Improving the Post Exploitation API

GSoC 2021 - Metasploit

Objective

The objective of the project is to improve the post exploitation API which is the part of metasploit framework with which the user interacts after getting the shell from target. However, at some places it is not much smooth and can be improved. One such area would be to add methods for shell sessions which are present in meterpreter filesystem API. That will make it easier for module writers to use file system API without taking care of session type.

Unified File System API

Currently if a filesystem API is present for meterpreter and not for shell the module writer has to drop the idea to use that API as it won't be compatible with shell sessions.

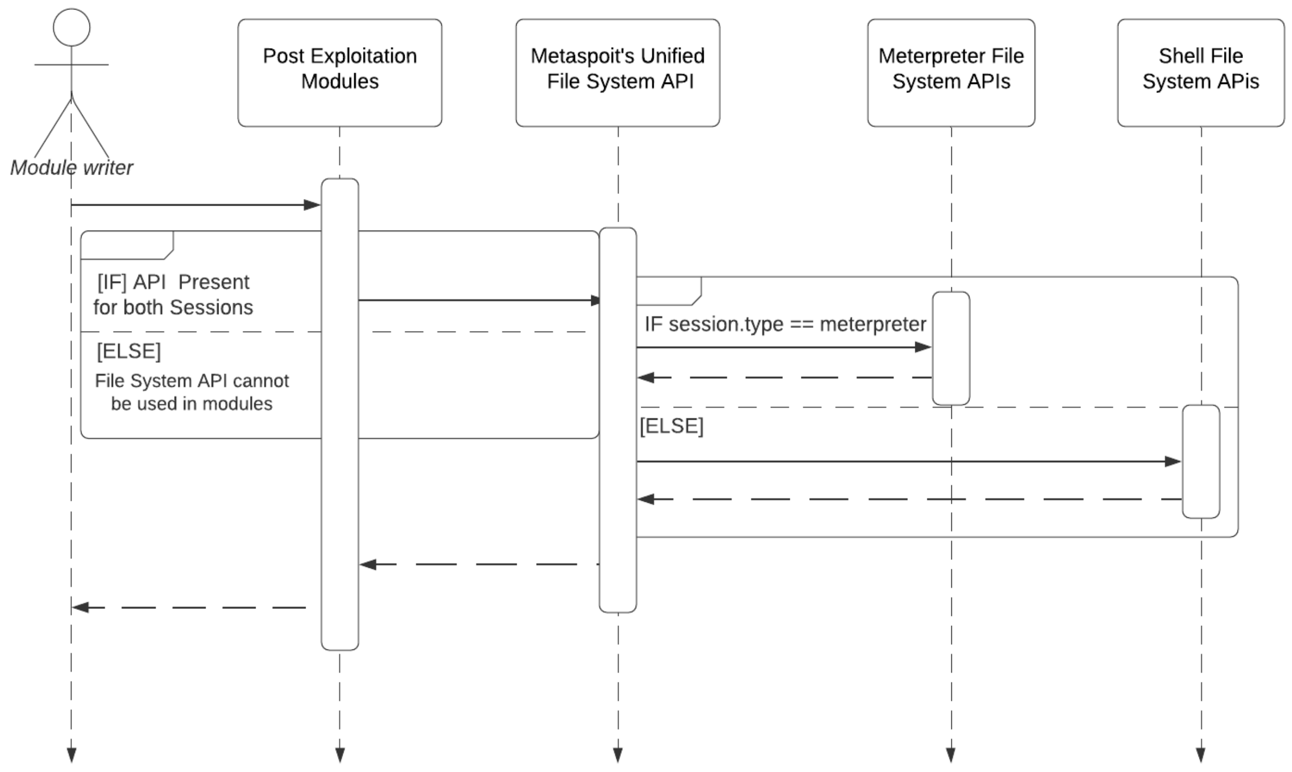

Module writer
Post Exploitation Modules
Metaspoit's Unified File System API
Meterpreter File System APIs
Shell File System APIs
[IF] API Present for both Sessions
[ELSE]
File System API cannot be used in modules
IF session.type == meterpreter
[ELSE]

