世界のセキュリティコンテスト (Capture the Flag) に挑戦しよう！

by CTF勉強会
愛甲健二

セキュリティコンテストとは？

- セキュリティ技術を競い合う、競技タイプの戦争ゲーム
- 世界では、一般的に”Capture the Flag”（旗取り合戦）と呼ばれる　（CTFと略される）
- 要するに、サーバを攻撃したりサーバを守ったりするだけのとても簡単なゲーム

具体的に何をするのか？

- 基本は、システムの中に隠されたパスワードを、いかに早く発見できるかを競う
- 対象がWebアプリなら、XSSやSQLインジェクションなどを利用する
- 対象がバイナリなら、オーバーフローやアルゴリズムの脆弱な部分などを利用する

どんな問題が出題されるのか？

- 問題は5つくらいの分野に分かれている

☺ 1. Binary　ー マシン語を解析する問題
　 2. Exploit　ー Bin系の脆弱性を探す問題
　 3. Web　　ー Web系の脆弱性を探す問題
☺ 4. Forensic ー バイナリの調査を行う問題
☺ 5. Trivia　ー 業界に関するトリビア問題

- チームを組み、それぞれの得意分野に挑む

Binary問題について

- Binary問題って具体的にどのようなもの？

☺ 1. Binary　－ マシン語を解析する問題
2. Exploit　－ Bin系の脆弱性を探す問題
3. Web　－ Web系の脆弱性を探す問題
4. Forensic　－ バイナリの調査を行う問題
5. Trivia　－ 業界に関するトリビア問題

- DEFCON CTF ‘08のBinary問題は…

Binary問題文

- まず実行ファイルが配布され、以下の問題文が提示される

ihatedns.allyourboxarebelongto.usに、配布したプログラムが動いているよ。さぁ、頑張ってパスワードを取得してくれ！

ファイルの特定 (壱)

- とりあえずもらったプログラムをfileしたら…

```
# file reversing400-b05c8059389c8ade8e1a10314f458be5
reversing400-b05c8059389c8ade8e1a10314f458be5: ELF 32-bit LSB shared
object, Intel 80386, version 1 (FreeBSD), stripped
```

- さらにstringsしたら…

```
# strings reversing400-b05c8059389c8ade8e1a10314f458be5
.....
sysent
module_register_init
__start_set_sysinit_set
.....
```

…なんか.koファイルっぽい

ファイルの特定 (弐)

- 試しにローカル環境でロードしてみると…

```
# kldload -v ./reversing400-b05c8059389c8ade8e1a10314f458be5
KLD reversing400-b05c8059389c8ade8e1a10314f458be5: depends on
captain_kmod – not available
```

- ロードできない…ただ、問題文には…

This code is running on ihatedns.allyourboxarebelongto.us.

- つまり、配布ファイルはシステムの一部であり、単体ではロードできないが、ihatednsサーバでは、システムが正常に動いている

やるべきことは？

- 問題ファイルを静的に解析し、動作を特定（IDAPro必須）
- 解析結果を元にExploitコードを作成
- Exploitコードをihatednsサーバ上で実行し、パスワードを得る　（ihatednsサーバには、別の脆弱性を利用してRemoteから侵入）

難読化の解除 (壱)

問題ファイルをIDAProで開くと…

```
start:
  .text:000009F0    EB FF          jmp short near ptr start+1
  .text:000009F2    C3             retn
  .text:000009F3    4B             dec ebx
```

しかし、最初のjmp命令はtext:000009F1へ
ジャンプするため、実際は、jmp命令後、
「FF C3 4B」という命令が処理される。

難読化の解除 (弐)

同じ逆アセンブルでもアプローチを変えると…

```
start:
  .text:000009F0    EB          db 0EBh
  .text:000009F1    FF C3       inc ebx
  .text:000009F3    4B          dec ebx
```

text:000009F1からの処理を見ると、ebxを加算して減算しているだけ。ebxの値に変化はない。要するに、この4バイトはNOPと同等

システムコールフック (壱)

- カーネル空間に存在するIDTを書き換えることで、システムコールフックが可能 (詳細↓)

システムコールフックを使用した攻撃検出 (**by FFRの金居氏**)

http://www.fourteenforty.jp/research/research_papers/SystemCall.pdf

- 考え方としては、Ring0で行われる関数フックだが、アプリケーションからすると、int 80h命令をフックされているかのように見える

システムコールフック (弐)

