelidikan forensik dunia maya secara penuh.

Langkah 2: Penanganan Bukti

Tujuan: Untuk menjaga keutuhan bukti; dan mampu membuktikan integritas bukti, di pengadilan hukum.

Pada langkah proses investigasi inilah bukti (biasanya digital tetapi mungkin juga hard copy) sebenarnya diserahkan kepada penyelidik forensik dunia maya. Langkah penanganan barang bukti terjadi terus menerus selama barang bukti berada di tangan penyidik forensik siber. Awalnya melibatkan pengambilan bukti, mungkin mengambil foto bukti, mendokumentasikan kondisi bukti, dan mencatat kerusakan yang sudah ada sebelumnya atau tanda-tanda aneh. Sebagian besar, jika tidak semua, interaksi penyelidik forensik dunia maya dengan bukti dapat termasuk dalam langkah penanganan bukti. Fokus khusus penanganan bukti, seperti yang disajikan di sini, adalah pergerakan bukti, biasanya bersifat fisik versus elektronik (lihat Gambar 10.4).



Gambar 10.4 Langkah 2: Penanganan Barang Bukti

Penegakan hukum mungkin paling wajib, karena persyaratan khusus dan kepatuhan hukum, untuk mengikuti prosedur penanganan barang bukti yang ketat. Misalnya, beberapa penyelidik forensik dunia maya di lingkungan perusahaan mungkin melewatkan memotret dan mendokumentasikan tanda yang tidak biasa pada hard drive yang terdapat di lingkungan mereka, di mana ini mungkin merupakan prosedur standar oleh profesional penegak hukum.

Kasus-kasus yang diajukan ke penegak hukum kadang-kadang bersifat lebih serius dan cenderung kriminal versus kasus-kasus yang biasanya muncul dalam lingkungan perusahaan. Konsekuensi kesalahan penanganan bukti dalam investigasi kriminal mungkin lebih tinggi, sehingga persyaratan untuk kontrol lebih tinggi. Prinsip yang diterima secara umum dalam forensik dunia maya adalah lebih baik terlalu berhati-hati; mendokumentasikan setiap langkah, memotret setiap bagian, membuat banyak catatan, dan seterusnya, daripada kurang rajin dan gagal mendokumentasikan bukti yang berpotensi kritis.

Penanganan bukti sedang berlangsung dan terjadi selama penyelidikan. Ini mencakup bagaimana bukti disimpan saat tidak digunakan dan bagaimana bukti ditangani saat digunakan. Penanganan barang bukti juga dapat mencakup penanganan peralatan dan alat penyelidik forensik dunia maya. Alat ini mencakup apa pun yang akan terhubung ke, dilampirkan, atau berinteraksi dengan bukti, perangkat lunak, atau perangkat keras. Misalnya, pemblokir penulisan perangkat keras yang tidak dirawat dengan benar mungkin dapat menyebabkan korslet papan sirkuit pada hard drive bukti. Kesalahan penanganan alat tersebut dapat menyebabkan potensi kerusakan atau penghancuran barang bukti.

Peralatan apa pun yang digunakan penyelidik forensik dunia maya untuk melakukan penyelidikan terhadap barang bukti perlu didokumentasikan; ini termasuk perangkat keras, perangkat lunak, dan media penghubung. Peralatan ini harus dipelihara dengan baik dan berfungsi dengan baik. Nama, versi, upgrade, model, dan informasi relevan lainnya yang berkaitan dengan peralatan yang digunakan (perangkat keras dan perangkat lunak) dapat dianggap penting dan oleh karena itu didokumentasikan per kasus. Berbulan-bulan dan mungkin bertahun-tahun berlalu setelah penyelidikan, mungkin tidak mudah untuk mengingat versi perangkat lunak mana yang digunakan untuk penyelidikan.

Tahap selanjutnya dari penanganan barang bukti, saat kasusnya selesai dan berakhir, mungkin melibatkan pengembalian barang bukti kepada pemilik aslinya, jika diperbolehkan oleh hukum. Setiap kali kepemilikan barang bukti berpindah tangan, baik saat diterima atau dikembalikan, lacak balak harus ditegakkan. Lacak balak adalah prosedur penanganan barang bukti yang melacak barang bukti saat barang bukti berpindah kepemilikan. Lacak balak biasanya berbentuk formulir atau semacam dokumen. Formulir biasanya dilibatkan karena tanda tangan dari penerima dan pemberi bukti diperlukan.

Formulir lacak balak biasanya berisi beberapa bidang berikut:

- Nomor kasus atau penugasan
- Tanggal
- Waktu
- Nomor seri/model
- Bidang deskripsi
- Lokasi

Formulir lacak balak memiliki banyak tampilan. Dalam beberapa kasus, formulir dapat dipecah menjadi dua bagian terpisah:

1. Deskripsi barang—menjelaskan barang yang ditransfer. Ini akan mencakup nomor seri dan model, pembuatan, dan deskripsi (lihat Gambar 10.5).
2. Tanda tangan—Transfer hak asuh—bagian ini mencakup tanggal dan waktu transfer dan tanda tangan pemberi pelepasan dan tanda tangan penerima (lihat Gambar 10.6).

Kemasan #:	Keterangan:		
Membuat:	Model:	Serial #:	Item Agensi #:
Kemasan #:	Keterangan:		
Membuat:	Model:	Serial #:	Item Agensi #:
Kemasan #:	Keterangan:		
Membuat:	Model:	Serial #:	Item Agensi #:

Gambar 10.5 Formulir Lacak Balak (Bagian 1)

Paket #	Tanggal Waktu	Diterbitkan oleh	Diterima oleh	Alasan
	Tanggal	Nama/Agensi	Nama/Agensi	

	Waktu	Tanda tangan	Tanda tangan	
	Tanggal	Nama/Agensi	Nama/Agensi	
	Waktu	Tanda tangan	Tanda tangan	
	Tanggal	Nama/Agensi	Nama/Agensi	
	Waktu	Tanda tangan	Tanda tangan	

Gambar 10.6 Formulir Lacak Balak (Bagian 2)

Tujuan utama penanganan bukti adalah memastikan bahwa data tidak diubah, dimanipulasi, atau diubah dengan cara apa pun dari bentuk aslinya, sehingga pelestarian dan pelacakan bukti.

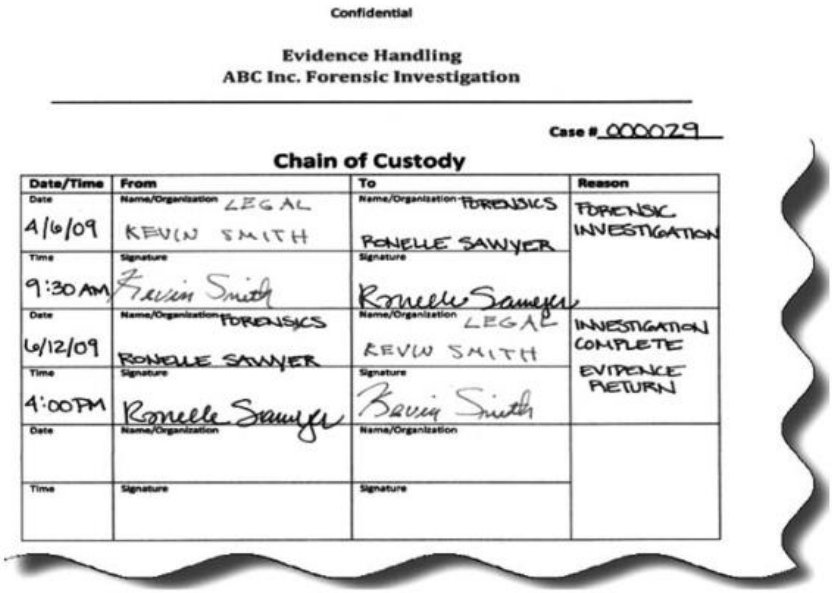
Maklum: Seperti yang telah dijelaskan sebelumnya, penanganan barang bukti tidak harus mengikuti permintaan awal secara berurutan. Langkah-langkah ini dapat terjadi secara bersamaan, atau dalam beberapa kasus lacak balak atau penanganan barang bukti dapat terjadi terlebih dahulu. Dalam kasus di mana penegak hukum dipanggil ke TKP, para profesional ini mungkin harus menangani dan mengumpulkan bukti dengan baik sebelum diarahkan oleh pengadilan tentang apa yang perlu dilakukan dengan bukti tersebut.

Hal penting untuk diingat adalah lacak balak perlu terjadi sebelum, selama, dan setelah transfer barang bukti. Memulai proses lacak balak awal tidak boleh terjadi setelah bukti diperiksa, karena kemampuan untuk membuktikan siapa yang memiliki akses ke bukti akan memungkinkan akuntabilitas atas keutuhan bukti. Akuntabilitas ini merupakan langkah penting dalam melestarikan bukti (lihat Gambar 10.7).

Gambar 10.7 Langkah 2: Penanganan Barang Bukti

Ronelle menerima bukti berupa hard drive 80GB dari departemen Hukumnya. Itu dikirim dengan tangan dalam wadah tertutup. Setelah diterima, Ronelle membuka wadah dan menemukan hard drive dan formulir lacak balak. Formulir sudah diisi dan menunggu tanda tangan Ronelle.

Ronelle memberi tanggal pada formulir itu, menandatangani namanya di bidang penerima, dan menyalinnya. Ronelle menyimpan yang asli (untuk penyimpanan catatan internal dan untuk memverifikasi jejak audit dan lacak balak "jejak") dan mengembalikan salinannya ke pengirim untuk catatan mereka. (Lihat Gambar 10.8.)



Confidential

Evidence Handling
ABC Inc. Forensic Investigation

Case # 000029

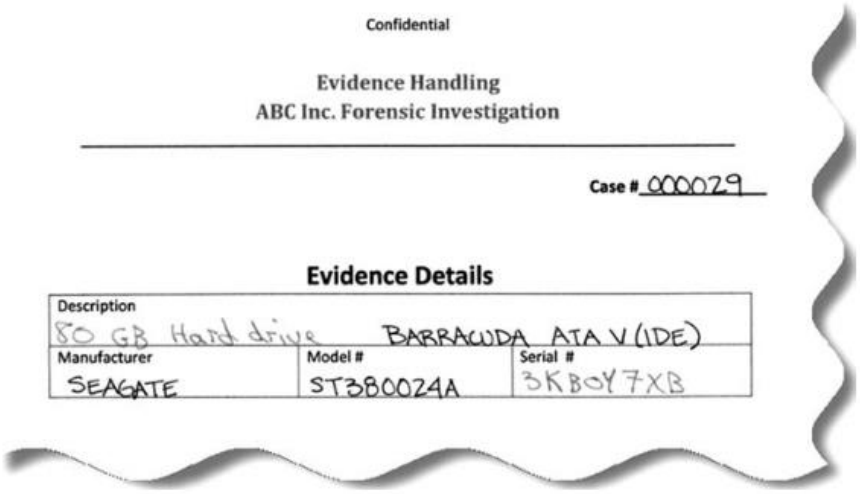
Chain of Custody

Date/Time	From	To	Reason
Date 4/6/09	Name/Organization LEGAL KEVIN SMITH	Name/Organization FORENSICS RONELLE SAWYER	FORENSIC INVESTIGATION
Time 9:30 AM	Signature Kevin Smith	Signature Ronelle Sawyer	
Date 6/12/09	Name/Organization FORENSICS RONELLE SAWYER	Name/Organization LEGAL KEVIN SMITH	INVESTIGATION COMPLETE EVIDENCE RETURN
Time 4:00 PM	Signature Ronelle Sawyer	Signature Kevin Smith	
Date	Name/Organization	Name/Organization	
Time	Signature	Signature	

Gambar 10.8 Langkah 2: Formulir Lacak Balak—Bidang Tanda Tangan

Departemen Hukum mengisi bagian "Dari" pada dokumen lacak balak bersama dengan deskripsi singkat dan nomor seri hard drive; namun, sisa formulir dibiarkan kosong. Ini tidak biasa, karena sebagian besar responden pertama yang pada awalnya bertanggung jawab untuk mengumpulkan bukti mungkin tidak memiliki latar belakang teknis untuk melampaui detail identifikasi makro dasar dan mungkin menganggapnya terlalu detail.

Ronelle mengonfirmasi bahwa nomor seri cocok dengan formulir lacak balak dan hard drive yang ada di dalam wadah dan melanjutkan untuk mengisi baris yang tersisa dari dokumen transfer hak asuh, seperti model hard drive dan detail untuk mendeskripsikan dan mengidentifikasi hard drive , di kolom deskripsi. (Lihat Gambar 10.9.)



Confidential

Evidence Handling
ABC Inc. Forensic Investigation

Case # 000029

Evidence Details

Description		
80 GB Hard drive BARRACUDA ATA V (IDE)		
Manufacturer	Model #	Serial #
SEAGATE	ST380024A	3KBOY7XB

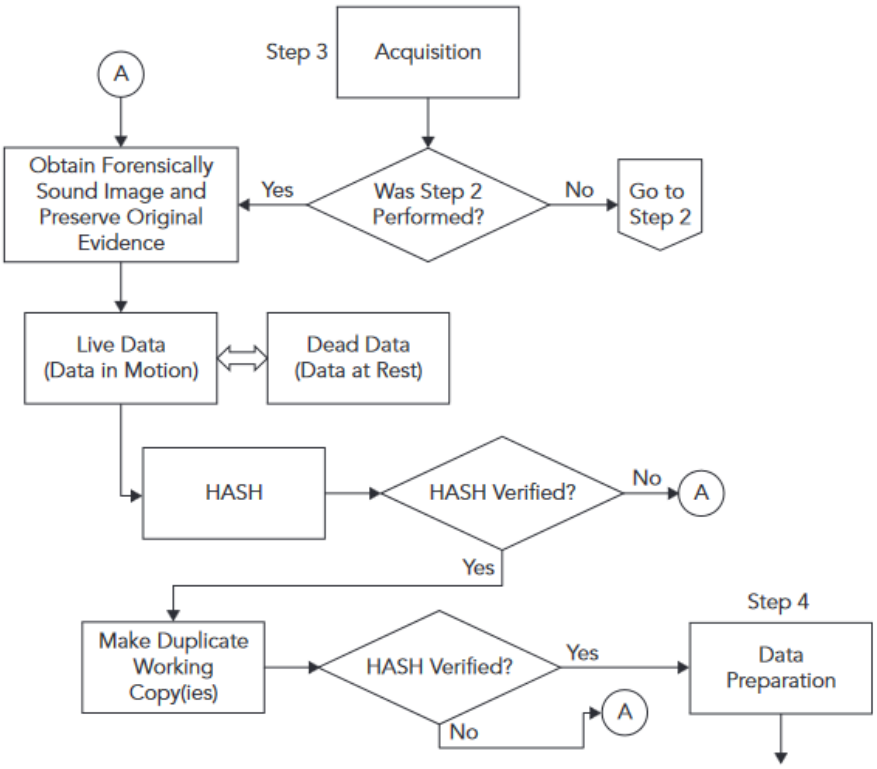
Gambar 10.9 Langkah 2: Formulir Lacak Balak—Detail Bukti

Setelah menyelesaikan dokumen lacak balak, Ronelle mengamankan bukti (hard drive) dengan menyegelnya di wadahnya dan menguncinya di lemari besi forensik, di dalam laboratorium forensik.

Langkah 3: Perolehan Bukti

Tujuan: Memperoleh citra yang sehat secara forensik dan mempertahankan integritas bukti asli.

Akuisisi bukti umumnya mengacu pada pencitraan bukti dengan cara yang sehat secara forensik. Cara yang sehat secara forensik adalah cara di mana bukti tidak diubah. Tidak ada yang ditulis, diubah, diubah, atau dimodifikasi pada barang bukti (lihat Gambar 10.10 dan 10.11).

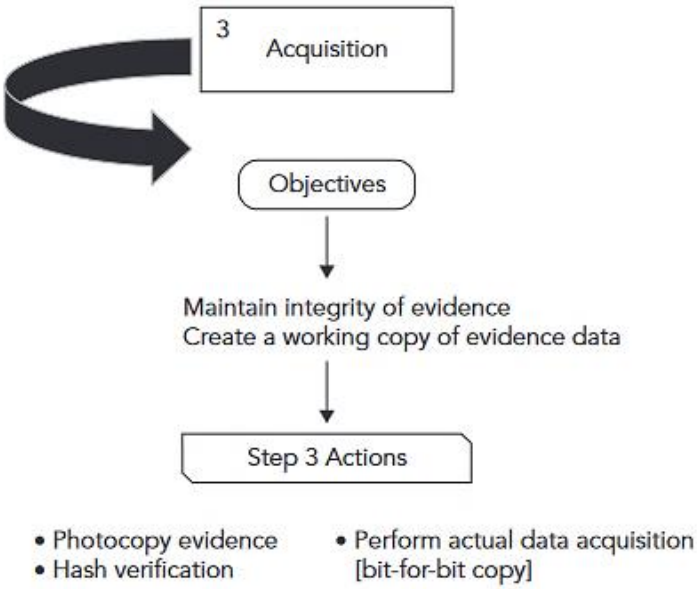


Gambar 10.10 Langkah 3: Proses Akuisisi

Akuisisi bukti dapat, secara teori, termasuk dalam bagian penanganan bukti sebelumnya. Sebagian besar, jika tidak semua, interaksi penyelidik forensik dunia maya dengan bukti dapat berada di bawah penanganan bukti. Seperti yang telah disebutkan sebelumnya, bagian penanganan barang bukti membahas pemindahan barang bukti secara fisik dan penyerahan barang bukti, yang merupakan tindakan yang membutuhkan lacak balak. Garis ini menjadi buram.

Pertimbangkan kasus akuisisi jaringan langsung. Akuisisi jaringan langsung adalah salah satu di mana penyelidik mungkin perlu terhubung ke jaringan organisasi dan memperoleh sistem dari jarak jauh, sehingga tidak pernah benar-benar menguasai sistem secara fisik. Namun, ada yang perlu diperhatikan, ada "penyerahan" barang bukti yang terjadi, meski dilakukan dari jarak jauh. Dalam hal ini, perolehan dan penanganan barang bukti akan

terjadi secara bersamaan. Namun demikian, meskipun kedua aktivitas ini dapat terjadi secara bersamaan, keduanya tetap merupakan aktivitas terpisah yang layak dipisahkan untuk didiskusikan.



Gambar 10.11 Langkah 3: Tujuan Akuisisi

Dalam contoh akuisisi jaringan, proses penanganan bukti akan menyiratkan bahwa penyelidik forensik dunia maya memperoleh dan melaksanakan lacak balak yang tepat melalui perolehan alamat IP yang akurat untuk terhubung ke jaringan. "Tindakan" akuisisi kemudian akan memerlukan penggunaan perangkat lunak akuisisi jaringan yang sehat secara forensik untuk memperoleh sistem (mengumpulkan bukti digital).

Bukti

Mungkin penting untuk berhenti sejenak dalam diskusi kita tentang proses investigasi untuk membahas berbagai jenis bukti yang mungkin ditemui oleh penyidik.

Menurut Aturan Bukti Federal, ada tiga klasifikasi atau jenis bukti:

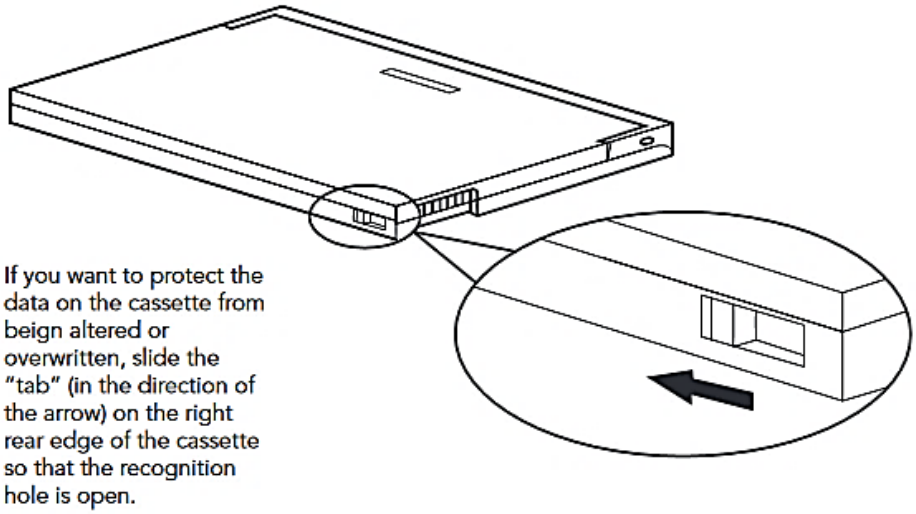
1. Asli. Yang “asli” dari suatu tulisan atau rekaman adalah tulisan atau rekaman itu sendiri atau rekanannya yang dimaksudkan untuk memiliki efek yang sama oleh orang yang membuat atau menerbitkannya. Foto “asli” mencakup negatif atau cetakan apa pun darinya. Jika data disimpan di komputer atau perangkat serupa, semua hasil cetakan atau keluaran lain yang dapat dibaca dengan penglihatan, yang ditunjukkan untuk mencerminkan data secara akurat, adalah "asli".
2. Gandakan. “Duplikat” adalah pasangan yang dihasilkan oleh impresi yang sama seperti aslinya, atau dari matriks yang sama, atau dengan fotografi, termasuk perbesaran dan miniatur, atau dengan perekaman ulang mekanis atau elektronik, atau dengan reproduksi kimiawi, atau oleh teknik setara lainnya yang secara akurat mereproduksi aslinya.
3. Bukti terbaik. Aturan bukti terbaik adalah aturan pembuktian hukum umum yang dapat ditelusuri kembali setidaknya sejauh abad kedelapan belas. Di OMychundv.Barker (1745) 1 Serangan, 21, 49; 26 ER 15, 33, Lord Harwicke menyatakan bahwa tidak ada bukti yang dapat diterima kecuali "yang terbaik yang dimungkinkan oleh sifat kasus ini." Aturan

umum adalah bahwa bukti sekunder, seperti salinan atau faksimili, tidak akan diterima jika ada dokumen asli, dan bukan tidak tersedia karena rusak atau keadaan lain yang menunjukkan tidak tersedianya.

Umumnya, "Bukti Terbaik" menyatakan bahwa bukti adalah yang terbaik dalam keadaan aslinya. Karena itu, merupakan tanggung jawab dan prioritas penyelidik forensik dunia maya untuk menyimpan bukti apa pun yang diterima dalam keadaan aslinya (asli untuk penyelidik). Barang bukti yang diterima penyidik forensik siber bisa berupa salinan barang bukti asli, namun sejauh menyangkut penyidik barang bukti (walaupun berupa salinan dan bukan asli) yang diterima adalah barang bukti yang paling baik. Setiap perubahan atau perubahan pada bukti ini dapat menimbulkan konsekuensi bencana dan dapat mengakibatkan ketidakmampuan untuk menyerahkan bukti apa pun yang dikumpulkan selama penyelidikan ke pengadilan. Keterbatasan atau ketidakmampuan seperti itu dapat mengakibatkan penghentian kasus dan kegagalan untuk mengadili pihak yang bersalah.

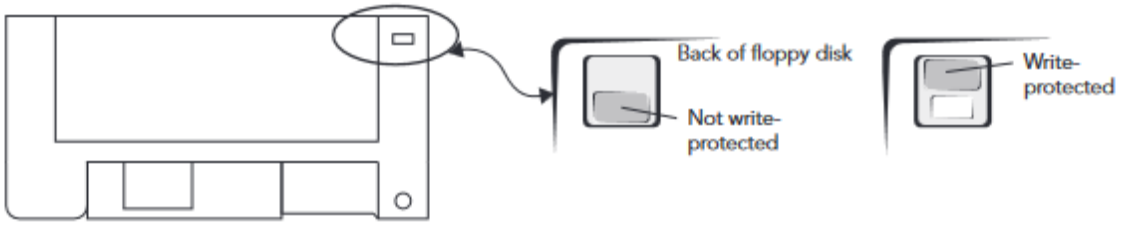
Untuk melestarikan bukti terbaik ini, penting bahwa setiap tindakan yang diambil selama proses investigasi tidak akan melakukan apa pun untuk menghancurkan, memanipulasi, mengubah, atau mengubah bukti tersebut dengan cara apa pun. Pentingnya hal ini tidak dapat dilebih-lebihkan. Sejalan dengan mantra ini, oleh karena itu penting juga untuk bersiap membuktikan, jika diperlukan, integritas pekerjaan di pengadilan. Oleh karena itu, hampir sama pentingnya untuk mendokumentasikan semua langkah individu yang diambil dalam memastikan pelestarian bukti selama proses investigasi.

Intinya, dua komponen utama perolehan bukti pada akhirnya melestarikan bukti dan mendokumentasikan langkah-langkah yang memastikan pelestarian tersebut. Saat memperoleh bukti, sangat penting untuk menggunakan pemblokir tulis. Pemblokir penulisan perangkat keras lebih disukai daripada pemblokir penulisan perangkat lunak, tetapi masing-masing memiliki aplikasi penggunaan yang sesuai. Write blocker, sambil membiarkan data disalin dari perangkat, pada saat yang sama mencegah apa pun untuk ditulis ke perangkat perangkat. Ingat kaset lama? Anda dapat menekan tab atas dan mencegah penimpaan lagu yang ada di kaset. (Lihat Gambar 10.12.)



