

AntiRE en Masse

Investigating Ferrie's Documented AntiUnpacking Tricks in the World's Worst Mal-Families

Kurt Baumgartner, VP Behavioral Threat Research
PCTools ThreatFire

Presented at Virus Bulletin 2009
<http://www.virusbtn.com/>



Agenda:

- **Ferrie's Virus Bulletin Series – Anti-unpacking Tricks I – VII**
 - **CARO Conference Paper and Presentation**
 - **Packers, Crypto, Compression, AntiRE**
 - **Unpacking Tools**
 - **Malware Anti-unpacking - Worst Families of 2008,2009**
 - **Observations, Implications, Conclusion**
-

Ferrie's Virus Bulletin Papers – Anti-unpacking Tricks Series I-VII

- Seven part Addendum to his 25 page CARO workshop paper
- Anti-debug, anti-emulation, anti-dumping, anti-interception

Pre-Virus Bulletin Series - Ferrie's CARO Presentation and Accompanying Paper

- Four anti-unpacking trick categories
- 25 pages dedicated to anti-dumping, anti-debug, anti-emulation, anti-interception
- 2 pages = anti-dump, 18 pages = anti-debug, 3 pages = anti-emulation, 1 page = anti-interception
- 8 reversing/debugging tools covered specifically
- 4 anti-utilities mentioned

Ferrie's Virus Bulletin Series

Part One

Anti-dump

Anti-debug

Anti-emulation

Part Two

Implemented anti-debug

Part Three

Syser Anti-debug

Part Four

"Speculative" Generic Anti-debug

Part Five

"Speculative" Anti-Ollydbg Tricks

Part Six

"Speculative" Anti-Ollydbg-plugin
Tricks

Part Seven

Mixed bag of "Speculative" Tricks

Ferrie's Papers - Packer Content

Mentions publicly available legitimate and grey (19)

Yoda's Crypter, Armadillo, Shrinker, VMprotect, ExeCrypter, Themida, SafeDisc, HyperUnpackMe2, Yoda's Protector, PC Guard, Invius, ASProtect, tELock, PECompact, MEW, PE-Crypt32, TryGames, ASPack, RLPack

Mentions underground packers (3)

Tibs, MSLRH, NsAnti

Spectrum of “Packers” - Legitimate, Grey, Malicious

Legitimate executable packers compress PE file contents, adding an unpacking stub used to decompress contents at runtime.

Open source gold standard - UPX

“UPX is a free, portable, extendable, high-performance executable packer for several different executable formats.

It achieves an excellent compression ratio and offers very fast decompression.

Your executables suffer no memory overhead or other drawbacks for most of the formats supported, because of in-place decompression.”

<http://upx.sourceforge.net/#abstract>

Themida

“Advanced Windows software protection system, developed for software developers who wish to protect their applications against advanced reverse engineering and software cracking.”

<http://www.themida.com/>

Packers' Characteristics

Shared by grey and malicious “packers”

- Much more than simply compression and runtime decompression
- Anti-debug, anti-dump, anti-interception, anti-emulation
- Layers of protection target reversers' efforts

Specific to malicious packers

- Ease of server side polymorphism
- Frequently built for compatibility with multiple packers of buyer/distributor's choosing
- Distributed throughout layers of underground markets
- Layers of protection most often target file scanner's best efforts

Unpacking Tools and Public Resources

Ollydbg, Titan, Windbg, the Ether project. Syser, SoftICE, ImpRec, LordPE, PETools, OllyDump by Gigapede, FKMA's PE Dumper, AdvancedOlly, StrongOD, InvisibleOD, Syser, IDA Pro

OpenRce.org, Woodmann RCE Library and forums, Google +
Google Translate

Mebroot/Sinowal

Overview

Packed malcode components

Anti-unpacking tricks

Mebroot Overview

- Mebroot is an actively distributed Mbr infecting downloader and dropper component delivered via “outdated” commodity exploit packs and delivering banking password stealers and bot components
- Most functionality documented in last year’s fine “Stoned” paper from Kimmo Kasselín and Elia Florio
- Slight changes in code this year show development in progress, although not on the scale anticipated in the paper

Mebroot Overview (cont.)

