VULNERABILITIES ASSESSMENT REPORT

Tyler Dao

SECU73000 – INTRODUCTION TO SOFTWARE SECURITY – SECTION 1

Contents

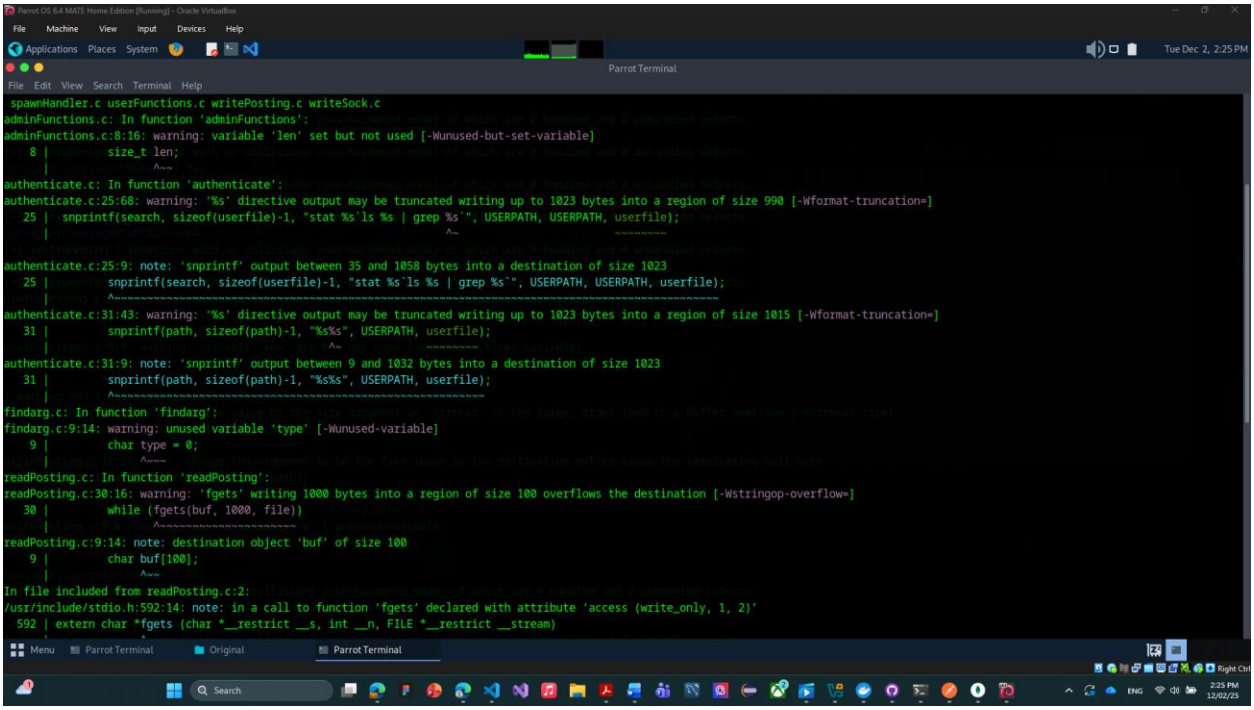
Code Review

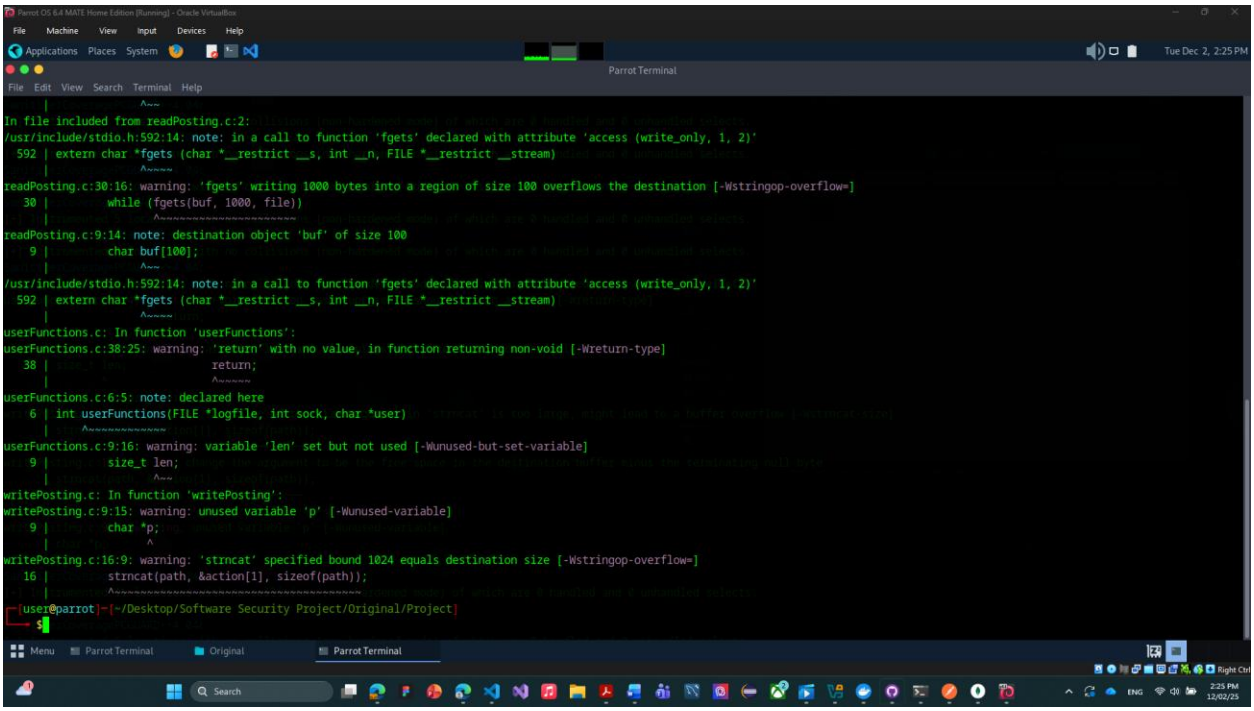
To run vulnerabilities assessment, I used g++ compiler to find buffer overflow issues, unused variable issues; as well as manually check the code to find security issues, these are issues found:

