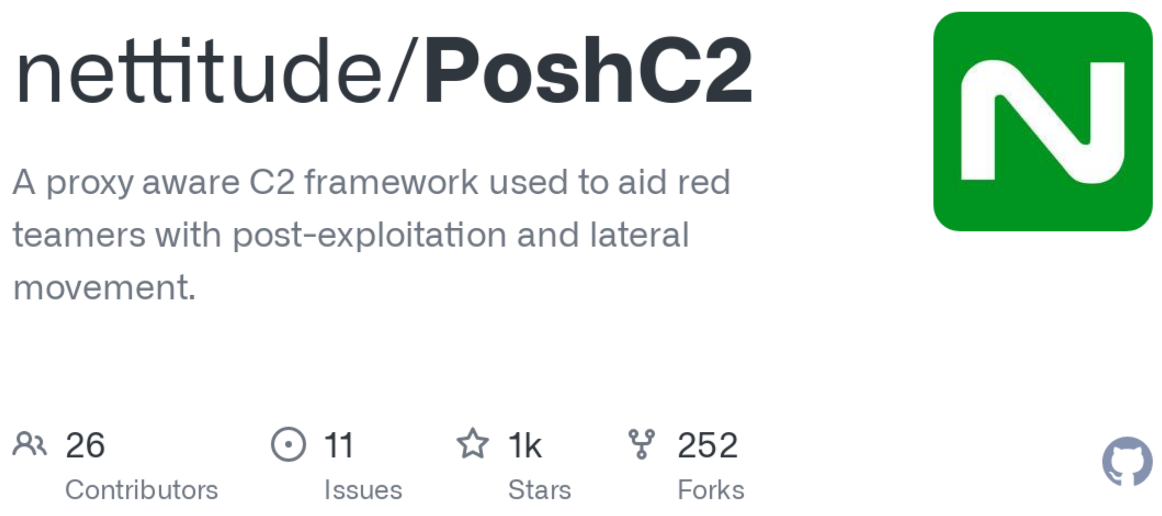
nettitude/PoshC2: A proxy aware C2 framework used to aid red teamers with post-exploitation and lateral movement.

github.com/nettitude/PoshC2_Python/

nettitude



nettitude/**PoshC2**

A proxy aware C2 framework used to aid red teamers with post-exploitation and lateral movement.

26 Contributors

11 Issues

1k Stars

252 Forks

PoshC2

Docker Image CI passing

PoshC2 is a proxy aware C2 framework used to aid penetration testers with red teaming, post-exploitation and lateral movement.

PoshC2 is primarily written in Python3 and follows a modular format to enable users to add their own modules and tools, allowing an extendible and flexible C2 framework. Out-of-the-box PoshC2 comes PowerShell/C# and Python2/Python3 implants with payloads written in PowerShell v2 and v4, C++ and C# source code, a variety of executables, DLLs and raw shellcode in addition to a Python2/Python3 payload. These enable C2 functionality on a wide range of devices and operating systems, including Windows, *nix and OSX.

Other notable features of PoshC2 include:

- Consistent and Cross-Platform support using Docker.
- Highly configurable payloads, including default beacon times, jitter, kill dates, user agents and more.
- A large number of payloads generated out-of-the-box which are frequently updated.
- Shellcode containing in-build AMSI bypass and ETW patching for a high success rate and stealth.
- Auto-generated Apache Rewrite rules for use in a C2 proxy, protecting your C2 infrastructure and maintaining good operational security.
- A modular and extensible format allowing users to create or edit C#, PowerShell or Python3 modules which can be run in-memory by the Implants.
- Notifications on receiving a successful Implant via Pushover or Slack.
- A comprehensive and maintained contextual help and an intelligent prompt with contextual auto-completion, history and suggestions.
- Fully encrypted communications, protecting the confidentiality and integrity of the C2 traffic even when communicating over HTTP.
- Client/Server format allowing multiple team members to utilise a single C2 server.
- Extensive logging. Every action and response is timestamped and stored in a database with all relevant information such as user, host, implant number etc. In addition to this the C2 server output is directly logged to a separate file.
- PowerShell-less implants that do not use System.Management.Automation.dll using C# or Python2/Python3.
- A free and open-source SOCKS Proxy using SharpSocks
- HTTP(S) and SMB named-pipe comms for implants combined with Implant Daisy-chaining for reaching networks that do not have access to the internet

Documentation

We maintain PoshC2 documentation over at https://poshc2.readthedocs.io/en/latest/

Find us on #Slack - poshc2.slack.com (to request an invite send an email to labs@nettitude.com)

Install

You can install PoshC2 directly or use the Docker images, instructions for both are below.