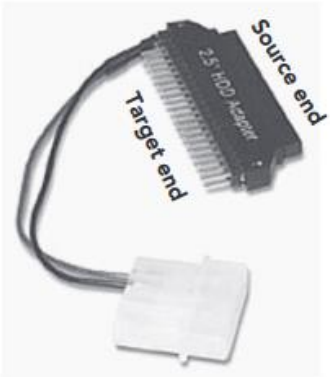
Gambar 10.12 Write Blocker Kaset

Floppy disk memiliki sesuatu yang serupa, tetapi kuncinya dapat dihidupkan atau dimatikan, memungkinkan data diubah pada floppy jika seseorang berubah pikiran. (Lihat Gambar 10.13.) CD dan DVD yang tidak dapat ditulis diblokir.



Gambar 10.13 Write-Protect "Switch" pada Floppy Disk

Semua ini adalah contoh pemblokir penulisan perangkat keras. Di bidang forensik ada banyak jenis pemblokir penulisan lainnya. Inti dari perangkat ini biasanya adalah adaptor, yang di satu ujung dihubungkan ke sumber dan di ujung lainnya dihubungkan ke target. (Lihat Gambar 10.14.)



Gambar 10.14 Adaptor Hard Drive Laptop ke IDE

Sumbernya adalah bukti (dalam contoh kasus kami, hard drive Jose McCarthy) dan targetnya adalah di mana bukti akan disalin (drive yang telah disanitasi). Juga penting untuk mendapatkan salinan atau gambar bit demi bit bila memungkinkan. Penting untuk memahami perbedaan antara gambar bit-for-bit versus hanya salinan. Selain kemungkinan mengubah bukti, sejumlah besar bukti akan dilewati atau dilewatkan hanya dengan menyalin folder dari sistem atau membuat salinan logis. Gambar bit-demi-bit memungkinkan semua data yang ada di dalam drive ditangkap, termasuk ruang yang tidak terisi, ruang kendur, struktur sistem file, dan sebagainya. Bit-for-bit, seperti namanya, menyalin setiap bit, kata demi kata, dan mentransplantasikan setiap bit ke target. Ini bukan salinan logis tetapi salinan fisik, salinan bit demi bit.

Forensik dunia maya tidak selalu dilakukan pada hard drive; sebagaimana disebutkan, ada akuisisi jaringan yang harus dihadapi serta akuisisi ponsel/PDA, dll. Salinan bit demi bit tidak selalu layak atau dalam beberapa kasus memungkinkan. Pada server file korporat misalnya, setiap karyawan dapat dialokasikan ruang untuk menyimpan file kerja; ruang ini terkadang disebut sebagai Direktori HoMe. Ini adalah ruang pribadi pengguna (untuk

penggunaan bisnis) di jaringan perusahaan. Akuisisi server yang berisi ratusan Direktori Rumah karyawan mungkin tidak dijamin.

Akuisisi logis hanya dari Direktori Beranda subjek yang ditentukan mungkin merupakan alternatif terbaik untuk memperoleh bukti ini. Hal yang sama berlaku untuk email perusahaan. Banyak server email perusahaan, seperti Exchange dan Domino (masing-masing digunakan untuk Microsoft Outlook dan Lotus Notes) menggabungkan data mereka ke dalam file besar yang disebut PST dan NSF. Jenis file ini dapat ditemukan dan berada di seluruh sistem email perusahaan. Mungkin tidak perlu mencitrakan semua server atau larik. Pencitraan file tunggal dalam kasus seperti itu mungkin cukup. Mungkin dapat diterima untuk memasang file-file ini (lihat Bab 5 untuk diskusi kita tentang pemasangan file) dan gambar hanya komponen yang berkaitan dengan subjek yang sedang diselidiki.

Hashing

Hash biasanya merupakan fungsi matematika atau algoritme yang mengubah kumpulan data berukuran variabel menjadi kumpulan data acak yang tidak berubah-ubah. Panjang data hash yang dihasilkan bergantung pada tipe hash dan fungsi yang diinginkan. Dalam forensik, keunikan dan kekhasan sangat penting, sehingga panjang hash 128 bit dan 256 bit tidak jarang untuk memastikan keunikan bukti.

Hash biasanya ditampilkan dalam HEX. Setiap karakter heksadesimal dapat mengkodekan 4 bit data biner. Jadi, nilai HEX 64 karakter setara dengan 256 bit biner. Demikian juga, nilai HEX 128 akan mewakili hash 512-bit. Meskipun, saat ini dalam forensik dunia maya, hash 128-bit sudah cukup untuk menetapkan keunikan data.

Keunikan nilai hash penting dalam cyber forensik sehingga tidak ada dua bukti yang identik. Pertimbangkan sidik jari sebagai analogi; setiap orang di planet ini memiliki sidik jari yang unik. Mencocokkan sidik jari di TKP dengan sidik jari individu tertentu tidak akan berpengaruh jika sidik jari tidak unik; Nilai hash 256-bit memungkinkan 2256 nilai unik.

Dalam forensik dunia maya, keunikan nilai hash ini juga akan memastikan integritas bukti. Untuk memastikan integritas bukti, sebelum bukti diperoleh terlebih dahulu di-hash. Bukti kemudian diperoleh dan bukti itu di-hash untuk kedua kalinya. Nilai hash antara bukti pra-akuisisi dan pasca-akuisisi harus identik untuk memastikan integritas bukti.

Setiap perubahan pada bukti setelah mengambil hash asli akan menyebabkan hash berikutnya menjadi sangat berbeda. Recall hash benar-benar acak saat dibuat, sehingga setiap perubahan data terlepas dari ruang lingkup, jumlah, atau pentingnya perubahan akan menghasilkan hash acak yang benar-benar baru. Setiap perubahan dalam data akan sepenuhnya mengubah hasil algoritmik (yaitu, nilai hash).

Saat ini, HASH legal (baca pengadilan dicoba dan diuji) yang paling umum diterima yang digunakan adalah MD5 (hash 128-bit) atau SHA-1 (hash 160-bit). Setelah bukti asli dicitrakan secara forensik dan hash diverifikasi, bukti asli harus disimpan dengan benar mengikuti prosedur penanganan bukti cerdas. Bukti yang diperoleh semula tidak boleh digunakan dalam penyelidikan terperinci dan proses penemuan elektronik. Hanya salinan bukti, setelah diverifikasi dengan hashing, yang boleh digunakan, sedangkan bukti yang diperoleh semula dikunci.

Gambar 10.15 Image MASSter Write Blocker

Ada berbagai alat berbeda yang memungkinkan pencitraan forensik, ada yang gratis dan ada yang cukup mahal. Semua alat forensik biasanya akan menyediakan pencitraan bit-demi-bit dan biasanya akan berisi rutinitas verifikasi hash bawaan. Ada perangkat keras forensik yang dibuat khusus untuk pencitraan forensik, seperti Image MaSSter's Solo. Ada suite perangkat lunak forensik (paket perangkat lunak dengan banyak fungsi), yang memungkinkan pencitraan serta aktivitas forensik terkait lainnya seperti editor HEX, pemasangan drive dan file, pencarian, pengindeksan, parsing, dan banyak tugas lainnya. Contoh suite forensik tersebut termasuk EnCase dari Guidance Software dan FTK Access Data.

Ronelle menerima bukti drive dari departemen Hukumnya. Dia mengikuti prosedur lacak balak yang tepat dan menyimpan drive bukti di lemari besi bukti departemennya. Seiring waktu Ronelle pergi ke lemari besi untuk mengambil bukti, menyelesaikan prosedur lacak balak dengan menandatangani bukti, dan membawa bukti ke mesin forensiknya. Ronelle memperoleh buktinya menggunakan pemblokir tulis EnCase Enterprise dan ImageMasster (lihat Gambar 10.15).

Ronelle membuka file kasus baru di EnCase dan menghubungkan drive bukti ke sistem akuisisi forensiknya melalui pemblokir penulisan. EnCase dapat mengidentifikasi media fisik (yaitu, drive yang sebenarnya dihapus dari mesin kerja Jose) dan memasang drive. Verifikasi hash MD5 dari seluruh disk bukti dilakukan, mengembalikan nilai hash 59a34105247fb3a26e4bc411fea32eb4. Ronelle menggunakan fungsi Encase untuk melakukan akuisisi disk fisik penuh.

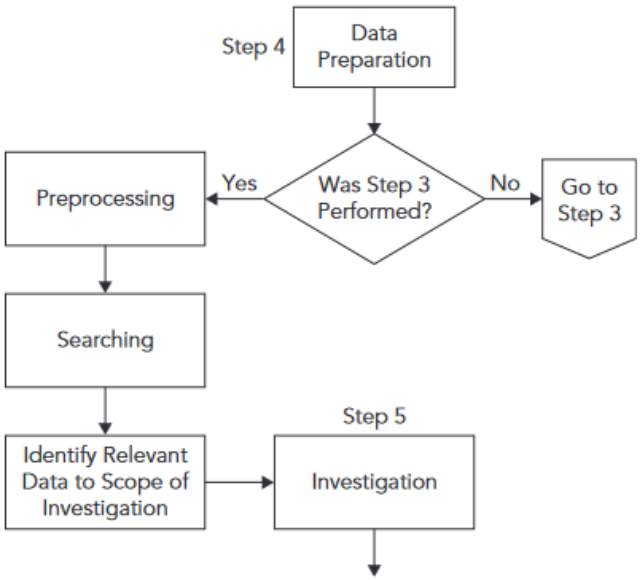
Setelah proses akuisisi selesai dengan sukses, tanpa kesalahan, (menurut rutinitas perangkat lunak EnCase), Ronelle melakukan verifikasi hash lagi dari data yang diperoleh. Nilai hash dari bukti asli dan gambar forensik yang dibuat melalui Encase cocok: 59a34105247fb3a26e 4bc411fea32eb4.

Ronelle kemudian mengembalikan bukti asli ke lemari besi dan sekali lagi mengikuti prosedur lacak balak yang telah ditetapkan dengan memeriksa kembali bukti ke dalam lemari

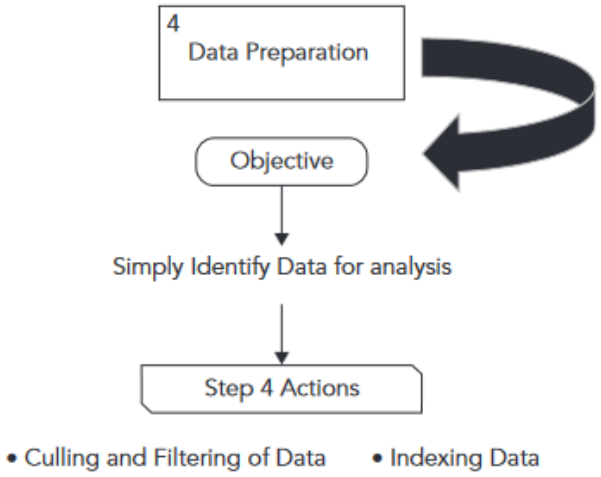
besi dan mencatatnya pada dokumen lacak balak. Ronelle sekarang memiliki salinan persis dari seluruh cakram fisik, yang diperolehnya tanpa mengubah bukti aslinya.

Langkah 4: Persiapan Data

Tujuan: Menyiapkan dan mengidentifikasi data untuk analisis dan investigasi (lihat Gambar 10.16 dan 10.17).



Gambar 10.16 Langkah 4: Persiapan Data



Gambar 10.17 Langkah 4: Tujuan Persiapan Data

Persiapan data dapat dibagi menjadi dua jenis:

1. Preprocessing
2. Pemasangan

Preprocessing

Preprocessing melibatkan langkah-langkah yang menyiapkan bukti sebelum bukti dicari. Pencarian bukti, di sisi lain, membawa hasil, yang kemudian perlu diselidiki.

Pemasangan

Pemasangan sistem pengarsipan (lihat Bab 5) yang terkandung dalam sepotong bukti diperlukan agar struktur data dapat dilihat, sehingga organisasi data dapat dibuat jelas. Tanpa mengumpulkan bukti, tugas sederhana untuk menemukan file bisa menjadi sangat rumit dan membosankan. Bayangkan harus secara manual mengidentifikasi jenis partisi dan kemudian mengukir setiap file secara manual dengan menelusuri setiap byte dari sistem file. Membosankan adalah pernyataan yang meremehkan.

Alat forensik (serta alat lain, yaitu sistem operasi) dapat dengan cepat memasang drive dan sistem filenya, menyajikan data dengan cara yang relatif intuitif. Selain itu, file yang lebih kompleks perlu "dipasang". File ZIP dan format file lainnya memerlukan pemasangan agar dapat dibaca dan karenanya dapat dicari.

Pulihkan File yang Dihapus

Pemulihan item yang dihapus melibatkan pemulihan file dari ruang yang tidak terisi. Biasanya, alat yang memulihkan file ini mencari header dan footer file, sehingga membutuhkan file yang utuh sepenuhnya. Ingat dari Bab 4 bahwa ketika sebuah file dihapus, file tersebut pertama kali dikirim ke folder item yang dihapus. Kemudian, ketika file dihapus dari item yang dihapus, ruangnya menjadi tersedia, atau tidak terlokasi. File tidak dipindahkan atau dihapus secara ajaib; ruangnya hanya tersedia untuk beberapa file lain yang akan ditulis. Hingga ruang ini ditimpa, data yang ada di dalam file masih ada di drive.

Ingat, hanya entri dalam sistem file yang diubah sedemikian rupa sehingga pengambilan oleh sistem operasi tidak dimungkinkan. Namun, entri sistem file bukan merupakan bagian dari file yang sebenarnya. File dapat dihapus dan mengambang di suatu tempat di ruang yang tidak terisi namun masih utuh sepenuhnya. Alat forensik dapat menemukan, mengidentifikasi, dan mengembalikan file-file ini. Tentu saja beberapa metadata yang biasanya terlihat dengan file yang disimpan akan hilang, tetapi file itu sendiri mungkin dalam kondisi murni. File dapat ditimpa sebagian dan tidak dapat dipulihkan secara keseluruhan, namun potongan file tersebut masih dapat ditemukan dan digunakan sebagai bukti.

Memverifikasi

Memverifikasi data memastikan data sesuai dengan klaimnya. Seperti yang telah dibahas, ada proses verifikasi bukti yang dicapai dengan pencocokan hash; tetapi, di sini verifikasi menyiratkan upaya untuk mengungkap penyembunyian bukti yang disengaja. Memverifikasi jenis file sangat penting karena tersangka dapat mengganti nama file dengan cepat dan mudah.

Misalnya, apakah dokumen kata (mis., .docx) benar-benar merupakan dokumen kata, atau berupa gambar? Ingat dari Bab 4 kita membahas file dan bagaimana file didefinisikan dalam header file. Tugas lain yang membosankan bagi manusia adalah menelusuri setiap file yang memeriksa header untuk akurasi. Diperparah dengan keharusan mengekstrak atau mengukir setiap file secara manual, seperti yang dijelaskan sebelumnya, pemeriksaan forensik dunia maya tidak akan pernah berakhir. Fungsi verifikasi file ini adalah fungsi yang terdapat di

sebagian besar paket perangkat lunak forensik dunia maya karena sangat penting untuk pra-pemrosesan bukti.

Verifikasi juga menyiratkan verifikasi jika mungkin partisi pada hard drive telah dihapus. Sekali lagi, ini mungkin upaya penyembunyian. Memeriksa Master Boot Record (MBR) dan Volume Boot Records (VBR) akan mengungkap adanya partisi tambahan, yang mungkin tidak dipasang oleh sistem operasi. Untuk penyegaran pada MBR lihat Bab 5 dan untuk VBR lihat Bab 8. Perangkat lunak forensik mengotomatiskan tugas manual ini, memungkinkan identifikasi lebih cepat dari partisi yang berpotensi hilang.

Pengindeksan

Beberapa alat forensik dapat mengindeks data yang terkandung dalam bukti. Pada dasarnya, pengindeksan mengatur semua data dalam "database", yang memungkinkan hasil pencarian cepat. Sama seperti Google dengan cepat merespons dengan hasil pencarian, demikian juga alat forensik saat data diindeks. Google telah mengindeks banyak Internet (situs web), memungkinkan hasil pencarian cepat. Jika bukan karena pengindeksan ini, Google harus mulai mencari setiap situs setiap kali ada permintaan. Pengindeksan awal, kadang-kadang disebut perayapan, membutuhkan waktu tetapi manfaat setelah diindeks mungkin bermanfaat. Beberapa waktu dikorbankan di bagian depan untuk beberapa pencarian cepat di bagian belakang.

Mencari

Pencarian pada dasarnya melibatkan penyempurnaan bukti asli menjadi subset data yang lebih kecil yang kemudian dapat diselidiki. Pencarian bisa luas artinya untuk mencakup cakupan data yang luas, seperti data antara rentang tanggal atau jenis file tertentu. Ini juga dapat menyiratkan pencarian aktual untuk kriteria yang lebih tepat, seperti kata kunci. Pengindeksan sangat meningkatkan efisiensi pencarian ketika target yang tepat atau tepat, mungkin kata kunci, terlibat.

Pemfilteran dapat membedakan dirinya dari pencarian karena biasanya cakupannya lebih luas, mungkin berlaku untuk data yang jatuh ke dalam subkumpulan. Misalnya, data yang termasuk dalam rentang tanggal tertentu, atau semua data dari jenis file tertentu. Culling adalah proses mengumpulkan data halus yang relevan dengan investigasi. Ini berbeda dari pemfilteran karena pemusnahan biasanya memerlukan upaya eDiscovery. Ini adalah pengumpulan data yang berkaitan dengan individu atau kasus tertentu.

Itu semua masih melibatkan penyempurnaan bukti menjadi bagian yang lebih kecil. Alih-alih mencari semua data dalam suatu organisasi, hanya data yang berkaitan dengan kasus yang perlu dicari. Misalnya, hanya menelusuri profil subjek yang terlibat dalam insiden, bukan semua profil pengguna yang ada di server, atau mungkin menghapus data yang tidak berada dalam cakupan (mis., menghapus file yang dapat dijalankan dari subset).

Pencarian juga berlaku untuk menemukan potongan bukti tersebut di ruang kosong yang tidak lagi menjadi bagian dari file. Sebagian besar file dapat ditimpa, tetapi terkadang teks ASCII, termasuk sisa kalimat atau paragraf, dapat dipulihkan. Bagaimanapun Anda memutuskan untuk mendefinisikan istilah-istilah ini, pada akhirnya, semua tindakan melibatkan pencarian bukti. Ronelle sekarang memiliki citranya di EnCase. Dia telah memverifikasi citranya, jadi dia sekarang memulai langkah pemrosesan bukti dari

penyelidikan. Ronelle telah berhasil memasang gambar dan dapat melihat struktur file yang terdapat di drive. Ini menyiratkan bahwa Master File Table (MFT) belum diubah.

Jose McCarthy, dalam upaya untuk menyembunyikan tindakannya, membuat dua partisi pada hard drive 80 GB miliknya; satu di mana dia gunakan untuk aktivitas sehari-hari dan yang kedua dia gunakan untuk menyimpan data pribadinya. Ketika dihadapkan dengan kemungkinan tuntutan pidana, dia dengan cepat menghapus partisi kedua, sehingga membuatnya "tidak terlihat" oleh sistem operasi.

Gambar 10.18 Label Hard Drive

Dari label hard drive (Gambar 10.18), Ronelle dapat dengan jelas melihat bahwa hard drive adalah 80 GB, tetapi bagaimana dia dapat mengidentifikasi ukuran partisi? Bagaimana dia bisa memperhitungkan semua 80 GB data? Jika partisi kedua tidak teridentifikasi, dia mungkin kehilangan banyak bukti—dalam hal ini, semua bukti penting.

Sebagai penyelidik forensik dunia maya, Ronelle, berdasarkan protokol, memeriksa tabel partisi dan melihat partisi sebesar 60 GB. Titik awal dan akhir dari partisi tunggal itu dengan cepat ditentukan. Tanpa menggunakan alat forensik, jika Ronelle tidak memiliki pengetahuan tentang spesifikasi dan fungsi Tabel Partisi, bagaimana penyelidik kami dapat menjelaskan 20 GB yang hilang? Mengidentifikasi akhir dari partisi pertama akan membantu Ronelle dalam menemukan titik awal dari partisi yang dihapus. Tinjau Bab 7 untuk penyegaran dalam mengidentifikasi partisi.

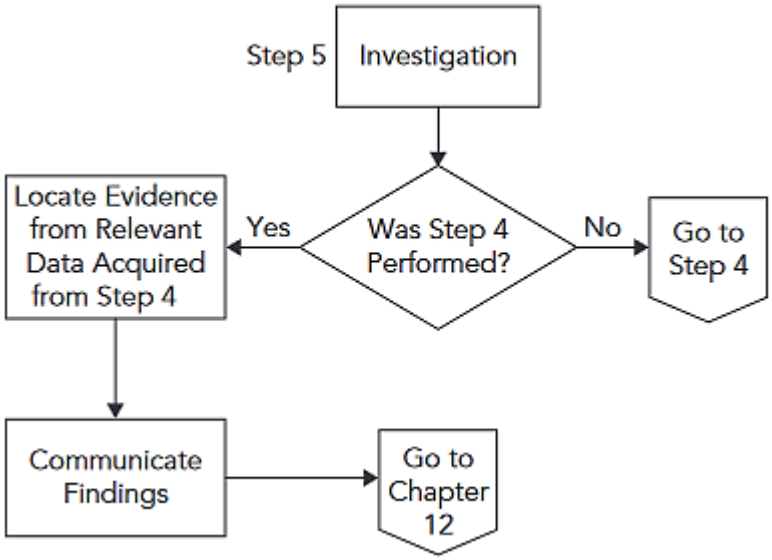
Akhirnya, partisi yang dihapus dipulihkan dan semua data yang terkait dengan partisi kedua diambil, diakses, dan dianalisis. Ronelle juga telah memulihkan semua file yang terhapus. Dia memverifikasi semua jenis file dan kemudian mengindeks semua konten drive. Dalam kasus pencurian IP kami, Ronelle Sawyer sedang menyelidiki apakah Jose McCarthy berpotensi terlibat dalam distribusi kekayaan intelektual organisasinya secara tidak sah kepada pesaing, Janice Witcome, Direktur Pelaksana Perusahaan XYZ.

Ronelle dihadapkan dengan memeriksa jutaan keping data bukti potensial yang berada di hard drive Jose, mencari pepatah jarum di tumpukan jerami. Ronelle telah memfilter semua data dari hard drive Jose untuk mengungkapkan hanya data yang telah dibuat atau diubah dalam ruang lingkup penyelidikan, dan rentang tanggal (10 Mei 2008–26 Maret 2009) penyelidikan. Dari data yang tersisa, Ronelle kemudian mencari data tersebut untuk kata kunci tertentu yang terkait dengan kasus yang sedang diselidiki, yaitu, “Janice”, “Whitcome”, dan “XYZ”.

Langkah 5: Investigasi

Tujuan: Menemukan data yang sesuai dengan kriteria pencarian.

Investigasi adalah langkah dalam proses yang menemukan data yang cocok dengan kriteria yang diidentifikasi dalam permintaan investigasi. Bagian inilah yang akhirnya menemukan senjata merokok pepatah. Tentu, lacak balak yang dilakukan secara tidak benar dapat menghancurkan sebuah kasus, tetapi tanpa bukti ini mungkin tidak akan ada kasus (lihat Gambar 10.19 dan 10.20).

