CS 647: Counter Hacking Techniques

Pippin Assessment Report

Fall 2024

Aleyna Aydin
Nicholas Richmond
Shraddha Sawarna

9/15/24

Executive Summary:

This team was tasked with assessing the risk posed by the Pippin account to the virtual machines network. Upon logging into the student account in the network, they were required to find a file named "Pippingflag.txt." This involved navigating into the Pippin folder within their student account, decompressing and converting files until they discovered a text file containing a key that would grant access to the Pippin account.

Access was gained due to negligence in how the key was hidden. The key file was given a very telling name, "PippinsEyesOnly." By using the file command, the team identified the type of file, decompressed the folder, and moved files until they found evidence of a file containing the key. With that information, they SSH'd into Pippin's account and successfully retrieved the key file.

Pippin's method of hiding the key file relied on a technique known as "security through obscurity," where secrecy is used to conceal information. This approach is not secure because, if an attacker can figure out how the user obscured the information, they can uncover the key file and gain unauthorized access to the account. The team strongly recommends encrypting the key file to ensure it has active protection and enforcing strict access controls to prevent unauthorized access to accounts through other accounts on the same virtual machine or network.

In addition to the Pippin account access, it is crucial to address a separate security concern regarding Nmap, a technique used to identify open ports and services on a networked device. By sending specially crafted packets to various ports on a target system, Nmap determined which ports are active and which services are running. This process helped in mapping out the network and assessing the security of the target by revealing potential entry points and vulnerabilities. Through the use of the Nmap command, the team was able to recover the Pippinflag.txt file by searching for the smbclient that had permissions to read the file. Recommendations to prevent the team's findings would include enforcing strict network access controls. Both cases highlight the need for robust security practices, including encryption, timely updates, and stringent access management to safeguard sensitive information and systems.

1 Objectives

1.1 Pippin Account Access

The team was tasked with accessing the ‘pippin’ user account on the system. The provided file, SaveForPippin.zip**,** served as the initial point of entry for identifying potential vulnerabilities and planning the exploitation process.

1.2 Flag Recovery

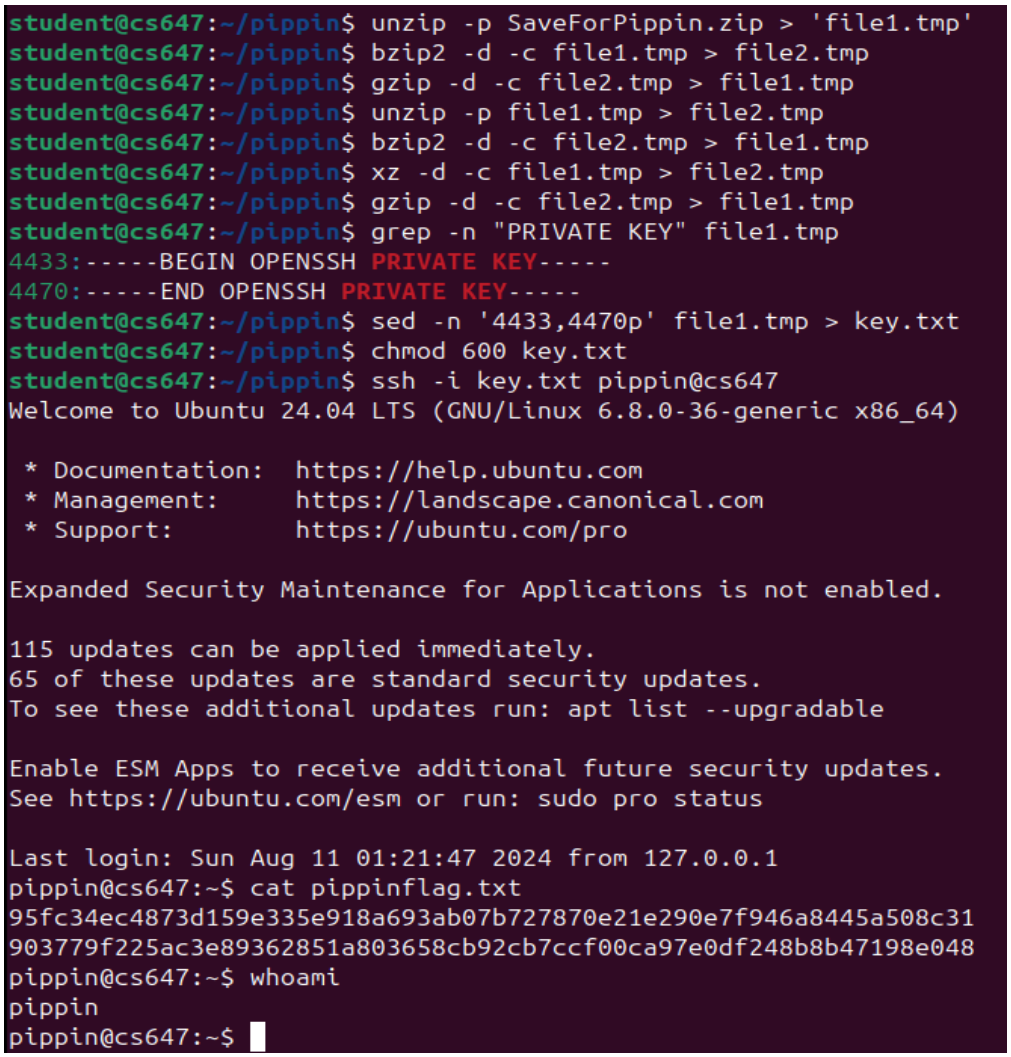
After successfully accessing the ‘pippin’ account, the next objective was to retrieve a flag file (pippin.txt) as proof of the exploit's success.

