

CSE 545 - Software Security - Portfolio Report

Arizona State University
Christopher Richard Bilger
ASU ID: REDACTED

Abstract - This paper will review the problems, solutions, design considerations, personal contributions, and lessons learned from the project given during the CSE 545 course.

1. Introduction

The project for CSE 545 Software Security was to design and implement tools that would aid my team (Team 31) during the Project Capture The Flag event. During my team's preliminary Zoom conference calls, I had the idea of creating a script runner in Python. The intent was for the script runner to automate and drastically simplify the process of discovering vulnerabilities in the running services, writing code that will exploit those vulnerabilities, and creating patches for our team's services to remove the vulnerabilities that we found. Through this portfolio report, I will cover the project completed in this course, my role in completing the project, and what I have since learned and will continue to use from the project.

2. Solutions

The end product that my team ended up creating was a modular (by design) Python 2/3 script that enabled us to create and run various other scripts and tools. Each of the individual scripts/tools that we created were able to be used stand-alone or through the script runner. Below, I will explain the design and implementation of the script runner as well as some of the individual modules. There are numerous other scripts/modules that we created for this project, but they ended up not being as useful during the CTF event as we would have preferred.

1. For the first section of this project (the script runner^[2]), we worked on implementing a modular import system that would import all python files in a specific direction and then would run those python scripts with the default and user-supplied arguments. During this section of the project, we

also created a template script that aided in simplifying the future creation of scripts/modules. The first two images below are showing how the modules are dynamically imported and how the script runner argument system functions. The argument system that we created takes in an argument **m** as the module name and can handle optional arguments, **a**, in the form of [arg1] [arg2] [...]. There is also an optional list argument that will display helpful information about all of the dynamically imported modules.

```
module_name = "modules.{0}".format(pathlib.Path(potential_module).stem)
imported_module = importlib.import_module(module_name)
```

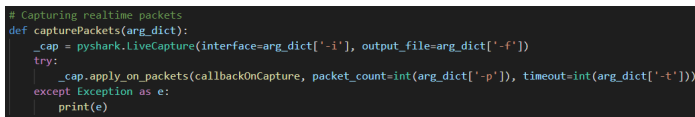
Fig. 1. Code snippet showing the dynamic import of script-runner modules

```
parser = argparse.ArgumentParser(description='ASU CSE 545 - Team 31 - CTF Project')
parser.add_argument("-a", "--module", help="the module to run", dest="module", metavar="module", nargs=1)
parser.add_argument("-s", "--args", help="the arguments to run with the module", dest="args", metavar="args", nargs="*", default=[])
parser.add_argument("-l", "--list", help="list all of the available modules to run", dest="list", action="store_true")
args = parser.parse_args()
```

Fig. 2. Code snippet which lists the available command-line arguments for the script-runner

2. The second section of the project was the longer of the two sections. This consisted of designing and programming the individual modules that we intended to use during the CTF event. Below I will explain the solutions that my team came up with for the net-analyzer and the template-exploit.
3. The “net-analyzer”^[3] script was intended to be a network traffic analyzer that we would use to check whether specific network activity was from the game bot or the other teams. This module unfortunately was not very helpful during the CTF event because we didn't end up needing to analyze any network

traffic or activity. I think that this module would be more helpful during a CTF event that has a higher emphasis on network-related detection and authorization. If we had more time during the CTF to focus our efforts on network-related data processing, then I think that we might have been able to detect and identify the packets from the game bot to separate them from the other teams. The idea was, that we would then be able to block packets from other teams while still enabling the game bot to operate as it normally would. Below is a photograph of the “pyshark” Python module capturing network packets in real-time and sending them to a handler function. As you can see from the picture, we also allowed the user that is running the module a high degree of customizability for the packets that they wanted to capture.



```
# Capturing realtime packets
def capturePackets(arg_dict):
    _cap = pyshark.LiveCapture(interface=arg_dict['-i'], output_file=arg_dict['-f'])
    try:
        _cap.apply_on_packets(callbackOnCapture, packet_count=int(arg_dict['-p']), timeout=int(arg_dict['-t']))
    except Exception as e:
        print(e)
```

Fig. 3. Function showing how packets are captured in realtime with configuration capabilities

4. The “template-exploit” script was extremely helpful during the CTF event. This script automated the execution of the exploit that we supplied to it as well as automated the submission of flags to the SWPAG^[4] endpoint. This script was designed from the beginning to allow the exploit creator to easily copy and paste the template to a new python file, add their exploit code, and then run the script to see if it was a successful exploit. Below is a picture of the template opening a connection to the SWPAG client, iterating over all of the teams’ services, running the exploit code for our team to obtain the open ports, and information about those ports that the individual services were running on. We were

not aware until it was already designed and built, that the service ports would be known in advance and that running port scanning was essentially unnecessary for this competition. I think that this script is still very useful for other competitions where the service ports are not known in advance because it will allow the team(s) running it to find port information very quickly and without the other teams knowing that we are scanning them.

