Introduction

In this lab you will become familiar with the basic Linux utilities used for the configuration and diagnose of the network system and firewall.

This document describes the tasks you will do in the lab and must not be viewed as a manual or tutorial (although rather detailed information is provided). You are expected to use the slides from the lecture, to read the documents available under the “Useful Links” page in Canvas and to make full use of the search engines on the web. See also Deliverables section at the end of this document.

The lab environment is provided as a VirtualBox appliance, which is a set of virtual machines (VMs) bundled together. You will need to install VirtualBox on your laptop/computer in order to instantiate the environment. VirtualBox is available for free for all major operating systems and can be downloaded from https://www.virtualbox.org/wiki/Downloads. You find a detailed VirtualBox manual at https://www.virtualbox.org/manual/UserManual.html

You should also install the VirtualBox Extension Pack. The Extension Pack can make your work with VirtualBox more comfortable by allowing you to share the clipboard and folders between VMs and the host[1] where VirtualBox is running. Note that after you have installed the Extension Pack on the host you must also install the Guest Additions in every VM that you want using the features from the Extension Pack. To do that, start the VM and after the guest OS is loaded select the `Insert Guest Additions CD image` from the VirtualBox drop-down `Devices` menu. However, beware that version number of the Extension Pack installed in the VM must match the VirtualBox version number. If you move your VM after installing the Guest Additions to another host with a different VirtualBox version, you must reinstall the Guest Additions. Otherwise, there is a risk that various features will malfunction and for the guest OS to freeze or crash.

Note that the Extension Pack is not essential for completing the lab. In fact, during the lab you will learn how to forward SSH connections between the host and the guests. This will allow you to use the GUI on the host for doing cut-and-paste between host and guests. This should be an acceptable solution for most cases.

You will also need to install Wireshark on your host, if you don’t have it already. You can download Wireshark from https://www.wireshark.org/download.html.

If you are running Windows on your host, in addition you will need a SSH client. The easiest way to get that is to install the Putty MSI installer from http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

[1] When using VMs, the *host* denotes the computer on which VirtualBox is running. The *guest* is the VM running under the control of VirtualBox. The *host OS* and *guest OS* refer to the operating system running on the host and guest, respectively.

IMPORTANT:
When you solve the tasks below make sure you take screenshots to prove that your changes have the desired effect. You will be asked to include the screenshots in the report to be submitted after you complete the tasks.

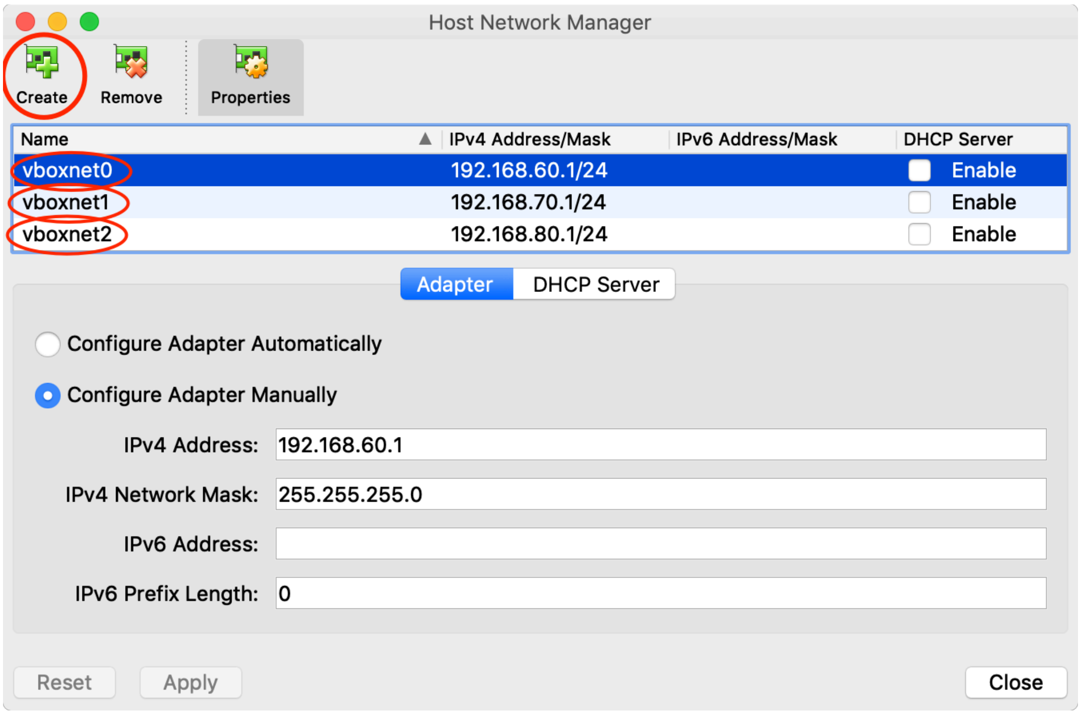
Import the VirtualBox appliance

Shutdown VirtualBox VMs that are currently running on your computer, if any. This will reduce the risk of errors due to overlapping network configurations or resource drain on the host.

Download ET2595.ova appliance as instructed in Canvas. Start VirtualBox and choose File→Import Appliance. Select ET2595.ova. Go to VirtualBox Preferences and select Network→Host-only networks.

If you have used VirtualBox before there may be host-only networks defined already. We will leave those alone, and create new ones.

Select the menu option File→Host Network Manager… from the VirtualBox menu. Create a host-only network by clicking the + icon as shown[2] in Figure 1. Enter IPv4 address 192.168.60.1 and netmask 255.255.255.0 as shown in Figure 1. Leave the IPv6 fields as they are. The DHCP Server option (under the DHCP Server button) must be disabled so it doesn't interfere with your experiments.



Name	IPv4 Address/Mask	IPv6 Address/Mask	DHCP Server
vboxnet0	192.168.60.1/24		□ Enable
vboxnet1	192.168.70.1/24		□ Enable
vboxnet2	192.168.80.1/24		□ Enable

○ Configure Adapter Automatically

● Configure Adapter Manually

IPv4 Address: 192.168.60.1

IPv4 Network Mask: 255.255.255.0

IPv6 Address:

IPv6 Prefix Length: 0

Figure 1 Adding a host-only network

[2] The GUI may look slightly different depending on the operating system. The figures depict VirtualBox under Mac OS X. The names of the host-only interfaces (circled in red in Figure 1) may differ as well. On Mac/Linux, the name starts with `vboxnet` and on Windows with `VirtualBox Host-Only Ethernet Adapter.`

Create a second host-only interface and set IPv4 address 192.168.70.1 and netmask 255.255.255.0. Finally, create a third host-only interface and set IPv4 address to 192.168.80.1 and netmask to 255.255.255.0.

Make a note of the name of the host-only network adapters created (see Figure 1). In this lab module you will use the host-only adapter with IP address 192.168.60.1, but the other host-only adapters will be active. The other host-only adapters will be used in later modules.

