PROJECT REPORT

CS Project 2- April 01, 2024

Group: Techies

Vooha sree Chitta (A05306488)

Charishma Maganti (A05317920)

D. Dhanush Narasimha Reddy (A05307038)

Section I:

Introduction:

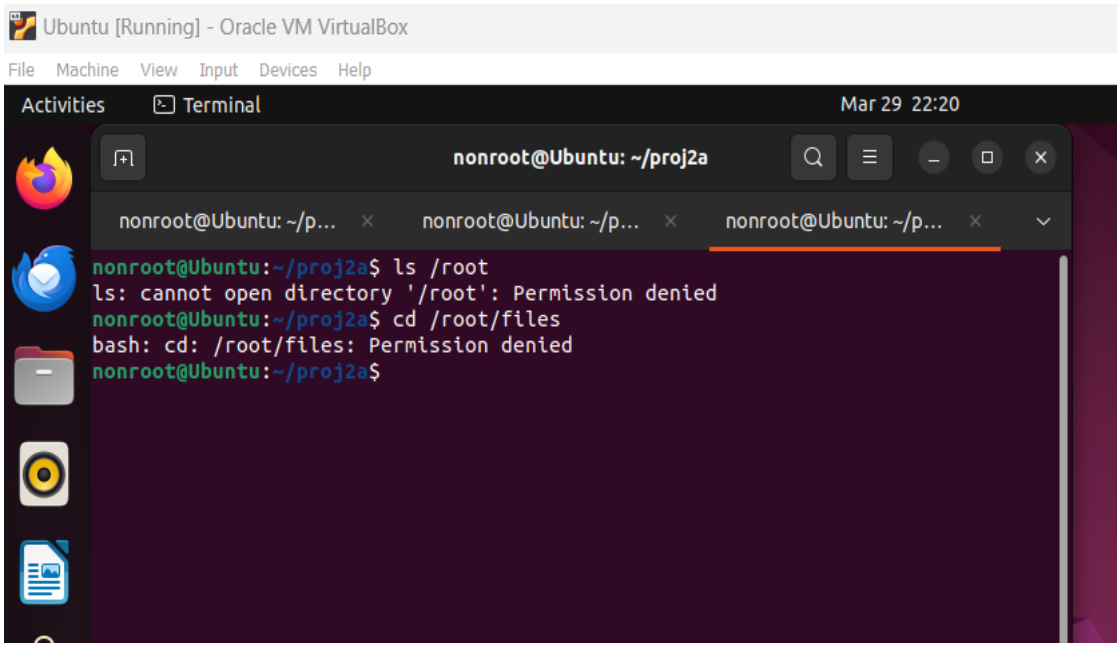
The primary objective of this project is to create a sandbox network using virtualization technologies and implement security policies to enforce a secure environment. Through this project, we aim to:

1) Creating nonroot users in A.1 and B.1 and try to access /root/files location using SSH login.
2) Building programs tcps,tcph and tcpc after modifying ip address and paths in those files using given make file.
3) Analyzing the echo program using gdb in A.1 and B.1.
4) Usage of DVMA with respect to attack one from another.
5) Capturing attacked packets using wireshark.
6) Cracking the password of klepetko.net for user50 using c program.
7) Cracking the password of klepetko.net for user50 using msfconsole.
8) Cracking the password of klepetko.net using usernames and passwords from different txt files in msfconsole.
9) Retrieving files secret.pdf.enc1 and secret.pdf.enc2 from klepetko.net. Retrieving files secret.pdf.enc1 and secret.pdf.enc2 from klepetko.net.
10) Developing a decryption program to decrypt secret.pdf.enc1 where it is encrypted by performing XOR on a block of 8 bytes with a key.
11) Creating a test file to store “abcdefgh” and choosing key as “0x000000000000001C” to encrypt the test file and decrypt it using dec2.c.
12) Creating a bruteforce attack program to crack the key of test file and use the same program to retrieve key for secret.pdf.enc2

	Vooha	**Charishma**	**Dhanush**
Task 1	Yes	Partial	Partial
Task 2	Partial	Yes	Partial
Task 3	Partial	Partial	Yes
Task 4	Yes	Partial	Partial
Task 5	Yes	Partial	Partial
Task 6	Partial	Partial	Yes
Task 7	Partial	Yes	Partial

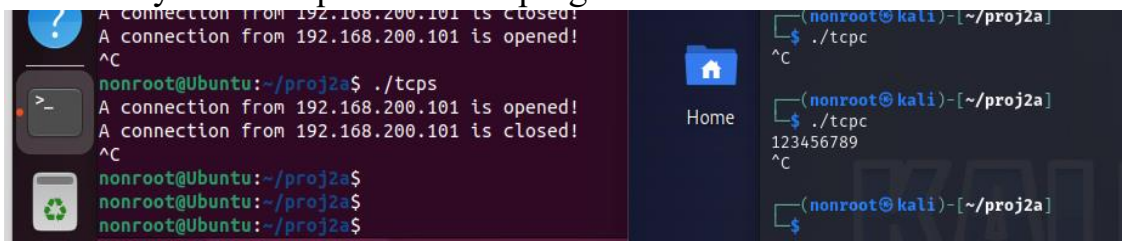
Section II: (Task I):

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

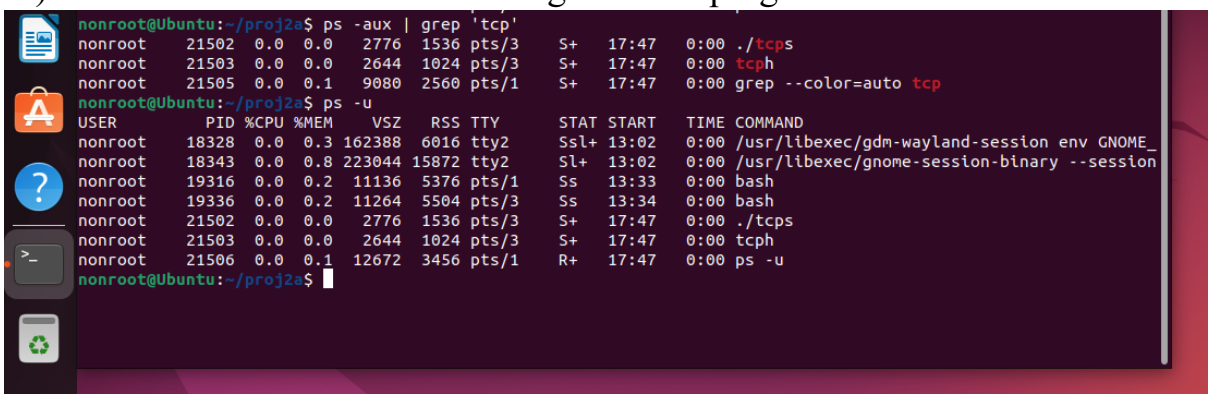


ii) Find and report exactly how many bytes are needed to crash the echo program.

