Fast and Generic Malware Triage Using openioc_scan Volatility Plugin

TAKAHIRO HARUYAMA (@CCI_FORENSICS)
INTERNET INITIATIVE JAPAN INC.

Digital Forensics Research Conference Europe 2015

Who am I?

- Forensic Investigator & Malware Analyst at Internet Initiative Japan Inc.
  - For details, please check our technical reports (IIR: Internet Infrastructure Review)
    - http://www.iij.ad.jp/en/company/development/iir/index.html
- Presentations and Hands-on classes
  - Black Hat Briefings USA/Europe/Asia
  - SANS Digital Forensics and Incident Response Summit
  - The Computer Enterprise and Investigations Conference
  - FIRST Technical Colloquium
  - etc...
- Blog
  - http://takahiroharuyama.github.io/
    - plugins/scripts for Volatility Framework, IDA Pro, Immunity Debugger and EnCase
- EnCase Certified Examiner since 2009

Overview

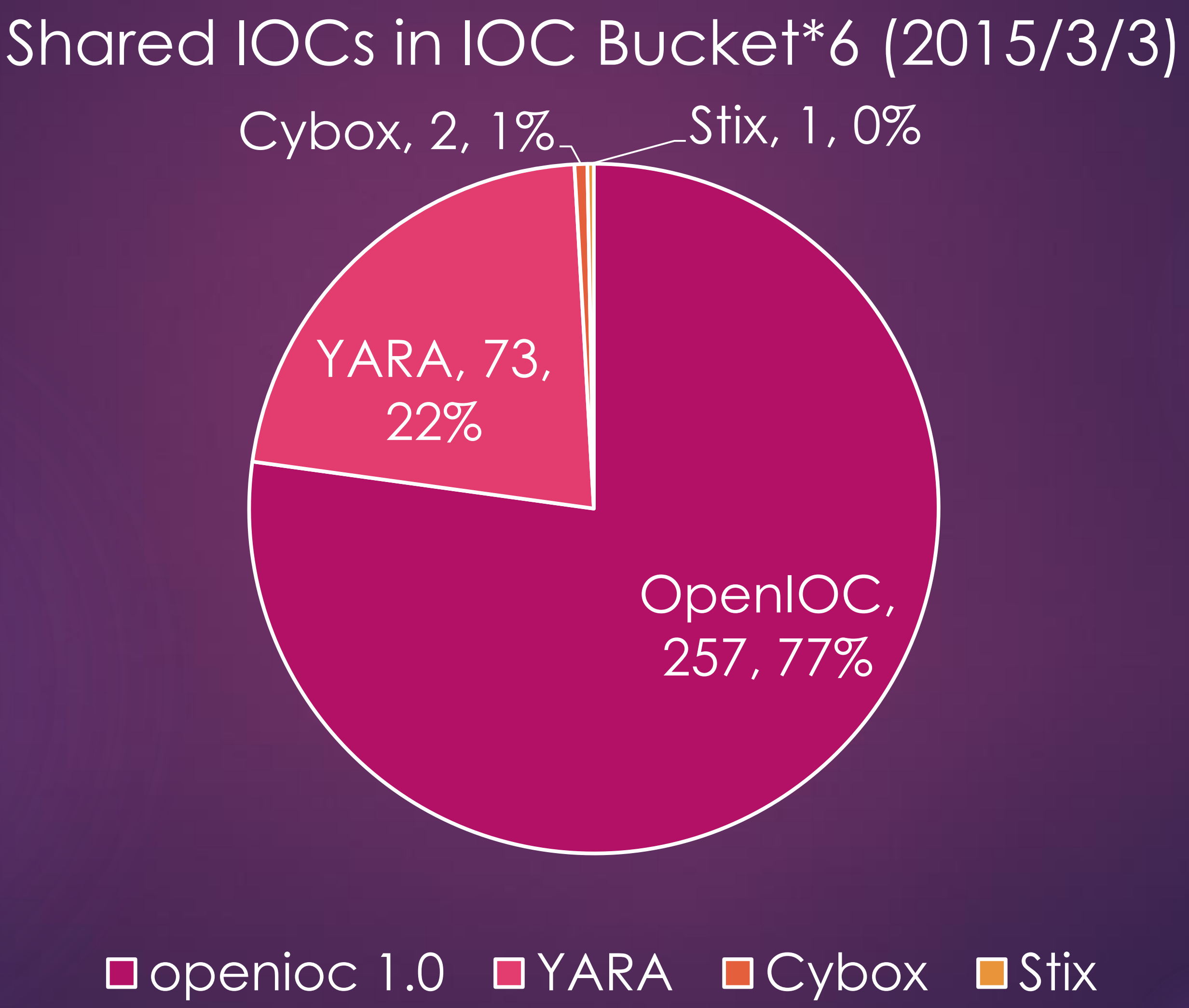
- Motivation
- “openioc_scan” Volatility Framework Plugin
- Generic IOCs

Motivation

IOC (Indicator Of Compromise)

- A piece of information that can be used to search for or identify potentially compromised systems*1
  - e.g., network-based IOC (IP/URL), host-based IOC (file hash)
  - Useful to detect known threats
- Some implementations and standards
  - YARA*2
  - OpenIOC*3
  - Cybox*4
  - Stix*5
  - etc...

Why OpenIOC?


Shared IOCs in IOC Bucket*6 (2015/3/3)
Cybox, 2, 1%
Stix, 1, 0%
YARA, 73, 22%
OpenIOC, 257, 77%
openioc 1.0
YARA
Cybox
Stix

Existing OpenIOC tools

- Free tools provided by Mandiant
  - IOC Finder*7
    - scan live systems
  - Redline*8
    - scan acquired memory images
    - safer and faster than live scan
      - I proposed “Volatile IOCs” for Redline at SANS DFIR Summit*9
- Problem
  - closed-source ☹

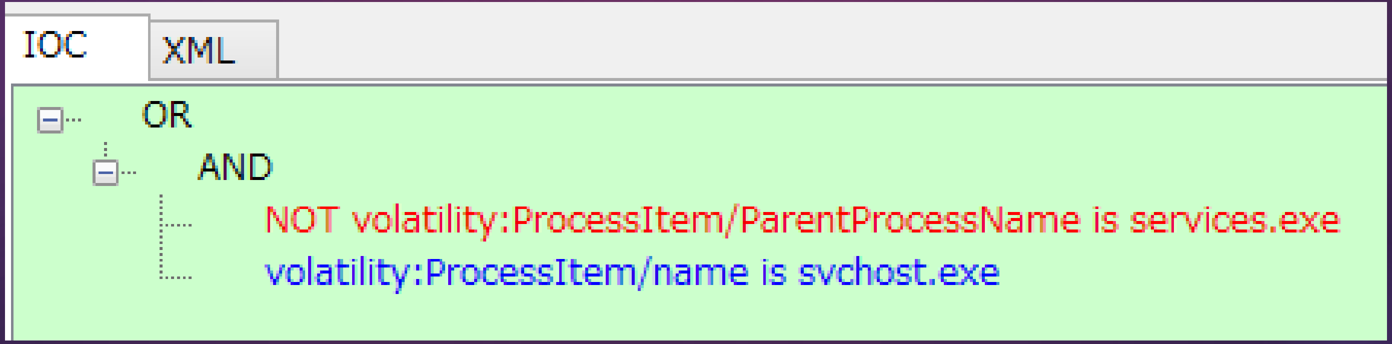
“openioc_scan” Volatility Framework Plugin

