GitHub - antonioCoco/SharPyShell: SharPyShell - tiny and obfuscated ASP.NET webshell for C# web applications

By antonioCoco

Archived: 2026-04-05 13:02:18 UTC

SharPyShell is a tiny and obfuscated ASP.NET webshell that executes commands received by an encrypted channel compiling them in memory at runtime.

SharPyShell supports only C# web applications that runs on .NET Framework >= 2.0
VB is not supported atm.

Usage

```
 python3 SharPyShell.py generate -p somepassword
python3 SharPyShell.py interact -u http://target.url/sharpyshell.aspx -p somepassword
```

Requirements

Python version >= 3.6

and

```
pip3 install -r requirements.txt
```

Description

SharPyShell is a post-exploitation framework written in Python that are capable of:

- Generate obfuscated webshell (generate);
- Simulate a windows terminal as an interaction for the webshell (interact).

The main aim of this framework is providing the penetration tester a series of tools to ease the post exploitation phase once an exploitation has been succesfull against an IIS webserver.

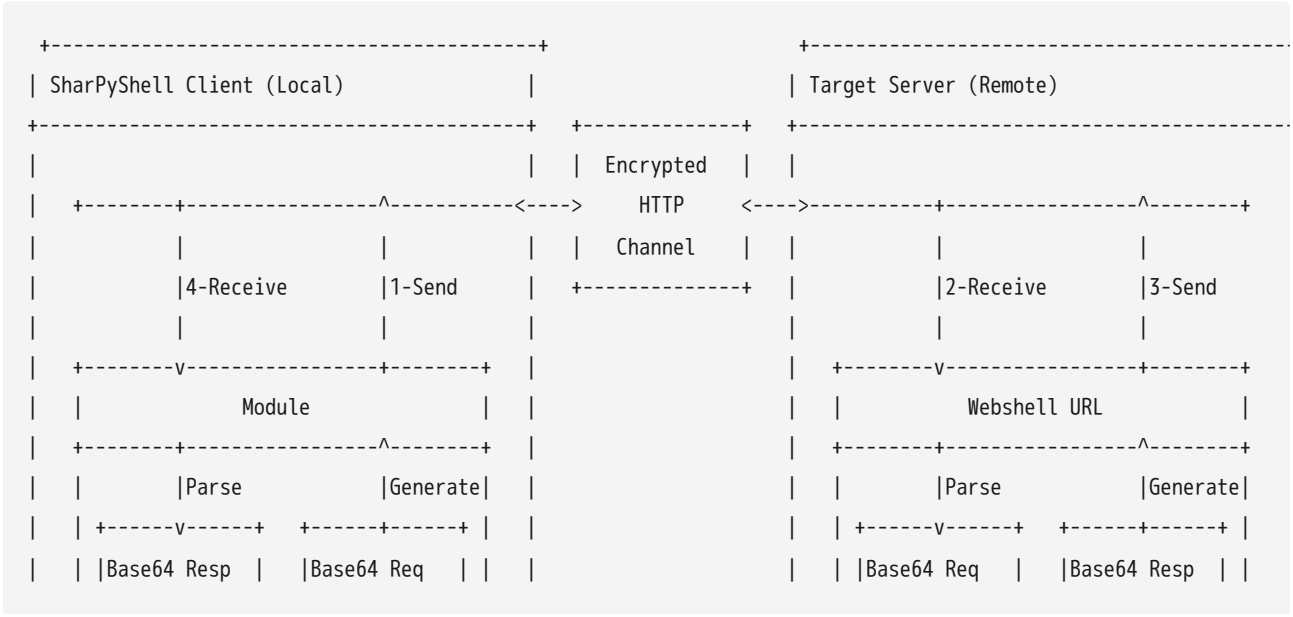
This tool is not intended as a replacement of the frameworks for C2 Server (i.e. Meterpreter, Empire, ecc..) but this should be used when you land to a fully restricted server where inbound and outbound connections are very limited.
In this framework you will have all the tools needed to privesc, netdiscovery and lateral movement as you are typing behind the cmd of the target server.