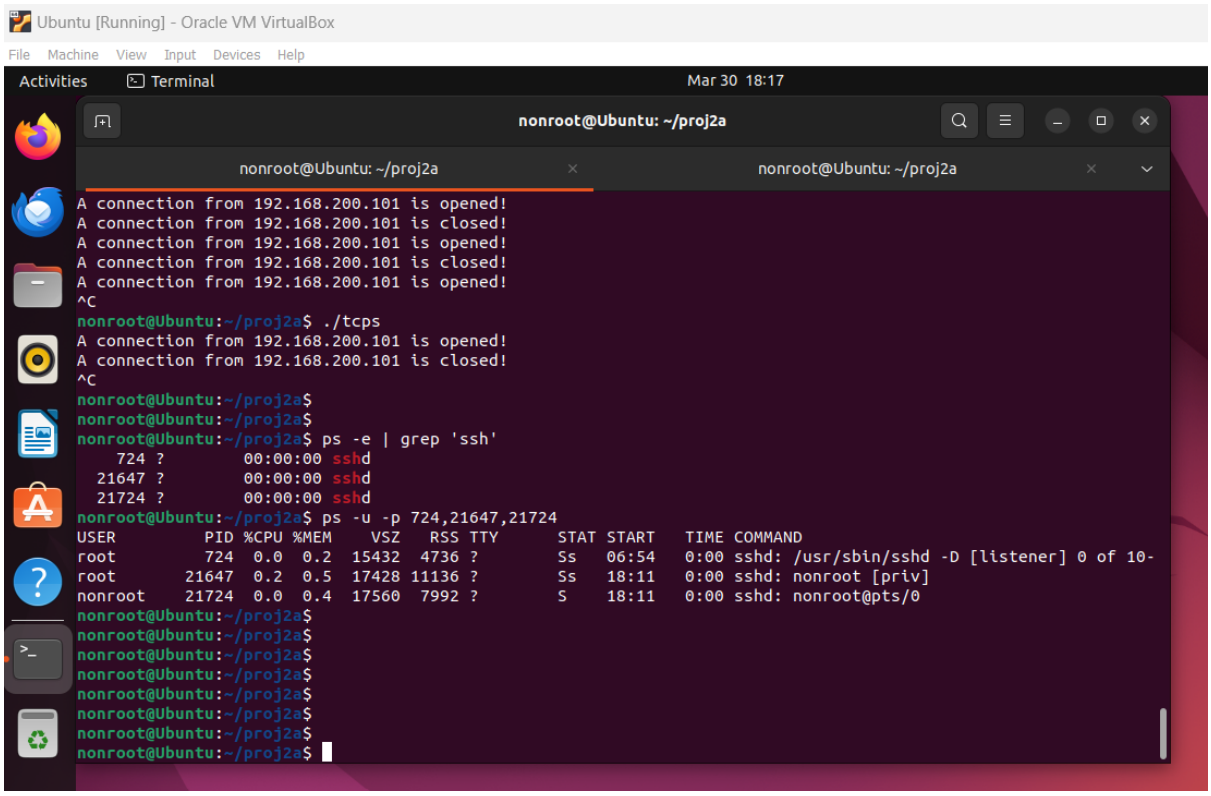
9 Bytes are required for the program to crash



iii) Show which user ID is running the echo program in A.1.

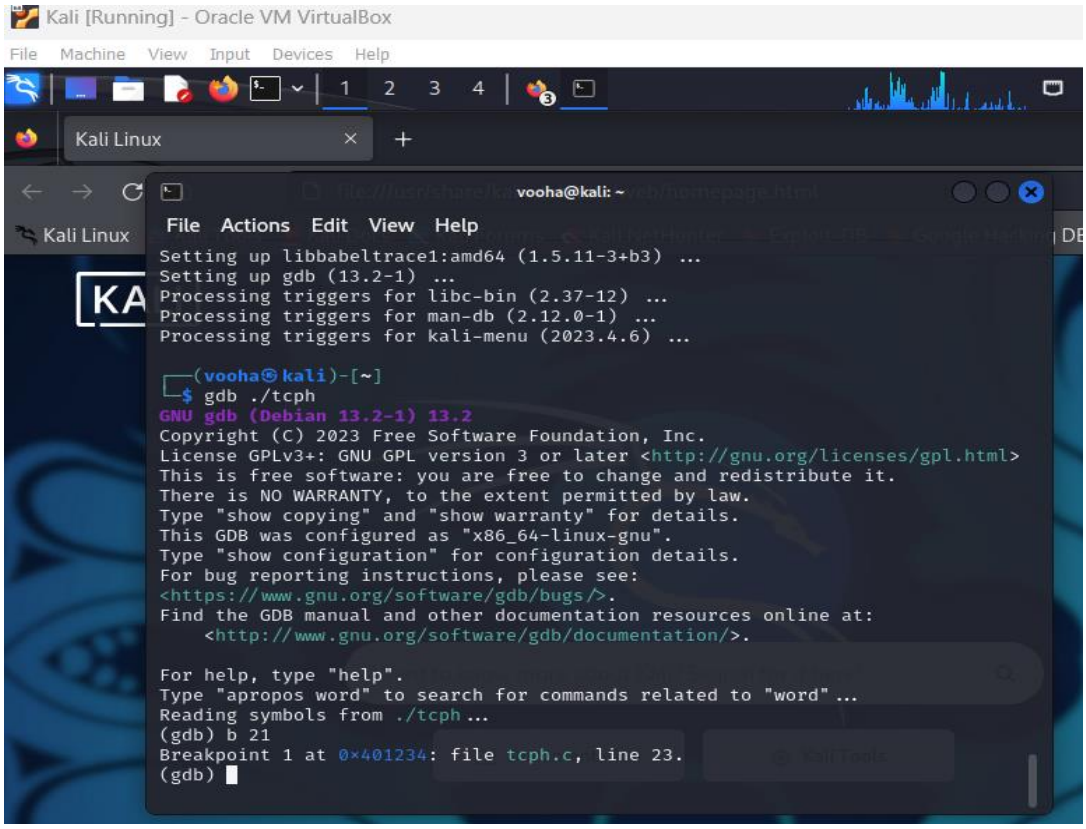


iv) Show which user ID is running the SSH service in A.1.

