Autopsyを使って不正アクセス調査に挑戦

デジタルフォレンジック実践編

実践編ではThe Honeynet Projectの2つの問題を使って、実際にフォレンジックツールでどのような調査ができるか、どのような観点で調査するかを紹介していく。

文●Lesson1:kazamiya
Lesson2:伊原秀明(@port139)

LESSON 1 The Forensic Challengeの演習課題

Forensic Challenge Imagesよりダウンロード
http://old.honeynet.org/challenge/images.html

Lesson1では、The Forensic Challenge※で公開されたイメージを使う。10年近く前に使用された問題であるため、OSの動作など細かい点は現在とは異なるが、不正侵入調査の基礎を学ぶには良い題材である。ある時点でIDSのSnortにより不審なログを検出し、稼働状態のマシンからddでイメージを取得した、という経緯になっている。調査では、5W1Hを明らかにしていくという考え方で取り組むことになる。言いかえれば、いつ何が起きていたか、という過去の事象を掘り起こしていく。今回はAutopsyのタイムラインと検索機能を用いてどのような調査ができるか紹介したいと思う。

◉事前の準備作業

まずはAutopsy上でCaseを作成する。Case名は「Forensic Challenge」、Investigator Nameは「HJ」とする。忘れずにブラウザーのJavaScriptは無効にしておこう。また、発見した内容を随時記録するために日本語入力可能な状態にしておくとよい。タイムゾーン設定はHELPメニューのTime Zonesに記載されているリストから選ぶ。今回は「CST6CDT」が正しい設定である。調査対象となるddイメージは6つあり、各イメージとマウントポイントの対応は表1のとおりである。

表に従いイメージを追加していくが、Autopsyではイメージ投入時にMD5の計算や検証ができるようになっている。各イメージのハッシュ値を入力し、一致するか検証しておこう(図1)。

表1 イメージファイルとマウントポイントの対応情報

ファイル名	マウントポイント	MD5
honeypot.hda8.dd	/	8f244a87b8d38d06603396810a91c43b
honeypot.hda1.dd	/boot	a1dd64dea2ed889e61f19bab154673ab
honeypot.hda6.dd	/home	4a20a173a82eb76546a7806ebf8a78a6
honeypot.hda5.dd	/usr	c1e1b0dc502173ff5609244e3ce8646b
honeypot.hda7.dd	/var	1b672df23d3af577975809ad4f08c49d
honeypot.hda9.dd	swap	b763a14d2c724e23ebb5354a27624f5f

図1 イメージファイルの同一性検証

Image File Details画面のData Integrity項目で「Add the following MD5 hash value for this image」を選択、既存のMD5値を入力し、「Verify hash after importing?」にチェックを入れてADDを押す

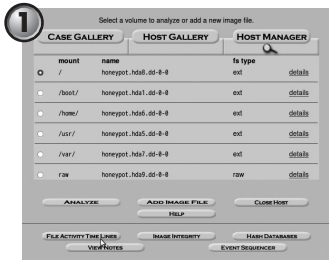
この時にMD5値を計算し、入力された値と一致すれば次の画面でIntegrity Check Passedと表示される

※ The Forensic Challenge http://old.honeynet.org/challenge/

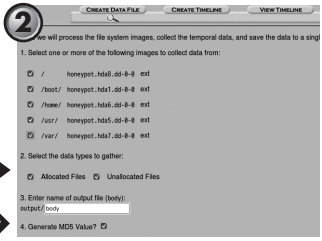
◉タイムライン解析

6つのイメージが無事登録されれば調査の環境が整ったことになる。それではタイムラインを作成しよう（図2・3）。Autopsyは MAC times 情報を元にしてタイムラインを作成することができる。MAC timeとは、ファイルシステムがファイルやディレクトリ単位で記録管理しているタイムスタンプのことであり、最終修正時刻（m）、最終アクセス時刻（a）、最終ステータス変更時刻（c）の3つを総称した意味として使われている（厳密にはファイルシステムによって異なるが、今回対象のExt系のファイルシステムではこの意味と同じである）。「最終」と付いているように、これらのタイムスタンプは、最後に更新された情報のみを記録している。タイムスタンプ情報を集め時系列に並べて、手がかりを探す方法がタイムライン解析である。

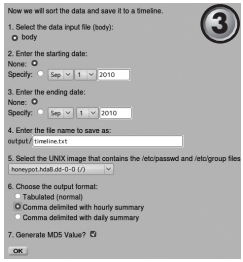
図2 タイムラインの作成



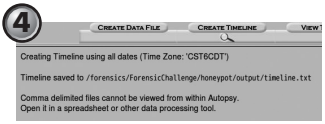
6つのイメージが追加された直後のメイン画面である。ここで「FILE ACTIVITY TIME LINES」をクリックするとタイムライン作成画面に移動する



