

Task1

For task 1, we used Ubuntu 16.04 LTS VMs for the client and the server. The server has apache installed as the webserver and open ssh configured to test our SSH in later tasks. The Client is only connected to the “ClientNetwork”, Server is only connected to the “ServerNetwork”, FirewallA is connected to the “ClientNetwork” and the “Internet”, and FirewallB is connected to the “ServerNetwork” and the “Internet”.

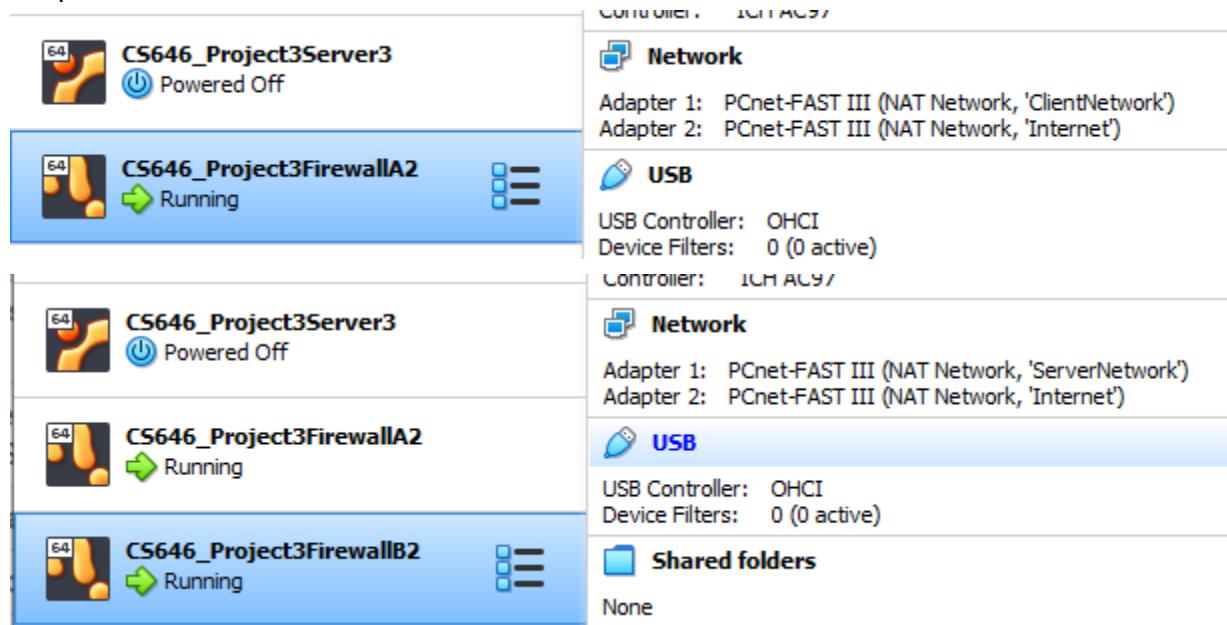
ClientNetwork	10.47.1.0/24
ServerNetwork	10.47.2.0/24
Internet	10.47.3.0/24

The IP addresses are statically assigned as follows:

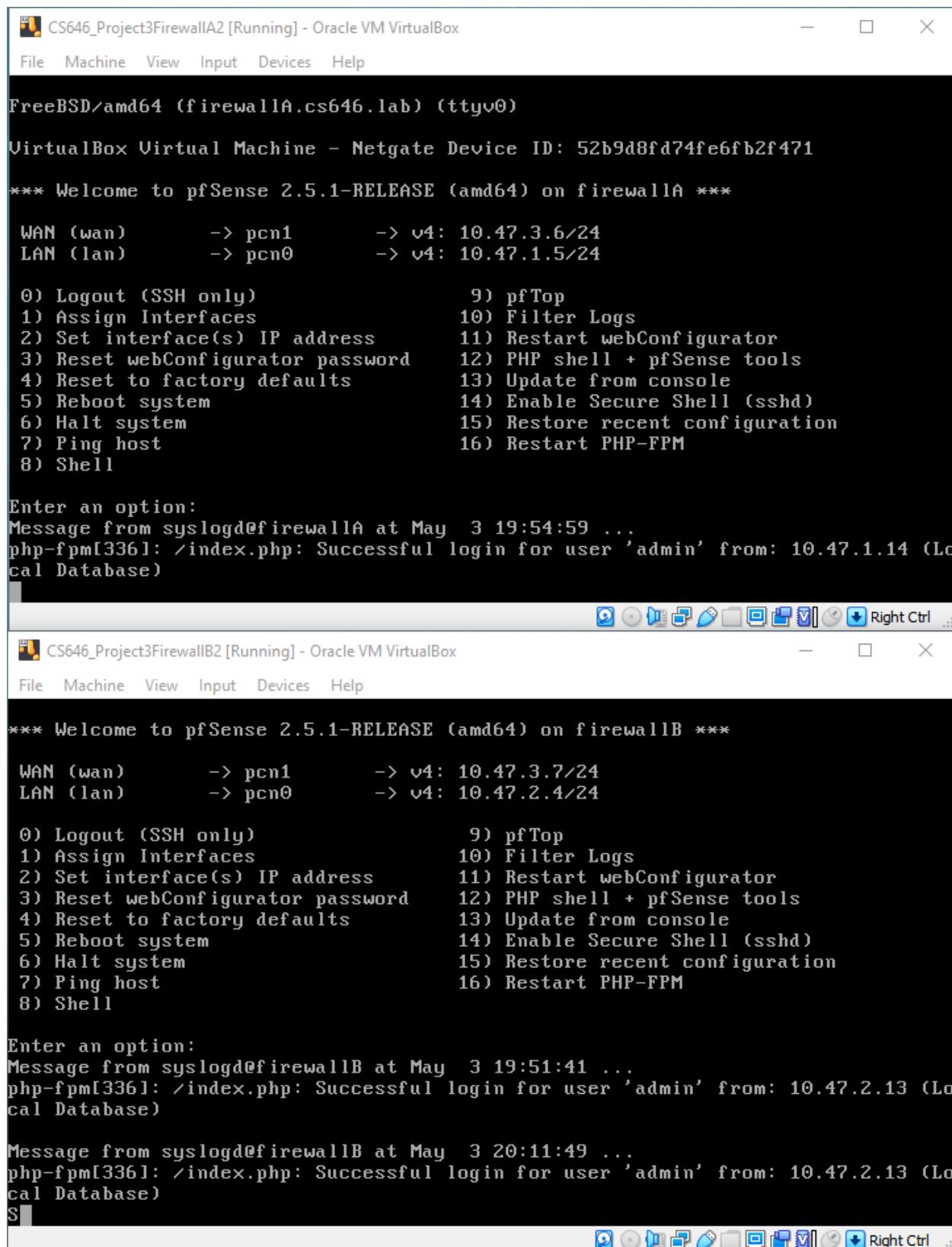
Client	10.47.1.14
Server	10.47.2.13
FirewallA LAN Interface	10.47.1.5
FirewallA WAN Interface	10.47.3.6
FirewallB LAN Interface	10.47.2.4
FirewallB WAN Interface	10.47.3.7

Below are screenshots displaying the IP addresses of the devices used.

Adapters FirewallA and FirewallB are connected to.



IP addresses of FirewallA and FirewallB



The image shows two Oracle VM VirtualBox windows side-by-side. Both windows have a title bar "CS646_Project3FirewallA2 [Running] - Oracle VM VirtualBox" and a menu bar "File Machine View Input Devices Help". The left window displays the configuration for "firewallA.cs646.lab" and the right window displays the configuration for "firewallB.cs646.lab". Both windows show a command-line interface for pfSense 2.5.1-RELEASE (amd64) with various configuration options and log messages from syslogd.

firewallA.cs646.lab Configuration:

```
FreeBSD/amd64 (firewallA.cs646.lab) (ttyv0)
VirtualBox Virtual Machine - Netgate Device ID: 52b9d8fd74fe6fb2f471
*** Welcome to pfSense 2.5.1-RELEASE (amd64) on firewallA ***
WAN (wan)      -> pcn1      -> v4: 10.47.3.6/24
LAN (lan)      -> pcn0      -> v4: 10.47.1.5/24

0) Logout (SSH only)          9) pfTop
1) Assign Interfaces          10) Filter Logs
2) Set interface(s) IP address 11) Restart webConfigurator
3) Reset webConfigurator password 12) PHP shell + pfSense tools
4) Reset to factory defaults   13) Update from console
5) Reboot system               14) Enable Secure Shell (sshd)
6) Halt system                 15) Restore recent configuration
7) Ping host                   16) Restart PHP-FPM
8) Shell

Enter an option:
Message from syslogd@firewallA at May  3 19:54:59 ...
php-fpm[336]: /index.php: Successful login for user 'admin' from: 10.47.1.14 (Local Database)
```

firewallB.cs646.lab Configuration:

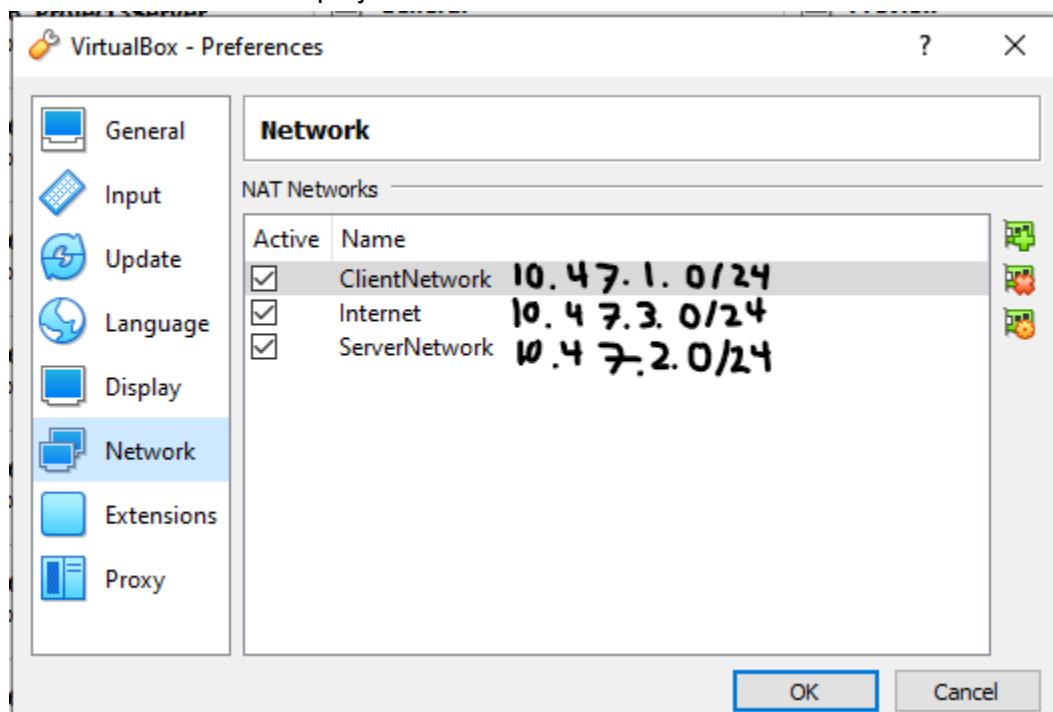
```
*** Welcome to pfSense 2.5.1-RELEASE (amd64) on firewallB ***
WAN (wan)      -> pcn1      -> v4: 10.47.3.7/24
LAN (lan)      -> pcn0      -> v4: 10.47.2.4/24

0) Logout (SSH only)          9) pfTop
1) Assign Interfaces          10) Filter Logs
2) Set interface(s) IP address 11) Restart webConfigurator
3) Reset webConfigurator password 12) PHP shell + pfSense tools
4) Reset to factory defaults   13) Update from console
5) Reboot system               14) Enable Secure Shell (sshd)
6) Halt system                 15) Restore recent configuration
7) Ping host                   16) Restart PHP-FPM
8) Shell

Enter an option:
Message from syslogd@firewallB at May  3 19:51:41 ...
php-fpm[336]: /index.php: Successful login for user 'admin' from: 10.47.2.13 (Local Database)

Message from syslogd@firewallB at May  3 20:11:49 ...
php-fpm[336]: /index.php: Successful login for user 'admin' from: 10.47.2.13 (Local Database)
```

Virtual Networks for the project.