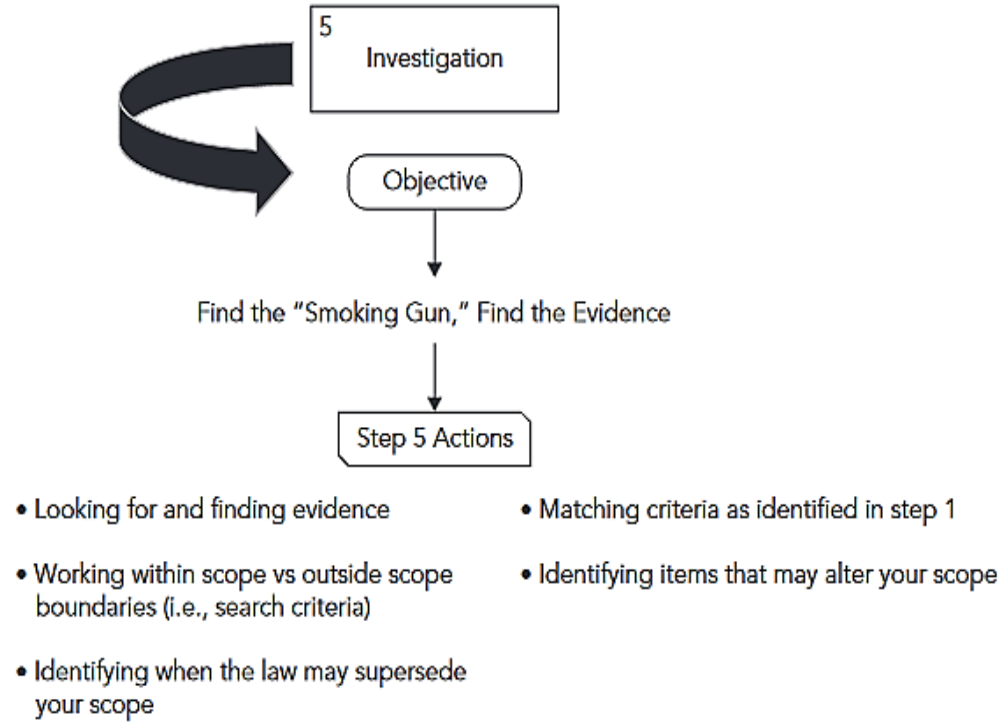


Gambar 10.19 Langkah 5: Investigasi

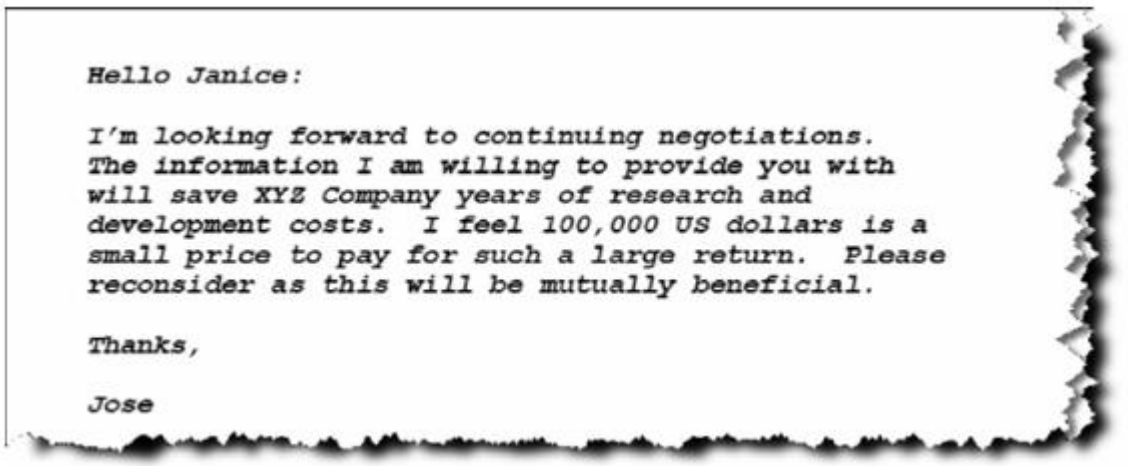
Investigasi adalah kasus khusus dan pada akhirnya ada jalan tak terbatas dari mana kasus dapat berkembang. Setiap kasus berbeda dan mungkin tidak perlu mengungkap semua bukti dalam semua kasus yang ada di meja penyidik; namun, dalam beberapa kasus, setiap batu mungkin perlu dibalik.

Pemfilteran dan pencarian Ronelle telah mengembalikan dua dokumen: "Systemm32.dll" (lihat Gambar 10.21) dan dokumen kedua bernama "bukti yang

memberatkan.doc." Ingat dari Bab 4 bahwa Ronelle mengidentifikasi file bernama "Systemm32.dll" (lihat Gambar 4.10, 4.12, dan 4.13), yang sebenarnya merupakan dokumen memberatkan yang diganti namanya oleh Jose dalam upaya untuk menyembunyikan aktivitasnya.



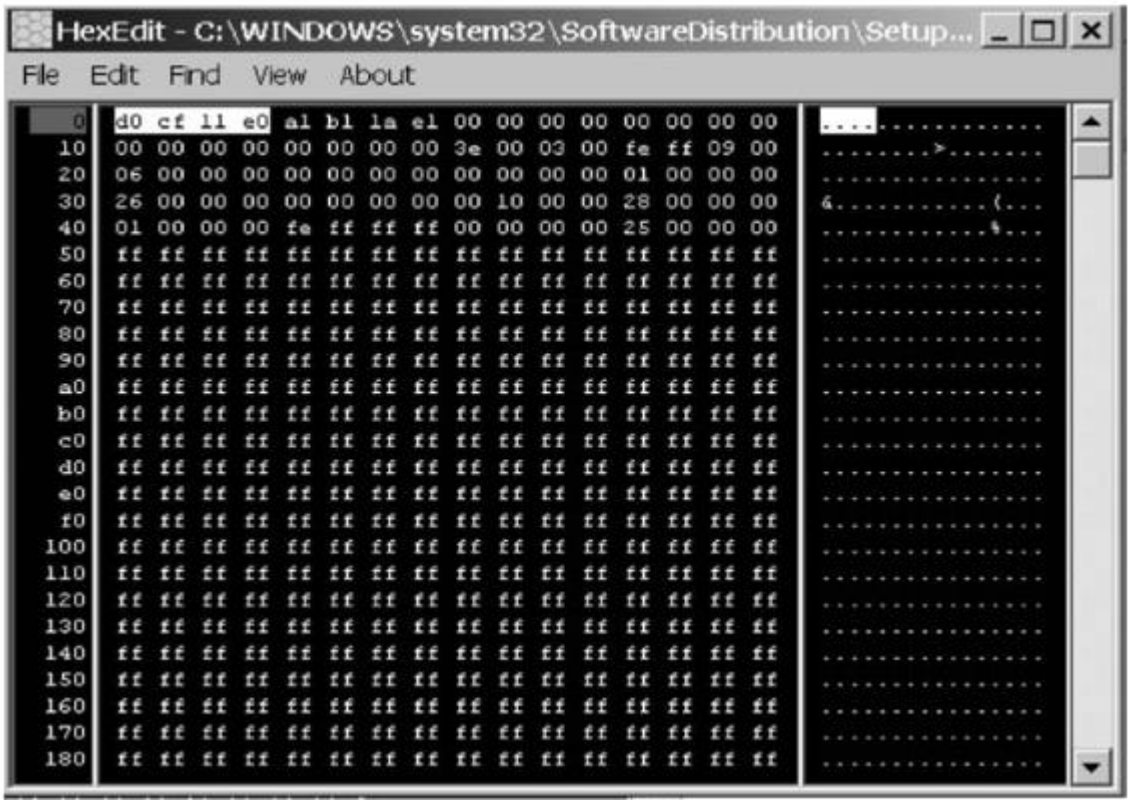
Gambar 10.20 Langkah 5: Tujuan Investigasi



Gambar 10.21 Systemm32.dll dan bukti yang memberatkan.doc

Dokumen kedua bernama "bukti yang memberatkan.doc" terletak di partisi kedua 20 GB yang dihapus. Nama dokumen, "bukti yang memberatkan," dibuat semata-mata untuk tujuan kasus berbasis teks ini, karena menamai dokumen dengan cara yang begitu jelas akan menyederhanakan proses forensik komputer dan juga menjadi tidak logis.

Dokumen ini tampaknya berisi bukti yang dicari oleh departemen Hukum Ronelle. Mari kita periksa surat dari Jose McCarthy melalui editor HEX, ditunjukkan pada Gambar 10.22.



Gambar 10.22 Surat dari Jose McCarthy Ditampilkan melalui Editor HEX

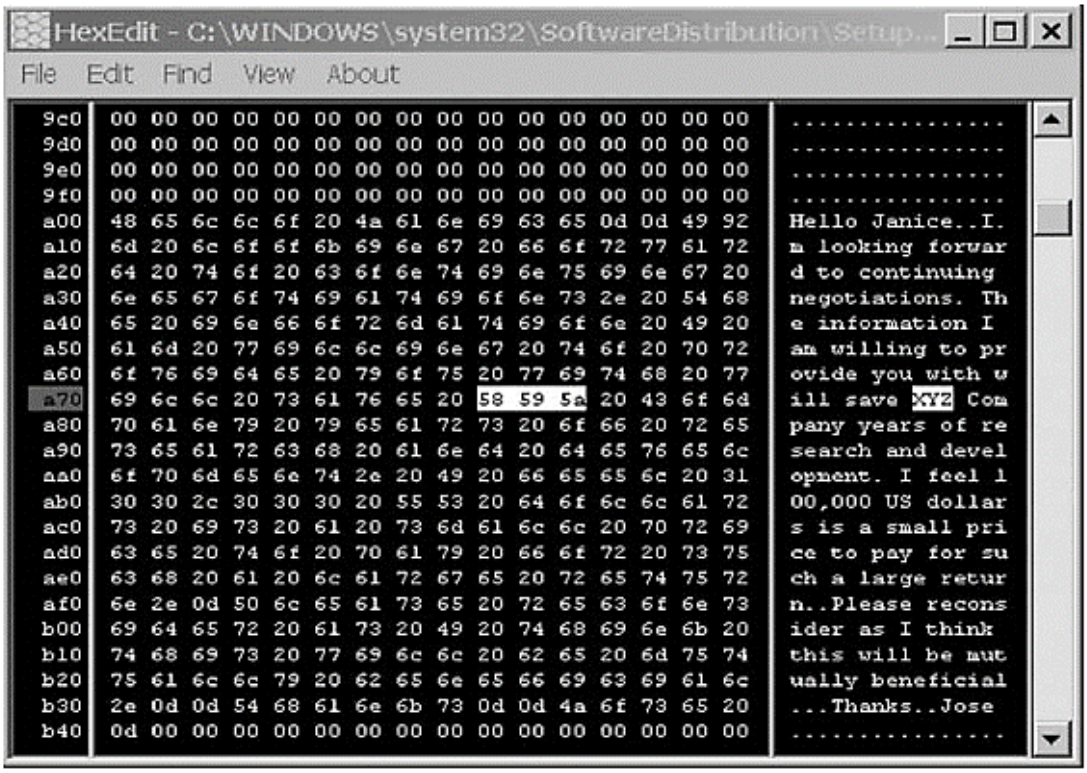
Kita dapat dengan mudah melihat tanda tangan file dokumen, d0 cf 11 e0 (Gambar 10.22), yang mengidentifikasi dokumen ini sebagai dokumen Word. Jadi meskipun dokumen ini akan diganti namanya dengan ekstensi .dll, meninjau melalui editor HEX akan dengan mudah mengidentifikasi dokumen ini sebagai dokumen Word.

Dua dari kata kunci Ronelle ("XYZ" [diidentifikasi oleh HEX 58595a] dan "Janice") yang diidentifikasi dalam dokumen yang diambil dari hard drive Jose McCarthy (lihat Gambar 10.23). Ini menunjukkan pentingnya menyiapkan kata kunci sedemikian rupa yang akan menjerat sebanyak mungkin hits nyata. Misalnya, kata kunci "Janice Witcome" tidak akan menghasilkan "hit" pada dokumen yang baru saja ditinjau. Memahami perilaku manusia merupakan keterampilan penting saat melakukan penyelidikan; seperti dalam kasus Ronelle, orang biasanya berinteraksi berdasarkan nama depan.

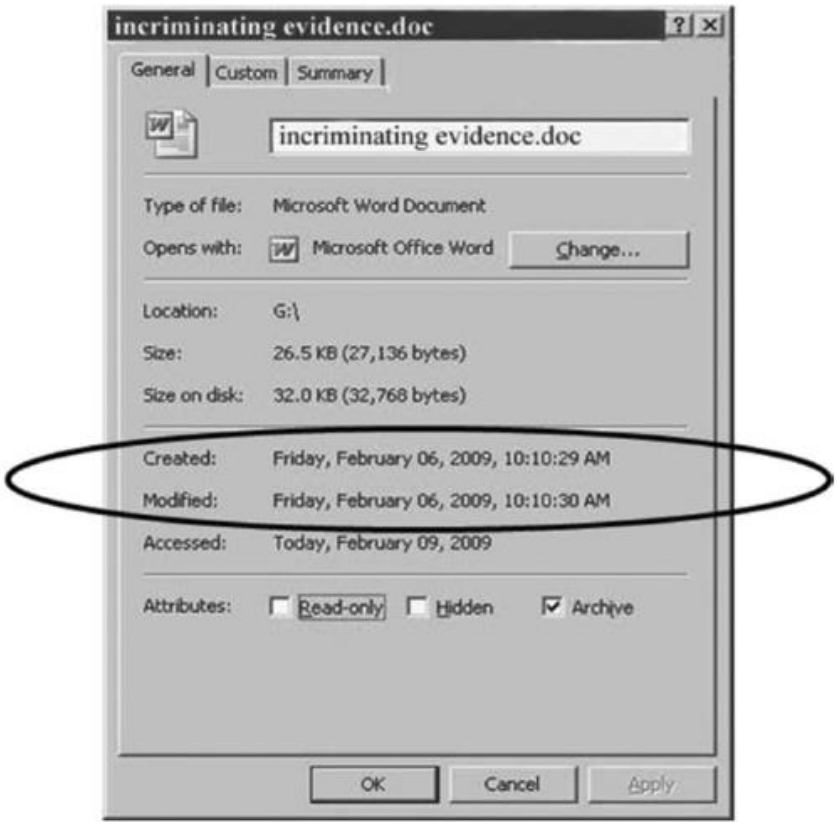
Ronelle membuat penemuan yang mengejutkan ketika memeriksa dokumen ini. Dia melihat beberapa inkonsistensi dengan perubahan tanggal/waktu dokumen (lihat Gambar 10.24). Waktu pembuatan "bukti yang memberatkan.doc", seperti yang terlihat pada properti dokumen dan dilihat melalui MS Word, tidak sesuai dengan waktu pembuatan dokumen jika dilihat melalui metadata sistem pengarsipan. Properti dokumen MS Word dapat dilihat dengan mengklik kanan pada dokumen dan memilih properti.

Metadata "waktu pembuatan" yang diambil dari sistem file menunjukkan waktu pembuatan 10:10:28. Alat forensik yang digunakan Ronelle menampilkan data ini. Dia tidak yakin bagaimana itu diturunkan dan apakah itu akurat. Ronelle mampu mengekstrak data dari entri direktori sistem file seperti yang terlihat pada Gambar 10.25.

Ronelle memahami bahwa entri tersebut bukanlah entri NTFS tetapi entri FAT, dan tidak yakin mengapa ada perbedaan.



Gambar 10.23 Identifikasi Istilah Kunci Kata Pencarian



Gambar 10.24 Inkonsistensi dengan Tanggal/Waktu Modifikasi Dokumen

Byte Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ASCII	I	n	c	r	i	m	~	1	D	O	C		•	•	N	Q	F	:	I	:	•	•	O	Q	F	:	•	•	•	j	•	•
Hex	41	53	49	4D	50	4C	7E	31	44	4F	43	20	00	8C	4E	51	46	3A	49	3A	00	00	4F	51	46	3A	02	00	00	6A	00	00

Gambar 10.25 Ekstrak Data dari Entri Direktori Sistem File

10.3 WAKTU

Waktu adalah konsep penting dalam forensik karena lebih dari satu alasan. Seperti kebanyakan topik dalam forensik, seluruh buku dapat ditulis hanya dengan satu konsep waktu dan relevansi waktu dalam penyelidikan secara keseluruhan. Melangkah mundur dari investigasi Ronelle, waktu dan peran waktu dalam investigasi cyber forensik akan dibahas di Bab 11. Masalah perbedaan waktu yang dibahas akan diperiksa lebih lanjut untuk menentukan mengapa atau bagaimana hal ini bisa terjadi, dengan fokus pada menjawab pertanyaan, "apakah perbedaan seperseratus detik penting atau relevan dalam penyelidikan forensik?" Memahami mengapa perbedaan ini ada bisa menjadi perbedaan antara berhasil mengajukan kasus versus terlihat tidak kompeten selama deposisi, atau lebih buruk lagi, sebagai ahli yang bersaksi di pengadilan.

10.4 RINGKASAN

Di Bab 10, kami memperkenalkan Investigative Smart Practices dalam format umum agar sesuai dengan sebagian besar jenis organisasi forensik dan sebagian besar jenis kasus. Instruksi baris demi baris yang tepat untuk menjalankan investigasi forensik dunia maya yang lengkap secara logis tidak mungkin untuk disajikan, karena setiap organisasi yang melakukan penyelidikan forensik dunia maya akan memiliki pendekatan, prosedur, kebijakan, dan metode mereka sendiri, beberapa ditentukan oleh undang-undang, orang lain dengan preferensi internal dan protokol.

Investigasi yang dilakukan oleh penegak hukum, misalnya, terhadap pornografi anak versus investigasi internal perusahaan terhadap pencurian kekayaan intelektual pada akhirnya dapat menemukan bukti yang diperlukan untuk mengadili yang bersalah; namun, pendekatan, langkah-langkah yang diambil, dan proses untuk mencapai tujuan tersebut mungkin sama sekali berbeda dan didukung oleh protokol dan dokumentasi yang sama sekali berbeda. Adalah bijaksana untuk dicatat bahwa sama anehnya dengan perbedaan antar organisasi, begitu pula perbedaan antar kasus.

Praktik Cerdas Investigasi berikut dimaksudkan untuk cakupan yang luas dan digunakan sebagai pedoman:

- Langkah 1: Kontak/Permintaan Awal. Validitas dan ruang lingkup permintaan investigasi ditetapkan. Fungsi ini dapat dilakukan oleh seseorang di luar bidang forensik dunia maya. Misalnya, hal ini dapat ditentukan oleh hakim melalui perintah pengadilan atau mungkin melalui departemen SDM dalam sebuah organisasi besar.
- Langkah 2: Penanganan Barang Bukti. Integritas bukti dipertahankan sepanjang keseluruhan kasus. Proses ini terjadi setiap kali bukti ditangani. Mempertahankan

integritas bukti sangat penting, seperti halnya mampu membuktikan integritas bukti di pengadilan.

- Langkah 3: Perolehan Bukti. Langkah ini melibatkan perolehan citra forensik dari bukti asli. Perolehan bukti pasti termasuk dalam Langkah 2, Penanganan Bukti; namun, langkah ini lebih berfokus pada perolehan bukti versus penanganan bukti selama perolehan.
- Langkah 4: Persiapan Data. Mempersiapkan dan mengidentifikasi data untuk analisis dan investigasi. Ini adalah "menganalisis" semua data untuk memastikan pencarian yang valid dan lengkap. Ini termasuk memasang file kompleks, memverifikasi jenis file, memulihkan item yang dihapus, dan hal lain yang akan menyiapkan data untuk penyelidikan akhir (Langkah 5).
- Langkah 5: Investigasi. Berfokus untuk menemukan data yang cocok dengan kriteria pencarian yang ditentukan. Langkah ini cenderung sedikit lebih subyektif daripada yang lain, karena penyelidik mungkin perlu memeriksa hasil pencarian, membuang positif palsu, dan mengidentifikasi bagian penting dari bukti, yang biasanya tidak mudah disebut "bukti yang memberatkan". dokter."

Seperti halnya nilai hash bukti, setiap kasus unik. Keunikan inilah yang membuat forensik dunia maya menjadi bidang yang menantang.

BAB 11
WAKTU DAN FORENSIK

Satu-satunya alasan waktu adalah agar semuanya tidak terjadi sekaligus.

-Albert Einstein

11.1 APA ITU WAKTU?

Tergantung di mana Anda mencari jawabannya:

Astronomi: Dimensi yang membedakan masa lalu, sekarang, dan masa depan. Dalam relativitas, waktu digambarkan sebagai dimensi geometris, analog dengan dimensi ruang.[1]

Fisika: Durasi pengukuran kuantitas, biasanya mengacu pada proses periodik seperti rotasi Bumi atau getaran radiasi elektromagnetik yang dipancarkan dari atom tertentu.

Mekanika klasik: Waktu adalah mutlak dalam arti bahwa waktu suatu peristiwa tidak bergantung pada pengamat.

Filsafat: Waktu adalah ukuran jam. Kami menggunakan waktu untuk menempatkan acara secara berurutan satu demi satu, dan kami menggunakan waktu untuk membandingkan berapa lama acara berlangsung.[2]

Teknologi informasi: Stempel waktu adalah urutan karakter, yang menunjukkan tanggal dan/atau waktu terjadinya peristiwa tertentu. Stempel waktu adalah waktu di mana suatu peristiwa direkam oleh komputer, bukan waktu peristiwa itu sendiri.

Ilmu komputer: Waktu Unix, yang menjelaskan titik waktu, didefinisikan sebagai jumlah detik yang berlalu sejak tengah malam Proleptic Coordinated Universal Time (UTC) tanggal 1 Januari 1970, tidak termasuk detik kabisat.

Menurut teori relativitas, itu tergantung pada kerangka acuan pengamat. Waktu dianggap sebagai koordinat keempat yang diperlukan, bersama dengan tiga koordinat spasial, untuk menentukan suatu peristiwa.

Waktu adalah kuantitas satu dimensi yang digunakan untuk mengurutkan peristiwa, dan untuk mengukur durasi peristiwa dan interval di antara peristiwa tersebut. Waktu dihitung dalam istilah komparatif (seperti lebih lama, lebih pendek, lebih cepat, lebih cepat, dan lebih lambat) atau dalam istilah numerik menggunakan satuan (seperti detik, menit, jam, dan hari).

Rekonstruksi peristiwa sebagai bagian dari forensik dunia maya dan respons insiden dapat melibatkan peristiwa hanya dari satu sistem, serta peristiwa yang diperoleh dari berbagai sumber yang terpisah secara geografis, masing-masing dengan jamnya sendiri. Teknik yang sangat berguna untuk rekonstruksi peristiwa adalah "garis waktu". Di sini, peristiwa diskrit yang memiliki stempel waktu yang terkait dengannya diurutkan ke dalam garis waktu. Stempel waktu dapat diperoleh dari metadata sistem file, log sistem, atau data aplikasi. Bergantung pada sumber peristiwa, ini dapat memberikan urutan peristiwa yang

terperinci yang terjadi pada suatu sistem (atau banyak peristiwa), yang memungkinkan penyelidik untuk merekonstruksi urutan peristiwa yang terjadi.

Saat mempertimbangkan stempel waktu yang direkam oleh sistem komputasi sebagai bukti dalam investigasi, beberapa faktor perlu dipertimbangkan. Waktu pada sistem komputasi disimpan oleh jam perangkat keras sistem dan dalam beberapa kasus oleh jam sistem perangkat lunak tambahan. Bergantung pada keakuratan jam ini, bagaimana jam tersebut diinisialisasi, dan apakah jam tersebut disinkronkan, jam dapat sangat berbeda dari waktu "sebenarnya". Selain itu, jam dapat disalahartikan berada di zona waktu yang salah atau disetel ke waktu yang salah, jam dapat dimanipulasi secara sewenang-wenang, dan dapat berjalan cepat atau lambat (skew jam).

Waktu biasanya hanya kemajuan. Jika Anda memiliki program komunikasi yang berjalan di komputer yang berbeda, waktu masih harus dimajukan, bahkan jika Anda berpindah dari satu komputer ke komputer lain. Jelas jika satu sistem di depan yang lain, yang lain berada di belakang yang satu itu. Dari perspektif pengamat eksternal, beralih di antara sistem ini akan menyebabkan waktu melompat maju dan mundur, efek yang tidak diinginkan.

Sebagai konsekuensinya, jaringan yang terisolasi dapat menjalankan waktu yang salah, tetapi segera setelah Anda terhubung ke Internet, efeknya akan terlihat. Bayangkan saja beberapa pesan email tiba lima menit sebelum dikirim, dan ada balasan dua menit sebelum pesan dikirim.

11.2 PROTOKOL WAKTU JARINGAN

Network Time Protocol (NTP) adalah protokol Internet yang digunakan untuk menyinkronkan jam komputer ke beberapa referensi waktu.

NTP menyediakan mekanisme untuk menyinkronkan waktu dan mengoordinasikan distribusi waktu dalam Internet yang besar dan beragam yang beroperasi dengan kecepatan dari gelombang biasa hingga gelombang cahaya. Ini menggunakan desain waktu yang dapat dikembalikan di mana subnet terdistribusi dari server waktu, yang beroperasi dalam konfigurasi master-slave hierarkis yang mengatur diri sendiri, menyinkronkan jam lokal di dalam subnet dan dengan standar waktu nasional melalui kabel atau radio.

NTP dirancang untuk menghasilkan tiga produk: clock offset, roundtrip delay, dan dispersion, yang semuanya relatif terhadap clock referensi yang dipilih. Offset jam mewakili jumlah yang diperlukan untuk menyesuaikan jam lokal agar sesuai dengan jam referensi. Penundaan pulang-pergi memberikan kemampuan untuk meluncurkan pesan agar tiba di jam referensi pada waktu yang ditentukan. Dispersi mewakili kesalahan maksimum jam lokal relatif terhadap jam referensi. Karena sebagian besar server waktu host akan disinkronkan melalui server waktu peer lain, ada dua komponen di masing-masing dari ketiga produk ini, yang ditentukan oleh peer relatif terhadap sumber referensi utama waktu standar dan yang diukur oleh host relatif terhadap peer.

Masing-masing komponen ini dikelola secara terpisah dalam protokol untuk memfasilitasi kontrol kesalahan dan pengelolaan subnet itu sendiri. Mereka tidak hanya menyediakan pengukuran offset dan delay yang presisi, tetapi juga batas kesalahan maksimum yang pasti, sehingga antarmuka pengguna tidak hanya dapat menentukan waktu,

tetapi juga kualitas waktu. Keakuratan yang dapat dicapai oleh NTP sangat bergantung pada ketepatan jam perangkat keras lokal dan kontrol perangkat yang ketat serta latensi proses.