Issue no.	File Name	Line	What	Risk
1	adminFunctions.c	8	“len” variable is unused	Unused variable
2	athenticate.c	16	Wrong size for memset, in the code, it takes sizeof(1024) but it’s actually take the size of an integer which is 8 bytes	Buffer overflow
3	athenticate.c	19-20	Hard coded password	Attacker may guess the backdoor password
4	athenticate.c	25	snprintf() may write between 35 bytes to 1058 bytes into destination with size of 511 bytes	Buffer overflow
5	authenticate.c	26	system() runs search command without sanitizing	Injection attack → Attacker can inject a command in the username and run it on bash.
6	authenticate.c	31	snprintf() may write up to 1032 bytes into destination size of 1024 bytes	Buffer overflow
7	authenticate.c	48	Password is only checked by first 3 characters	Improper logic
8	findarg.c	9	“type” variable is unused	Unused variable
9	readPosting.c	14-15	Using strcpy and strcat may cause buffer overflow if action[1] is long	Buffer overflow

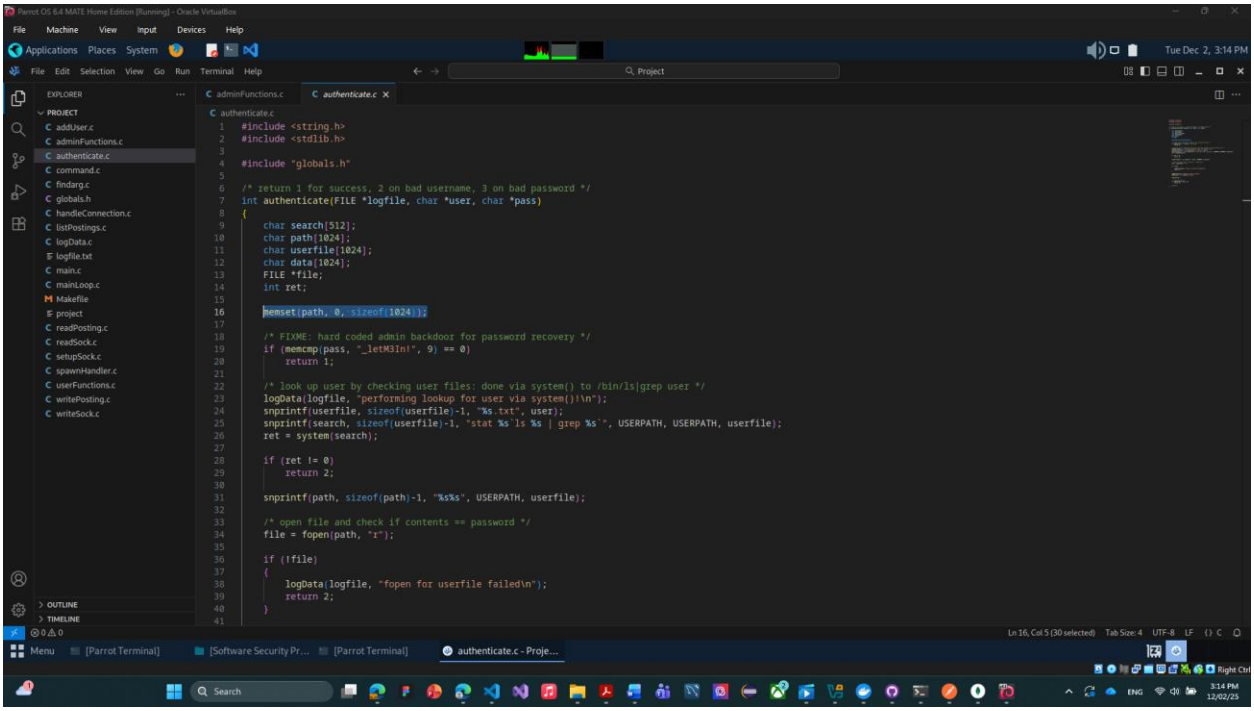
10	readPosting.c	30	Fgets() writes maximum 1000 bytes into a 100-byte buffer	Buffer overflow
11	userFunctions.c	9	“len” variable is unused	Unused variable
12	userFunctions.c	38	The function returns nothing, it should return an integer	Unchecked return
13	writePosting.c	9	“p” variable is unused	Unused variable
14	writePosting.c	16	strncat() might cause buffer overflow because it appends &action[1] into path, but path also has POSTINGPATH	Buffer overflow
15	writePosting.c	20	fopen() takes argument directly from action[1]	Dangerous method → Attacker can gain access to any file on the server.