FirewallA's pfSense webpage

A screenshot of the pfSense Status / Dashboard page. The left sidebar includes icons for Firewall, Services, Status, Diagnostics, Help, and a search bar. The main content area has several panels: 'System Information' (listing Name: firewallA.cs646.lab, User: admin@10.47.1.14, System: VirtualBox Virtual Machine, BIOS: innotek GmbH, Version: 2.5.1-RELEASE, CPU Type: Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz, Kernel PTI: Disabled, MDS Mitigation: Inactive, Uptime: 00 Hour 22 Minutes 35 Seconds, Current date/time: Mon May 3 20:16:08 UTC 2021, DNS server(s): 127.0.0.1, Last config change: Mon May 3 18:25:49 UTC 2021, State table size: 0% (77/403000), MBUF Usage: 0% (816/1000000), Load average: 0.67, 0.79, 0.65, CPU usage: Retrieving CPU data, Memory usage: low), 'Netgate Services And Support' (Contract type: Community Support Only), 'NETGATE AND pfSense COMMUNITY SUPPORT RESOURCES' (describing Netgate support resources), and 'Interfaces' (listing WAN (10.47.3.6) and LAN (10.47.1.5) with 100baseTX connections). A red box highlights the 'Community Support Only' contract type.

Raphael Archer, Ryan Rozanitis
CS 646 Project 3 Deliverable

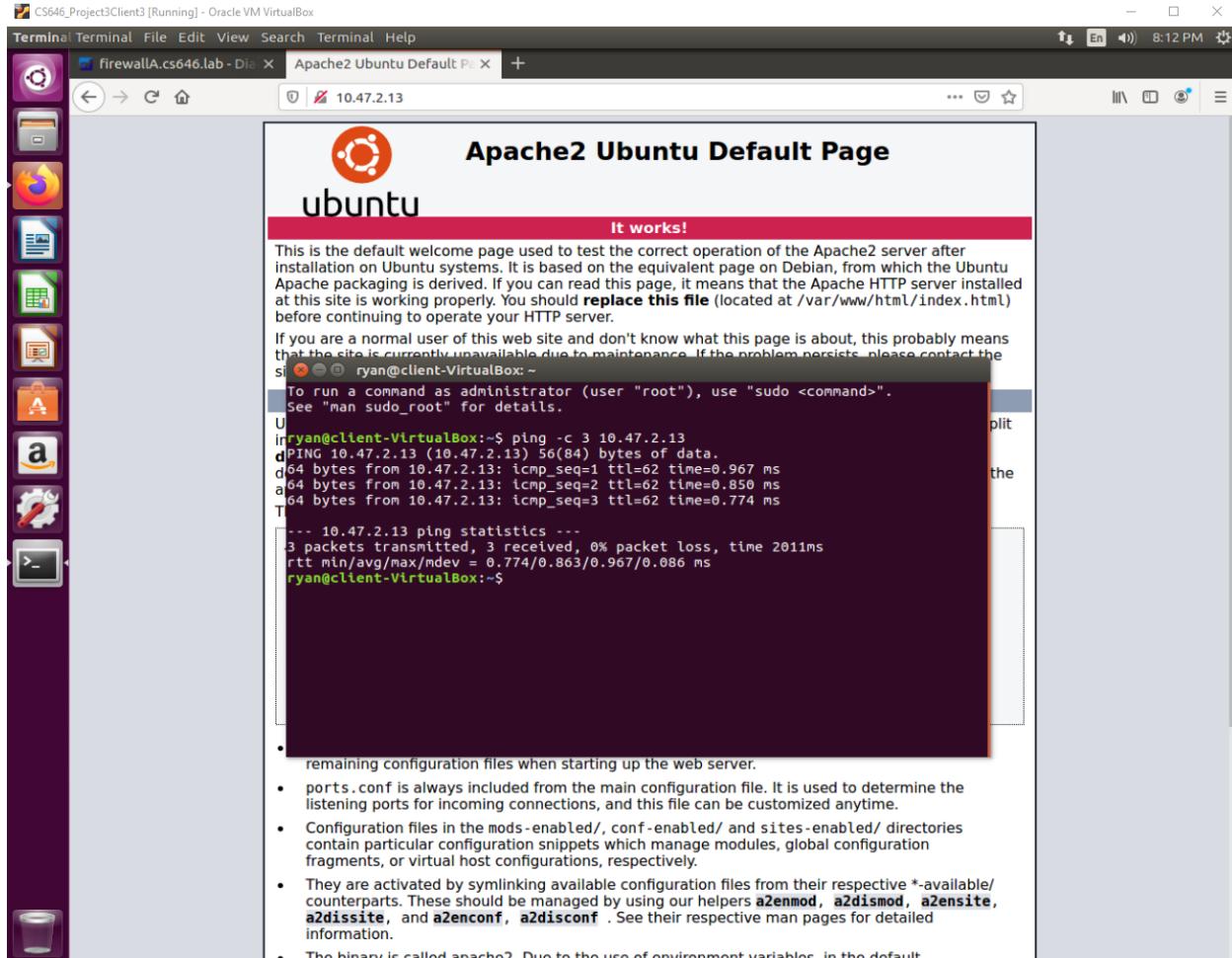
FirewallB's pfSense webpage

The screenshot shows the pfSense Status / Dashboard page. The left sidebar contains icons for various system management tools. The main content area is divided into several sections:

- System Information:** Includes fields for Name (firewallB.cs646.lab), User (admin@10.47.2.13 (Local Database)), System (VirtualBox Virtual Machine, Netgate Device ID: 156fc188ded9eed65883), BIOS (Vendor: innotek GmbH, Version: VirtualBox, Release Date: Fri Dec 1 2006), Version (2.5.1-RELEASE (amd64) built on Mon Apr 12 07:50:14 EDT 2021 FreeBSD 12.2-STABLE), CPU Type (Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz AES-NI CPU Crypto: Yes (inactive)), Kernel PTI (Disabled), MDS Mitigation (Inactive), Uptime (01 Hour 55 Minutes 08 Seconds), Current date/time (Mon May 3 20:16:34 UTC 2021), DNS server(s) (127.0.0.1), Last config change (Mon May 3 18:25:12 UTC 2021), State table size (0% (36/403000) Show states), MBUF Usage (0% (906/1000000)), Load average (0.80, 0.88, 0.77), and CPU usage (progress bar).
- Netgate Services And Support:** Shows Contract type (Community Support, Community Support Only). It also includes a section titled "NETGATE AND pfSense COMMUNITY SUPPORT RESOURCES" with links to Upgrade Your Support, Netgate Global Support FAQ, Official pfSense Training by Netgate, Netgate Professional Services, and Visit Netgate.com.
- Interfaces:** Lists two interfaces: WAN (100baseTX <full-duplex>, IP 10.47.3.7) and LAN (100baseTX <full-duplex>, IP 10.47.2.4).

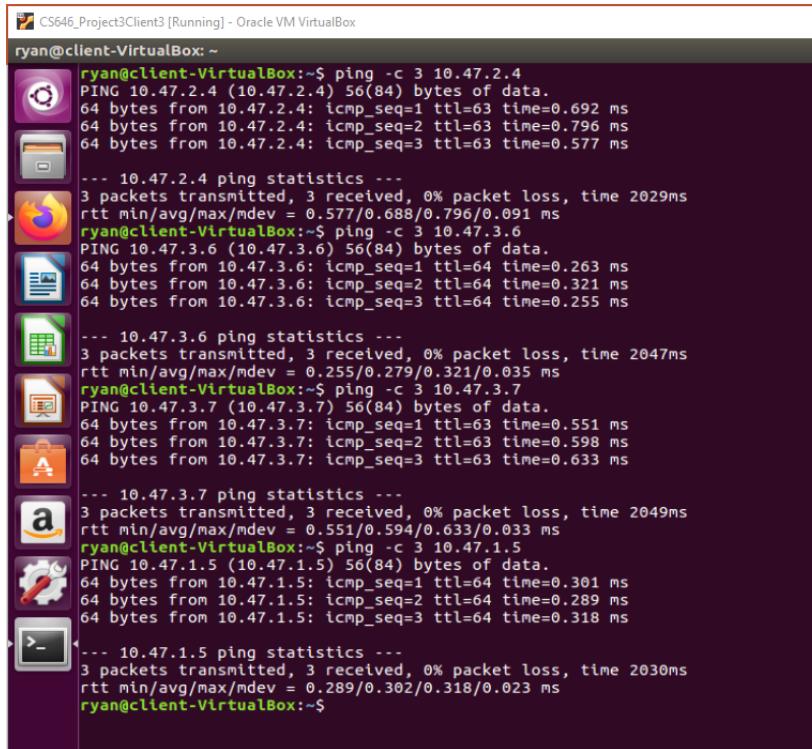
Raphael Archer, Ryan Rozanitis
CS 646 Project 3 Deliverable

Ping test to web server from client. You can see in the top left corner we are on device CS646_Project3Client3. In the background is the working web page on 10.47.2.13. The terminal shows the ping test to 10.47.2.13.