Networking

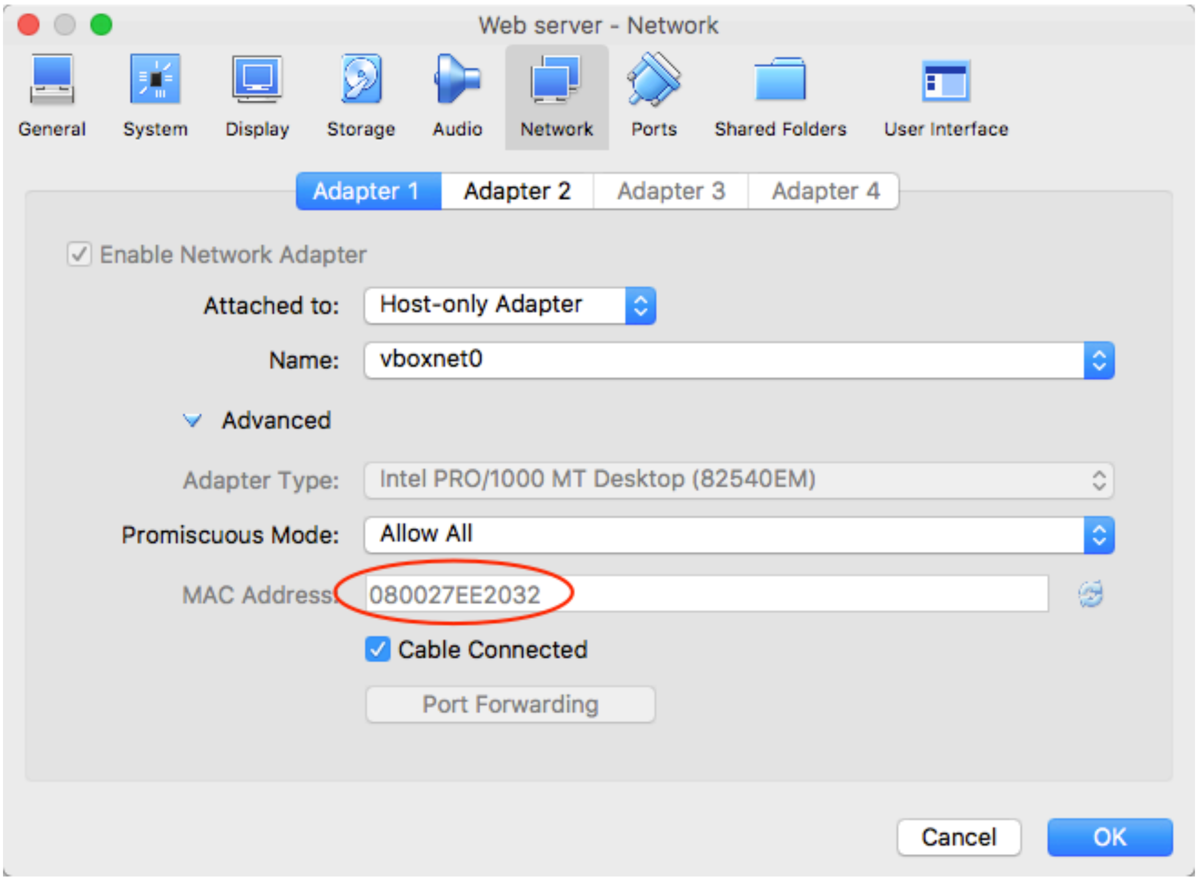
Start the Server A VM and open a Terminal. An easy way to do that is to click on the "Search your computer" button located at the top left pane and then enter "Terminal" in the text box that appears. Network admins and security experts spend a lot of time in the terminal!

The Server A VM is running Ubuntu Linux Desktop 18.04.1 with the latest updates installed. There are three network interfaces connected to the VM: two host-only adapters and a NAT adapter. You can see this on the host side by selecting the Server A VM in VirtualBox, clicking on the Settings button and then selecting the Network option. The network configuration[3] is shown in Figure 2. If you click on the Advanced option, more information about the adapter will be revealed. Particularly important is the MAC address of the interface, circled in red in Figure 2. The MAC address will allow you to identify the interfaces inside the guest OS.

Task 1: MAC addresses

Identify the MAC address of the configured adapters in the Web Server VM and write them down.

[3] If you use a Windows host, the host-only will be called `VirtualBox Host-Only Ethernet Adapter`, possibly with #2 or #3 added at the end. On Mac and Linux, they are called `vboxnet0, vboxnet1` and so on.



Web server - Network

☑ Enable Network Adapter

Attached to: Host-only Adapter

Name: vboxnet0

Advanced

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode: Allow All

MAC Address: 080027EE2032

☑ Cable Connected

Port Forwarding

Figure 2 Network adapters (interfaces)

The guest Linux OS will assign various names to the interface. The old naming scheme was *eth0*, *eth1* etc. In the new scheme, the adapters are named depending on how the Ethernet card is connected in the (emulated) hardware.

To list the interfaces available in the Linux guest, go to the Terminal you opened earlier and enter the following commands (only the part written in italics.

sudo ip link

All Linux command have a corresponding manual page containing information about the features of the command and syntax to turn them on or off. For example, you can enter the following commands to view the manual pages for the commands above:

man ip

Use the arrow keys to navigate in the man page and press `q` to quit and return to terminal. To search for a word, enter / followed by the string you want to search (no space between the slash character and the string). Press `n` to go the next occurrence of the searched string, and `N` to find previous occurrence.

Task 2: Network interfaces

In the interface list shown by one of the command above, use the MAC numbers from Task 1 to identify which interface is the NAT interface and which ones are the host-only interfaces.

The command *ip link* does not show any IP addresses associated with the links. This is because the *link* switch shows only link layer properties (and IP addresses are used at the network layer). To show IP addresses enter the following command:

sudo ip address

Sometimes, the wealth of information coming from iproute-2 can be distracting. One way to reduce that is to show only IPv4 or IPv6 information.

sudo ip -4 address
sudo ip -6 address

We will not use IPv6 in this lab.

Task 3: IP addresses, netmasks and subnet

Note down what IP addresses and netmasks are assigned to which interfaces. Derive the network addresses (the subnets) associated with each address and note them down. Remember from the lecture that the network address can be computed from the following operation (AND is the bitwise AND operation):

(IP address) AND (network mask)

These IP addresses were statically assigned to the interfaces. You can see that by examining the contents of the file `/etc/networking/interfaces` that defines how interfaces must configured during boot. You can view the contents of the file using one of the following commands:

sudo cat /etc/network/interfaces	this will print the contents in the terminal
sudo less /etc/network/interfaces	type **q** to exist the program

You can also view the contents with an editor such as *leafpad*[4], *atom*, *nano*, or *gedit*. Just remember to prepend the command *sudo*, which causes the following command to execute with root (admin) privileges allowing you to access root-protected content.

[4] If you are new to Linux, try to use *leafpad* or *gedit*. They should provide a rather familiar GUI for editing text files. If you edit files over an SSH connection, try *nano*. If you aim to become a pro, learn *vi* – it's the default text editor on most UNIX/Linux installations.

Virtualized network environment in VirtualBox

The NAT interface allows your VM to communicate with the outside world. Packets that go from the VM to the outside world over the NAT interface have their source IP address replaced with that of the gateway. Packets going in the opposite direction have their destination address (which is the address of the gateway) replaced with the IP address of the VM.