Seperti atribut file lainnya, stempel tanggal dan stempel waktu dicatat dalam sistem file selain file. Seperti disebutkan, stempel tanggal tingkat sistem file tidak mengganggu stempel tanggal yang disimpan dalam file itu sendiri.

11.3 DATA STEMPEL WAKTU

Data timestamp yang terdapat di dalam sistem file adalah metadata (data tentang data). Seperti yang telah kita bahas di Bab 9, data ini disimpan dalam catatan sistem file, yang dalam Sistem Operasi Windows akan disimpan dalam Entri Direktori jika sistem pengarsipan adalah FAT atau disimpan dalam MFT jika sistem pengarsipan adalah NTFS. Data stempel waktu juga dapat ditemukan di lokasi lain seperti log sistem dan data aplikasi, misalnya. Stempel waktu ditentukan dengan menyalin nilai jam lokal saat ini ke stempel waktu ketika beberapa peristiwa penting, seperti kedatangan pesan, terjadi. Untuk mempertahankan akurasi tertinggi, penting agar hal ini dilakukan sedekat mungkin dengan driver perangkat keras atau perangkat lunak yang terkait dengan peristiwa tersebut.

Berbagai program memasukkan catatan tanggal dan waktu mereka sendiri ke dalam konten file atau meta-tag, seperti produk Microsoft Office dan perangkat lunak lainnya. Properti file tanggal dan waktu tingkat sistem file tidak mengganggu stempel waktu dan tanggal yang disimpan dalam file itu sendiri. Stempel waktu yang terdapat dalam sistem dapat diperkenalkan dengan berbagai cara, misalnya dari header email atau data cookie HTTP. Jadi, mungkin ada data stempel waktu yang ditemukan dalam bukti yang diperkenalkan dari sistem eksternal lain.

Varietas Stempel Waktu

- Tanggal dan waktu pembuatan adalah atribut file atau folder yang menentukan tanggal dan waktu pembuatan asli file atau folder tersebut. Ketika file atau folder dibuat, atau pertama kali disimpan ke suatu lokasi, tanggal dan waktu pembuatan dicatat dalam sistem file. Biasanya, atribut tanggal dan waktu ini tidak pernah diubah setelahnya karena tujuan utamanya adalah untuk menunjukkan tanggal dan waktu pembuatan file asli.
- Tanggal dan waktu terakhir diubah adalah atribut file atau folder yang menunjukkan, pada tingkat sistem file, kapan file atau folder diubah terakhir kali. Saat konten file dimodifikasi dan disimpan, tanggal dan stempel waktu modifikasi terakhir diatur agar sesuai dengan waktu saat perubahan dilakukan. Tujuan dari atribut ini adalah untuk menunjukkan kapan perubahan terakhir pada file atau folder dilakukan. Pekerjaan apa pun dengan dokumen, gambar, dan file lain secara otomatis mengubah (memodifikasi) stempel waktu file. Namun, dalam beberapa kasus mungkin penting untuk mengembalikan tanggal dan waktu modifikasi terakhir file kembali ke tanggal dan waktu tertentu (mis., untuk menyortir, merekam, mencatat, atau tujuan lain).
- Atribut tanggal akses (juga disebut tanggal akses terakhir) menunjukkan pada tingkat sistem file saat file diakses terakhir kali. Atribut tanggal akses terakhir berbeda dari tanggal modifikasi dan properti file. Meskipun tanggal dan waktu modifikasi menunjukkan kapan perubahan terakhir dilakukan pada konten file, tanggal akses terakhir tidak selalu

mengandaikan perubahan konten. Saat file dimodifikasi, modifikasi dan tanggal akses terakhir diatur ke nilai yang sama. Namun, jika file hanya dibuka untuk dilihat, disalin, dipindahkan, atau—dengan kata lain—jika file atau folder digunakan dengan cara apa pun, tanggal akses terakhir diubah menjadi tanggal saat ini.

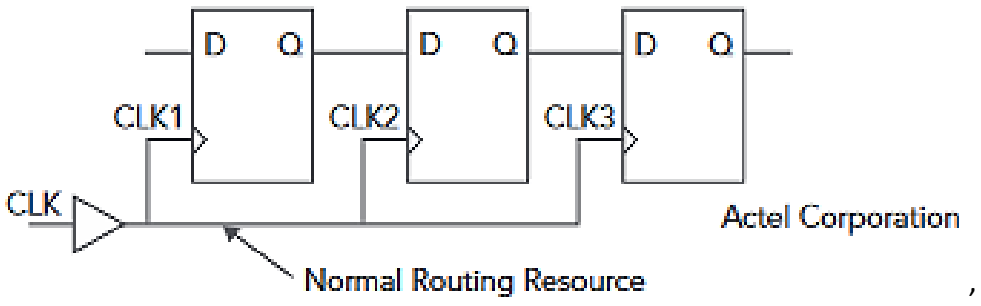
Setiap melihat, menyalin, atau memindahkan dokumen, gambar, dan file lainnya secara otomatis mengubah stempel tanggal terakhir diakses dari file tersebut. Kadang-kadang, waktu akses terakhir diubah karena pengoperasian beberapa program dengan file atau folder. Itu mungkin perangkat lunak antivirus, yang mengakses file untuk tujuan pemindaian virus atau Windows Explorer, yang mengekstrak ikon dari file yang dapat dieksekusi sehingga mengubah tanggal akses terakhir. Resolusi file terakhir diakses tanggal satu jam untuk NTFS dan satu hari untuk FAT. Ini berarti bahwa hanya jam atau hari akses yang akan direkam untuk properti file ini, dengan melewatkan menit atau jam akses yang sesuai. Untuk kinerja sistem file yang lebih baik, pembaruan waktu akses terakhir dinonaktifkan di Windows Vista secara default. Dengan demikian, atribut waktu akses terakhir ditetapkan saat membuat file dan tidak diubah sesudahnya, bahkan jika file tersebut dimodifikasi. Namun, dimungkinkan untuk mengaktifkan pembaruan waktu akses terakhir jika diperlukan.

11.4 MELACAK WAKTU

Dalam investigasi forensik digital, mengetahui waktu yang tepat ketika suatu peristiwa terjadi dapat menjadi sangat penting. Ini mungkin karena kebutuhan untuk mengkorelasikan peristiwa dari sistem yang berbeda, menetapkan alibi, atau mencari tahu kapan peristiwa terjadi sehubungan dengan peristiwa di dunia nyata.

—Brett Tjaden

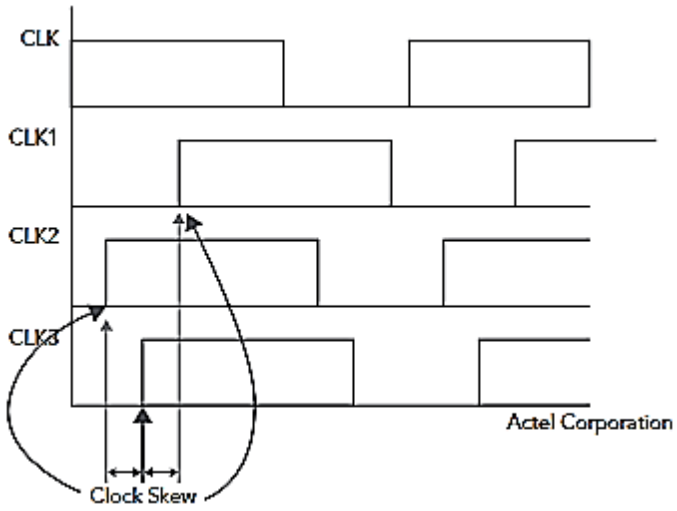
Ini adalah prinsip desain mendasar bahwa pengaturan waktu harus memenuhi pengaturan register dan persyaratan waktu tunggu. Penundaan propagasi data dan kemiringan jam adalah bagian dari perhitungan ini. Mencatat register yang berdekatan secara berurutan pada tepi yang sama dari jam miring tinggi berpotensi menyebabkan pelanggaran waktu atau bahkan kegagalan fungsional. Gambar 11.1 menunjukkan contoh berdekatan secara berurutan register, di mana sumber perutean lokal telah digunakan untuk merutekan sinyal clock. Dalam situasi ini, kemungkinan besar terjadi kemiringan jam yang nyata.



Gambar 11.1 Register Berdekatan Berurutan dengan Clock Skew

Pada Gambar 11.1, semua register memiliki clock pada edge yang sama, tetapi waktu kedatangan edge berbeda pada setiap register. Gambar 11.2 menunjukkan contoh kemiringan jam untuk rangkaian yang ditunjukkan pada Gambar 11.1.

Tujuan dalam rekonstruksi peristiwa forensik dunia maya adalah mengurutkan semua peristiwa yang penting sehingga hubungan sebab-waktu dapat ditetapkan untuk penyelidikan. Jika semua peristiwa yang perlu kita rekonstruksi diberi stempel waktu oleh jam sistem yang sama, beberapa faktor yang dibahas di atas mungkin tidak menjadi masalah karena waktunya akan berbeda dari waktu "nyata" secara konsisten dan urutan peristiwa dapat ditetapkan.



Gambar 11.2 Fluktuasi Waktu Kedatangan Jam Pada Rangkaian Gambar 11.1

Namun, kemungkinan manipulasi jam merupakan faktor yang selalu perlu dipertimbangkan. Secara khusus, ketika jam diset kembali dalam waktu, waktu yang identik dapat direkam untuk peristiwa yang terjadi sebelum dan sesudah perubahan jam. Selain itu, beberapa sistem komputasi benar-benar terisolasi, dan stempel waktu eksternal diperkenalkan dalam banyak cara.

Misalnya, informasi dari header email atau data cookie HTTP mungkin memainkan peran penting dalam penyelidikan, di mana stempel waktu eksternal dan internal perlu dikorelasikan. Juga, setiap kali bukti digital digunakan untuk menetapkan atau mendukung titik waktu ketika peristiwa di dunia fisik terjadi, stempel waktu dari sistem yang terlibat perlu diterjemahkan ke waktu referensi ini.

Server web umumnya lebih menarik untuk penyelidikan forensik, karena stempel waktu dari server web sering dapat ditemukan di mesin klien, dan HTTP adalah protokol yang digunakan secara luas dalam banyak aspek komputasi saat ini.

—Brett Tjaden

11.5 MODEL JAM DAN TIME BOUNDING

Ketika penyelidik forensik dunia maya perlu menentukan jam berapa jam komputer disetel pada waktu tertentu di UTC, atau mencari tahu jam berapa stempel waktu yang

direkam pada sistem benar-benar terjadi, dia perlu menggunakan model jam atau waktu. - teknik mengikat.

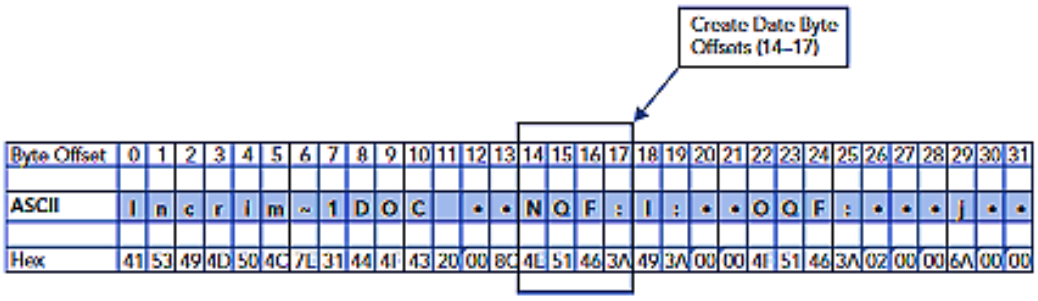
Model jam yang diperkenalkan oleh Malcolm Stevens (Unification of Relative Time Frames for Digital Forensics) menggunakan referensi waktu (biasanya UTC) sebagai dasar, dan kemudian penyelidik harus menentukan offset waktu referensi untuk setiap jam yang diinginkannya. menggambarkan. Offset ini bisa bersifat dinamis, meskipun Stevens tidak menguraikan bagaimana mendefinisikan offset yang mengubah nilainya dari waktu ke waktu. Model ini memiliki kekuatan dalam menunjukkan nilai apa yang dimiliki jam komputer dari waktu ke waktu, tetapi untuk memetakan stempel waktu host kembali ke waktu referensinya, invers dari semua offset perlu dihitung dan diterapkan ke stempel waktu. Selain itu, Stevens mengabaikan fakta bahwa nilai stempel waktu yang ditemukan pada host dapat dipetakan kembali ke host beberapa kali dalam waktu referensi, yang merupakan hasil dari jam host yang diatur kembali ke waktu.

Teknik time-bounding mencoba untuk menetapkan urutan kausal antara peristiwa individu dan dengan demikian urutan keseluruhan dari semua peristiwa pada sistem. Jika beberapa dari peristiwa tersebut dapat dikaitkan dengan suatu waktu, dimungkinkan untuk menentukan rentang waktu ketika peristiwa lain harus terjadi.

Teknik time-bounding kemungkinan besar akan digunakan bersamaan dengan model jam: model akan memberikan beberapa cap waktu yang terkait dengan peristiwa sementara teknik time-bounding dapat digunakan untuk memverifikasi atau menyangkal bahwa deskripsi jam yang diturunkan untuk suatu komputer konsisten dengan bukti yang ditemukan selama penyelidikan forensik.

11.6 MS-DOS 32-BIT TIMESTAMP: TANGGAL DAN WAKTU

Waktu Windows 32-bit adalah format waktu yang digunakan oleh Microsoft dalam sistem pengarsipan FAT untuk stempel waktu. Kami melihatnya di entri direktori, khususnya pada waktu pembuatan (byte offset 14–17) dan waktu penulisan terakhir (byte offset 22–25). Kami dengan jelas melihat pada Gambar 11.3 bahwa presentasi ASCII dari offset byte tanggal (14-17) [NQF:] tidak disimpan dengan cara yang seharusnya disajikan.



Gambar 11.3 Representasi ASCII dari Offset Byte Tanggal Buat (14–17)

Tanggal dan waktu sedemikian rupa sehingga bersifat matematis karena bersifat inkremental. Mewakili data inkremental seperti itu dengan ASCII akan sulit dan akan cepat menjadi tidak praktis, karena setiap detik dalam waktu membutuhkan representasi binernya sendiri. Seperti yang kita ketahui, waktu tidak terbatas dan kita tidak memiliki jumlah bit yang

tidak terbatas untuk memenuhi semua hasil yang mungkin untuk merepresentasikan waktu. Jadi, pendekatan matematis harus digunakan.

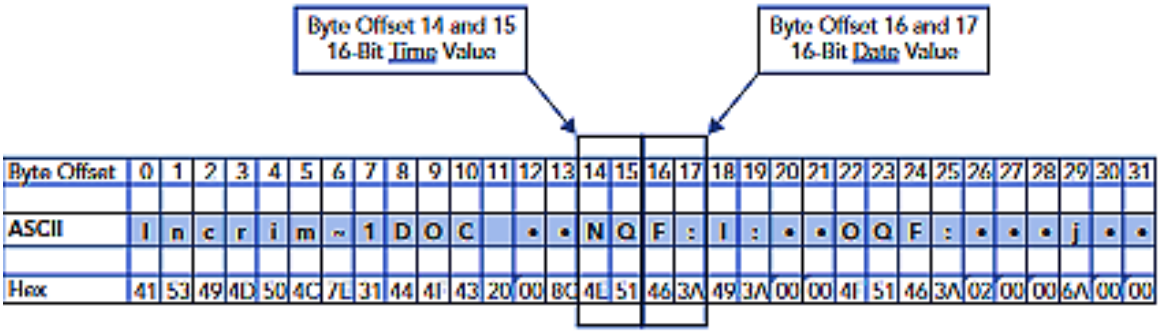
Seperti yang dapat kita lihat pada Gambar 11.3, buat tanggal dan waktu diberi empat nilai atau byte HEX. Kita tahu ada delapan bit per byte, jadi 8(4) = 32 total bit. Nilai tanggal dan waktu MS-DOS adalah nilai 32-bit yang menentukan bulan, hari, tahun, dan waktu hari (jam, menit, detik).

Sistem waktu 32-bit beroperasi sebagai dua nilai 16-bit:

- Satu nilai 16-bit yang berisi tanggal
- Satu nilai 16-bit yang berisi waktu

Dengan kata lain, tanggal dan waktu masing-masing disimpan secara terpisah dalam nilai biner dua byte. Dua nilai 16-bit = satu nilai 32-bit. Jadi, 32 bit diperlukan untuk merepresentasikan tanggal dan waktu.

Oleh karena itu, pada Gambar 11.4:



Gambar 11.4 Nilai Waktu dan Tanggal 16-Bit

Dua (2) byte 4E 51 = 16 bit = Waktu
Dua (2) byte 46 3A = 16 bit = Tanggal

Kami menyebutkan bahwa mengonversi tanggal dan waktu dari HEX ke ASCII bukanlah solusinya. Bagaimana dengan mengonversi nilai HEX ke desimal yang setara? Keluarkan kalkulator ilmiah lama.

Mengubah nilai HEX, byte 4E 51, mewakili Waktu, menghasilkan nilai desimal 20.049. Mengubah nilai HEX, byte 46 3A, mewakili nilai Tanggal, menghasilkan nilai desimal 17.978. Agar dapat dihitung dengan benar, Tanggal dan Waktu memerlukan tiga bidang data:

Tanggal = Tahun, Bulan, dan Hari
Waktu = Jam, Menit, dan Detik

Catatan: mekanisme ketepatan waktu ini berfungsi untuk tujuan yang ditentukan oleh Microsoft. Perlu diingat, ada banyak area di mana metode ketepatan waktu yang agak tidak tepat dan luas ini memiliki kekurangan yang serius. Banyak hal dalam sains, kedokteran,

manufaktur, dan bidang serupa memerlukan ketelitian yang lebih tinggi: milidetik atau mikrodetik untuk dilacak.

Jelas, persyaratan ini akan memiliki beberapa kelemahan serius dalam lingkungan seperti itu yang mencatat waktu hanya untuk detik, misalnya. Tetapi untuk tujuannya dan pada zamannya stempel waktu MS-DOS 32-bit sudah cukup, dan bagi kebanyakan orang dalam kehidupan sehari-hari, tiga bidang untuk mencatat tanggal dan tiga bidang untuk waktu pelacakan sudah lebih dari cukup.

Bahkan ketika NTP digunakan, mungkin ada sedikit perbedaan jam antara komputer yang berbeda. Meskipun perbedaan waktu berada dalam rentang milidetik, hal ini dapat menjadi masalah jika diperlukan tingkat akurasi yang tinggi, yang berpotensi membuat korelasi waktu menjadi sulit lagi. Untuk sebagian besar investigasi forensik dunia maya, hal ini mungkin tidak penting.

Kami memerlukan tiga bidang terpisah untuk tanggal namun hanya memiliki dua byte untuk melakukannya, dan hal yang sama berlaku untuk waktu. Dua byte sama dengan 16 bit, dan 16 bit dapat dipisahkan untuk mewakili tiga bidang (lihat Gambar 11.5).

Two (2) Bytes 14 and 15
16-Bit Time Value

Hours	Minutes	Seconds
5 bits	6 bits	5 bits
01010	001010	01110

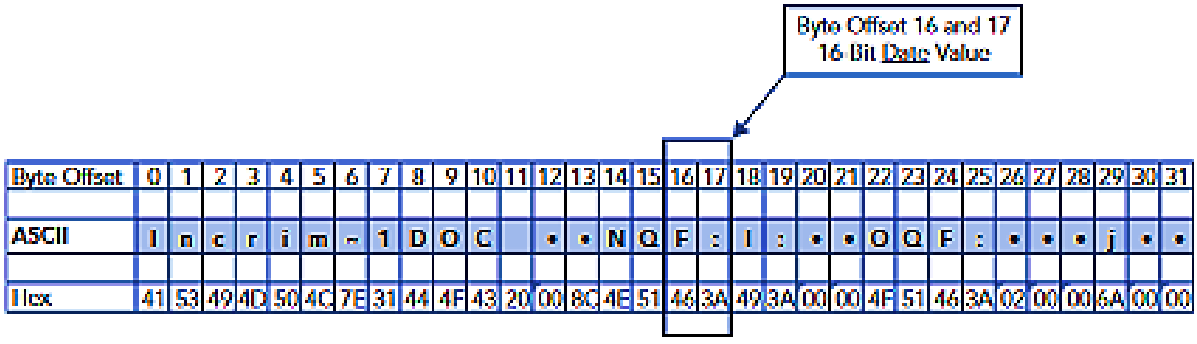
Two (2) Bytes 16 and 17
16-Bit Date Value

Year	Month	Day
7 bits	4 bits	5 bits
0011101	0010	00110

Gambar 11.5 Dua Byte, 16 Bit: Tiga Bidang Diperlukan untuk Mewakili Tanggal dan Waktu

11.7 PENENTUAN TANGGAL

Seperti waktu, tanggal disimpan sebagai nilai biner dua byte. Jadi nilai tanggal di sini, diwakili oleh offset byte 16-17, adalah HEX 463A (lihat Gambar 11.6).



Gambar 11.6 Nilai Penyimpanan Tanggal 46 3A

Pertama, yang perlu kita lakukan adalah membalikkan endianness menjadi little endian karena data ini diurutkan seperti itu. Ini menghasilkan tanggal yang direpresentasikan sebagai HEX 3A46.

Tabel 11.1 Bentuk Biner HEX 3A46

3	A	4	6
0011	1010	0100	0110

Selanjutnya kita memecah nilai HEX menjadi bentuk biner mentahnya (yaitu, menjadi bit; lihat Tabel 11.1).

Kami membutuhkan empat set 4, atau dua set 8, sehingga total 16 bit (0011101001000110).

Ketiga, bilangan biner disusun kembali dan dibagi menjadi tiga bagian yang mewakili tiga bidang tanggal: Tahun, Bulan, dan Hari. Mari baca lebih lanjut untuk melihat bagaimana hal ini dicapai.

Bulan

Berapa bulan dalam satu tahun? 12! Berapa banyak bit yang diperlukan untuk mewakili 12 nilai individual dan terpisah?

1 bit = 2 nilai (contoh sakelar lampu on/off)
2 bit = 4 nilai (contoh musim)
3 bit = 8 nilai
4 bit = 16 nilai

Empat bit memberi kita 16 nilai potensial, cukup untuk mewakili 12 bulan dalam setahun.

Hari

Berapa hari (maksimum) dalam sebulan? Sekali lagi pertanyaan yang mudah. Secara alami, 31 hari adalah jumlah maksimum yang dibutuhkan. Berapa banyak bit yang diperlukan untuk mewakili 31 nilai individu dan terpisah?

1 bit = 2 nilai
2 bit = 4 nilai
3 bit = 8 nilai
4 bit = 16 nilai
5 bit = 32 nilai

Lima bit memberi kita 32 nilai potensial, cukup untuk mewakili maksimum 31 hari per bulan.

Tahun

Berapa tahun dalam kekekalan? Kami sejauh ini telah menetapkan sembilan bit untuk mewakili bulan dan hari; ini memberi kita tujuh bit untuk mewakili nilai tahun.

1 bit = 2 nilai
2 bit = 4 nilai
3 bit = 8 nilai
4 bit = 16 nilai
5 bit = 32 nilai
6 bit = 64 nilai
7 bit = 128 nilai

Jadi kami memiliki 128 nilai potensial untuk tahun, yang dibatasi oleh perhitungan apa pun. Rumus waktu/tanggal Microsoft 32-bit menjadi rumit lagi selama bertahun-tahun. Menurut MS DOS, dunia dimulai pada 1 Januari 1980. Nilai biner tujuh bit diubah menjadi desimal dan

nilai yang dihasilkan ditambahkan ke MS DOS mulai tahun 1980. Nilai yang dihasilkan kemudian adalah tahun sebenarnya.

Tabel 11.2 Representasi Biner Tahun + Bulan + Hari

Tahun	**Bulan**	**Hari**
7 bit 0011101	4 bit 0010	5 bit 00110

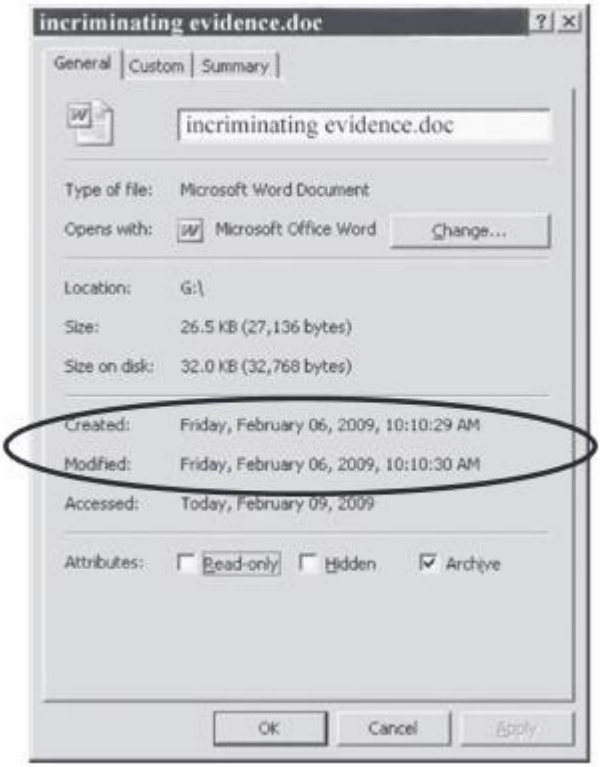
Mari kita lihat contoh kita, beralih dari Tabel 11.1, dan menetapkan bit tahun, bulan, dan hari yang sesuai (lihat Tabel 11.2). Konversikan nilai biner untuk tahun, bulan, dan hari menjadi padanan desimalnya masing-masing (lihat Tabel 11.3). Ingatlah untuk menemukan nilai yang tepat untuk tahun; kita harus menambahkan nilai desimal yang diperoleh, dalam hal ini 29, ke nilai awal tahun Microsoft 1980 (lihat Tabel 11.4).