“openioc_scan” Volatility Framework Plugin

- Volatility Framework*10
  - open-source memory forensic tool
  - list unallocated kernel objects (e.g., dead process, unloaded kernel module)
- openioc_scan plugin
  - supports only Windows (Vista or later)
  - 3 python packages required
    - lxml*11
    - ioc_writer*12
    - colorma*13

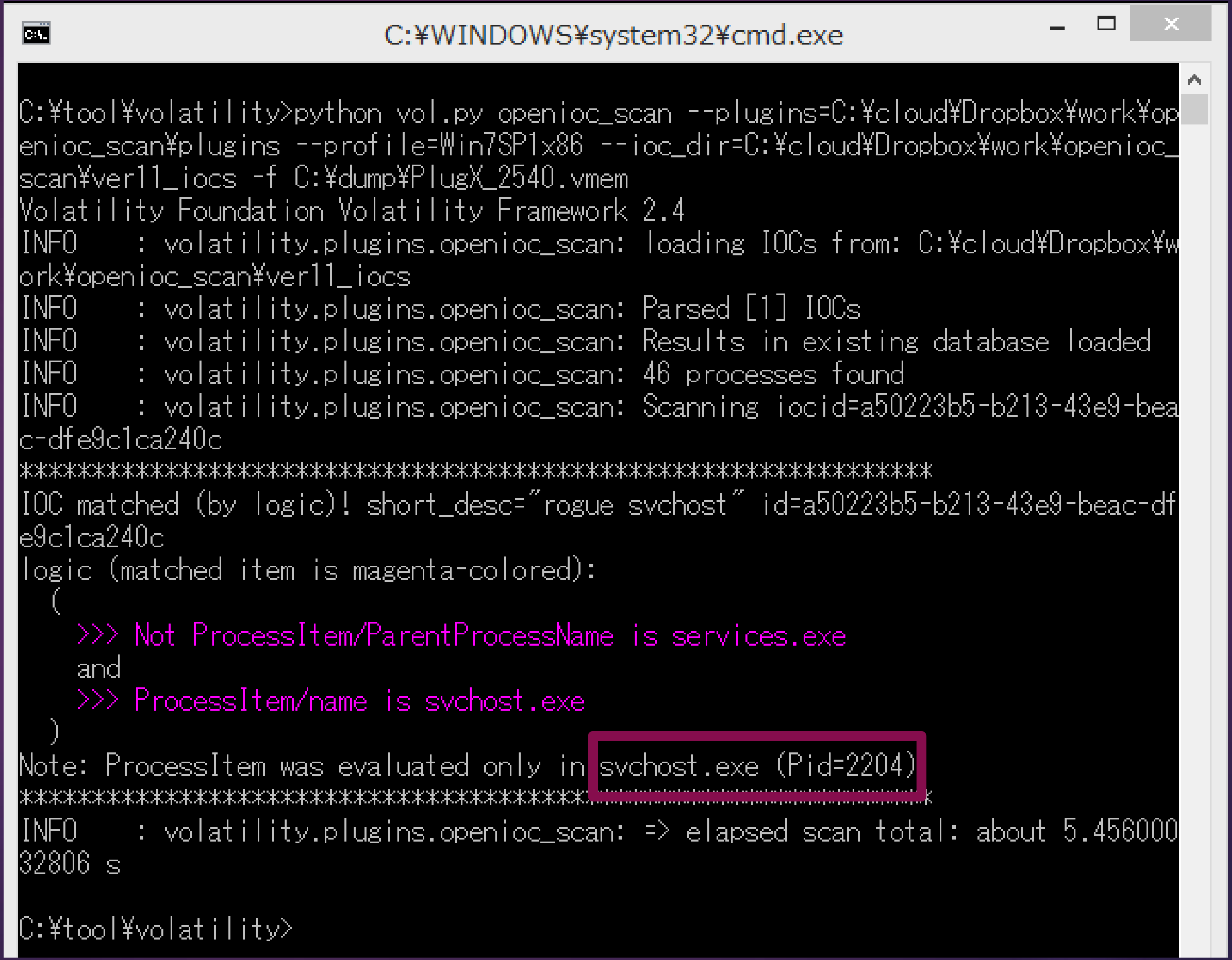
Generating IOCs for openioc_scan

- openioc_scan accepts OpenIOC 1.1 format, not 1.0
  - case sensitiveness
  - regular expression (“matches” condition)
  - “parameters” (explain later)
- PyIOCe*14 made by Sean Gillespie
  - support editing OpenIOC 1.1 format files
  - should import the latest “terms” and “parameters” for openioc_scan*15



```
OR
  AND
    NOT volatility:ProcessItem/ParentProcessName is services.exe
    volatility:ProcessItem/name is svchost.exe
```

Execution



```
C:¥tool¥volatility>python vol.py openioc_scan --plugins=C:¥cloud¥Dropbox¥work¥op
enioc_scan¥plugins --profile=Win7SP1x86 --ioc_dir=C:¥cloud¥Dropbox¥work¥openioc_
scan¥ver11_iocs -f C:¥dump¥PlugX_2540.vmem
Volatility Foundation Volatility Framework 2.4
INFO    : volatility.plugins.openioc_scan: loading IOCs from: C:¥cloud¥Dropbox¥w
ork¥openioc_scan¥ver11_iocs
INFO    : volatility.plugins.openioc_scan: Parsed [1] IOCs
INFO    : volatility.plugins.openioc_scan: Results in existing database loaded
INFO    : volatility.plugins.openioc_scan: 46 processes found
INFO    : volatility.plugins.openioc_scan: Scanning iocid=a50223b5-b213-43e9-bea
c-dfe9c1ca240c
****************************************************************
IOC matched (by logic)! short_desc="rogue svchost" id=a50223b5-b213-43e9-beac-df
e9c1ca240c
logic (matched item is magenta-colored):
  (
    >>> Not ProcessItem/ParentProcessName is services.exe
    and
    >>> ProcessItem/name is svchost.exe
  )
Note: ProcessItem was evaluated only in svchost.exe (Pid=2204)
****************************************************************
INFO    : volatility.plugins.openioc_scan: => elapsed scan total: about 5.456000
32806 s

C:¥tool¥volatility>
```