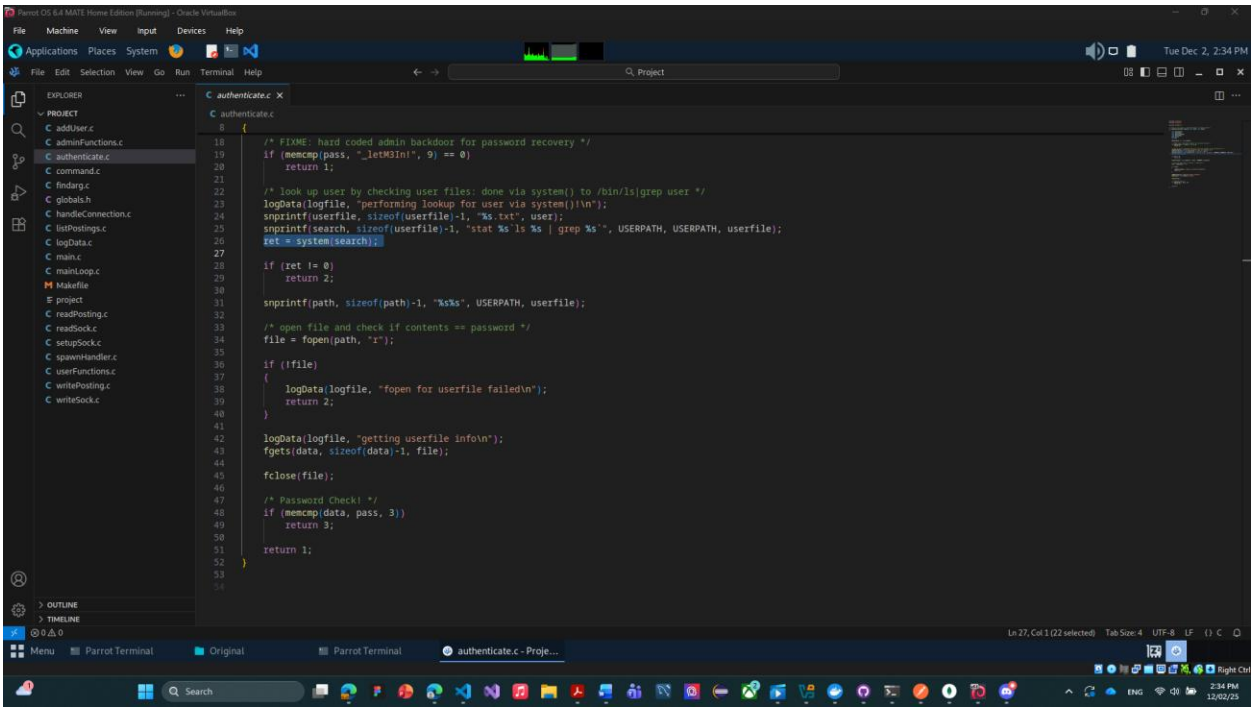
Issues found by g++ compiler

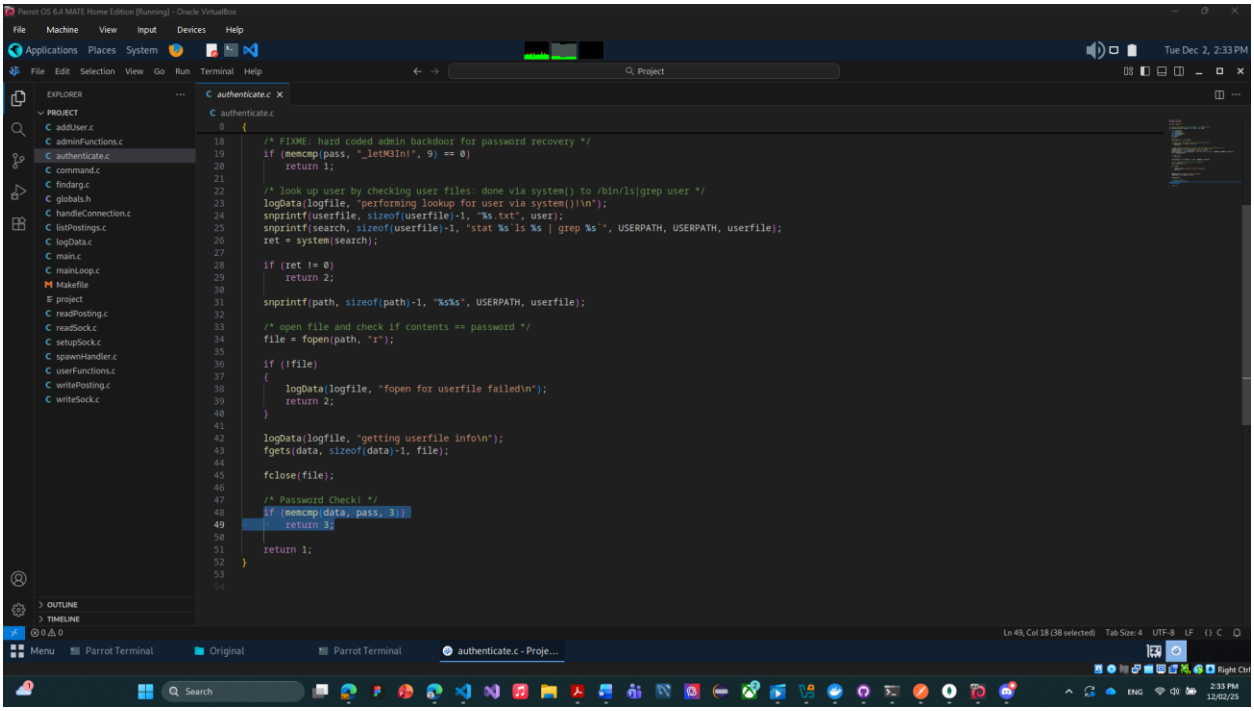


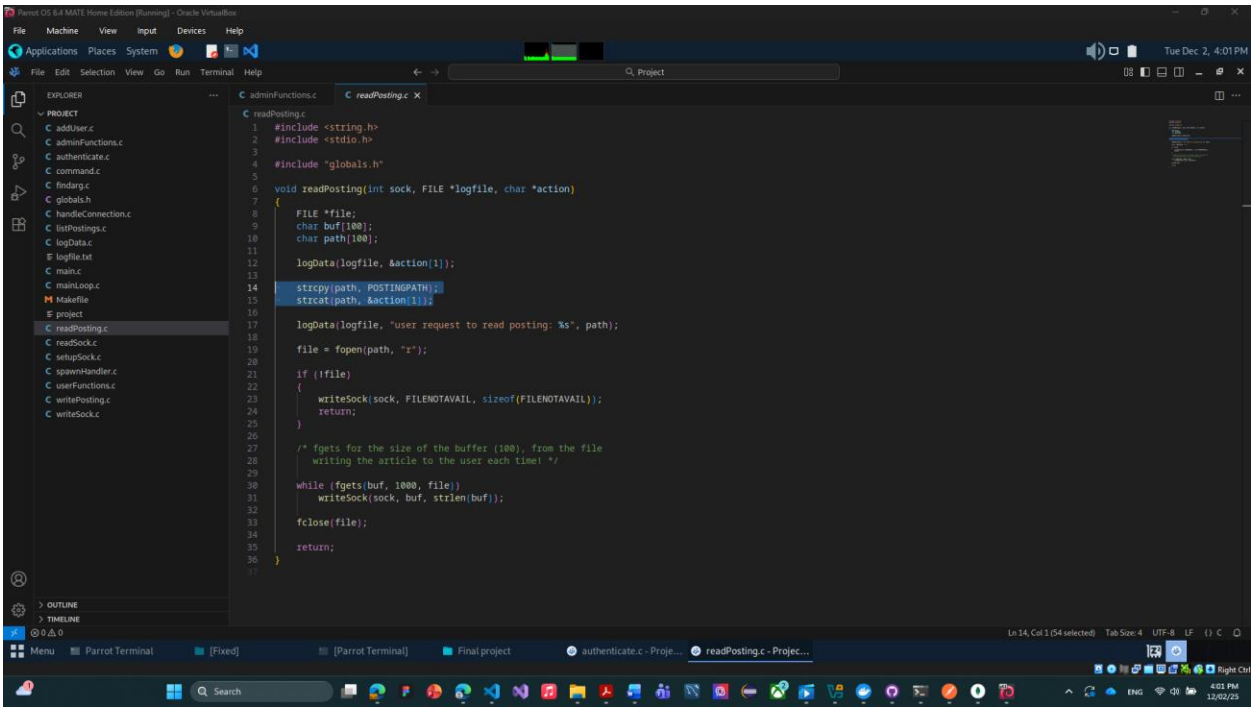


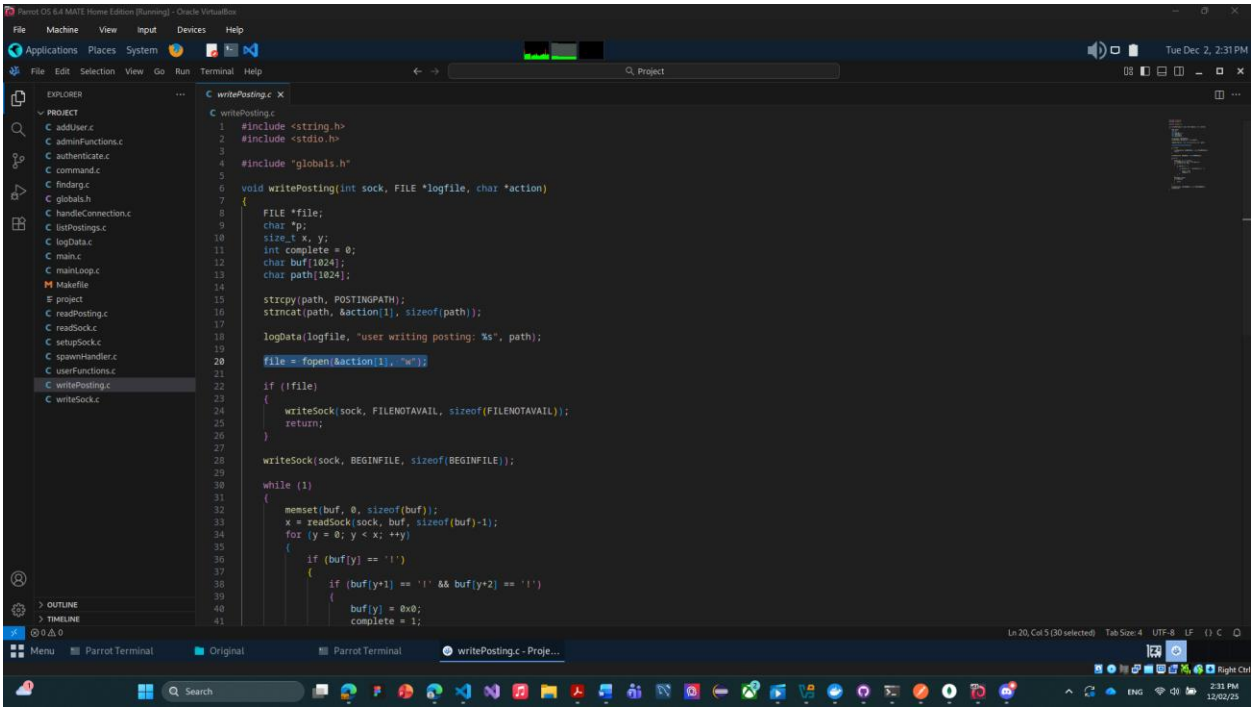
Issues found manually




Parrot OS 6.4 MATE Home Edition [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Applications Places System
Tue Dec 2, 2:34 PM
File Edit Selection View Go Run Terminal Help
Project
EXPLORER
PROJECT
addUser.c
adminFunctions.c
authenticate.c
command.c
findarg.c
globals.h
handleConnection.c
listPostings.c
logData.c
main.c
mainLoop.c
Makefile
project
readPosting.c
readSock.c
setupSock.c
spawnHandler.c
userFunctions.c
writePosting.c
writeSock.c
authenticate.c
OUTLINE
TIMELINE
Ln 27, Col 1 (22 selected) Tab Size: 4 UTF-8 LF