Raphael Archer, Ryan Rozanitis
CS 646 Project 3 Deliverable

More ping tests from the Client to all.



```
ryan@client-VirtualBox:~$ ping -c 3 10.47.2.4
PING 10.47.2.4 (10.47.2.4) 56(84) bytes of data.
64 bytes from 10.47.2.4: icmp_seq=1 ttl=63 time=0.692 ms
64 bytes from 10.47.2.4: icmp_seq=2 ttl=63 time=0.796 ms
64 bytes from 10.47.2.4: icmp_seq=3 ttl=63 time=0.577 ms

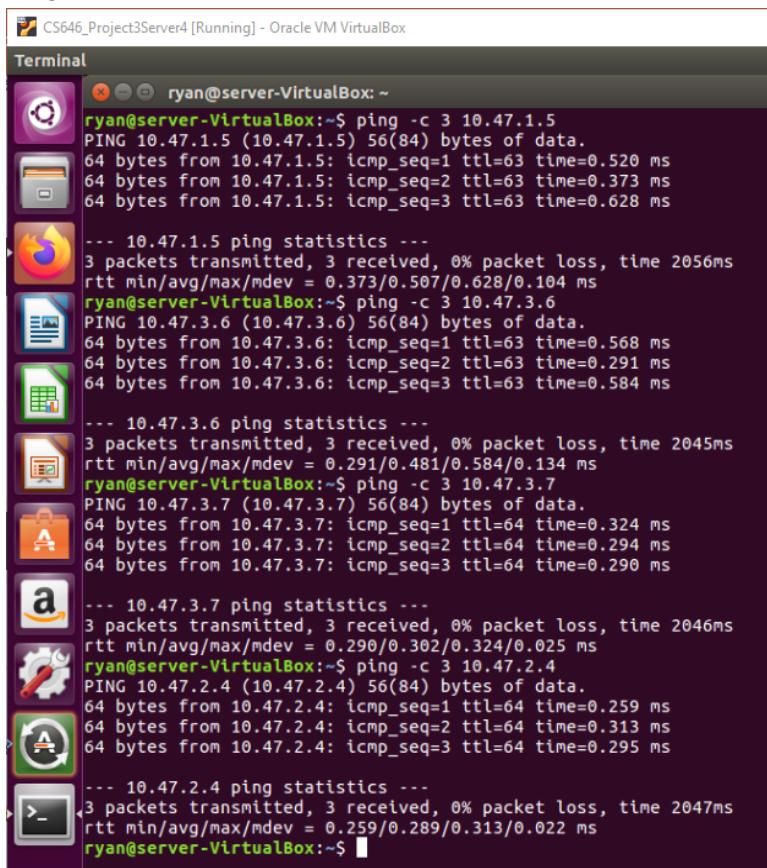
--- 10.47.2.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2029ms
rtt min/avg/max/mdev = 0.577/0.688/0.796/0.091 ms
ryan@client-VirtualBox:~$ ping -c 3 10.47.3.6
PING 10.47.3.6 (10.47.3.6) 56(84) bytes of data.
64 bytes from 10.47.3.6: icmp_seq=1 ttl=64 time=0.263 ms
64 bytes from 10.47.3.6: icmp_seq=2 ttl=64 time=0.321 ms
64 bytes from 10.47.3.6: icmp_seq=3 ttl=64 time=0.255 ms

--- 10.47.3.6 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2047ms
rtt min/avg/max/mdev = 0.255/0.279/0.321/0.035 ms
ryan@client-VirtualBox:~$ ping -c 3 10.47.3.7
PING 10.47.3.7 (10.47.3.7) 56(84) bytes of data.
64 bytes from 10.47.3.7: icmp_seq=1 ttl=63 time=0.551 ms
64 bytes from 10.47.3.7: icmp_seq=2 ttl=63 time=0.598 ms
64 bytes from 10.47.3.7: icmp_seq=3 ttl=63 time=0.633 ms

--- 10.47.3.7 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2049ms
rtt min/avg/max/mdev = 0.551/0.594/0.633/0.033 ms
ryan@client-VirtualBox:~$ ping -c 3 10.47.1.5
PING 10.47.1.5 (10.47.1.5) 56(84) bytes of data.
64 bytes from 10.47.1.5: icmp_seq=1 ttl=64 time=0.301 ms
64 bytes from 10.47.1.5: icmp_seq=2 ttl=64 time=0.289 ms
64 bytes from 10.47.1.5: icmp_seq=3 ttl=64 time=0.318 ms

--- 10.47.1.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2030ms
rtt min/avg/max/mdev = 0.289/0.302/0.318/0.023 ms
ryan@client-VirtualBox:~$
```

Ping test from Server to all



```
ryan@server-VirtualBox:~$ ping -c 3 10.47.1.5
PING 10.47.1.5 (10.47.1.5) 56(84) bytes of data.
64 bytes from 10.47.1.5: icmp_seq=1 ttl=63 time=0.520 ms
64 bytes from 10.47.1.5: icmp_seq=2 ttl=63 time=0.373 ms
64 bytes from 10.47.1.5: icmp_seq=3 ttl=63 time=0.628 ms

--- 10.47.1.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2056ms
rtt min/avg/max/mdev = 0.373/0.507/0.628/0.104 ms
ryan@server-VirtualBox:~$ ping -c 3 10.47.3.6
PING 10.47.3.6 (10.47.3.6) 56(84) bytes of data.
64 bytes from 10.47.3.6: icmp_seq=1 ttl=63 time=0.568 ms
64 bytes from 10.47.3.6: icmp_seq=2 ttl=63 time=0.291 ms
64 bytes from 10.47.3.6: icmp_seq=3 ttl=63 time=0.584 ms

--- 10.47.3.6 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2045ms
rtt min/avg/max/mdev = 0.291/0.481/0.584/0.134 ms
ryan@server-VirtualBox:~$ ping -c 3 10.47.3.7
PING 10.47.3.7 (10.47.3.7) 56(84) bytes of data.
64 bytes from 10.47.3.7: icmp_seq=1 ttl=64 time=0.324 ms
64 bytes from 10.47.3.7: icmp_seq=2 ttl=64 time=0.294 ms
64 bytes from 10.47.3.7: icmp_seq=3 ttl=64 time=0.290 ms

--- 10.47.3.7 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2046ms
rtt min/avg/max/mdev = 0.290/0.302/0.324/0.025 ms
ryan@server-VirtualBox:~$ ping -c 3 10.47.2.4
PING 10.47.2.4 (10.47.2.4) 56(84) bytes of data.
64 bytes from 10.47.2.4: icmp_seq=1 ttl=64 time=0.259 ms
64 bytes from 10.47.2.4: icmp_seq=2 ttl=64 time=0.313 ms
64 bytes from 10.47.2.4: icmp_seq=3 ttl=64 time=0.295 ms

--- 10.47.2.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2047ms
rtt min/avg/max/mdev = 0.259/0.289/0.313/0.022 ms
ryan@server-VirtualBox:~$
```

Task2

In order to block SSH, we followed two principles. First, we wanted to block/reject the packets at the door. We don't even want them getting into the firewall if it is bound for the server on port 22. Second, we want to also guarantee that this works on both firewalls. So we start with FirewallB's WAN interface.

Below is how we captured the packets to determine what rule to make. We selected Port 22 and 80 (SSH and HTTP) so that we could make sure that the IP was the same.

The screenshot shows the 'Packet Capture Options' configuration page. Key settings include:

- Interface:** WAN
- Promiscuous:** Enabled (checkbox checked)
- Address Family:** IPv4 Only
- Protocol:** Any
- Host Address:** (empty field)
- Port:** 22 | 80

We found that 10.47.3.6 was the source and 10.47.2.13 was the destination. So, we make a firewall rule on the WAN interface, **blocking** traffic with that source and destination on port 22.

Rules (Drag to Change Order)											Actions
	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input type="checkbox"/>	0 /840 B	IPv4 TCP	10.47.3.6	*	10.47.2.13	22 (SSH)	*	none		BlockedSSHfromClient	

We decided to specifically block SSH to only the server. We don't know what else this network will be used for in the future, so no need to block everything.

In the next screenshot, you can see the packet capture from this. We tried to SSH a few times and refreshed the default apache webpage to prove it worked.

Packets Captured

```
21:03:30.146049 IP 10.47.3.6.3129 > 10.47.2.13.80: tcp 0
21:03:30.146479 IP 10.47.2.13.80 > 10.47.3.6.3129: tcp 0
21:03:30.147471 IP 10.47.3.6.3129 > 10.47.2.13.80: tcp 0
21:03:30.148122 IP 10.47.3.6.3129 > 10.47.2.13.80: tcp 447
21:03:30.148404 IP 10.47.2.13.80 > 10.47.3.6.3129: tcp 0
21:03:30.148898 IP 10.47.2.13.80 > 10.47.3.6.3129: tcp 1448
21:03:30.148904 IP 10.47.2.13.80 > 10.47.3.6.3129: tcp 1448
21:03:30.148908 IP 10.47.2.13.80 > 10.47.3.6.3129: tcp 629
21:03:30.149397 IP 10.47.3.6.3129 > 10.47.2.13.80: tcp 0
21:03:30.180927 IP 10.47.3.6.3129 > 10.47.2.13.80: tcp 401
21:03:30.181959 IP 10.47.2.13.80 > 10.47.3.6.3129: tcp 180
21:03:30.224612 IP 10.47.3.6.3129 > 10.47.2.13.80: tcp 0
21:03:33.204879 IP 10.47.3.6.25281 > 10.47.2.13.22: tcp 0
21:03:33.524711 IP 10.47.3.6.35560 > 10.47.2.13.22: tcp 0
21:03:34.228048 IP 10.47.3.6.25281 > 10.47.2.13.22: tcp 0
21:03:35.094226 IP 10.47.2.13.80 > 10.47.3.6.3129: tcp 0
21:03:35.095201 IP 10.47.3.6.3129 > 10.47.2.13.80: tcp 0
21:03:35.095520 IP 10.47.2.13.80 > 10.47.3.6.3129: tcp 0
```

And here is a screenshot of the terminal on the Client. We first show that it was possible to connect before the firewall rule is added. We also explicitly chose to block here. On the WAN, we don't even want a reject, only block.

```
ryan@client-VirtualBox:~$ ssh ryan@10.47.2.13
ryan@10.47.2.13's password:
Welcome to Ubuntu 16.04.7 LTS (GNU/Linux 4.15.0-112-generic x86_64)

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

188 packages can be updated.
154 updates are security updates.

Last login: Mon May  3 20:57:20 2021 from 10.47.3.6
ryan@server-VirtualBox:~$ exit
logout
Connection to 10.47.2.13 closed.
ryan@client-VirtualBox:~$ ssh ryan@10.47.2.13
^Z
[1]+  Stopped                  ssh ryan@10.47.2.13
ryan@client-VirtualBox:~$ ssh ryan@10.47.2.13
^Z
[2]+  Stopped                  ssh ryan@10.47.2.13
ryan@client-VirtualBox:~$
```

Raphael Archer, Ryan Rozanitis
CS 646 Project 3 Deliverable

Next, we configure a Firewall Rule on FirewallA's LAN interface. This time, the source is 10.47.1.14 and destination is 10.47.2.13. This is because we have not gone through the firewall yet, we are still in the LAN. In the previous example, the packets were coming through the "Internet" (10.47.3.0/24 subnet).

