byt3bl33d3r Update README.md 08b1c61 · last year 293 Commits

.github	Update FUNDING.yml	4 years ago
silenttrinity	Recompiled & obfuscated the C# exe/d...	3 years ago
tests	Fixed tests	4 years ago
.gitignore	Initial commit for v0.4.6	5 years ago
CONTRIBUTING.md	Basically a complete re-write, now uses...	5 years ago
LICENSE	Basically a complete re-write, now uses...	5 years ago
Makefile	Fixed error in shebang line when creati...	5 years ago
NOTES.md	Basically a complete re-write, now uses...	5 years ago
Pipfile	Merges #134, adds posh_stageless sta...	4 years ago
Pipfile.lock	Merges #134, adds posh_stageless sta...	4 years ago
README.md	Update README.md	last year
TODO.md	Basically #27 with some minor mods, c...	5 years ago
requirements.txt	Merges #134, adds posh_stageless sta...	4 years ago
st.py	Initial commit for v0.4.6	5 years ago

README GPL-3.0 license

SILENTTRINITY Tests failing

SILENTTRINITY

About

An asynchronous, collaborative post-exploitation agent powered by Python and .NET's DLR

c-sharp dotnet python3 ironpython post-exploitation security-tools red-teams dotnet-script boolang dotnet-dlr

Readme
GPL-3.0 license
Activity
2.2k stars
111 watching
403 forks
Report repository

Releases

No releases published

Sponsor this project

byt3bl33d3r Marcello

patreon.com/byt3bl33d3r

ko-fi.com/byt3bl33d3r

Learn more about GitHub Sponsors

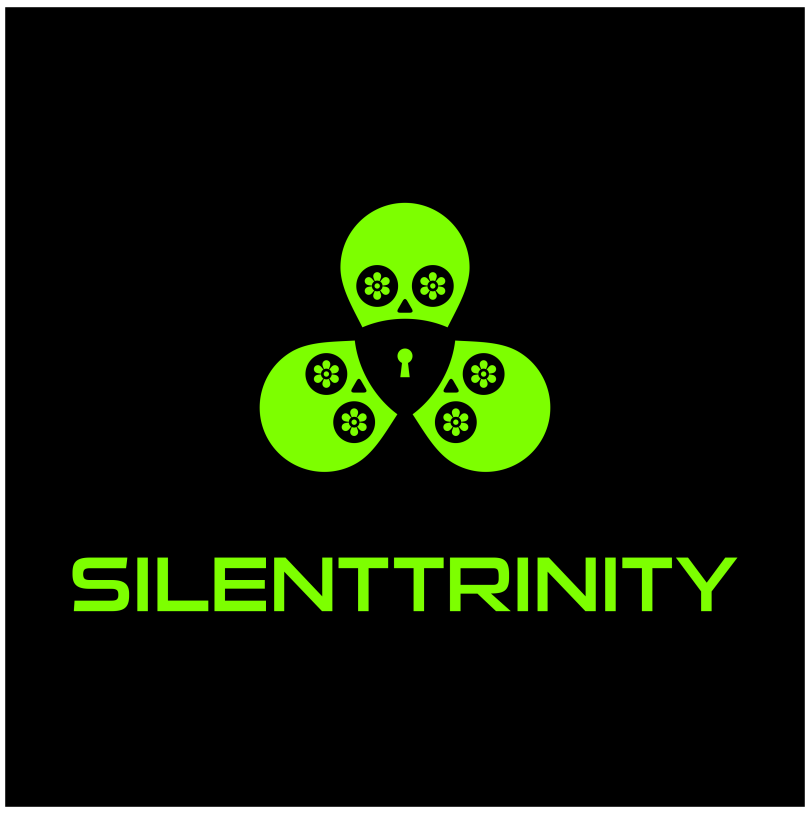
Packages

No packages published

Contributors 13

Languages

Boo 69.1% Python 28.2%
XSLT 2.3% Other 0.4%



SILENTTRINITY is modern, asynchronous, multiplayer & multiserver C2/post-exploitation framework powered by Python 3 and .NETs DLR. It's the culmination of an extensive amount of research into using embedded third-party .NET scripting languages to dynamically call .NET API's, a technique the author coined as BYOI (Bring Your Own Interpreter). The aim of this tool and the BYOI concept is to shift the paradigm back to PowerShell style like attacks (as it offers much more flexibility over traditional C# tradecraft) only without using PowerShell in anyway.

Some of the main features that distinguish SILENTTRINITY are:

- **Multi-User & Multi-Server** - Supports multi-user collaboration. Additionally, the client can connect to and control multiple Teamservers.
- **Client and Teamserver Built in Python 3.7** - Latest and greatest features of the Python language are used, heavy use of Asyncio provides ludicrous speeds.
- **Real-time Updates and Communication** - Use of Websockets allow for real-time communication and updates between the Client and Teamserver.
- **Focus on Usability with an Extremely Modern CLI** - Powered by prompt-toolkit.
- **Dynamic Evaluation/Compilation Using .NET Scripting Languages** - The SILENTTRINITY implant Naga, is somewhat unique as it uses embedded third-party .NET scripting languages (e.g. Boolang) to dynamically compile/evaluate tasks, this removes the need to compile tasks server side, allows for real-time editing of modules, provides greater flexibilty and stealth over traditional C# based payloads and makes everything much more light-weight.
- **ECDHE Encrypted C2 Communication** - SILENTTRINITY uses Ephemeral Elliptic Curve Diffie-Hellman Key Exchange to encrypt all C2 traffic between the Teamserver and its implant.
- **Fully Modular** - Listeners, Modules, Stagers and C2 Channels are fully modular allowing operators to easily build their own.
- **Extensive logging** - Every action is logged to a file.
- **Future proof** - HTTPS/HTTP listeners are built on Quart & Hypercorn which also support HTTP2 & Websockets.

Call for Contributions

I'm just one person developing this mostly in my spare time, I do need to have a life outside of computers (radical idea, I know).

This means that if anyone finds this tool useful and would like to see X functionality added, the best way to get it added is to submit a Pull Request.

Be the change you want to see in the world!

As of the time of writing the most useful thing you can contribute are post-ex modules: this would allow me to concentrate efforts on the framework itself, user experience,

QOL features etc...

To do this, you're going to have to learn the Boo programming language (the Boo wiki is amazing and has everything you'd need to get started), if you know Python you'll find yourself at home :).

Check out some of the existing modules, if you've written an Empire module before you'll see its very similar. Finally you can start porting over post-ex modules from other C2 frameworks such as Empire.

Documentation, Setup & Basic Usage

The documentation is a work in progress but some is already available in the Wiki.

See here for install instructions and here for basic usage.

I recommend making wild use the `help` command and the `-h` flag :)

Author

Marcello Salvati (@byt3bl33d3r)

Acknowledgments, Contributors & Involuntary Contributors

(In no particular order)

- @devinmadewell for some awesome modules.
- @RemiEscourrou for some awesome modules.
- @nicolas_dbresse a.k.a @Daudau for contributing an insane amount of modules.
- @C_Sto for helping me with some of the .NET ECDHE implementation details and keeping my sanity.
- @davidtavarez for making some amazing contributions including a cross-platform stager.
- @mcohmi a.k.a daddycocoaman, for being awesome and making code contributions including modules.
- @SkelSec for the amazing work on Pypykatz and for being a general inspiration.
- @cobbr_io for writing SharpSploit which was heavily used as a reference