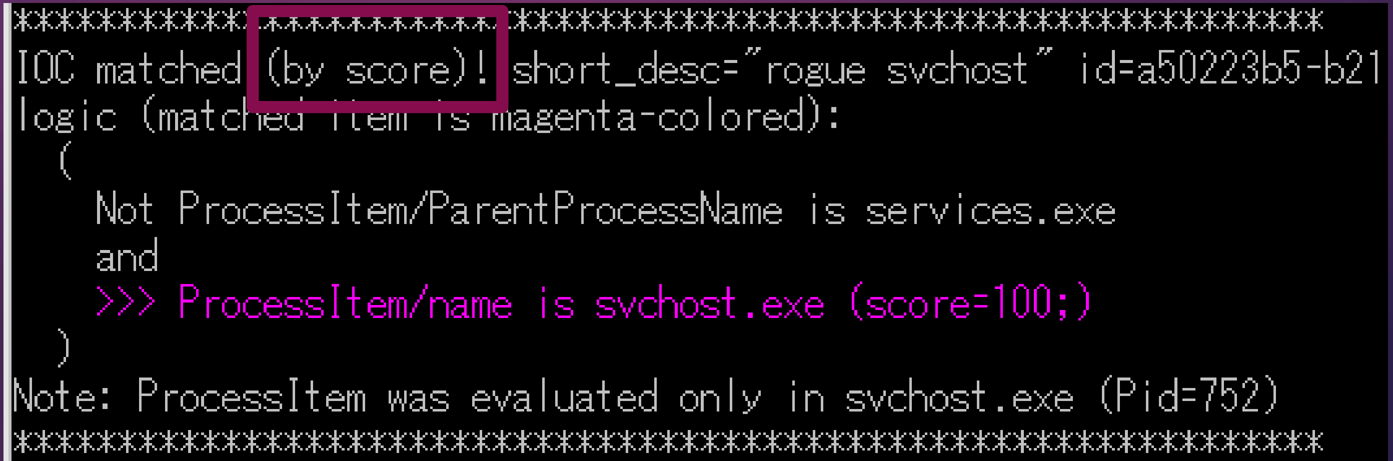
Supported 36 IOC Terms

- ProcessItem and DriverItem are evaluated per one process/driver
- I recommend KISS (Keeping IOCs Simple and Short)

Term Category	Term Examples
ProcessItem	name, command line, parent name, DLL path, DKOM detection, code injection detection, imported/dynamic generated API, string, handle name, network connection, IAT/EAT/inline hooked API, enabled privilege name
RegistryItem	metadata of executables cached by OS (ShimCache)
ServiceItem	service name/description/command line
DriverItem	name, imported/dynamic generated API, string, hooked IRP function table, callback function type, timer function detection
HookItem	hooked SSDT entry
FileItem	filename/size/path based on carved MFT entry

Parameters

- metadata for each IOC term supported in OpenIOC 1.1
- openioc_scan supports 3 parameters*16
  - score
    - additionally evaluate IOCs based on integer values (>=100)
  - detail
    - display not only matched substring but also total one
  - note
    - comment about the term



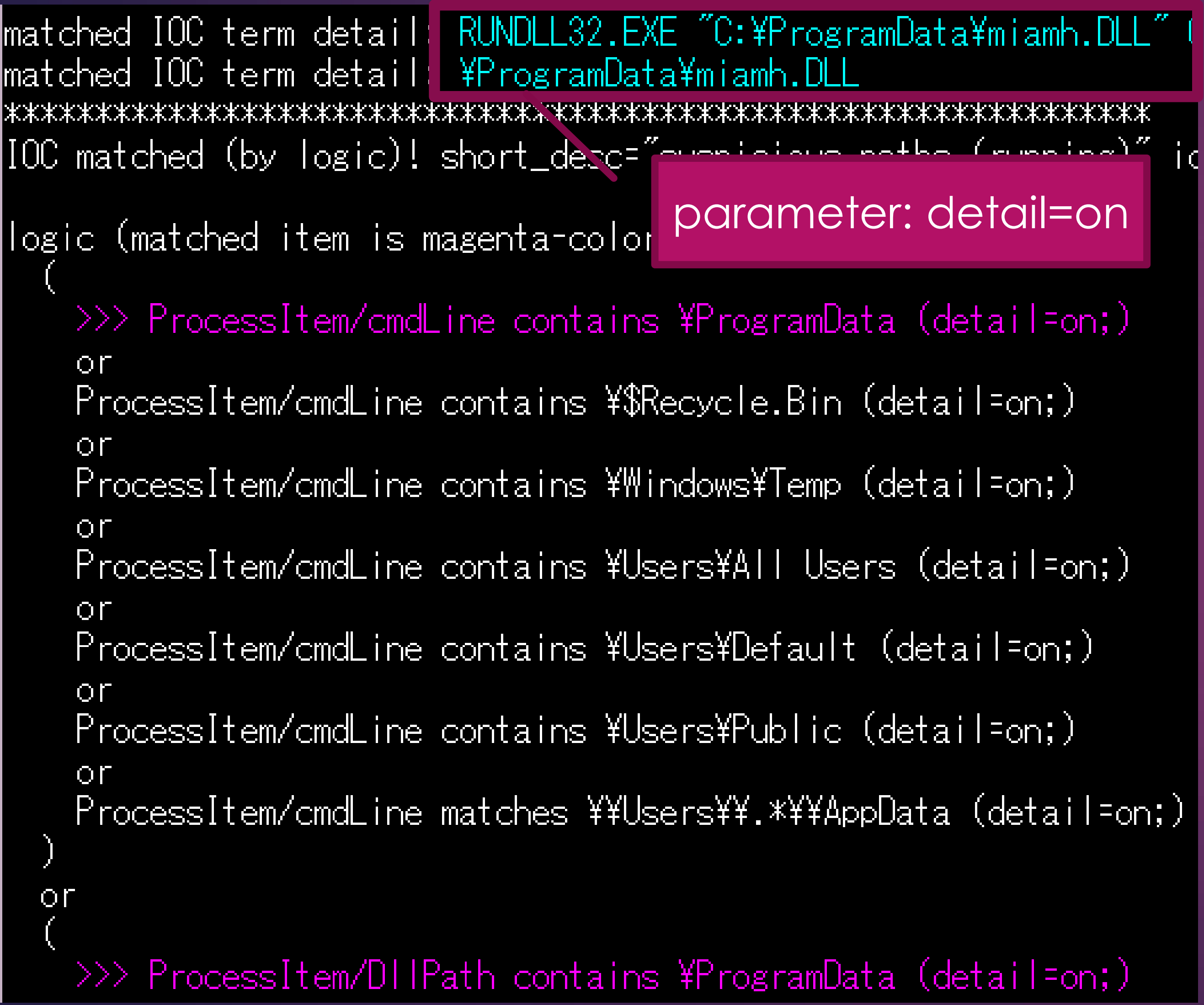
```
*****************************************************************************
IOC matched (by score)! short_desc="rogue svchost" id=a50223b5-b21
logic (matched item is magenta-colored):
  (
    Not ProcessItem/ParentProcessName is services.exe
    and
    >>> ProcessItem/name is svchost.exe (score=100;)
  )
Note: ProcessItem was evaluated only in svchost.exe (Pid=752)
*****************************************************************************
```