In the file /etc/network/`interfaces` you can find information related to which gateway (10.0.98.2) the guest OS is using. Similarly, in /etc/resolv.conf you see which DNS server (10.0.98.3) is being used. It is important to understand that these IP addresses are assigned to a NAT interface emulated internally by VirtualBox. The NAT interface is not visible with any commands, but behaves like a real Ethernet interface. When packets are sent to it (using one of the IP addresses above), they are received by the VirtualBox process.

The VM is configured to instruct VirtualBox to run a DNS proxy on its behalf on IP address 10.0.98.3. When VirtualBox receives DNS requests on this address from the VM, it passes them on to the DNS proxy. Then, the proxy forwards them to the DNS server configured on the host. The corresponding reply from the DNS server is forwarded by the proxy from the host to the VM interface that sent the request.

Similarly, when packets are sent to the gateway (10.0.98.2), they are received by the VirtualBox process. In this case, VirtualBox uses its own NAT engine to do network address translation, as explained above, and then asks the host to forward the packets to the destination. Incoming packets for the VM are forwarded by the host to the VirtualBox process. VirtualBox does network address translation and then sends them to the VM NAT-designated interface. We call this a NAT-designated interfaces because the guest OS is not aware that packets sent over this interface are subject to NAT – there is no NAT configured in the guest OS.

It is crucial to realize that the NAT engine from VirtualBox is completely decoupled from the any NAT (or other firewall settings) that you will configure on the VM. If, during the lab, you find that the VM cannot communicate with the outside world, don't attempt to reconfigure the host network settings as a first step. Most likely, the problem is related to the network settings in the guest OS.

The host-only interfaces allow multiple VMs to communicate with each other and with the host. However, the VMs are not allowed to communicate with the outside world directly through a host-only interface. VirtualBox implements a host-only network by creating a software-based interface in the host OS. The host-only interfaces are called *vboxnet0, vboxnet1, vboxnet2* etc., if you are running Mac or Linux host. If your host is using Windows, then the host-only interfaces are called *VirtualBox Host-Only Ethernet Adapter*, *VirtualBox Host-Only Ethernet Adapter #2*, *VirtualBox Host-Only Ethernet Adapter #3* etc.

Task 4: Host-only interfaces

If your host OS is Mac OS X or Linux open a Terminal and enter the following command:
ifconfig -a

Otherwise, if your host OS is based on MS Windows open a Command Prompt (as admin) and enter the following command
ipconfig /all

Note down the IP address and netmask assigned to each host-only interface. Which host-only interface on the host is connected to which host-only interface in the guest? Explain how you reached that conclusion.

Now, you will practice commands that allow you to access the routing table in the host OS.

Task 5: Routing tables in the host OS

If your host OS is Mac OS X open a Terminal and enter the following command:

```
netstat –f inet -rn
```

Otherwise, if your host is a Linux OS open a terminal and enter the following commands:

```
ip -4 route
```

Finally, if your host OS is based on MS Windows open a Command Prompt (as admin) and enter the following command

```
route -4 PRINT
```

Note down over what interface you can reach the default gateway for your host.

You will do the same thing for the guest OS (i.e., switch to the VM now!)

Task 6: Routing tables in the guest OS

In the guest OS (i.e., in the VM) open a Terminal and enter the following command:

```
ip -4 route
```

Note down over what interface you can reach the default gateway for your host. Is it the NAT interface or the host-only interface?

Start Wireshark both in the host and in the guest. Select the corresponding host-only interface to capture traffic with each started Wireshark instance. That is, on the host select the host-only interface that corresponds to the host-only interface that Server A guest OS is using. In the guest, choose the host-only interface that corresponds to the host-only interface selected on the host. Remember that in this lab module we care only about the host-only interface that is assigned IP address 192.168.60.100 by the guest OS. This is the host-only interface that must be used.

Task 7: Ping the host-based host-only interface

In the terminal in the guest OS ping the IP address corresponding to the host-only interface in the host OS. After 4-5 seconds, stop the ping and the Wireshark capture. Examine the ICMP traffic from the two Wireshark instances. Is it identical? Take screenshots to prove it!

We are now going to configure your virtualized environment to ease the diagnose of the following tasks and also to make it a bit more comfortable. In VirtualBox, select the Server A VM and click Settings. Go to Network→ Adapter3 (the NAT adapter), click Advanced and then Port Forwarding. At this point, click the + sign to add a new forwarding rule.

Under Name enter SSH. If necessary, change the protocol field to show TCP. Leave empty the Host IP field and enter 10022 in the Host Port field. Leave empty the Guest IP field and enter 22 in the Guest Port field. Click OK to close the Port Forwarding dialog and the click OK again to close the Settings window and save your changes.

What you have done was to add a forwarding rule to the NAT engine in VirtualBox. The rule allows you to establish a SSH connection from your host OS to the guest OS. An SSH connection is equivalent of having remote (encrypted) access to terminal in the guest OS. Essentially the port 10022 on the host is hardwired to the port 22 on the guest OS where the SSH server is listening for incoming connections.

Task 8: ssh into VM via localhost

If you are using Mac or Linux enter the following command on your host:

ssh -p student@10022 localhost

You may see a message that begins with

The authenticity of host '[localhost]:10022 ([127.0.0.1]:10022)' can't be established.

This happens because this is the first time you are connecting to this SSH server and the SSH client want to make sure you authenticate the server. Enter *yes* to continue.

Otherwise, if you are using Windows, start Putty and enter localhost in Host name (or IP address) field and 10022 in the Port field. Then click the Open button. As described for Mac/Linux, you may be asked to authenticate the server. Select *Yes* to indicate you trust the host.

Your shell prompt should say **student@serverA.** You have now a remote shell opened to Server A and you can run any console commands in it (as long as they don't require a GUI). Get a screenshot to prove it!

You will now add forwarding rules for HTTP and HTTPS.

Task 9: Add forwarding rules for HTTP and HTTPS in VirtualBox

Use the same technique as outlined above to add forwarding rules for HTTP and HTTPS. Lookup the official port numbers used by HTTP and HTTPS. Choose the Host port by adding 10,000 to the official port number for HTTP and HTTPS, respectively. The Guest ports will use the official numbers.

What Host port numbers did you use? Start a web browser on the host and show that you can connects to the apache2 server in the guest over HTTP and HTTPS, respectively.

The changes made in Task 9 allow a user from the host to browse the web server in the VM guest OS over HTTP and HTTPS.

Firewall (iptables)

The first part of the document was meant to get you comfortable with the lab environment, both on the host and the guest. In this part, you will begin to configure the firewall on the guest side (i.e., inside the VM). There should be no need to make modifications to the firewall on the host side. If you want to reset the firewall rules, just reboot the VM. You will learn later how to do it in a more elegant way.

Before continuing, make sure you have read the *iptables* documents provided in Its Learning, under Links. Also, the manual pages for iptables, *man iptables* and *man iptables-extensions,* are very useful sources of information.

You may recall from our firewall lecture that there are five default chains in Netfilter: PREROUTING, INPUT, FORWARD, OUTPUT, and POSTROUTING (see Figure 3 on next page). There are also four tables[5]: raw, mangle, nat, and filter. Each chain is associated with several tables, but only the OUTPUT chain uses all of tables.