CREATE DATA FILEで対象データの範囲を決める。基本はすべてにチェックをして対象にしておけば問題ない

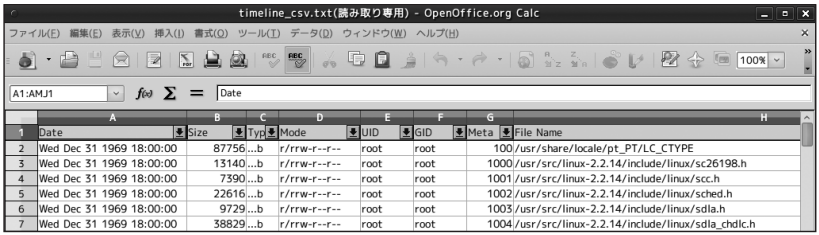


CREATE TIMELINEで対象時間の範囲などを調整する。最初の段階では指定せずにNoneとしておくのがよい。5.はUnix系OSにのみ関係がある項目で、パスワードファイルのあるパーティションを指定しておくと、ユーザーIDやグループIDが数値ではなく名前で表示されるようになる。6.は出力フォーマットを選択する項目。本格的な解析向けには表計算ソフトで扱えるようにCSV形式で出力する



実行後の画面。生成したファイルの場所情報（画面では/forensics/ForensicChallenge/honeypot/output/timeline.txt）が表示される

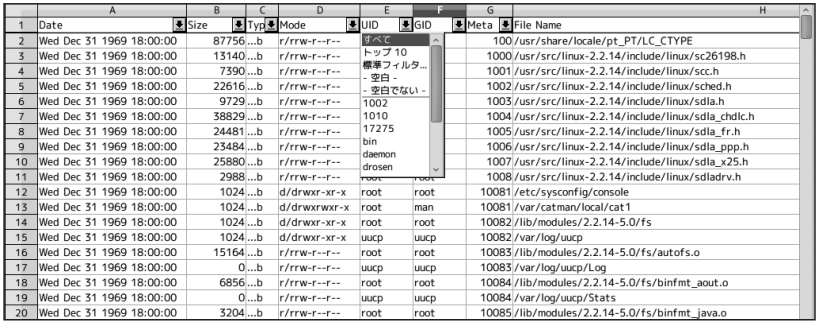
図3 生成データの読み込み



	A	B	C	D	E	F	G	H
1	Date	Size	Typ	Mode	UID	GID	Meta	File Name
2	Wed Dec 31 1969 18:00:00	87756	...b	r/rrw-r--r--	root	root	100	/usr/share/locale/pt_PT/LC_CTYPE
3	Wed Dec 31 1969 18:00:00	13140	...b	r/rrw-r--r--	root	root	1000	/usr/src/linux-2.2.14/include/linux/sc26198.h
4	Wed Dec 31 1969 18:00:00	7390	...b	r/rrw-r--r--	root	root	1001	/usr/src/linux-2.2.14/include/linux/scc.h
5	Wed Dec 31 1969 18:00:00	22616	...b	r/rrw-r--r--	root	root	1002	/usr/src/linux-2.2.14/include/linux/sched.h
6	Wed Dec 31 1969 18:00:00	9729	...b	r/rrw-r--r--	root	root	1003	/usr/src/linux-2.2.14/include/linux/sdla.h
7	Wed Dec 31 1969 18:00:00	38829	...b	r/rrw-r--r--	root	root	1004	/usr/src/linux-2.2.14/include/linux/sdla_chdlc.h

1行目を固定にし、かつオートフィルターを設定した状態を作る。左から、時間（Date）、ファイルサイズ（Size）、タイムスタンプのタイプ（Type）、ファイルタイプ/パーミッション（Mode）、ユーザーID（UID）、グループID（GID）、iノード番号（Meta）、ファイル名（File Name）と続く。タイムスタンプのタイプとは、m、a、cのうちどのタイムスタンプであるかを示していて、複数のタイムスタンプが同じであれば、ma..のように表示される。図では「…b」と表示されているが、これはファイル作成時間（b）をAutopsyでサポートしているものの、今回対象のExt2では該当のタイムスタンプを記録しておらず、0に相当する時間（1970/1/1 00:00からタイムゾーンの6時間を引いた時間）が表示されているためである