2 Attacks

2.1 SSH Key Exposure

Exploit #1	**SSH Key Exposure**	
Description	The file named SaveForPippin.zip contained a Secure Shell (SSH) key, which granted access to the 'pippin' user account. This key was hidden through several layers of compression and embedded within a 2.6-megabyte text file. Once the key was extracted and used to log into the 'pippin' user account, it allowed for the recovery of the flag.	
Objectives	1.1 - Pippin Account Access	
Assumptions	The file program could identify the compression algorithm applied at each stage of the file's compression process. If this assumption turned out to be incorrect, further analysis would be necessary to determine the compression method, but the overall strategy would still be applicable.	
Findings	The vulnerability discovered was that the Pippin user insecurely tried to protect their SSH key by embedding it within the SaveForPippin.zip file. Once the key was retrieved, allowing access to the account, the flag file was found in the user’s home directory at /home/pippin/pippinflag.txt.	
Mitigations	To ensure secure SSH key management, follow these best practices: ● Encrypt SSH keys with a passphrase to add an extra layer of protection. ● Regularly rotate SSH keys to minimize risks associated with key compromise. ● Revoke any unused keys to prevent unauthorized access. These measures can help safeguard systems and maintain robust security.	
Tools Used	`file`	This command displays the file type if it matches a known signature.
	`unzip`	This command decompresses ZIP archive data. When used with the -p flag, this command extracts the file to stdout.
	`bzip2`	When used with the -d flag, this command decompresses bzip2 compressed data. When used with the -c flag, this command outputs to stdout.
	`xz`	When used with the -d flag, this command decompresses xz compressed data. The file name needs to have a “.xz” extension.

		When used with the -c flag, the command outputs to stdout, bypassing the extension requirement.
	`gzip`	When used with the -d flag, this command decompresses gzip compressed data. The file name needs to have a ".gz" extension. When used with the -c flag, this command outputs to stdout, bypassing the extension requirement.
	`sed`	This command extracts the SSH private key from the file. When used with the -n option and the argument ‘xxxx, xxxxp,’ this command prints only the lines between xxxx and xxxx to stdout.
	`grep`	This command searches a file for the string "PRIVATE KEY". When used with the -n flag, this command shows the line numbers of output lines.
	`ssh`	This command logs into to the pippin account on the system. When used with the -i flag, the command specifies a private key file to use when logging in. SSH requires that the file's permissions be set to 600.
Commands Used with Syntax	1. `unzip -p SaveForPippin.zip > 'file1.tmp'` 2. `bzip2 -d -c file1.tmp > file2.tmp` 3. `gzip -d -c file2.tmp > file1.tmp` 4. `unzip -p file1.tmp > file2.tmp` 5. `bzip2 -d -c file2.tmp > file1.tmp` 6. `xz -d -c file1.tmp > file2.tmp` 7. `gzip -d -c file2.tmp > file1.tmp` 8. `grep -n "PRIVATE KEY" file1.tmp` 9. `sed -n '4433,4470p' file1.tmp > key.txt` 10. `chmod 600 key.txt` 11. `ssh -i ./key.txt pippin@cs647` 12. `cat pippinflag.txt` 13. `whoami`	

