ハニーネットプロジェクト
14章4項375～16章1項425

1730010

27,Jul,2018

14.4 ウォークスルー:リバース大賞

- **ウォークスルー**【 walk-through 】
  - **ウォークスルー**とは、連絡通路、立ち稽古、リハーサル、実地検証、通り抜けられる、などの意味を持つ英単語。
  - システム開発の分野では、開発プロジェクトのメンバーが一同に会し、仕様や構成の問題点を探したり解決策を討論したりする作業のことをウォークスルーという。[IT用語辞典 e-Words, http://e-words.jp/]

ハニーネット上で不正にdownload & installされた謎のプログラム(the-binary)をリバースエンジニアリング

[http://old.honeynet.org/reverse/the-binary.tar.gz]

14.4.1 情報収集

```
$ file the-binary
the-binary: ELF 32-bit LSB executable, Intel 80386,
                        version 1 (SYSV), statically linked, stripped
$ strings the-binary | more
 :
@(#) The Linux C library 5.3.12
 :
$ strace the-binary
execve(“the-binary”, [“the-binary”], [/* 28 vars */]) = 0
personality(PER_LINUX)                                 = 0
geteuid()                                  = 1001
_exit(-1)
$ objdump –d the-binary
[ . . . ]
804828b:        push    $0x1
804828d:        push    $0x11
804828f:        call    0x80569bc
8048284:        add     $0x8,%esp
```

ELF: UNIXで利用されるフォーマット

Intel 80386: i386…linux, *BSD

statically linked: 静的リンク

stripped: シンボル削除済

strace: 呼び出されるシステムコールをトレスする

14.4.1 情報収集


```
$ objdump -j .comment -s the-binary | head
  the-binary: file format elf32-i386
  Contents of section .comment:
        0000 …… ……  .GCC: (GNU) 2.7.
        0010 …… ……  2.1.2..GCC: (GNU
        0020 …… ……  ) 2.7.2 ..GCC: (G
        0030 …… ……  NU) 2.7.2.1.2..G
         :
```

ELF:プログラムのロード方法,追加セクション(コメント, etc..)

コメントは一般にコンパイラバージョン番号で埋められる

結論(仮)
the-binaryではlibc 5.3.12が使われていてgcc2.7.2でコンパイルされている。
検索すると(https://distrowatch.com)、Slackware3.1, RedHatLinux4.2に含まれていることがわかる。

14.4.1 情報収集

Slackware3.1: 1996

Red Hat Linux4.2: 1997

↓

- ふるい! (the-binaryが不正利用されたのは2002年)
  - 数年前にコンパイルされた？
  - 最近でも古いバージョンを使う理由があった？

14.4.1 情報収集

- the-binaryのlibcで使われているオブジェクトについて、
- Slackware, Red Hat Linux 向けのlibcライブラリをバージョンを分けて比較した。

- 一致数が最も多かったものはSlackware 3.1

↓

- the-binaryはSlackware3.1でコンパイルされている

14.4.2 逆アセンブリリストの取得

- the-binaryはシンボル削除されていた(stripped)
- シンボルテーブルがないと、ライブラリ関数やユーザに作られたコードなどの、どれが呼び出されているのか判断できない。
- が、オリジナルのオブジェクトファイル(Slackware3.1のlibcなど)にはシンボルテーブルが含まれている。

↓

- the-binaryのシンボルテーブルが復元できる！

シンボルテーブルの復元

the-binaryの0x8001000の位置にfoo.oが見つかった、と仮定すると

foo.o			**the-binary**	
0x000	bar	加算 →	0x8001000	bar
0x120	baz		0x8001120	baz

シンボルテーブルが再現できる

出てくるオブジェクトファイル毎に復元を行うと、43000行の逆アセンブルリストが2500行まで減った。
残ったリストを調べると、main含め10個のユーザーコード関数を調べれば良いことが判明。
これをfunc1～func9と名付ける。

14.4.3 逆コンパイルと分析

プログラムの概要を理解するために有用な関数(呼び出し回数)リスト

_IO_sprintf	(11)	__libc _chdir	(1)
__libc_fork	(14)	__libc_setsid	(4)
accept	(1) サーバに使う	bind	(1)
execl	(1)	gethostbyname	(5)
inet_addr	(7)	listen	(1)サーバに使う
recv	(2)	send	(1)
sendto	(6)	setenv	(2)
signal	(15)	socket	(7)
system	(2)	unsetenv	(1)

14.4.3 逆コンパイルと分析

プログラムの中の文字列定数を調べる

$ objdump –s –j .rodata the-binary | head -20

:

:

*setenv*と両立する内容

“PATH”

“/sbin:/bin/:/usr/sbin:/usr/bin:/usr/local/bin:.”