- Deployed via multiple exploits, with the most popular targeting Adobe Acrobat Reader vulnerabilities
- The downloader (miniloader) copies out another component at runtime. It is a binary similar to itself but much larger, and calls `CreateProcessA` triggered by an audio event activated as a `timeSetEvent` event callback, previous variants called `SetWinEventHook`

Mebroot Packed Components

- Downloader/miniloader
- Dropped user-mode executable files
- Drivers

Mebrook - Anti-unpacking tricks

- Anti-emulation - packer starts off the entry point with multiple api calls with bogus parameters with flow control based on return codes
These calls usually are functionally based in FileIO and handle fetches.
Changing calls and parameters = server side polymorphism

“Example code looks like this”:

```
DeviceIoControl(IoControlCode = 0, hDevice = Null)
```

```
GetExitCodeThread(hThread = Null, pExitCode = 0012fee0)
```

```
GetFileVersionInfoSizeA(FileName = "", pHandle = 0012fee0)
```

However, the mebrook custom packer implements this same process heap usage across every binary for the past year

Mebroot - Anti-unpacking tricks (cont.)

- Junk code placed in between the functions and the misaligned offset that is jumped into confuses Olly's analysis capabilities. Removing analysis from the module returns proper disassembly. Spaghetti jumps have dried up.
- Crypto is most often modified standard reference implementations
- Api strings are custom crypted, decoded and import table built at runtime

Mebroot - Anti-unpacking tricks (cont.) – Modified XTEA Decryption Loop

dec(z,(((y shl 4) xor (y shr 5)) + y) xor (sum+key[(sum shr 11) and 3]));

0x243f6a88 is ripped here
(proposed key
for Schneier's
Blowfish algorithm)

Anti-crude crypto detection?

```

004067E3  POPFD
004067E4  MOV EBX, [ARG.2]
004067E7  MOV ESI, ECX
004067E9  SHR ESI, 5
004067EC  MOV EDI, ECX
004067EE  SHL EDI, 4
004067F1  XOR ESI, EDI
004067F4  PUSHFD
004067F6  JMP SHORT uninstal.0040681B
004067F7  POPFD
004067F7  SUB [ARG.1], 243F6A88
004067FE  ADD ESI, ECX
00406800  XOR ESI, EDI
00406802  SUB EAX, ESI
00406804  PUSHFD
00406805  PUSH EAX
00406806  MOV EAX, EDX
00406808  AND EAX, 0
0040680B  CMP EAX, 0
0040680E  POP EAX
0040680F  JE uninstal.004068C2
00406815  ADD EDI, ESP
00406817  SUB EDI, ESP
00406819  POPFD
0040681A  RETN
0040681B  POPFD
0040681C  MOV EDI, [ARG.1]
0040681F  SHR EDI, 0B
00406822  AND EDI, 3
00406825  MOV EDI, DWORD PTR DS:[EBX+EDI*4]
00406828  ADD EDI, [ARG.1]
0040682B  PUSHFD
0040682C  PUSH EAX

```


Waledac

Overview

Packed malcode components

Anti-unpacking tricks

Waledac Overview

- Currently active, simply changed operational model from exclusively themed spam campaigns
- Waledac packer distributed with other software than spambot
- Packer targets both file scanners and reversers

- Currently 2 packed malcode components most active in India and Great Britain:
 - Current downloader
 - Email address harvesting, P2P chatting, DDoS'ing, spamming bot

- Packer's anti-unpacking tricks
 - Multiply aligned entry point
 - Int 0x2e, Int 3
 - Time locks

Waledac (Bredolab?) Downloader Overview

- Served via “poorly filtered” affiliate distribution network, i.e. `pubcut[autogen].59.to/clickcontroller/9006/files/ “8c01.tmp”`
- Automated domain generation scheme:
`pubcutranrat.59.to, pubcutpopgot.59.to, pubcutpopgot.59.to,etc`
Interesting that we now have downloaders served from Tongan (African) registered domains retrieving malicious file payloads from servers hosted in Amsterdam and the Russian Federation
- Nginx/0.8.15 http servers
- Currently downloads Waledac spambot and FakeAv
 - `95.211.8.215/pr/pic/abc_c.exe (%temp%_EX-08.exe = Waledac spambot)`
Russian Federation, phones back to `http://topwale.com/index.php`
 - `91.212.220.123/pr/pic/spyware.exe (Antivirus 2008/9 variant)`
Amsterdam, Netherlands

Waledac Downloader - Overview (cont.)