Details	To develop the first version of the attack, the team used the `file` command to identify the types of files the zip files were to decompress them. Various compression methods or false tags were employed to obscure the security measures. The file was compressed using multiple formats such as bzip2 and gzip, requiring the team to unzip and move the contents between folders until they uncovered a directory containing information about a 2.6-megabyte text file with ASCII characters that included a key. The team then used the `~/.ssh` command to navigate to the SSH folder, where they searched for "PRIVATE KEY" to locate a matching file. This file revealed the start and end markers of the key on lines 4433 and 4470.  *Screenshot 1: Flag file exploit and recovery.*

```
student@cs647:~/pippin$ unzip -p SaveForPippin.zip > 'file1.tmp'
student@cs647:~/pippin$ bzip2 -d -c file1.tmp > file2.tmp
student@cs647:~/pippin$ gzip -d -c file2.tmp > file1.tmp
student@cs647:~/pippin$ unzip -p file1.tmp > file2.tmp
student@cs647:~/pippin$ bzip2 -d -c file2.tmp > file1.tmp
student@cs647:~/pippin$ xz -d -c file1.tmp > file2.tmp
student@cs647:~/pippin$ gzip -d -c file2.tmp > file1.tmp
student@cs647:~/pippin$ grep -n "PRIVATE KEY" file1.tmp
4433:-----BEGIN OPENSSH PRIVATE KEY-----
4470:-----END OPENSSH PRIVATE KEY-----
student@cs647:~/pippin$ sed -n '4433,4470p' file1.tmp > key.txt
student@cs647:~/pippin$ chmod 600 key.txt
student@cs647:~/pippin$ ssh -i key.txt pippin@cs647
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-36-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

115 updates can be applied immediately.
65 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Sun Aug 11 01:21:47 2024 from 127.0.0.1
pippin@cs647:~$ cat pippinflag.txt
95fc34ec4873d159e335e918a693ab07b727870e21e290e7f946a8445a508c31
903779f225ac3e89362851a803658cb92cb7ccf00ca97e0df248b8b47198e048
pippin@cs647:~$ whoami
pippin
pippin@cs647:~$
```

2.2 Nmap Port Scanning

<table>
<tr><th>Exploit #2</th><th colspan="2">Nmap Port Scanning</th></tr>
<tr><td>Description</td><td colspan="2">Once access is gained to the ‘pippin’ account, it was necessary to recover the contents of the pippinflag.txt file which is guarded by a passcode. Within the pippin account, access was granted to search through files and their contents to discover any information that would aid in the file’s recovery.</td></tr>
<tr><td>Objectives</td><td colspan="2">1.2 - Flag Recovery</td></tr>
<tr><td>Assumptions</td><td colspan="2">Through port scanning, specific ports should show open services that can be used to further investigate the discovery of the pippinflag.txt file. If this fails, there will need to be a change in approach since the attack did not provide any information of value.</td></tr>
<tr><td>Findings</td><td colspan="2">The service ‘Samba’ was running on ports 139 and 445. When parsing through the pippin account .cache directory, this same service is found. By then gaining access to the Samba client, pippinflag.txt can be recovered.</td></tr>
<tr><td>Mitigations</td><td colspan="2">To prevent port scanning on a machine:
Implement firewall rules to block unnecessary ports and set rate limits.
Implement an intrusion detection system to detect and alert scanning activity.
Implement network segmentation to isolate critical systems.
These measures can help safeguard systems and maintain robust security.</td></tr>
<tr><td rowspan="4">Tools Used</td><td><code>cd</code></td><td>This command allows movement into a directory.</td></tr>
<tr><td><code>exit</code></td><td>This command leaves all directories</td></tr>
<tr><td><code>nmap</code></td><td>This command scans for open ports and any service currently running on them. When used with -p flag, specific ports can be searched.</td></tr>
<tr><td><code>smbclient</code></td><td>This command accesses and interacts with shared resources on servers that use the Server Message Block (SMB) protocol</td></tr>
<tr><td>Commands Used with Syntax</td><td colspan="2"><code>ls -la</code>
<code>cat .passwords.csv</code>
<code>exit</code>
<code>nmap -p 139,445 -sV localhost</code>
<code>cd .cache</code>
<code>cd samba</code>
<code>smbclient //localhost/pippinshare -U pippinflag%eTAfkODkNLYg</code>
<code>ls</code>
<code>cat pippinflag.txt</code></td></tr>
</table>

Details	The procedure taken in the recovery of the pippinflag.txt file began with the searching of all files on the 'pippin' account. This includes both the hidden and unhidden files. Through these displayed files, the .passwords.csv file was found. This file included details including Pippin's private credentials and notes such as his bank account, password, email, etc. Most importantly the team found the local smb share and password necessary for logging into the Samba client. Once discovering these credentials, the team used nmap port scanning to scan the open ports, 139 and 445, which were shown running the service, Samba. After finding the Samba service in the directory '. cache' and subdirectory 'Samba,' the password found before was used to log into the client and finally allow access to the pippinflag.txt file since it only allowed permissions for Samba to access its contents. By exploiting the Samba client, we were able to display the file contents.