Generic IOCs

Considering Generic IOCs

- Currently, IOCs are applied to “known” threats
  - file hash and URL are mostly one-time and effective for only specific incidents
- openioc_scan can detect unknown ones based on generic traits
  - unusual executable paths
  - web injection
  - position independent code (PIC)
  - code injection
  - bypassing UAC dialog
  - hiding data in NTFS $EA
  - lateral movement in targeted attack

Unusual Paths (“Iron Man” Method*17)



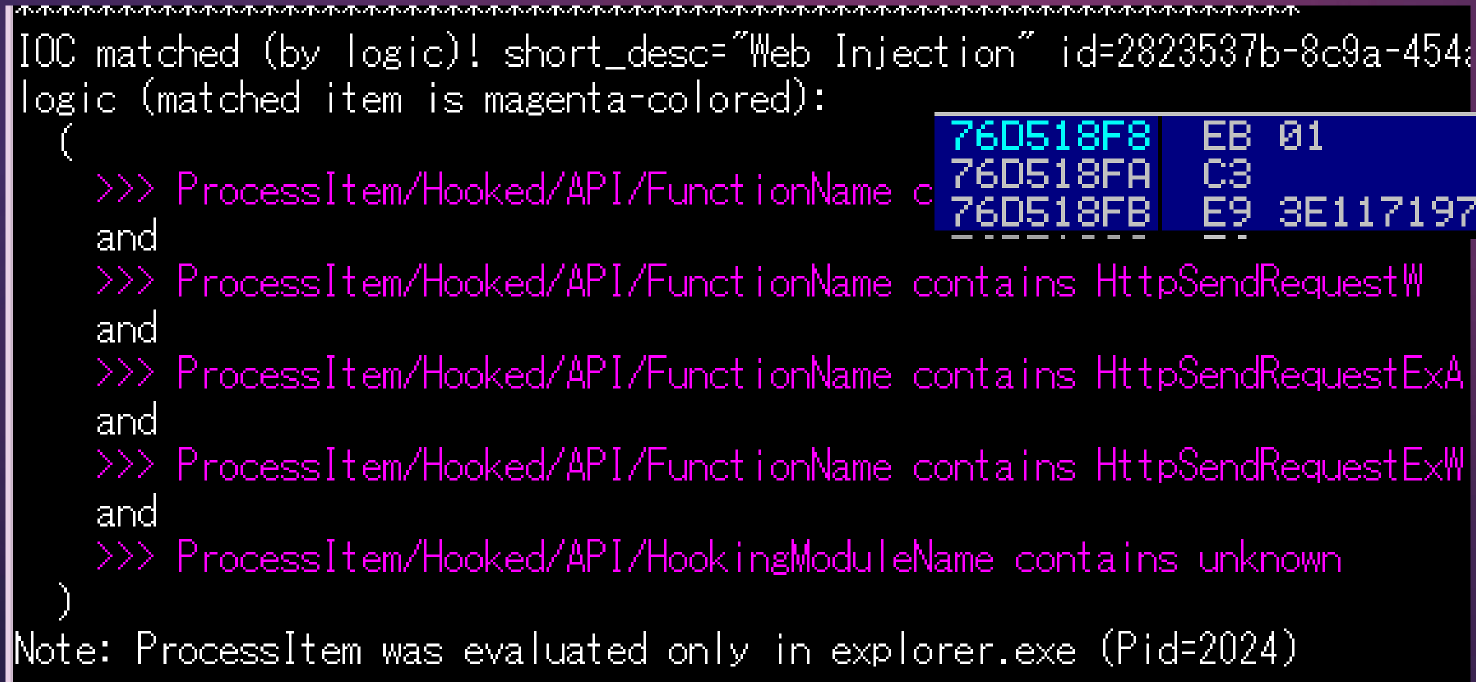
```
matched IOC term detail: RUNDLL32.EXE "C:¥ProgramData¥miamh.DLL" (
matched IOC term detail: ¥ProgramData¥miamh.DLL
***********************************************************************
IOC matched (by logic)! short_desc="suspicious paths (running)" ic

logic (matched item is magenta-color
  (
    >>> ProcessItem/cmdLine contains ¥ProgramData (detail=on;)
    or
    ProcessItem/cmdLine contains ¥$Recycle.Bin (detail=on;)
    or
    ProcessItem/cmdLine contains ¥Windows¥Temp (detail=on;)
    or
    ProcessItem/cmdLine contains ¥Users¥All Users (detail=on;)
    or
    ProcessItem/cmdLine contains ¥Users¥Default (detail=on;)
    or
    ProcessItem/cmdLine contains ¥Users¥Public (detail=on;)
    or
    ProcessItem/cmdLine matches ¥¥Users¥¥.*¥¥AppData (detail=on;)
  )
  or
  (
    >>> ProcessItem/DllPath contains ¥ProgramData (detail=on;)
```

parameter: detail=on

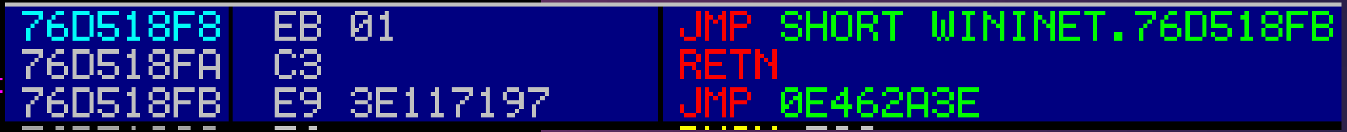
- generated two kinds of IOCs
  - exec paths in running processes
  - exec paths in ShimCache
- The former IOC caused less false positives than the latter one

Web Injection

- The indicators
  - HttpSendRequest APIs are hooked
  - The module name hooking APIs is unknown because of code injection
- detect EAT/IAT/inline hooks based on *apihooks* implementation
- Limitation
  - The inline hook detection checks only first 3 instructions and cheated by fake RET



```
IOC matched (by logic)! short_desc="Web Injection" id=2823537b-8c9a-454
logic (matched item is magenta-colored):
  (
    >>> ProcessItem/Hooked/API/FunctionName c
    and
    >>> ProcessItem/Hooked/API/FunctionName contains HttpSendRequestW
    and
    >>> ProcessItem/Hooked/API/FunctionName contains HttpSendRequestExA
    and
    >>> ProcessItem/Hooked/API/FunctionName contains HttpSendRequestExW
    and
    >>> ProcessItem/Hooked/API/HookingModuleName contains unknown
  )
Note: ProcessItem was evaluated only in explorer.exe (Pid=2024)
```