Direct install on Kali hosts

An install script is provided for installing PoshC2:

```
*** PoshC2 Install script ***
Usage:
./Install.sh -b <git branch> -p <Directory to clone PoshC2 to>

Defaults are master branch to /opt/PoshC2
```

Elevated privileges are required as the install script performs `apt` updates and installations.

```
curl -sSL
https://raw.githubusercontent.com/nettitude/PoshC2/master/Install.sh |
sudo bash
```

Alternatively the repository can be cloned down and the install script manually run.

```
sudo ./Install.sh
```

You can manually set the PoshC2 installation directory by passing it to the Install.sh script as the `-p` argument. The default is **/opt/PoshC2**:

```
curl -sSL
https://raw.githubusercontent.com/nettitude/PoshC2/master/Install.sh |
sudo bash -s -- -p /root/PoshC2
```

Cutting Edge Features

We want to keep the `master` branch stable to ensure that users are able to rely on it when required and for this reason changes can often be feature-complete but not yet present on `master` as they have not been tested completely and signed-off yet.

If you want to look at upcoming features in PoshC2 you can check out the `dev` branch, or any individual feature branches branched off of `dev` .

As features **are** tested before they are merged into `dev` this branch should still be fairly stable and operators can opt in to using this branch or a particular feature branch for their engagement. This does trade stablity for new features however so do it at your own discretion.

To use `dev` or a feature branch pass the branch name to the Install.sh script as the `-b` argument:

```
curl -sSL
https://raw.githubusercontent.com/nettitude/PoshC2/dev/Install.sh | sudo
bash -s -- -b dev
```

Note the URL includes the branch name also (here `dev` instead of `master`).

Installing for Docker

You can also run PoshC2 using Docker, this allows more stable and running and enables PoshC2 to easily run on other operating systems.

The Docker install does not clone PoshC2 as the PoshC2 images on Docker Hub are used, so only a minimal install of some dependencies and scripts are performed.

To start with, install Docker on the host and then add the PoshC2 projects directory to Docker as a shared directory if required for your OS. By default this is **/var/poshc2** on *nix and **/private/var/poshc2** on Mac.

Kali based hosts

Install script:

```
*** PoshC2 Install script for Docker ***
Usage:
./Install-for-Docker.sh -b <git branch>

Default is the master branch
```

Elevated privileges are required as the install script performs script installations.

```
curl -sSL
https://raw.githubusercontent.com/nettitude/PoshC2/master/Install-for-
Docker.sh | sudo bash
```

To use the `dev` or feature branches with Docker curl down the `Install-for-Docker.sh` on the appropriate branch and pass the branch name as an argument:

```
curl -sSL
https://raw.githubusercontent.com/nettitude/PoshC2/BRANCHNAME/Install-
for-Docker.sh | sudo bash -s -- -b BRANCHNAME
```

Windows

On Windows, import the PoshC2.psm1 PowerShell module.

```
Import-Module -DisableNameChecking
C:\PoshC2\resources\scripts\PoshC2.psm1
posh-project -PoshC2Dir "C:\PoshC2" -LocalPoshC2ProjectDir
"C:\PoshC2_Project" -Arg1 "-n" -Arg2 "newproject"
posh-config -PoshC2Dir "C:\PoshC2" -LocalPoshC2ProjectDir
"C:\PoshC2_Project"
posh-server -PoshC2Dir "C:\PoshC2" -LocalPoshC2ProjectDir
"C:\PoshC2_Project"
posh -PoshC2Dir "C:\PoshC2" -LocalPoshC2ProjectDir "C:\PoshC2_Project"
username
```

Running PoshC2

Create a new project:

```
posh-project -n <project-name>
```

Projects can be switched to or listed using this script:

```
[*] Usage: posh-project -n <new-project-name>
[*] Usage: posh-project -s <project-to-switch-to>
[*] Usage: posh-project -l (lists projects)
[*] Usage: posh-project -d <project-to-delete>
[*] Usage: posh-project -c (shows current project)
```

Edit the configuration for your project:

```
posh-config
```

Launch the PoshC2 server:

```
posh-server
```

Alternatively start it as a service:

```
posh-service
```

Separately, run the ImplantHandler for interacting with implants:

```
posh -u <username>
```

See https://poshc2.readthedocs.io/en/latest/ for full documentation on PoshC2.

Specifying a Docker tag

If you are using Docker you can specify the Docker image tag to run with the `-t` option to `posh-server` and `posh`.

E.g.

```
posh-server -t latest
```

Updating PoshC2 Installations

It is not recommended to update PoshC2 during an engagement. Incoming changes may be incompatible with an existing project and can result in erratic behaviour.

When using a git cloned version of PoshC2 you can update your PoshC2 installation using the following command:

```
*** PoshC2 Update Script ***
Usage:
posh-update -b <git branch>

Default is the master branch
```

Using older versions

You can use an older version of PoshC2 by referencing the appropriate tag. Note this only works if you have cloned down the repository. You can list the tags for the repository by issuing:

```
git tag --list
```

If you have a local clone of PoshC2 you can change the version that is in use while offline by just checking out the version you want to use:

```
git reset --hard <tag name>
```

For example:

```
git reset --hard v4.8
```

However note that this will overwrite any local changes to files, such as changes to the configuration files, and you may have to re-run the install script for that version or re-setup the environment appropriately.

License / Terms of Use

This software should only be used for **authorised** testing activity and not for malicious use.

By downloading this software you are accepting the terms of use and the licensing agreement.