3 Flags

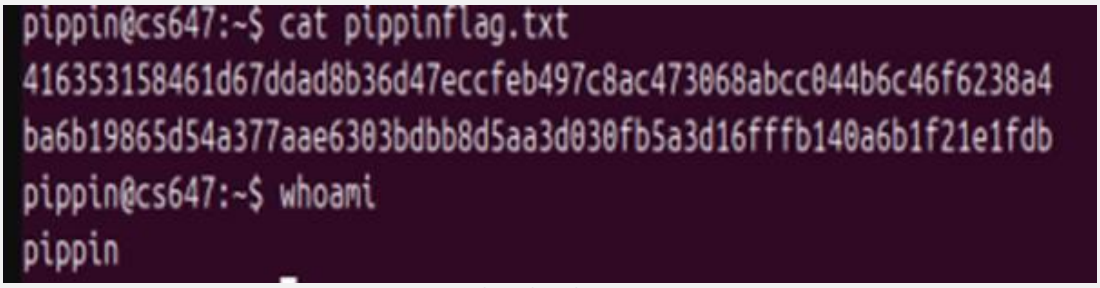
3.1 Aleyna Aydin

Flag Contents

416353158461d67ddad8b36d47eccfeb497c8ac473068abcc044b6c46f6238a4

ba6b19865d54a377aae6303bdbb8d5aa3d030fb5a3d16fffb140a6b1f21e1fdb

Screenshot



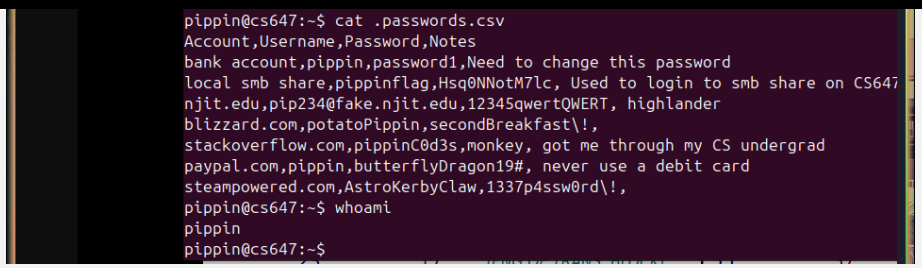
Screenshot 2: Contents of the flag file and the whoami *command.*

3.2 Nicholas Richmond

Flag Contents

95fc34ec4873d159e335e918a693ab07b727870e21e290e7f946a8445a508c31
903779f225ac3e89362851a803658cb92cb7ccf00ca97e0df248b8b47198e048

Screenshot



Screenshot 3: Contents of the flag file and the whoami *command.*

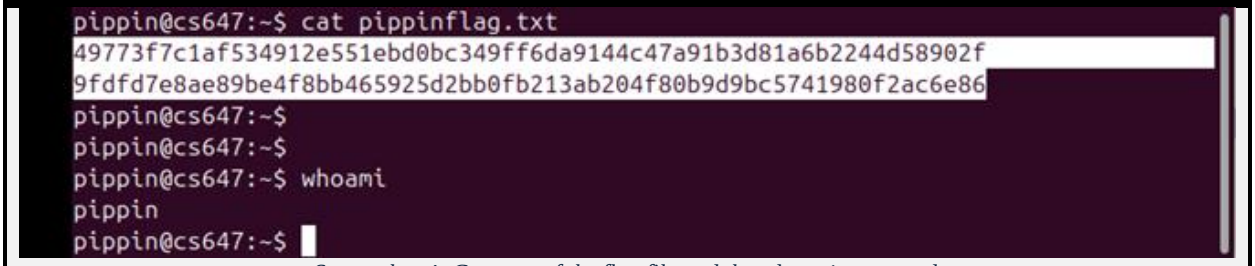
3.3 Shraddha Sawarna

Flag Contents

49773f7c1af534912e551ebd0bc349ff6da9144c47a91b3d81a6b2244d58902f

9fdfd7e8ae89be4f8bb465925d2bb0fb213ab204f80b9d9bc5741980f2ac6e86

Screenshot



Screenshot 4: Contents of the flag file and the whoami *command.*