- ~20kb in size delivered via client side
- Server side polymorphism
- Entry point trick + time lock puzzle

Waledac Downloader - Anti-unpacking tricks

- Packer EP begins with Int 0x2e 0xc0000005 location generation trick

Anti-debug and anti-emulation

Persistent AVkill

Custom encoding and standard reference crypto

Waledac Downloader AntiRE Tricks – Anti-Debug/Anti-Emulation

- Hidden” Int 2e eax = 0x0000 0000, edx = 0x02eb 2ecd
- Debugger Anti-attach, much like a sysenter
- Predictably returns 0xc000 0005 in eax and the location of the offending call in edx, both required to transfer control to expected location

```

00401058 8c01.<ModuleEnt 81C1 35547489  add    ecx, 89745435
0040105E          33C0          xor    eax, eax
00401060          BA CD2EEB02  mov    edx, 2EB2ECD
00401065          EB FA          jmp    short 8c01.00401061
00401067          C1C8 15      ror    eax, 15
0040106A          2BD0          sub    edx, eax
0040106C          81C2 974E0000 add    edx, 4E97
  
```

- Edx return value without debugger vs. debugger stepping

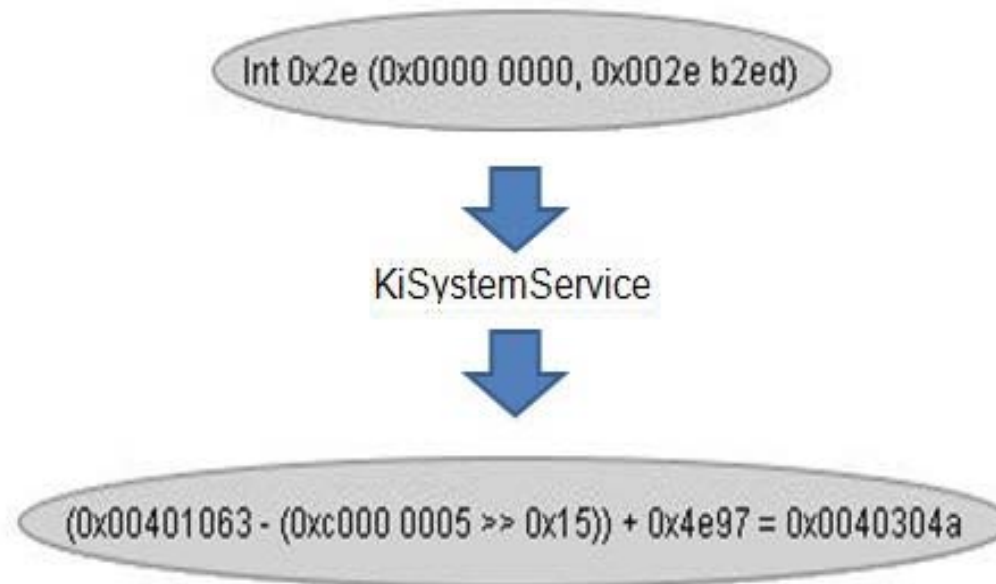
| | | | | | |
|---------------|------|-------------------------|---------------|--|--|
| CD 2E | int | 2E | | | |
| EB 02 | jmp | short 8c01.00401067 | | | |
| EB FA | jmp | short 8c01.00401061 | | | |
| ^ EB FA | | | | | |
| C1C8 15 | ror | eax, 15 | | | |
| 2BD0 | sub | edx, eax | | | |
| 81C2 974E0000 | add | edx, 4E97 | | | |
| 52 | push | edx | | | |
| FF1424 | call | near dword ptr ss:[esp] | 8c01.004030FA | | |
| 60 00 | push | 00 | | | |

Registers (FPU)

| | |
|-----|------------------------|
| EAX | 00002E00 |
| ECX | 0012FFC4 |
| EDX | 004030FA 8c01.004030FA |
| EBX | 7FFDF000 |
| ESP | 0012FFC0 |
| EBP | 0012FFF0 |
| ESI | 00000000 |

Waledac Anti-emulation/anti-debug KiSystemService predictable error condition

Visual/mathematical representation of the EP anti-debug/anti-emulation



Anti-debug/Anti-emulation (cont.)