States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
0 / 0 B	*	*	*	LAN Address	80	*	*		Anti-Lockout Rule	
0 / 0 B	IPv4 TCP	10.47.1.14	*	10.47.2.13	22 (SSH)	*	none			

Instead of blocking here, we politely reject. Clients within the LAN shouldn't be sitting around waiting for a response. We let them know so that we don't waste our clients time.

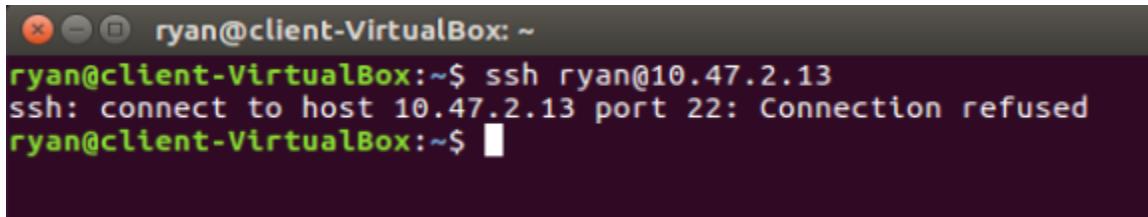
We then test it out.

Below we have our packet capture. We only tried to SSH once, to which we saw the connection was refused. In our packet capture, we highlighted the line where SSH was blocked. The 0 signifies nothing passed through in this case.

We also have some communication between our firewallA pfSense webpage and the client itself, which is signified by any packet that contains a 10.47.1.5. The packets that contain 10.47.2.13.80 are our packets sent to the apache web server across http.

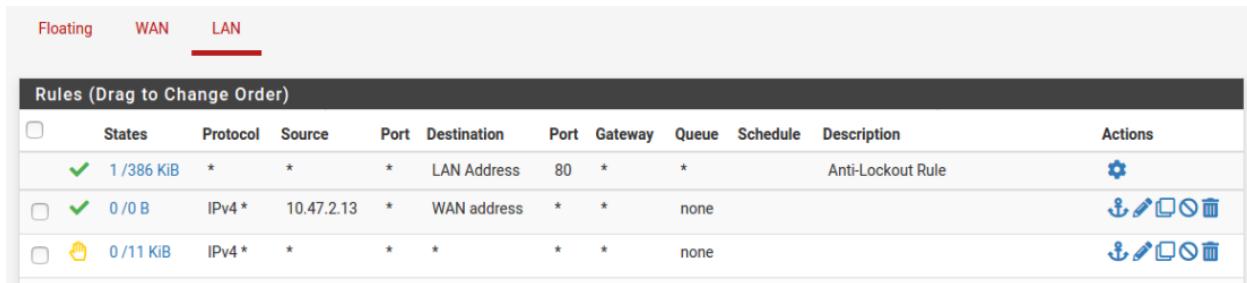
Packets Captured	
21:10:43.315850 IP 10.47.1.5.80 > 10.47.1.14.58572: tcp 1448	
21:10:43.315867 IP 10.47.1.5.80 > 10.47.1.14.58572: tcp 1448	
21:10:43.315870 IP 10.47.1.5.80 > 10.47.1.14.58572: tcp 1448	
21:10:43.316168 IP 10.47.1.14.58572 > 10.47.1.5.80: tcp 0	
21:10:43.316178 IP 10.47.1.5.80 > 10.47.1.14.58572: tcp 1448	
21:10:43.316184 IP 10.47.1.5.80 > 10.47.1.14.58572: tcp 1448	
21:10:43.316187 IP 10.47.1.5.80 > 10.47.1.14.58572: tcp 1448	
21:10:43.316192 IP 10.47.1.5.80 > 10.47.1.14.58572: tcp 59	
21:10:43.317893 IP 10.47.1.14.58572 > 10.47.1.5.80: tcp 0	
21:10:45.736221 IP 10.47.1.14.54210 > 10.47.2.13.22: tcp 0	
21:10:45.736256 IP 10.47.2.13.22 > 10.47.1.14.54216: tcp 0	
21:10:49.069570 IP 10.47.1.14.45218 > 10.47.2.13.80: tcp 0	
21:10:49.070104 IP 10.47.2.13.80 > 10.47.1.14.45218: tcp 0	
21:10:49.070296 IP 10.47.1.14.45218 > 10.47.2.13.80: tcp 0	
21:10:49.070365 IP 10.47.1.14.45218 > 10.47.2.13.80: tcp 447	
21:10:49.070724 IP 10.47.2.13.80 > 10.47.1.14.45218: tcp 0	
21:10:49.071151 IP 10.47.2.13.80 > 10.47.1.14.45218: tcp 1448	
21:10:49.071157 IP 10.47.2.13.80 > 10.47.1.14.45218: tcp 1448	
21:10:49.071163 IP 10.47.2.13.80 > 10.47.1.14.45218: tcp 629	
21:10:49.071305 IP 10.47.1.14.45218 > 10.47.2.13.80: tcp 0	
21:10:49.100242 IP 10.47.1.14.45218 > 10.47.2.13.80: tcp 401	

In this picture we see the Connection refused message. This important distinction in our rules proves to us that this is getting caught at the LAN interface on FirewallA.



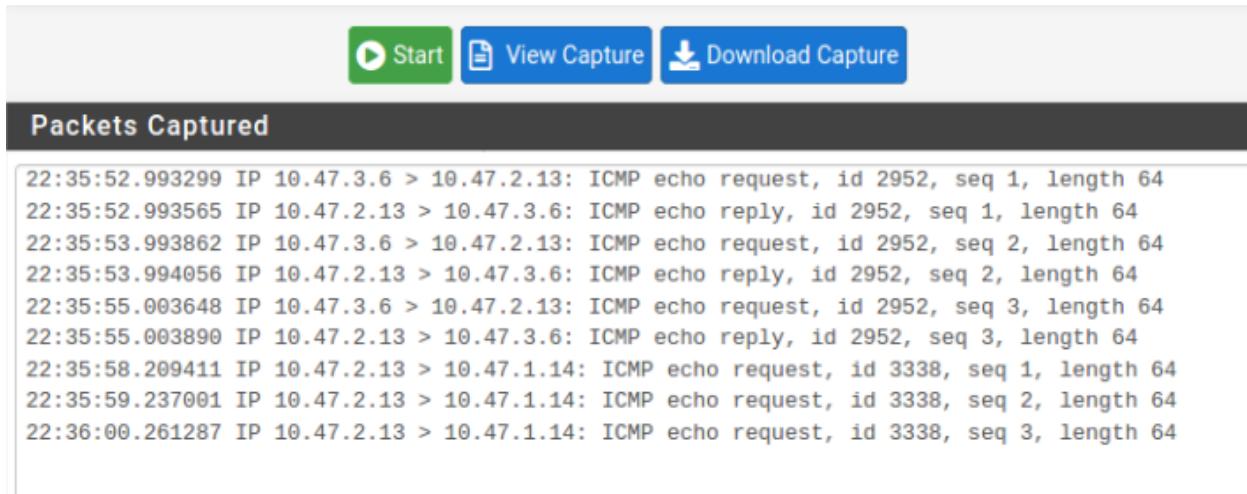
```
ryan@client-VirtualBox:~$ ssh ryan@10.47.2.13
ssh: connect to host 10.47.2.13 port 22: Connection refused
ryan@client-VirtualBox:~$
```

Next, we want to stop unsolicited messages from the server to the client. Our thinking on this was to block it at the LAN interface of FirewallB. So we added a Firewall Rule on the LAN Interface that rejects all traffic. Just above that, we add another rule that states to allow any IPv4 packets from the network 10.47.2.0



Rules (Drag to Change Order)											Actions
	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input checked="" type="checkbox"/>	1 /386 KIB	*	*	*	LAN Address	80	*	*		Anti-Lockout Rule	
<input checked="" type="checkbox"/>	0 /0 B	IPv4 *	10.47.2.13	*	WAN address	*	*		none		 
<input checked="" type="checkbox"/>	0 /11 KIB	IPv4 *	*	*	*	*	*	*	none		 

To test this, we went to the packet capture tab and only looked for ping requests. In theory, ICMP echo requests from the client to the server should pass through the LAN, but ICMP echo requests from the server to anyone will be blocked. Below are the results of our packet capture with this rule.



Packets Captured	
22:35:52.993299 IP 10.47.3.6 > 10.47.2.13: ICMP echo request, id 2952, seq 1, length 64	22:35:52.993565 IP 10.47.2.13 > 10.47.3.6: ICMP echo reply, id 2952, seq 1, length 64

22:35:53.993862 IP 10.47.3.6 > 10.47.2.13: ICMP echo request, id 2952, seq 2, length 64
22:35:53.994056 IP 10.47.2.13 > 10.47.3.6: ICMP echo reply, id 2952, seq 2, length 64
22:35:55.003648 IP 10.47.3.6 > 10.47.2.13: ICMP echo request, id 2952, seq 3, length 64
22:35:55.003890 IP 10.47.2.13 > 10.47.3.6: ICMP echo reply, id 2952, seq 3, length 64
22:35:58.209411 IP 10.47.2.13 > 10.47.1.14: ICMP echo request, id 3338, seq 1, length 64
22:35:59.237001 IP 10.47.2.13 > 10.47.1.14: ICMP echo request, id 3338, seq 2, length 64
22:36:00.261287 IP 10.47.2.13 > 10.47.1.14: ICMP echo request, id 3338, seq 3, length 64

As you can see, the ICMP packets from 10.47.3.6 are allowed to pass through. These pings are actually from the client, 10.47.1.14. However, when we try to ping from our server to the client, we only see ICMP echo requests. This is due to our firewall rule only allowing packets out that use a WAN address, or came in from the WAN first and therefore were solicited.

Raphael Archer, Ryan Rozanitis
CS 646 Project 3 Deliverable

As a last little detail, we just wanted to clean up our firewall rules a bit. The final firewall rulesets for Task 2 is contained in the screenshot below

FirewallA WAN final ruleset. We decided to only allow TCP, UDP and ICMP traffic. Seems safe for the time being.

Rules (Drag to Change Order)											
	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input type="checkbox"/>	✓ 0 /4.40 MiB	IPv4 TCP/UDP	10.47.3.0/24	*	10.47.1.0/24	*	*	none			
<input type="checkbox"/>	✓ 0 /0 B	IPv4 ICMP any	10.47.3.0/24	*	10.47.1.0/24	*	*	none			
<input type="checkbox"/>	✗ 0 /43 KiB	IPv4 *	*	*	*	*	*	*	none		

Add Add Delete Save

Firewall LAN final ruleset. Here, we are preventing SSH first. That stops the client from using SSH to connect to the server. The final rule is to not allow just anything to pass through the LAN interface. The other two were default rules added by pfSense before any modification.