Packets move from chain to chain. When a packet reaches a chain, it will traverse all the tables associated with the chain in the order listed above. You use the *iptables* command to

[5] There can be more than four tables if specific kernel extensions are turned on (e.g., SElinux)

insert rules in tables that define how packets are being handled in a particular chain. When you write, for example,

iptables -t filter –A INPUT <filter> -j <action>

it means that packets that reach the filter table in the INPUT chain and match the filter <filter> (e.g., src 10.0.0.5) will be subjected to <action> (e.g., DROP or ACCEPT).

The left side of Figure 3 shows the processing path for a packet arriving at a host's network card. The packet first enter the PREROUTING chain. Without going too much into details, inside the PREROUTING chain the packet will enter the raw, mangle and nat tables, in this order. The green box labeled Connection tracking refers to the kernel module that is used for implementing a stateful packet filter firewall.

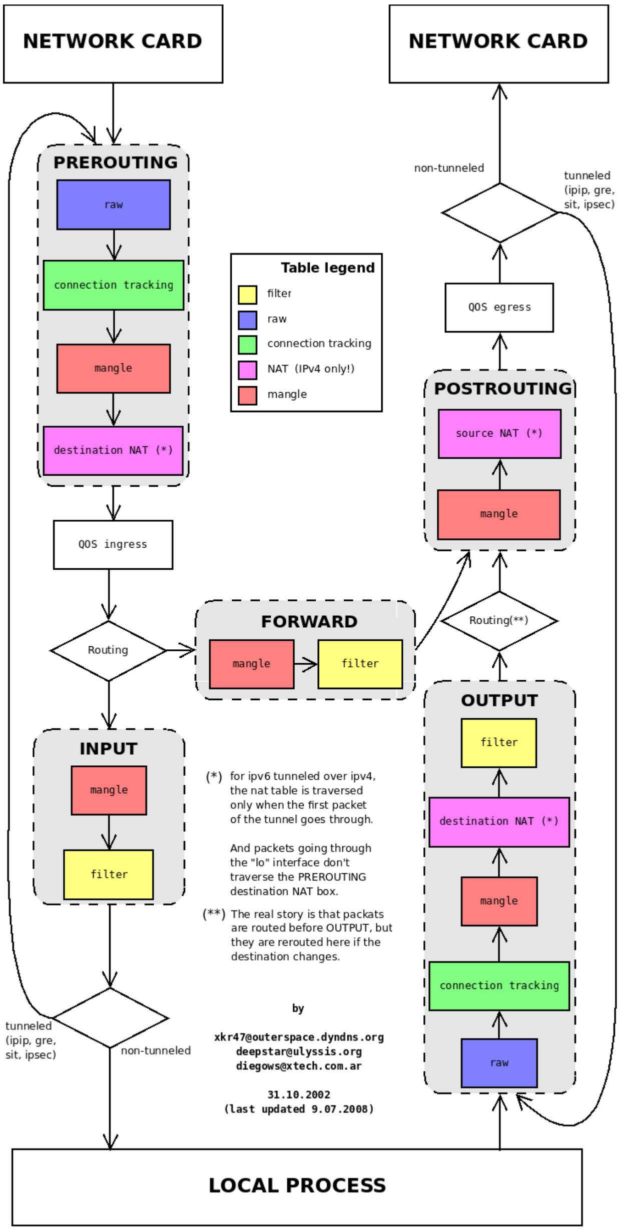


Figure 3 Netfilter packet processing (from Jonas Berlin aka. xkr47)
http://xkr47.outerspace.dyndns.org/netfilter/packet_flow/

If the packet is not dropped it is passed to the routing module. The module decides if the final destination of the packet is on the current host or if the packet must be forwarded to another host on the path to the final destination. If this host is the destination host, the packet enters the INPUT chain and is passed to the mangle table and then to the filter table. Otherwise, it is passed to the FORWARD chain and then to the POSTROUTING chain.

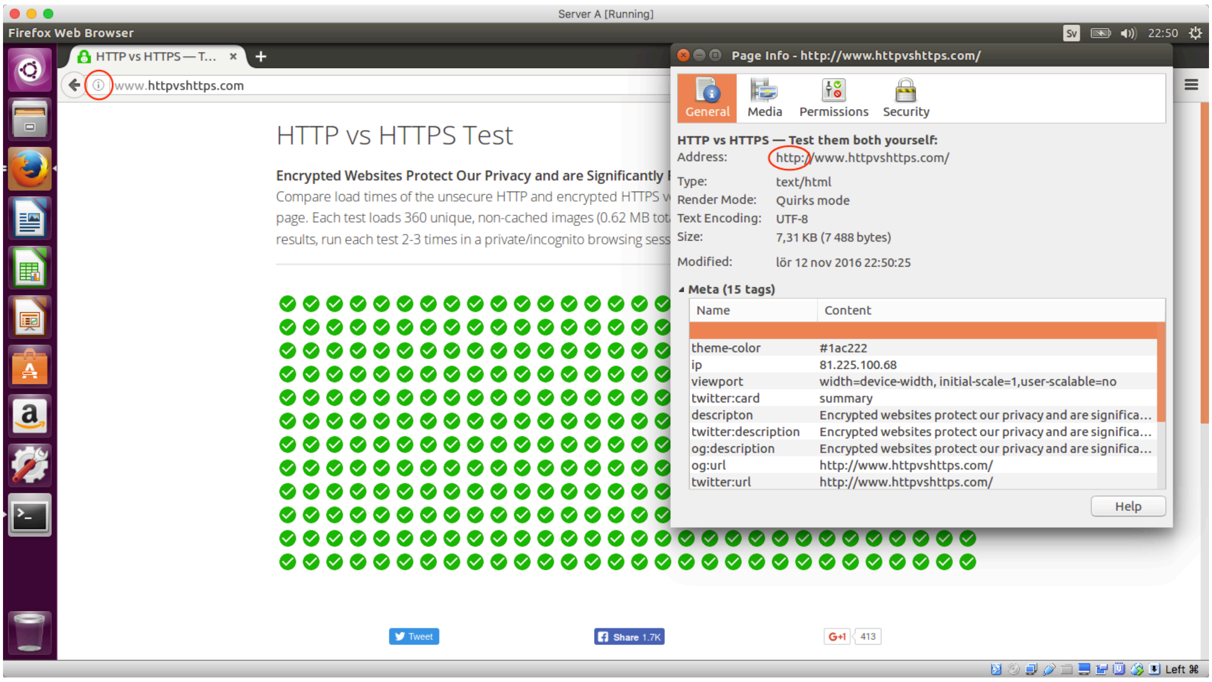
A packet send by a local process on this host enters first the OUTPUT chain and traverses the raw, mangle, nat, and filter tables. Accepted packets are processed by the routing module where the outgoing network interface is selected. Then, the packet enters the POSTROUTING chain where, for example, NAT processing may occur.

Task 10: Default firewall policy and rules

Use the iptables command to find out the default policy and the rules installed by default in the VM in the tables *filter*, *mangle,* and *nat*. You can see the default policy when you list the rules in each table

Note what is the default policy for each table.