Tabel 11.3 Setara Desimal Tahun + Bulan + Hari

	Tahun	**Bulan**	**Hari**
	7 bit	4 bit	5 bit
Biner	0011101	0010	00110
Desimal	29	2	6

Tabel 11.4 Menentukan Nilai Desimal yang Benar untuk Tahun + Bulan + Hari

	Tahun	**Bulan**	**Hari**
	7 bit	4 bit	5 bit
Biner	0011101	0010	00110
Desimal	29	2	6
TAMBAHKAN Nilai MS "Dunia Dimulai".	1980		
Nilai Tanggal HEX 3A46	**2009**	**2**	**6**



Gambar 11.7 Bukti yang memberatkan.doc

Jika kita melihat properti dokumen (lihat Gambar 11.7), kita dapat memverifikasi temuan tahun, bulan, dan hari kita.

Nilai tanggal HEX 3A46 sama dengan 6 Februari 2009 (2-6-2009) dan nilai waktu HEX 4E51 sama dengan 10:10:28 AM.

Setelah memeriksa bagaimana tanggal MS-DOS 16-bit ditentukan, sekarang kita mengalihkan perhatian kita untuk mengeksplorasi dengan tepat bagaimana dua byte atau 16 bit yang digunakan untuk mewakili waktu ditentukan.

11.8 PENENTUAN WAKTU

Untuk menentukan waktu yang tepat, pertama-tama kita harus membalik endianness saat ini dari waktu HEX dari 4E51 (lihat Gambar 11.4, offset byte 14 dan 15) ke little endian, sehingga data diurutkan seperti itu, Time = 514E.

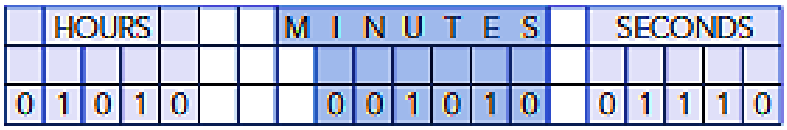
Kedua, kita perlu memecah nilai HEX menjadi bentuk biner mentahnya, menjadi bit. Kami membahas ini sebelumnya di Bab 7 saat menentukan lokasi Cylinder Head Sector (CHS). Konsep 32-bit timestamp mungkin rumit tetapi semua komponen dan konsepnya, seperti endianness dan reordering bits, telah dibahas di bab-bab sebelumnya. Dengan demikian, mengubah setiap elemen Time, byte offset 51 dan 4E, menjadi ekivalen biner representatifnya menghasilkan nilai biner masing-masing 01010001 dan 01001110. Kami membutuhkan empat set empat, atau dua set delapan, menghasilkan total 16 bit untuk mewakili Waktu.

Ketiga, bilangan biner disusun ulang dan dibagi menjadi tiga bagian yang mewakili tiga "elemen" waktu: jam, menit, dan detik, seperti yang ditunjukkan pada Gambar 11.8. Seperti biasa, jawaban mengapa jenis pengaturan dan pemisahan biner khusus ini adalah bahwa ada

kode tertulis yang membuat hal ini terjadi. Tetapi mengapa menulis kode untuk memecah waktu sedemikian rupa? Mengapa pemisahan lima bit biner (jam), enam bit (menit), dan lima bit (detik)?

Alasannya sudah dibahas di bab pertama buku ini. Itu semua ada hubungannya dengan kemungkinan hasil atau kombinasi. Seperti yang telah dibahas, untuk merepresentasikan keadaan on atau off dari saklar lampu, Anda hanya memerlukan satu bit—a1 (On) atau 0 (Off). Ide yang lebih kompleks harus direpresentasikan (misalnya, musim: musim dingin, musim semi, musim panas, dan musim gugur).

Dua kemungkinan hasil dapat diwakili dengan satu bit, namun ada empat musim. Jadi, untuk menyajikan empat musim, Anda memerlukan cukup bit untuk memberi Anda empat kemungkinan hasil. Dua bit dapat memberi Anda empat kemungkinan hasil: 00, 01, 10, dan 11. Sekarang masing-masing nilai tersebut dapat ditetapkan ke satu musim. Hal yang sama berlaku untuk waktu. Mari baca lebih lanjut untuk melihat bagaimana waktu dihitung dan direpresentasikan.



Gambar 11.8 Setara Biner Jam, Menit, Detik (0101000101001110)

Jam

Berapa jam dalam satu hari? Mudah, 24! Berapa banyak bit yang diperlukan untuk mewakili 24 nilai individu dan terpisah? Ingat dari bab kami sebelumnya:

1 bit = 2 nilai (contoh sakelar lampu on/off)
2 bit = 4 nilai (contoh empat musim)
3 bit = 8 nilai
4 bit = 16 nilai
5 bit = 32 nilai

Ingat biner adalah Basis 2 jadi semuanya dikuadratkan. Jadi, lima bit memberi kita 32 nilai potensial, cukup untuk mewakili 24 jam sehari. Akibatnya, total 16 bit untuk mewakili waktu, kurang dari lima bit per jam, sama dengan sisa 11 bit.

Menit

Berapa menit dalam satu jam? Sekali lagi, mudah: 60 menit. Berapa banyak bit yang diperlukan untuk mewakili 60 nilai individual dan terpisah?

1 bit = 2 nilai
2 bit = 4 nilai
3 bit = 8 nilai
4 bit = 16 nilai
5 bit = 32 nilai
6 bit = 64 nilai

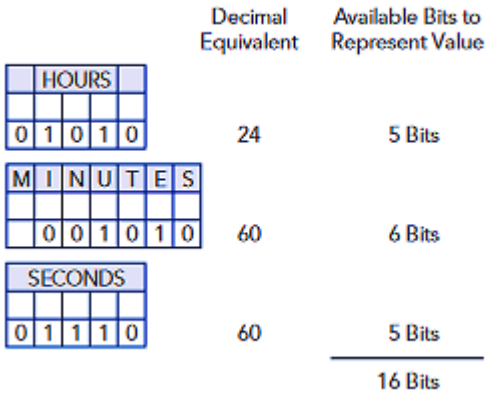
Enam bit memberi kita 64 nilai potensial, cukup untuk mewakili 60 menit dalam satu jam. 11 bit tersisa, kurang dari enam bit untuk menit, sama dengan lima bit tersisa.

Detik

Berapa detik dalam satu jam? Terlalu mudah, 60! Berapa banyak bit yang diperlukan untuk mewakili 60 nilai individual dan terpisah?

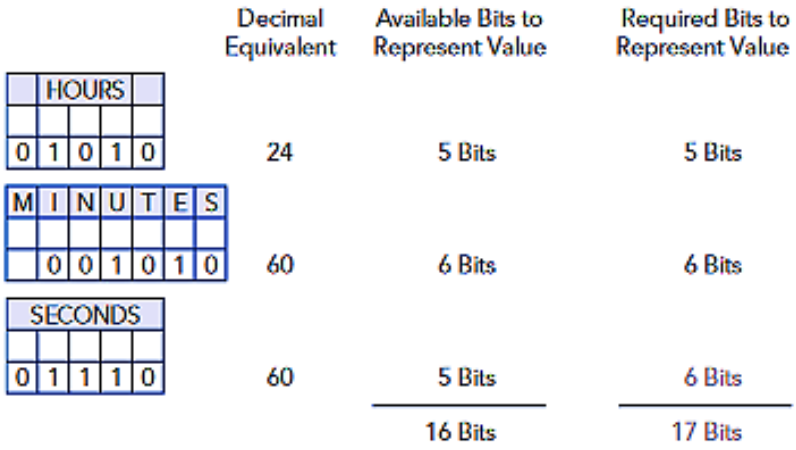
1 bit = 2 nilai
2 bit = 4 nilai
3 bit = 8 nilai
4 bit = 16 nilai
5 bit = 32 nilai
6 bit = 64 nilai

Diperlukan enam bit untuk mewakili 60 detik individu dengan benar dalam satu menit. Lihat Gambar 11.9.



Gambar 11.9 Tersedia 16 Bit untuk Mewakili Jam + Menit + Detik

Tapi kami hanya memiliki lima bit tersisa! Wah, kita butuh enam bit. Tunggu, hanya tersisa lima bit untuk mewakili detik. Apa yang telah terjadi? Bagaimana kita bisa secara akurat merepresentasikan detik dengan hanya lima bit atau 32 total nilai potensial? Kita tidak bisa! Logikanya, karena hanya tersisa lima bit, dan kita membutuhkan nilai maksimum 60, Microsoft menggunakan nilai 30 dan mengalikannya dengan 2, sehingga memberi kita nilai maksimum untuk detik. Lihat Gambar 11.10. Sistem waktu 32-bit Microsoft pada dasarnya membulatkan ke detik terdekat!



Gambar 11.10 17 Bit Diperlukan untuk Mewakili Jam + Menit + Detik

Jika kita memecah hari menjadi detik individu, kita menemukan bahwa ada 86.400 detik dalam 24 jam sehari. Entri direktori dua byte untuk tanggal berisi 16 bit. Berapa banyak kemungkinan nilai unik yang ada untuk 16 bit? Jawaban: 65.536. Kekurangannya dengan cepat terlihat di sini juga.

Kembali ke file kita "Incriminating evidence.doc" (Gambar 11.11). Dari byte offset 14 dan 15, bit 0–4 mewakili jam, bit 5–10 mewakili variabel menit, dan bit 11–15 mewakili detik. (Lihat Tabel 11.5.)

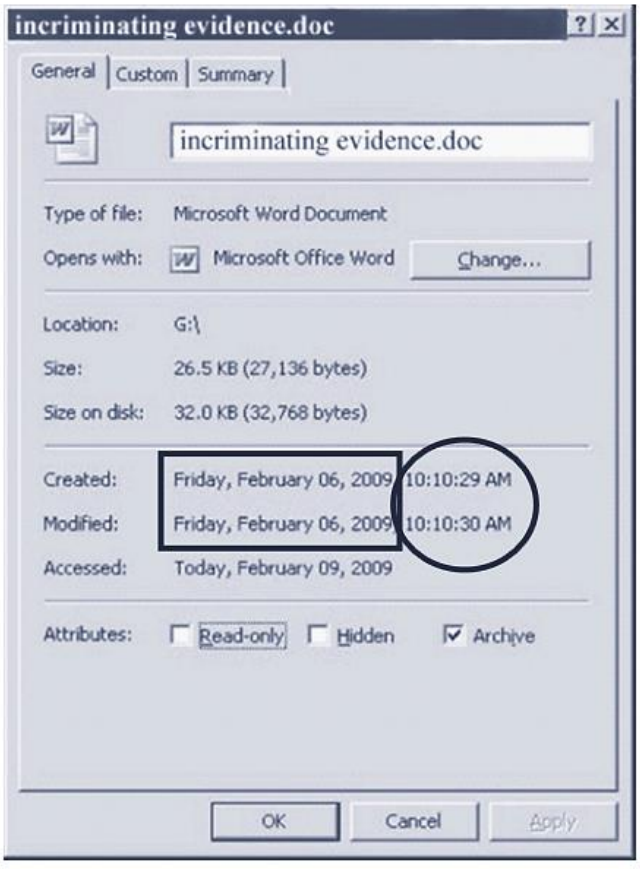
Byte Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ASCII	I	n	c	r	i	m	~	1	D	O	C		•	•	N	Q	F	:	I	:	•	•	O	Q	F	:	•	•	•	j	•	•
Hex	41	53	49	4D	50	4C	7E	31	44	4F	43	20	00	8C	4E	51	46	3A	49	3A	00	00	4F	51	46	3A	02	00	00	6A	00	00

Gambar 11.11 Bukti yang memberatkan.doc

Tabel 11.5 Setara Desimal untuk HEX 514E

Bit	Bit 0–4	Bit 5–10	Bit 11–15
Biner	0-1-0-1-0	0-0-1-0-1-0	0-1-1-1-0
Desimal	10	10	14
Perwakilan	Jam	Menit	Detik

Catatan: Hasil desimal (detik) harus dikalikan dengan dua (2).



Gambar 11.12 Bukti yang memberatkan.doc

Kita dapat melihat waktu, oleh karena itu, adalah 10:10:28. (Perhatikan bahwa jam didasarkan pada jam 24 jam, jadi, jam 10 pagi direpresentasikan dengan nilai 10 dan jam 10 malam akan direpresentasikan dengan nilai 22, misalnya.) Jangan lupa untuk mengalikan detik, dalam contoh kita (yaitu, 14) dikalikan 2 untuk mendapatkan nilai yang benar untuk indikator "detik" (yaitu, 28). Karena nilai ini dikalikan dengan dua maka akan selalu genap, oleh karena itu dibulatkan ke detik genap terdekat. Jika kita melihat properti dokumen (lihat Gambar 11.12) kita dapat memverifikasi temuan tanggal dan waktu seperti yang dijelaskan sebelumnya.

Nilai tanggal HEX 3A46 sama dengan 6 Februari 2009 (2-6-2009) dan nilai waktu HEX 4E51 sama dengan 10:10:28 AM.

11.9 KETIDAKTEPATAN WAKTU

Create Time, yang kami hitung dari data yang terkandung dalam entri direktori FAT untuk dokumen yang memberatkan bukti.doc (lihat Tabel 11.5), adalah 10:10:28, namun jendela properti, seperti yang ditunjukkan pada Gambar 11.12, menunjukkan waktu 10:10:29. Entri direktori juga berisi tanggal dan waktu file dibuat atau terakhir diubah. Waktu akurat hanya untuk plus atau minus 2 detik karena disimpan dalam bidang dua byte, yang hanya dapat menyimpan 65.536 nilai unik (sehari berisi 86.400 detik unik). Bidang waktu dibagi menjadi detik (lima bit), menit (enam bit), dan jam (lima bit). Tanggal dihitung dalam hari menggunakan tiga subbidang: hari (lima bit), bulan (empat bit), dan tahun—1980 (tujuh bit). Dengan angka tujuh-bit untuk tahun dan waktu yang dimulai pada tahun 1980, tahun tertinggi yang dapat diekspresikan adalah 2107. Dengan demikian, MS-DOS memiliki masalah Y2108 bawaan.15 Ketidakakuratan waktu 32-bit windows staMp terlihat jelas di sini. Kami dapat melihat perbedaan ini di sini karena file diakses dari partisi NTFS, partisi aktif. File ini disimpan di partisi FAT16 (tidak aktif) tetapi diakses dari partisi NTFS yang aktif dengan OS Windows XP. Ada metadata yang terkandung dalam file aplikasi seperti Microsoft Word, dan ini memungkinkan sistem file NTFS aktif yang kuat untuk mengambil waktu yang lebih tepat.

11.10 RINGKASAN

Dalam investigasi forensik dunia maya, seperti halnya dalam investigasi forensik kriminal dan medis, mengetahui waktu yang tepat ketika suatu peristiwa terjadi dapat menjadi sangat penting. Ini bisa jadi karena kebutuhan untuk mengkorelasikan peristiwa dari sistem yang berbeda, membangun alibi, atau untuk mencari tahu kapan peristiwa terjadi sehubungan dengan peristiwa di dunia nyata.

Jelas, kurangnya kesepakatan global tentang waktu berpotensi menghambat penyelidikan forensik di mana ada kebutuhan untuk korelasi kejadian untuk kejadian yang dikumpulkan dari sumber yang berbeda, serta untuk korelasi kejadian komputer dengan kejadian di dunia nyata. Banyaknya jumlah host yang tidak disinkronkan ke UTC menunjukkan bahwa ada kemungkinan besar terjadinya masalah korelasi peristiwa ketika lebih dari satu host terlibat dalam analisis forensik. Hal ini terutama berlaku jika diperlukan tingkat presisi yang tinggi untuk stempel waktu.

Harus diakui bahwa sinkronisasi jam pada dasarnya membutuhkan waktu yang lama dan banyak perbandingan untuk mempertahankan ketepatan waktu. Meskipun hanya beberapa pengukuran yang biasanya memadai untuk menentukan waktu lokal secara andal hingga satu detik atau lebih, periode berjam-jam dan lusinan pengukuran diperlukan untuk mengatasi kemiringan osilator dan mempertahankan waktu lokal hingga urutan milidetik. Dengan demikian, akurasi yang dicapai secara langsung bergantung pada waktu yang dibutuhkan untuk mencapainya.

Ketidakakuratan waktu sangat luas dan luas. Ketidakakuratan dapat berupa ketidakakuratan seluruh sistem ke server NTP, atau khusus sistem karena kemiringan jam. Ketidakakuratan dapat spesifik untuk lokasi geografis tertentu karena membingungkan zona waktu atau sistem file sistem operasi tertentu, membulatkan detik ganjil ke detik genap terdekat. Perbedaan bisa sepanjang waktu mengizinkan atau sesingkat satu detik.

Ketidakakuratan waktu ini sangat besar jika dibandingkan dengan ketidakakuratan waktu stempel waktu MS-DOS 32-bit satu detik. Sangat tidak mungkin bahwa ketidaktepatan waktu satu detik (seperti yang terlihat dalam stempel waktu MS-DOS 32-bit) akan sangat penting untuk menguatkan ketidakbersalahan (atau kesalahan) seseorang. Satu kesamaan ketidakakuratan waktu adalah bahwa mereka dapat mendiskreditkan seorang ahli, seperti penyelidik forensik dunia maya.

Penasihat lawan dapat mencoba mendiskreditkan penyelidik karena menyatakan ketidakakuratan dalam pemeriksaan bukti (misalnya, file dibuat pada 10:10:28 padahal sebenarnya dibuat pada 10:10:29). Bisakah ini dijelaskan? Ya itu bisa. Mungkinkah ini menimbulkan keraguan dalam pikiran juri tentang keakuratan penyelidikan? Ya.

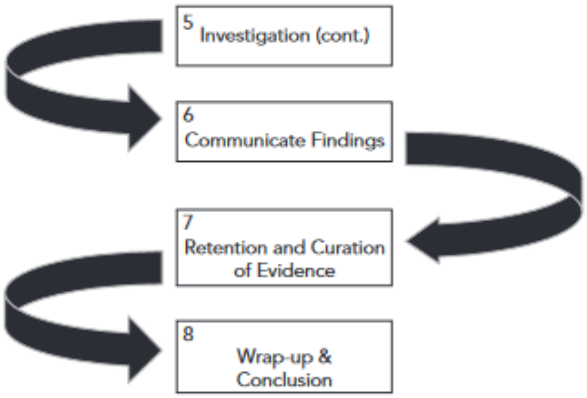
Dari pembahasan berikut tentang pentingnya waktu, baik untuk penyelidikan maupun penyelidik forensik dunia maya, di Bab 12 kita akan bergabung kembali dengan penyelidikan Ronelle terhadap aktivitas Jose, pada langkah penutup dari proses Praktik Cerdas Investigasi yang dimulai di Bab 10, mengkomunikasikan temuan.

BAB 12
INVESTIGASI
PENUTUPAN INSIDEN

Saya pikir saya melakukannya dengan cukup baik, mengingat saya memulai hanya dengan setumpuk kertas kosong.

—Steve Martin

Meskipun prosesnya cukup linier (lihat Gambar 12.1), "langkah-langkah" proses investigasi tidak harus berturut-turut atau berturut-turut; mereka mungkin tumpang tindih dan bervariasi tergantung pada kasus. Fase itu sendiri memungkinkan penyesuaian khusus yang sesuai dengan berbagai persyaratan: legal, sah, perusahaan, atau lainnya.



Gambar 12.1 Langkah-langkah Proses Investigasi

12.1 PRAKTIK CERDAS INVESTIGASI FORENSIK

LANGKAH 5: INVESTIGASI (LANJUTAN)

"penyelidikan"

1. Tindakan menyelidiki sesuatu atau seseorang; pemeriksaan atau penelitian formal atau sistematis.
2. Penyelidikan formal atau studi sistematis.

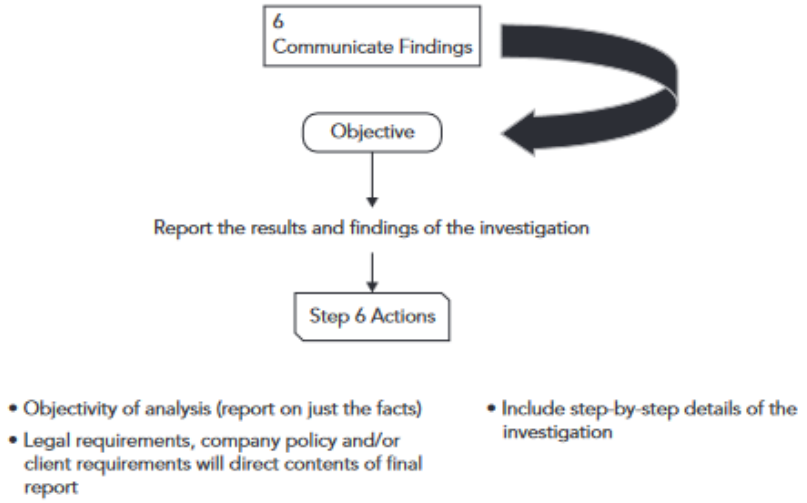
Dalam beberapa keadaan, investigasi forensik dunia maya dapat didefinisikan sebagai tindakan mengekstraksi data untuk memenuhi kriteria pencarian tertentu; sesuatu yang mudah dicapai dengan cara otomatis. Definisi ini tentu bisa berlaku jika di bawah kewajiban kontraktual atau perintah pengadilan. Aspek "Sherlock Holmes" dari suatu penyelidikan mungkin tidak selalu muncul dengan sendirinya; penyelidikan mungkin sangat terbatas pada kriteria pencarian. Kadang-kadang, penyelidik forensik dunia maya menjadi terlalu peka terhadap sifat sistematis dan menjadi tergantung dan akhirnya bergantung pada alat forensik untuk ekstraksi data. Jika tidak ingin tahu, Ronelle tidak akan memperhatikan, atau memperhatikan, perbedaan waktu yang diamati selama penyelidikan. Tantangan atau pertanyaan inilah yang, kadang-kadang, membentuk sifat investigasi yang dialami dalam forensik dunia maya. Sifat investigasi ini terjamin dengan kemajuan teknologi.

Sistem Operasi sangat banyak dan sering berubah, bersama dengan pemutakhiran, tambalan, pembaruan, paket layanan, dan sebagainya yang konstan. Peningkatan volume dan keragaman jenis perangkat elektronik juga memperkenalkan berbagai jenis dan versi sistem operasi baru. Sifat dinamis dari lapangan inilah yang memicu kebutuhan akan banyak penyelidikan. Seringkali penting bagi penyelidik untuk mundur dan menjelajahi "lubang kelinci" atau ketidakteraturan (perbedaan stempel waktu 32-bit dalam kasus Ronelle, misalnya) untuk akhirnya dapat menjelaskan mengapa hal seperti itu ada. Latihan apa pun untuk memperluas atau mendapatkan pemahaman yang lebih baik tentang bidang yang begitu penting patut untuk ditelusuri.

Sekarang dipersenjatai dengan pemahaman yang kuat tentang perbedaan waktu, Ronelle merasa percaya diri untuk memahami sepenuhnya data yang dia ekstrak. Jika diminta, Ronelle dapat memeriksa data yang dikumpulkan dan mungkin memperoleh beberapa kata kunci tambahan atau kriteria pencarian lainnya. Bagaimana hasil penyidik akan bervariasi secara drastis di lanskap forensik. Ronelle mempresentasikan temuan awal kepada manajemen dan menyarankan pencarian kata kunci baru. Dalam keadaan ini, departemen Hukum membuat keputusan bahwa mereka memiliki apa yang mereka butuhkan dan menghentikan penyelidikan. Alasan untuk menunda penyelidikan dapat sangat bervariasi. Salah satu faktor penentu utama adalah aspek keuangan; melakukan investigasi forensik dunia maya bisa menjadi urusan yang mahal.

LANGKAH 6: KOMUNIKASIKAN TEMUAN

Penyelidik forensik dunia maya sekarang perlu mengomunikasikan hasil dan temuan investigasi, yang dapat disampaikan adalah laporan penyelidik. Lihat Gambar 12.2. Penyelidik forensik siber bertanggung jawab untuk secara akurat dan obyektif melaporkan temuannya dan hasil analisis pemeriksaan bukti digital. Pendokumentasian adalah proses yang berkelanjutan selama pemeriksaan/penyelidikan; penting untuk secara akurat mencatat langkah-langkah yang diambil selama proses pemeriksaan bukti digital. Semua dokumentasi harus lengkap, akurat, dan komprehensif. Laporan yang dihasilkan harus ditulis untuk audiens yang dituju.