Rules (Drag to Change Order)											
	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input checked="" type="checkbox"/>	✓ 1 /248 KiB	*	*	*	LAN Address	80	*	*		Anti-Lockout Rule	
<input type="checkbox"/>	✗ 0 /360 B	IPv4 TCP	10.47.1.14	*	10.47.2.13	22 (SSH)	*	none			
<input type="checkbox"/>	✓ 0 /75 KiB	IPv4 *	LAN net	*	*	*	*	none		Default allow LAN to any rule	
<input type="checkbox"/>	✓ 0 /0 B	IPv6 *	LAN net	*	*	*	*	none		Default allow LAN IPv6 to any rule	
<input type="checkbox"/>	✗ 0 /114 KiB	IPv4 *	*	*	*	*	*	none			

Add Add Delete Save

FirewallB WAN final ruleset. This one is similar to the FirewallA WAN ruleset in that we decided to only allow TCP, UDP and ICMP traffic. The only difference is the first rule, which is blocking any SSH request from 10.47.3.6, where our clients traffic will come from, to 10.47.2.13, our server.

Rules (Drag to Change Order)											
	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input checked="" type="checkbox"/>	✗ 0 /0 B	IPv4 TCP	10.47.3.6	*	10.47.2.13	22 (SSH)	*	none		BlockedSSHfromClient	
<input type="checkbox"/>	✓ 0 /4 KiB	IPv4 TCP/UDP	10.47.3.0/24	*	10.47.2.0/24	*	*	none			
<input type="checkbox"/>	✓ 0 /20 KiB	IPv4 ICMP any	10.47.3.0/24	*	10.47.2.0/24	*	*	none			
<input type="checkbox"/>	✗ 0 /2.37 MiB	IPv4 *	*	*	*	*	*	*	none		

Add Add Delete Save

Raphael Archer, Ryan Rozanitis
CS 646 Project 3 Deliverable

FirewallIB LAN final ruleset. We started by turning off the default rules. We don't want the server going out and doing anything without being solicited. The first rule is required by pfSense. The second rule is our "only respond to solicited traffic" rule. Third is to reject any other traffic. Currently this is ok because this subnet only contains the server.

Rules (Drag to Change Order)											Actions
	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input checked="" type="checkbox"/>	✓ 2 /510 Kib	*	*	*	LAN Address	80	*	*		Anti-Lockout Rule	
<input type="checkbox"/>	✓ 0 /0 B	IPv4 *	10.47.2.13	*	WAN address	*	*	none			
<input type="checkbox"/>	✋ 0 /11 Kib	IPv4 *	*	*	*	*	*	none			
<input type="checkbox"/>	✓ 0 /0 B	IPv4 *	LAN net	*	*	*	*	none		Default allow LAN to any rule	
<input type="checkbox"/>	✓ 0 /0 B	IPv6 *	LAN net	*	*	*	*	none		Default allow LAN IPv6 to any rule	

Add Add Delete Save Separator

Task3

To configure our client to use the external IP of firewallA, we create the following NAT rule.

Interface	Source	Source Port	Destination	Destination Port	NAT Address	NAT Port	Static Port	Description	Actions
WAN	10.47.1.14/32	*	*	*	WAN address	*			
WAN	127.0.0.0/8	*	*	500 (ISAKMP)	WAN	*	✓	Auto created rule for ISAKMP - localhost to WAN	

Any Outbound connections will be passed through this list of NAT rules. We decided to force any packets from source 10.47.1.14 to be translated to the WAN interface IP address, 10.47.3.6.

To test this, we do 4 packet traces (one at each interface). During the ICMP-only packet trace, we first do a 3 ping test from our client (10.47.1.14) to our web server (10.47.2.13) with Outbound NAT rules disabled. We then turn on Manual Outbound NAT rules, which uses our previously configured NAT rule, and do another 3 ping test. Here are the results of the LAN and WAN interface on firewallA. The LAN and WAN are identical since they

As packets pass out of the WAN interface, the WAN IP address will take the place of 10.47.1.14.

```

11:15:36.626563 IP 10.47.1.14 > 10.47.2.13: ICMP echo request, id 2723, seq 1, length 64
11:15:36.627246 IP 10.47.2.13 > 10.47.1.14: ICMP echo reply, id 2723, seq 1, length 64
11:15:37.627456 IP 10.47.1.14 > 10.47.2.13: ICMP echo request, id 2723, seq 2, length 64
11:15:37.628012 IP 10.47.2.13 > 10.47.1.14: ICMP echo reply, id 2723, seq 2, length 64
11:15:38.633890 IP 10.47.1.14 > 10.47.2.13: ICMP echo request, id 2723, seq 3, length 64
11:15:38.634481 IP 10.47.2.13 > 10.47.1.14: ICMP echo reply, id 2723, seq 3, length 64
11:15:47.614263 IP 10.47.1.14 > 10.47.2.13: ICMP echo request, id 2727, seq 1, length 64
11:15:47.614820 IP 10.47.2.13 > 10.47.1.14: ICMP echo reply, id 2727, seq 1, length 64
11:15:48.617825 IP 10.47.1.14 > 10.47.2.13: ICMP echo request, id 2727, seq 2, length 64
11:15:48.618301 IP 10.47.2.13 > 10.47.1.14: ICMP echo reply, id 2727, seq 2, length 64
11:15:49.641718 IP 10.47.1.14 > 10.47.2.13: ICMP echo request, id 2727, seq 3, length 64
11:15:49.642289 IP 10.47.2.13 > 10.47.1.14: ICMP echo reply, id 2727, seq 3, length 64

```

On both interfaces for FirewallB, we can see the difference. Our second set of Pings has been modified to show 10.47.3.6 instead of 10.47.1.14 for our source IP. The first 6 ICMP echo messages are with no Outbound NAT rules and the second three are the pings just after using the manual Outbound NAT rules

Packets Captured

```
11:15:36.233707 IP 10.47.1.14 > 10.47.2.13: ICMP echo request, id 2723, seq 1, length 64
11:15:36.234023 IP 10.47.2.13 > 10.47.1.14: ICMP echo reply, id 2723, seq 1, length 64
11:15:37.234322 IP 10.47.1.14 > 10.47.2.13: ICMP echo request, id 2723, seq 2, length 64
11:15:37.234610 IP 10.47.2.13 > 10.47.1.14: ICMP echo reply, id 2723, seq 2, length 64
11:15:38.240775 IP 10.47.1.14 > 10.47.2.13: ICMP echo request, id 2723, seq 3, length 64
11:15:38.241103 IP 10.47.2.13 > 10.47.1.14: ICMP echo reply, id 2723, seq 3, length 64
11:15:47.220956 IP 10.47.3.6 > 10.47.2.13: ICMP echo request, id 7086, seq 1, length 64
11:15:47.221222 IP 10.47.2.13 > 10.47.3.6: ICMP echo reply, id 7086, seq 1, length 64
11:15:48.224273 IP 10.47.3.6 > 10.47.2.13: ICMP echo request, id 7086, seq 2, length 64
11:15:48.224493 IP 10.47.2.13 > 10.47.3.6: ICMP echo reply, id 7086, seq 2, length 64
11:15:49.248488 IP 10.47.3.6 > 10.47.2.13: ICMP echo request, id 7086, seq 3, length 64
11:15:49.248785 IP 10.47.2.13 > 10.47.3.6: ICMP echo reply, id 7086, seq 3, length 64
```

We decided to test this again by connecting to the web server. We ran a packet trace looking for 10.47.2.13 and port 80. We turned off the Outbound NAT rules for the first hard refresh, and then turned on Manual Outbound NAT rules for the second refresh. (*By hard refresh we mean we hit CTRL+R. Without that we might get that cached page*).

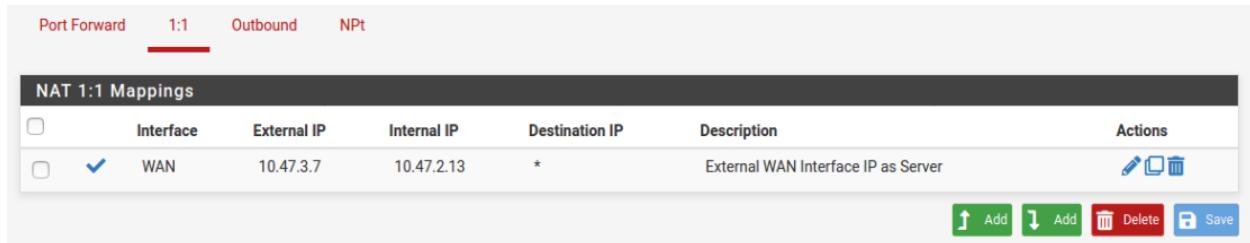
Last capture start	May 4th, 2021 11:38:02 am.
Last capture stop	May 4th, 2021 11:38:32 am.

Packets Captured

```
11:38:11.094050 IP 10.47.2.10.80 > 10.47.1.14.47282: tcp 100
11:38:11.738478 IP 10.47.1.14.47282 > 10.47.2.13.80: tcp 0
11:38:16.607136 IP 10.47.2.13.80 > 10.47.1.14.47282: tcp 0
11:38:16.607643 IP 10.47.1.14.47282 > 10.47.2.13.80: tcp 0
11:38:16.607823 IP 10.47.2.13.80 > 10.47.1.14.47282: tcp 0
11:38:22.498388 IP 10.47.3.6.51351 > 10.47.2.13.80: tcp 0
11:38:22.498715 IP 10.47.2.13.80 > 10.47.3.6.51351: tcp 0
11:38:22.499238 IP 10.47.3.6.51351 > 10.47.2.13.80: tcp 0
11:38:22.500434 IP 10.47.3.6.51351 > 10.47.2.13.80: tcp 447
11:38:22.500723 IP 10.47.2.13.80 > 10.47.3.6.51351: tcp 0
11:38:22.501141 IP 10.47.2.13.80 > 10.47.3.6.51351: tcp 1448
11:38:22.501147 IP 10.47.2.13.80 > 10.47.3.6.51351: tcp 1448
11:38:22.501150 IP 10.47.2.13.80 > 10.47.3.6.51351: tcp 629
11:38:22.501613 IP 10.47.3.6.51351 > 10.47.2.13.80: tcp 0
11:38:22.526132 IP 10.47.3.6.51351 > 10.47.2.13.80: tcp 401
11:38:22.526571 IP 10.47.2.13.80 > 10.47.3.6.51351: tcp 180
11:38:22.571020 IP 10.47.3.6.51351 > 10.47.2.13.80: tcp 0
11:38:27.438428 IP 10.47.2.13.80 > 10.47.3.6.51351: tcp 0
11:38:27.441151 IP 10.47.3.6.51351 > 10.47.2.13.80: tcp 0
11:38:27.441338 IP 10.47.2.13.80 > 10.47.3.6.51351: tcp 0
```