- 問題ファイルにIDTを変更する部分がある
  (sidtが目印で、以降の処理がフックコード)

```
.text:00001104                  sidt    fword ptr [ebp-14h]
.text:00001104 ; ---------------------------------------------------------
.text:00001108                  db 0EBh ; d
.text:00001109 ; ---------------------------------------------------------
.text:00001109                  inc     eax
.text:0000110B                  dec     eax
.text:0000110C                  inc     eax
.text:0000110D                  dec     eax
.text:0000110E                  mov     ecx, [ebp-12h]
.text:00001111                  lea     edx, [ecx+400h]
(省略)
```

uprintfまでの道のり (壱)

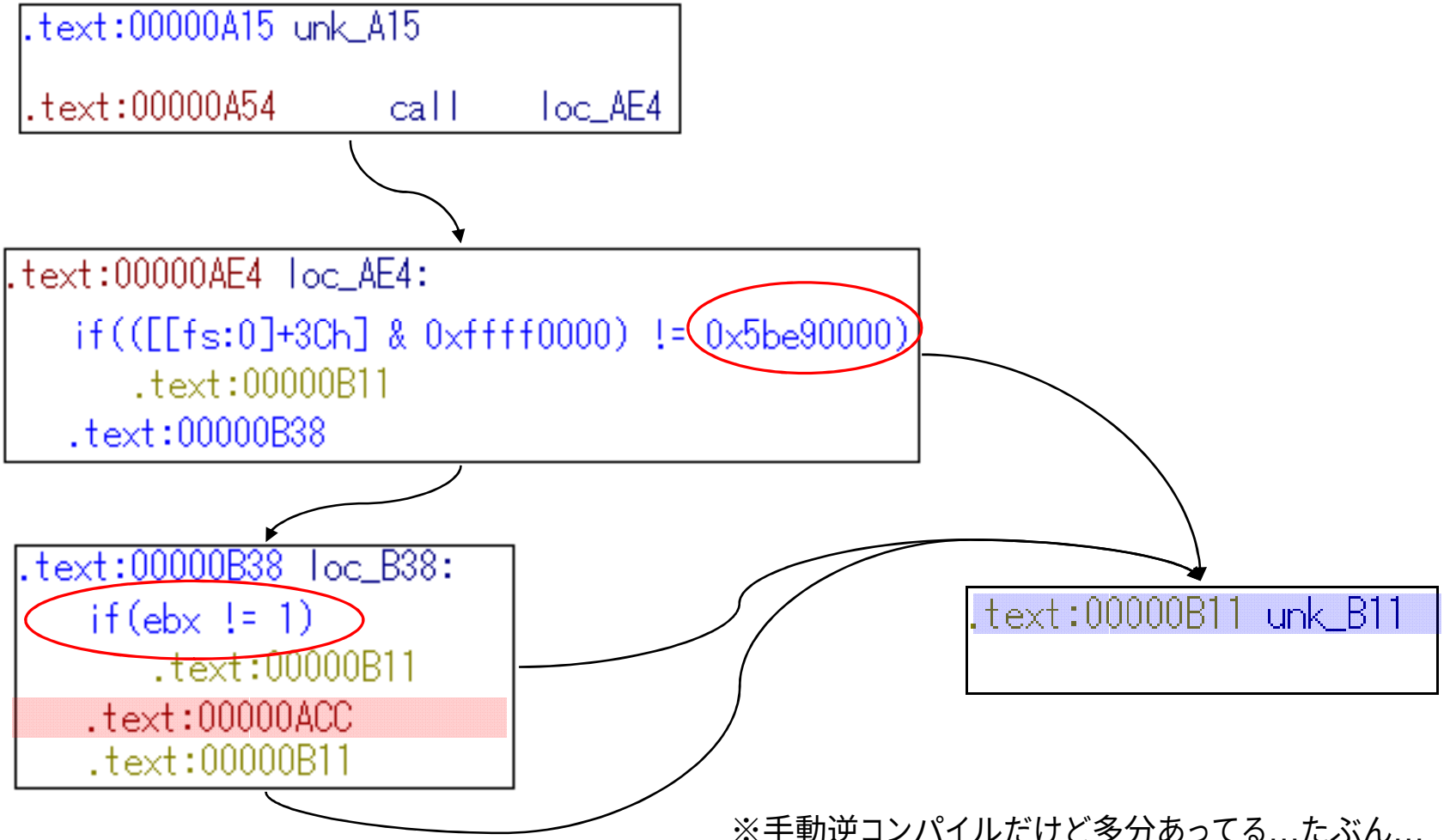
- uprintfが呼び出されている箇所

```
.text:00000ACC loc_ACC:
.text:00000ACC                  push    ebp
.text:00000ACD                  mov     ebp, esp

.text:00000AD5                  push    offset currentpid
.text:00000ADA                  call    uprintf
```

