Volatile IOCs for Fast Incident Response

Takahiro Haruyama (@cci_forensics)
Internet Initiative Initiative Japan Inc.
Overview

- Background
- Volatile IOCs
- Discussion
- Wrap-up
BACKGROUND
Background

- Malware makes advance for persistence
  - Evading AV detection
    - e.g., SpyEye web panel provides AV tester for generated bots
  - Reliable installation
    - e.g., ZeroAccess suspends (and disables if possible) OS security functions before its installation

- We can’t avoid malware infection
  - If infected once, forensic investigation and malware analysis are needed
Incident Response against Malware Infection Case

- Memory Forensics
- Acquisition
- Analysis
- Several Hours

- Disk Forensics
- Acquisition
- Analysis
- Several Days

- Malware Analysis
- Dynamic Analysis
- Static Analysis
Incident Response against Malware Infection Case

Important for fast triage

Several Hours

Several Days
Memory Forensics for Investigating Malware

Experts
Identify the kind of malware
- figure out malware behavior early

Intermediate analysts
Find suspicious process/driver/connection
- useful, but not sure

Experience/knowledge required
IOC (Indicator of Compromise)

- Technical characteristics of known threats from an expert point of view
- OpenIOC [1]
  - An extensible XML schema for defining IOC
  - It enables to share expert knowledge easily

Technical characteristics

- file system
- registry hive
- network, etc...
Motivation

- Malware identification on memory forensics phase is effective for fast triage
  - OpenIOC fills the gap between experts and ordinary analysts

- Problems of existing IOCs
  - Most IOCs depend on non-volatile evidence
  - Difficult to reuse for identifying variants
    - e.g., filename, hash value, IP, URL

Define IOCs based on only volatile evidence & make them reusable
VOLATILE IOCS
Volatile IOC Generation Cycle

1. Extract indicators from analysis result
2. Define IOC using IOC Editor [2]
3. Analyze RAM using Redline [3]
4. Check the IOC report
5. Find the cause of false positive/negative
OpenIOC Indicators in RAM

- What kind of indicators can be defined?
  - ProcessItem
  - DriverItem
  - HookItem
  - ModuleItem (not sure)

- Useful indicators
  - sign of code injection
  - imported/exported functions
  - strings
    - header signature of data structure
    - function/protocol related string
    - de-obfuscated string
  - process arguments
  - etc...
Volatile IOC Examples

- Generated volatile IOCs for famous malware
  - ZeuS
  - SpyEye
  - PoisonIvy
  - ZeroAccess

- Used several samples per single-species malware for the IOC generation
  - Memory images were acquired on Windows XP / 7
ZeuS

ZeuS is a crimeware kit that was first discovered around 2007
- It steals IDs like online banking accounts by using web injection
- 2.0.8.9 source code was leaked in 2011
  - Several variants are discovered
- In more detail, see our report [4]
ZeuS: Overlay

- ZeuS generates install information called PESETTINGS
- PESETTINGS is encrypted and saved with header as “overlay”
  - The header signature “DAVE” used

```
00 push size RC4KEY ; size
push [ebp+rc4Key] ; src
FF+lea ecx, [ebp+rc4key_copied]
push ecx ; dst
FF call Mem::copy(void *, void const *, ulong)
FF+lea ecx, [ebp+rc4key_copied]
push ecx
mov edx, eax
mov ecx, esi
FF call Crypt::rc4(void *, ulong, Crypt::RC4KEY *)
```

Signature “DAVE”
ZeuS: Code Injection

- ZeuS injects itself into other processes
- Code-injected memory region has two differences in memory management information (VAD: Virtual Address Descriptor)

VAD has "protection flag" about read/write/execute

Usual exe/dll includes a pointer to File Object (memory mapped file)
ZeuS: Code Injection Sign in Memory Forensic Tool

- File Object information = null
- protection flag = EXECUTE_READ_WRITE
ZeuS: Imported Functions

