



VULNERABILITIES ASSESSMENT REPORT

Tyler Dao

SECU73000 – INTRODUCTION TO SOFTWARE SECURITY – SECTION 1



Contents

Code Review	2
Issues found by g++ compiler	3
Issues found manually	4
‘Proof-of-concept’ exploit	6
Steps to run exploit scrip	7
Code fix	8

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

```

Parrot OS 4.4 MATE Home Edition (running) - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
File Edit View Search Terminal Help
spawnHandler.c userFunctions.c writePosting.c writeSock.c
adminFunctions.c: In function 'adminFunctions':
adminFunctions.c:8:16: warning: variable 'len' set but not used [-Wunused-but-set-variable]
8 |     size_t len;
  |     ~~~~~^
authenticate.c: In function 'authenticate':
authenticate.c:25:68: warning: '%s' directive output may be truncated writing up to 1023 bytes into a region of size 990 [-Wformat-truncation=]
25 |     snprintf(search, sizeof(userfile)-1, "stat %s ls %s | grep %s", USERPATH, USERPATH, userfile);
  |     ~~~~~^
authenticate.c:25:9: note: 'snprintf' output between 35 and 1058 bytes into a destination of size 1023
25 |     snprintf(search, sizeof(userfile)-1, "stat %s ls %s | grep %s", USERPATH, USERPATH, userfile);
  |     ~~~~~^
authenticate.c:31:43: warning: '%s' directive output may be truncated writing up to 1023 bytes into a region of size 1015 [-Wformat-truncation=]
31 |     snprintf(path, sizeof(path)-1, "%s%s", USERPATH, userfile);
  |     ~~~~~^
authenticate.c:31:9: note: 'snprintf' output between 9 and 1032 bytes into a destination of size 1023
31 |     snprintf(path, sizeof(path)-1, "%s%s", USERPATH, userfile);
  |     ~~~~~^
findarg.c: In function 'findarg':
findarg.c:9:14: warning: unused variable 'type' [-Wunused-variable]
9 |     char type = 0;
  |     ~~~~~^
readPosting.c: In function 'readPosting':
readPosting.c:30:16: warning: 'fgets' writing 1000 bytes into a region of size 100 overflows the destination [-Wstringop-overflow=]
30 |     while (fgets(buf, 1000, file))
  |     ~~~~~^
readPosting.c:9:14: note: destination object 'buf' of size 100
9 |     char buf[100];
  |     ~~~~~^
In file included from readPosting.c:2:
/usr/include/stdio.h:592:14: note: in a call to function 'fgets' declared with attribute 'access (write_only, 1, 2)'
592 | extern char *fgets (char *__restrict __s, int __n, FILE *__restrict __stream)
  |     ~~~~~^

```

```
Parrot OS 6.4 MATE Home Edition (Running) - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
File Edit View Search Terminal Help

In file included from readPosting.c:2:
/usr/include/stdio.h:592:14: note: in a call to function 'fgets' declared with attribute 'access(write_only, 1, 2)'
592 | extern char *fgets(char *__restrict __s, int __n, FILE *__restrict __stream)
      |
readPosting.c:30:16: warning: 'fgets' writing 1000 bytes into a region of size 100 overflows the destination [-Wstringop-overflow=]
30 |     while (fgets(buf, 1000, file))
      |
readPosting.c:9:14: note: destination object 'buf' of size 100
9 |     char buf[100];
      |
/usr/include/stdio.h:592:14: note: in a call to function 'fgets' declared with attribute 'access(write_only, 1, 2)'
592 | extern char *fgets(char *__restrict __s, int __n, FILE *__restrict __stream)
      |
userFunctions.c: In function 'userFunctions':
userFunctions.c:38:25: warning: 'return' with no value, in function returning non-void [-Wreturn-type]
38 |     return;
      |
userFunctions.c:6:5: note: declared here
6 | int userFunctions(FILE *logfile, int sock, char *user)
      |
userFunctions.c:9:16: warning: variable 'len' set but not used [-Wunused-but-set-variable]
9 |     size_t len;
      |
writePosting.c: In function 'writePosting':
writePosting.c:9:15: warning: unused variable 'p' [-Wunused-variable]
9 |     char *p;
      |
writePosting.c:16:9: warning: 'strncat' specified bound 1024 equals destination size [-Wstringop-overflow=]
16 |     strncat(path, &action[1], sizeof(path));
      |
user@parrot:~/Desktop/Software Security Project/Original/Project
```