Gambar 12.2 Langkah 6: Komunikasikan Temuan

Tujuan dari laporan adalah untuk:

1. Kirimkan hasil kriteria pencarian yang ditentukan dalam permintaan.
2. Dokumentasikan temuan dengan cara yang tidak memihak dan akurat dan berikan informasi otoritas yang bertanggung jawab, untuk membantu membuat keputusan apakah akan mengambil tindakan korektif, remedial, atau disipliner.
3. Mengatur informasi sehingga setiap orang dapat membaca dan memahami laporan tanpa mengacu pada lampiran atau materi lainnya.

Penyidik forensik dunia maya biasanya dilarang menerbitkan, mendistribusikan, atau dengan cara apa pun mengomunikasikan, secara independen, isi laporan kepada siapa pun selain orang yang berwenang.

12.2 KARAKTERISTIK LAPORAN CYBER FORENSIC YANG BAIK

Kejelasan, kelengkapan, objektivitas, dan akurasi merupakan ciri laporan yang baik. Laporan harus cukup jelas sehingga orang lain dapat memahami maksud penulis. Namun lebih dari itu, harus ditulis dengan sangat jelas sehingga orang lain tidak mungkin salah memahami maksud penulisnya. Kejelasan dihasilkan dari sebuah laporan yang berisi susunan fakta dan analisis yang ringkas dan sistematis yang dinyatakan dalam istilah yang tepat dan netral. Kelengkapan menentukan bahwa semua informasi yang secara wajar ingin dipertimbangkan oleh manajer yang berhati-hati sebelum mengambil keputusan harus muncul dalam laporan. Objektivitas bisa dibilang merupakan sifat yang paling penting. Seorang penyelidik yang mengungkap data yang memberatkan tidak dapat membiarkan dirinya terombang-ambing, sehingga mempengaruhi disposisi laporan. Sederhananya, penyidik \u200b\u200btidak bisa "berpihak". Kurangnya objektivitas akan berdampak negatif pada gaya dan nada laporan. Sebagai Sersan. Joe Friday akan menyatakan, "Hanya fakta, Bu."

Akurasi mensyaratkan tidak ada kesalahan dalam melaporkan fakta atau mengidentifikasi orang, tempat, peristiwa, tanggal, dokumen, dan hal-hal nyata lainnya. Aturan praktis yang baik mengharuskan seseorang yang tidak tahu apa-apa tentang kasus tersebut dapat membaca laporan, memahami sepenuhnya apa yang terjadi, dan merasa percaya diri dalam mengambil keputusan berdasarkan isinya.

Gaya dan Nada

Apakah tuduhan itu dipertahankan atau disangkal, sebagian besar laporan menyampaikan kabar buruk kepada seseorang. Gaya dan nada yang tepat membuat berita lebih mudah diterima; gaya atau nada yang tidak pantas menghalangi penerimaan dan resolusi yang tepat. Gaya bervariasi dari satu orang ke orang lain, tetapi pendekatan langsung yang sederhana, tanpa bahasa yang penuh warna, adalah cara paling efektif untuk menyampaikan fakta. Nadanya juga harus netral, tidak menghakimi, meyakinkan dalam kesopanan bahasanya, dan tidak provokatif dalam uraiannya. Gaya, nada, dan kejelasan harus saling melengkapi; masing-masing ditangani dengan baik cenderung untuk mencapai yang lain.

Analisis

Dalam sebagian besar investigasi, lebih banyak informasi dikumpulkan daripada yang diperlukan untuk mencapai kesimpulan. Beberapa informasi berlebihan; informasi lain tidak

berkaitan dengan keputusan. Terkadang informasinya bertentangan. Dalam kasus-kasus di mana tindakan remedial, disipliner, atau hukum dimungkinkan, keputusan untuk menerima kesimpulan dalam sebuah laporan kemungkinan besar hanya akan dibuat setelah pemeriksaan semua bahan pembuktian yang terkumpul. Jadi, memutuskan informasi apa yang akan diperlakukan sebagai bukti dan bagaimana menghadapinya dalam laporan itu penting. Jika laporan tersebut tampaknya tidak membahas bukti yang relevan secara adil, kesimpulannya dapat ditolak.

Beberapa masalah umum meliputi:

1. Bukti yang dipertimbangkan, tetapi tidak diandalkan, harus didiskusikan dalam laporan jika ada kemungkinan orang lain ingin mempertimbangkannya atau mempertanyakan kelengkapan laporan jika tidak disebutkan. Ini penting ketika ada bukti yang bertentangan. Kegagalan untuk membahas dan menjelaskan mengapa satu versi peristiwa diandalkan sebagai pengganti bukti yang bersaing akan menyebabkan pembaca yang mengetahui konflik tersebut mempertanyakan objektivitas penulis.
2. Bukti yang berlebihan atau berulang dapat diringkas jika berasal dari berbagai sumber yang tidak menyajikan informasi unik.
3. Analisis pembuktian harus menyatukan semua fakta dokumenter, fisik, dan kesaksian yang berkaitan dengan tuduhan untuk mencapai kesimpulan.

Fakta-fakta yang diandalkan untuk mencapai setiap kesimpulan harus jelas bagi pembaca. Ketika standar yang berlaku itu sendiri tidak jelas, atau kesaksian bertentangan, penalaran yang mengarah pada kesimpulan tidak selalu jelas. Dalam hal itu, analisis dalam laporan harus menjelaskan kepada pembaca bagaimana penyidik sampai pada kesimpulan.[3]

Isi Laporan

Berita acara penyidik dapat berupa ringkasan singkat hasil pemeriksaan yang dilakukan terhadap barang yang diserahkan untuk dianalisis. Sekali lagi, seperti kebijakan forensik, laporannya akan bervariasi secara drastis antara organisasi dan jenis organisasi. Laporan perusahaan akan memiliki audiens yang berbeda dari laporan dari penegak hukum.

Laporan tersebut dapat meliputi:

- Identitas lembaga pelapor atau departemen peminta.
- Seperti yang terlihat pada halaman judul laporan contoh: “Investigasi Forensik, ABC Inc.” (lihat Lampiran) Ini mengidentifikasi “penulis” laporan.
- Pengidentifikasi kasus atau nomor pengiriman.
- Pengidentifikasi unik adalah kunci dalam melacak apa pun, misalnya kasus unik nomor untuk tiket dukungan meja bantuan atau insiden keamanan.
- Nama penyidik perkara.
- Identitas pengirim.
- Tanggal penerimaan.
- Tanggal laporan.
- Daftar deskriptif barang yang diserahkan untuk pemeriksaan, termasuk nomor urut, membuat, dan model.
- Daftar deskriptif item yang digunakan untuk menyelidiki termasuk perangkat keras dan perangkat lunak.

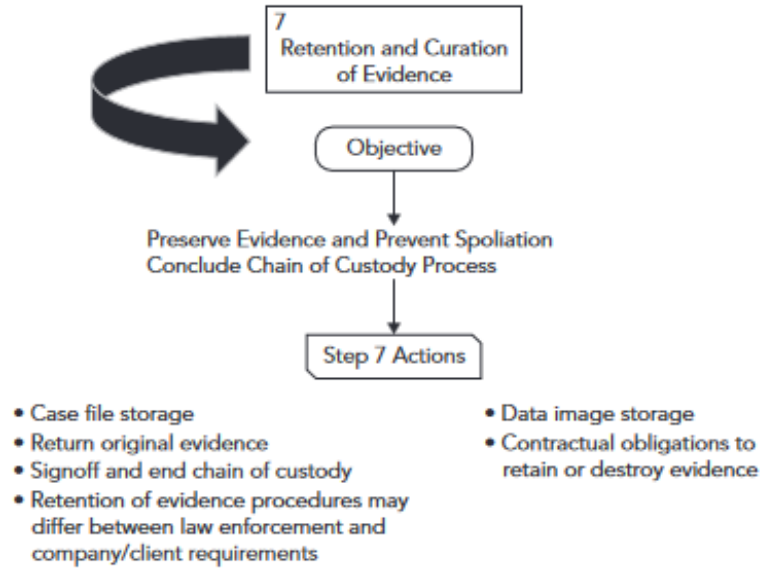
- Penjelasan singkat tentang langkah-langkah yang dilakukan selama pemeriksaan, seperti pencarian string, pencarian gambar grafik, dan memulihkan file yang terhapus.
- Dokumentasi/formulir lacak balak.
- Hasil/kesimpulan.

Ronelle menyertakan salinan formulir lacak balak (CoC) dengan laporan tersebut, yang asli tersisa dengan bukti drive. Ronelle menandatangani formulir itu dua kali, sekali setelah diterima dan sekali lagi setelah dikembalikan. Ronelle tidak menandatangani formulir CoC setiap kali dia memasukkan atau mengeluarkannya dari lemari besi. Gudang bukti memiliki formulir masuk/keluar lacak balak sendiri untuk menguncinya di lemari besi atau menghapusnya.

Ketika Ronelle akhirnya mengembalikan drive bukti, departemen forensik di ABC Inc. akan mengamankan bukti di lemari besi yang membutuhkan tanda tangan untuk memeriksa bukti.

LANGKAH 7: RETENSI DAN KURASI BUKTI

Penanganan barang bukti fisik mencakup berbagai kegiatan mulai dari TKP hingga ruang sidang. Di beberapa yurisdiksi, hal ini dapat meluas hingga periode setelah persidangan, termasuk penghancuran atau pembuangan barang bukti. Bukti harus diatur dan diberi label sedemikian rupa agar tidak mengubah bukti dan memungkinkan untuk dengan mudah diidentifikasi atau diambil kembali di kemudian hari. (Lihat Gambar 12.3.)



Gambar 12.3 Langkah 7: Retensi dan Kurasi Bukti

Kerumitan dalam menangani bukti digital meningkat karena potensi replikasi data. Ketika disajikan dengan bukti asli, penyelidik biasanya akan membuat gambar forensik untuk diselidiki, dengan asumsi verifikasi hash tentunya. Karena bukti bersifat digital, gambar forensik sekarang juga dapat dianggap sebagai bukti.

Retensi bukti yang digunakan di sini menyiratkan penyimpanan, pengarsipan, penghancuran, atau pengembalian semua bukti. Ini termasuk fisik maupun logis: bukti asli,

gambar forensik, hard drive, komputer, telepon, CD, kaset, foto, laporan, catatan forensik, dan sebagainya. Tidak semua bukti kasus identik dan oleh karena itu kebijakan retensi dapat bervariasi antar jenis. Misalnya, setelah penyelidikan malware perusahaan, laptop dapat dibersihkan, dicitra ulang, dan dikembalikan ke pengguna atau mungkin diproduksi kembali.

Namun, pihak yang dituntut dalam kasus pornografi anak sebaiknya tidak mengharapkan laptop dikembalikan. Kebijakan retensi akan lebih bervariasi di antara jenis organisasi yang berbeda. Juga, dalam organisasi yang sama, kebijakan penyimpanan untuk bukti asli dapat berbeda dari gambar forensik atau bahkan laporan akhir. Kebijakan penyimpanan dan pengarsipan data forensik akan sangat bergantung pada jenis organisasi, dan mungkin tunduk pada kebijakan atau undang-undang utama yang ada. Bergantung pada praktiknya, periode retensi dapat bersifat kontraktual, kebijakan perusahaan, atau bahkan diwajibkan oleh hukum.

Praktik forensik swasta kecil mungkin memiliki kebijakan internal mengenai kebijakan retensi kecuali ditentukan lain oleh kontrak. Semua ini harus terjadi dalam batas-batas hukum, tentu saja.

Apapun retensi yang diputuskan, keamanan, organisasi, dan identifikasi bukti sangat penting. Bukti harus disimpan sesuai dengan klasifikasi datanya. Jika bukti bersifat rahasia, maka bukti tersebut tidak boleh disimpan di file share yang dapat diakses oleh semua orang atau di lemari file yang tidak aman.

Saat menyimpan bukti digital, penyelidik forensik dunia maya harus:

1. Memastikan bahwa bukti digital diinventarisasi sesuai dengan kebijakan lembaga.
2. Pastikan bahwa bukti digital disimpan di lingkungan yang aman dan terkendali iklim atau lokasi yang tidak terkena suhu atau kelembapan ekstrem.
3. Pastikan bukti digital tidak terkena medan magnet, kelembapan, debu, getaran, atau elemen lain yang dapat merusak atau menghancurkannya.

Bukti digital yang berpotensi berharga termasuk tanggal, waktu, dan pengaturan konfigurasi sistem dapat hilang karena penyimpanan yang lama jika baterai atau sumber daya yang menyimpan informasi ini rusak. Jika berlaku, mereka yang bertanggung jawab atas pengangkutan barang bukti harus memberi tahu petugas barang bukti bahwa perangkat elektronik bertenaga baterai dan memerlukan perhatian segera untuk menyimpan data yang tersimpan di dalamnya.

Perhatian awal terhadap kesulitan dalam melestarikan informasi digital difokuskan pada umur panjang media fisik tempat informasi disimpan. Bahkan di bawah kondisi penyimpanan terbaik, media digital bisa rapuh dan memiliki umur simpan yang terbatas. Selain itu, perangkat, proses, dan perangkat lunak baru menggantikan produk dan metode yang digunakan untuk merekam, menyimpan, dan mengambil informasi digital dalam siklus yang menakjubkan selama dua hingga lima tahun.

Mengingat tingkat perubahan teknologi seperti itu, bahkan media yang paling rapuh pun mungkin hidup lebih lama dari ketersediaan pembaca yang berkelanjutan untuk media tersebut. Upaya untuk melestarikan media fisik dengan demikian hanya memberikan solusi parsial jangka pendek untuk masalah umum pelestarian informasi digital. Memang, keusangan

teknologi merupakan ancaman yang jauh lebih besar terhadap informasi dalam bentuk digital daripada kerapuhan fisik yang melekat pada banyak media digital.

Jika memungkinkan, media yang biasa digunakan (daripada beberapa media penyimpanan yang tidak jelas) harus digunakan untuk pengarsipan. Akses ke bukti harus sangat dibatasi, dan harus didokumentasikan dengan jelas. Kontrol harus ada untuk mendeteksi akses tidak sah ke area penyimpanan arsip.

Sementara bukti digital yang diperoleh sebagai bagian dari penyelidikan forensik dunia maya tunduk pada prosedur khusus yang mengatur penyimpanan, penyimpanan, dan pengarsipan yang dapat diterima, pemeriksaan standar penyimpanan data di luar bidang forensik dunia maya dapat bermanfaat bagi penyelidik forensik dalam mengembangkan pemahaman yang komprehensif. kebijakan penyimpanan dan pengarsipan untuk bukti digital.

ISO 15489:2001

Kurasi digital yang sukses bergantung pada alur kerja yang kuat, yang mempertimbangkan siklus hidup lengkap sumber daya digital dari awal hingga pembuangan atau pemilihan untuk pelestarian jangka panjang. Pengembangan dan dokumentasi kebijakan, tanggung jawab, otoritas, dan skema pelatihan untuk manajemen sumber daya digital sama pentingnya dengan desain dan implementasi sistem untuk menyerap, mengelola, menyimpan, merender, dan mengaktifkan akses.

Ada banyak manfaat yang terkait dengan pengembangan, dokumentasi, dan kepatuhan terhadap metodologi alur kerja termasuk kemampuan merancang sistem yang efektif untuk semua pengguna, kemampuan untuk segera mematuhi persyaratan hukum, peraturan, dan standar, pengenalan aset, dan kemampuan untuk mengumpulkan dan melestarikan informasi dalam jangka panjang.

ISO 15489:2001 mendahului standar alur kerja kurasi digital terkenal lainnya, OAIS (Open Archival Information Systems Reference Model—ISO 14721:2003). ISO 15489 dianggap oleh banyak orang lebih mudah dipahami, karena memberikan panduan yang lebih konkret tentang pengelolaan arsip.

Dikembangkan terutama untuk manajemen catatan bisnis, ISO 15489:2001 dapat diterapkan pada manajemen catatan yang dibuat oleh aktivitas apa pun, dan juga berlaku untuk informasi digital atau hard copy.

Standar memberikan panduan untuk memastikan bahwa rekaman tetap otoritatif melalui penyimpanan karakteristik esensialnya: keaslian, keandalan, kegunaan, dan integritas. Ini menjelaskan cara memastikan rekod dikuratori dengan benar, mudah diakses, dan didokumentasikan dengan benar sejak pembuatan selama diperlukan.

ISO 15489:2001 mengidentifikasi bagaimana manajemen catatan yang sistematis dapat memastikan bahwa keputusan dan kegiatan organisasi, individu, atau proyek di masa mendatang dapat didukung melalui akses siap pakai ke bukti tindakan dan kegiatan bisnis di masa lalu, sambil memfasilitasi kepatuhan terhadap peraturan terkait. lingkungan. Kebutuhan akan kebijakan dan tanggung jawab manajemen arsip yang terdefinisi dengan jelas dan terdokumentasi dengan baik, di dalam organisasi atau proyek, bersama dengan manfaat yang akan diperoleh dari pengembangan dan penerapan, diuraikan.

Proses manajemen rekaman:
1. Tangkap
2. Pendaftaran
3. Klasifikasi
4. Klasifikasi akses dan keamanan
5. Identifikasi status disposisi
6. Penyimpanan
7. Gunakan dan lacak
8. Pelaksanaan disposisi

ISO 14721:2003

ISO 14721:2003 menentukan model referensi untuk sistem informasi arsip terbuka (OAIS). Tujuan dari ISO 14721:2003 ini adalah untuk membangun sistem pengarsipan informasi, baik digital maupun fisik, dengan skema organisasi yang terdiri dari orang-orang yang menerima tanggung jawab untuk menyimpan informasi dan membuatnya tersedia untuk komunitas yang ditunjuk.

Model referensi ini membahas berbagai fungsi pelestarian informasi arsip termasuk penyerapan, penyimpanan arsip, pengelolaan data, akses, dan penyebaran.

Model referensi juga:
1. Mengatasi migrasi informasi digital ke media baru dan membentuk model data yang digunakan untuk merepresentasikan informasi, peran perangkat lunak dalam pelestarian informasi, dan pertukaran informasi digital antar arsip.
2. Mengidentifikasi antarmuka internal dan eksternal ke fungsi arsip, dan mengidentifikasi sejumlah layanan tingkat tinggi pada antarmuka ini.
3. Memberikan berbagai contoh ilustratif dan beberapa rekomendasi “praktik terbaik”.
4. Mendefinisikan sekumpulan tanggung jawab minimal untuk sebuah arsip yang disebut OAIS, dan mendefinisikan arsip maksimal untuk menyediakan serangkaian istilah dan konsep yang berguna.

Model OAIS yang dijelaskan dalam ISO 14721:2003 dapat diterapkan pada arsip apa pun. Ini secara khusus berlaku untuk organisasi dengan tanggung jawab menyediakan informasi untuk jangka panjang. Ini termasuk organisasi dengan tanggung jawab lain, seperti pemrosesan dan distribusi sebagai respons terhadap kebutuhan program.

Penyelidik forensik dunia maya, meskipun bukan tanggung jawab pekerjaan utamanya, harus tetap mengikuti dinamika dan persyaratan yang berantai di bidang kurasi digital yang terus berkembang.

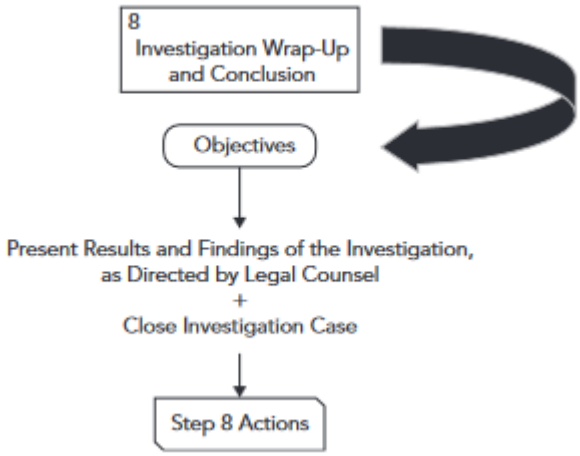
LANGKAH 8: PENUTUP DAN KESIMPULAN INVESTIGASI

Penyelesaian investigasi, sekali lagi, adalah ilmu nonspesifik, yang berarti bahwa setiap perusahaan, departemen, atau agensi akan memiliki, berdasarkan kebijakan, prosedur dan metodologinya sendiri untuk meringkas, menyelesaikan, menutup, dan menghentikan investigasi. Memang, beberapa tindakan yang diambil dalam langkah terakhir ini mungkin ditentukan oleh undang-undang, terutama jika penyelidikan tersebut akan bergerak

melampaui tindakan disipliner internal yang diambil oleh manajemen perusahaan ke penuntutan oleh otoritas lokal, negara bagian, atau federal.

Namun, secara umum biasanya ada beberapa tugas, langkah, atau tindakan umum yang akan menjadi tahap akhir investigasi, yang akan menghabiskan waktu penyelidik dan menjauhkannya dari ilmu sebenarnya dalam melakukan analisis dan investigasi forensik dunia maya. Tugas dan tindakan ini, bagaimanapun, sama pentingnya untuk mengakhiri penyelidikan dengan sukses seperti langkah-langkah yang diambil oleh penyelidik pada tahap awal penyelidikan.

Gambar 12.4 meringkas langkah-langkah ini dan kami membahasnya secara singkat di sini. Pembaca harus menyadari, bagaimanapun, bahwa tidak setiap tindakan atau tugas yang dibahas di sini dilakukan di setiap organisasi atau bahkan secara kohesif dalam satu langkah seperti yang telah kami sajikan di sini. Banyak organisasi dapat memilih untuk memisahkan beberapa tugas yang ditunjukkan pada Gambar 12.4 menjadi langkah-langkah independen.



Gambar 12.4 Langkah 8: Penutup dan Kesimpulan

12.3 PERAN PENYELIDIKI SEBAGAI SAKSI AHLI

Sebelum membahas secara singkat apa peran dan tanggung jawab penyelidik sebagai saksi ahli, pertama-tama kita perlu mendefinisikan, tepatnya, apa itu saksi ahli?

Menurut beberapa sumber hukum yang sebenarnya, saksi ahli dan keahlian serta kesaksiannya dapat didefinisikan sebagai:

1. Kesaksian berupa pendapat atau lainnya berdasarkan ilmu pengetahuan, teknik, atau pengetahuan khusus lainnya; saksi harus memenuhi syarat, sebagaimana ditentukan oleh pengadilan. Aturan 702, Aturan Pembuktian Federal (FRE).
2. "Tidak seperti saksi biasa, seorang ahli diizinkan untuk memberikan pendapat secara luas, termasuk pendapat yang tidak didasarkan pada pengetahuan atau pengamatan langsung." Daubert v. Merrell Dow Pharmaceuticals, Inc., 509 U.S. 579, 592 (1993).
3. Perbedaan utama lainnya: saksi ahli, tidak seperti saksi "awam", dapat menjawab pertanyaan hipotetis berdasarkan informasi yang disajikan pada atau sebelum dengar pendapat, termasuk fakta dan data yang tidak dapat diterima sebagai bukti. Asplundh Mfg.Div. v. Benton Harbor Eng'g, 57 F.3d 1190, 1202 n.16 (3d Cir. 1995); Peraturan 703, FRE.

Seseorang juga harus mempertimbangkan jenis kesaksian yang akan diberikan oleh penyelidik forensik dunia maya (yaitu, kesaksian fakta, yang bersaksi semata-mata untuk fakta-fakta dalam pengetahuan pribadi saksi, atau kesaksian opini awam, yang merupakan opini atau kesimpulan yang didasarkan pada persepsi saksi sendiri dan tidak didasarkan pada ilmu pengetahuan, teknis, atau pengetahuan khusus lainnya dalam lingkup Peraturan 702 [Peraturan 701, FRE]).

Bagaimanapun, mengambil peran sebagai saksi ahli memiliki tanggung jawab dan risiko yang signifikan.

Peran Saksi Ahli

Peran utama saksi ahli adalah memberikan evaluasi dan pendapat yang andal kepada pencari fakta yang didasarkan pada pengetahuan ilmiah, teknis, atau pengetahuan khusus lainnya. Sebagaimana dijelaskan dalam Peraturan Federal tentang Pembuktian 702 (Kesaksian Para Ahli), keterangan saksi ahli dimaksudkan untuk sangat berguna dalam membantu pengadilan memahami bukti dan fakta yang dipermasalahkan. Untuk memenuhi peran tersebut, saksi ahli harus menyampaikan evaluasi yang sesuai dengan pengetahuan dan pengalaman yang diterima. Pada akhirnya, tergantung pada pencari fakta untuk menentukan keandalan, relevansi, dan bobot kesaksian ahli.