- Imported functions indicate functions of the malware
  - Collecting information of infected machines (e.g., GetLengthSid)
  - Anti-Forensics (e.g., SetFileTime)
  - Code injection (e.g., CreateToolhelp32Snapshot)
  - Web injection (e.g., HttpSendRequest*)
ZeuS: Obfuscated Strings

- ZeuS obfuscates important strings
  - The strings are decoded on the execution
  - De-obfuscated strings are good indicators

```c
struc_encoded_str <5Dh, 0Ah, 4289C4h> ; (no xref) image/jpeg
struc_encoded_str <0B3h, 2Dh, 4289D0h> ; Software¥Microsoft¥Windows¥Currentversion¥Run
struc_encoded_str <5, 3Bh, 428A00h> ; SOFTWARE¥Microsoft¥Windows NT¥CurrentVersion¥ProfileList¥%
struc_encoded_str <2Bh, 10h, 428A3Ch> ; ProfileImagePath
struc_encoded_str <14h, 0Fh, 428A50h> ; unknown¥unknown
struc_encoded_str <8Dh, 4Ch, 428A60h> ; :d
  ; rd /S /Q "%s"
  ; rd /S /Q "%s"
  ; if exist "%s" goto d
  ; if exist "%s" goto d
struc_encoded_str <0BAh, 1Dh, 428AB0h> ; grabbed¥%S_%02u_%02u_%02u.txt
struc_encoded_str <0D7h, 19h, 428AD0h> ; Grabbed data from: %s

struc_encoded_str <14h, 30h, 428AECh> ; %s%s
  ; Referer: %s
  ; User input: %s
  ; %sPOST data:
```
ZeuS: Indicators of Some Variants

- "imodule" variant [5]
  - Many obfuscated strings are added
    - e.g., New command downloads DLLs and execute their functions
- Citadel variant
  - Not infect Russian/Ukrainian PCs (GetKeyboardLayoutList)
  - Detect sandboxes for anti analysis

```c
struc_encoded_str <16h, 15h, 429190h> ; (no xref) user_activate_imodule
struc_encoded_str <0E4h, 14h, 4291A8h> ; (no xref) user_restart_imodule
struc_encoded_str <77h, 0Eh, 4291C0h> ; (no xref) user_start_syn
struc_encoded_str <13h, 0Dh, 4291D0h> ; (no xref) user_stop_syn
struc_encoded_str <96h, 13h, 4291E0h> ; (no xref) user_start_ssh_scan
```

```c
push 2Ah ; *saespase*
lea esi, [ebp+wstr_safespace]
pop eax
```

```c
call CryptedStrings:::getW(ushort,wchar_t *)
push 29h ; *bufferzone*
lea esi, [ebp+wstr_bufferzone]
pop eax
```

```c
call CryptedStrings:::getW(ushort,wchar_t *)
push 27h ; *virtualbox*
lea esi, [ebp+wstr_virtualbox]
pop eax
```

```c
call CryptedStrings:::getW(ushort,wchar_t *)
```
### ZeuS: IOC

<table>
<thead>
<tr>
<th><strong>AND</strong></th>
<th><strong>OR</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Process String contains DAVE</td>
<td>Process Section Injected contains True</td>
</tr>
<tr>
<td><strong>AND</strong></td>
<td></td>
</tr>
<tr>
<td>Process Section Imported Function contains GetLengthSid</td>
<td>Process Section Imported Function contains SetFileTime</td>
</tr>
<tr>
<td>Process Section Imported Function contains CreateToolhelp32Snapshot</td>
<td>Process Section Imported Function contains WriteProcessMemory</td>
</tr>
<tr>
<td>Process Section Imported Function contains CreateRemoteThread</td>
<td>Process Section Imported Function contains HttpSendRequestW</td>
</tr>
<tr>
<td>Process Section Imported Function contains HttpSendRequestExW</td>
<td>Process Section Imported Function contains HttpSendRequestA</td>
</tr>
<tr>
<td>Process Section Imported Function contains HttpSendRequestExA</td>
<td></td>
</tr>
<tr>
<td><strong>OR</strong></td>
<td><strong>OR</strong></td>
</tr>
<tr>
<td>Process String contains ZCID:</td>
<td>Process String contains *.txt</td>
</tr>
<tr>
<td>Process String contains if exist &quot;$s&quot; goto d</td>
<td>Process String contains if exist &quot;$s&quot; goto d</td>
</tr>
<tr>
<td>Process String contains if exist &quot;$s&quot; goto d</td>
<td></td>
</tr>
</tbody>
</table>