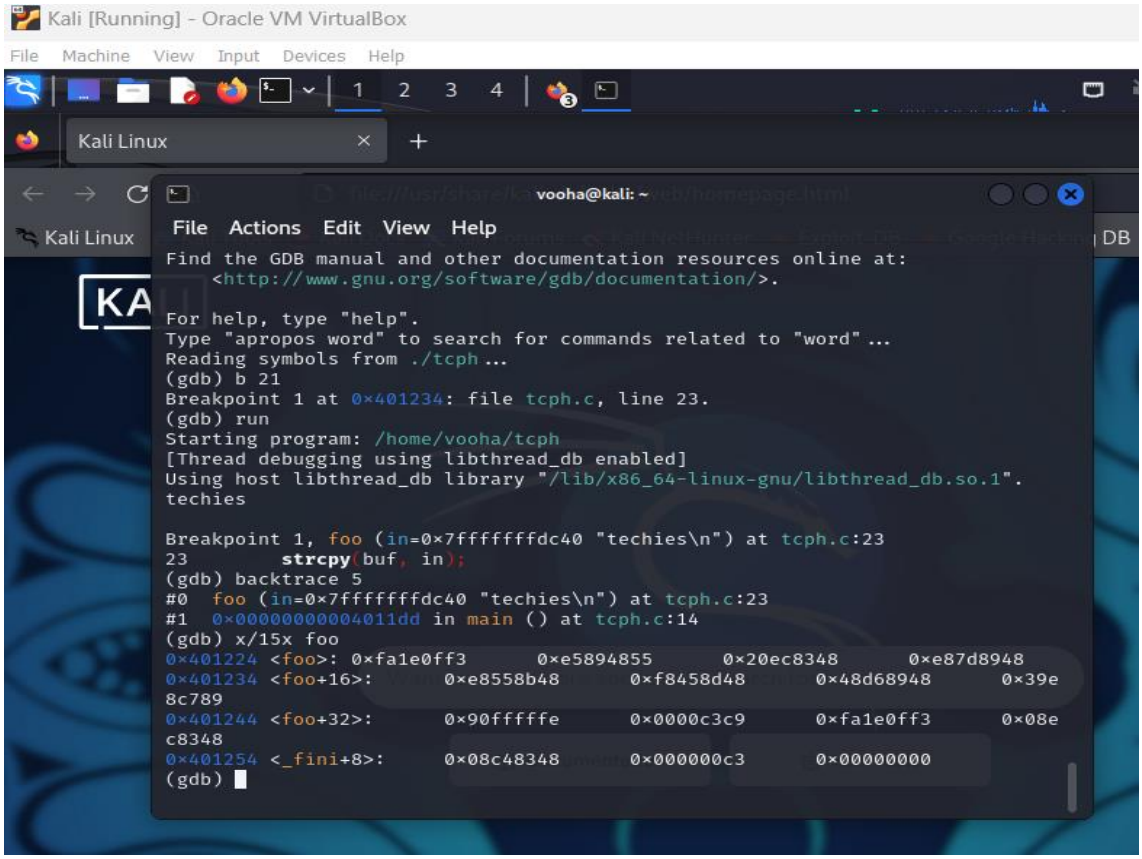


Section III: (Task II):

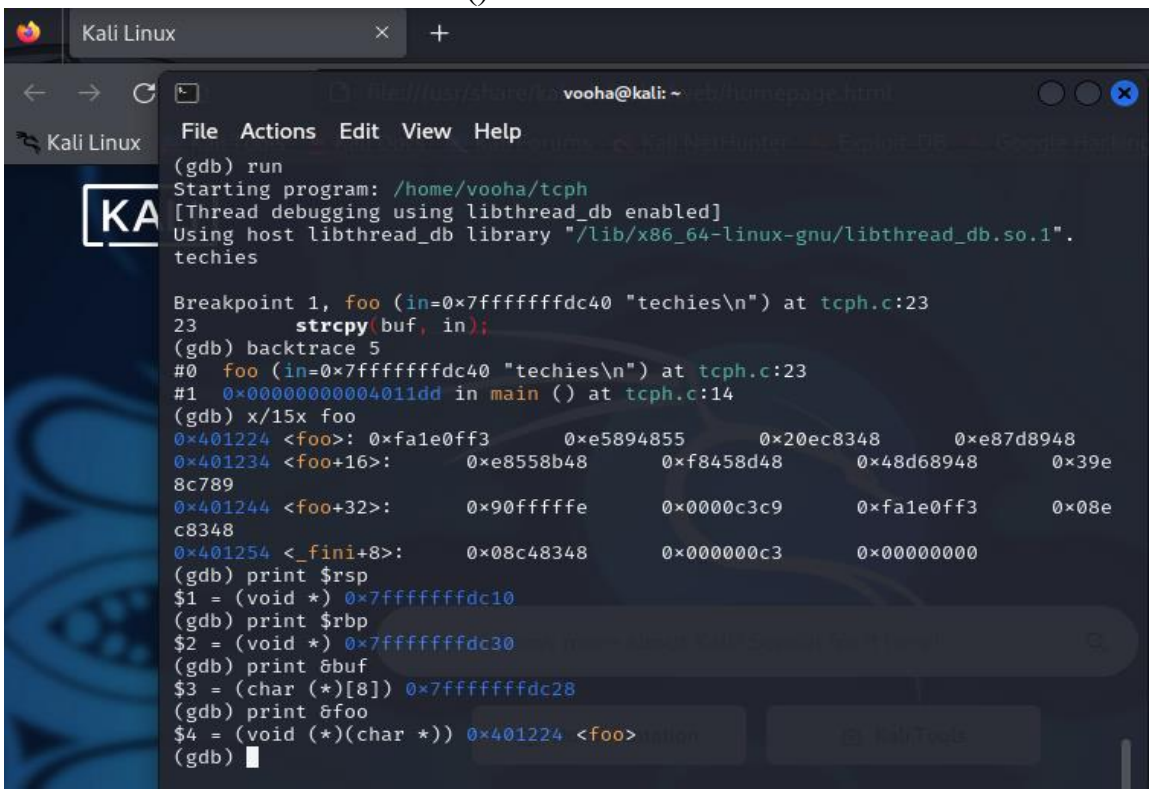
i) Show a screenshot of gdb running to a breakpoint in foo() of tcph in B.1.



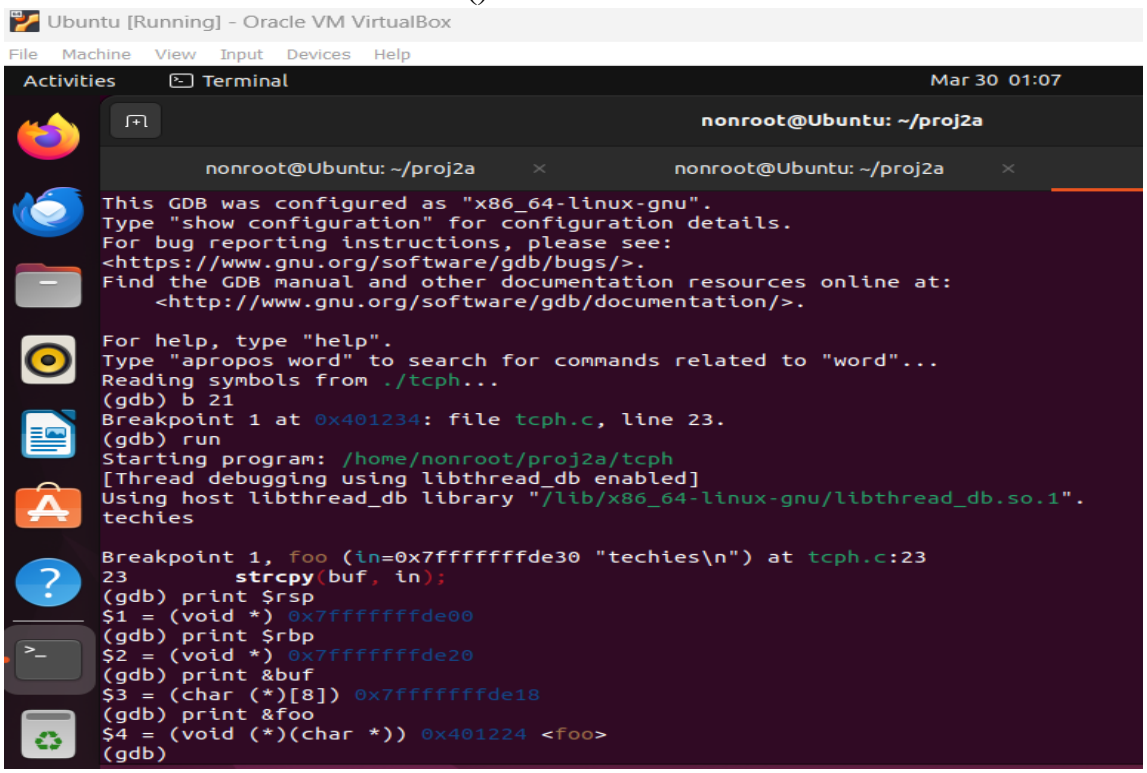
ii) Show a screenshot of gdb showing the stack of foo() of tcph in B.1.



iii) Report the values of $rsp, $rbp, the address of buf, and the address of the return address of foo() in B.1.



iv) Report the values of $rsp, $rbp, the address of buf, and the address of the return address of foo() in A.1.