図4　タイムラインの解析



	A	B	C	D	E	F	G	H
1	Date	Size	Typ	Mode	UID	GID	Meta	File Name
2	Wed Dec 31 1969 18:00:00	87756	...b	r/rrw-r--r--			100	/usr/share/locale/pt_PT/LC_CTYPE
3	Wed Dec 31 1969 18:00:00	13140	...b	r/rrw-r--r--			1000	/usr/src/linux-2.2.14/include/linux/sc26198.h
4	Wed Dec 31 1969 18:00:00	7390	...b	r/rrw-r--r--			1001	/usr/src/linux-2.2.14/include/linux/scc.h
5	Wed Dec 31 1969 18:00:00	22616	...b	r/rrw-r--r--			1002	/usr/src/linux-2.2.14/include/linux/sched.h
6	Wed Dec 31 1969 18:00:00	9729	...b	r/rrw-r--r--			1003	/usr/src/linux-2.2.14/include/linux/sdla.h
7	Wed Dec 31 1969 18:00:00	38829	...b	r/rrw-r--r--			1004	/usr/src/linux-2.2.14/include/linux/sdla_chdlc.h
8	Wed Dec 31 1969 18:00:00	24481	...b	r/rrw-r--r--			1005	/usr/src/linux-2.2.14/include/linux/sdla_fr.h
9	Wed Dec 31 1969 18:00:00	23484	...b	r/rrw-r--r--			1006	/usr/src/linux-2.2.14/include/linux/sdla_ppp.h
10	Wed Dec 31 1969 18:00:00	25880	...b	r/rrw-r--r--			1007	/usr/src/linux-2.2.14/include/linux/sdla_x25.h
11	Wed Dec 31 1969 18:00:00	2988	...b	r/rrw-r--r--			1008	/usr/src/linux-2.2.14/include/linux/sdladrv.h
12	Wed Dec 31 1969 18:00:00	1024	...b	d/drwxr-xr-x	root	root	10081	/etc/sysconfig/console
13	Wed Dec 31 1969 18:00:00	1024	...b	d/drwxrwxr-x	root	man	10081	/var/catman/local/cat1
14	Wed Dec 31 1969 18:00:00	1024	...b	d/drwxr-xr-x	root	root	10082	/lib/modules/2.2.14-5.0/fs
15	Wed Dec 31 1969 18:00:00	1024	...b	d/drwxr-xr-x	uucp	uucp	10082	/var/log/uucp
16	Wed Dec 31 1969 18:00:00	15164	...b	r/rrw-r--r--	root	root	10083	/lib/modules/2.2.14-5.0/fs/autofs.o
17	Wed Dec 31 1969 18:00:00	0	...b	r/rrw-r--r--	uucp	uucp	10083	/var/log/uucp/Log
18	Wed Dec 31 1969 18:00:00	6856	...b	r/rrw-r--r--	root	root	10084	/lib/modules/2.2.14-5.0/fs/binfmt_aout.o
19	Wed Dec 31 1969 18:00:00	0	...b	r/rrw-r--r--	uucp	uucp	10084	/var/log/uucp/Stats
20	Wed Dec 31 1969 18:00:00	3204	...b	r/rrw-r--r--	root	root	10085	/lib/modules/2.2.14-5.0/fs/binfmt_java.o

UIDの一覧を確認した画面である。数字があることがわかるが、これは現在のパスワードファイルに登録されていないことを示すので、過去には存在していたが削除されたアカウントに関する情報が得られる可能性が高い

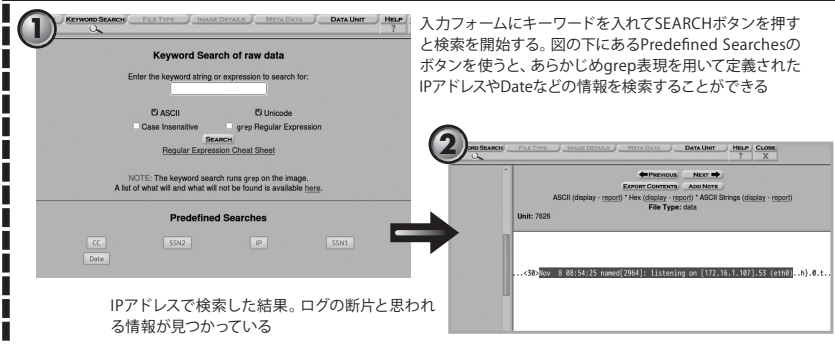
	A	B	C	D	E	F	G	H
1	Date	Size	Typ	Mode	UID	GID	Meta	File Name
101631	Wed Nov 08 2000 08:54:43	493031	..c.	-/rrwxr-xr-x	1002	users	63104	/usr/$OrphanFiles/OrphanFile-63104 (deleted)
101633	Wed Nov 08 2000 08:54:43	1037887	..c.	-/rrwxr-xr-x	1002	users	63106	/usr/$OrphanFiles/OrphanFile-63106 (deleted)
101635	Wed Nov 08 2000 08:54:43	1079584	..c.	-/rrwxr-xr-x	1002	users	63108	/usr/$OrphanFiles/OrphanFile-63108 (deleted)
101637	Wed Nov 08 2000 08:54:43	1111098	..c.	-/rrwxr-xr-x	1002	users	63110	/usr/$OrphanFiles/OrphanFile-63110 (deleted)
101639	Wed Nov 08 2000 08:54:43	41427	..c.	-/rrwxr-xr-x	1002	users	63112	/usr/$OrphanFiles/OrphanFile-63112 (deleted)
101641	Wed Nov 08 2000 08:54:43	1768665	..c.	-/rrwxr-xr-x	1002	users	63114	/usr/$OrphanFiles/OrphanFile-63114 (deleted)
101643	Wed Nov 08 2000 08:54:43	7166	..c.	-/rrwxr-xr-x	1002	users	63116	/usr/$OrphanFiles/OrphanFile-63116 (deleted)
101645	Wed Nov 08 2000 08:54:43	1172532	..c.	-/rrwxr-xr-x	1002	users	63118	/usr/$OrphanFiles/OrphanFile-63118 (deleted)
101647	Wed Nov 08 2000 08:54:43	173212	..c.	-/rrwxr-xr-x	1002	users	63120	/usr/$OrphanFiles/OrphanFile-63120 (deleted)
101649	Wed Nov 08 2000 08:54:43	1815	..c.	-/rrw-r--r--	1002	users	63122	/usr/$OrphanFiles/OrphanFile-63122 (deleted)
101650	Wed Nov 08 2000 08:54:43	1141797	..c.	-/rrwxr-xr-x	1002	users	63123	/usr/$OrphanFiles/OrphanFile-63123 (deleted)
101652	Wed Nov 08 2000 08:54:43	1029928	..c.	-/rrwxr-xr-x	1002	users	63125	/usr/$OrphanFiles/OrphanFile-63125 (deleted)
101670	Wed Nov 08 2000 08:56:05	9	..c.	-/rrw-r--r--	17275	games	109799	/usr/$OrphanFiles/OrphanFile-109799 (deleted)
101671	Wed Nov 08 2000 08:56:05	38	..c.	-/rrw-r--r--	17275	games	109800	/usr/$OrphanFiles/OrphanFile-109800 (deleted)
101672	Wed Nov 08 2000 08:56:05	318	..c.	-/rrw-r--r--	17275	games	109804	/usr/$OrphanFiles/OrphanFile-109804 (deleted)
101673	Wed Nov 08 2000 08:56:05	3051	..c.	-/rrw-r--r--	17275	games	109810	/usr/$OrphanFiles/OrphanFile-109810 (deleted)
101674	Wed Nov 08 2000 08:56:05	2730	..c.	-/rrw-r--r--	17275	games	109811	/usr/$OrphanFiles/OrphanFile-109811 (deleted)
101675	Wed Nov 08 2000 08:56:05	2286	..c.	r/rrw-r--r--	17275	games	109816	/usr/man/.Ci/.temp2 (deleted)
101676	Wed Nov 08 2000 08:56:05	2414	..c.	r/rrw-r--r--	17275	games	109818	/usr/man/.Ci/.temp3 (deleted)