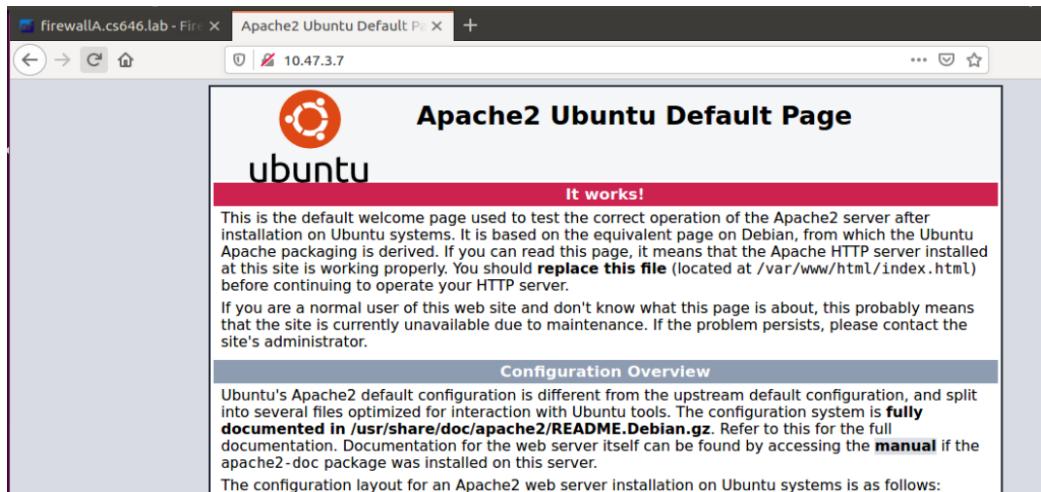
Our packet capture perfectly shows us our Outbound Rules in action. The first refresh before the 11:38:20 used 10.47.1.14 as our source IP address. After turning on the Outbound NAT Rules, we then see the all of our connection after 11:38:20 are using 10.47.3.6 instead of 10.47.1.14.

Now that we completed the outbound NAT rules for our client, we need to publish the server to an outside interface. To achieve this, we configured a 1:1 NAT rule. All this NAT rule does is tell the firewall to push packets to 10.47.3.7 to 10.47.2.13.



The screenshot shows a network configuration interface with tabs for Port Forward, 1:1, Outbound, and Npt. The 1:1 tab is selected, displaying a table titled 'NAT 1:1 Mappings'. A single entry is listed: 'WAN' as the Interface, '10.47.3.7' as the External IP, '10.47.2.13' as the Internal IP, and '*' as the Destination IP. The description is 'External WAN Interface IP as Server'. Below the table are buttons for Add, Delete, and Save.

After typing in 10.47.3.7 into the browser address bar on the client, we see the apache webpage, which is what we expected. The IP address has been properly translated to the internal IP after reaching the firewall



We prove this by doing two packet captures, one at each WAN interface. Unsurprisingly, the packet captures are identical. We performed a single hard refresh of the webpage on 10.47.3.7 after starting the packet capture. This is the packet capture on firewallA's WAN interface.

Raphael Archer, Ryan Rozanitis
CS 646 Project 3 Deliverable

Last capture start	May 4th, 2021 10:36:47 am.
Last capture stop	May 4th, 2021 10:36:55 am.

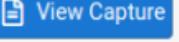
 Start  View Capture  Download Capture

Packets Captured

```
10:36:51.984201 IP 10.47.3.6.21044 > 10.47.3.7.80: tcp 0
10:36:51.984869 IP 10.47.3.7.80 > 10.47.3.6.21044: tcp 0
10:36:51.985106 IP 10.47.3.6.21044 > 10.47.3.7.80: tcp 0
10:36:51.986259 IP 10.47.3.6.21044 > 10.47.3.7.80: tcp 446
10:36:51.986842 IP 10.47.3.7.80 > 10.47.3.6.21044: tcp 0
10:36:51.987358 IP 10.47.3.7.80 > 10.47.3.6.21044: tcp 1448
10:36:51.987374 IP 10.47.3.7.80 > 10.47.3.6.21044: tcp 1448
10:36:51.987378 IP 10.47.3.7.80 > 10.47.3.6.21044: tcp 629
10:36:51.987554 IP 10.47.3.6.21044 > 10.47.3.7.80: tcp 0
10:36:52.017926 IP 10.47.3.6.21044 > 10.47.3.7.80: tcp 399
10:36:52.017790 IP 10.47.3.7.80 > 10.47.3.6.21044: tcp 180
10:36:52.059321 IP 10.47.3.6.21044 > 10.47.3.7.80: tcp 0
```

As you can see, our 10.47.3.6 firewall is communicating directly with 10.47.3.7 on port 80. It is using the firewallB's IP with port 80 to communicate with the web server. In earlier tasks, we had seen that pings to 10.47.2.13 were going straight to the internal IP directly, instead of stopping at 10.47.3.6 like we see here. Below is our identical packet trace on firewallB's WAN.

Last capture start	May 4th, 2021 10:36:48 am.
Last capture stop	May 4th, 2021 10:36:57 am.

 Start  View Capture  Download Capture

Packets Captured

```
10:36:51.713793 IP 10.47.3.6.21044 > 10.47.3.7.80: tcp 0
10:36:51.714154 IP 10.47.3.7.80 > 10.47.3.6.21044: tcp 0
10:36:51.714728 IP 10.47.3.6.21044 > 10.47.3.7.80: tcp 0
10:36:51.715835 IP 10.47.3.6.21044 > 10.47.3.7.80: tcp 446
10:36:51.716135 IP 10.47.3.7.80 > 10.47.3.6.21044: tcp 0
10:36:51.716633 IP 10.47.3.7.80 > 10.47.3.6.21044: tcp 1448
10:36:51.716640 IP 10.47.3.7.80 > 10.47.3.6.21044: tcp 1448
10:36:51.716643 IP 10.47.3.7.80 > 10.47.3.6.21044: tcp 629
10:36:51.717046 IP 10.47.3.6.21044 > 10.47.3.7.80: tcp 0
10:36:51.747075 IP 10.47.3.6.21044 > 10.47.3.7.80: tcp 399
10:36:51.747538 IP 10.47.3.7.80 > 10.47.3.6.21044: tcp 180
10:36:51.788539 IP 10.47.3.6.21044 > 10.47.3.7.80: tcp 0
```

And now we did one final packet trace, running the same test again. This time on the LAN interface on firewallB. We expect to see 10.47.3.6 communicating with 10.47.2.13. This is because Network Address Translation has already occurred. The 10.47.3.7 has been converted to 10.47.2.13 and is communicating with the web server.

Raphael Archer, Ryan Rozanitis
CS 646 Project 3 Deliverable

Last capture start	May 4th, 2021 10:43:13 am.
Last capture stop	May 4th, 2021 10:43:22 am.

 Start  View Capture  Download Capture

Packets Captured

```
10:43:18.803808 IP 10.47.3.6.27119 > 10.47.2.13.80: tcp 0
10:43:18.804118 IP 10.47.2.13.80 > 10.47.3.6.27119: tcp 0
10:43:18.804657 IP 10.47.3.6.27119 > 10.47.2.13.80: tcp 0
10:43:18.805359 IP 10.47.3.6.27119 > 10.47.2.13.80: tcp 446
10:43:18.805636 IP 10.47.2.13.80 > 10.47.3.6.27119: tcp 0
10:43:18.806243 IP 10.47.2.13.80 > 10.47.3.6.27119: tcp 1448
10:43:18.806261 IP 10.47.2.13.80 > 10.47.3.6.27119: tcp 1448
10:43:18.806266 IP 10.47.2.13.80 > 10.47.3.6.27119: tcp 629
10:43:18.806919 IP 10.47.3.6.27119 > 10.47.2.13.80: tcp 0
10:43:18.837459 IP 10.47.3.6.27119 > 10.47.2.13.80: tcp 399
10:43:18.837927 IP 10.47.2.13.80 > 10.47.3.6.27119: tcp 180
10:43:18.878730 IP 10.47.3.6.27119 > 10.47.2.13.80: tcp 0
```

Task4

In order to create the VPN tunnel with IPSEC, we first had to make sure that our firewall rules were set properly. We cleared most of the firewall rules from previous tasks so that all traffic was allowed to pass through. We then also added a new firewall rule to the newly turned on IPSEC firewall rule area that also allowed any IPV4 traffic to pass through.

The screenshot shows a table of firewall rules under the 'IPsec' tab. A single rule is listed:

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input checked="" type="checkbox"/>	0 /1008 B	IPv4 *	*	*	*	*	*	none			

Below the table are buttons for 'Add', 'Delete', 'Save', and 'Separator'.

Next we began what is called phase 1. Here we set up the remote gateway, description, internet protocol, encryption algorithm and pre-shared key. For firewallA , our remote gateway as 10.47.3.7. This is telling IPSEC that the other endpoint of our tunnel is 10.47.3.7. We similarly set up the reverse remote gateway for firewallB, 10.47.3.6.

The screenshot shows the 'Tunnels' configuration page. Under 'General Information', the 'Remote Gateway' field is set to '10.47.3.7'. This field is circled in red.

The screenshot shows the 'Tunnels' configuration page. Under 'General Information', the 'Remote Gateway' field is set to '10.47.3.6'. This field is circled in red.

Raphael Archer, Ryan Rozanitis
CS 646 Project 3 Deliverable

We then configured our pre-shared key and encryption algorithm for phase 1. Near bubble 1 in the picture below, we generated a pre-shared key and copied it over to the other firewall as well. We wanted this to be as secure as possible as per our task 4 requirements, so generating a random key is best practice.

For the Encryption Algorithm, we had to do a bit of research. Pfsense's configuration documentation told us to use at least AES128-GCM with AES-XCBC and DH group 14 or higher. We ended up deciding the AES256-GCM is more secure and stuck with AES-XCBC as our hash. Our key length remained 128 bits, which is the maximum size. We then did a bit more independent research to determine what the best DH group was. Originally we thought DH group 32 would be the best, or ed25519, but enough articles pushed us to use DH group 21 as the standard to use with AES256-GCM while using a 128 bit key.

The screenshot shows two configuration pages for Pfsense. The top section, "Phase 1 Proposal (Authentication)", includes fields for "Authentication Method" (Mutual PSK), "My Identifier" (My IP address), "Peer Identifier" (Peer IP address), and "Pre-Shared Key" (a long hex string). The bottom section, "Phase 1 Proposal (Encryption Algorithm)", includes fields for "Encryption Algorithm" (AES256-GCM), "Key length" (128 bits), "Hash" (AES-XCBC), and "DH Group" (21). Both sections have notes and a "Generate new Pre-Shared Key" button.