Hubungan pengacara-ahli mencakup hierarki dan organisasi yang umumnya dipahami dengan baik oleh semua yang terlibat: pengacara bertanggung jawab atas keseluruhan proses dan memberikan arahan dan strategi; ahli mengevaluasi data kasus dan informasi dan menyajikan temuan, kesimpulan, dan pendapat, selain memberikan bimbingan teknis sepanjang jalan. Proses ini sederhana dalam penampilan saja. Litigasi yang melibatkan partisipasi saksi ahli seringkali merupakan proses yang sangat rumit, panjang, dan membosankan dengan banyak variabel yang dapat memengaruhi hasil akhir suatu kasus. Beberapa variabel bersifat manusiawi dan seringkali sulit atau bahkan tidak mungkin untuk dikendalikan atau diprediksi.

Langkah-Langkah Deposisi Sebelum Presentasi Pengadilan Formal

Untuk memerintah dalam "ilmu yang buruk," pengadilan diberdayakan untuk bertindak sebagai penjaga gerbang untuk mengecualikan kesaksian ahli yang tidak dapat diandalkan atau tidak. Perubahan besar dalam undang-undang ini memiliki beberapa efek yang diinginkan tetapi tampaknya tidak sepenuhnya menyelesaikan masalah. Dalam praktiknya, memang sangat sulit bagi pengadilan untuk menjalankan peran gatekeeper secara adil dan efektif, dan fungsi gatekeeper tidak diterapkan secara konsisten di semua yurisdiksi. Bagi para ahli, menjadi "Dauberted" adalah masalah serius yang dapat berdampak negatif pada karier seseorang.

Saksi ahli sering distigmatisasi secara publik sebagai dikompromikan secara etis, dianggap oleh beberapa orang sebagai tidak lebih dari "senjata sewaan". Stigma ini lahir dari kesalahpahaman dan dari sifat manusia yang tidak dapat dihindari. Konsep bahwa siapa pun yang mengenakan tarif per jam yang tinggi akan mengatakan apa saja untuk memuaskan pihak yang membayar, bersama dengan beberapa contoh pelanggaran profesional yang dipublikasikan dengan baik, memperkuat stigma ini.

Kenyataannya, saksi ahli yang bertahan lama harus menunjukkan perilaku profesional dan etis yang kuat. Saksi ahli yang khas dapat bekerja untuk penggugat dalam satu kasus dan tergugat dalam kasus lain. Saksi ahli tidak mampu untuk mengajukan klaim dan tuduhan yang keliru atau berlebihan yang bertentangan atau melampaui apa yang dapat didukung oleh fakta-fakta dalam bukti dan dengan metodologi ilmiah atau teknis yang sehat.

Melakukan hal itu pada akhirnya dapat terbukti tidak efektif untuk kasus tersebut dan memicu banding. Diekspos sebagai tidak etis juga bisa berarti akhir dari karir profesional yang memuaskan. Pendapat pengadilan dan transkrip deposisi dan kesaksian persidangan merupakan catatan publik. Catatan itu berfungsi sebagai alat kontrol kualitas yang efektif yang dapat dikonsultasikan dengan pengacara dan pencari fakta. Untuk berhasil sebagai saksi ahli, kredibilitas dan ketelitian harus melengkapi pendidikan dan pengalaman.

Tidak ada resep universal untuk memberikan kesaksian ahli yang efektif atau hubungan profesional yang sukses antara pengacara dan ahli. Namun, ada beberapa pertimbangan yang sering penting.

Pertama, untuk kesaksian ahli, penting untuk:

- Persiapkan secara ekstensif dan ulangi kesaksian dengan pengacara pengadilan.
- Menyampaikan pendapat secara sederhana dengan bahasa yang mudah dipahami dan demonstratif, terlepas dari betapa rumitnya dasar-dasar opini mungkin.
- Sampaikan kesaksian kepada pencari fakta.
- Batasi jawaban dan penjelasan pada pertanyaan terbuka.
- Dalam ujian silang, jawablah hanya pertanyaan terbuka, gunakan singkatan jawaban; menghindari terseret ke dalam perdebatan.
- Simpan penjelasan terperinci untuk pengalihan untuk mengatasi kritik yang mungkin diajukan dalam pemeriksaan silang.
- Tunjukkan rasa hormat kepada pengadilan dan amati kesopanan yang tepat.

Kedua, untuk hubungan ahli-pengacara yang sukses, penting bagi ahli untuk:

- Menjadi ahli, bukan pengacara.
- Beradaptasi dengan kepribadian dan modus operandi pengacara tanpa kompromi dalam kinerja dan kualitas kerja.
- Menyerahkan produk kerja tepat waktu.
- Memberi tahu pengacara tentang kemajuan, kemunduran, dan kesulitan lainnya.
- Melacak anggaran karena dapat menjadi faktor pembatas.[17]

Mengamankan Hak Akses ke Catatan Kasus dan Data Kasus Terkait

Semua catatan kasus dan data kasus yang dikumpulkan dan dikembangkan oleh penyelidik harus dianggap rahasia pertama dan hak milik kedua, baik organisasi yang mengizinkan penyelidikan atau entitas (lokal, negara bagian, federal) yang melakukan penyelidikan. Dengan demikian, apakah sebagai penyelidik Anda adalah karyawan atau pihak ketiga yang independen, penting untuk memastikan bahwa Anda telah mengatasi masalah mendapatkan akses hukum ke semua catatan kasus Anda, terutama jika akan ada selang waktu. waktu antara melakukan investigasi tingkat lapangan dan presentasi aktual temuan Anda, baik secara internal kepada manajemen atau dalam deposisi atau pengadilan. Sangat tidak mungkin bahwa setelah menyelesaikan penyelidikan Anda, akan ada presentasi langsung dari

temuan Anda; akan ada jeda waktu alami di antara peristiwa-peristiwa ini. Jeda waktu mungkin tergantung pada tindakan investigasi lebih lanjut yang harus diambil oleh entitas eksternal lainnya, penundaan untuk bertindak sebagai bagian dari manajemen, penumpukan kasus yang mencegah tanggapan segera dari penasihat hukum, atau berkas pengadilan penuh yang menunda kehadiran Anda sebagai seorang saksi ahli di pengadilan.

Pertimbangan ini sangat penting, terutama karena waktu berlalu dan ingatan cenderung kabur dan memudar, dan detail yang tepat serta spesifikasi kritis hanya dapat diingat dengan meninjau catatan kasus seseorang. Ini merupakan pertimbangan yang sangat penting bagi penyelidik yang merupakan karyawan internal perusahaan. Bagaimana Anda mendapatkan akses ke catatan kasus Anda jika Anda tidak lagi dipekerjakan oleh organisasi yang sama tempat Anda melakukan penyelidikan? Jika, seperti kebanyakan penyelidik, Anda mengerjakan banyak kasus secara bersamaan, Anda mungkin memiliki banyak catatan kasus yang mungkin perlu Anda akses.

Sebagai pihak ketiga yang independen, apakah Anda, melalui kontrak, menetapkan bahwa Anda dapat menyimpan salinan semua catatan kasus yang berkaitan dengan penyelidikan yang dilakukan untuk klien? Apakah Anda memiliki kemampuan untuk melindungi dokumen-dokumen ini saat Anda memilikinya? Catatan kasus ini dan informasi yang terkandung di dalamnya dapat dianggap rahasia dan penghapusan materi tersebut dari organisasi atau situs klien tanpa persetujuan tertulis sebelumnya dapat melanggar hukum atau bahkan merupakan pelanggaran kontrak.

Dalam situasi apa pun, sebagai penyelidik yang dikontrak oleh karyawan atau pihak ketiga, kemampuan untuk mengingat informasi teknis penting dengan jelas, ringkas, dan akurat tanpa ragu sering kali bergantung pada akses ke catatan kasus seseorang. Memastikan akses semacam itu merupakan langkah penting dalam praktik cerdas investigasi penyelidik secara keseluruhan.

Bergantung pada sifat dan status penyelidikan dan undang-undang hukum, data yang disita, dikumpulkan, dan diperiksa oleh penyelidik dapat dikembalikan kepada pemilik data, dipegang oleh departemen atau badan penyelidik sampai kasus tersebut dibawa ke pengadilan, disimpan oleh penyidik. penyelidik forensik dunia maya, atau dibuang dan dimusnahkan sesuai instruksi dari penasihat hukum dan kebijakan perusahaan.

Menutup Berkas Kasus

Langkah ini tentunya akan berpedoman pada departemen, instansi, atau kebijakan perusahaan. Kasus dapat ditutup, misalnya, berdasarkan keputusan akhir di pengadilan, keputusan pihak manajemen untuk tidak mengambil tindakan tambahan atau lebih lanjut, atau saran dari penasehat hukum mengenai kelayakan dan kelayakan bukti yang dikumpulkan untuk menahan presentasi yang sukses di persidangan.

Setelah kasus ditutup dan berkas, catatan, catatan, laporan, dan transkrip penyelidik telah diserahkan kepada manajer kasus, penasihat hukum, atau perwakilan manajemen, tanggung jawab penyelidik kini telah selesai. Tanggung jawab untuk pengamanan, penanganan, transportasi, pengarsipan, dan disposisi file-file ini sekarang berada di tangan penerima.

Apakah Anda seorang karyawan atau penyelidik kontrak pihak ketiga, Anda memiliki satu tugas terakhir yang harus dilakukan. Anda harus mendapatkan persetujuan dari penerima:

1. Tanda terima, merinci semua berkas, catatan, catatan, laporan, dan transkrip yang diserahkan oleh penyidik. Log tanda terima ini, selain itu, harus menunjukkan nomor kasus, tanda tangan penerima, dan tanggal.
2. Apabila bukan merupakan dokumen resmi internal, maka surat dari penerima yang memuat nomor perkara, tanda tangan penerima, dan tanggal yang menyatakan bahwa perkara telah ditutup secara resmi dan tidak ada hal-hal atau hal-hal yang belum diselesaikan yang memerlukan tindakan lebih lanjut dari pihak penyelidik.

Penilaian Kendali Mutu Pasca Investigasi dan "Pelajaran yang Dipetik"

Praktik cerdas investigasi menjamin bahwa keseluruhan proses investigasi harus mengandung kemampuan untuk terus melihat ke belakang pada proses, untuk tujuan evaluatif dan penilaian, dengan tujuan keseluruhan perbaikan proses yang berkelanjutan. Langkah terakhir dari proses investigasi memerlukan, pada tingkat yang ditentukan oleh kebijakan organisasi atau diamanatkan oleh undang-undang, suatu pengendalian mutu dan aktivitas penilaian.

Pertama, beberapa definisi sangat membantu:

- Kegiatan penjaminan mutu mencakup perencanaan kegiatan manajemen mutu dan verifikasi bahwa kegiatan tersebut telah dilaksanakan.
- Aktivitas kendali mutu mencakup implementasi aktual dari aktivitas manajemen mutu dan dokumentasinya.

Proses penilaian/kontrol kualitas dapat atau tidak dapat dilakukan oleh penyelidik forensik dunia maya, dapat dilakukan secara mandiri atau melalui proses peninjauan tim, dan dapat dibentuk sebelumnya secara internal atau melalui pihak ketiga yang dikontrak secara independen. Terlepas dari pendekatannya, kinerja proses peninjauan semacam itu sangat penting dalam memastikan bahwa kebijakan dan prosedur investigasi diikuti sesuai dengan pedoman yang ditetapkan oleh preseden hukum, kebijakan manajemen, dan praktik investigasi forensik dunia maya yang diterima secara umum dan uji tuntas profesional.

Proses penilaian/kontrol kualitas yang layak harus menentukan organisasi, prosedur, dokumentasi, pengujian, dan metode yang akan digunakan untuk memberikan kualitas, sesuai dengan kebijakan investigasi, pedoman, dan metodologi investigasi forensik cyber yang terbukti.

Program investigasi forensik pasca cyber yang khas harus membahas, namun tidak terbatas pada, elemen-elemen berikut:

- Tanggung jawab manajemen
- Tanggung jawab dan kompetensi penyelidik
- Proses dan prosedur investigasi
- Identifikasi bukti, perolehan, ketertelusuran, retensi, dan pembuangan
- Kontrol bukti
- Inspeksi, pengukuran, dan pengujian peralatan diagnostik forensik dan perangkat lunak
- Dokumentasi laporan akhir, distribusi, dan kontrol

▪ Pemeliharaan catatan penilaian
▪ Melanjutkan pelatihan penyelidik
▪ Tindakan korektif terhadap kebijakan, prosedur, dan metodologi sesuai kebutuhan
▪ Arahan untuk audit kualitas dan penilaian yang sedang berlangsung

Kemampuan untuk membuktikan proses investigasi forensik siber yang terus diperiksa dan berkualitas sangat penting dalam membangun dan mempertahankan kredibilitas berkelanjutan dari bukti digital yang diperoleh melalui proses tersebut.

12.4 RINGKASAN

Sebagai penyelidik forensik dunia maya, laporan Anda harus mengomunikasikan semua temuan secara objektif, karena dapat digunakan di pengadilan dan Anda dapat dipanggil sebagai saksi (ahli atau bukan), untuk membelanya. Lebih baik mendokumentasikan secara berlebihan daripada tidak; dengan demikian, selalu berbuat salah di sisi hati-hati dan overdocument. Bukan penghinaan bagi penyelidik forensik dunia maya untuk disebut teliti, hati-hati, teliti, atau metodis; ini terutama benar ketika Anda dipanggil ke kursi saksi.

Memutuskan apa yang harus dilakukan dengan bukti yang dikumpulkan setelah penyelidikan ditutup adalah masalah kritis yang harus ditangani melalui kebijakan yang ditetapkan dengan baik, ditetapkan dengan baik, dan diterapkan. Penilaian diri berkelanjutan terhadap personel, proses, prosedur, kebijakan, perangkat keras, perangkat lunak, dan metodologi yang digunakan untuk melakukan setiap investigasi forensik dunia maya sangat penting dalam membangun kredibilitas bukti yang diajukan oleh penyelidik forensik dunia maya, melalui proses investigasi.

BAB 13
RINGKASAN PROSES FORENSIK CYBER

Kebutuhan Komunikasi dan berbagi informasi terus menjadi kekuatan pendorong teknologi. Upaya ini dimulai dengan fajar manusia dan berlanjut hari ini. Dari lukisan gua awal hingga tablet batu, hingga mesin cetak, hingga data yang disimpan secara elektronik dan Internet, kebutuhan dan keinginan kita untuk berbagi informasi terus tumbuh secara geometris.

Kemampuan berkomunikasi secara elektronik telah menyumbang banyak kemajuan yang kita miliki dalam masyarakat saat ini, bersama dengan banyak kemudahan. Namun, setiap kali ada yang baik, yang buruk tidak jauh di belakang; seperti halnya segala sesuatu dalam hidup ada keseimbangan yang baik antara yang baik dan yang buruk. Tidak terkecuali dengan data yang disimpan secara elektronik. Kejahatan akan terus terjadi terlepas dari kemajuan teknologi yang digunakan oleh mereka yang melakukan kejahatan dan mereka yang bersumpah untuk menegakkan hukum. Perbedaannya adalah proses terjadinya kejahatan dan bagaimana kejahatan itu harus diselidiki.

Buku ini membahas proses di mana data berasal, disimpan, dipindahkan, dimanipulasi, dan dianalisis untuk menilai relevansinya sebagai bukti. Seperti semua mata pelajaran harus ada awal yang logis dan demikian pula, kesimpulan yang logis. Perjalanan kami dimulai di Bab 1 dan akar dari semua informasi yang dikomunikasikan secara elektronik, representasi data biner.

13.1 BINER

Untuk menyelidiki data elektronik atau komunikasi komputasi dengan benar, pertama-tama perlu dipahami bagaimana kita, sebagai spesies, berupaya mengkodifikasi kemampuan kita untuk berkomunikasi secara elektronik di dunia dengan hanya dua kemungkinan keadaan, dunia keberadaan biner.

Biner adalah skema pengkodean Basis 2 yang berfungsi baik dengan paradigma dua keadaan seperti elektronik. Dengan elektronik ada dua kemungkinan keadaan, hidup dan mati. Ini adalah dasar untuk semua komunikasi elektronik. Ketika merangkai hidup/mati ini bersama-sama, komunikasi yang kompleks dapat dicapai, tidak hanya untuk merepresentasikan pola komunikasi manusia yang paling dasar, tetapi juga pola abjad dan numerik yang kompleks, yang pada akhirnya memungkinkan orang untuk mewakili seluruh bahasa.

Representasi pola bahasa yang kompleks untuk komunikasi digital dimulai dengan blok bangunan utama, bit, yang diwakili oleh satu (1) atau nol (0). Cukup mengatur dan mengelompokkan satu dan nol bersama memungkinkan untuk semua representasi data elektronik, dari dokumen teks sederhana hingga film definisi tinggi.

Menetapkan metode untuk memasangkan karakter alfabetis dengan persamaan biner karakter menghasilkan kode karakter, yang telah berkembang menjadi rangkaian karakter

yang lebih kompleks, lebih lanjut memungkinkan kita untuk tidak hanya memperluas kemampuan kita untuk merepresentasikan rentang karakter yang lebih besar tetapi juga mengontrol bagaimana komputer menyimpan, memanipulasi, dan mengirimkan data.

Representasi biner dari angka dan karakter diperlukan saat bekerja di dunia yang terbatas hanya pada dua keadaan deskripsi atau keberadaan (misalnya listrik atau magnet). Untungnya bagi kita, dunia manusia kita lebih kuat, lebih berwarna, dan ada di banyak negara bagian, jauh melampaui kehidupan biner. Ini juga lebih sulit dan memakan waktu jika, sebagai manusia, kita diharuskan untuk melakukan semua perhitungan, komunikasi, dan sebagainya, dengan angka atau huruf yang diwakili oleh grup dan pasangan 1 dan 0 (mis., 01001000 01100101 01101100 01101100 01101111 alih-alih "Halo").

13.2 BINER—DECIMAL—ASCII

Mengubah bilangan biner menjadi desimal yang setara sangat penting untuk mendapatkan pemahaman yang lebih mendalam tentang bagaimana data disimpan, dipindahkan, dimanipulasi, dan diproses dan bagaimana perlakuan data ini sangat penting untuk pemahaman yang lebih baik tentang forensik dunia maya.

Sebuah komputer hanya memproses bit-bit yang tersimpan dalam biner, ia tidak dapat mengenali atau memproses karakter “&” dalam bentuk asalnya; kita manusia, sebaliknya, tidak memproses biner dengan mudah. Jalan tengahnya adalah dengan nilai desimal. Kita dapat lebih mudah dan mudah memahami informasi yang sama, diwakili dan disajikan dalam bentuk desimal.

Nilai desimal (10 karakter desimal unik: 0, 1, 2, 3, 4, 5, 6, 7, 8, dan 9) adalah perhitungan matematis dari biner, bukan representasi visual dari biner.

Kunci untuk mengonversi nilai biner menjadi padanan desimalnya adalah keberadaan (atau ketiadaan) "arus" yang diwakili oleh nilai biner dari sakelar "0" atau "1" atau karakter biner. Jika ada nilai biner di placeholder, nilai diaktifkan, diwakili oleh nilai satu. Jika tidak ada nilai biner yang menempati placeholder, maka nilainya dimatikan, yang diwakili oleh nilai nol. Jika sakelar biner (atau nilai) AKTIF ("1"), maka nilai desimal AKTIF, artinya ditambahkan atau dihitung saat menentukan nilai desimal total. Jika sakelar biner MATI ("0"), maka nilai desimal tidak dihitung atau ditambahkan saat menentukan total ekuivalen desimal. Mari kita ambil contoh nilai biner 01011000, menggunakan informasi pada Gambar 13.1.

Kekuatan	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
Setara Desimal	128	64	32	16	8	4	2	1	
		↓		↓	↓				
Nilai Biner	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**	
		↓		↓	↓				**Menambahkan nilai desimal**
Nilai Desimal	0	**64**	0	**16**	**8**	0	0	0	0+**64**+0+**16**+**8**+0+0+0=**88**

Gambar 13.1 Konversi Biner ke Desimal

Telusuri bilangan biner dan jika bilangan binernya adalah 1, turunkan pangkat dua dan tuliskan di kotak yang sesuai pada garis nilai desimal. Jika bilangan binernya adalah 0, masukkan 0 ke dalam kotak. Ubah bilangan biner menjadi desimal dengan menjumlahkan nilai desimal yang Anda masukkan ke dalam setiap kotak. Jumlah angka adalah ekuivalen desimal dari angka biner. Biner 01011000 sama dengan nilai desimal 88. Mengapa repot-repot mengonversi biner ke desimal? Prosesor komputer bekerja dengan perhitungan matematis, bukan huruf, simbol, dan kata, namun manusia berkomunikasi melalui huruf dan kata. Dengan demikian, komputer membutuhkan cara untuk merepresentasikan simbol manusia secara matematis. Nilai biner dapat dihitung secara matematis menjadi nilai desimal, dan nilai desimal dapat ditetapkan ke nilai ASCII (simbol manusia). (Lihat Tabel 13.1.)

TABEL 13.1 Cuplikan Tabel ASCII

Biner	Desimal	Simbol ASCII	Keterangan
00110000	48	0	Nol
00110001	49	1	Satu
00110010	50	2	Dua
00110011	51	3	Tiga
00110100	52	4	Empat
00110101	53	5	Lima
00110110	54	6	Enam
00110111	55	7	Tujuh
00111000	56	8	Delapan
00111001	57	9	Sembilan

Nilai desimal dirujuk ke nilai yang sesuai dalam bagan karakter (ASCII atau UniCode) oleh Sistem Operasi (OS) dan/atau perangkat lunak yang digunakan. Mengonversi biner ke desimal itu mudah ketika nilai biner yang akan dikonversi kecil, tetapi ketika nilai biner bertambah besar, angkanya bisa menjadi agak besar dan membosankan.

Misalnya, asumsikan nilai biner 010110000101100101011010. Nilai ini mungkin tampak menakutkan, tetapi hanya setara dengan 3 byte atau 24 bit. Jika kita mengonversi string biner ini ke nilai desimalnya yang setara dengan mengaktifkan nilai posisi "on" yang diwakili oleh 1s dan membiarkan "off" posisi tersebut nilai diwakili oleh 0s, rangkaian angka kita akan terlihat seperti ini:

010110000101100101011010

mati + 4194304 + mati + 1048576 + 524288 + mati + mati + mati + mati + 16384 + mati + 4096 + 2048 + mati + mati + 256 + mati + 64 + mati + 16 + 8 + mati + 2 + mati

Ketika akhirnya dijumlahkan, rangkaian nilai biner ini akan menghasilkan hasil 5.790.042. Proses penguraian nilai biner menjadi padanan desimalnya bisa menjadi sangat membosankan, memakan waktu, dan sangat mahal, terutama jika string nilai biner lebih dari tiga byte. Bayangkan mengonversi video definisi tinggi!

13.3 DATA VERSUS KODE

Sebuah dokumen atau file lain memiliki apa yang kadang-kadang disebut sebagai header atau “kode” yang merupakan data tambahan yang ditempatkan di awal blok data yang disimpan atau dikirimkan. Dalam transmisi data, data yang mengikuti header disebut body. Header, pada dasarnya, mengikat blok data yang mengikuti header (badan) ke perangkat lunak yang diperlukan untuk membukanya atau mengaksesnya.

Misalnya, jika Anda membuat dokumen menggunakan Microsoft (MS) Word, dokumen tersebut tidak dapat dibuka menggunakan pembaca/aplikasi Adobe Acrobat. Ini karena ada kode yang disematkan di dalam dokumen apa pun yang dibuat menggunakan MS Word, yang memberi tahu sistem operasi bahwa hanya MS Word (atau perangkat lunak lain yang kompatibel) yang diperlukan untuk membuka dokumen.

Jika kode yang mengikat dokumen ke perangkat lunak asli entah bagaimana ditimpa atau "dihapus", perangkat lunak tidak akan dapat menyusun kembali dokumen ke dalam format aslinya atau ke dalam format yang dapat dibaca oleh pengguna, sehingga menyebabkan dokumen menjadi tidak dapat diakses dan dibaca oleh pengguna.

Beberapa dari data ini, seperti teks yang memberatkan (kemunculan "XYZ" dalam kasus yang dirujuk di seluruh buku, misalnya), mungkin masih berada di dokumen, di disk, atau di dalam hard drive. Untuk penyelidik forensik dunia maya untuk mencari kata kunci yang terkandung dalam data yang disita dari seluruh hard drive (atau bahkan dari data yang dipersempit ke folder tertentu atau gambar tertentu di dalam hard drive pengguna), sebaiknya gunakan HEX untuk melakukannya. tugas yang sangat besar.

13.4 HEX

Hexadecimal, atau singkatnya HEX, benar-benar merupakan representasi nilai-nilai biner yang ramah manusia. Melihat data sebagai representasi (atau nilai) HEX memungkinkan penyelidik forensik dunia maya melampaui aplikasi atau file. Ini memungkinkan untuk melihat semua data yang terkandung dalam file termasuk sisa-sisa file lama atau bahkan dihapus. Penting untuk dipahami bahwa tidak semua nilai biner dapat diubah menjadi ASCII yang dapat dibaca. ASCII adalah kode, berdasarkan urutan abjad bahasa Inggris, dan tidak semua data yang ada di dalam komputer harus berbasis teks (ASCII). Ada banyak program atau aplikasi perangkat lunak yang ditulis dalam kode pemrograman yang tidak berbasis ASCII.