Figure 1: Current state of Post Exploitation File System API

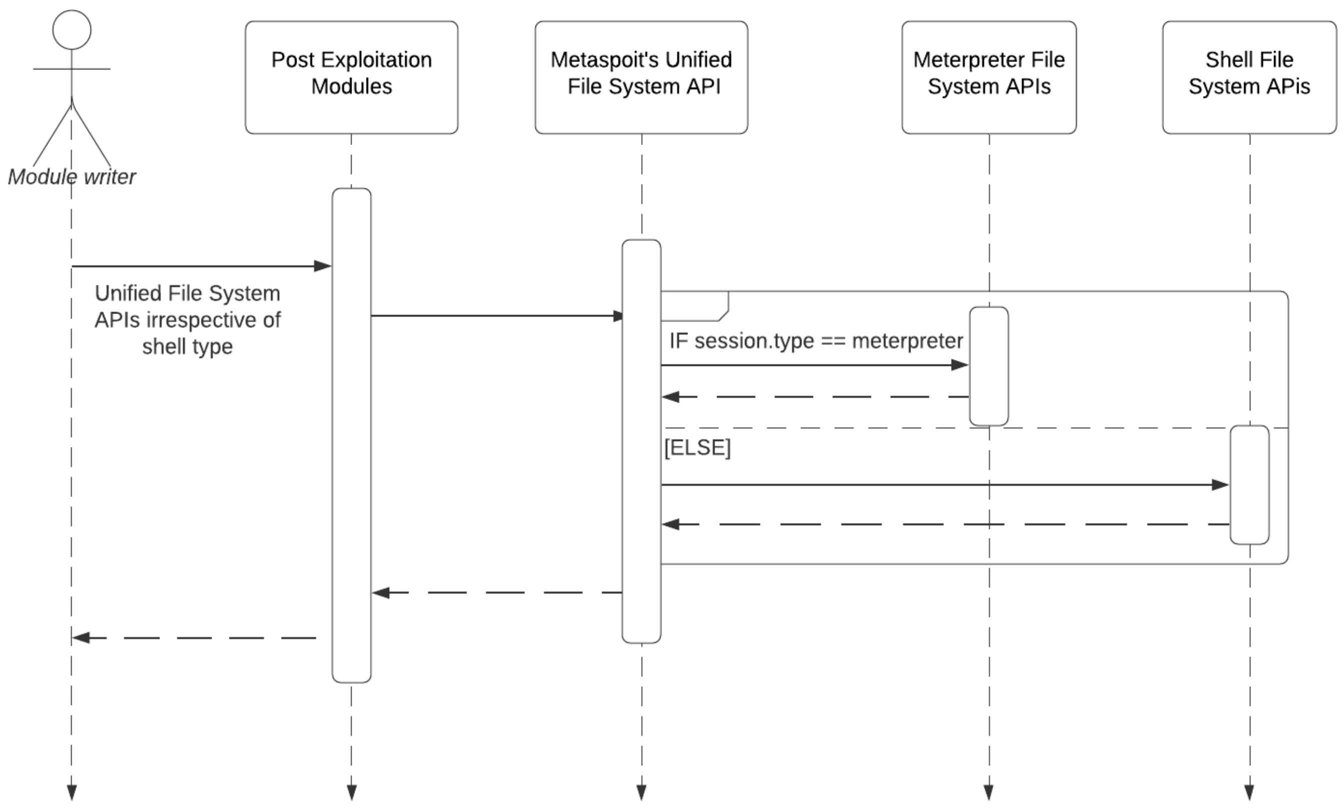

Module writer
Post Exploitation Modules
Metaspoit's Unified File System API
Meterpreter File System APIs
Shell File System APIs
Unified File System APIs irrespective of shell type
IF session.type == meterpreter
[ELSE]

Figure 2: After addition of missing Shell methods

A layer of abstraction can be added in order to make the modules use File System API without caring about what type of shell is obtained. The operations in File System API should return the same data, take the same type of arguments and fail under the same conditions irrespective of session type.