The next step is to verify that a user from the VM can view web sites over HTTP and HTTPS. In the Server A VM, open Firefox and browse a web site using HTTP, which is the non-encrypted version of the protocol. For example, try http://www.httpvshttps.com. Click the inscribed letter (i) shown in Figure 4, click the right-angle bracket >, and then click "More information", and finally select the General button. You should be looking at a window as the one shown in Figure 4. You can see that the web page was served over HTTP.



HTTP vs HTTPS Test

Encrypted Websites Protect Our Privacy and are Significantly

Compare load times of the unsecure HTTP and encrypted HTTPS v
page. Each test loads 360 unique, non-cached images (0.62 MB tot
results, run each test 2-3 times in a private/incognito browsing sess

HTTP vs HTTPS — Test them both yourself:

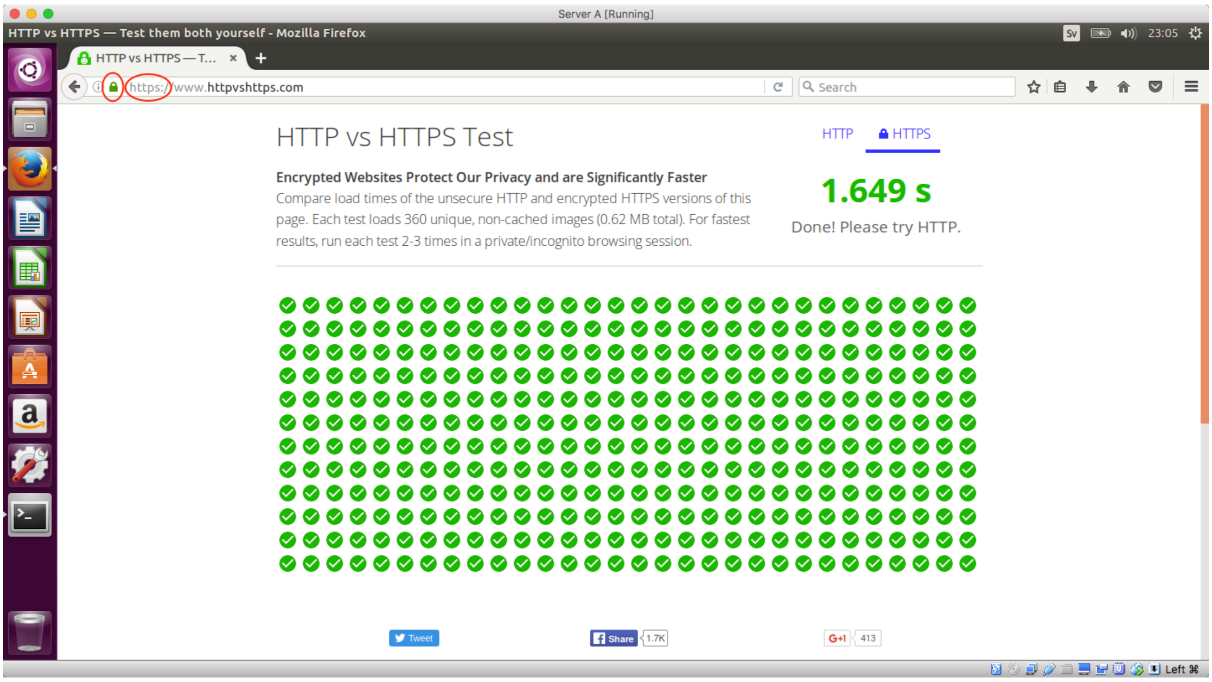
Address:	http://www.httpvshttps.com/
Type:	text/html
Render Mode:	Quirks mode
Text Encoding:	UTF-8
Size:	7,31 KB (7 488 bytes)
Modified:	lör 12 nov 2016 22:50:25

Meta (15 tags)

Name	Content
theme-color	#1ac222
ip	81.225.100.68
viewport	width=device-width, initial-scale=1,user-scalable=no
twitter:card	summary
descripton	Encrypted websites protect our privacy and are significa…
twitter:description	Encrypted websites protect our privacy and are significa…
og:description	Encrypted websites protect our privacy and are significa…
og:url	http://www.httpvshttps.com/
twitter:url	http://www.httpvshttps.com/

Figure 4 HTTP URL

Now, browse the HTTPS version. In this case the HTTP transactions are encrypted. For example, try https://www.httpvshttps.com. You should see *https* shown in the navigation field as well as a green lock (both encircled in red in Figure 5).



HTTP vs HTTPS Test

Encrypted Websites Protect Our Privacy and are Significantly Faster

Compare load times of the unsecure HTTP and encrypted HTTPS versions of this page. Each test loads 360 unique, non-cached images (0.62 MB total). For fastest results, run each test 2-3 times in a private/incognito browsing session.

1.649 s

Done! Please try HTTP.

Figure 5 HTTPS URL

You may notice that the page loads a lot faster with HTTPS then with HTTP. This seems counterintuitive at first because encryption should slow things down, if anything. The reason for the performance boost is that the page uses HTTP/2 when serving content over HTTPS. HTTP/2 is the new HTTP(S) standard the brings forward several optimizations such as multiplexed streams, request prioritization, header compression and server-initiated streams. The major web clients only support HTTP/2 over TLS, and thus plain HTTP does not receive the same benefits. For more details, see https://samrueby.com/2015/01/26/why-is-https-faster-than-http/.

Task 11: Block HTTP-browsing in the guest OS

Enter iptables rules to block **the user from the guest OS** from browsing HTTP pages, but retain the ability to browse HTTPS. You should still be able to browse HTTP traffic produced by the apache2 server, when browsing from **the host**.

Note the rules you have used. Verify that the rules are effective.

Task 12: Block Apache web server from serving content over HTTP

Enter iptables rules so that **the host** cannot view HTTP content from the apache2 server in the guest OS, but retain the ability to see content served over HTTPS

Note the rules you have used. Verify that the rules are effective.

Task 13: Unblock HTTP-browsing in the guest OS

Undo what you have in Task 11 so the user from the guest OS can browse both HTTP and HTTPS traffic.

Note the commands that you have used. Verify that the rules are effective.

When you set up the firewall rules, it is useful if you can collect them in a script file. Such a script file can be used to easily reload the rules on demand. Also, it can be easily customized for and copied to other hosts. A firewall script template, `firewall.sh.template`, has been prepared for you and is available in the `student` home directory on Server A. Copy the template to a working file called `firewall.sh`:

cp firewall.sh.template firewall.sh

From now on, all your firewall changes are made to `firewall.sh`. The template file will be your backup in case you need to have a clean firewall configuration.

Open the `firewall.sh` with a text editor. At the top of the file you will find variables for NAT interface, NAT IP address, host-only interface, host-only IP address and DNS server. Verify that the values assigned to those variables correspond to the environment you have in Server A. Change them as needed.

After customizing `firewall.sh` to you your environment, execute the script by entering

sudo ./firewall.sh

You have now reset the firewall to initial state. All your previous rules are gone.

Task 14: Use firewall.sh to configure the firewall

Modify the script to bring Server A firewall to the state you had Task 13 (guest OS can view HTTP and HTTPS pages, but apache2 server is blocked from serving HTTP content).