Kode pemrograman ini tidak dimaksudkan untuk dilihat di ASCII, ini dimaksudkan untuk menjalankan suatu fungsi. Ingatlah dari diskusi kita sebelumnya bahwa semua fungsi komputer didasarkan pada matematika, bukan bahasa Inggris (atau bahasa Prancis, China, Slavia, Yunani, Arab, atau lainnya); Oleh karena itu, kode harus didasarkan pada prinsip matematika, bukan prinsip tata bahasa.

13.5 DARI DATA MENTAH KE FILE

Ada ratusan format data yang berbeda (database, pengolah kata, spreadsheet, gambar, video, dll.). Ada juga format untuk program yang dapat dieksekusi (.exe, .bat, .dll) pada platform yang berbeda (Windows, Mac, Linux, Unix, dll.). Setiap format menentukan bagaimana urutan bit dan byte ditata, dengan ASCII menjadi salah satu yang paling mudah

bagi manusia untuk diuraikan atau dibaca. File teks hanyalah file yang menyimpan teks apa pun, dalam format seperti ASCII atau UTF-8, dengan sedikit karakter kontrol jika ada. Ada berbagai jenis file digital yang berisi informasi pemformatan khusus yang memungkinkan akses, penyimpanan, atau "manipulasi" file. “Manipulasi” ini dapat terjadi melalui sistem operasi itu sendiri, atau dapat terjadi melalui program “induk” yang diinstal pada sistem operasi.

Program induk adalah program yang digunakan untuk membuat, mengeksekusi, atau mengakses file. Dalam kebanyakan kasus, file akan berisi data dan tanda tangan file, yang darinya perangkat lunak induknya (atau sistem operasi) akan dapat mengidentifikasi dan menangani operasinya. Informasi tanda tangan file terkandung dalam apa yang terkadang disebut sebagai "header file". Data yang terkandung dalam header file tidak terlihat oleh pengguna biasa, namun sangat penting agar file berfungsi seperti yang dirancang. Data inilah yang terkandung dalam header file yang digunakan untuk mengidentifikasi format file.

Nilai HEX terlihat ketika metode untuk mengekstrak data yang dapat dibaca dari file mungkin tidak lagi layak, terjadi misalnya ketika informasi header hilang atau rusak. Meskipun file tidak dapat diidentifikasi dan tidak dapat dibuka oleh perangkat lunak asli atau yang kompatibel, penyelidik forensik dunia maya dapat mencari padanan biner dari beberapa representasi ASCII di seluruh hard drive.

Penyelidik akan menemukan nilai ini terlepas dari tanda tangan file yang dimodifikasi atau hilang. Seperti yang telah dibahas sebelumnya, berkali-kali dalam operasi normal sehari-hari dan pemrosesan file, file yang dihapus dan metadata yang terkait akan ditimpa sebagian, mungkin kehilangan seluruh tanda tangan file atau informasi pemformatan penting lainnya dan bahkan beberapa teks. Namun, jika nilai biner yang mewakili sepotong bukti (mis., "XYZ") tetap berada dalam sisa-sisa file, maka dapat ditemukan.

13.6 MENGAKSES FILE

Sebagian besar file perlu dipasang oleh sistem operasi (atau beberapa perangkat lunak) agar dapat diakses dalam penggunaan normal sehari-hari. Agar hal ini terjadi, sistem operasi perlu melakukan boot sehingga dapat mengidentifikasi struktur file dan lokasi file untuk menyajikan data dengan cara yang dapat dibaca.

Proses booting ini penting, karena merupakan proses pemasangan bukti yang akan diselidiki oleh penyidik. Saat mengakses informasi pada suatu sistem, pemasangan sistem file sangat penting. Pentingnya Master Boot Record (MBR) dan isinya, seperti tabel partisi (PT), adalah semua bit informasi relevan yang dapat memiliki pengaruh penting dalam penyelidikan. Pemahaman yang kuat tentang proses boot diperlukan jika, tanpa alasan lain, mengetahui kapan bukti diubah dan dengan demikian menghindari kontaminasi bukti dengan pencitraan. Penyelidik forensik dunia maya, seperti penyelidik mana pun, terkadang akan bertanggung jawab untuk mengumpulkan dan menangkap bukti.

Data dapat ditulis ke hard drive (misalnya, bukti potensial) selama proses boot, mengubah bukti. Mengetahui kapan dan bagaimana data diubah pada bukti (hard drive atau lainnya) tidak hanya penting saat menyelidiki bukti, tetapi juga penting saat memperoleh bukti. Selama proses boot data sistem file utama (atau partisi), dalam banyak kasus, ditulis ke

hard drive, sehingga tanggal diubah dan file ditulis dan diubah. Sangat penting untuk penyelidikan yang baik untuk tidak mengubah bukti yang telah dipercayakan kepada Anda untuk dicitrakan dengan cara yang sehat secara forensik. Mem-boot komputer, dengan cara yang tidak terkendali, dapat sangat mencemari integritas data yang terkandung dalam bukti (hard drive). Ini akan dianalogikan dengan seorang detektif pembunuhan yang menginjak-injak cipratan darah di TKP. Kalaupun sang detektif bisa menghilangkan jejak kakinya, setidaknya kualitas pekerjaan dan kompetensinya akan dipertanyakan.

13.7 ENDIANNESS

Dalam forensik dunia maya, bagaimana data disimpan pada drive merupakan informasi yang sangat penting, karena seringkali, penyelidik forensik dunia maya harus melihat data mentah (melalui editor HEX) untuk kemungkinan bukti; mengetahui bagaimana informasi ditulis ke disk, bagaimana data direpresentasikan dan disajikan secara fisik dan logis, sangatlah penting.

Memahami konsep endianness diperlukan untuk memahami sepenuhnya bagaimana sistem berbasis matematika menangani atau menginterpretasikan data, seperti apakah bilangan bulat direpresentasikan dari kiri ke kanan atau kanan ke kiri. Tidak semua data biner diperlakukan sama. Cara menangani biner (HEX, dalam pandangan kami) semuanya tergantung pada arsitektur sistem, kodenya. Saat sistem mem-boot, ia akan menemukan kode yang akan memerintahkannya untuk menjalankan set instruksi.

Secara umum, dalam komputasi, endianness hadir dalam dua rasa: big endian dan little endian. Di big endian, unit (atau byte) paling signifikan dari bidang data diurutkan terlebih dahulu atau dibenarkan kiri. Namun, dengan little endian, unit (atau byte) yang paling tidak signifikan dari bidang data diurutkan terlebih dahulu dengan byte paling signifikan di sebelah kanan (yaitu, dibenarkan kanan).

Endianness menjelaskan bagaimana data multi-byte direpresentasikan oleh sistem komputer dan ditentukan oleh arsitektur CPU dari sistem tersebut. Sayangnya, tidak semua sistem komputer didesain dengan arsitektur Endian yang sama. Perbedaan dalam arsitektur Endian adalah masalah ketika perangkat lunak atau data dibagi antara sistem komputer. Analisis sistem komputer dan antarmukanya akan menentukan persyaratan implementasi perangkat lunak Endian.

13.8 PARTISI

Ada perbedaan halus antara volume dan partisi, dan terkadang garis di antara keduanya bisa kabur. Volume ada di tingkat OS logis, dan partisi ada di tingkat fisik, khusus media. Terkadang ada korespondensi satu-ke-satu, tetapi tidak selalu. Partisi adalah kumpulan (secara fisik) sektor berurutan dan volume adalah kumpulan (secara logis) sektor yang dapat dialamatkan. Di sinilah letak perbedaannya—data yang terkandung dalam sebuah volume mungkin tampak berurutan, tetapi hanya secara logis. Partisi adalah area hard drive yang ditentukan oleh entri di tabel partisi MBR, dan dikenali di seluruh sistem. Partisi ditafsirkan oleh kode yang terkandung dalam sektor yang sama, MBR, dan partisi biasanya

subdivisi. Seperti namanya itu adalah proses memecah sesuatu yang lebih besar menjadi potongan-potongan kecil.

Volume adalah area yang ditentukan atau ditafsirkan oleh sistem operasi. Volume dikenali oleh sistem operasi dan akan memiliki huruf drive yang terkait dengannya. Ini sering digunakan secara sinonim dengan istilah drive atau disk. Namun, sifat logis ayat fisik dari partisi dan volume tidak selalu eksklusif satu sama lain. Perbedaan atau kesamaan terkadang menjadi kabur karena tidak diciptakan dengan gagasan tentang orang lain. Faktanya, berkali-kali mereka adalah hal yang sama. Mungkin yang paling penting, sebuah volume berisi sistem file, yang unik untuk sistem operasi dan hanya dipahami oleh sistem operasi tertentu.

13.9 SISTEM FILE

Sistem file adalah alat yang digunakan untuk menyimpan dan mengambil data di komputer. Ini adalah alat yang melacak alokasi cluster, dan memungkinkan hierarki direktori, folder, dan file. Sistem file menangani dan mengelola semua cluster yang terkandung dalam volume.

Sistem file biasanya ditentukan selama pembuatan partisi; pada titik inilah partisi "menjadi" volume. Sistem file menentukan bagaimana dan di mana file ditempatkan pada hard drive, dengan tujuan mencoba mengoptimalkan kecepatan pengambilan data. Kami mungkin tahu di mana dokumen itu berada secara logis dalam struktur folder, tetapi kami tidak menyadari (dan memang benar), mengenai bit spesifik mana pada hard drive yang dialokasikan untuk dokumen individual ini. Ini bukanlah sesuatu yang perlu diperhatikan oleh pengguna akhir; namun, sistem file komputer harus mengetahuinya, jika tidak, saat kita mengklik ikon dokumen Word, tidak akan terjadi apa-apa.

Berbagai sistem pengarsipan dan komponennya mungkin memiliki nama yang berbeda dan penempatan fisiknya pada drive dapat bervariasi, tetapi secara fungsional semua sistem file memerlukan bagian yang serupa—bagian yang mengidentifikasinya, bagian yang mengidentifikasi datanya, dan bagian yang berisi data itu sendiri.

13.10 WAKTU

Dalam investigasi forensik dunia maya mengetahui waktu yang tepat sangatlah penting. Memahami garis waktu peristiwa sangat penting dalam memahami kapan peristiwa terjadi sehubungan dengan semua peristiwa lainnya. Ketidakakuratan waktu sangat luas dan luas. Ketidakakuratan dapat berupa ketidakakuratan seluruh sistem hingga server NTP, atau spesifik sistem karena kemiringan jam. Ketidakakuratan dapat spesifik untuk lokasi geografis tertentu karena membingungkan zona waktu atau sistem file sistem operasi tertentu, membulatkan detik ganjil ke detik genap terdekat.

Perbedaan waktu bisa sepanjang waktu mengizinkan atau sesingkat satu detik (atau kurang), seperti ketidakakuratan waktu stempel waktu MS-DOS 32-bit. Sangat tidak mungkin bahwa ketidaktepatan waktu satu detik (seperti yang terlihat dalam stempel waktu MS-DOS 32-bit) akan sangat penting untuk menguatkan ketidakbersalahan seseorang; namun, satu kesamaan ketidakakuratan waktu adalah bahwa mereka dapat mendiskreditkan seorang ahli, terutama ahli / penyelidik forensik dunia maya!

13.11 PROSES INVESTIGASI

Instruksi baris demi baris yang tepat untuk menjalankan investigasi forensik dunia maya yang lengkap secara logis tidak mungkin untuk disajikan, karena setiap organisasi yang melakukan investigasi forensik akan memiliki pendekatan, prosedur, kebijakan, dan metode mereka sendiri—beberapa ditentukan oleh undang-undang, yang lain oleh preferensi internal dan protokol. Namun, ada Praktik Cerdas Investigasi umum, yang mungkin sesuai dengan sebagian besar jenis organisasi forensik dan sebagian besar jenis kasus.

Investigasi pornografi anak yang dijalankan oleh penegak hukum versus investigasi pencurian kekayaan intelektual yang dijalankan oleh departemen forensik perusahaan pada akhirnya dapat menemukan bukti yang diperlukan untuk menuntut yang bersalah; namun, pendekatan, langkah-langkah yang diambil, dan proses untuk mencapai tujuan tersebut mungkin sama sekali berbeda dan didukung oleh dokumentasi yang sama sekali berbeda.

Sama anehnya dengan perbedaan di antara berbagai jenis organisasi, begitu pula perbedaan di antara kasus-kasus. Praktik Cerdas Investigasi yang disajikan dalam buku ini dimaksudkan untuk cakupan yang luas dan digunakan sebagai pedoman.

Langkah 1: Kontak/Permintaan Awal

Validitas dan ruang lingkup permintaan investigasi ditetapkan. Fungsi ini dapat dilakukan oleh seseorang di luar bidang forensik dunia maya. Misalnya, hal ini dapat ditentukan oleh hakim melalui perintah pengadilan atau mungkin melalui departemen SDM dalam sebuah organisasi besar.

Langkah 2: Penanganan Bukti

Integritas bukti Harus dipertahankan selama investigasi berlangsung. Proses ini terjadi setiap kali bukti ditangani. Mempertahankan integritas bukti sangat penting, tetapi sama pentingnya juga mampu membuktikan integritas bukti di pengadilan.

Langkah 3: Perolehan Bukti

Langkah ini melibatkan perolehan citra forensik dari bukti asli. Perolehan bukti pasti termasuk dalam Langkah 2, Penanganan Bukti; namun, langkah ini lebih berfokus pada perolehan bukti versus penanganan bukti selama perolehan.

Langkah 4: Persiapan Data

Mempersiapkan dan mengidentifikasi data untuk analisis dan investigasi. Langkah ini berfokus pada "menganalisis" data untuk memastikan pencarian yang valid dan lengkap. Ini termasuk memasang file kompleks, memverifikasi jenis file, memulihkan item yang dihapus, dan hal lain yang akan menyiapkan data untuk penyelidikan akhir (Langkah 5).

Langkah 5: Investigasi

Berfokus untuk menemukan data yang cocok dengan kriteria pencarian yang ditentukan. Langkah ini cenderung sedikit lebih subyektif daripada yang lain, karena penyelidik mungkin perlu memeriksa hasil pencarian, membuang positif palsu, dan mengidentifikasi bagian penting dari bukti, yang biasanya tidak mudah disebut "bukti yang memberatkan". dokter."

Langkah 6: Pelaporan

Pelaporan adalah cara penyelidikan merinci temuannya dan mengkomunikasikannya kepada klien, manajemen, penegak hukum, dan/atau pemohon. Mengkomunikasikan temuan

sangat tergantung pada struktur organisasi. Lingkungan perusahaan mungkin memiliki persyaratan atau template pelaporan yang berbeda dengan yang ada di penegakan hukum, misalnya.

Beberapa dari langkah-langkah ini akan terjadi secara bersamaan sementara beberapa mungkin terjadi di luar "urutan". Seperti halnya nilai hash bukti, setiap kasus unik. Keunikan inilah yang membuat forensik dunia maya menjadi bidang yang menantang.

Langkah 7: Retensi dan Kurasi Bukti

Retensi dan kurasi bukti mungkin berada di bawah langkah “penanganan bukti” yang mencakup semua, namun karena masalah unik dan kerumitannya dibahas dengan sendirinya. Langkah ini melibatkan penanganan pasca bukti, menyiratkan penyimpanan, pengarsipan, penghancuran, atau pengembalian semua bukti. Retensi bukti kemungkinan akan mencakup penanganan semua bukti yang terkait dengan investigasi, baik itu hard drive fisik, file digital, dan mungkin catatan tulisan tangan penyelidik.

Seperti halnya semua langkah, persyaratan sangat bergantung pada jenis praktik forensik (misalnya, penegakan hukum, perusahaan, pemerintah, swasta) dan dapat bervariasi dari kasus ke kasus. Akan ada berbagai persyaratan hukum, kontrak, prosedural, keuangan, dan bisnis, yang pada akhirnya akan menetapkan batasan untuk langkah penanganan bukti akhir ini.

Langkah 8: Penutup dan Kesimpulan Investigasi

Ringkasan investigasi memiliki makna yang luas dan mencakup aktivasi pasca investigasi yang terstruktur secara longgar di sekitar mempertahankan pekerjaan penyelidik forensik dunia maya.

Hal ini dapat terjadi dalam bentuk menjadi saksi ahli dalam kasus pidana, wawancara oleh SDM atau Internal Audit untuk investigasi perusahaan, atau mungkin peer review. Ini mungkin juga termasuk pemeriksaan diri internal seperti "pelajaran yang didapat" atau penilaian kendali mutu.

13.12 RINGKASAN

Profesional forensik tradisional menggunakan sidik jari, pengetikan DNA, dan analisis balistik untuk membuat kasus mereka. Penyidik forensik dunia maya mengandalkan berbagai teknologi untuk mengumpulkan, memeriksa, dan mengevaluasi data dalam upaya untuk menetapkan maksud, kesalahan, motif, cara, metode, dan kerugian, akibat kejahatan yang dilakukan oleh, dengan, atau melalui perangkat apa pun yang mampu mengakses , mengambil, memproses, dan menyimpan data elektronik. Peran penyelidik forensik dunia maya adalah dalam penemuan, pengumpulan, dan analisis data, yang mengarah pada identifikasi bukti digital. Kami berharap buku ini memberi Anda dasar yang kuat untuk membangun atau memperluas pemahaman tentang bagaimana data mentah diurai melalui proses forensik dunia maya, yang menghasilkan bukti digital.

DAFTAR PUSTAKA

"Chapter 17—Disk and File System Basics," TechNet, 2010 Microsoft Corporation.

"What Is a Hex Editor?" Hexprobe, retrieved December 10, 2009,

A. Damuri, N. Isnain, R. A. Priyatama, Y. I. Chandra and A. S. Putra, "E-Learning Proposal System in Public Secondary School Learning," International Journal of Educational Research & Social Sciences (IJERSC), vol. 2, p. 270–275, 2021

A. Hadi, A, I. Riadi, & Sunardi, Forensik Bukti Digital Pada Solid State Drive (SSD) NVMe Menggunakan Metode National nstitute Standart and Technology (NIST). Seminar Nasional Teknologi Fakultas Teknik Universitas Krisnadwipayana, Jakarta 17 Juli 2019.

A. Medikano, H. Ludiya, R. Wirawan, P. M. Akhirianto, S. Rachmawati, A. Sebayang, D. Efriyenty, R. Riko, I. Svinarky, B. J. Tama and A. . S. Putra, "Smart Transportation for Jakarta Smart City Residents," International Conference on Global Optimization and Its Applications 2021, vol. 1, no. 1, pp. 21-21, 2021.

A. Saputra, A. Fahrudin, A. S. Putra, N. Aisyah and V. Valentino, "The Effectiveness of Learning Basic Mathematics through Dice Games for 5-6 Years Old at TKIT Al-Muslim," International Journal of Educational Research & Social Sciences, vol. 2, no. 6, pp. 1698-1703, 2021.

A. Tanner and D. Dampier, "Concept mapping for digital forensic investigations," IFIP Adv. Inf. Commun. Technol., vol. 306, pp. 291–300, 2009, doi: 10.1007/978-3-642-04155-6_22.

A. Yudhana, R. Umar, and A. Ahmadi, "Digital Evidence Identification on Google Drive in Android Device Using NIST Mobile Forensic Method," vol. 6, no. 1, pp. 54–63, 2019.

B. Givan, R. Amalia, A. I. Sari, S. H. Winarno and A. S. Putra, "Effective Use of E-Money through Online Shopping in E-Commerce," International Journal of Educational Research & Social Sciences, vol. 2, no. 6, pp. 1692-1697, 2021.

Bellevue Linux Users Group (BLUG), Magic Number Definition," The Linux Information Project, August 21, 2006, retrieved January 2010, www.linfo.org/magic_number.html.

Carvey, Harlan, "Windows Forensics and Incident Recovery," Addison Wesley, 2005.

CSIC, "Net Losses : Estimating the Global Cost of Cybercrime", Washington DC, 2014

D. Brezinski and T. Killalea, "Guidelines for Evidence Collection and Archiving," RFC 3227, The Internet Society, February 2001, retrieved September 2011,

H. Sugiarto, I. Sumadikarta, M. Ryansyah, M. H. Fakhriza and A. S. Putra, "Application Design" Test Job Application" On Android OS Using The AHP Algorithm," International Journal of Educational Research & Social Sciences, vol. 2, no. 5, pp. 1173-1180, 2021.

H. W. Arman Syah Putra, ""“Intelligent Traffic Monitoring System (ITMS) for Smart City Based on IoT Monitoring”," 1st 2018 Indonesian Association for Pattern Recognition International Conference, INAPR 2018 - Proce vol, 2019.

L. Ablon, M. C. Libicki, and A. A. Golay, “Markets for Cybercrime Tools and Stolen Data: Hackers’ Bazaar”, Santa Monica USA, 2014.

M. N. Al-Azhar, Digital Forensic: Panduan Praktis investigasi Komputer. Jakarta, Indonesia: Salemba Infotek, 2012.

M. R. Clint, M. Reith, C. Carr, and G. Gunsch, “An Examination of Digital Forensic Models”, Int. J. Digit. Evid., vol. 1, no. 3, pp. 1–12, 2002.

N. K. Dewi and A. S. Putra, "Prosiding International Conference of Universitas Pekalongan," Prosiding International Conference on Education of Suryakancana 2021 (ICONNECTS 2021), pp. 321-326, 2021.

P. K. Dhamarsa, Safrizal, . S. P. Arman and Suyanto, "Perancangan Aplikasi ITBU Career Center Berbasis Website Menggunakan PHP dan MYSQL," TEKINFO UPI YAI, pp. 1-105, 2019.

P. K. Dhamarsa, Safrizal, . S. P. Arman and Suyanto, "Perancangan Aplikasi ITBU Career Center Berbasis Website Menggunakan PHP dan MYSQL," TEKINFO UPI YAI, pp. 1-105, 2019.

Preserving Digital Information Report of the Task Force on Archiving of Digital Information, commissioned by The Commission on Preservation and Access and The Research Libraries Group, May 1, 1996, retrieved September 2011,

R. Hermawan, M. T. Habibie, D. Sutrisno, A. S. Putra and N. Aisyah, "Decision Support System For The Best Employee Selection Recommendation Using Ahp (Analytic Hierarchy Process) Method," International Journal of Educational Research & Social Sciences, vol. 2, no. 5, pp. 1218-1226, 2021.

Ramadhan, Rizdqi Akbar, Prayudi, Yudi and Sugiantoro, Bambang.Implementasi Dan Analisis Forensika Digital Pada Fitur Trim Solid State Drive. 2017, TEKNOMATIKA, 9(2), 1-13.

Rochmadi, “Deteksi Bukti Digital Pada Adrive Cloud Storage Menggunakan Live Forensik,” CyberSecurity dan Forensik Digital, vol. 2, no. 2, hal. 65-68, 2019.

Rosalina, Vidila, Suhendarsah, Andri and Natsir, M. Analisis Data Recovery Menggunakan Software Forensic: Winhex And X-Ways Forensic. 2016, PROSISKO, 3(1), 51-55.

S. Saad, R. Umar, & A. Fadlil, “Analisis Forensik Aplikasi Dropbox pada Android Menggunakan Metode NIJ pada Kasus Penyembunyian Berkas,” Jurnal Sains Komputer & informatika (J-SAKTI), vol. 4, no. 2, hal.293-299. 2020.

Searle, S. “A Brief History of Character Codes in North America, Europe, and East Asia,” TRON Web, Sakamura Laboratory, University Museum, University of Tokyo, August 6, 2004, retrieved October 2009,

Yudi Prayudi, Ahmad Ashari, Tri K Priyambodo, "A Proposed Digital Forensics Business Model to Support Cybercrime Investigation in Indonesia", IJCNIS, vol.7, no.11, pp.1-8, 2015.DOI: 10.5815/ijcnis.2015.11.01

Digital Forensik

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

BIO DATA PENULIS

Penulis memiliki berbagai disiplin ilmu yang diperoleh dari Universitas Diponegoro (UNDIP) Semarang. dan dari Universitas Kristen Satya Wacana (UKSW) Salatiga. Disiplin ilmu itu antara lain teknik elektro, komputer, manajemen dan ilmu sosiologi. Penulis memiliki pengalaman kerja pada industri elektronik dan sertifikasi keahlian dalam bidang Jaringan Internet, Telekomunikasi, Artificial Intelligence, Internet Of Things (IoT), Augmented Reality (AR), Technopreneurship, Internet Marketing dan bidang pengolahan dan analisa data (komputer statistik).

Penulis adalah pendiri dari Universitas Sains dan Teknologi Komputer (Universitas STEKOM) dan juga seorang dosen yang memiliki Jabatan Fungsional Akademik Lektor Kepala (Associate Professor) yang telah menghasilkan puluhan Buku Ajar ber ISBN, HAKI dari beberapa karya cipta dan Hak Paten pada produk IPTEK. Penulis juga terlibat dalam berbagai organisasi profesi dan industri yang terkait dengan dunia usaha dan industri, khususnya dalam pengembangan sumber daya manusia yang unggul untuk memenuhi kebutuhan dunia kerja secara nyata.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :

YAYASAN PRIMA AGUS TEKNIK

JL. Majapahit No. 605 Semarang

Telp. (024) 6723456. Fax. 024-6710144

Email : penerbit_ypat@stekom.ac.id

Digital Forensik

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id