画面はTypeが「..c.」であるレコードを表示した状態である。cはファイルの管理情報（メタデータ）が最後に修正された時間を示す。mはファイルの内容が最後に修正された時間であり、ファイルを移動、コピーしても維持する性質がある。つまり、「..c.」に合致する大半のファイルはm<c<aの状態にあり、OSがインストールされた時のシステムファイル（インストール後、更新はなくアクセスだけがあるファイル）を示す。この結果から、インストールした時間帯をある程度特定できる

	A	B	C	D	E	F	G	H
1	Date	Size	Typ	Mode	UID	GID	Meta	File Name
101630	Wed Nov 08 2000 08:54:43	0	mac.	-/drwxr-xr-x	1002	users	63103	/usr/$OrphanFiles/OrphanFile-63103 (deleted)
101632	Wed Nov 08 2000 08:54:43	0	mac.	-/drwxr-xr-x	1002	users	63105	/usr/$OrphanFiles/OrphanFile-63105 (deleted)
101634	Wed Nov 08 2000 08:54:43	0	mac.	-/drwxr-xr-x	1002	users	63107	/usr/$OrphanFiles/OrphanFile-63107 (deleted)
101636	Wed Nov 08 2000 08:54:43	0	mac.	-/drwxr-xr-x	1002	users	63109	/usr/$OrphanFiles/OrphanFile-63109 (deleted)
101638	Wed Nov 08 2000 08:54:43	0	mac.	-/drwxr-xr-x	1002	users	63111	/usr/$OrphanFiles/OrphanFile-63111 (deleted)
101640	Wed Nov 08 2000 08:54:43	0	mac.	-/drwxr-xr-x	1002	users	63113	/usr/$OrphanFiles/OrphanFile-63113 (deleted)
101642	Wed Nov 08 2000 08:54:43	0	mac.	-/drwxr-xr-x	1002	users	63115	/usr/$OrphanFiles/OrphanFile-63115 (deleted)
101644	Wed Nov 08 2000 08:54:43	0	mac.	-/drwxr-xr-x	1002	users	63117	/usr/$OrphanFiles/OrphanFile-63117 (deleted)
101646	Wed Nov 08 2000 08:54:43	0	mac.	-/drwxr-xr-x	1002	users	63119	/usr/$OrphanFiles/OrphanFile-63119 (deleted)
101648	Wed Nov 08 2000 08:54:43	0	mac.	-/drwxr-xr-x	1002	users	63121	/usr/$OrphanFiles/OrphanFile-63121 (deleted)
101651	Wed Nov 08 2000 08:54:43	0	mac.	-/drwxr-xr-x	1002	users	63124	/usr/$OrphanFiles/OrphanFile-63124 (deleted)
101662	Wed Nov 08 2000 08:56:02	0	mac.	r/rrw-r--r--	root	root	12103	/var/log/tempxfer (deleted-realloc)
101663	Wed Nov 08 2000 08:56:02	0	mac.	r/rrw-r--r--	root	root	12103	/var/log/xferlog
101664	Wed Nov 08 2000 08:56:02	7974	mac.	r/rrw-r--r--	root	root	12104	/var/log/messages
101666	Wed Nov 08 2000 08:56:02	268	mac.	r/rrw-r--r--	root	root	12111	/var/log/secure
101668	Wed Nov 08 2000 08:56:05	0	mac.	-/drwxr-xr-x	17275	games	109792	/usr/$OrphanFiles/OrphanFile-109792 (deleted)
101669	Wed Nov 08 2000 08:56:05	0	mac.	r/drwxr-xr-x	17275	games	109793	/usr/lib/yp/ypxfr_2perday-RPMDELETE (deleted)
101680	Wed Nov 08 2000 08:56:05	0	mac.	r/drwxr-xr-x	17275	games	109870	/usr/man/.Ci/.temp6 (deleted)
101691	Wed Nov 08 2000 08:56:05	0	mac.	-/drwxr-xr-x	17275	games	125254	/usr/$OrphanFiles/OrphanFile-125254 (deleted)