Some of the methods which are missing for shell sessions are :

- Search : Searches for files containing matching patterns on a remote system, optionally recursive.
- Download : Downloads set of files from a remote system.
- Upload : Uploads a set of files to a remote system.
- Stat : Returns a stat object which has very useful methods to query file attributes like *setuid?*, *directory?* and many others.

For example search command can be implemented with the help of recursive listing and fetching name, path and size of the files. Stat class can be implemented with the help of *stat* command on unix systems. A Proof of Concept can be found here

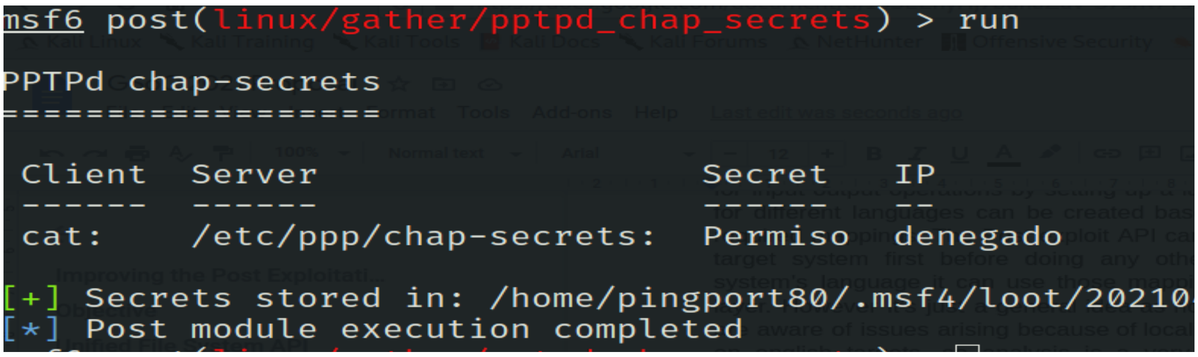
Localization Support

Operating Systems support many different languages. If the session is from an OS of different language, there can be some issues where the module might break because of wrong input output strings sent in the process of post exploitation as different languages might have different names for the same thing.

This can be improved by adding the support for localization. Both the language which system is using and the language of the current user can be queried with the help of windows registry keys. The strings in the post exploitation operations can be modified based on the language of the remote system. A mapping could be added to point the modification required for a specific language. For example, if the module needs to use the path which includes “Program Files” which may vary with locale, this feature should substitute it with the appropriate string based on the locale of the target system.

The general idea is to first analyze the modules and strings needed to be changed for input/output operations by setting up a lab with non-english systems. String files for different languages can be created based on that analysis which will have the required mappings. The Post exploit API can be made to query the language of the target system first before doing any other operation and based on the target system’s language it can use those mappings for the modules, being the middle layer. However it’s just a general idea as not many community members and users are aware of issues arising because of localization as most of the people work on English targets, so analysis is a very big part of what exactly needs to be implemented. But this part will be done in the way that it will be easily extended for other languages by adding string files.

One such place where localization is the cause of the issue is the ‘pptpd_chap_secrets.rb’ module, where the module searches for ‘permission denied’ like English messages which may vary with locale. So when run against non-english target system it will treat the error message as data. In the image below the phrase “*permiso denegado*” which is “*permission denied*” in Spanish is treated as data.



```
msf6 post(linux/gather/pptpd_chap_secrets) > run

PPTPd chap-secrets
==================

 Client   Server                   Secret    IP
 ------   ------                   ------    --
 cat:     /etc/ppp/chap-secrets:   Permiso   denegado

[+] Secrets stored in: /home/pingport80/.msf4/loot/20210
[*] Post module execution completed
```

Figure 3 : “*permiso denegado*” being treated as data

Timeline

Pre GSoC (Before May 17th)

- Read the wikis for development.
- Went through many parts of the codebase like protocol libraries, console related libraries, scanners etc.
- Explored the codebase, found some issues and fixed some.
- Had some discussions with mentors and community members about the codebase and future enhancements.
- Explored some libraries which are external to metasploit like rex-text.