Next we had to add our phase 2 for IPSEC. The important distinction between the two firewalls at this stage is the remote network address. For FirewallA, the remote network is 10.47.2.0/24

The screenshot shows the "Tunnels" configuration page under "General Information". It includes fields for "Mode" (Tunnel IPv4), "Local Network" (LAN subnet), "NAT/BINAT translation" (None), and "Remote Network" (Network type set to 10.47.2.0/24). The "Remote Network" field is circled in red.

Raphael Archer, Ryan Rozanitis
CS 646 Project 3 Deliverable

For FirewallB, the remote network is 10.47.1.0/24.

Tunnels Mobile Clients Pre-Shared Keys Advanced Settings

General Information

Disabled	<input type="checkbox"/> Disable this phase 2 entry without removing it from the list.
Mode	Tunnel IPv4
Local Network	LAN subnet
Type	Address
Local network component of this IPsec security association.	
NAT/BINAT translation	None
Type	Address
If NAT/BINAT is required on this network specify the address to be translated	
Remote Network	Network
Type	Address
Remote network component of this IPsec security association.	
Description	A description may be entered here for administrative reference (not parsed).

Like phase 1, the encryption at phase 2 is the same for both firewalls. We use AES256-GCM with 128 bits and Key group 21.

Phase 2 Proposal (SA/Key Exchange)

Protocol	ESP
Encapsulating Security Payload (ESP) is encryption, Authentication Header (AH) is authentication only.	
Encryption Algorithms	<input type="checkbox"/> AES Auto
<input type="checkbox"/> AES128-GCM Auto	
<input type="checkbox"/> AES192-GCM Auto	
<input checked="" type="checkbox"/> AES256-GCM 128 bits	
<input type="checkbox"/> Blowfish Auto	
<input type="checkbox"/> 3DES	
<input type="checkbox"/> CAST128	
Note: Blowfish, 3DES, and CAST128 provide weak security and should be avoided.	
Hash Algorithms	<input type="checkbox"/> MD5 <input type="checkbox"/> SHA1 <input type="checkbox"/> SHA256 <input type="checkbox"/> SHA384 <input type="checkbox"/> SHA512 <input type="checkbox"/> AES-XCBC
Note: Hash is ignored with GCM algorithms. MD5 and SHA1 provide weak security and should be avoided.	
PFS key group	21 (nist ecp521)
Note: Groups 1, 2, 5, 22, 23, and 24 provide weak security and should be avoided.	

After this, we save both the phase 2 in the firewalls. We can then test and see if they connect by clicking the connect button.

Raphael Archer, Ryan Rozanitis
CS 646 Project 3 Deliverable

The screenshot shows two separate network interfaces, FirewallA and FirewallB, each with its own IPsec Status and SPDs tables.

FirewallA IPsec Status:

IPsec ID	Description	Local	Remote	Role	Timers	Algo	Status
con100000: #13	FirewallB	ID: 10.47.3.6 Host: 10.47.3.6:500 SPI: 45d9f5890e733e97	ID: 10.47.3.7 Host: 10.47.3.7:500 SPI: 2a94c55f9e023b6e	IKEv2 initiator	Rekey: 24868s (06:54:28) Reauth: Disabled	AES_GCM_16 (256) PRF_AES128_XCBC ECP_521	ESTABLISHED 15 seconds (00:00:15) ago Disconnect

IPsec ID	Local subnets	Local SPI(s)	Remote subnets	Times	Algo	Stats	
con100000: #16	10.47.1.0/24	Local: c52a8a92 Remote: c7f9dcfa	10.47.2.0/24	Rekey: 3065 seconds (00:51:05) Life: 3585 seconds (00:59:45) Install: 15 seconds (00:00:15)	AES_GCM_16 (256) IPComp: none	Bytes-In: 0 (0 B) Packets-In: 0 Bytes-Out: 0 (0 B) Packets-Out: 0	Disconnect

FirewallB IPsec Status:

IPsec ID	Description	Local	Remote	Role	Timers	Algo	Status
con100000: #12	FirewallA	ID: 10.47.3.7 Host: 10.47.3.7:500 SPI: 2a94c55f9e023b6e	ID: 10.47.3.6 Host: 10.47.3.6:500 SPI: 45d9f5890e733e97	IKEv2 responder	Rekey: 23974s (06:39:34) Reauth: Disabled	AES_GCM_16 (256) PRF_AES128_XCBC ECP_521	ESTABLISHED 28 seconds (00:00:28) ago Disconnect

IPsec ID	Local subnets	Local SPI(s)	Remote subnets	Times	Algo	Stats	
con100000: #9	10.47.2.0/24	Local: c7f9dcfa Remote: c52a8a92	10.47.1.0/24	Rekey: 3158 seconds (00:52:38) Life: 3572 seconds (00:59:32) Install: 28 seconds (00:00:28)	AES_GCM_16 (256) IPComp: none	Bytes-In: 0 (0 B) Packets-In: 0 Bytes-Out: 0 (0 B) Packets-Out: 0	Disconnect

We quickly look at SAD and SPD. SADs are tables that contain the information for a Security Association. SPDs tell what packets IPSEC should apply to. In the SPDs picture, you can see that it specifies the direction. This is telling IPSEC the direction the packets are flowing when IPSEC should be keeping track of them. We can also see the tunnel endpoints are flip flopped in the pictures, which makes sense. Our Firewalls are mirroring each other as endpoints in the VPN tunnel.

The screenshot shows two separate network interfaces, FirewallA and FirewallB, each with its own SAD and SPDs tables.

FirewallA SAD and SPDs:

Source	Destination	Protocol	SPI	Enc. alg.	Auth. alg.	Data
10.47.3.6	10.47.3.7	ESP	cb39ad84	aes-gcm-16		840 B Delete
10.47.3.7	10.47.3.6	ESP	c8002f63	aes-gcm-16		504 B Delete

FirewallB SAD and SPDs:

Source	Destination	Protocol	SPI	Enc. alg.	Auth. alg.	Data
10.47.3.7	10.47.3.6	ESP	c8002f63	aes-gcm-16		840 B Delete
10.47.3.6	10.47.3.7	ESP	cb39ad84	aes-gcm-16		504 B Delete

Raphael Archer, Ryan Rozanitis
CS 646 Project 3 Deliverable

Overview		Leases		SPDs
Source	Destination	Direction	Protocol	Tunnel endpoints
10.47.2.0/24	10.47.1.0/24	◀ Inbound	ESP	10.47.3.7 -> 10.47.3.6
10.47.1.0/24	10.47.2.0/24	▶ Outbound	ESP	10.47.3.6 -> 10.47.3.7

Overview		Leases		SPDs
Source	Destination	Direction	Protocol	Tunnel endpoints
10.47.1.0/24	10.47.2.0/24	◀ Inbound	ESP	10.47.3.6 -> 10.47.3.7
10.47.2.0/24	10.47.1.0/24	▶ Outbound	ESP	10.47.3.7 -> 10.47.3.6

After completing a quick SSH test, we see the packets in and out change. This proves to us that the packets are moving through the tunnel.

Overview	Leases	SADs	SPDs				
IPsec Status							
IPsec ID	Description	Local	Remote	Role	Timers	Algo	Status
con100000: #13	FirewallB	ID: 10.47.3.6 Host: 10.47.3.6:500 SPI: 45d9f5890e733e97	ID: 10.47.3.7 Host: 10.47.3.7:500 SPI: 2a94c55f9e023b6e	IKEv2 initiator	Rekey: 24587s (06:49:47) Reauth: Disabled	AES_GCM_16 (256) PRF_AES128_XCBC ECP_521	ESTABLISHED 296 seconds (00:04:56) ago 
IPsec ID	Local subnets	Local SPI(s)	Remote subnets	Times	Algo	Stats	
con100000: #16	10.47.1.0/24	Local: c52a8a92 Remote: c7f9dcfa	10.47.2.0/24	Rekey: 2784 seconds (00:46:24) Life: 3304 seconds (00:55:04) Install: 296 seconds (00:04:56)	AES_GCM_16 (256) IPComp: none	Bytes-In: 4,038 (4 KIB) Bytes-Out: 4,424 (4 KIB)	

We also confirm this by looking at the logs. On firewallA, we see a request generated, packet sent, and then packet received and response parsed. This is our ssh test we just performed a moment ago.

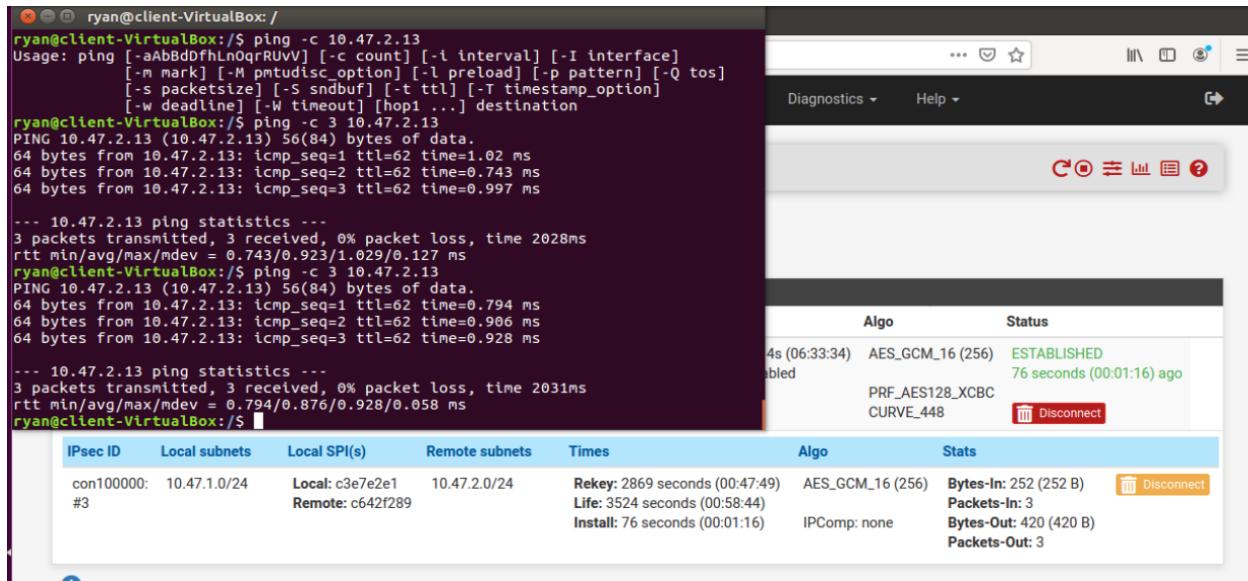
May 4 01:46:25	charon	67921	10[ENC] <con100000 13> generating INFORMATIONAL request 25 []
May 4 01:46:25	charon	67921	10[NET] <con100000 13> sending packet: from 10.47.3.6[500] to 10.47.3.7[500] (57 bytes)
May 4 01:46:25	charon	67921	10[NET] <con100000 13> received packet: from 10.47.3.7[500] to 10.47.3.6[500] (57 bytes)
May 4 01:46:25	charon	67921	10[ENC] <con100000 13> parsed INFORMATIONAL response 25 []

We can also see the logs from firewallB showing a mirror image of the above. First we receive a packet and parse it, then generate a response and send it out.