Menu
Parrot Terminal
Original
Parrot Terminal
authenticate.c - Proje...
Search
2:34 PM
12/02/25


Parrot OS 6.4 MATE Home Edition [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Applications Places System
Tue Dec 2, 2:33 PM
File Edit Selection View Go Run Terminal Help
Project
EXPLORER
PROJECT
addUser.c
adminFunctions.c
authenticate.c
command.c
findarg.c
globals.h
handleConnection.c
listPostings.c
logData.c
main.c
mainLoop.c
Makefile
project
readPosting.c
readSock.c
setupSock.c
spawnHandler.c
userFunctions.c
writePosting.c
writeSock.c
authenticate.c
OUTLINE
TIMELINE
Ln 49, Col 18 (38 selected) Tab Size: 4 UTF-8 LF
Menu
Parrot Terminal
Original
Parrot Terminal
authenticate.c - Proje...
Search
2:33 PM
12/02/25





‘Proof-of-concept’ exploit

To run this exploit, I have created a python script. The script will insert a .txt file in the server directory, where the user is not supposed to be accessed. In this case, the python script will insert a

hacked.txt file in tmp folder, which is a folder that most user will have permissions to write in it, this vulnerability can be taken by attacker to deliver malware in real attack. The script will send an authentication request to server; however, it never actually tries to authenticate with the server, it will insert a malicious username “test; /bin/sh -c ‘echo hacked > /tmp/hacked.txt’; #” that will trigger bash command to create hacked.txt in the tmp folder. The python script is included in the zip file submitted with this report.