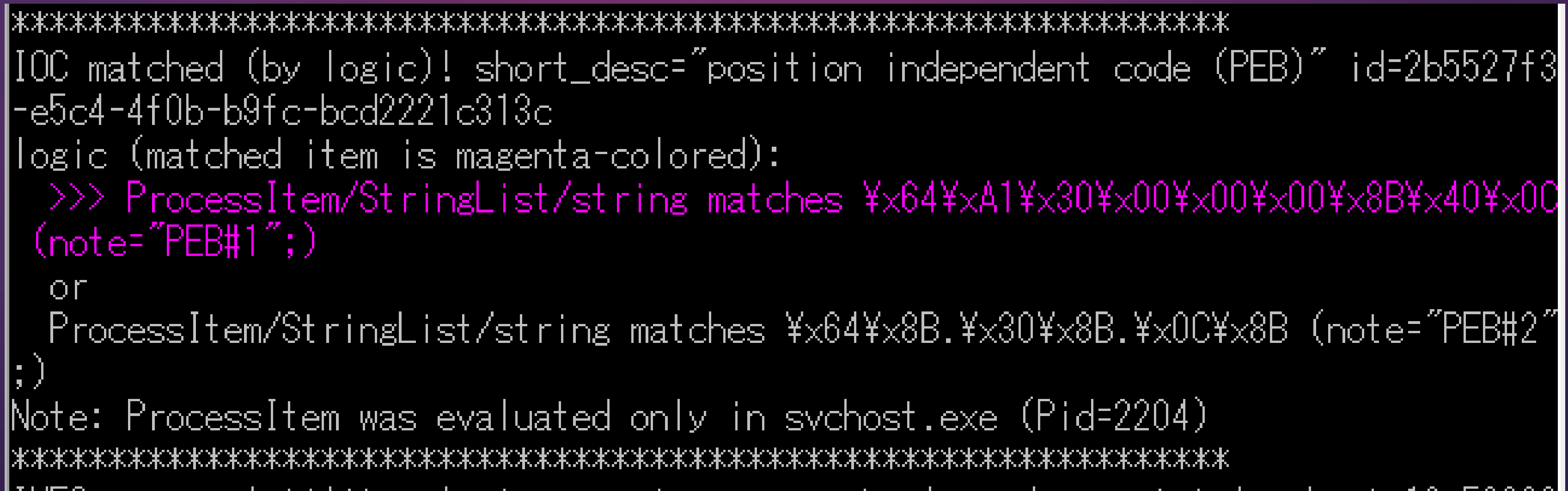


```
76D518F8   EB 01          JMP SHORT WININET.76D518FB
76D518FA   C3             RETN
76D518FB   E9 3E117197    JMP 0E462A3E
```

fake RET by SpyEye

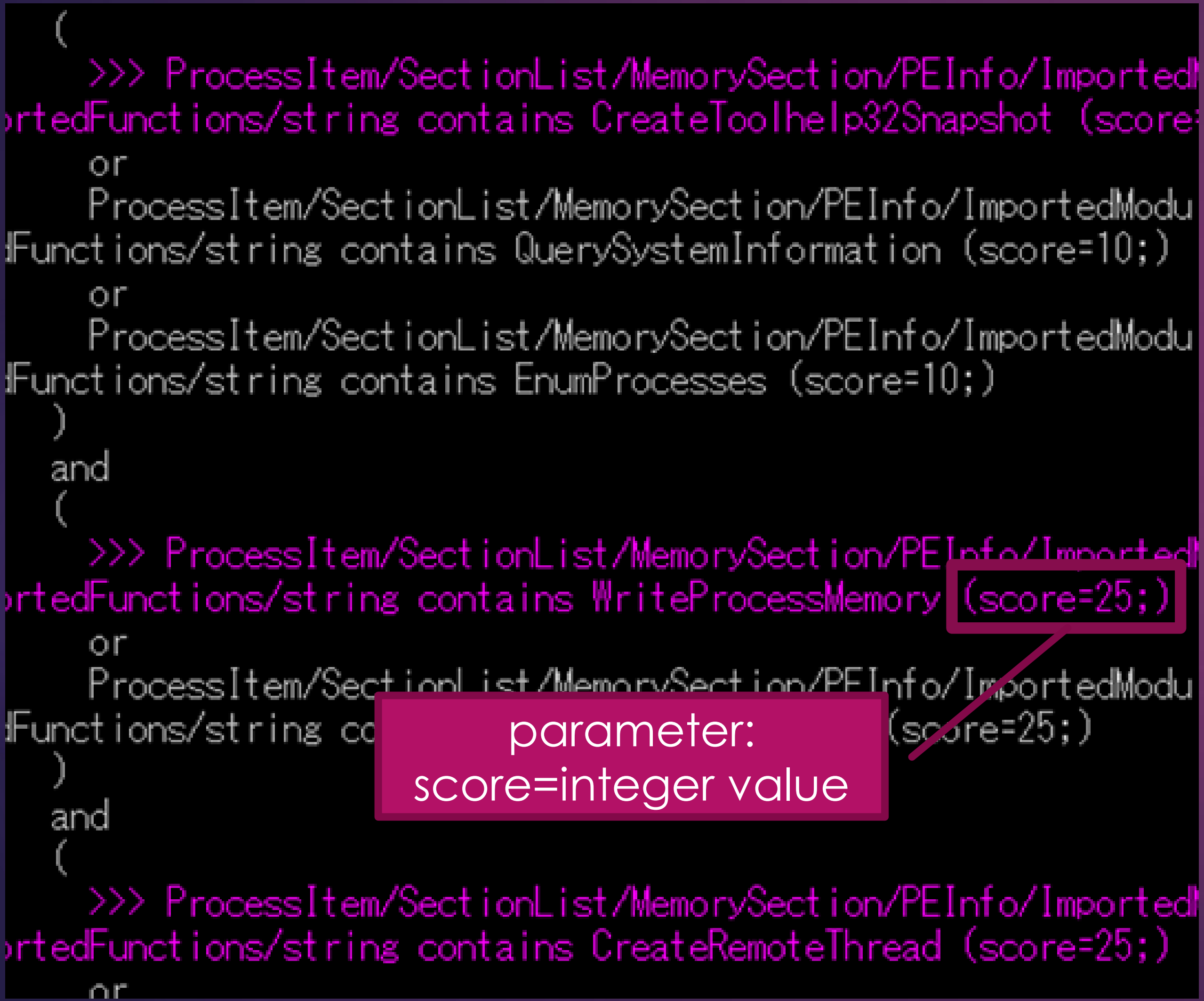
Position Independent Code (PIC)