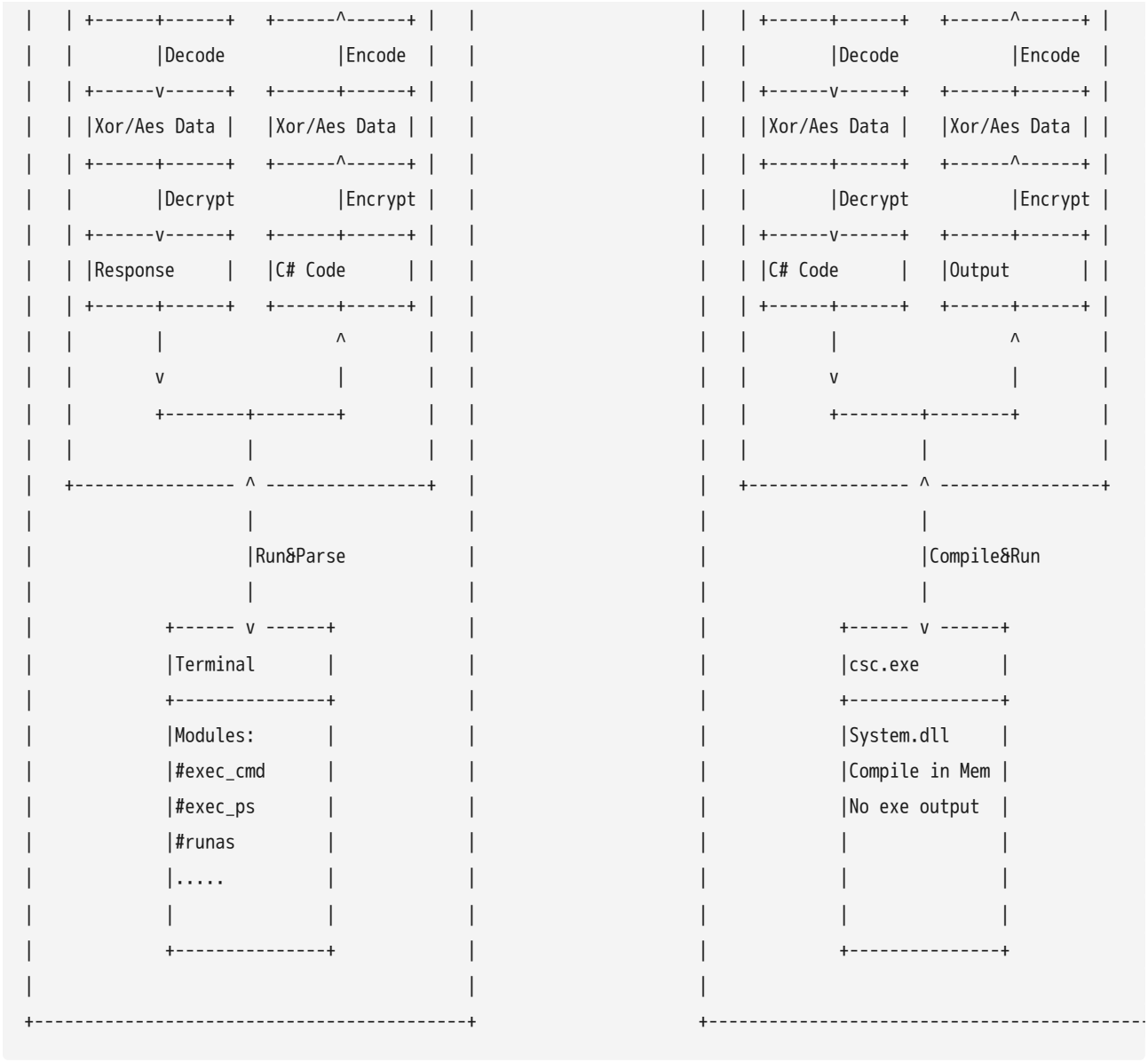
Moreover this framework aim to be stealthy as much as possible implementing in memory execution for c# code and powershell modules.

The obfuscation implemented in SharPyShell aim to evade both file signatures and network signatures ids.
For the network signatures evasion, a fully encrypted channel has been developed for sending commands and receiving outputs.
The evasion for file signatures has been achieved using Reflection on a precompiled dll in charge of runtime compiling c# code.

Technical Diagram

Generated with asciiflow.com





Modules

```
 #download               Download a file from the server
#exec_cmd               Run a cmd.exe /c command on the server
#exec_ps                Run a powershell.exe -nop -noni -enc 'base64command' on the server
#inject_dll_reflective  Inject a reflective DLL in a new (or existing) process
#inject_dll_srdi        Inject a generic DLL in a new (or existing) process
#inject_shellcode       Inject shellcode in a new (or existing) process
#invoke_ps_module       Run a ps1 script on the target server
#invoke_ps_module_as    Run a ps1 script on the target server as a specific user
#lateral_psexec         Run psexec binary to move laterally
#lateral_wmi            Run builtin WMI command to move laterally
#mimikatz               Run an offline version of mimikatz directly in memory
#net_portscan           Run a port scan using regular sockets, based (pretty) loosely on nmap
#privesc_juicy_potato   Launch InMem Juicy Potato attack trying to impersonate NT AUTHORITY\SYSTEM
#privesc_powerup        Run Powerup module to assess all misconfiguration for privesc
```

```
#runas                  Run a cmd.exe /c command spawning a new process as a specific user
#runas_ps               Run a powershell.exe -enc spawning a new process as a specific user
#upload                 Upload a file to the server
```

Windows version tested

Windows Server >= 2008 Standard x64

Credits

- @newfinal100 (for the fancy logo!)
- @weevely3
- @juicy-potato
- @PowerSploit
- @mimikatz