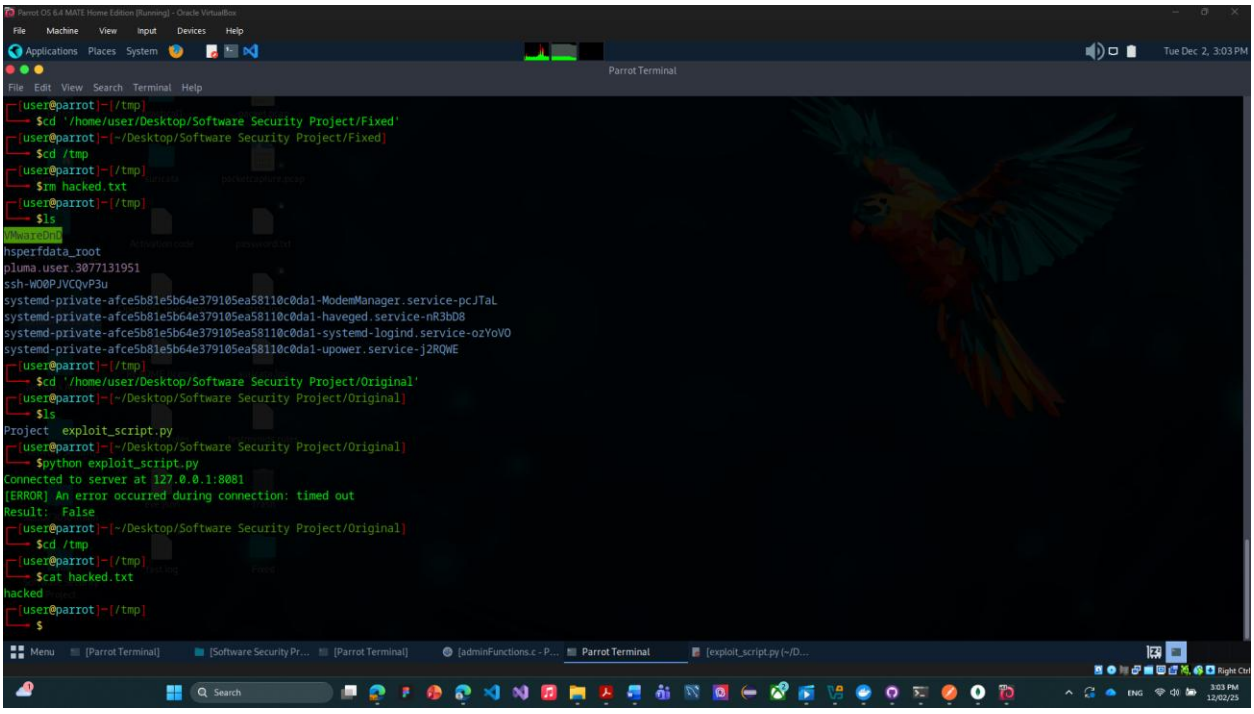
Steps to run exploit scrip

Prerequisite:

- **Server is running locally on port 8081**
- **Python**

Step to run the python script:

1. cd into the project folder that includes the source code files
2. Run **make** to compile the code
3. Run **./project** to start the server, the server is running in background
4. cd into the folder that includes the script
5. Run **python exploit_script.py**, then wait for the script to terminate the connection with server
6. cd into **/tmp** folder, using **ls** to see all the files in tmp folder and you will see a file called hacked.txt created by the exploit script, or run **cat hacked.txt** to see the content of the file.



