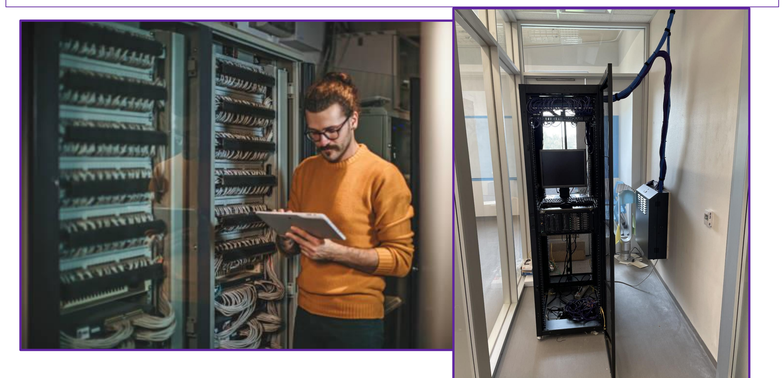
Broadening the Knowledge Cybersecurity Students Utilizing Simulated Server Exploitation

William Lister, Dr. Christopher Ivancic
Department of Computer Science, Stephen F. Austin State University

Abstract

This project aims to gamify concepts from cybersecurity, particularly the use of the Linux command line interface. Users are given an objective to complete using Linux command where they obtain points upon completion.

The experience, nicknamed “LumberHack”, is designed to be deployed on the Stephen F. Austin State University Computer Science server, where students can learn individually, and gain experience in using a Linux server.



Benefit

Aspiring cybersecurity enthusiasts often learn about the tools used to prevent malicious attacks, but performing the attacks allows for individuals to think from a different perspective.

Performing sanctioned attacks on an intentionally vulnerable set of programs lets cybersecurity students understand what is happening on the other side of the screen and puts them in the mindset of a potential attacker.

Command Line Interface

Command Line Interface (CLI) is a text-based user interface used to run programs, manage computer files, and interact with the computer. It does not have icons, a pointer, or a graphic user interface (GUI).

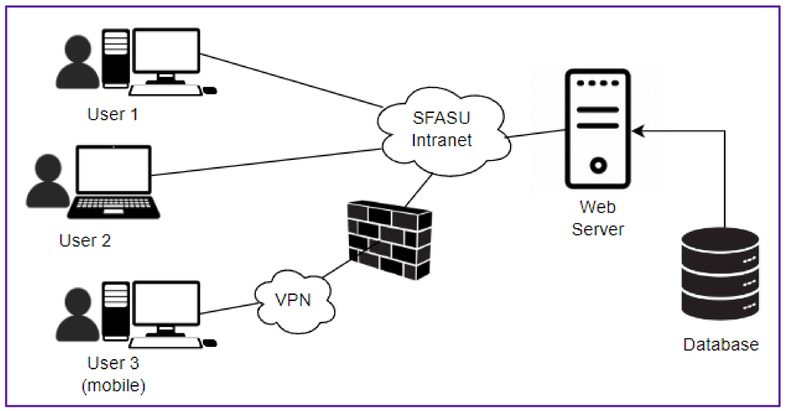
```
┌──(root㉿kali)-[/etc]
└─# cd /etc

┌──(root㉿kali)-[/etc]
└─# ls -la
total 1588
drwxr-xr-x 186 root     root      12288 Apr
drwxr-xr-x  18 root     root       4096 Apr  2 18:24 ..
```

Capture The Flag

The project employs the Capture the Flag (CTF) game style. In particular, the type of CTF games involves computer science and finding a hidden String of text within files.

LumberHack consists of multiple intentionally vulnerable programs, documents, or other files that hide a secret ‘flag’ in it. The objective is for the user to hack, decrypt, or break the files to gain access to the flag by any means necessary from the Linux CLI.

Once a user captures their flag, they can submit the flag to the server for points. Harder problems net the user more points while easier problems give fewer.



Prototype

Depicted below are instances of an attempt at the second problem created. The user first generates the problem, and the program informs the user about what happened and a hint to solve the problem.

In this case, a note about searching for a word or phrase in a text file is shown.

```
┌──(kali㉿kali)-[~/Documents/sfaCS/sfaCSkali]
└─$ generateP2
New file problem2.txt created.
Hint: Can I search for a word within a file?

┌──(kali㉿kali)-[~/Documents/sfaCS/sfaCSkali]
└─$ ls
problem2.txt

┌──(kali㉿kali)-[~/Documents/sfaCS/sfaCSkali]
└─$ cat problem2.txt
%eNBXjmMdf*a}k0kmg7q9|~GO⊣_gOmSLA[Rw"e= _e
                                        Csy/zl5ZYk%z`M
                                                    8$
```

```
┌──(kali㉿kali)-[~/Documents/sfaCS/sfaCSkali]
└─$ ls
problem2.txt

┌──(kali㉿kali)-[~/Documents/sfaCS/sfaCSkali]
└─$
sfaCS{gr3p_1s_c00l_4f9ceb21}

┌──(kali㉿kali)-[~/Documents/sfaCS/sfaCSkali]
└─$ submit problem2 sfaCS{gr3p_1s_c00l_4f9ceb21}
Problem solved!
```

To solve the problem, the user must use a Linux command to search the text file and submit the flag with the submit command.

To generate this problem, a file is made with random characters, the flag, and more random characters. The flag is unique to the user and problem, so flags cannot be shared.

```
# Problem number
prob = "problem2"
# Generate a random flag for the problem
flag = 'sfaCS{gr3p_1s_c00l_' + hashlib.sha256((str(os.getlogin()) + prob).encode()).hexdigest()[:8] + '}'
hashed_flag = hashlib.sha256(str(flag).encode()).hexdigest()  # Hash the flag

# Write the flag to the problem file
with open('problem2.txt', 'w') as f:
    f.write(''.join(random.choices(string.printable, k=random.randint(5000, 6000))))
    f.write('\n' + flag + '\n')
    f.write(''.join(random.choices(string.printable, k=random.randint(5000, 6000))))
```

Future Work

Currently, this program runs off a virtual machine on a local system with only a few attemptable problems.

Moving forward, LumberHack would be migrated to the SFA computer science server, problems can be added by authorized users of the system, a larger collection of problems, and a public scoreboard for all users of the system to see.

Future versions of LumberHack will also include an administrator panel to easily add programs and aliases, create new users on the system, and check each user’s submissions and total score.

Contact

William Lister
Department of Computer Science
1936 North St.
Nacogdoches, Texas 75965
listerwl@jacks.sfasu.edu

Acknowledgements

This research was supported by the College of Sciences and Mathematics as part of the 2024 Capstone Research at Stephen F. Austin State University. I’d like to extend a thank you to Stephen F. Austin State University for the opportunity to further my experience in Cybersecurity.

Thank you to the Computer Science department for the workspace and the equipment used throughout the entirety of the project. And a special thank you to Dr. Jeremy Becnel and Dr. Christopher Ivancic for their assistance and guidance in this project.

References

1. GitHub Repository (https://github.com/Williuhm/CaptureTheFlag)
2. PicoCTF (https://picoctf.org/)