Issues found manually

```
Parrot OS 6.4 MATE Home Edition (Running) - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
File Edit Selection View Go Run Terminal Help

EXPLORER
PROJECT
C addUsers.c
C adminFunctions.c
C authenticate.c
C command.c
C findarg.c
C global.h
C handleConnection.c
C listPostings.c
C logData.c
C logfile.txt
C main.c
C mainLoop.c
C Makefile
C project
C readPosting.c
C readSock.c
C setupSock.c
C spawnHandler.c
C userFunctions.c
C writePosting.c
C writeSock.c

C authenticate.c
1 #include <string.h>
2 #include <stdlib.h>
3
4 #include "globals.h"
5
6 /* return 1 for success, 2 on bad username, 3 on bad password */
7 int authenticate(FILE *logfile, char *user, char *pass)
8 {
9     char search[512];
10    char path[1024];
11    char userfile[1024];
12    char data[1024];
13    FILE *file;
14    int ret;
15
16    memset(path, 0, sizeof(1024));
17
18    /* FIXME: hard coded admin backdoor for password recovery */
19    if (memcmp(pass, "_letMeIn!", 9) == 0)
20        return 1;
21
22    /* look up user by checking user files: done via system() to /bin/ls|grep user */
23    logData(logfile, "performing lookup for user via system()\n");
24    sprintf(userfile, sizeof(userfile)-1, "%s.txt", user);
25    sprintf(search, sizeof(userfile)-1, "stat %s | grep %s", USERPATH, userfile);
26    ret = system(search);
27
28    if (ret != 0)
29        return 2;
30
31    sprintf(path, sizeof(path)-1, "%s%s", USERPATH, userfile);
32
33    /* open file and check if contents == password */
34    file = fopen(path, "r");
35
36    if (!file)
37    {
38        logData(logfile, "fopen for userfile failed\n");
39        return 2;
40    }
41}
```

```
18 /* FIXME: hard coded admin backdoor for password recovery */
19 if (memcmp(pass, "_letM3in!", 9) == 0)
20     return 1;
21
22 /* look up user by checking user files: done via system() to /bin/ls|grep user */
23 logData(logfile, "performing lookup for user via system()\n");
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);
27
28 if (ret != 0)
29     return 2;
30
31 snprintf(path, sizeof(path)-1, "%s%s", USERPATH, userfile);
32
33 /* open file and check if contents == password */
34 file = fopen(path, "r");
35
36 if (!file)
37 {
38     logData(logfile, "fopen for userfile failed\n");
39     return 2;
40 }
41
42 logData(logfile, "getting userfile info\n");
43 fgets(data, sizeof(data)-1, file);
44
45 fclose(file);
46
47 /* Password Check! */
48 if (memcmp(data, pass, 3))
49     return 3;
50
51 return 1;
52 }
53
54
```

```
47 /* Password Check! */
48 if (memcmp(data, pass, 3))
49     return 3;
50
51 return 1;
52 }
53
54
```

‘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.