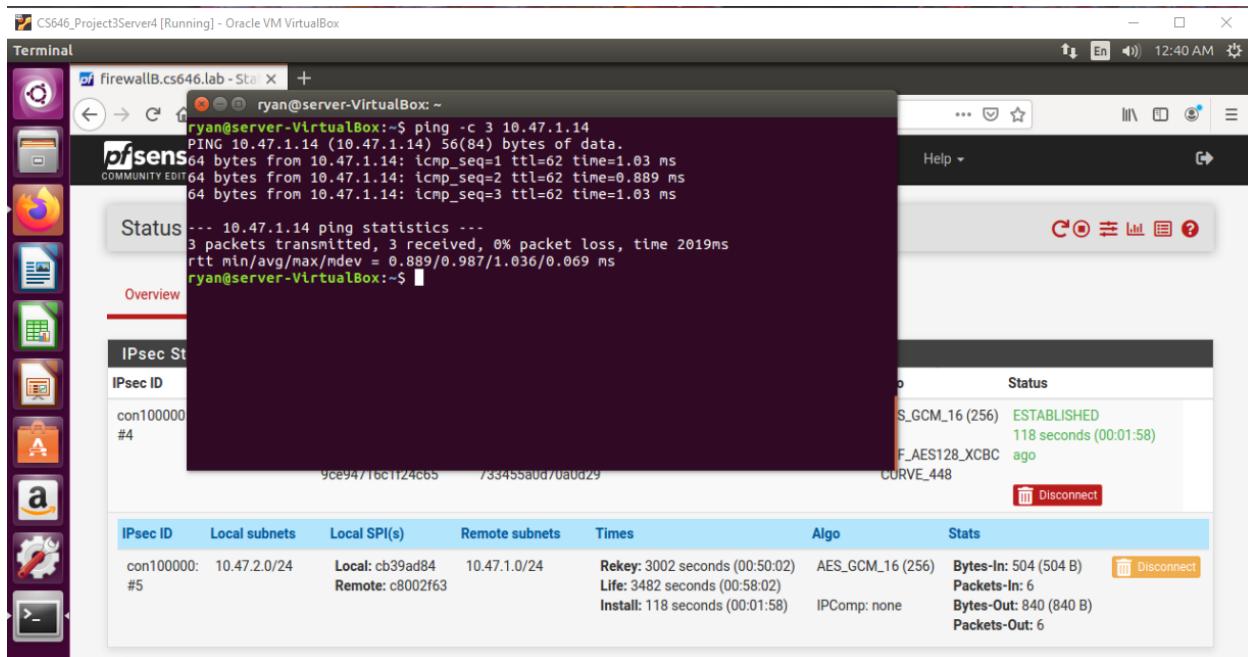
May 4 01:46:34	charon	45056	07[NET] <con100000 12> received packet: from 10.47.3.6[500] to 10.47.3.7[500] (57 bytes)
May 4 01:46:34	charon	45056	07[ENC] <con100000 12> parsed INFORMATIONAL request 26 []
May 4 01:46:34	charon	45056	07[ENC] <con100000 12> generating INFORMATIONAL response 26 []
May 4 01:46:34	charon	45056	07[NET] <con100000 12> sending packet: from 10.47.3.7[500] to 10.47.3.6[500] (57 bytes)

Raphael Archer, Ryan Rozanitis
CS 646 Project 3 Deliverable

We restart the connection, to get 0 packets in and our to make sure that our connection is working. We then sent 3 pings from 10.47.1.14 to 10.47.2.13. In response, we see 3 packets in and 3 packets out.



Similarly, we do the reverse on the server. We send 3 pings from 10.47.2.13 to 10.47.1.14. Now we see 6 packets in and out of our VPN tunnel.



Raphael Archer, Ryan Rozanitis
CS 646 Project 3 Deliverable

Lastly, we do one final packet capture using the apache web server as our test. We set up the packet capture on firewallA and firewallB to only look for packets looking for port 80. We also specify to look on our IPSEC interface.

Packet Capture Options

Interface	IPsec
Select the interface on which to capture traffic.	
Promiscuous	<input type="checkbox"/> Enable promiscuous mode Non-promiscuous mode captures only traffic that is directly relevant to the host (sent by it, sent or broadcast to it, or routed through it) and does not show packets that are ignored at network adapter level. Promiscuous mode ("sniffing") captures all data seen by the adapter, whether or not it is valid or related to the host, but in some cases may have undesirable side effects and not all adapters support this option. Click Info for details Info
Address Family	Any
Select the type of traffic to be captured.	
Protocol	Any
Select the protocol to capture, or "Any".	
Host Address	
This value is either the Source or Destination IP address, subnet in CIDR notation, or MAC address. Matching can be negated by preceding the value with "!". Multiple IP addresses or CIDR subnets may be specified. Comma (",") separated values perform a boolean "AND". Separating with a pipe (" ") performs a boolean "OR". MAC addresses must be entered in colon-separated format, such as xxx:xx:xx:xx:xx or a partial address consisting of one (xx), two (xx:xx), or four (xxxx:xx) segments. If this field is left blank, all packets on the specified interface will be captured.	
Port	80
The port can be either the source or destination port. The packet capture will look for this port in either field. Matching can be negated by preceding the value with "!". Multiple ports may be specified. Comma (",") separated values perform a boolean "AND". Separating with a pipe (" ") performs a boolean "OR". Leave blank if not filtering by port.	

The results below prove our IPSEC VPN tunnel is active and functional. First we have the packet capture on firewallA. We can see the client and the web server communicating over HTTP. We also see the client initiated the connection, since the first packet captured is 10.47.1.14.45336 > 10.47.2.13.80. The (authentic, confidential) also reaffirms that this is a VPN connection, since we had to authenticate using the pre-shared key and encrypt using AES156-GCM with all of our other parameters.

Last capture start	May 4th, 2021 1:59:51 am.
Last capture stop	May 4th, 2021 1:59:58 am.
Start View Capture Download Capture	
Packets Captured	
01:59:55.630002 (authentic,confidential): SPI 0xc7f9dcfa: IP 10.47.1.14.45336 > 10.47.2.13.80: tcp 0 01:59:55.630668 (authentic,confidential): SPI 0xc52a8a92: IP 10.47.2.13.80 > 10.47.1.14.45336: tcp 0 01:59:55.630946 (authentic,confidential): SPI 0xc7f9dcfa: IP 10.47.1.14.45336 > 10.47.2.13.80: tcp 0 01:59:55.631211 (authentic,confidential): SPI 0xc7f9dcfa: IP 10.47.1.14.45336 > 10.47.2.13.80: tcp 447 01:59:55.631764 (authentic,confidential): SPI 0xc52a8a92: IP 10.47.2.13.80 > 10.47.1.14.45336: tcp 0 01:59:55.632256 (authentic,confidential): SPI 0xc52a8a92: IP 10.47.2.13.80 > 10.47.1.14.45336: tcp 1448 01:59:55.632284 (authentic,confidential): SPI 0xc52a8a92: IP 10.47.2.13.80 > 10.47.1.14.45336: tcp 1448 01:59:55.632293 (authentic,confidential): SPI 0xc52a8a92: IP 10.47.2.13.80 > 10.47.1.14.45336: tcp 629 01:59:55.632545 (authentic,confidential): SPI 0xc7f9dcfa: IP 10.47.1.14.45336 > 10.47.2.13.80: tcp 0 01:59:55.670295 (authentic,confidential): SPI 0xc7f9dcfa: IP 10.47.1.14.45336 > 10.47.2.13.80: tcp 401 01:59:55.671164 (authentic,confidential): SPI 0xc52a8a92: IP 10.47.2.13.80 > 10.47.1.14.45336: tcp 180 01:59:55.714983 (authentic,confidential): SPI 0xc7f9dcfa: IP 10.47.1.14.45336 > 10.47.2.13.80: tcp 0	

Raphael Archer, Ryan Rozanitis
CS 646 Project 3 Deliverable

We stopped firewallIB's packet capture a second later to show the difference, but other than that last capture stop time everything else is identical, which makes sense since both firewalls are communicating through the same VPN tunnel.

The screenshot shows a packet capture interface with the following details:

- Last capture start: May 4th, 2021 1:59:51 am.
- Last capture stop: May 4th, 2021 1:59:59 am.

Buttons at the top: Start (green), View Capture (blue), Download Capture (blue).

Packets Captured

```
01:59:54.716367 (authentic,confidential): SPI 0xc7f9dcfa: IP 10.47.1.14.45336 > 10.47.2.13.80: tcp 0
01:59:54.716719 (authentic,confidential): SPI 0xc52a8a92: IP 10.47.2.13.80 > 10.47.1.14.45336: tcp 0
01:59:54.717258 (authentic,confidential): SPI 0xc7f9dcfa: IP 10.47.1.14.45336 > 10.47.2.13.80: tcp 0
01:59:54.717548 (authentic,confidential): SPI 0xc7f9dcfa: IP 10.47.1.14.45336 > 10.47.2.13.80: tcp 447
01:59:54.717815 (authentic,confidential): SPI 0xc52a8a92: IP 10.47.2.13.80 > 10.47.1.14.45336: tcp 0
01:59:54.718272 (authentic,confidential): SPI 0xc52a8a92: IP 10.47.2.13.80 > 10.47.1.14.45336: tcp 1448
01:59:54.718300 (authentic,confidential): SPI 0xc52a8a92: IP 10.47.2.13.80 > 10.47.1.14.45336: tcp 1448
01:59:54.718344 (authentic,confidential): SPI 0xc52a8a92: IP 10.47.2.13.80 > 10.47.1.14.45336: tcp 629
01:59:54.718907 (authentic,confidential): SPI 0xc7f9dcfa: IP 10.47.1.14.45336 > 10.47.2.13.80: tcp 0
01:59:54.756122 (authentic,confidential): SPI 0xc7f9dcfa: IP 10.47.1.14.45336 > 10.47.2.13.80: tcp 401
01:59:54.756494 (authentic,confidential): SPI 0xc52a8a92: IP 10.47.2.13.80 > 10.47.1.14.45336: tcp 180
01:59:54.801545 (authentic,confidential): SPI 0xc7f9dcfa: IP 10.47.1.14.45336 > 10.47.2.13.80: tcp 0
```