:

二つ目の*setenv*は次と関連している

“TERM”

“linux”

ーーーーー

‘/bin/csh -f -c “%s” ’

‘/bin/csh -f -c “%s” 1> %s 2>&1’

フォーマット文字列「%s」があるので、sprintfに渡されると思われる

「1>%s 2>&1」はシェルリダイレクト。systemコールへの引数として利用されている可能性がある

→**実行時に決定されるシェルを実行している。　バックドアサーバを立てるプログラム**

the-binary

```
main{
   :
   :
  socket(2,3,11)
  loop {
          recv
           :
          }
   :
  }
```

socket(domain, type, protocol);

prptocol…IP(0), ICMP(1), TCP(6), UDP(17)

(11)…? 11はリストにない→通信隠蔽？→サーバを立てて、攻撃者と通信する？

その他の関数から、IPアドレスを偽装やパケット送信のループが見つかる。

→Dos攻撃に使用？

まとめ

i386 UNIX 向け(Linuxの可能性高)のCで書かれたプログラム。

パーミッションがある場合、shの使えるバックドアサーバを仕込む。

通信隠蔽をしながら、バックドアからDos攻撃の実行も可能。

プログラム構造

```
go_daemon()                  #fork() setsid() fork() chdir() close()
socket()
loop() {
  recv()
  func9()                    #func9()で受信パケットを前処理
  switch {                   #処理結果によってcase分け
    case 0x1:    func8(); func1()                                  #攻撃者へのデータの送信(状態通知？)
    case 0x2:
    case 0x3:   sprintf("/bin/csh  -f -c ¥"%s¥" 1> %s 2>&1")       #攻撃者へのデータの送信
                system(); fopen(); fread(); func8(); func1(); fclose()
    case 0x4:   func4()                                           #Dos攻撃　（ループ処理なので）
    case 0x5:   func6()                                           #Dos攻撃
    case 0x6:   socket(); bind(); listen(); accept(); setenv(); execl()   #シェルバックドア
    case 0x7:   sptintf("/bin/csh  -f -c ¥"%s¥");  system()        #シェルコマンドの実行
    case 0x8:
    case 0x9:   func4()                                           #Dos攻撃
    case 0xa:   func7()                                           #Dos攻撃
    case 0xb:   func7()                                           #Dos攻撃
    case 0xc:   func5()                                           #Dos攻撃
  }}
```

case 0x3: sprintf(“/bin/csh -f -c ¥”%s¥” 1> %s 2>&1”) system(); fopen(); fread(); func8(); func1(); fclose()

- stdoutとstderrをファイルに書き込み、コマンドを実行している。
- その後、書き込んだファイルを読み込み、func8と1を呼び出す。

↓

- 何かを書いて、その内容でなにかしている……

なにか

func8(): 文字列を使う。何かをエンコーディング
func1(): func2()を呼び出し
func2(): row socketを生成し、プロトコル11でパケットを送信

ファイルの内容を攻撃者に送っていると推測される

main, func1~9

main ：受信したコマンドに従ってアクション
func1 : func2のラッパ
func2 : 攻撃者へデータ送信
func4 : DoS攻撃
func5 : DoS攻撃
func6 : DoS攻撃
func7 : DoS攻撃
func8 : 送信データのエンコード
func9 : 受信データのデコード

- 送信前にfunc8()でエンコードしているなら、
- 向こうの送信データもデコードされていそう。

↓

- 受信直後のfunc9()はデコードでは？

- 仮説は手動逆コンパイルの手助けになる。わざと分かりづらく記述されていても、プログラムの目的が想定されていれば理解が速くなる。

逆コンパイル

- プログラムが何をどうするかはわかったため、次はその詳細を知る
  →Cに逆コンパイルし、ソースを分析する
- どのような最適化オプションが使用されたかの確認手順

```
loop {
        逆コンパイル

        オプション付きコンパイル

        逆アセンブル

        オリジナルと比較
}
```

- このように、逆コンパイルの結果が正しいかの確認の手段は講じるべき

- ソースコードが確保できたら、リバースエンジニアリングは終了
  →分析へ

14.5 まとめ

- リバースエンジニアリングでは、ソースコードのないプログラムを分析した。(通常、静的分析と能動的分析の両方を経由)
  - 静的分析
    - 徹底的　　(情報収集→逆アセンブリ→逆コンパイル)
  - 能動的分析
    - 高速　　(トレース→ブラックボックス分析→デバッグ)
- 分析の前に、目標を明確にすることで方向が定まり、時間もかからない

14.5 まとめ　the-binary

- 分散DoSツールの片割れエージェント
  - SYNフラッド
  - jolt2型
  - DNSレスポンスフラッド
- C言語、Slackware3.1上でコンパイル、gccオプション -static -O が使われた
- 検出妨害のための工夫がなされている
- 攻撃者とエンコードデータを隠蔽通信、認証も信頼性もない。
- データ送信の際に、デコイが用意されている
- オンデマンドでバックドアrootシェルを提供する
- シェルコマンド機能の提供、オプションで出力を返す

14.6 リソース リバースエンジニアリングに役立つトピック

- アセンブリ言語プログラミング
  - 教科書が多い。他アーキテクチャへ応用可。
- コンパイラ理論
  - 乗除算ビットシフトなど技法を知ると、コード理解が捗る。
- 逆コンパイル
  - ソースコードのないプログラムをソースに戻すには……。[URL失効]
- エクスプロイトコーディング
  - メモリ内のプログラムの配置を理解。ハッカー演習。1[URL失効]
  - 脆弱性をつくのはたぶん楽しい。2[https://www.hackerslab.org/]
- プログラムデバッグ
  - デバッガに精通しよう。
- プログラミング理論
  - アルゴリズムならなんでもできた方がいい。