If you do something wrong and completely loose connectivity in Server A, reboot the VM. It will start with the firewall in the initial state.

After you verify you have the correct rules, note them down (take a screenshot from the editor).

Remove the rules from Task 14 (or comment them out). Execute the script (or reboot the VM) to reset you firewall to the initial state.

You will now change the default firewall policy for all tables to DROP (since we all agreed during lecture that is much safer than ALLOW).

Task 15: Change default firewall policy to DROP

Modify firewall.sh to implement the default DROP policy. What do you expect will happen with the connectivity if you install the policy? Execute the script to install the new policy. Try pinging the outside world from the VM. Also, try pinging the loopback[6] interface `lo`. Was your expectation correct?

Note the changes you made to the firewall rules to implement the default DROP policy

When configuring the firewall, you will eventually add an erroneous rule by mistake. Finding these errors can be challenging. Often, the tell-tale of such error is that a service stops communicating because its traffic is being dropped. If you run the command

iptables -t filter -L –v -n

you will see packet and byte counters associated with each rule. The counter values increase with each packet that matches the corresponding rule. Running the command multiple times while the service is unresponsive may help in identifying the faulty rule. The `–n` switch at the end of the command shows numbers instead of names for IP addresses and ports. This may yield a less cluttered output. Note that interface names are shown in the output only when the -v switch is used.

You can get additional help by configuring iptables to explicitly log all packets that are being dropped. To do that you need to append the following rules at the end of the firewall script:

[6] Also, called `localhost`. The difference between loopback and localhost is that loopback is the name of the *interfac*e that is assigned the IP address 127.0.0.1 while localhost is the name associated with IP address 127.0.0.1. When you ping localhost or 127.0.0.1, you are in fact pinging the loopback interface (which is shown as `lo` by the `ifconfig` command).

```
# Create logging chains
$IPT -t filter -N input_log
$IPT -t filter -N output_log
$IPT -t filter -N forward_log

# Set some logging targets for DROPPED packets
$IPT -t filter -A input_log -j LOG --log-level notice --log-
prefix "input drop: "
$IPT -t filter -A output_log -j LOG --log-level notice --log-
prefix "output drop: "
$IPT -t filter -A forward_log -j LOG --log-level notice --log-
prefix "forward drop: "
echo "Added logging"

# Return from the logging chain to the built-in chain
$IPT -t filter -A input_log -j RETURN
$IPT -t filter -A output_log -j RETURN
$IPT -t filter -A forward_log -j RETURN


# These rules must be inserted at the end of the built-in
# chain to log packets that will be dropped by the default
# DROP policy
$IPT -t filter -A INPUT -j input_log
$IPT -t filter -A OUTPUT -j output_log
$IPT -t filter -A FORWARD -j forward_log
```

NOTE: The rules are already appended at the end of the firewall script, for your convenience. You just need to uncomment them, by removing the hash character (#). It is very important that the last three rules shown above, are the rules added to your chain, as mentioned in the preceding comment. If you don't do that you will not see only DROPPED packets in the logs, but also accepted packets and you won't be able to tell the difference.

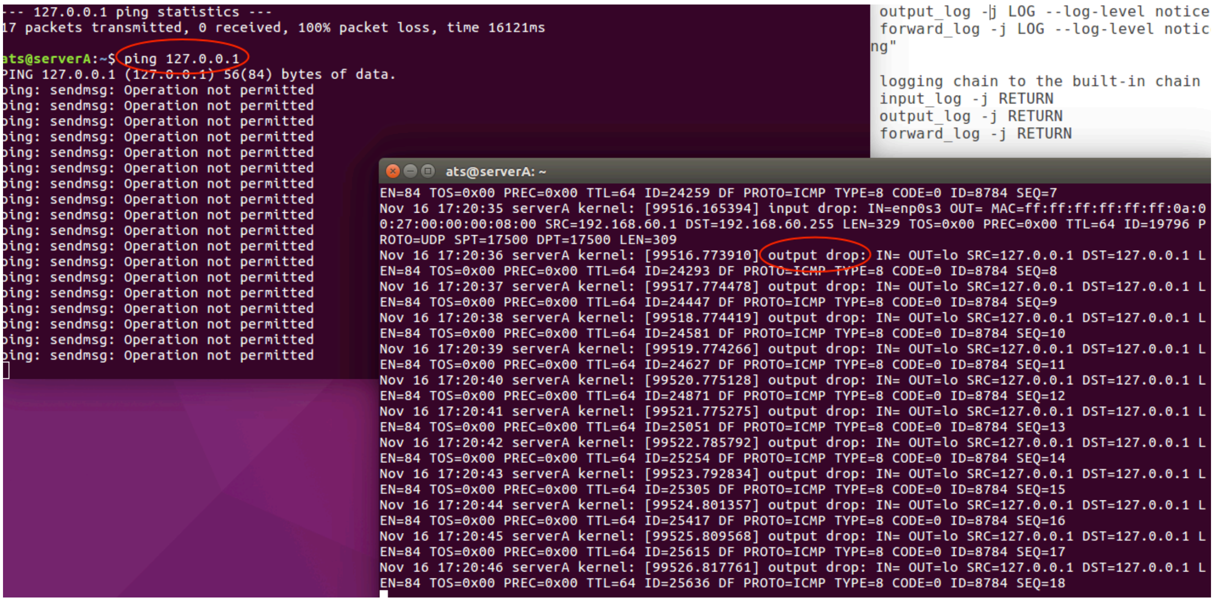
Task 16: Logging DROPPED packets

Enable the rules shown above in the firewall script. Try to understand what each rule does before running the firewall script. Consult the man page for `iptables-extensions` to understand how the LOG target works.

Execute the firewall script. Start a new Terminal window. In one of the terminal windows, enter the following command:

sudo tail -f /var/log/kern.log

You will be able to see live logs from the Linux kernel. In the other Terminal window start pinging the loopback interface, as shown in Figure 6. Now, what do you think is blocking your ping?



```
--- 127.0.0.1 ping statistics ---
17 packets transmitted, 0 received, 100% packet loss, time 16121ms

ats@serverA:~$ ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
```

```
 output_log -j LOG --log-level notice
 forward_log -j LOG --log-level notic
ng"

 logging chain to the built-in chain
 input_log -j RETURN
 output_log -j RETURN
 forward_log -j RETURN
```