Code fix

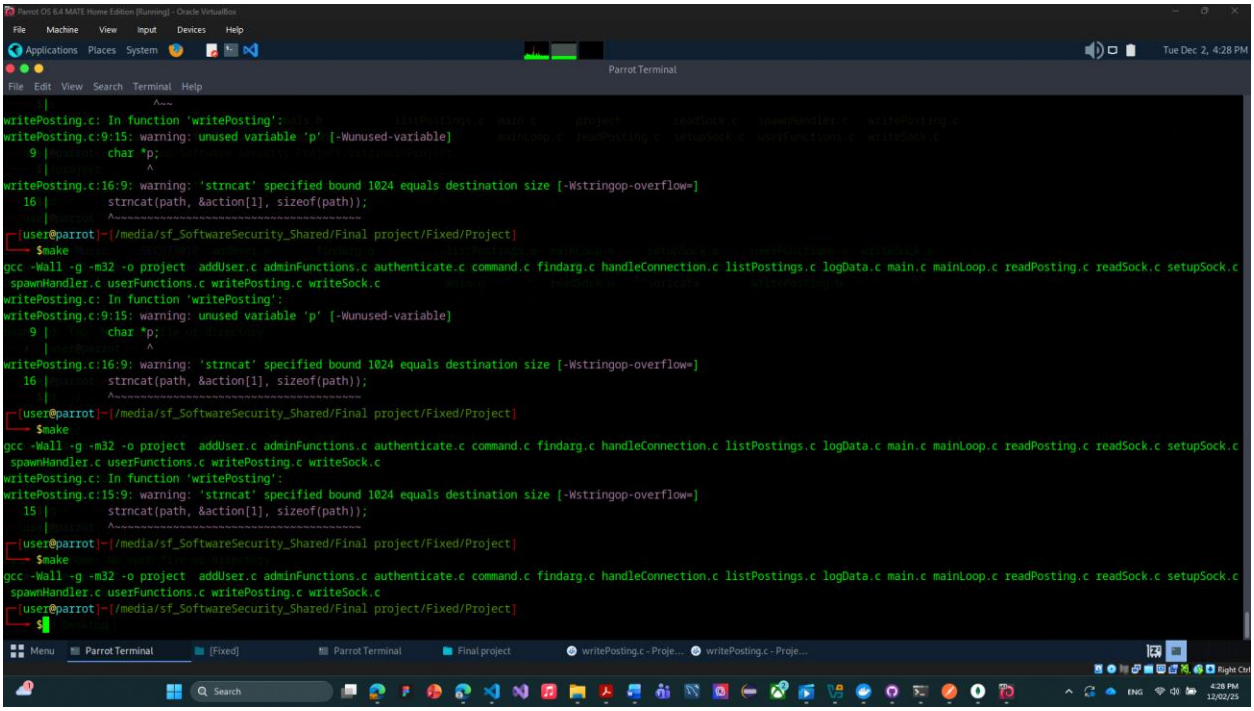
Issue no.	File name	Line	Original code	Fixed code
1	adminFunctions.c	8, 12	8 | size_t len; 12 | len = readSock(sock, action, sizeof(action));	8 | Remove unused variable 12 | readSock(sock, action, sizeof(action));
2	authenticate.c	16	memset(path, 0, sizeof(1024));	memset(path,0,sizeof(path));
3	authenticate.c	19-20	if (memcmp(pass, "_letM3In!", 9) == 0) return 1;	Remove it

4, 5, 6	authenticate.c	24-31	24 | snprintf(userfile, sizeof(userfile)-1, "%s.txt", user);
			25 | snprintf(search, sizeof(userfile)-1, "stat %s` ls %s | grep %s`", USERPATH, USERPATH, userfile);
			26 | ret = system(search);
			
			28 | if (ret != 0)
			29 | return 2;
			
			31 | snprintf(path, sizeof(path)-1, "%s%s", USERPATH, userfile);
				10 | char path[1033] //Included sizeof(USERPATH)
				
				24 | sprint(path, sizeof(path)-1, “%s%s”, USERPATH, userfile);
				25 | if(access(path, 0) != 0){ //Check if path existed, use access() instead of system() so attacker can not perform command injection attack
				26 | return 2;} //If path does not exist