15章 集中データコレクションと分析

- 集中データコレクション
  - データを一箇所で管理したい。
- 様々なタイプのデータを一箇所で管理できると、特に分析において強い。
  - ファイアウォール
  - IDS（侵入検知システム）
  - ローネットワークキャプチャ
  - キーストロークログ . . .etc.
- 情報を共有し、世界の趨勢を調べたり全貌を理解するのに有用。

15.1 データの集中化

- syslogなどでlogメッセージをサーバに送信する
  - 一般的でデフォルトで組み込まれがち。
  - UDPなので通信の保証はない
  - 特定の価値ある情報を取り出し解析するのに時間がかかりやすい
- SQLデータベースを用いる
  - ハニーネットアライアンスはMySQL
    - MySQLは様々なOSプラットフォームに対応し、高速かつ無償
  - 導入が難しい

    →導入方法を紹介

15.1.1 ファイアウォールログ

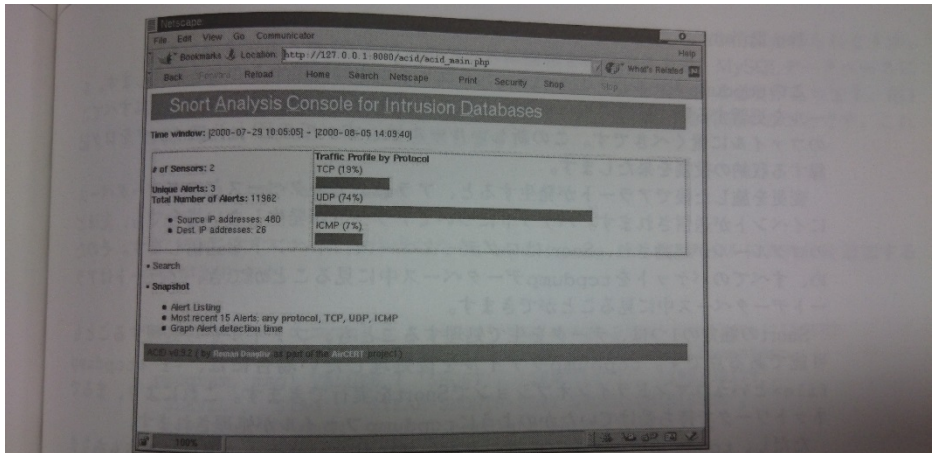
- だいたいsyslogサーバにlogを送信している。
- テキストファイルのままでは綿密なデータ分析に向かない。
- 正規表現を使う人もいるが、SQLのほうがいい。
    - ハニーネットアライアンスは世界中の様々なファイアウォールのlogに対して、共通した標準フォーマットを作った
    - FISQ (Firewall SQL Import Script)を作成し、logをSQLにインポート
    - [URL失効]

15.1.2 IDSログ

- IDS (Intrusion Detection System)
  - 不正侵入検知システム
  - 最も集中化しやすい
  - IDS製品は基本的にSQLデータベースに記録する機能を備えている。
- Snortの場合、下記でローカルとリモートに記録する。

  output database: alert, mysql, user=root password=test dbname=snort host=remotehost
- データベースを閲覧するのにACID(Analysis Console for Intrusion Datavases)が人気
  - 分類されており、検索も可能
  - イベントを相互に関連付けられる

  →調査が楽



15.1.3 tcpdumpログ

- ハニーネット経由のパケットを記録するのに使用
  - バイナリファイルに出力→ブラウジングに工夫がいる
  - 既存のデコーダは中央データコレクションに有用ではない

- SnortでSQLにインポートすると解決。
  - スニファモードでtcpdumpの機能を多く取り入れている。
  - データベースに出力も行える。

  output database: log, mysql, user=root password=test dbname=tcpdump host=remotehost
  - すべてのパケットを記録するためのルールの記述(local.rules)

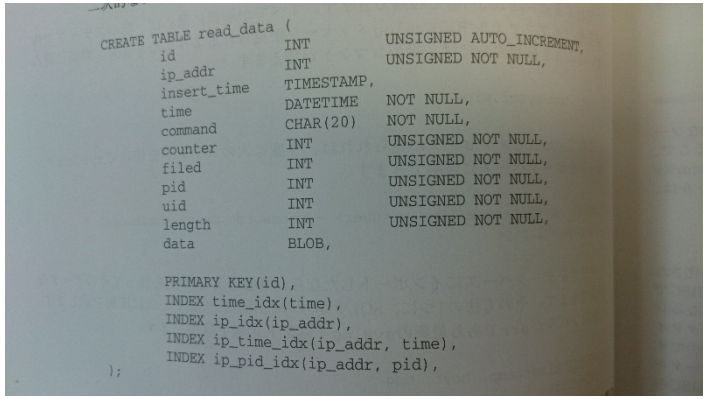
  log ip any any <> any any (msg: “tcpdump”)
  - ※DoS攻撃で満杯にならないように監視しなければならない
  - データの生処理も可能!