Section IV: (Task III):

i) Show a screenshot that the echo program is exploited.

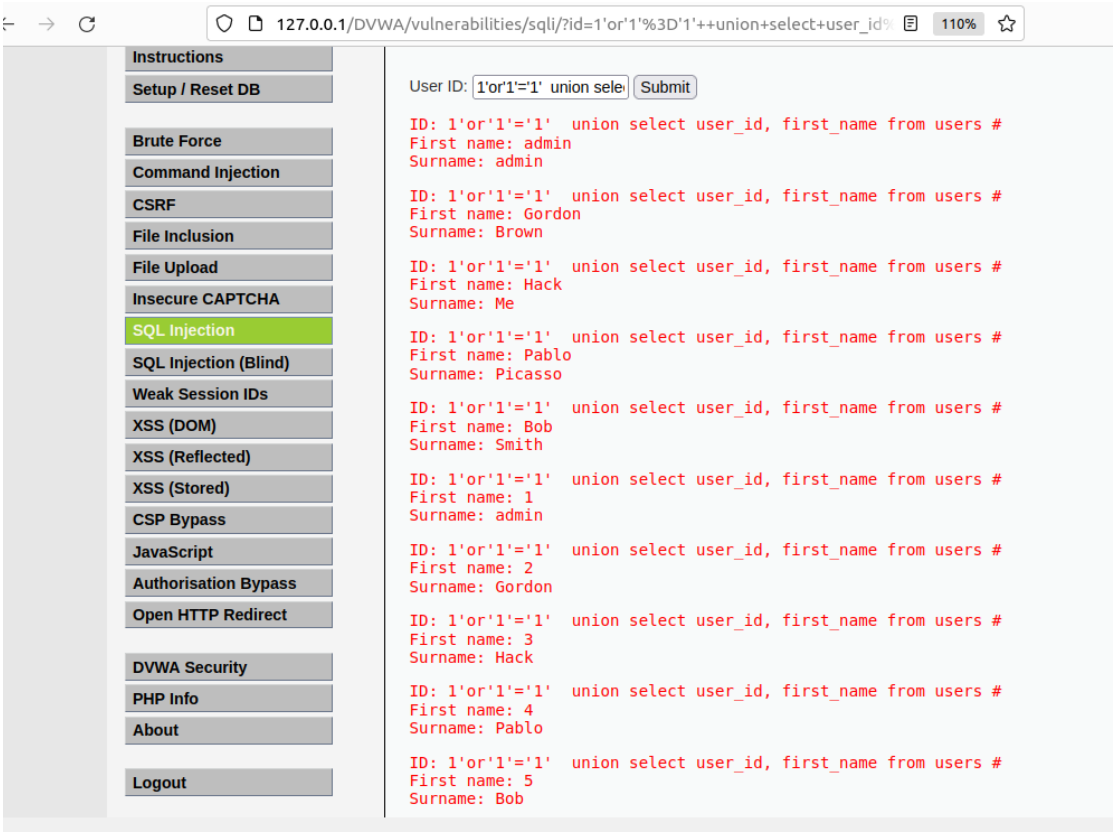
ii) Show the exploiting packet captured in B.1.

iii) Report how you retrieve the files from A.1 to B.1. Give steps in detail.
By doing scp to the files scp /root/files/*
vooha@192.168.200.101:/home/vooha/

iv) Show the content of the smallest file in the retrieved files.

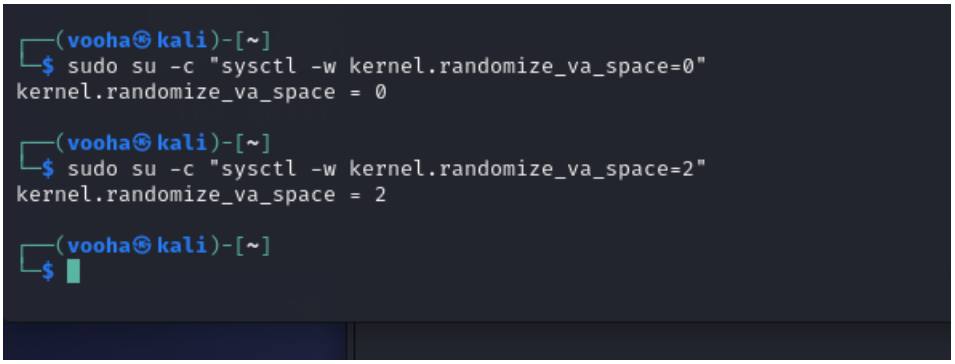
v) Show the injected SQL statement.
Select first_name,last_name from users where user_id = 1 or '1' = '1'
UNION select user_id, first_name from users #

vi) Show the screenshot of the web page that show all user` IDs, first names, and last names.



Section V:

i) One defense mechanism is to randomize the address space of stack memory (so called randomization). The shell scripts, enablerandom.sh and disablerandom.sh, are provided to show how to enable or disable the defense mechanism.



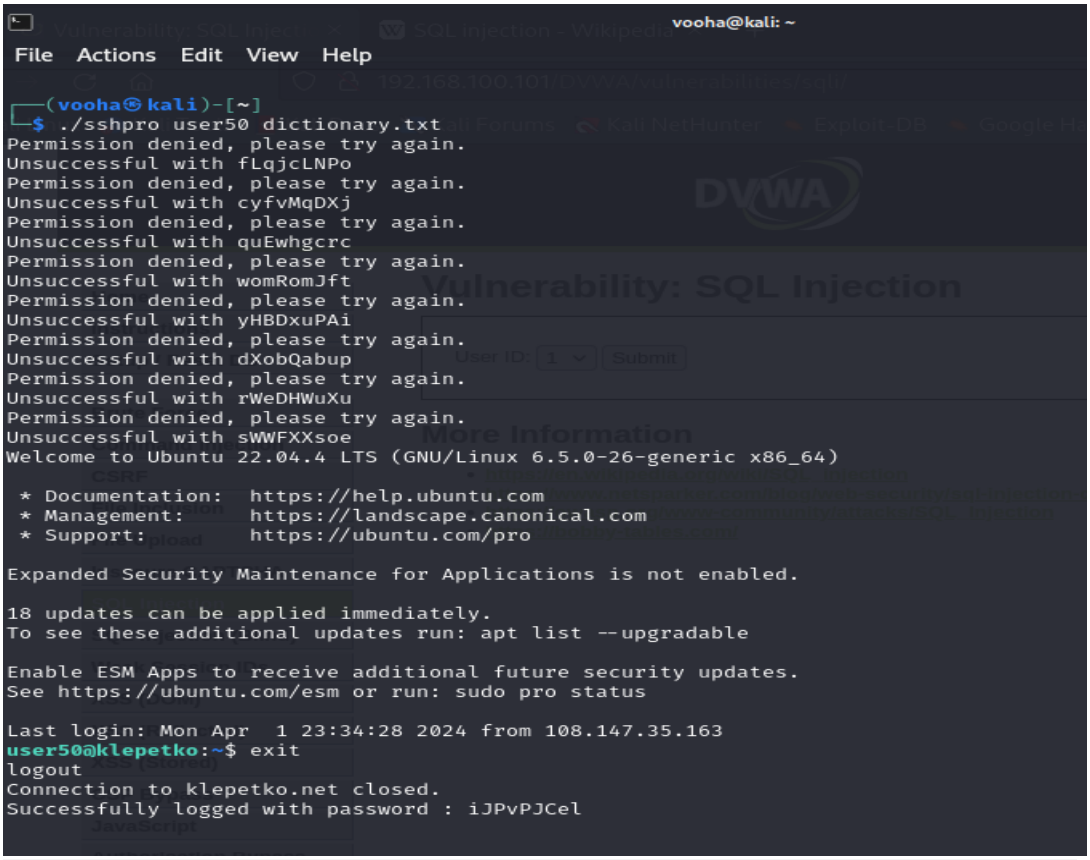
ii) Discuss the reason that randomization can defeat the attack.
Randomization of the address space of stack memory is an effective defense mechanism because it makes it much harder for attackers to predict the memory layout of a process. When an attacker attempts to exploit a vulnerability, such as a buffer overflow, they often rely on