- **“overlay” signature**
- **code injection sign**
- **imported functions**
- **de-obfuscated strings**
- **indicators for “imodule” variant**
- **indicators for Citadel variant**
3 variants used

- 2.0.8 (source-code leaked version)
- “imodule” variant
- Citadel
SpyEye

- Another Crimeware Kit after ZeuS
- 1.3.45 builder cracked in 2011
  - IIJ analyzed and reported its behavior \([6]\)
- No update since 1.3.48?
  - But the botnet is still active
SpyEye: Resource Data

- “C*” includes data
  - C1: install configuration
  - C2: config.bin
  - C3: password for decrypting config.bin
- “SC*” includes code
  - SC1: code for injection
  - SC2: code for collecting C1 data
SpyEye: C1

- SC2 uses C1 signature “!EYE”
SpyEye: API hash

- SpyEye uses 4-byte hash value for calling APIs
- Imported functions are not useful for IOC
SpyEye: Exported Functions

- SpyEye includes plugin DLLs in config.bin
- The exported functions are unique

Most SpyEye variants include this function exported by the plugin
SpyEye: Obfuscated Strings

- De-obfuscated strings on execution are good indicators

```
push    edi    ; nSize
lea     eax, [ebp+Buffer]
push    eax    ; lpBuffer
push    48707389h    ; getting string "flymode"
call    cci_xor_specified_by_dword_value
pop     ecx
push    eax    ; lpName
call    ds:GetEnvironmentVariableA
test    eax, eax
jz      short loc_4040CD
```

```
+call    GetModuleFileNameA
push    0BE037055h    ; getting string "explorer.exe"
call    cci_xor_specified_by_dword_value
pop     ecx
push    eax
lea     eax, [ebp+Filename]
push    eax
+call    StrStrIA
```
SpyEye: Sending stolen data

- Fixed format with characteristic strings
## SpyEye: IOC

### C1 signature

- Process String contains `!EYE`
- `OR`
  - Process Section Injected contains `True`
  - `OR`
    - Process Section Exported Function contains `SpyEye_Init`
    - Process Section Exported Function contains `SpyEye_Start`
    - Process Section Exported Function contains `SpyEye_Stop`
    - Process Section Exported Function contains `TakeGateToCollector`
    - Process Section Exported Function contains `TakeBotGuid`
- `OR`
  - `AND`
    - Process String contains `hooked_func`
    - Process String contains `process_name`
    - Process String contains `func_data`
  - `OR`
    - Process String contains `flymode`
    - Process String contains `POP3 : %s:%s@%s`
    - Process String contains `ftp://%s:%s@%s`
SpyEye: Result

- 3 versions used
  - 1.3.10
  - 1.3.45 (compiled using builder)
  - 1.3.48
PoisonIvy

- RAT (Remote Access Tool) used for targeted attack
  - Everyone can download it at the web site
  - The distributed version is 2.3.2
    - It cannot run on Windows 7
      - Some hackers patched to work well
    - Some custom-built versions also exist
PoisonIvy: Fragmented Code Injection

- PoisonIvy doesn’t inject its entire image [7]
  - Redline cannot detect the injection
- Specify the description of protection flag directly
  - Caution: Redline 1.9.1 cannot display the memory regions of fragmented code injection 😞