ats@serverA: ~

```
EN=84 TOS=0x00 PREC=0x00 TTL=64 ID=24259 DF PROTO=ICMP TYPE=8 CODE=0 ID=8784 SEQ=7
Nov 16 17:20:35 serverA kernel: [99516.165394] input drop: IN=enp0s3 OUT= MAC=ff:ff:ff:ff:ff:ff:0a:0
0:27:00:00:00:08:00 SRC=192.168.60.1 DST=192.168.60.255 LEN=329 TOS=0x00 PREC=0x00 TTL=64 ID=19796 P
ROTO=UDP SPT=17500 DPT=17500 LEN=309
Nov 16 17:20:36 serverA kernel: [99516.773910] output drop: IN= OUT=lo SRC=127.0.0.1 DST=127.0.0.1 L
EN=84 TOS=0x00 PREC=0x00 TTL=64 ID=24293 DF PROTO=ICMP TYPE=8 CODE=0 ID=8784 SEQ=8
Nov 16 17:20:37 serverA kernel: [99517.774478] output drop: IN= OUT=lo SRC=127.0.0.1 DST=127.0.0.1 L
EN=84 TOS=0x00 PREC=0x00 TTL=64 ID=24447 DF PROTO=ICMP TYPE=8 CODE=0 ID=8784 SEQ=9
Nov 16 17:20:38 serverA kernel: [99518.774419] output drop: IN= OUT=lo SRC=127.0.0.1 DST=127.0.0.1 L
EN=84 TOS=0x00 PREC=0x00 TTL=64 ID=24581 DF PROTO=ICMP TYPE=8 CODE=0 ID=8784 SEQ=10
Nov 16 17:20:39 serverA kernel: [99519.774266] output drop: IN= OUT=lo SRC=127.0.0.1 DST=127.0.0.1 L
EN=84 TOS=0x00 PREC=0x00 TTL=64 ID=24627 DF PROTO=ICMP TYPE=8 CODE=0 ID=8784 SEQ=11
Nov 16 17:20:40 serverA kernel: [99520.775128] output drop: IN= OUT=lo SRC=127.0.0.1 DST=127.0.0.1 L
EN=84 TOS=0x00 PREC=0x00 TTL=64 ID=24871 DF PROTO=ICMP TYPE=8 CODE=0 ID=8784 SEQ=12
Nov 16 17:20:41 serverA kernel: [99521.775275] output drop: IN= OUT=lo SRC=127.0.0.1 DST=127.0.0.1 L
EN=84 TOS=0x00 PREC=0x00 TTL=64 ID=25051 DF PROTO=ICMP TYPE=8 CODE=0 ID=8784 SEQ=13
Nov 16 17:20:42 serverA kernel: [99522.785792] output drop: IN= OUT=lo SRC=127.0.0.1 DST=127.0.0.1 L
EN=84 TOS=0x00 PREC=0x00 TTL=64 ID=25254 DF PROTO=ICMP TYPE=8 CODE=0 ID=8784 SEQ=14
Nov 16 17:20:43 serverA kernel: [99523.792834] output drop: IN= OUT=lo SRC=127.0.0.1 DST=127.0.0.1 L
EN=84 TOS=0x00 PREC=0x00 TTL=64 ID=25305 DF PROTO=ICMP TYPE=8 CODE=0 ID=8784 SEQ=15
Nov 16 17:20:44 serverA kernel: [99524.801357] output drop: IN= OUT=lo SRC=127.0.0.1 DST=127.0.0.1 L
EN=84 TOS=0x00 PREC=0x00 TTL=64 ID=25417 DF PROTO=ICMP TYPE=8 CODE=0 ID=8784 SEQ=16
Nov 16 17:20:45 serverA kernel: [99525.809568] output drop: IN= OUT=lo SRC=127.0.0.1 DST=127.0.0.1 L
EN=84 TOS=0x00 PREC=0x00 TTL=64 ID=25615 DF PROTO=ICMP TYPE=8 CODE=0 ID=8784 SEQ=17
Nov 16 17:20:46 serverA kernel: [99526.817761] output drop: IN= OUT=lo SRC=127.0.0.1 DST=127.0.0.1 L
EN=84 TOS=0x00 PREC=0x00 TTL=64 ID=25636 DF PROTO=ICMP TYPE=8 CODE=0 ID=8784 SEQ=18
```

Figure 6 Kernel logs

Task 17: Enable traffic from loopback interface

Add firewall rules to enable all type of traffic to and from your loopback interface. Verify that pinging localhost works. You should also be able to SSH into localhost via the loopback interface:

ssh localhost

You should be prompted for the password. Enter the password. You are now inside an SSH session. Terminate the session by pressing Ctrl and D simultaneously.

Note the rules that you added.

The typical configuration for a server firewall trusts the users/applications running on the server and assumes the outside world is malicious. This means that you want to allow the server to initiate connections to the outside world, but be very restrictive in terms of what is allowed from the outside.

For a starter, we would to be able to ping the other interfaces attached to server A. However, in this case we allow specifically only outgoing ICPM Echo Request messages and incoming ICMP Echo Reply messages.

Task 18: Allow Server A to ping the other interfaces

Modify the firewall rules to allow ping traffic initiated from Server A. For ping traffic, you need to allow outgoing ICMP Echo Request and incoming ICMP Echo Reply messages.

Verify that the added rules work and note them down.

At this point we would like to be able to ping the outside world as well. As mentioned earlier in this document, your VM reaches the outside world through the NAT interface.

To make things comfortable you want to be able to use DNS names, so that, for example, you can ping www.google.com without having to resolve the DNS name manually.

Task 19: Allow Server A to ping all hosts

Add firewall rules to allow Server A to lookup names with help of the DNS server configured in `/etc/network/interfaces`. Modify the ICMP rules to allow sending IMCP Echo Request to any server and receiving the corresponding ICMP Echo Replies. No other type of ICMP traffic must be allowed with the outside world.

To verify DNS rules you can use the host or nslookup command:

host www.bth.se
nslookup www.bth.se

Verify your rule and note them down.

Start the Firefox web browser inside the Server A VM. Try to browse your favorite web site. You will notice that it is not possible. Currently, only ping and DNS traffic with the outside world is allowed to pass through the firewall.

In a standard firewall configuration, we should allow all TCP connections initiated from the host where the firewall is running, towards any destination, while at the same time blocking connections initiated from the outside. This approach will implicitly enable HTTP and HTTPS traffic initiated by Server A, since these are running over TCP.

However, you might remember from the lectures (or previous courses) that TCP uses a three-way handshake mechanism. When Server A initiates a connection, it sends a TCP segment with the SYN flag turned on and an initial sequence number, ISN A. The recipient accepts the incoming connection by replying with its own TCP segment where the flag SYN is turned on and containing an own initial sequence number, ISN B. In addition, the ACK flag is turned on and the acknowledgement sequence number is set to ISN A+1 to signal that the initial sequence number sent by Server A was accepted. Finally, Server A accepts the offered ISN B by sending a TCP segment with the ACK flag turned and an acknowledgement sequence number set to ISN B+1.

You will find it easy to add a rule that allows outgoing TCP connections from Server A to any destination, where the initial TCP segment is sent. But how will you define the iptables rule for the corresponding incoming TCP segment (2nd segment in the handshake) where the SYN and ACK flags are set? It's not the flags that is problem! The problem is that the answer can come from any host on the Internet, since you have not limited who Server A can talk to. In very specific scenarios, we may define a list of IP addresses that Server A is allowed to communicate with. But, for the general case, this is unacceptable. Just think about your laptops, pads and smart phones that use firewalls also, and still let you connect over TCP to any IP address you want to visit. The "trick" these devices use is to tell the firewall that it is OK to accept incoming TCP segments for connections that were initialized by the device. This means they maintain **state** about each connection – they run a stateful firewall.