15.1.4 システムログ

- 情報の関連付けに役立つ（かも）
- SQLにインポートしやすいフォーマットを吐き出すsyslog-ngなどを選ぶ
- インポート手順
  1. syslog-ng用のパイプファイルの作成
  2. syslog-ng.confへの書き込み
  3. syslogメッセージが確認できたら、データベースに保存

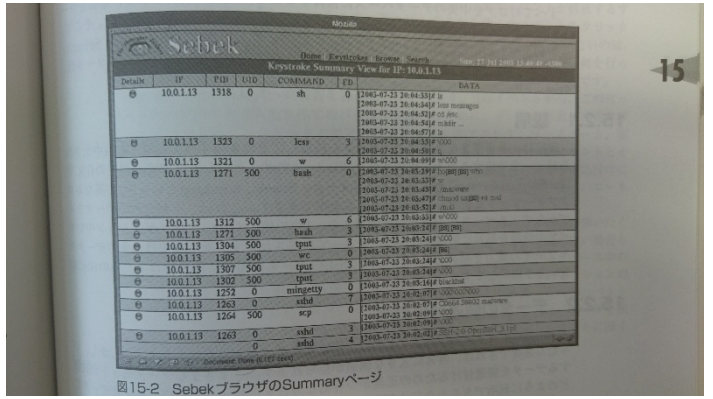
15.1.5 キーストロークログ

- 攻撃者の行動のキャプチャ (重要!)
- 侵入のためになにを、侵入してからなにをしているか…をキーストロークから見る
- Sebekを用いてキーロギングし、データをMySQLへ保存



```
CREATE TABLE read_data (
        id              INT             UNSIGNED AUTO_INCREMENT,
        ip_addr         INT             UNSIGNED NOT NULL,
        insert_time     TIMESTAMP,
        time            DATETIME        NOT NULL,
        command         CHAR(20)        NOT NULL,
        counter         INT             UNSIGNED NOT NULL,
        filed           INT             UNSIGNED NOT NULL,
        pid             INT             UNSIGNED NOT NULL,
        uid             INT             UNSIGNED NOT NULL,
        length          INT             UNSIGNED NOT NULL,
        data            BLOB,

        PRIMARY KEY(id),
        INDEX time_idx(time),
        INDEX ip_idx(ip_addr),
        INDEX ip_time_idx(ip_addr, time),
        INDEX ip_pid_idx(ip_addr, pid),
);
```



Sebek

Keystroke Summary View for IP: 10.0.1.13

Details	IP	PID	UID	COMMAND	FD	DATA
	10.0.1.13	1318	0	sh	0	
	10.0.1.13	1323	0	less	3	
	10.0.1.13	1321	0	w	6	
	10.0.1.13	1271	500	bash	0	
	10.0.1.13	1312	500	w	6	
	10.0.1.13	1271	500	bash	3	
	10.0.1.13	1304	500	tput	3	
	10.0.1.13	1305	500	wc	0	
	10.0.1.13	1307	500	tput	3	
	10.0.1.13	1302	500	tput	3	
	10.0.1.13	1252	0	mingetty	0	
	10.0.1.13	1263	0	sshd	7	
	10.0.1.13	1264	500	scp	0	
	10.0.1.13	1263	0	sshd	3	
			0	sshd	4	

15

図15-2 Sebekブラウザのサマリーページ

- データベースは、SebekWebでブラウズできた。
- また、Webベースの分析インターフェースも備わっていた。

15.1.6 データ集中化まとめ

- ファイアウォールログ
- IDSログ
- tcpdumpログ
- システムログ
- キーストロークログ

- これらを記録できる環境を整えた。
- 次は、
  - 情報を一元ブラウジングしたい。
    - 難しくはないが、時間がかかる場合がある。
  - データを有効利用したい。
    - データの相互関連付けで、攻撃の最中に何があったのかをより詳細に分析できる。
- そのために、**Honeynet Security Consoleを作成した。**

15.2 Honeynet Security Console

- データベースが中央に保持された？
- データから出来事を見極めるためには、適切なツールと知識も必要
- HSC(Honeynet Security Console)を使って必要な作業を実行

- 下記リンクでツール公開中！(公開してない！)

[http://www.activeworx.org/hsc](**ドメイン売出し中！**)

15.2.1 Honeynet Security Console

- Windowsアプリケーション
- ログから、関連付け、集中化されたビューを作る
  - Snort IDS ログ
  - syslog ログ
  - ファイアウォールログ
  - Sebek ログ
  - tcpdump ログ

15.2.2 データ関連付けの例

- すべてを集中化するのは作業が大変な場合があるが、達成には大きな価値がついてくる。
- 例として、Apacheサーバに対する攻撃について吟味する
  - HTTPS, port433
- 被攻撃時点で脆弱性は既知
- しかし、「本当に」なにが