- considered 3 kinds of binary sequences to detect PIC
  - access to PEB (e.g., mov eax, fs:dword_30; mov eax, [eax+0Ch])
  - "GetPC" code (e.g., call $+5; pop)
    - False positives found
  - API Hash (e.g., rol13AddHash32 of CreateFileA = 0xCACA3B9B)
    - Scanning all API hash patters is wasteful
- IOC of PEB access is better than others
- Limitation is to detect only x86 codes



```
******************************************************************
IOC matched (by logic)! short_desc="position independent code (PEB)" id=2b5527f3
-e5c4-4f0b-b9fc-bcd2221c313c
logic (matched item is magenta-colored):
  >>> ProcessItem/StringList/string matches ¥x64¥xA1¥x30¥x00¥x00¥x00¥x8B¥x40¥x0C
 (note="PEB#1";)
  or
  ProcessItem/StringList/string matches ¥x64¥x8B.¥x30¥x8B.¥x0C¥x8B (note="PEB#2"
;)
Note: ProcessItem was evaluated only in svchost.exe (Pid=2204)
******************************************************************
```

Code Injection

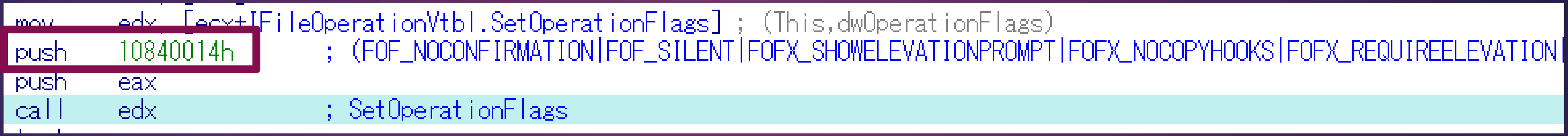


```
  (
    >>> ProcessItem/SectionList/MemorySection/PEInfo/Imported
ortedFunctions/string contains CreateToolhelp32Snapshot (score
    or
    ProcessItem/SectionList/MemorySection/PEInfo/ImportedModu
dFunctions/string contains QuerySystemInformation (score=10;)
    or
    ProcessItem/SectionList/MemorySection/PEInfo/ImportedModu
dFunctions/string contains EnumProcesses (score=10;)
  )
  and
  (
    >>> ProcessItem/SectionList/MemorySection/PEInfo/Imported
ortedFunctions/string contains WriteProcessMemory (score=25;)
    or
    ProcessItem/SectionList/MemorySection/PEInfo/ImportedModu
dFunctions/string co                                 (score=25;)
  )
  and
  (
    >>> ProcessItem/SectionList/MemorySection/PEInfo/Imported
ortedFunctions/string contains CreateRemoteThread (score=25;)
    or
```

- 3 IOCs combined with *malfind* condition
  1. commonly-used APIs
     - extended *impscan* to check dynamically-generated API tables and injected code sections
     - not work on wow64 process due to *impscan* limitation
  2. unknown hooking module name
  3. hex patterns of PIC
- The 3rd one is much faster and accurate
  - Term "*InjectedHexPattern*"

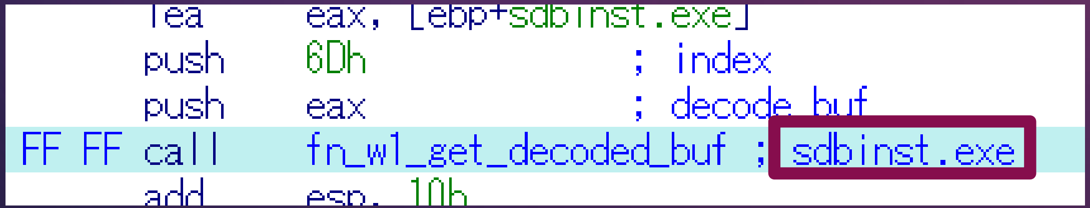
Bypassing UAC Dialog

- Two UAC bypassing techniques
  - DLL load-order hijacking*18
  - malicious SDB installation*21
- defined the characteristic code sequence / strings / APIs
- Limitation
  - There may be other methods bypassing UAC

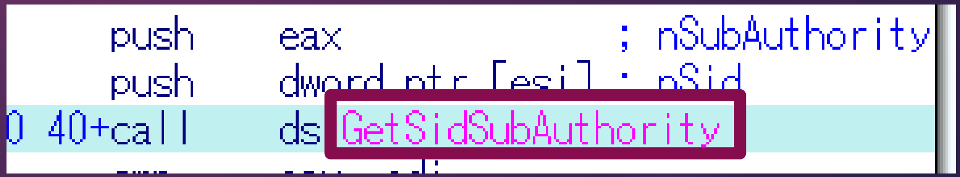


```
mov     edx, [ecx+IFileOperationVtbl.SetOperationFlags] ; (This,dwOperationFlags)
push    10840014h       ; (FOF_NOCONFIRMATION|FOF_SILENT|FOFX_SHOWELEVATIONPROMPT|FOFX_NOCOPYHOOKS|FOFX_REQUIREELEVATION|
push    eax
call    edx             ; SetOperationFlags
```

COM method called by PlugX



```
      lea     eax, [ebp+sdbinst.exe]
      push    6Dh                   ; index
      push    eax                   ; decode_buf
FF FF call    fn_w1_get_decoded_buf ; sdbinst.exe
      add     esp, 10h
```

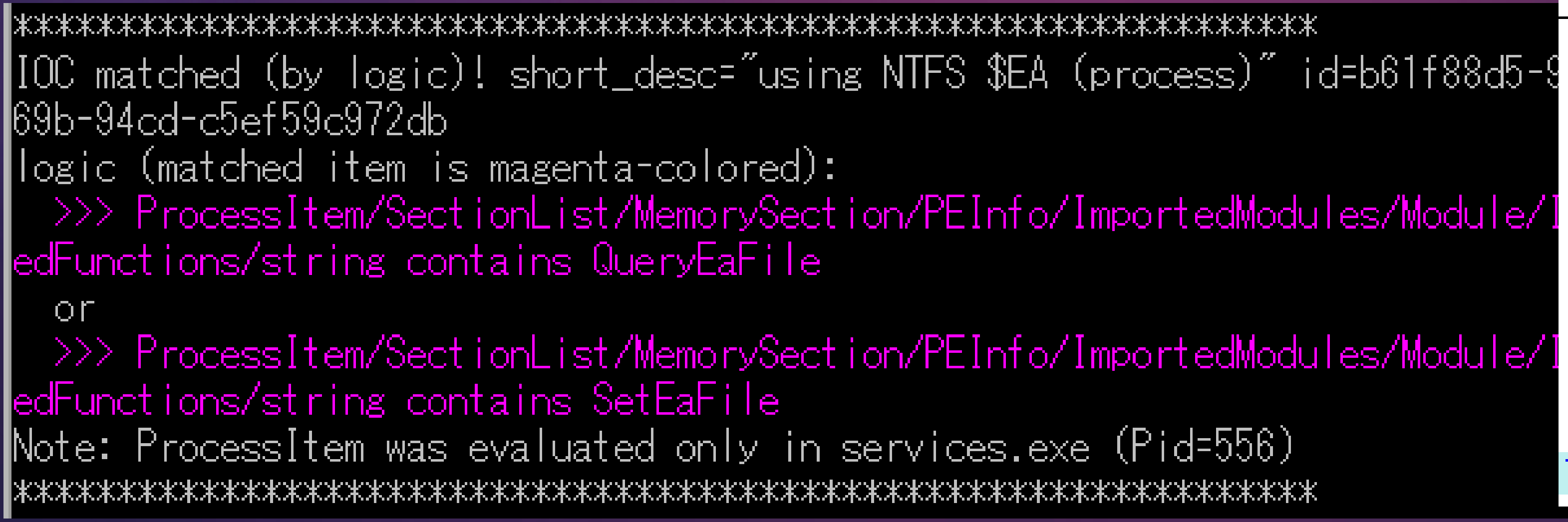


```
       push    eax               ; nSubAuthority
       push    dword ptr [esi]   ; pSid
0 40+call      ds:GetSidSubAuthority
```