画面はTypeで「mac.」であるレコードを表示した状態である。これはm=a=cのファイル、言いかえればそのマシン上で新しくファイルが作成されて、その後ファイルにアクセスした形跡がないことを示す。侵入された可能性があると考えられる時間の範囲内になっていれば、意図的に改ざんされたファイルの可能性がある

図5 キーワード検索と検索結果画面



入力フォームにキーワードを入れてSEARCHボタンを押すと検索を開始する。図の下にあるPredefined Searchesのボタンを使うと、あらかじめgrep表現を用いて定義されたIPアドレスやDateなどの情報を検索することができる

IPアドレスで検索した結果。ログの断片と思われる情報が見つかっている

◉タイムライン解析

さて、いよいよ解析作業に入るわけだが、取りかかりとしてまずは直近の時間からさかのぼっていくか、不審な動きがあったという情報がある場合はその時間帯を中心に見ていくなど、調査する人やケースによってそれぞれだろう。もちろん1行ずつ確認していってもよいが、一般にデータ量は膨大になるため、すべてを確認するのは現実的でない。例としてオートフィルター機能を活用した手がかりの収集方法について紹介しよう。オートフィルターを使えば、設定した条件に合致したデータを抽出できるため、見落としがちな情報を浮き彫りにすることができる。

タイムラインの解析中に注意しなければならないのは、ファイルのタイムスタンプは容易に改ざん可能であるということだ。調査を困難にするために、意図的にタイムスタンプを変更するマルウェアも実際に存在するので、今見ているタイムスタンプの情報が正しいとは限らないことを念頭に置きつつ、調査に取り組む姿勢が重要である。

ついでに削除状態のファイルについて触れておこう。タイムラインを眺めているとファイル名の後ろに（deleted）や（deleted-realloc）と付いているものがあることに気づくはずだ。どちらも削除状態として検出、つまりファイルシステム上で再利用可能の状態にある領域に残っていた情報を解釈した結果を表示している。ここで、（deleted-realloc）は別のファイルによってもともとそのファイルが使っていたメタデータの領域が上書きされていることを示している。そのため、（deleted-realloc）と付いているファイル名と、一緒に表示されているメタデータは対応しない。（deleted）は該当のメタデータ領域が再利用可能なままであるが、それでも別のファイルによってすでに利用されていた可能性はあるので、他の手がかりとあわせて、削除されたファイルの情報であったか判断することになる。

◉未割り当て領域からの文字列検索

タイムライン解析でも不審な点が見つかったが、実際にログやヒストリーファイルを参照してみると明らかに記録されている内容が少ないことに気づくだろう。侵入者が侵入の痕跡を隠ぺいするためにこれらのファイルを細工したことが予想される。

そのような状況で役に立つキーワード検索についても紹介しておこう。Autopsyでは、イメージに対してキーワード検索することができる。未使用領域も対象になるため、過去にあった断片的な情報が得られる可能性がある。ここではスワップパーティションのイメージに対して実行してみる。スワップパーティションはもともとメモリ上にあったデータを退避するための領域であるため、調査に役立つ情報が得られる可能性がある。キーワードにはgrep表現も使えるため、探したい情報に応じて活用するとよいだろう。

なお、Autopsyはフラグメントされているデータをつなげて検索しないため、本来ヒットすべきデータを見落す可能性がある。また、Shift_JISやEUC-JPなどの日本語の文字コードには対応していない。検索機能を使う場合はこれらの注意点を理解した上で実行する必要がある。

LESSON 2 Scan24の演習課題

The Honeynet PROJECTよりダウンロード
http://old.honeynet.org/scans/scan24/

Scan24は筆者がフォレンジック調査のトレーニングなどで課題としてよく利用している演習課題である。この問題は中級者向けの課題となっているが、データサイズが小さく、それほどトリッキーな問題でないこともあり、初心者でも楽しめる内容となっている。