<table>
<thead>
<tr>
<th>Trust Status</th>
<th>Name</th>
<th>Injected</th>
<th>Protection</th>
<th>Region Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undetermined</td>
<td>EXECUTE_READ NoChange SecNoChange</td>
<td>800 Kilobytes (819,200 Bytes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Undetermined</td>
<td>EXECUTE_READ NoChange SecNoChange</td>
<td>3 Megabytes (3,145,728 Bytes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Undetermined</td>
<td>EXECUTE_READ NoChange SecNoChange</td>
<td>1 Megabyte (1,048,576 Bytes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Undetermined</td>
<td>EXECUTE_READWRITE PrivateMemory MemCommit SecNoChange CopyOnWrite</td>
<td>4 Kilobytes (4,096 Bytes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Undetermined</td>
<td>EXECUTE_READWRITE PrivateMemory MemCommit SecNoChange CopyOnWrite</td>
<td>4 Kilobytes (4,096 Bytes)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Found in 75%... \Device\HarddiskVolume1\WINDOWS\system32\ole32.dll
- No Digital Signing... \Device\HarddiskVolume1\Program Files\Internet Explorer\iexplore.exe
- No Digital Signing... \Device\HarddiskVolume1\WINDOWS\system32\ertutil.dll
PoisonIvy: Behavior

- Most PoisonIvy specimens injects twice
  - explorer.exe
    - create iexplore.exe process and inject code to it
  - iexplore.exe (with "-nohome" argument)
    - connect to Poison Ivy GUI client
- Rare specimens are stand-alone
PoisonIvy: Behavior (Cont.)

- PIC (Position-Independent Code)
  - Imported functions cannot be used
- Codes for most functions are sent from client
  - There are few strings related to functions in the installed binary
- For identifying rare stand-alone specimens, use some strings in RAM based on heuristics
  - Not logical, but useful to some degree

```
call [esi+pi_struc.field_b1_kernel32_VirtualAllocEx]
```
PoisonIvy: IOC

- the 2nd injection

  - OR
  - AND
    - AND
      - Process Name is IEXPLORE.EXE
      - Process Arguments contains -nohome
    - OR
      - Process String contains SOFTWARE\Microsoft\Windows\CurrentVersion\Run
      - Process String contains Software\Microsoft\Active Setup\Installed Components\n    - OR
      - Process Protection contains EXECUTE_READWRITE PrivateMemory MemCommit SecNoChange
      - Process Protection contains EXECUTE_READWRITE PrivateMemory MemCommit MultipleSec
  - AND
    - Process Name is Explorer.EXE
    - OR
      - Process Protection contains EXECUTE_READWRITE PrivateMemory MemCommit SecNoChange
      - Process Protection contains EXECUTE_READWRITE PrivateMemory MemCommit MultipleSec
    - AND
      - Process String contains CONNECT %s:%i HTTP/1.0
      - Process String contains ?503
  - AND
    - Process String contains thj@h
    - Process String contains cks=u
    - Process String contains 6I*h<8

- the 1st injection

- heuristic strings
PoisonIvy: Result

- 10 specimens used (1 stand-alone specimen)
  - 2.3.2 (distributed version)
  - 3 specimens used for targeted attacks
  - 6 samples collected from Malware.lu

code injection 1st

**AND:**
- **ProcessItem/name is 'Explorer.EXE'**
  - **OR:**
    - **ProcessItem/SectionList /MemorySection/Protection contains 'EXECUTE_READWRITE PrivateMemory MemCommit SecNoChange CopyOnWrite'**
    - **ProcessItem/SectionList /MemorySection/Protection contains 'EXECUTE_READWRITE PrivateMemory MemCommit MultipleSecured'**
  - **AND:**
    - **ProcessItem/StringList /string contains 'CONNECT %s:%i HTTP/1.0'**
    - **ProcessItem/StringList /string contains '?503'**

code injection 2nd

**AND:**
- **ProcessItem/arguments contains '-nohome'**
- **ProcessItem/name is 'IEXPLORE.EXE'**
  - **OR:**
    - **ProcessItem/StringList /string contains 'SOFTWARE\Microsoft \Windows \CurrentVersion\Run'**
    - **ProcessItem/StringList /string contains 'Software\Microsoft \Active Setup\Installed Components'**
  - **AND:**
    - **ProcessItem/StringList/string contains '?503'**
    - **ProcessItem/StringList/string contains 'thj@h'**
    - **ProcessItem/StringList/string contains 'cks=u'**
    - **ProcessItem/StringList/string contains '6l*h<8'**