起こったかはわからない。

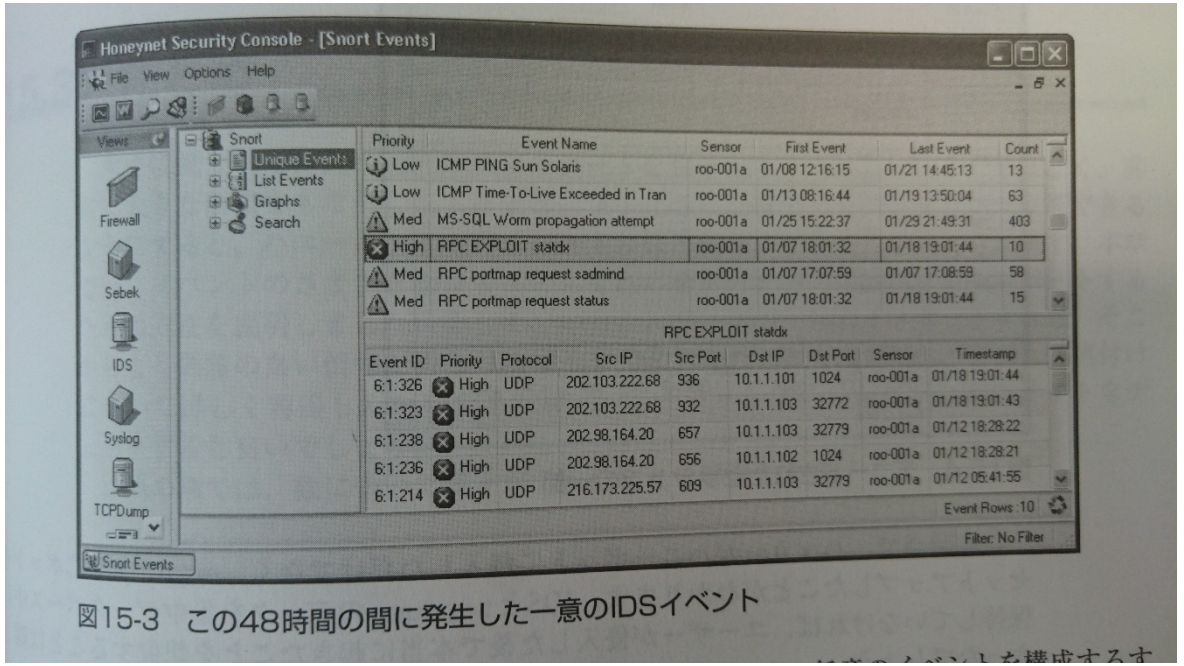


図15-3　この48時間の間に発生した一意のIDSイベント

15.2.2 データ関連付けの例

- HSCでイベントを調べると、ブラックハットが何を行ったのか、本当に侵入したのか、単に試しただけなのかなどが簡単にわかる。
  - アドレスやポート、時間近似性からデータを関連付け
  - デコードされたパケットの中身を調査
- 正確とは限らないが、有用な情報

- セッションの一部を見ると、
  - 侵入　…だけでなく
  - eggdrop IRCボットのセットアップ
- を行っていることがわかる。

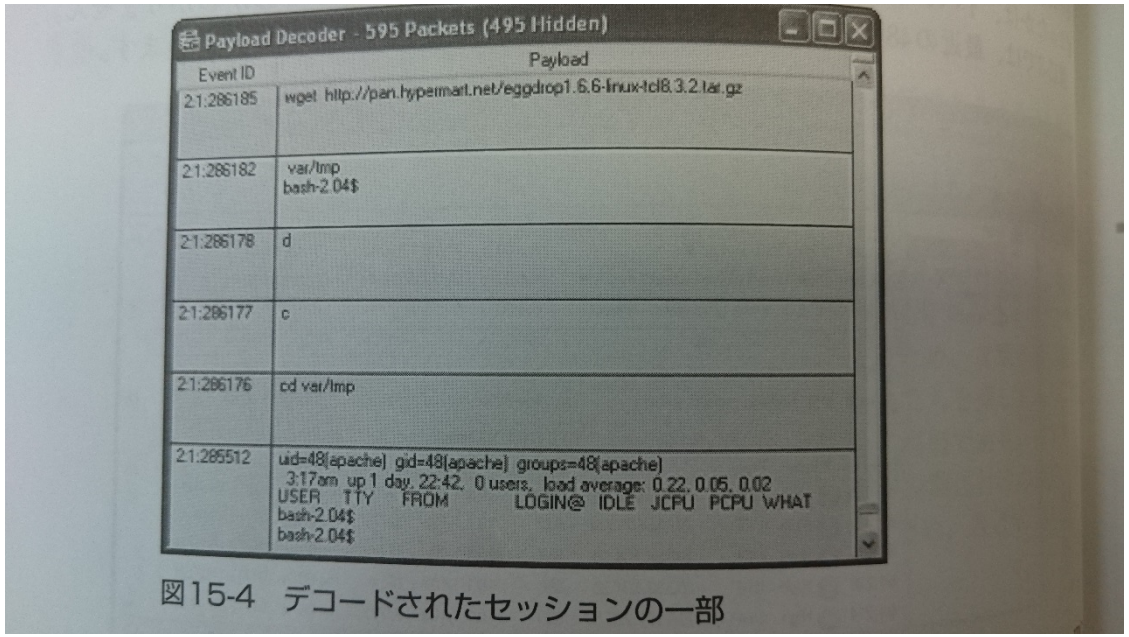


図15-4　デコードされたセッションの一部

15.3 まとめ

- データの集中化は難しくない。
- データに価値をもたらす。
- 特にデータベース化は価値が高くなる。
- 集中化とブラウジングの環境を整えるとイベントの標準化単純化につながる。
- 関連付けにより、より分析しやすくなる。