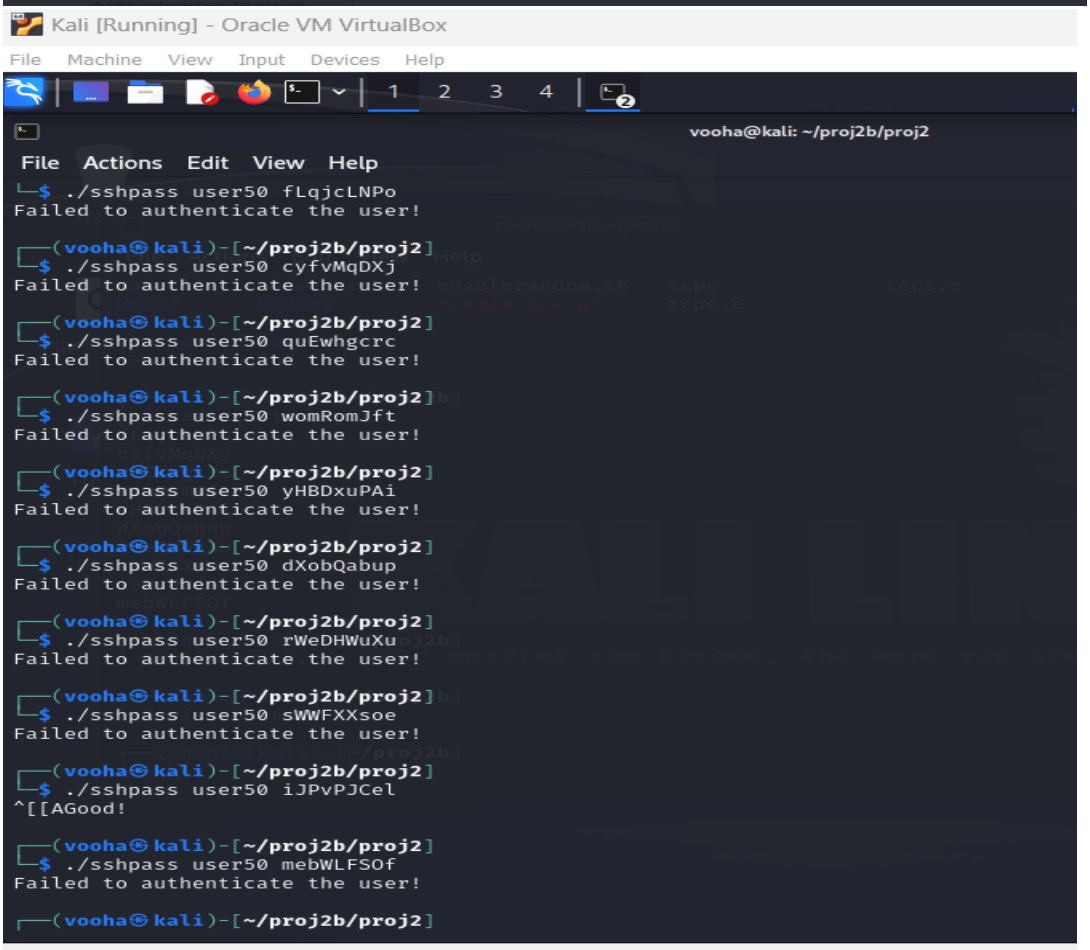
knowing the exact memory addresses of certain data or code within the process's memory space. By randomizing the addresses, the attacker cannot reliably determine where specific data or code resides, making it significantly more difficult to exploit successfully. It works by introducing entropy into the memory layout, making it unpredictable. This means that even if an attacker successfully finds a vulnerability and gains control of the program's execution flow, they are less likely to be able to locate and manipulate critical data or code necessary to compromise the system.

iii) Assume only the low 16 bits of the stack address is randomized. What is the probability that an exploiting packet can compromise the server? Assume an attacker can send 10 exploiting packets every second. How long will it take for the attacker to compromise the server?
If only low 16 bits are randomised then 2^{16} addresses are possible.
So, to guess that address for him the probability would be $1/2^{16} = 1/65536$.
Total time taken to compromise on the server would be $1/(1/2^{16}))$ for each packet.
If attacker sends 10 exploiting packets per second then it requires for him to compromise on each addresses would be 65536*10 = 182 hours

Section VI: (Task IV):

i) Show the screen shot of your program in B.1 when you are testing each password and obtaining the password to ssh klepetko.net as “user50”.





ii) Report how long it takes to test each password on average.