まずScan24の課題内容だが、大雑把には次のようなストーリーになっている。『法執行機関が高校生に麻薬を販売していた密売人（Joe Jacobs）を逮捕したところ、自宅からフロッピーディスク（FD）が1枚発見された。あなたのミッションは、このFDのイメージファイルから、麻薬の密売人が他の高校にも麻薬を販売していたかどうか証拠となるデータを、一定期間内に探し出してほしい』というものだ。詳細なストーリーは設問のWebに掲載されているthe police reportを参照していただくとして、この課題には5個の設問と、6問目にボーナス問題が用意されている。

基本的なゴールは、この麻薬の密売人が他の高校にも販売していたとする証拠データの発見だ。しかし、具体的に何を発見したらよいのか明確には書かれていない。これは実際の案件でもそうなのだが、具体的に何を発見したらよいか明確でなく、さまざまな手がかりや情報を精査していくことで最終的に証明しようとしている内容に関するデータを揃えるといった具合になる。

なお、Scan24の課題はフロッピーディスクなのでFATファイルシステムとなっている、マイクロソフトからFAT32に関する仕様書※1をダウンロードし参照しながら挑戦すれば、ファイルシステムの構造についてもわかりやすくなるはずだ。

◉ FDイメージ

Scan24のWebから課題ファイル（FDのddイメージファイルをzipで圧縮している）をダウンロードすることができる。まずはzipフィルを展開すると、imageというサイズ約1.4MBのファイルが展開される。このファイルはFDのRAWイメージなので、このままでは内容を閲覧して調査しにくい。解析の方法として、このイメージファイルを再びFDメディアにddコマンドで書き戻し、WindowsからFDにアクセスするという方法もあるだろう。物理メディアに書き戻す利点としては、Windowsからアクセスした場合にはどの様に見えるのか把握できる点、Windows用のさまざまなツールを解析に利用することができるという点がある。物理的なメディアに書き戻さずとも、「仮想な背中※2」にて提供されているVirtual Disk Driverなどを使い、仮想的にイメージファイルをドライブとしてマウントしてWindowsからアクセスするという方法もあるだろう。

コンピューターフォレンジック調査では、通常証拠となる電子データに対して書き込みを行なわないように書き込み禁止装置（ライトブロッカー）を利用してデバイスにアクセスするのが基本だが、FDであればライトプロテクトスイッチにより簡単に書き込み禁止が実施できる。

物理的なメディアにリストアしてアクセスするのではなく、イメージファイルのままで解析するということであれば、フォレンジック用ツールなりバイナリエディタの出番となる。FATファイルシステムくらい目視で読める！　という方は

Scan24の演習　問題文

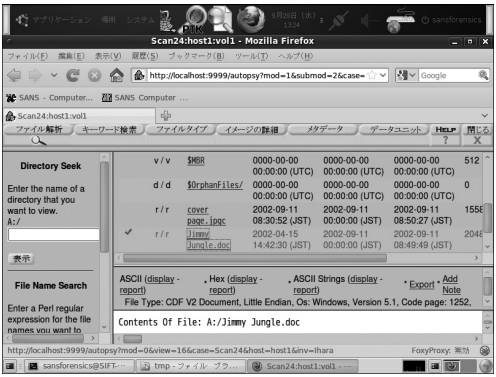
1. Who is Joe Jacob's supplier of marijuana and what is the address listed for the supplier?
2. What crucial data is available within the coverpage.jpg file and why is this data crucial?
3. What (if any) other high schools besides Smith Hill does Joe Jacobs frequent?
4. For each file, what processes were taken by the suspect to mask them from others?
5. What processes did you (the investigator) use to successfully examine the entire contents of each file?

Bonus Question:

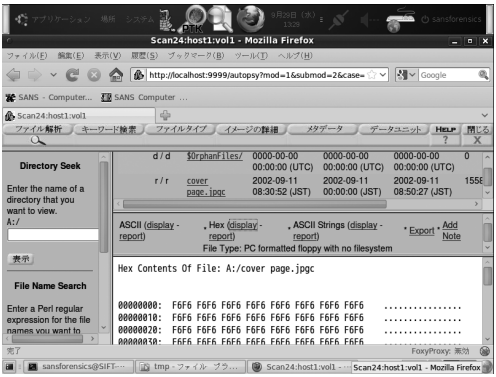
6. What Microsoft program was used to create the Cover Page file. What is your proof (Proof is the key to getting this question right, not just making a guess).

※1　http://www.microsoft.com/whdc/system/platform/firmware/fatgen.mspx
※2　http://chitchat.at.infoseek.co.jp/vmware/indexj.html

攻略のヒント FDイメージの解析



まずは今見えているファイルがそれぞれ何か、どのようなデータであるかを確認してみよう。Autopsy上でファイル名をクリックすれば、そのデータ内容が表示される。この時、Autopsyは自動的にクリックしたファイルのファイルタイプ識別を行ってくれる。ブラウザー画面上の真ん中にFile Type:の項目として識別した結果が表示されるので参考になるだろう。例えば、coverpage.jpgcというファイルは、クリックするとファイルタイプとして「PC formatted floppy with no filesystem」というメッセージが表示されるはずだ