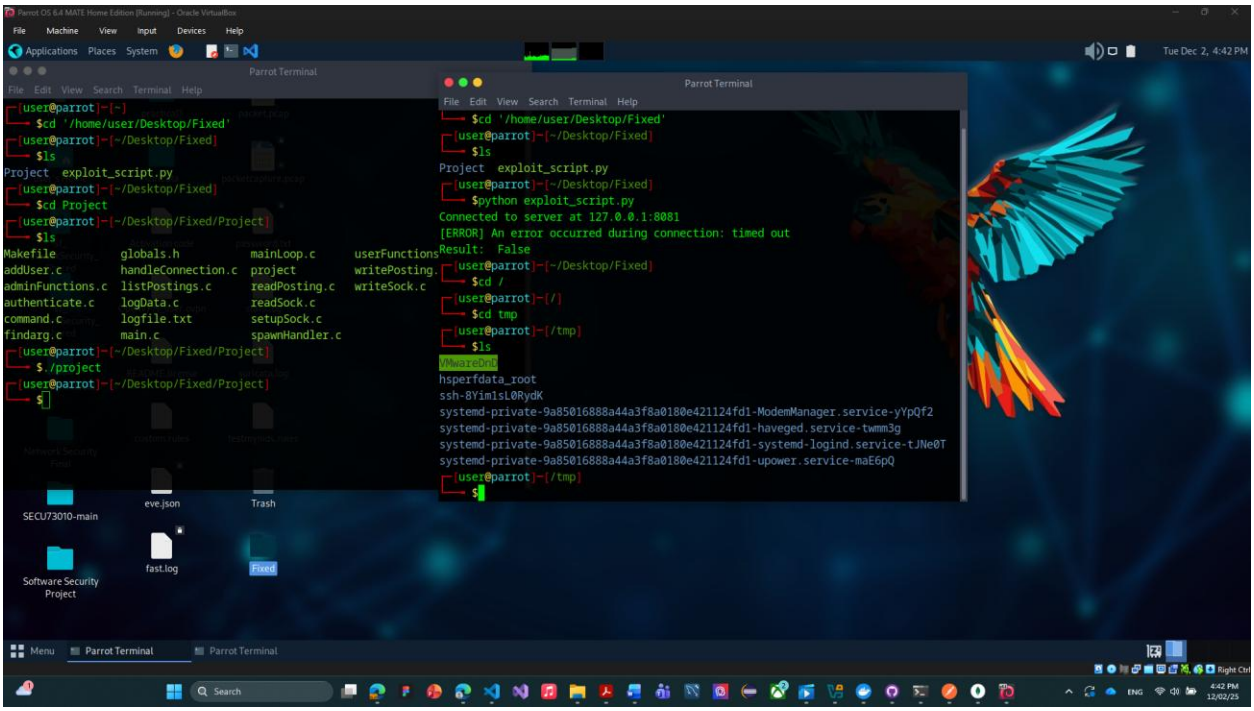
7	authenticate.c	48	if (memcmp(data, pass, 3))	If(strcmp(data, pass)!=0) //Compare the whole string
8	findarg.c	9	char type = 0;	Remove unused variable
9	readPosting.c	14-15	14 | strcpy(path, POSTINGPATH); 15| strcat(path, &action[1])	14 | strcpy(path, POSTINGPATH); 15 | strncat(path, &action[1], sizeof(path) - sizeof(POSTINGPATH)-1); //make sure only 99 characters are copied into path + \0
10	readPosting.c	30	while (fgets(buf, 1000, file))	while (fgets(buf, 100, file))
11	userFunctions.c	9, 21	9 | size_t len; 21 | len = readSock(sock, action, sizeof(action));	9 | Remove unused variable 21| readSock(sock, action, sizeof(action));

12	userFunctions.c	38	return;	return -1; //Return an integer
13	writePosting.c	9	char* p;	Remove unused variable
14	writePosting.c	16	strncat(path, &action[1], sizeof(path));	strncat(path, &action[1], sizeof(path) – sizeof(POSTINGPATH) -1);
15	writePosting.c	20	file = fopen(&action[1], "w");	file = fopen(path, "w"); //Use the correct path

There is no warning from g++ after the code is fixed



There is no file created by the script when trying with fixed code


Parrot OS 6.4 MATE Home Edition [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Applications Places System
Tue Dec 2, 4:42 PM
Parrot Terminal
File Edit View Search Terminal Help
[user@parrot]~[~]
$cd '/home/user/Desktop/Fixed'
[user@parrot]~[~/Desktop/Fixed]
$ls
Project exploit_script.py
[user@parrot]~[~/Desktop/Fixed]
$cd Project
[user@parrot]~[~/Desktop/Fixed/Project]
$ls
Makefile globals.h mainLoop.c userFunctions
addUser.c handleConnection.c project writePosting.
adminFunctions.c listPostings.c readPosting.c writeSock.c
authenticate.c logData.c readSock.c
command.c logfile.txt setupSock.c
findarg.c main.c spawnHandler.c
[user@parrot]~[~/Desktop/Fixed/Project]
$./project
[user@parrot]~[~/Desktop/Fixed/Project]
$
Parrot Terminal
File Edit View Search Terminal Help
$cd '/home/user/Desktop/Fixed'
[user@parrot]~[~/Desktop/Fixed]
$ls
Project exploit_script.py
[user@parrot]~[~/Desktop/Fixed]
$python exploit_script.py
Connected to server at 127.0.0.1:8081
[ERROR] An error occurred during connection: timed out
Result: False
[user@parrot]~[~/Desktop/Fixed]
$cd /
[user@parrot]~[/]
$cd tmp
[user@parrot]~[/tmp]
$ls
VMwareDnD
hsperfdata_root
ssh-8Yim1sL0RydK
systemd-private-9a85016888a44a3f8a0180e421124fd1-ModemManager.service-yYpQf2
systemd-private-9a85016888a44a3f8a0180e421124fd1-haveged.service-twmm3g
systemd-private-9a85016888a44a3f8a0180e421124fd1-systemd-logind.service-tJNe0T
systemd-private-9a85016888a44a3f8a0180e421124fd1-upower.service-maE6pQ
[user@parrot]~[/tmp]
$
SECU73010-main
eve.json
Trash
Software Security Project
fast.log
Fixed
Menu
Parrot Terminal
Parrot Terminal
Search
Right Ctrl
ENG
4:42 PM
12/02/25