stand alone variant
ZeroAccess

- Shift of ZeroAccess
  - Communication
    - from server-type botnet to P2P botnet
  - Implementation
    - from kernel-mode to user-mode
- Instructed to carry out click fraud and Bitcoin mining, but not limited
ZeroAccess User-mode: Installation

- **Dropper behavior**
  - extract main DLL from inline CAB and install it
  - inject code twice to explorer.exe
    - old user-mode version patches services.exe
  - bypass UAC misusing DLL load order in Windows
    - execute itself again as DLL with privilege

- **Many indicators, but not useful**
  - The installed DLL is different binary
    - After reboot, the dropper IOC vanishes
ZeroAccess User-mode: Protocol-related Strings

- P2P communication
  - UDP
    - send/receive bot commands
      - All commands are encoded by rotation XOR
  - TCP
    - upload/download plugin files
    - The callback functions differentiate socket status by checking dword tags
ZeroAccess User-mode: Imported Functions

- Low level APIs
  - Zw* (e.g., ZwQueryEaFile)
  - Ldr* (e.g., LdrGetProcedureAddress)
  - Rtl* (e.g., RtlImageNtHeader)

- Crypt APIs
  - e.g., CryptVerifySignatureW
ZeroAccess kernel-mode: Indicators

- Imported functions in kernel driver
  - e.g., KeInsertQueueApc
- Hidden volume accessed from user process
  - “ACPI#PNP0303#2&da1a3ff&0”
- The name seems to be unchanged
  - Later user-mode ZeroAccess still checks it in installation
ZeroAccess: IOC

- hidden volume name
- user-mode: XOR initial key
- user-mode: socket status tags
- user-mode: UDP bot commands
- user-mode: imported functions
ZeroAccess: IOC (Cont.)

kernel-mode: imported functions in driver
ZeroAccess: Result

- 4 specimens used
  - 2 samples of latest user-mode version
  - old user-mode version (services.exe patched)
  - kernel-mode version

latest/old user-mode

cornel-mode
DISCUSSIONS
Ease of IOC Definition

- ZeuS > SpyEye > ZeroAccess > PoisonIvy
  - The more functions malware has, the more indicators can be defined
    - Detection of an entire image injection is a significant indicator (ZeuS/SpyEye)
    - Calling APIs based on hash values reduces indicators (SpyEye/PoisonIvy)

- Variants identification
  - Small change can be identified (ZeuS/SpyEye)
  - Drastic change needs to be defined separately (ZeroAccess)
Limitations

- Not useful for detecting droppers/downloaders (e.g., Pony)
- OpenIOC tools are great, but..
  - cannot define binary patterns like YARA
    - e.g., PIC in PoisonIvy
  - cannot define “AND” combination of each item
    - e.g., ProcessItem and DriverItem in ZeroAccess
- cannot define regular expression
- cannot automate examinations \[8\]
- closed-source 😞
WRAP-UP
Wrap-up

- Volatile IOCs are effective for fast malware triage
  - Function-related indicators in memory can identify most variants
- Volatile IOC definitions require knowledge about malware
  - But everyone can use defined IOCs thanks to OpenIOC
- There are some limitations in OpenIOC tools
  - I expect Mandiant to improve them or disclose the sources
- Future work
  - Automation for scheduled tasks
    - Open-source (Volatility + YARA + pyioc[^9] + ?)
    - Other specifications (CybOX[^10] or IODEF[^11])
References

[1] The OpenIOC Framework (http://www.openioc.org/)

[2] IOC Editor (http://www.mandiant.com/resources/download/ioc-editor/)


[10] Cyber Observable eXpression (http://cybox.mitre.org/)