16章 プロファイリング

- 敵を知る(Know your enemy)
- これまでは戦略、技術、ツール、問題を扱った。
- 本章では思考、動機を取り上げる。
- 技術などと同じように重要。

セキュリティが破られるとどうなるのかを考えると……

- 受注生産個人商店→しょぼそう
- 政府、軍→やばそう

16.1 [ホワイト|ブラック]ハット集団の社会学的分析

- 今までは専門的な内容に焦点を当てていた。
- 本節では、彼らの社会を形成している構成員に焦点を当てる。
- 彼らはどのように社会形成しているのか？
  - 集団の存在意義
  - 個人、グループの動機
- 社会構造を理解し、構成員の振る舞いや行動を形成する大局的力について仮説を作る。

16.1.1 ハッカー、クラッカー、ブラックハット、ホワイトハット、アイデンティティクライシスと呼び名の力

- ハッカーに身分証はなく、それを名乗るわけにも行かない。
- (ホワイトハット、ブラックハットなどの)アイデンティティを自らにつける際の、ある種の努力も徒労になる。

- ハッカーとはそもそもコンピュータを使う集団を指していた？
- hacker
  n. 斧を使用して家具を作る人
  1. プログラムの詳細やシステムの探求や拡張を楽しむ人
  2. プログラミングを楽しむ人
  3. ハッキングの価値を認識できる人
  4. 短時間でのプログラミングが得意な人
  5. 特定のプログラムの専門家(例:UNIXハッカー)
  6. あらゆる種類の専門家、熱狂者(例:天文学ハッカー)
  7. 創造的に制限を克服したり、知的な挑戦を楽しむ人
  8. 機密性の高い情報を探す悪意ある侵入者(正:clacker)

16.1.1 ハッカー、クラッカー、ブラックハット、ホワイトハット、**アイデンティティクライシス**と呼び名の力

- ハッカーという言葉は、様々な意味があった……が、
- ニュースメディアによって犯罪者的な意味を持たされた。

- この否定的な意味を嫌がる“ハッカー”が、“クラッカー”を定義した。
  …[crack(解読)]より
- 社会的に肯定的に見られたいハッカーが、自身を“ホワイトハット”と呼び、否定されるハッカーを“ブラックハット”とした。
  →そもそもこの帽子すらも嫌がられる。

- “ハッカー”はコンピュータ使用者集団全体に対する汚名？

16.1.2 集団内の動機:個人、グループ それらの行動を理解するためのポイント

- 動機は、行動に至る理由を理解するための重要な要素
- 動機の理解は潜在的不正アクセスの予測を助ける
- ハッカーの動機の起源MICE(MEECES) MICEはMoney, Ideology, Compromise(妥協), Ego
  - Money　　金銭
    - 金銭だけちょっと例外
  - Entertainment 娯楽
  - Ego　　エゴ
  - Cause　　主義(イデオロギー)
  - Entrance　　社会グループへの加入
  - Status　　地位

金銭

- 初期の歴史では技術を悪用して資産を蓄えてはいけないという規範が強かった
  - 個人で小規模に利用される程度(有料サービスの無料使用、大学の成績の変更……)
  - 過激派は集団からも敬遠されていた
- 現在ではかなり様子が変わっている。(ランサムウェア、コインチェック事件……)
  - 脅迫よりもDoS攻撃ビジネスサービス？　ヤクザ的手法？
  - クレジットカード情報の窃盗や売買(通貨化)
- 才能のある個人をブラックハットとして雇用(前職が諜報員だったり)
  - 脅迫されて活動するブラックハットも

- 金で動くハッカーは過去、敬遠されていたが、今では組織・集団に

娯楽

- いたずらして楽しい！
- 会社や政府のWebサイトに違法侵入し、画像やテキストを書き換えたりアダルトサイトに転送したりして楽しむ。
- メールサーバでは、プライベートメールの公衆送信やなりすましメールで嫌がらせを行う。

- すべてのハッカー集団にとって主だった動機だと思われる

エゴ

- ハッカー集団にとってある程度共有されている動機。
- 障害を克服し、問題を革新的に解決することへの満足感。
  - 初心者から達人まで～
- チャレンジングな課題という魅力に勝てない。
  - 悪意はないが、侵入したい。
  - 逮捕や処罰のリスクより、挑戦と成功によるエゴの充足。