- ここでパスワードが出力されると推測！　さかのぼって、関数関係を調べ上げると…

uprintfまでの道のり (弐)



※手動逆コンパイルだけど多分あってる…たぶん…

uprintfまでの道のり (参)

- どうやらポイントは「text:00000B11」っぽい

```
.text:00000B11 unk_B11  db 0EBh ; ∢

.text:00000B17          movzx   edx, cx
.text:00000B1A          and     ebx, 7
.text:00000B1D          and     ecx, 0FFFF0000h
.text:00000B23          shl     ebx, 10h
.text:00000B26          lea     eax, ds:0[ecx*4]
.text:00000B2D          xor     eax, ebx
.text:00000B2F          or      edx, eax
.text:00000B31          mov     [esi+3Ch], edx
```

- とりあえず、これを元にExploitを作成

Exploitを作成

- 使用される可能性のある15個のシステムコールの呼び出しを総当りし、結果が「0x5be90000」となる全パターンを出力するコードを作成する

- Exploitを含めた詳細を知りたい方へ

http://ruffnex.oc.to/kenji/text/08bin400/DEFCON_CTF_08_Binary_400.pdf
http://ruffnex.oc.to/kenji/text/08bin400/ctf_bin400.zip
http://07c00.com/hiki/hiki.cgi?IDC+Script+for+IDAPro

Exploit実行

```
# cat exploit.c
int main(void)
{
        sendmsg(0, 0, 0);
        getpeername(0, 0, 0);
        close(0);
        close(0);
        close(0);
        read(0, 0, 0);
        open(0, 0, 0);
        exit(1);
}
# gcc exploit.c -o exploit
# ./exploit
BODY MASSAGE!
```

```
# cat exploit2.c
int main(void)
{
        recvfrom(0, 0, 0, 0);
        recvmsg(0, 0, 0);
        close(0);
        close(0);
        close(0);
        read(0, 0, 0);
        open(0, 0, 0);
        exit(1);
}
# gcc exploit2.c -o exploit2
# ./exploit2
BODY MASSAGE!
```