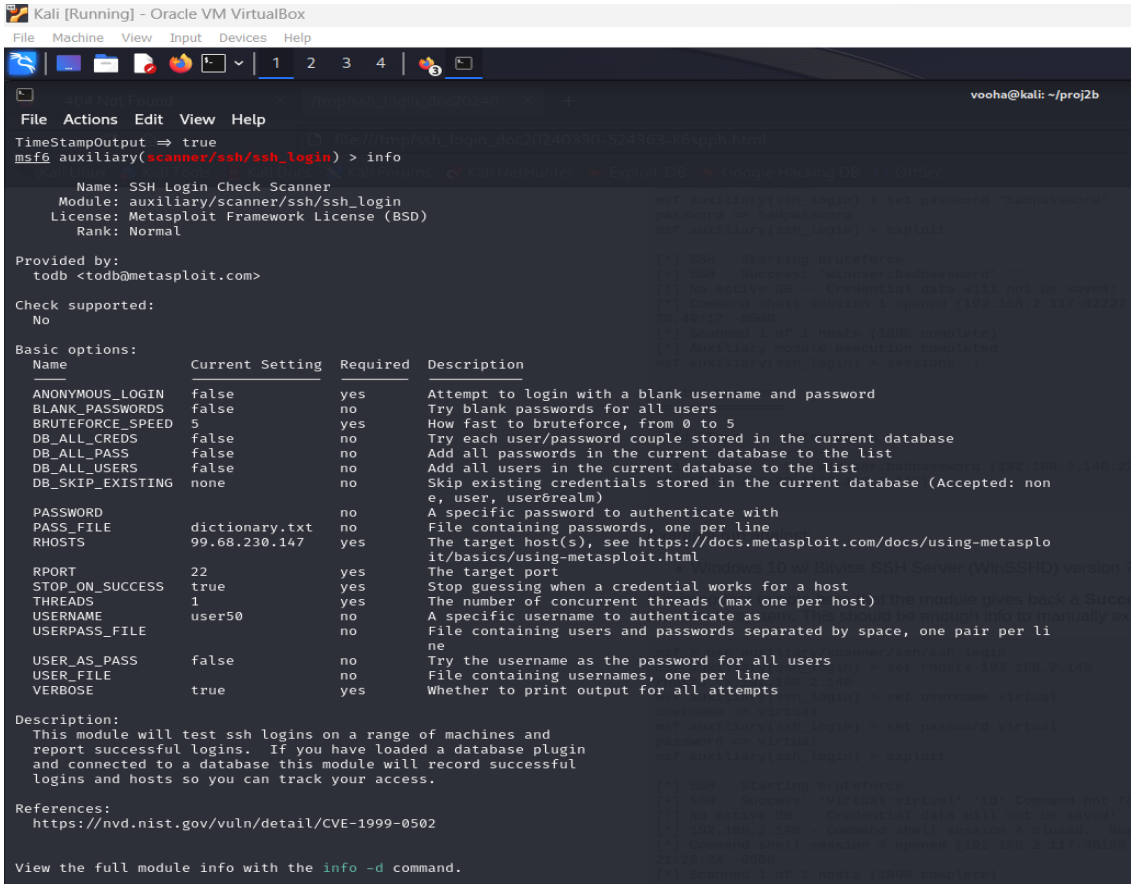
It took around 35 seconds to test 9 passwords. On an average it took 3.88 seconds.

iii) If the dictionary has 1 million passwords, estimate how long it will take to find the password with your program.
So to test million passwords = 1000000 * 3.88 = 3880000 seconds = 1077.77 hrs

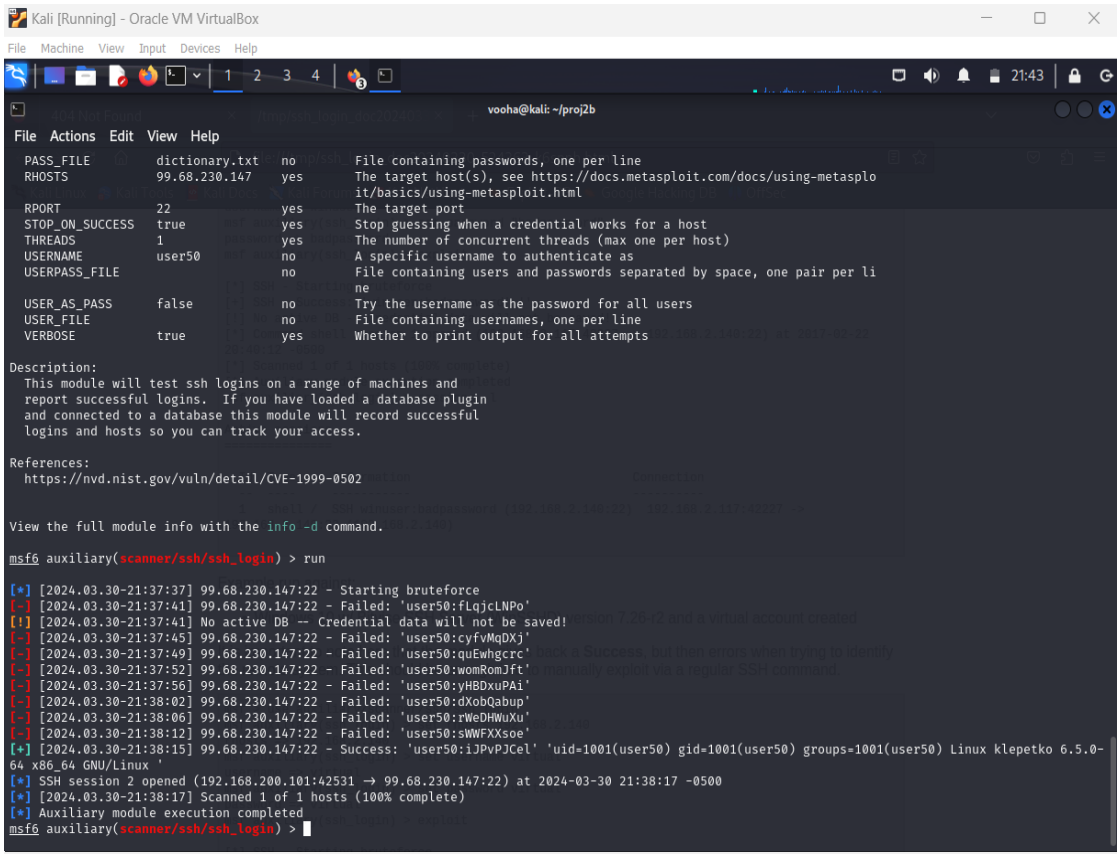
Section VII: (Task V):

For cracking “user50” to klepetko.net,

i) Show the screen shot of the parameters of the ssh login module. Use the “info” command in the MSF console.