これは、coverpage.jpgcというファイルのデータ内容をHEXモードで表示すると理由がわかるはずだが、ファイル名から期待できるJPEG画像ファイルのデータパターンではなく、¥xF6という値がファイルの先頭からずっと埋まっている。
余談ということでもないのだが、JPEG画像ファイルの先頭16進数パターンは、通常¥xFF¥xD8¥xFF...となっている。他にもいくつか興味深い点があるはずだ。例えば拡張子EXEのファイルなども存在しているが、これもファイルタイプに注目する必要がありそうだ。
ファイルシステム内にはWordの文書ファイルも存在している。Autopsy上での閲覧が難しい形式のファイルは、Exportを利用してイメージファイル内から個別ファイルとして取り出し、別途対応するアプリケーションで内容を確認するという方法をとる。課題のイメージ内にあるWordファイル内には英語の文書しかないので、ASCII Stringsの表示モードを使うことでテキスト文字列だけを簡単にAutopsy上から確認することも可能だ

バイナリエディタを使っていただくとして、The SleuthKit&Autopsy を利用するのであれば、ケースにイメージを追加するには、FAT ファイルシステムのボリューム（Partition）として追加すればよい。間違えて Disk Image として追加するとワーニング画面が表示されるはずだ。

◉画像ファイルの痕跡を探索する

設問の 2 番目では、coverpage.jpg ファイルに関する質問が出ている。ファイルの " 範囲 " というのがポイントだが、現時点ではファイル名 coverpage.jpgc をクリックしても画像を確認することができない。ファイル名として JPEG ファイルのような痕跡があるわけだから、FD 内の未割り当て領域にはまだこの JPEG 画像データが残っている可能性も考えられる。バイナリエディタを使って 16 進数パターンで JPEG ファイルのヘッダーを見つけるという手もあるが、SIFT 環境内にはこういったケースで便利な Foremost[※3] というツールが含まれている。

このツールはあらかじめ定義されたパターンを利用してファイルを復元するためのツールである。例えば、Pagefile.sys や Unallocated Clusters（未割り当てクラスター領域）に対して、JPEG 画像ファ

※3　http://foremost.sourceforge.net/

イルのリカバリーなどを行うのに向いている。

Foremost がリカバリーできるファイルの種別は多数あるので、指定したパターンを持つファイル（例えば JPEG）だけをリカバリーするのであれば、foremost -t jpeg image のようなパターンを指定すればよい。対応しているファイルをすべて指定したファイル内から取り出すのであれば、foremost -t all image とすれば output フォルダ配下に、各ファイルタイプごとに出力結果が保存される。Foremost を使う上で注意する必要がある点として、Foremost ではヘッダーパターンとフッターパターンを単純に切り出すだけで、ファイルシステムの構造は考慮していないという点である。例えば、JPEG 画像ファイルがフラグメントされている状態で Foremost を実行しても正しく画像ファイルがリカバリーされないことになる。

なお、Autopsy ではキーワード検索の画面内から、未割り当て領域を抽出して別途ファイルとして保存することができる。もし未割り当て領域だけを Foremost によるリカバリーの対象としたいのであれば、Extract Unallocated Sectors の機能を試してみていただきたい。

◉クラスターとスラックスペース

Windows が一般的に利用する FAT/NTFS ファイルシステムでは、ファイルなどが作成される場合、クラスターという単位でディスクへの書き込みが行なわれる。ハードディスクやフロッピーディスクの場合、通常は 1 セクターは 512 バイトとなっているが、1 セクター単位で書き込みや読み取りを行うと効率がよくない。このため、ファイルシステムのフォーマット時にいくつかのセクターを 1 つにまとめ、クラスターという単位で取り扱う。例えば、FD のような小さなファイルシステムの場合には、1 セクター =1 クラスターとなっているが、ハードディスクのようなサイズが大きなファイルシステムでは、8 セクター =1 クラスターとなっている。デフォルトでの割り当てクラスターサイズについては、マイクロソフトの文書番号 : 140365「NTFS、FAT、および exFAT の既定のクラスターサイズ」に記載されている。

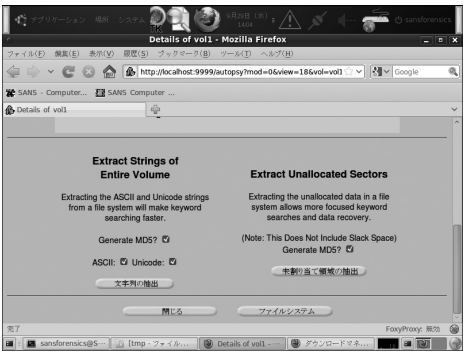
エクスプローラーからファイルのプロパティを見た時に、ファイルのサイズとディスク上のサイズという 2 つのサイズが表示されていることが確認できると思うが、サイズはファイルの論理サイズでありデータのサイズとなる。これに対してディスク上のサイズは、割り当てられているクラスターのサイズということになる。