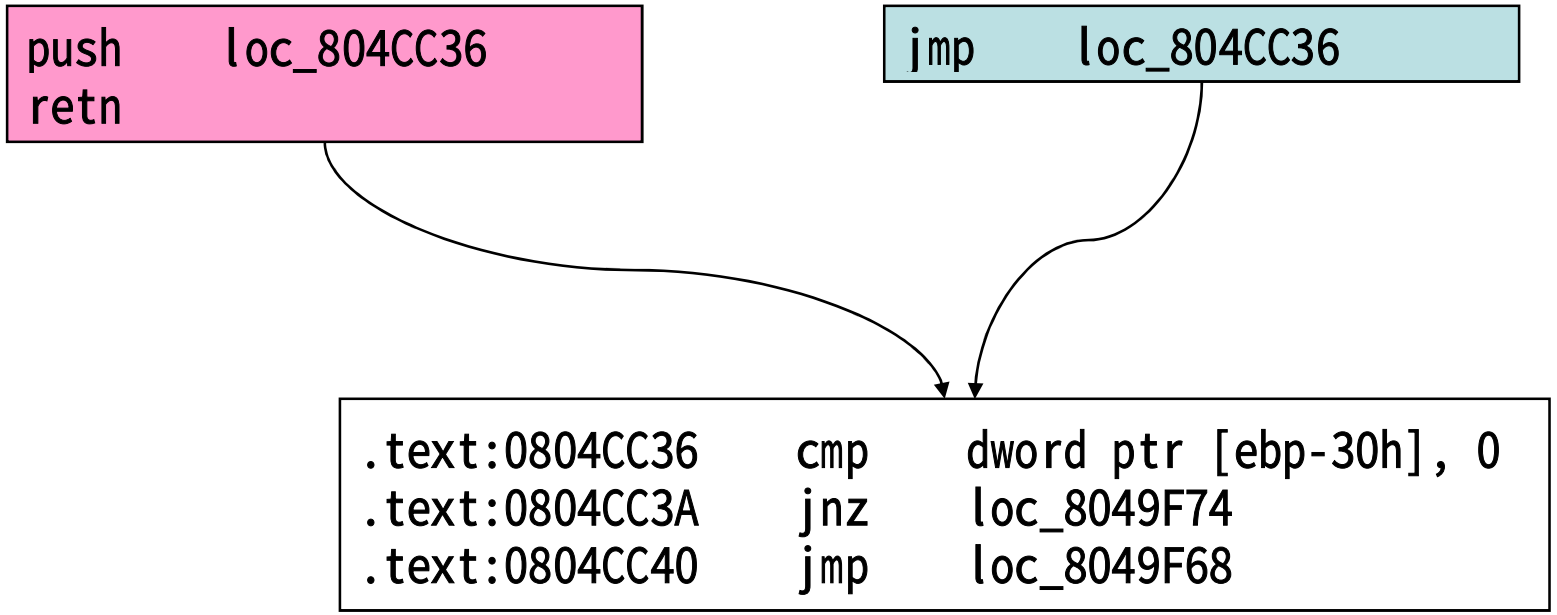
難読化は基本

- 難読化とは、一般的にはアセンブルコードの解析を困難にする技術
- アセンブラを読めるのは当たり前、その次のステップとしての能力が問われる
- 逆アセンブラ（IDAPro）が解析を誤るようなコードを解析する必要がある

難読化パターン (壱)

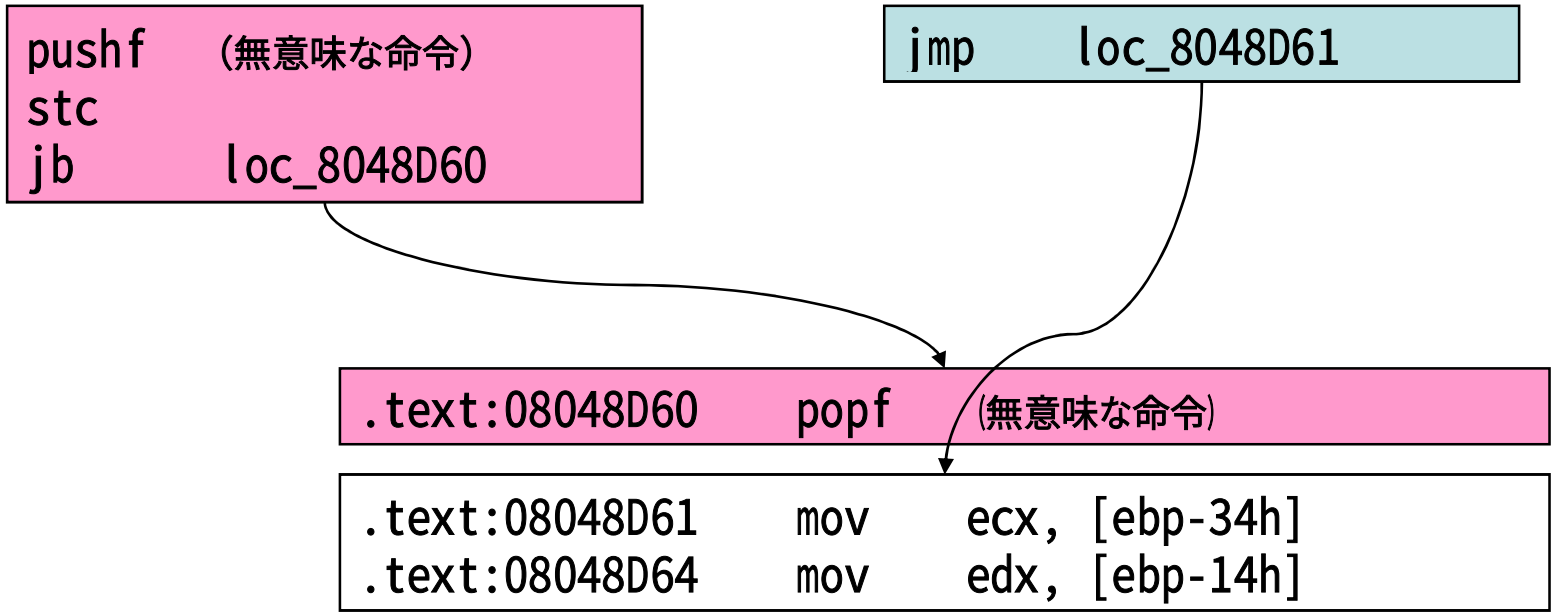
- jmpをpush, retに変換

  赤と青のコードは、どちらも同じ処理 (以後、同様)