主義(イデオロギー)　ハック行動主義

- 様々な要因による
  - 地政学上の適応、文化・宗教的影響、歴史上の出来事、社会問題……
  - 「すべての情報は自由にアクセスできるべき」
    - 会社の電話交換システムの技術情報を取り出して公開、誰でも利用できるように
  - 「高価なソフトウェアは低所得者を冷遇している」
    - ソフトウェアのコピーガードを無効化
  - 「自分の主義と合わないな」
    - テロリスト勢力のWebサイトをアダルトサイトに転送
    - 政党間でWebサイトの外観破壊やDoS攻撃
- 民間サイバー兵士の出現
  - デモや抗議文の提出だけでなく、直接攻撃が可能に
  - 国境に防衛機能はない
  - 国家間の緊張から政府Webサイト破壊合戦、過激になりワームの投入まで
  - より過激に、より執拗に、より被害甚大に……

社会グループへの加入

- 人は社会的ないきもの
- ハッカーも人間なので、社会を形成している。
  - 能力至上主義
  - 似た能力を持つグループが形成される傾向がある
  - 個人と集団の能力に開きがある場合、能力の高い側が集団への参加を望まない
    - 個人<<<集団　　→集団「初心者は混ざらないで…」
    - 個人>>>集団　　→個人「ここは入る価値ないな…」

- 集団に加入するためには、既存のメンバーに技術を評価される必要がある

  →実力誇示のためにエクスプロイトコードを書く

地位

- 自分の実力に応じて地位が変わる能力主義社会
- その実力の評価とはどのように…？　どうやって自分を他人に見せる？
  - 自分がどのようなシステムを持ってるか自慢する
    - 証明のため、説得力をもたせるために何かへの侵害を行う
  - 知識の披露・教授、情報交換
  - 誹謗中傷、ケンカ(相手のマシンを破壊したり、制御を奪ったら勝ち)
    - 不特定多数の他人を巻き込むケースもある
- この動機はハッカー大会で発散されている
  - 直接の対話を含めた手段で、互いの地位の手がかりを得る→動機が発散
- ただし、インサイダによる地位決定とは別
  - 情報システムにおいて、所属する会社よりも知識レベルが高いにもかかわらず一貫性のない行動が取られると動機ができる

16.1.3 ここまでのまとめ

- 動機を検討するときは、6つ(MEECIE)の観点で、それぞれ考える
    - それ以外もあるが、主要なものは6つ
    - Money　　金銭
    - Entertainment　娯楽
    - Ego　　エゴ
    - Cause　　主義(イデオロギー)
    - Entrance　　社会グループへの加入
    - Status　　地位

16.1.4 ホワイトハット/ブラックハット集団の社会構造

- 見た目の印象ほど簡単ではないこの集団について
  - 混沌ではなく、堅牢に社会だっている
  - 注意深く調査する
    - インタビューは有効だが、そも気軽に調査に応じてくれる人は少ない(いない)
- ハッカー集団に対する見解
  - サブカルチャーではなくカウンターカルチャー
    - 文化の価値観が伝統的な社会と完全衝突する。
  - 熱意は経済、政治、社会や軍事的な損害という目的を忘れさせる
  - 違法行為による重大な刑罰を甘く考えている、結果責任を拒否する
  - 自分たちの文化に対する情熱とデジタル世界への好奇心