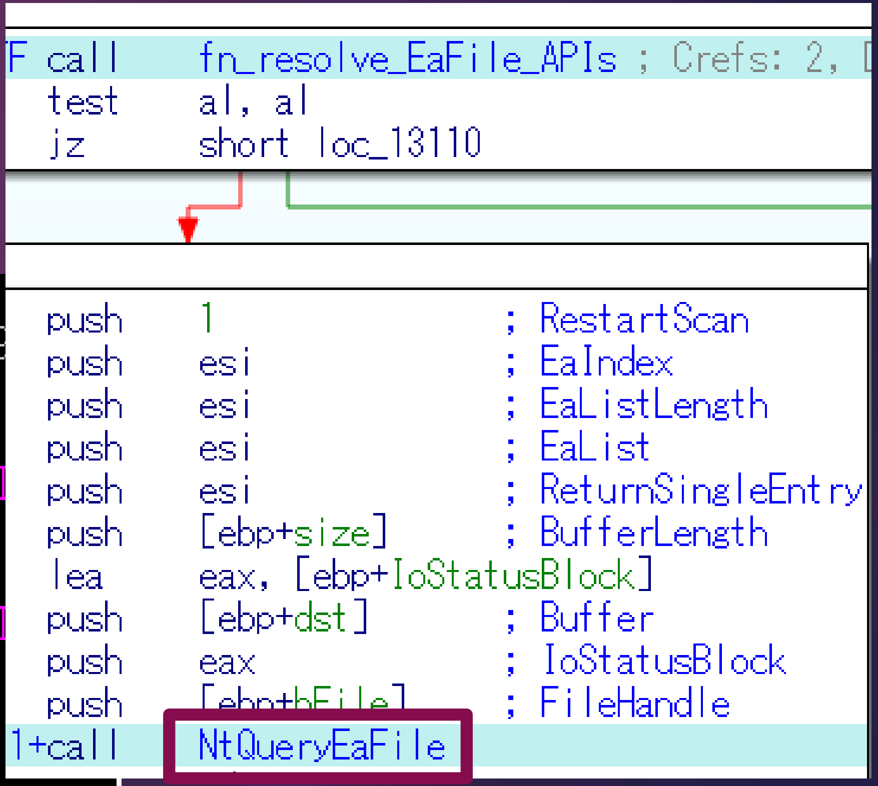
de-obfuscated string and API in Dridex

Hiding Data in NTFS $EA

- Some malware hides its code/data in NTFS extended attribute ($EA)
  - ZeroAccess (user-mode), Regin (kernel-mode)*22, etc…
- defined two IOCs (ProcessItem/DriverItem) based on APIs handling with $EA
- Limitation
  - not work on wow64 process
  - Some false positives found in kernel-mode



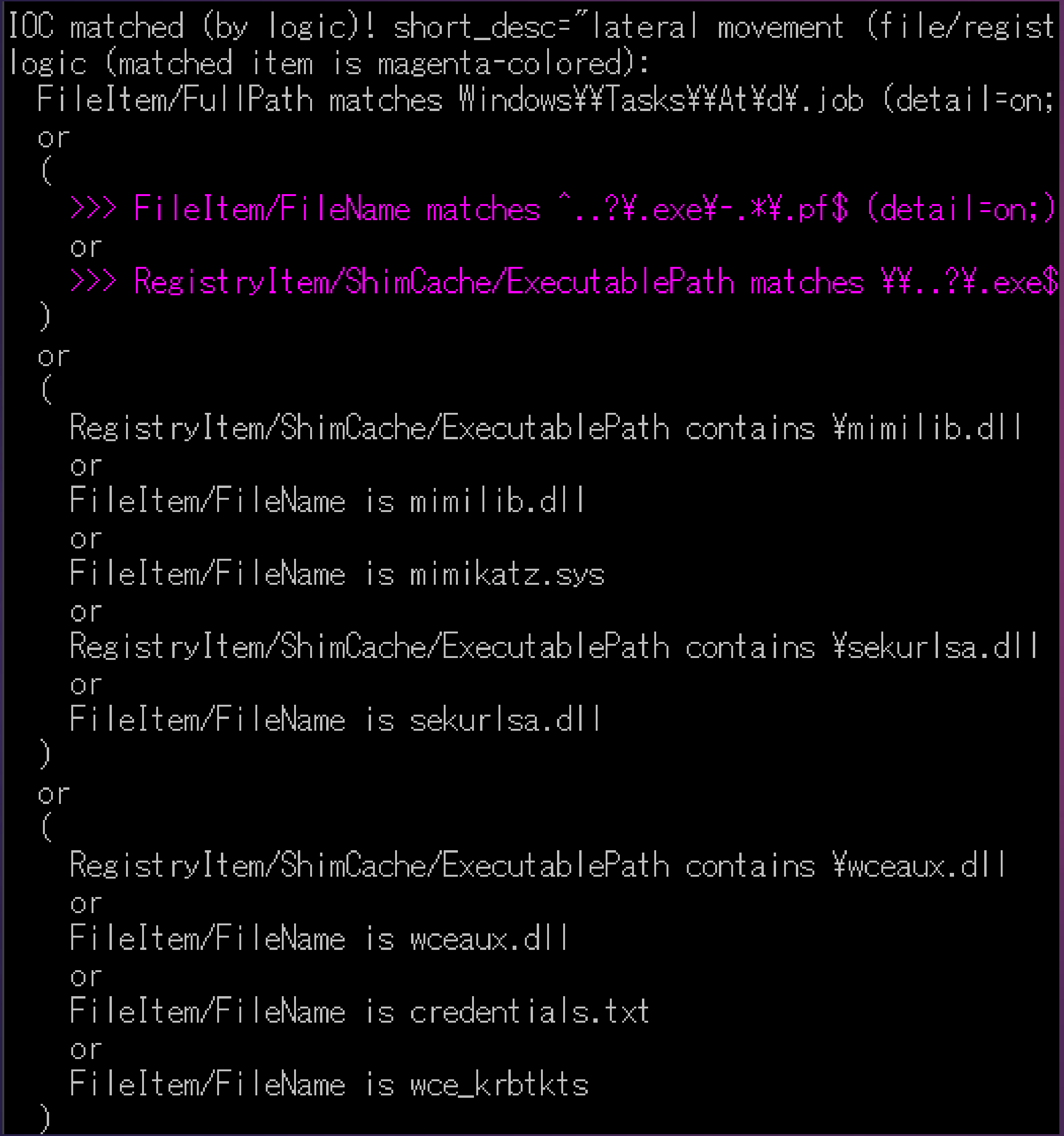
```
*****************************************************************
IOC matched (by logic)! short_desc="using NTFS $EA (process)" id=b61f88d5-9
69b-94cd-c5ef59c972db
logic (matched item is magenta-colored):
  >>> ProcessItem/SectionList/MemorySection/PEInfo/ImportedModules/Module/I
edFunctions/string contains QueryEaFile
  or
  >>> ProcessItem/SectionList/MemorySection/PEInfo/ImportedModules/Module/I
edFunctions/string contains SetEaFile
Note: ProcessItem was evaluated only in services.exe (Pid=556)
*****************************************************************
```



```
F call    fn_resolve_EaFile_APIs ; Crefs: 2, D
  test    al, al
  jz      short loc_13110

  push    1                   ; RestartScan
  push    esi                 ; EaIndex
  push    esi                 ; EaListLength
  push    esi                 ; EaList
  push    esi                 ; ReturnSingleEntry
  push    [ebp+size]          ; BufferLength
  lea     eax, [ebp+IoStatusBlock]
  push    [ebp+dst]           ; Buffer
  push    eax                 ; IoStatusBlock
  push    [ebp+hFile]         ; FileHandle
1+call    NtQueryEaFile
```