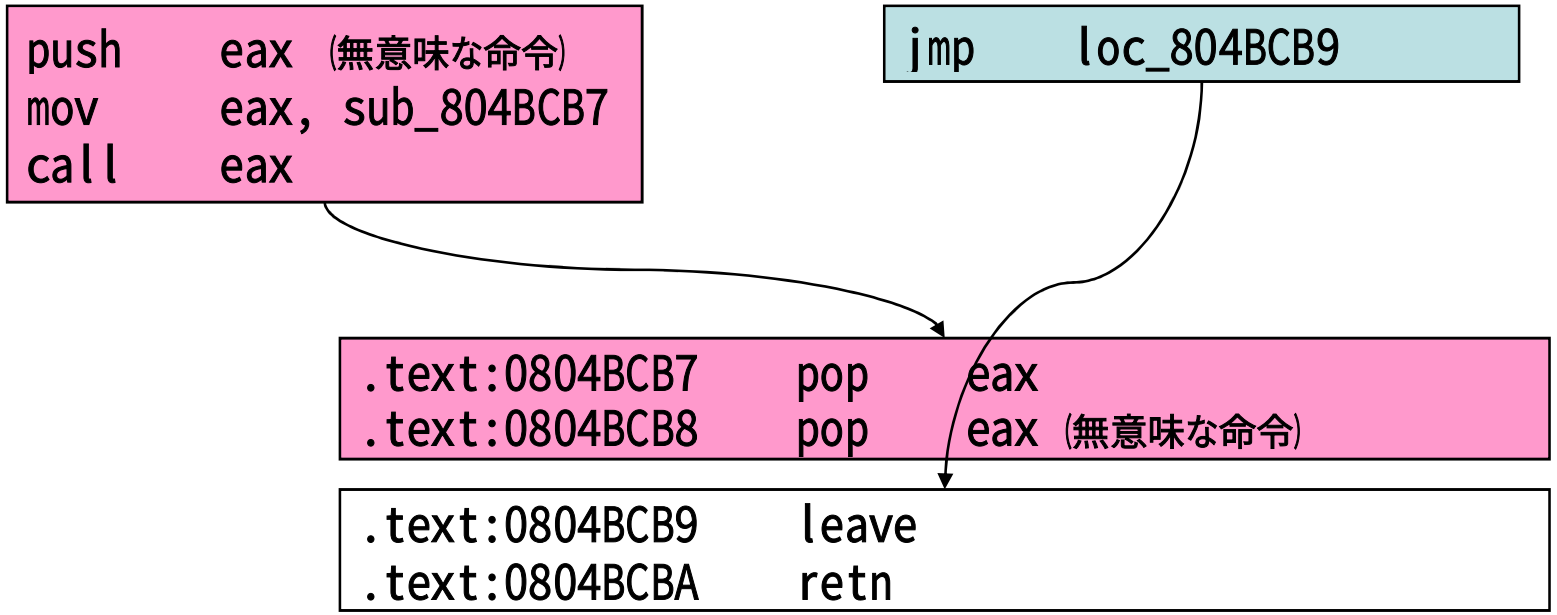
難読化パターン (弐)

- jmpをstc, jbに変換
- 無意味な命令 (pushf, popf) を追加



難読化パターン (参)

- jmpをmov, call, popに変換
- 無意味な命令 (push, pop) を追加



難読化の目的

- IDAProの性能に助けられた解析ができない
- 純粋な解析者としての能力が試される
- 難読化の種類は他にも…
  - 1つの関数を複数に分割して、jmp命令で繋ぐ
  - 関数アドレスのみを渡して、後で呼び出す

続いてForensic問題について

- Forensic問題って具体的にどんなもの？

1. Binary　－ マシン語を解析する問題
2. Exploit　－ Bin系の脆弱性を探す問題
3. Web　－ Web系の脆弱性を探す問題
4. Forensic － バイナリの調査を行う問題
5. Trivia　－ 業界に関するトリビア問題