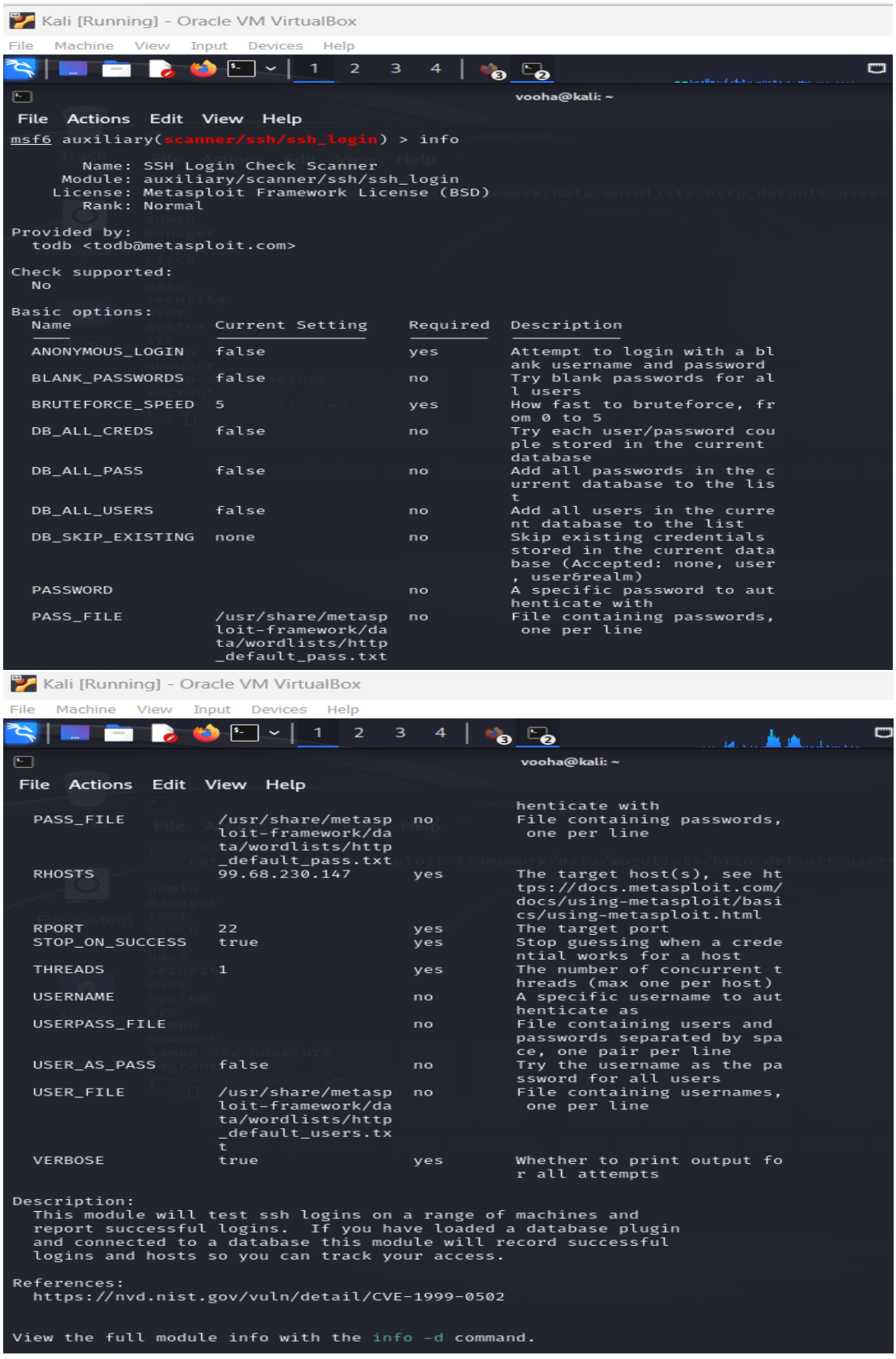
ii) Show the screen shot of finding the correct password in the MSF console.



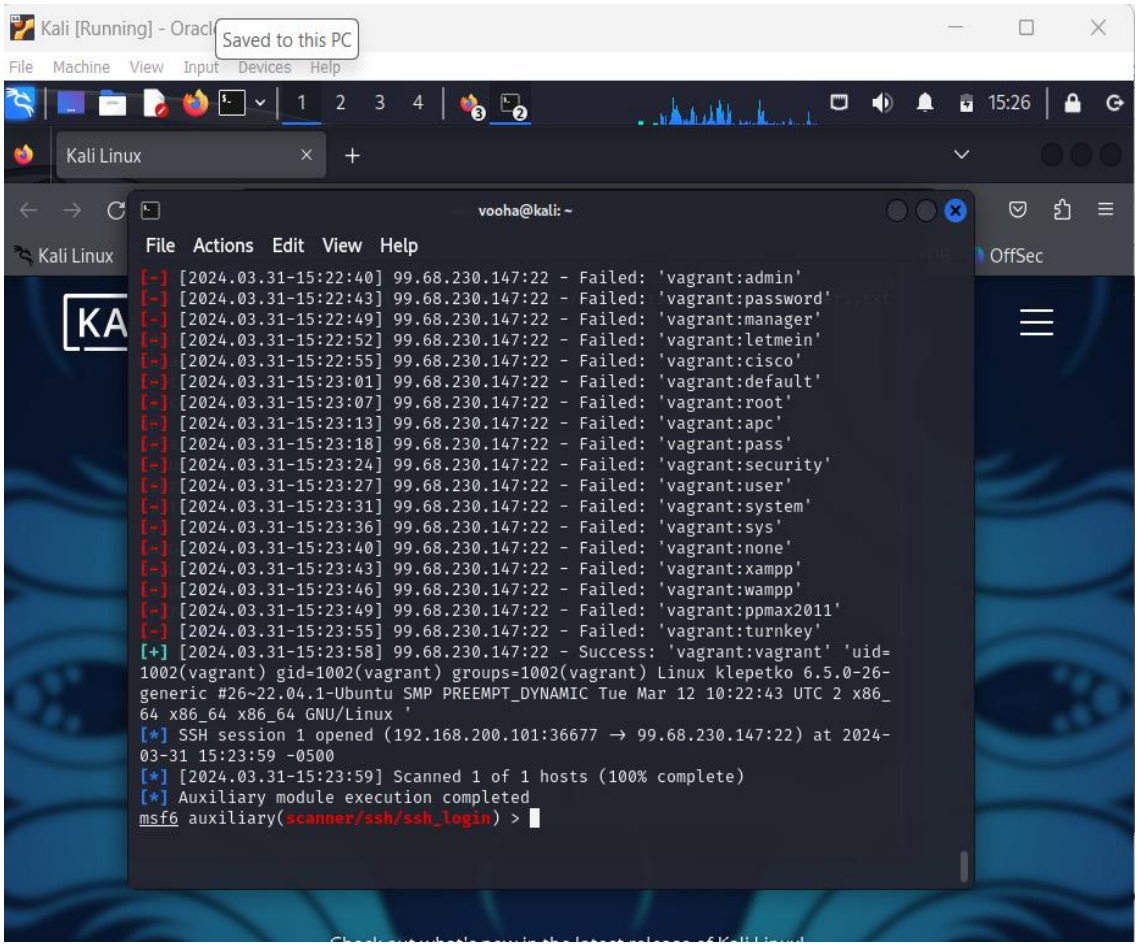
iii) Report how long it takes to test each password in average.
From the above screenshot as per the timestamps it took 38 seconds to crack 9 passwords. So, time taken to test each password in an average is 38/9 seconds = 4.22 seconds

For cracking ssh using username dictionary,

iv) Show the screen shot of the parameters of the ssh login module. Use the “info” command in the MSF console.



v) Show the screen shot of finding the correct username and password in the MSF console.



vi) Report how long it takes to test each password in average.
Total attempts made = 14 * 19 = 266
Time taken to crack the password = 1092 seconds (as per timestamps)
So average time taken to test each password = 1092/266 = 4.10 seconds