NtQueryEaFile resolved and called by Regin

Lateral Movement in Targeted Attack



```
IOC matched (by logic)! short_desc="lateral movement (file/regist
logic (matched item is magenta-colored):
  FileItem/FullPath matches Windows¥¥Tasks¥¥At¥d¥.job (detail=on;
  or
  (
    >>> FileItem/FileName matches ^..?¥.exe¥-.*¥.pf$ (detail=on;)
    or
    >>> RegistryItem/ShimCache/ExecutablePath matches ¥¥..?¥.exe$
  )
  or
  (
    RegistryItem/ShimCache/ExecutablePath contains ¥mimilib.dll
    or
    FileItem/FileName is mimilib.dll
    or
    FileItem/FileName is mimikatz.sys
    or
    RegistryItem/ShimCache/ExecutablePath contains ¥sekurlsa.dll
    or
    FileItem/FileName is sekurlsa.dll
  )
  or
  (
    RegistryItem/ShimCache/ExecutablePath contains ¥wceaux.dll
    or
    FileItem/FileName is wceaux.dll
    or
    FileItem/FileName is credentials.txt
    or
    FileItem/FileName is wce_krbtkts
  )
```

- IOCs finding artifacts generated by specific tools (*19, *20 and thanks to Junichi Hatta)
  - Windows CUI tools (e.g., at.exe)
  - SysInternals tools (e.g., psexec.exe)
  - PTH tools (e.g., wce.exe)
- two patterns
  - process-based
    - not useful
  - file/registry-based
    - heavily dependent on metadata
- difficult to define generic ones

Wrap-up

Wrap-up

- openioc_scan plugin for Volatility Framework
  - generic IOCs to detect unknown threats
    - Zero false positive is difficult, but useful for first triage
  - Some limitations due to the implementation of Volatility Framework
    - but we can improve them thanks to open-source ☺
- The tool and generic IOCs are available on my blog
  - http://takahiroharuyama.github.io/
- Share your own IOCs in the world!

Reference

- [1] Sharing Indicators of Compromise: An Overview of Standards and Formats
  - https://www.rsaconference.com/writable/presentations/file_upload/dsp-w25a.pdf
- [2] YARA - The pattern matching swiss knife for malware researchers
  - https://plusvic.github.io/yara/
- [3] The OpenIOC Framework
  - http://www.openioc.org/
- [4] CybOX - Cyber Observable Expression
  - https://cybox.mitre.org/
- [5] STIX - Structured Threat Information Expression
  - https://stix.mitre.org/
- [6] IOC Bucket
  - https://www.iocbucket.com/
- [7] IOC Finder
  - http://www.mandiant.com/resources/download/ioc-finder/
- [8] Redline
  - https://www.mandiant.com/resources/download/redline

Reference (Cont.)

- [9] Volatile IOCs for Fast Incident Response
  - https://digital-forensics.sans.org/summit-archives/DFIR_Summit/Volatile-IOCs-for-Fast-Incident-Response-Haruyama.pdf
- [10] volatilityfoundation/volatility
  - https://github.com/volatilityfoundation/volatility
- [11] lxml 3.2.1 : Python Package Index
  - https://pypi.python.org/pypi/lxml/3.2.1
- [12] mandiant/ioc_writer
  - https://github.com/mandiant/ioc_writer
- [13] colorama 0.3.3 : Python Package Index
  - https://pypi.python.org/pypi/colorama
- [14] yahoo/PyIOCe
  - https://github.com/yahoo/PyIOCe
- [15] Fast Malware Triage Using Openioc_scan Volatility Plugin
  - http://takahiroharuyama.github.io/blog/2014/08/15/fast-malware-triage-using-openioc-scan-volatility-plugin/
- [16] OpenIOC Parameters Used by Openioc_scan
  - http://takahiroharuyama.github.io/blog/2014/10/24/openioc-parameters-used-by-openioc-scan/

Reference (Cont.)

- [17] Finding Malware Like Iron Man Slide Decks
  - http://journeyintoir.blogspot.jp/2013/07/finding-malware-like-iron-man-slide.html
- [18] Bypassing Windows User Account Control (UAC) and ways of mitigation
  - http://www.greyhathacker.net/?p=796
- [19] Do not fumble the lateral movement
  - https://sysforensics.org/2014/01/lateral-movement.html
- [20] Pass-The-Hash: Gaining Root Access to Your Network
  - http://first.org/resources/papers/conference2014/first_2014_-_slaybaugh-_tim_-_pass_the_hash_20140623.pptx
- [21] A New UAC Bypass Method that Dridex Uses
  - http://blog.jpcert.or.jp/2015/02/a-new-uac-bypass-method-that-dridex-uses.html
- [22] THE REGIN PLATFORM - NATION-STATE OWNAGE OF GSM NETWORKS
  - https://securelist.com/files/2014/11/Kaspersky_Lab_whitepaper_Regin_platform_eng.pdf