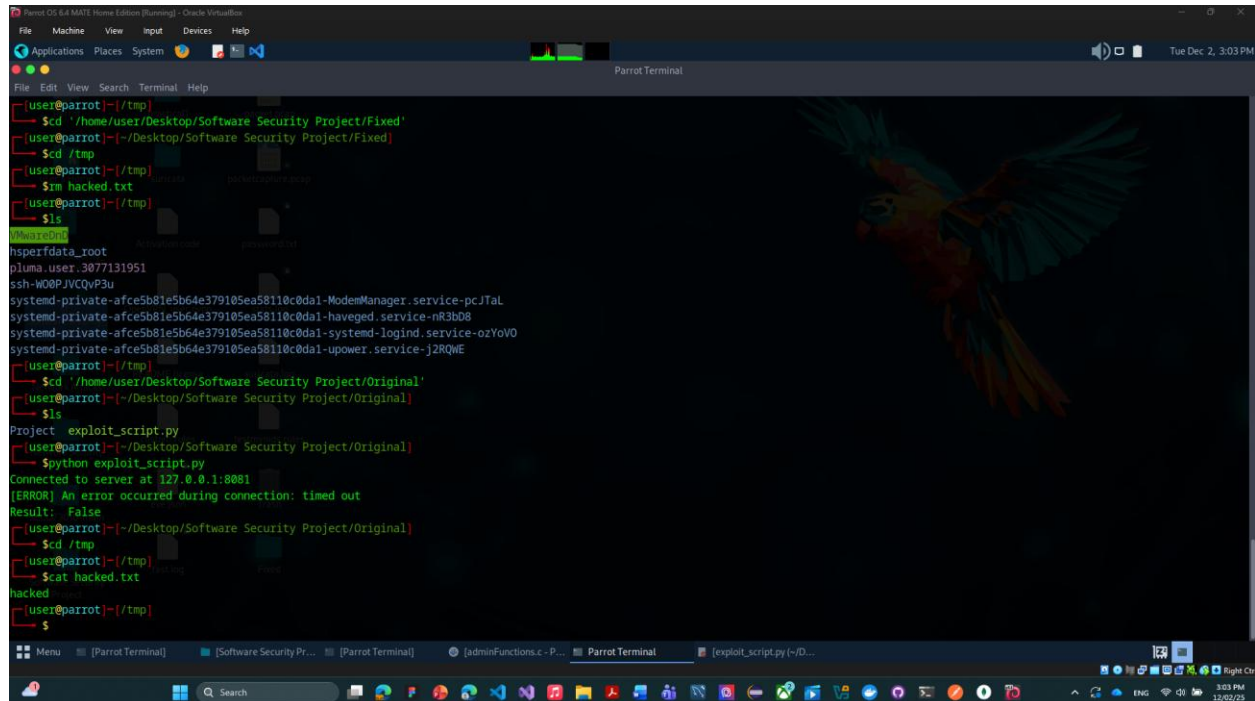
Steps to run exploit scrip

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.



```
user@parrot: ~/tmp
$ cd '/home/user/Desktop/Software Security Project/Fixed'
user@parrot: ~/Desktop/Software Security Project/Fixed
$ cd /tmp
user@parrot: ~/tmp
$ make
$ python exploit_script.py
$ ls
hacked.txt
user@parrot: ~/tmp
$ cd '/home/user/Desktop/Software Security Project/Original'
user@parrot: ~/Desktop/Software Security Project/Original
$ ls
Project  exploit_script.py
user@parrot: ~/Desktop/Software Security Project/Original
$ 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/Software Security Project/Original
$ cd /tmp
user@parrot: ~/tmp
$ cat hacked.txt
hacked
user@parrot: ~/tmp
$
```


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, "_letM3ln!", 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 sprintf(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

```

gcc -Wall -g -m32 -o project addUser.c adminFunctions.c authenticate.c command.c findarg.c handleConnection.c listPostings.c logData.c main.c mainLoop.c readPosting.c readSock.c setupSock.c
spawnHandler.c userFunctions.c writePosting.c writeSock.c
writePosting.c: In function 'writePosting':
writePosting.c:9:15: warning: unused variable 'p' [-Wunused-variable]
     char *p;
     ^
writePosting.c:16:9: warning: 'strncat' specified bound 1024 equals destination size [-Wstringop-overflow=]
     strncat(path, &action[1], sizeof(path));
     ^~~~~~
[user@parrot:~/media/sf_SoftwareSecurity_Shared/Final project/Fixed/Project]
$Snake
gcc -Wall -g -m32 -o project addUser.c adminFunctions.c authenticate.c command.c findarg.c handleConnection.c listPostings.c logData.c main.c mainLoop.c readPosting.c readSock.c setupSock.c
spawnHandler.c userFunctions.c writePosting.c writeSock.c
writePosting.c: In function 'writePosting':
writePosting.c:9:15: warning: unused variable 'p' [-Wunused-variable]
     char *p;
     ^
writePosting.c:16:9: warning: 'strncat' specified bound 1024 equals destination size [-Wstringop-overflow=]
     strncat(path, &action[1], sizeof(path));
     ^~~~~~
[user@parrot:~/media/sf_SoftwareSecurity_Shared/Final project/Fixed/Project]
$Snake
gcc -Wall -g -m32 -o project addUser.c adminFunctions.c authenticate.c command.c findarg.c handleConnection.c listPostings.c logData.c main.c mainLoop.c readPosting.c readSock.c setupSock.c
spawnHandler.c userFunctions.c writePosting.c writeSock.c
[user@parrot:~/media/sf_SoftwareSecurity_Shared/Final project/Fixed/Project]

```

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