Section VIII: (Task VI):

i) Show the screen shot of your cryptoanalysis program when you get the key.

```
┌──(vooha㉿kali)-[~/proj2b]
└─$ python3 dec.py
Keys generated : 0x 09bb2443decc
Keys generated : 0x 09bb2443decf
Keys generated : 0x 09bb2443dece
Keys generated : 0x 09bb2443dec9
Keys generated : 0x 09bb2443dec8
Keys generated : 0x 09bb2443decb
Keys generated : 0x 09bb2443deca
Decryption successful with key: 0x 09bb2443decc

┌──(vooha㉿kali)-[~/proj2b]
└─$
```

ii) Show the key.

```
┌──(vooha㉿kali)-[~/proj2b]
└─$ python3 dec.py
Keys generated : 0x 09bb2443decc
Keys generated : 0x 09bb2443decf
Keys generated : 0x 09bb2443dece
Keys generated : 0x 09bb2443dec9
Keys generated : 0x 09bb2443dec8
Keys generated : 0x 09bb2443decb
Keys generated : 0x 09bb2443deca
Decryption successful with key: 0x 09bb2443decc

┌──(vooha㉿kali)-[~/proj2b]
└─$
```

iii) Show the content of the encrypted file secret.pdf.enc1.

```
vooha@kali: ~/proj2b
File  Actions  Edit  View  Help
%PDF-1}^]^?@E0`º<94>¥'^?);+^Xh<81><94>¾Ô^S<8e>©ç7å<8e>S¸õ%zé^O!Îå<9b>dqh^U<81>}1ïï$r^@è<8c>^R^B¶^L÷©^VÜã
ÎáÔéõè¸<9d>ëª>ypb^VÁ`û\ãNfõ<96><8e>»¡¾ª<84><91>ïãÇ<8b>^S\^V?^NÔ^FÝq<88>æÎ<¦Ûµ_^^h~
T<82>^DÔ=s'ò^Yw´å <8f>/¬»<9c><9f>Q<92><88>X°^Eþ<89>d^B\<9c>mý^TL{¬Ú^Fa.^Z²^W_¸&Üm©òbÍ@S^^Ôã1ÒHÔËeúò^V^Y^
W<84>¢QT^]þ»:h<88>ç+y×^Pæò<90>d<91>"e   ò<85>r¼B±       U^FÄ2(õ^Ex<æ4)^CÕ<8d><8c>·^L6^BC        ^@Iœåõ=0
<89>ÎÕif h3s<84>ÿÝÂ<80>À^Açö¦ù^_k<9c>®ùqÂúÆ-º-q67âþo<9d>;<86>^QæÛ^T^FgÐ<84>wœsJoî<ðÜä?ïZ`Ü)^AM^Uv}¯gWí+<
86>t<96>l>Í<8a>^P<9f>]/úç^Zz^AµRÑ{f(o¦<88>°Ô"¾üã<96>+4^Y¶wÝF$t¾ <90>^MC^ZUS6Õ<8c>Ú¼<g^LøhB<8c>R^FSñÕ<95>
$^BÃõ^Q^U^U^Vg÷ê±ðw<96>ó<92>Q9P0h(^Mîfn¥»Âç<94>Ñ^Kr.Zû8^Nç^Yo^G?i99Þ^E^E²ãr<89>^\z1ÿ*i8Ò>¼óçÎV<81>û<92>N
\ýtEs^O¡iøp
\^Qõ<9d> <9d>^]^Nµ<90>Ázú"<84>lÈ¢±<88>Z<8d>Ø¿i¡Ú<9d>L?^M)^M¦+MeX       5
=rì ]}Ú6R<94>eÔè¡ãñc¡?<8a>Ú\GÚbÿ«kô4Øüd¿<8b>»^Qã}ÇE<83>ÀHÊÞ¿EÏÜ¢ôún%»ßÅTquv<95>@<90>ê^EÏöÂÛWË¨^]4Ï0Ç=Öà¾
<^G í<84>B~<8a>'^<82>Ï<9e>3ñ<99>L6j<90>ÜMY_Þøîç<84>F^D<8e>ÜX^?gË^_JzB^[FáôZØ^OÔ!3S¢O><99>fþt¢XUL ^TÜËNeÆ
üw^RGdu<9d>*<81>^FÁ<91>Î^Hg÷^W<94>ÖHlô<86>^V,n;ñûÃçõÏ6<9d>, ¾éÁ<9d>Üt<98>Ð°<9e>ÿë<8e>®?ý7×ê<ä"5ùÙÔm^^¾O^
]x<98><92>ø<86>|^F±;x<97>4Y)P<90>/Õ»^M^V÷¨l Qô ^Q<88>^Y(Ö^G<8d>,7^B0<84>;'ÜP®'F|ÔÌ^H^C ú
ÿ\ þþ£^[£¹R^@4¤><91>À#ÛIü
^H¯<86>Ô×Ej3<94>ãkS»·P÷G¾x+\ýxóuy.ZÉ^]ü'ä<9b>^BK^B^N¿^?ÜÔ4;LÞüÿÀ^\<90>^BÌ^_Ëê^R×.^_â"F±trêGÙµÍ<87>F½^Q¥0
d<8b>9^Y^BçxC^NÛË£^H º^^ÒÝJ®Ëv<91>9%2QÏ6Væ<81>t¿Á[æ¡³^OÈ^XÞFôE^1×j6føW<87>Q¾À+)<9f>ŸË8^F¤µ+ÔB^F<90>è<8f>
>Ù<97>?=gì,<95>ßþë<89>U`l¤<96>ñ^_^PöÕ^Y~!^G(Þ¿zû8Dq-^^g##^P,<86><y¤Ú,=<8e>.F^M|¤UP¶h¶.:%Ó3#ø@4÷Ó
Y¦<93>m_Ù¥æ<93>£ÿIß<95>^^«P>×BnA<99><85>^D<86>Ñô<86>ç¡<,<91>D`ä¸^]^Gÿ^C^HÈí<9c>RL^T¦à»Iy^VÉÛF$.<83>s<95>
<91>8¼öß³)^Z^_<92>Â<99> p
42.©ë>\^WëOV<8e>S¯;^\+Sv:êÊúnÀ§¡<85>5ø¼gœãuºx$P^^þ^üF<81>ä^Oh^@^QF^Aûúœ¿ßçl^RH*^L&<82>h(<89>2-º^B¦UM^^îæ
>P^Vyf<97>n<94>~<90>^Y!<85><9f>)ðpá¦^AªY¥^N^T£¾ªÄöï^U^N»¢Nx±·34¤Z<93><92>ôº<8a>^X^GAIcùø&Ô^L<9a>Ï÷(Ñ^N5^
Po<8f>nñ<91>¶ŒMÑ<8e>î<81>©^F]ºG<81>^R6ÍÐ4üÍ| ^KÖC^L^Dy¡<92>³-^W^OæÑs^W#vzhw^C<91>`^T'^W¯JVdÞÍÇ<8b>º|Êà0g
^U(ër<8a><9d><9d>[a<86>îi'<9c>±é×àÅ<8e><85>^P>^^ç<9b>o£ h \<87>Í|<8f>Æ ^Ks^RÍ7xï%5Í<90>[}<88><98><97>ëµT
+º0-<9c>EÅsû]<97>Ô§_å^]<84>W^[QídÔ#ÑkØûFb^Y¾ú²Å<8d>ê^TQ"ýÙQcÁ^F<9a>I³Ð<9b>{@b"Ä<89>ðCÛ-8^WE^PR ôÉ^NÑ;£<9
4>><8a>ü·ÅånÙ}£Ú*<89><85>;òP^\
@@@
```

Section IX: (Task VII):

Show the screen shot of your DES program when it deciphers the testing file.

i) Show the screen shot of your DES program when you are brute force cracking the key of secret.pdf.enc2.

ii) Report how many keys are tested in 10 minutes.

iii) Estimate how long it will take to find the key. Note that you may not be able to find the key given the current hardware.

Section X: (Conclusion):

Found issues in task 4 related to ssh null character one but resolved it by truncating the string

Found issues in instatllation of dvwa but resolved browsing followed proper steps. Unable to resolve issues with task3 and task 6 a part of it is resolved.