- DEFCON CTF ‘07のForensic問題は…

Forensic問題文

- まず、あるOSの物理メモリのダンプファイルが配布され、以下の問題文が提示される

> これはあるシステムの物理メモリダンプイメージだが、foo.exeの中にある“forensics challenge”という文字列の仮想アドレスを答えよ

- どうやらダンプファイルから特定のデータの仮想アドレスを求める問題のようだ…

解答までの流れ

- 何のシステム (OS) のメモリイメージであるかを特定する　(Windows? Linux? BSD?)
- 対象プロセス (foo.exe) をメモリイメージから探し、PDBを取得
- PDBと物理アドレスを元に、相対アドレスを突き止める

システム (OS) 特定

```
$ gunzip forensics500.dd.gz
$ strings forensics500.dd ¦ more
….
MDmd@
!This program cannot be run in DOS mode.
$ strings forensics500.dd ¦ grep "Windows“
....
A system file that is owned by Windows 2000 was replaced by an application
```

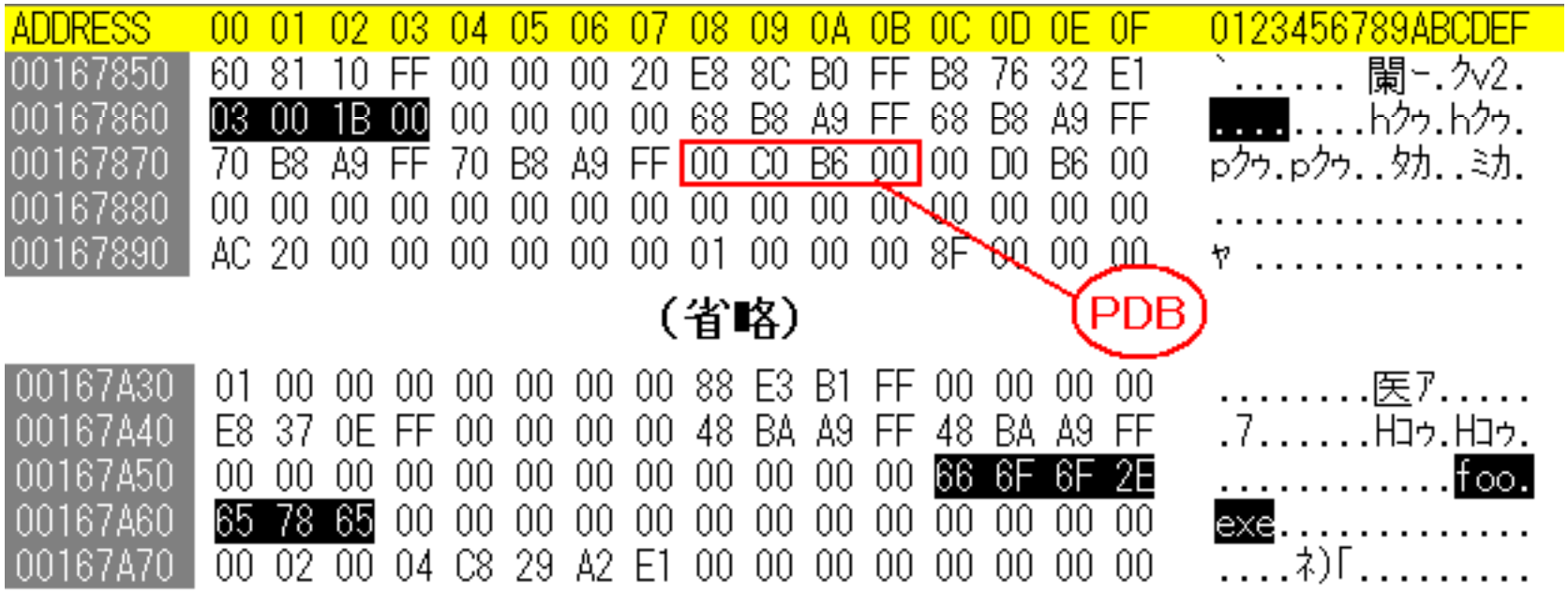
どうやらWindows2000のようだ…

ostest.plを使っても可！

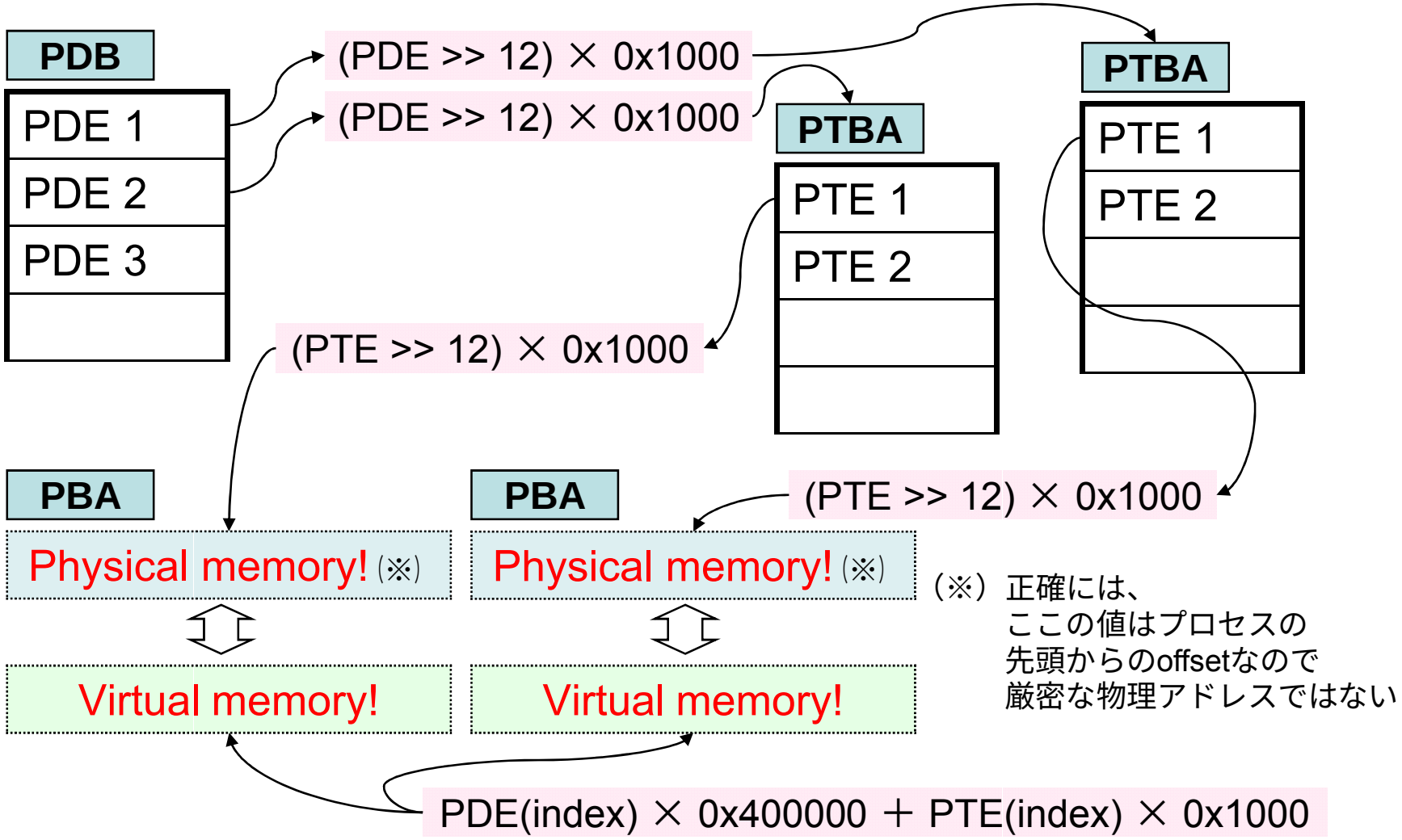
http://sourceforge.net/project/showfiles.php?group_id=164158&package_id=203967