データはクラスター単位で書き込まれるわけだが、例えば一度に 8 セクター（1 クラスター）の割り当てが行なわれても、実際に書き込まれるデータはそのうち 1 セクター分しか使っておらず、残り 7 セクター分は何も書き込まれないということもありえる。ファイルの論理サイズからクラスターの終端までの " 余っている " データ範囲を、ファイルスラックと呼んでいるが、ここに重要なデータが残っている場合がある。今回の Scan24 の課題と直接の関連はないが、例えば Windows XP のゴミ箱フォルダを空にした場合には、ゴミ箱内のファイルを管理しているインデックスファイル（INFO2）のサイズが 20 バイトまで小さくなる。この時、INFO2 ファイルのスラック領域上には、このゴミ箱に入っていたファイルのリストがまだ残っている場合がある。

このため、ファイルの内容が特に重要でないケースであっても、ファイルのスラック領域を確認することで、以前にそのセクターを利用していたファイルのデータ痕跡などから、重要な手がかりが得られる場合もあるのだ。

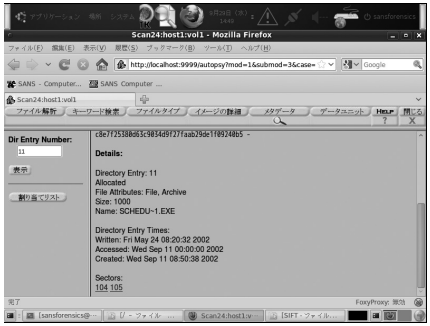
Autopsy 上からファイルのスラック領域を確認したい場合には、該当ファイルのメタデータからディレクトリエ

攻略のヒント　画像ファイルの痕跡



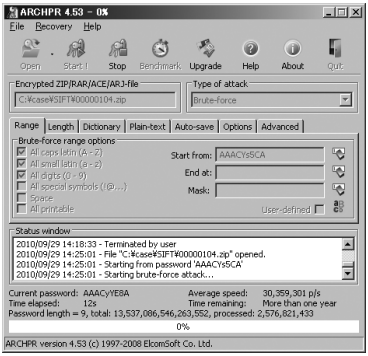
JPEG画像ファイルの終端の16進数パターンは「¥xFF¥xD9」となっている。上の図とあわせてヒントにしてほしい

攻略のヒント スラック領域



Autopsyからスラック領域を調査する

攻略のヒント パスワード解析



この課題では、解析に利用できる期間が定められており、一定の期限内に報告を上げる必要がある。ブルートフォースによるパスワード解析を行ってもおそらく指定されている期限内にパスワード解析が完了することはないと思われるため、パスワードを解除するにはさらなる工夫が必要になるだろう。ちなみに筆者はstrignsコマンドを使いASCII文字列を取り出し、それをそのまま辞書として利用したのだが見事に駄目だった。何が手順として問題となったのかは、答えを見つけた読者ならわかっていただけるだろう

ントリの番号を指定し、Sectors のところをチェックすれば、データが書き込まれているセクター範囲を確認することができる。データが書き込まれている最終セクター位置とその前後を確認することでスラック領域を調べられる。

◉パスワードの解析

この課題では、いずれかのタイミングで、zip ファイルのパスワード解析の必要性が出てくる。具体的にどの zip ファイルのパスワード解析が必要となるか書いてしまうとネタバレになるので避けるが、これを解けないと先に進めないことになる。

最近は GPU を利用してパスワードクラック速度を上げる方法も存在するが、今回の答えとなるパスワード文字列を、いま原稿を書いている手元の PC（Core2 Duo 搭載の一般的なスペック）で解析しようとした場合、結果が出るには 2 日と 1 時間という予測が表示されている。しかし、この予測数値は、筆者が答えのパスワード長と文字パターンを知っているのでそれに合うようにブルートフォースのパラメーターを調整した結果である。英数字（大文字・小文字）と数字のパターンで、文字列長だけ答えと同じ長さに設定した条件でブルートフォースした場合、予測期間は 1 年以上となっている。CPU を利用したブルートフォースでこの zip ファイルを解くのはかなり厄介なはずだ。

◉まとめ

あまり詳しく解説してしまうとネタバレになってしまうのだが、設問で問われているように、この FD イメージにはある種の細工が施されている。最近、日本では検察（検事）が FD のタイムスタンプを改ざんしたことが大きなニュースになったばかりだが、この FD に細工した人物はもう少しやっかいな仕掛けを施している。筆者からの助言としては、フォレンジックツールはファイルシステムなりの仕様に従ってパースを行なっているだけであり、ツールを信用しすぎるのはよくない、ということである。結局のところ、この課題を解く上でもっとも重要なのは、データを調べているあなたの眼であると言っても過言ではないだろう。

この課題は 2009 年までセキュリティ & プログラミングキャンプの解析コースで課題として利用していたのだが、過去に最も速く目的とするデータに到達した生徒は約 10 分程であった。むろんヒントは事前にいろいろと出してはいたのだが、律儀に TSK&Autopsy を使う生徒よりも、バイナリエディタで調査を行なった生徒の方が回答は速かった印象がある。読者の皆さんも手詰まりになったらバイナリエディタで眺めてみてはどうだろうか。