Previous Contributions

- https://github.com/rapid7/metasploit-framework/pull/14917
- https://github.com/rapid7/metasploit-framework/issues/14915
- https://github.com/rapid7/metasploit-framework/issues/14922
- https://github.com/rapid7/metasploit-framework/pull/14927
- https://github.com/rapid7/rex-text/pull/41
- https://github.com/rapid7/metasploit-framework/pull/15000

Community Bonding Period (May 17th - June 7th)

- Introduce myself and this project on Slack#GSoC
- Set up the environment.
- Go through the code and try to add features and find issues in the existing codebase to understand it better.
- Setup a new log for writing activities I do for the entire Summer.
- Go through the core Post Exploitation Libraries of various operating systems and the common ones.

Official Coding Period (June 7th - August 16th)

June 7th - June 13th

- Gather the FileSystem methods which shell sessions don’t have.
- Analyze those methods based on their functionality, parameters and return values.

June 13th - June 23rd

- Add the file operations which are present in meterpreter and not in shell sessions.
- Document and write tests for the functionality added.

June 23rd - July 8th

- Set up a lab with various non-english systems to analyze the necessary modifications required by localization.(eg. If a module is using some file in `C:\Program Files` it should be changed to `C:\Programmes` in case it is Windows French.)

Deliverables for First Evaluation(July 12th - July 16th)

- Unified File System API for Meterpreter and Shell Sessions.

July 8th - July 22nd

- Write APIs for adding/modifying strings for compatible input and output operations for non-english systems.
- Collect codes of different languages supported on windows.
- Write mappings to point to APIs based on language codes.

July 22nd - August 2nd

- Fix encoding errors when non 7-bit ascii characters are given.
- Deliver the documentation and tests.

August 2nd - August 16th

- Buffer Time for unexpected delay or emergency.
- Fix bugs and update documentation.

Deliverables after Final Evaluation(August 16th)

- Localization support.

Post GSoC (After August 16th)

- Continue as an active contributor and remain in touch with community members and mentors.
- Fix the FTP library duplication issue which I found in the Pre GSoC period.
- Start exploring the meterpreter codebase.
- Document the parts of the codebase which I come across.

About Me

I study Computer Science at Vidyavardhaka College of Engineering, Mysuru. I'm also a Cybersecurity student and Open Source enthusiast. I love playing CTFs and cyber war games on platforms like HackTheBox, Rootme and Vulnhub. I like to understand how things work and how we can break it. I like reading code in various languages. I've also conducted some workshops on Pentesting, Linux and Git at local Open Source Community(OSL), which is a student run community of FOSS enthusiasts.

I've been playing CTFs for 1 year on different platforms like HTB, THM, PicoCTF and some others. I've also done two pentest assessments as an individual for my college which included their network and website. I was able to compromise the root domain, all the subdomains and various network devices like CCTV cameras, routers, printers and some others. I have written one exploit for CVE-2019-17240 in python which bypasses the login limit protection and have also built a small terminal based chat system with the help of ruby, multithreading and socket Programming.

I've participated in some conferences and seminars like ACM winter School on Cyber Security. I like writing scripts and automating stuff. I am interested in all branches of cyber security but the branch which interests me the most is Reversing, System Internals and Binary Exploitation.

I'm also a moderator at Open Source Community, and attend many open source meetups every year. Open Source Lab(OSL) is also a place where I work before and after college hours, and learn and discuss new concepts and technologies with other members. We also organize and participate in coding contests and monthly code sprints.

Apart from all that my hobbies also include watching football and playing PC games.

Skills

- Major Pentesting tools
- Git
- Ruby
- Linux Privilege Escalation
- Windows Privilege Escalation
- Computer Networks
- Network Pentesting
- Web Pentesting
- Binary Exploitation
- OSINT
- PHP
- Python
- C
- SQL

GSoC Participation

This is the first time I’m applying for Google Summer of Code

Personal Details

Name: Gaurav Purswani
College: Vidyavardhaka College of Engineering, Mysuru, India
Email: gauravpurswani1234@gmail.com
Country: India
Github/Twitter/freenode: pingport80
Timezone: IST (GMT + 5:30) 2
Languages: English, Hindi, Sindhi