3. Results

I had some very interesting findings occur during the creation of my team’s project as well as the usage of our project during the CTF event. First of all, the project that we made was highly modular and horizontally scalable, meaning that adding and removing modules is incredibly easy to do. This lent itself to efficient usage of our time during the CTF event because our tools became an extension that we could use, but that didn’t distract us from our tasks. I found that creating tools before the CTF event, and I would imagine all CTF events, allows the players to communicate much more effectively and to work on vulnerability analysis and patching those vulnerabilities. I think that this project and others like it are incredibly important as they allow you to “dive straight into” the CTF competition without fumbling around with trying to manually backup services, install programs and plugins, get your environment setup, etc. Once the CTF event was over I read through our codebase and found numerous places where we could have lowered the tool’s friction even more. I found that if I were to go back and redo my team’s project with the knowledge that I have now I would have spent time exploring open source tools that have already been written for CTF competitions and used those as a baseline for the specific type of tools that are used most frequently. One final note about the results that I found from my team’s project during the CTF event is that our project guided us to find the vulnerabilities of our services and then to exploit them on the other teams. That is, our project gave us a tremendous amount of flexibility for open communication and teamwork as opposed to guiding us to work independently, as I think some other projects might have been doing.

4. Contributions

I think that my overall contributions to the group project aided greatly towards our CTF event success. Through my time spent working on this group project, I contributed to numerous individual submissions. I think that I played a huge role in the development of our team's vision for our project as well as the implementation of both documentation and code. Below I will explain in further detail the individual contributions that I made towards the success of my team's project.

Team Project Proposal: I wrote up the majority of the team project proposal document, which I then later submitted through Coursera. Of the five sections that were submitted in this document, I wrote four of them. These sections include the answers to questions numbered 1, 2, 4, and 5.

Team Project Status Update: This document ended up being shorter in length than the "Team Project Proposal", but my contributions were still made. I wrote the paragraphs that we submitted for the questions numbered 1 and 2.

Final Team Report: Much to the same effect as the previous two documents, especially the "Team Project Proposal", I wrote the majority of the answers to the questions given in this document. Questions numbered 1, 2, 3, and 4 were answered by myself and then further refined by my team members.

Project CTF: The PCTF portion of the project is by far the largest section to cover as far as my contributions are concerned. To start, I created, maintained, and shared the initial code repository with each of the team members as soon as my team (Team 31) was formed. I enabled full access to the GitHub^[1] repository for all team members so that there would be no barrier to entry. The initial commits that I made on the repository formed the core of the current script runner. I added an initial "example" script/module that was easily copy-and-pasted as needed. I wrote high-level instructions in the README.md file which explain how the program works and how it can be added to in the future. Once the design of our scripts was

completed, I created the auto-runner, flag-submitter, target-info, vm-info, service-backup, and template-exploit modules. During the CTF event, I worked with multiple team members to create the backup-exploit, sampleak-exploit, and flaskids-exploit modules. These three modules were based on the foundation of the template-exploit module. Once the CTF event was over I committed those three exploit modules as well as the script.sh and script2.sh executable files that can be found in the root directory of the repository.

5. Lessons Learned

The work that I contributed to this project helped me to better understand some core fundamentals of networking and how networks can be penetrated without software systems noticing. My career is currently in the web development field of software engineering and I plan to, eventually, transition into the web security field. I can now say that I have a much deeper understanding of web-related hacking such as XSS, SQL injection, and broken authentication mechanisms. Before taking this course and participating in this project I thought that I understood web technologies enough to prevent these known classes of vulnerabilities, but now I know for certain that I know to safely implement protections against these types of vulnerabilities as well as others. As I have previously said, this project has taught me that I am capable of succeeding in the cybersecurity field and it showed me how fun and interesting cybersecurity is. I learned that my security interest is not waning and that I can directly use the newfound skills from this course in my day-to-day life as a software engineer. I am passionate about cybersecurity and because of this project, I am now signing up for and interested in completing more hacking challenges and competitions.

Team #31:

Captain: Chris Bilger
Hao Deng
Arkodeb Maity
Ganesh Natarajan
Louis Neira
James Russell

References

- [1] “Where the World Builds Software.” GitHub, github.com/.
- [2] ChristopherBilg. “ChristopherBilg/Cse-545-PCTF-Team-31.” GitHub, 11 Feb. 2021, github.com/ChristopherBilg/cse-545-PCTF-team-31/blob/master/runner.py.
- [3] ChristopherBilg. “ChristopherBilg/Cse-545-PCTF-Team-31.” GitHub, github.com/ChristopherBilg/cse-545-PCTF-team-31/blob/master/modules/net-analyzer.py.
- [4] Shellphish. “Shellphish/Ictf-Framework.” GitHub, 31 Mar. 2020, github.com/shellphish/ictf-framework/blob/master/teaminterface_client/swpag_client/client.py.