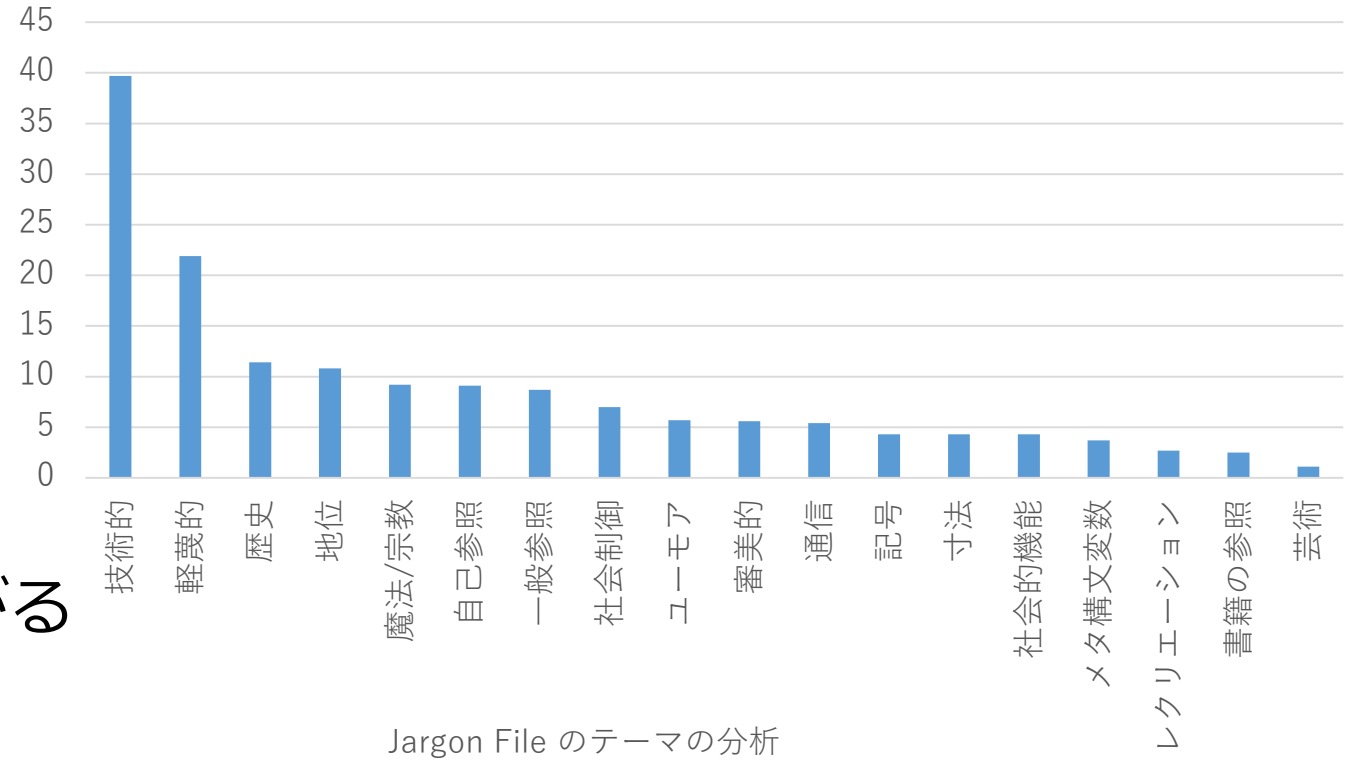
16.1.4 ホワイトハット/ブラックハット集団の社会構造

- 強力な能力主義としてのハッカー集団
  - 厳格な規範があり、価値観は共有され、組織構造は整備され、新たなメンバを見つけようとしている。
  - 集団での地位を外部にも誇示する。
    - 所属集団の主張、ブラックハット通貨の所有量の誇示
  - 会合は仮想的に行われる
    - 特に能力を重視し、グループの能力は均質になりがち。
    - レベルの低い“初心者”の参加は拒否
  - 地位の衝突はオフで解消
    - 解消できないと(できた場合も)内部の派閥争いを起こし、結束力を低下させる。
- ハッカー集団に影響する外部の力
  - 地政学的な力
    - 政治経済によって、能力に見合った職が持てないなど水準の低い暮らしを強要されると、能力の証明を行う必要がある(自尊心の回復)
    - エゴ、イデオロギーや金銭による動機

16.1.4 ホワイトハット/ブラックハット集団の社会構造

- ハッカー集団の社会構造の実像
  - このカウンターカルチャーの歴史や、重要な概念、思想、人々について書かれた記録 “Jargon File”
  - Jargon File をオープンコーディングすると、単語や語句は18通りのテーマに分類できる
    - **技術的**: ハード、ソフト、アルゴリズム、プロセスなど直接技術に関わる
    - **軽蔑的**: 人や物に対して軽蔑的な意味を込めて使用される
    - **歴史**: 重要な過去の出来事や、人やものを表す
    - **地位**: 集団の中で、人、出来事、物を見たときの地位や敬意を表す
    - **魔法/宗教**: 魔法や超自然的な特性を持つ個人、物、常識では説明できない出来事
    - **自己参照**: 人間の持つ特性をコンピュータに当てはめたもの、あるいは擬人的用法
    - **一般参照**: ハッカー社会の物事の表現に、一般的な文化の特徴を用いる
    - **社会制御**: --のプロセスにおいて直接使用される(例: flame, 受信者を嘲笑うE-mail
    - **ユーモア**: 直接的にユーモアを示そうとする
    - **審美的**: 洗練されていると思われる物、出来事、プロセス
    - **通信**: コンピュータ用語を実会話に用いる
    - **記号**: 技術的解釈を超えた記号(“bang” = “!”)
    - **寸法**: 特定の大きさ、単位
    - **社会的機能**: 社会的相互活動の側面を表す
    - **メタ構文変数**: 変数、可変性を表す
    - **レクリエーション**: 娯楽活動を表す
    - **書籍の参照**: 特定の書籍を表す
    - **芸術**: 直喩として芸術的な要素や物を表す

16.1.4 ホワイトハット/ブラックハット集団の社会構造



Jargon File のテーマの分析

- **技術的**なテーマが最も多い
- **軽蔑的**
  - 衝突→メッセージのやり取り
- **歴史**
  - 文化の誕生と成長を記録したがる
- **地位**
  - 社会構造から地位が重要視されていた
- **魔法/宗教**　←!?
  - 「処理Aを実行した際に正しく結果Bを導くか?」という疑問に答えられない
  - 原因と結果に断絶が生じ、再構成できない場合は“魔法”の実例と言える

16.1.5 ここまでのまとめ

- ホワイトハット/ブラックハット集団の社会構造について触れた
- カウンターカルチャーの成長と変化に伴い、多くの事柄が表面化する
- 意外な事柄を知り、固定観念が払拭され、集団の構成や分析洞察が示された