プロセスの特定

- バイナリから「0x03 0xXX 0x1b 0xXX」を探し、対象プロセス (foo.exe) を見つける



PDBから物理／仮想メモリへ



（※）正確には、
ここの値はプロセスの
先頭からのoffsetなので
厳密な物理アドレスではない

foo.exeモジュールを検索

```
$ hexdump -C 0xb6c000.mem ¦ grep "MZ"
00018000  4d 5a 90 00 03 00 00 00  04 00 00 00 ff ff 00 00  ¦MZ..............¦
00022000  4d 5a 90 00 03 00 00 00  04 00 00 00 ff ff 00 00  ¦MZ..............¦
00048000  4d 5a 90 00 03 00 00 00  04 00 00 00 ff ff 00 00  ¦MZ..............¦
0004f5c0  30 83 f9 ff 74 2b 21 45  fc 66 81 39 4d 5a 75 1d  ¦0...t+!E.f.9MZu.¦
$ hexdump -C 0xb6c000.mem ¦ grep "forensics"
0001f030  25 70 0a 00 66 6f 72 65  6e 73 69 63 73 20 63 68  ¦%p..forensics ch¦
0007c030  25 70 0a 00 66 6f 72 65  6e 73 69 63 73 20 63 68  ¦%p..forensics ch¦
$ strings 0xb6c000.map ¦ grep "0x18000"
    0x400000       0x18000    0x1000
```

- 0x400000 (V) と 0x18000 が対応！
- 「forensics challenge」は0x1f034にある！

以上のことから…

解答へ

「forensics challenge」という文字列の仮想アドレスを求める式は…

0x400000 + (0x1f034 - 0x18000) = 0x00407034

解答は「0x00407034」となる

最後にTrivia問題について

- Trivia問題って具体的にどのようなもの？

1. Binary　ー マシン語を解析する問題
2. Exploit　ー Bin系の脆弱性を探す問題
3. Web　ー Web系の脆弱性を探す問題
4. Forensic ー バイナリの調査を行う問題
5. Trivia　ー 業界に関するトリビア問題

- DEFCON CTF ‘06のTrivia問題は…

Trivia問題文

- Triviaは基本的に問題文のみが提示される

x86でいう“ \xEB\xFE ”と同じ意味を持つ、PowerPCのマシン語は何？

- とりあえず、0xEBはjmp命令であるから、PowerPCの命令リファレンスからjmp命令を引っ張ってくれば、解答に辿り付けそう

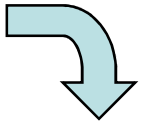
環境によって対処法を考える

- PowerPC環境があるなら…
  - 適当なプログラムを作成し、デバッガで適当なjmp (PowerPCの場合はb命令) を探す
- IDAPro製品版があるなら…
  - 適当なPowerPC用プログラムをIDAProで読み込ませて、b命令を探し出す
- どちらも無いなら…
  - Webに存在するPowerPC関連のリファレンスを探して、jmpに対応するものを見つける

Trivia解答

- PowerPCにおけるjmp命令は”0x48”
- さらに自分自身に飛ぶため、ジャンプ先は”0”

```
00000000  sub_00000000
00000000        b sub_00000000
```



- よって、解答は… “ 0x48 0x00 0x00 0x00 ”

お疲れ様でした

- 以上、主に3分野の内容を解説致しました

☺ 1. Binary　－ マシン語を解析する問題
2. Exploit　－ Bin系の脆弱性を探す問題
3. Web　－ Web系の脆弱性を探す問題
☺ 4. Forensic － バイナリの調査を行う問題
☺ 5. Trivia　－ 業界に関するトリビア問題

- ExploitやWebは、また別の機会にでも…

まとめ

- 世界のエンジニアと切磋琢磨できる
- 問題が解けると、ひとつ成長した気になれる
- 専門分野以外にも興味がわく
- きっと楽しい！

参考資料

Binary 400 問題ファイル
http://nopsr.us/ctf2008qual/reversing400-b05c8059389c8ade8e1a10314f458be5

Forensics 500 問題ファイル
http://nopsr.us/ctf2007prequal/forensics500-88b33eb4b8a2b9a49a632394ba746088.dd.gz

システムコールフックを使用した攻撃検出 (by FFRの金居氏)
http://www.fourteenforty.jp/research/research_papers/SystemCall.pdf

Windows(Forensic -> Memory)
http://www.kazamiya.net/node/18

Linking File Objects to Processes
http://computer.forensikblog.de/en/2009/04/linking_file_objects_to_processes.html#more

Reconstructing the Process Memory
http://computer.forensikblog.de/en/2006/04/reconstructing_the_process_memory.html

DEFCON CTF '07 Forensics 500
http://07c00.com/hiki/hiki.cgi?DEFCON+CTF+%2707+Forensics+500

他多数…

ご清聴、有難うございましたm(_ _)m

Any Questions?