- Fcomp ebx → Illegal use of a register
(Compare Floating Point Values and Set EFLAGS)

- How to handle in illegal register use in Olly?
Options -> Debugging Options -> Security ->
Allow stepping into 'Unknown commands'

Repeated useless MMX instructions intermixed

- Int 3 cc breakpoint exception can be mishandled by Olly

Matt Pietrek's SEH discussion:

<http://www.microsoft.com/msj/0197/Exception/Exception.aspx>

Waledac Downloader Simple Decoding Loops

Simple ROL "0x0e9" loop
surrounded by garbage code

```

00403249 MOV EAX,EDX
0040324B INC EAX
0040324C ROL DWORD PTR DS:[ECX],0E9
0040324F LEA EAX,DWORD PTR DS:[EAX]
00403251 CLD
00403252 MOV EBX,ESI
00403254 LEA EDX,DWORD PTR DS:[2D4B384A]
0040325A CMOVE EDX,ESP
0040325D AND DX,DX
00403260 PUSH EDX
00403262 DEC ESI
00403263 POP ESI
00403264 SAL ESI,3C
00403267 ADD ECX,4
0040326A PUSH -5
0040326C JF 8c01.00403273
00403272 INC EAX
00403273 POP EAX
00403274 STD
00403275 NOT EAX
00403277 NOT EAX
00403279 NOT EAX
0040327B LEA EAX,DWORD PTR DS:[7E126383]
00403281 LEA EAX,DWORD PTR DS:[EAX]
00403283 LEA EAX,DWORD PTR DS:[EAX]
00403285 CMP ECX,EDI
00403287 JL 8c01.004031FC
0040328D XCHG EAX,EAX
0040328F XCHG EDX,EAX
  
```

Waledac Downloader Decoding/Decryption Schemes

Modified TEA decryption loop
following simple rol loop

- Step into this decoded data
 - TEA loop 20 rounds
 - $\text{delta} = \text{x}9\text{e}3779\text{b}9$;
 - $\text{sum} = \text{delta} \ll 5$;
- ```

for (j=0; j<=31; j++) {
 z -= (y << 4)+key[2] ^ v+sum ^ (y >> 5)+key[3];
 y -= (z << 4)+key[0] ^ z+sum ^ (z >> 5)+key[1];
 sum -= delta;

```

```

PUSHAD
MOV EDI,DWORD PTR SS:[ESP+24]
MOV ESI,DWORD PTR SS:[ESP+2C]
MOV ECX,DWORD PTR SS:[ESP+28]
PUSHAD
MOV EBX,DWORD PTR DS:[EDI]
MOV ECX,DWORD PTR DS:[EDI+4]
MOV EAX,9E3779B9
SHL EAX,5
PUSH EDI
PUSH 20
POP EDI
MOV EBP,EBX
SHL EBP,4
ADD EBP,DWORD PTR DS:[ESI+8]
MOV EDX,EBX
XOR EDX,EBX
SHR EDX,5
ADD EDX,DWORD PTR DS:[ESI+C]
XOR EBP,EDX
SUB ECX,EBP
MOV EBP,ECX
SHL EBP,4
ADD EBP,DWORD PTR DS:[ESI]
MOV EDX,ECX
ADD EDX,EAX
XOR EBP,EDX
MOV EDX,ECX
SHR EDX,5
ADD EDX,DWORD PTR DS:[ESI+4]
XOR EBP,EBP
SUB EAX,9E3779B9
DEC EDI
JNZ SHORT 8c01.00403623
POP EDI
MOV DWORD PTR DS:[EDI],EBX
MOV DWORD PTR DS:[EDI+4],ECX
POPAD
ADD EDI,8
SUB ECX,7
LOPD SHORT 8c01.00403611
POPAD
RETN 0C

```

## Waldec Spambot Custom Crypto

- Decryption routines littered with garbage code (i.e. from smsspy.exe )

Underlying simple decryption cipher function embedded within 140 “active” function-less instructions looks like this:

```
mov bl, byte ptr ds:[esi]
ror bl, 4
rol bl, 1A
add bl, 0B
xor bl, 0D9
mov bh, bl
mov byte ptr ds:[esi], bh
```

```
for (i=0; i<0x669; i++) {
 x = y[i];
 y[i] = (((x >> 4) << 0x1a) + 0x0b ^ 0x0d9);
}
```