Iptables implements a stateful firewall through the connection tracking extension, `conntrack`. A rule, such as,

```
$IPT -t filter -A INPUT -p tcp -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

tells the kernel that any TCP packet from an ESTABLISHED connection (one where the initial SYN segment was sent before a timeout occurred or RST segment was received in response) must be accepted. The RELATED flag tells the connection tracking mechanism, that even related connections should be accepted (e.g., this is the case for FTP where there is a control and a data connection).

Task 20: Enable stateful firewall

Using the hint above to add rules to your firewall to enable TCP connections to be established to any destination, and thus allow you to browse web sites with the Firefox browser from Server A.

Verify and note down the rules.

Try to SSH from your host into the VM, like you did in Task 8. You will find that it does not work. What do you think is the problem? Look at the kernel logs and review again the section Virtualized network environment in VirtualBox. Try also to use the web browser on the host to view content on Server A over HTTP and HTTPS like you did in Task 9: Add forwarding rules for HTTP and HTTPS in VirtualBox. You will find again that it does not work.

Task 21: Enable SSH and HTTPS content from apache2 server for web browser on host

Add iptables rules that enable a user to ssh into Server A from the host. In addition, add rule to enable a web browser on the host to view content served by the apache2 server over HTTPS, but not HTTP.

Verify and note down the rules

Start now the Client A VM. If you list the iptables rules on it, you will find out that it uses the default ACCEPT policy. Try pinging Client A from Server A. It should work, if you have kept the ipables rules from the previous tasks. Try pinging Server A from Client A. You will find it does not work.

Task 22: Ping Server A from Client A

Fix firewall rules on Client A, or Server A, or both (you must decide the adequate host where to add the rules) to enable pinging in addition of the rules from the previous tasks. Use the commands from Task 4 and Task 6 to learn about the configured network environment on Client A.

Verify and note down your changes.

Try to ssh from Server A to Client A. It should work, if you kept the iptables from the previous tasks. Don't forget to exit the SSH session (if you established one) by pressing Ctrl and D simultaneously.

Try to ssh from Client A to Server A. You will find out it does not work.

Task 23: SSH from Client A to Server A

Fix firewall rules on Client A, or Server A, or both (you must decide the adequate host where to add the rules) to enable an SSH session to be established from Client A to Server A.

Verify and note down your changes. Don't forget to exit the SSH session (if you established one) by pressing Ctrl and D simultaneously.

NOTE: If you are new to Linux, you should know that you can use scp (which uses ssh in turn) to copy files remotely over an encrypted connection. For example, if you are on Server A, you can copy the firewall template to Client A with the following command:

scp firewall.sh.template student@192.168.60.111:~/

The tilde character (~) is shorthand for /home/ats. The ability to copy files remotely will come handy in later lab modules.

Start the Firefox web browser on Client A. Use it to try to browse your favorite web site (feel free to try both HTTP and HTTPS). Try to ping www.google.com from Client A. You will find out that Client A is completely cut off from the Internet. The only other host it can reach, is Server A. Can we help it to get Internet access (without creating a NAT interface with VirtualBox)?

We can do that by arranging for Server A to forward Internet traffic from Client A over its NAT interface. This mean we are asking Server A to be the default gateway for Client A.

The first thing to do is configure Client A to use Server A for its Internet traffic. But how is "Internet traffic" defined? If you did like you were instructed in Task 21, you know the subnets to which Client A is connected through its interfaces. Anything different from those subnets is considered "Internet" and should be send to the default gateway. This is achieved by adding a default route to the kernel IP routing table[7]. Our default route, should take all traffic than cannot be handled by Client A and send it to Server A.

Task 24: Add gateway and DNS server to Client A

Edit the file /etc/network/interfaces on Client A and add the following line:

```
gateway 192.168.60.100
```

This instructs Client A to use the Server A as default gateway

Edit the file /etc/resolv.conf on Client A and verify the 10.0.98.3 is listed as DNS server:

```
nameserver 10.0.98.3
```

This line instructs Client A to use the DNS proxy provided by VirtualBox. Save the file and reboot Client A.

The second thing we need to do, is tell Server A that need to forward traffic between its interfaces (for us, in particularly the host-only and the NAT interface).

Task 25: Enable IP forwarding on Server A

In the terminal on Server A enter the following commands:

```
sudo sysctl -w net.ipv4.ip_forward=1
sudo syscltl -p
```

The first command turns on IP forwarding and the second command applies the change to the running kernel.

[7] Strictly speaking, this a forwarding table since we have no routing protocol running. See http://aftabsiddiqui.com/index.php/ip-routing-table-rib-and-forwarding-table-fib/

Try to ping www.google.com from Client A. You will discover that is still not reachable. What is the problem? Use Wireshark, iptables and the other networking tools to narrow down the problem. You should see that the packets are dropped in the forward chain.

If you remember, our default policy on the FORWARD chain is to drop the packets.

Task 26: Change iptables to forward packets

Add the following iptables rules that will enable packet forwarding:

```
$IPT -t filter -A FORWARD -i $HIF -j ACCEPT
$IPT -t filter -A FORWARD -i $NIF -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

Another problem is that packets coming from Client A use private addresses[8]. Private addresses can be used by anyone and are therefore not unique on the Internet. All routers have (should have!) built-in rules that drop packets using private addresses in the source IP address or destination IP address field. We have, foolishly configured, Server A to forward packets containing private addresses, but the next-hop recipient, VirtualBox, discards them.

The common way to route traffic from/to private addresses on the Internet is through network address translation (NAT). In our case, we need to tell Server A to use NAT (more specifically source NAT – SNAT) on packets sent out over the NAT interface. This means that the source IP address in the packets sent to the Internet over Server A's NAT interface will be replaced with IP address assigned to that interface. For incoming replies, the destination IP address of the packet (which is set to the address of the NAT interface) is replaced with Client A's IP address. This is what is meant by address translation.

At this point you may (should!) react by pointing out that in Section Virtualized network environment in VirtualBox it was said that VirtualBox provides NAT for the NAT interface. That is still true! What happens here is that we have two levels of NAT: one from the host-only network to the NAT interface on Server A, and another from the NAT interface on Server A to the NAT interface emulated internally by VirtualBox. The emulated NAT interface receives packets from the Server A's NAT interface and replaces the source IP address in them with the public IP address of the host[9]. At this stage, the packets can be sent over the Internet. Replies are received by the host, which then replaces the destination IP address in the packets with the IP address of the Server A's NAT interface. If your brain is spinning at this moment, take a break, read again these two paragraphs and/or grab the lab guy for more explanations.

[8] Private address blocks start with 10. or 192.168.

[9] To make things really complicated, the host IP address may be private as well, and the home route would implement a 3rd level of NAT.

Task 27: Enable SNAT on Server A

To fix the problem outlined above you need to tell Server A to do SNAT on the NAT interface. You must add the following iptables rule:

```
$IPT -t nat -A POSTROUTING  -j SNAT -o $NIF --to $NIP
```

Can you now access the Internet from Client A?

Deliverables

You are expected to provide an **individual** report in PDF format, where you detail how you solved each task included in the lab guide. Provide screenshots as proof that your commands/changes work as intended. In tasks that required you to make configuration changes or modify firewall rules, list the modifications you made and explain where you made them (on which VM and in what file).

Your report must be written in English. On the first page make sure you include your full name (as it appears in Canvas), your personal number, and your e-mail address.

In addition to the report, upload also your firewall script, as it is after solving Task 27.