## Waledac Spambot Anti-Unpacking Tricks

~380 - ~420 kb, binary size is generally based on size of p2p peer list

- **Int 0x2e calls with invalid eax/edx parameters**

Code looks like this:

```
int 2e ; eax = 0xffffffff, edx = 0x0007f800
edx, ABCE2546 ; eax = 0xc000001c, edx = 0x0046687a
jnz 20090422.004669C6 ; "should" always jump to 0x004669c6
inc edi
```

## Waledac Spambot Anti-Unpacking Tricks (cont.)

Entry point stack manipulation for fake system thread re-entry followed by time-lock puzzle (i.e. onlyyou.exe)



[Click here](#) to view your card.

## Waledac Spambot Anti-Unpacking Tricks (cont.)

Entry point stack manipulation + system thread re-entry followed by time-lock puzzle (i.e. onlyyou.exe), the two look like this...

```
mov ecx, dword ptr ds:[ebx+8] ; stack [0006FFC0]=0006FFF0
add ecx, 8
mov ebx, 00460013 ; several bytes after ep
mov dword ptr ds:[ecx], ebx ; stack location for ret from kernel32
Jmp esi ; kernel32.77E7EB56
_BaseProcessStart -> ret -> 00460013
```

## Taterf...

# Packed.NsAnti/Frethog/Trojan.Lineage/Gampass

Overview

Packed malcode components

Anti-unpacking tricks

Flowchart

---

## Taterf/Packed.NsAnti/Gampass Overview

- Worm that drops and injects gaming password stealing components
- High prevalence partly due to its autorun delivery mechanism, partly because gamers hate giving up clock cycles and memory to security software
- Loads of access violations in its worm/dropper from only a few virtual locations
- Avkill abusing "undocumented" call to ZwQuerySystemInformation with parameter 0x0b, driver load via ZwLoadDriver
- Actively hosted and distributed across China and India
- File names stay unusually consistent for extended periods of time, amvo.exe, xmg.exe, help.exe, am.exe, klif.sys, etc



## **Taterf/Packed.NsAnti/Gampass Packed Components**

- Worm/dropper component
- Multiple dropped dlls

## Taterf/Packed.NsAnti/Gampass Anti-unpacking Tricks

- Custom crypto
- Multiple stages, loading and unpacking of code and dll's
- Thousands of access violations while it searches though memory
- Past finding OEP at each stage, no huge difficulty getting past first couple forced exceptions:  
EXCEPTION\_INT\_DIVIDE\_BY\_ZERO  
EXCEPTION\_SINGLE\_STEP

# Koobface

Overview

Packed malcode components

Anti-unpacking tricks

## Koobface Overview

- Identity/authentication, captcha cracking (via “online services”), password stealing worm exploiting multiple social networks’ trust, ease of connectivity and recognition confusion
- Often downloaded alongside other malware deliverables, like zbot, FakeAv, updated Virut variants, etc
- Most prevalent in US, Great Britain, Italy, etc over the past year
- Tweets!

## Koobface Packed Malcode Components

- Most prevalent binaries = Upx packed
- Break up in-memory strings by assigning string array char by char, breaking up strings and concatenating via sprintf
- Nothing from Ferrie's papers
- Uses some OLE for HTML insertion, making reversing difficult but no antiRE

## Zbot...Infostealer.Banker/Wsnpoem/Notos Overview

- Zbot is the server bot component of the Zeus crimeware kit being actively developed and distributed
- Cost - last week, posts offered weekly zbot FUD service for \$1400. Distributors offer “deep” interests more service and effort for more money
- Attack distribution – mostly in the U.S., England, Russia

## Zbot Packed Malcode Components

- Dropper/injector
- Injected malcode
- Anti-unpacking tricks per outer layer  
custom built cryptor  
layers depending on distributor

## Observations, Implications, Conclusions

- Custom packers coupled with “tweaked” encryption algorithms and Implementations
- Limited set of anti-unpacking tricks implemented in active families
- Limited to tricks that most cause problems for performance-sensitive file scanners
- Surprisingly low use of “the ultimate” VM implementation at this point