CS 491
Assignment 2

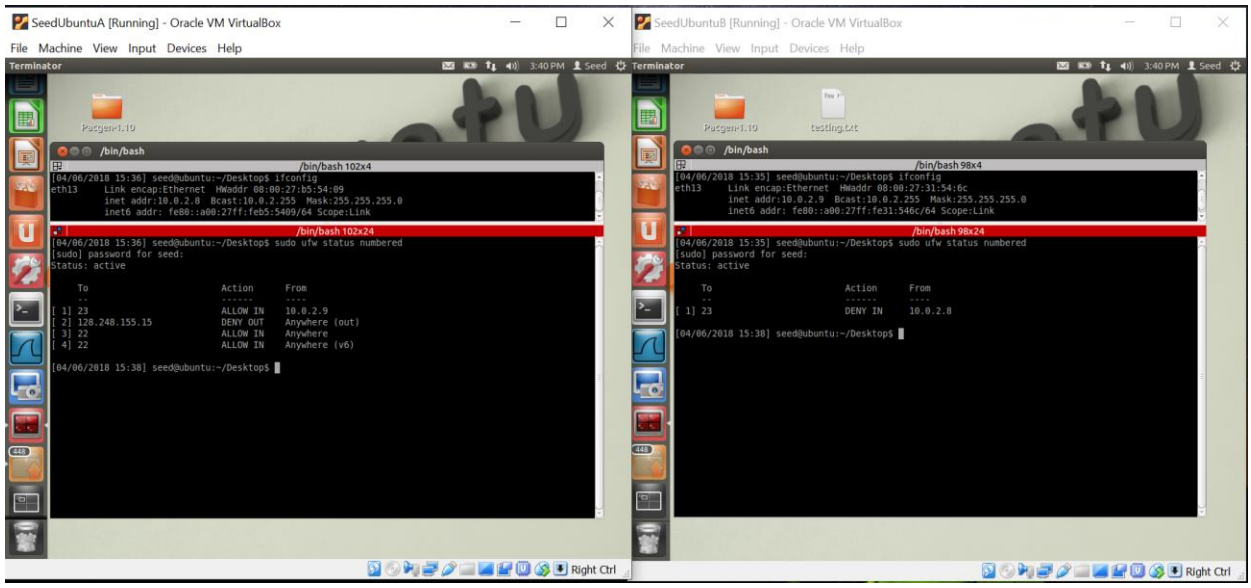
665336275 **Akshay Kataria**

Some fix things that won't change during the entire lab

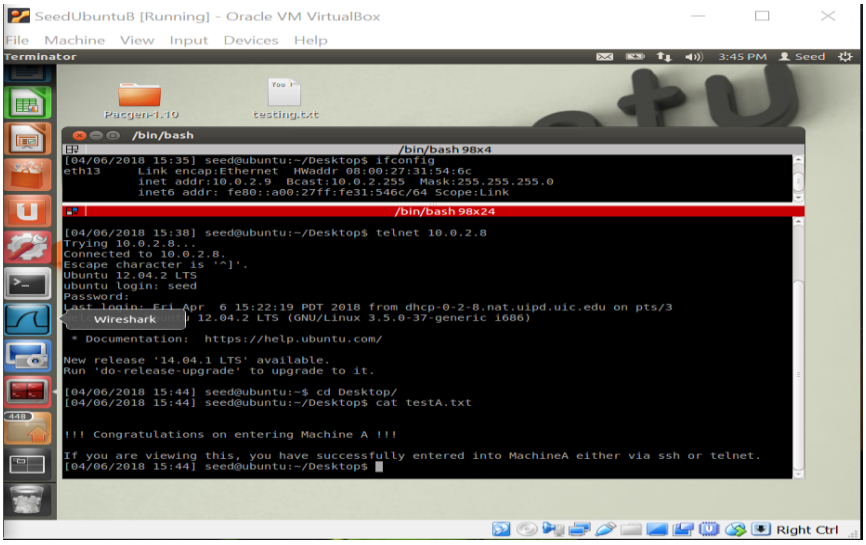
- Machine A's ip address: 10.0.2.8
- Machine B's ip address: 10.0.2.9
- Machine C's ip address: 10.0.2.10

Task 1: Using Firewall

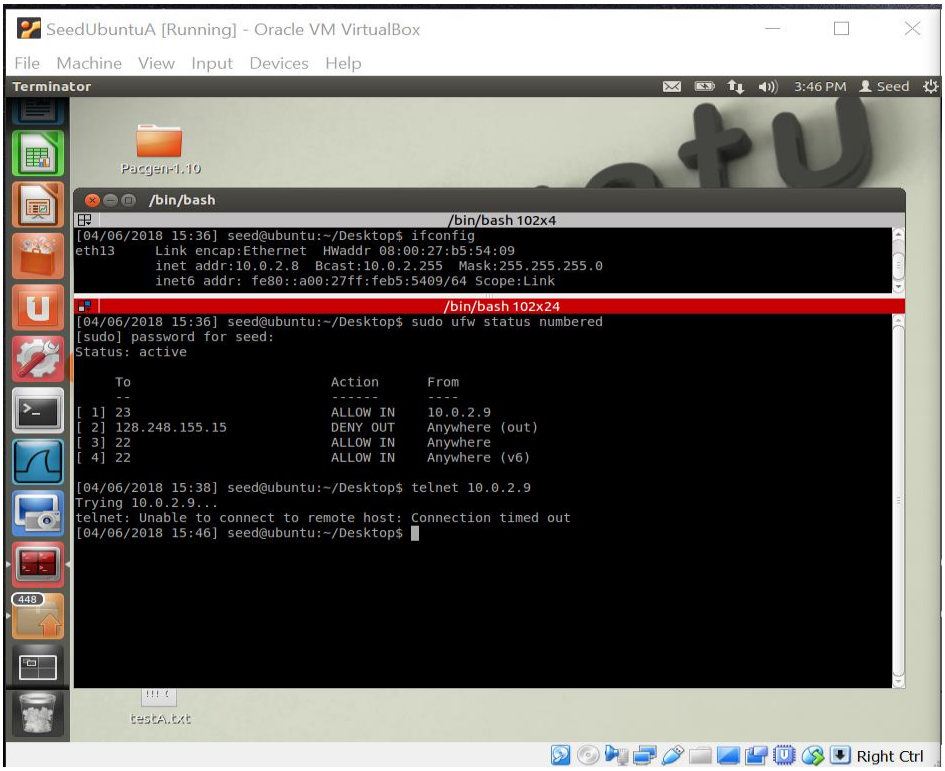
- Prevent Machine A from doing telnet to Machine B
- Prevent Machine A from doing telnet to Machine B
- Prevent Machine A from visiting an external website. Keep in mind that some websites have several IP address.



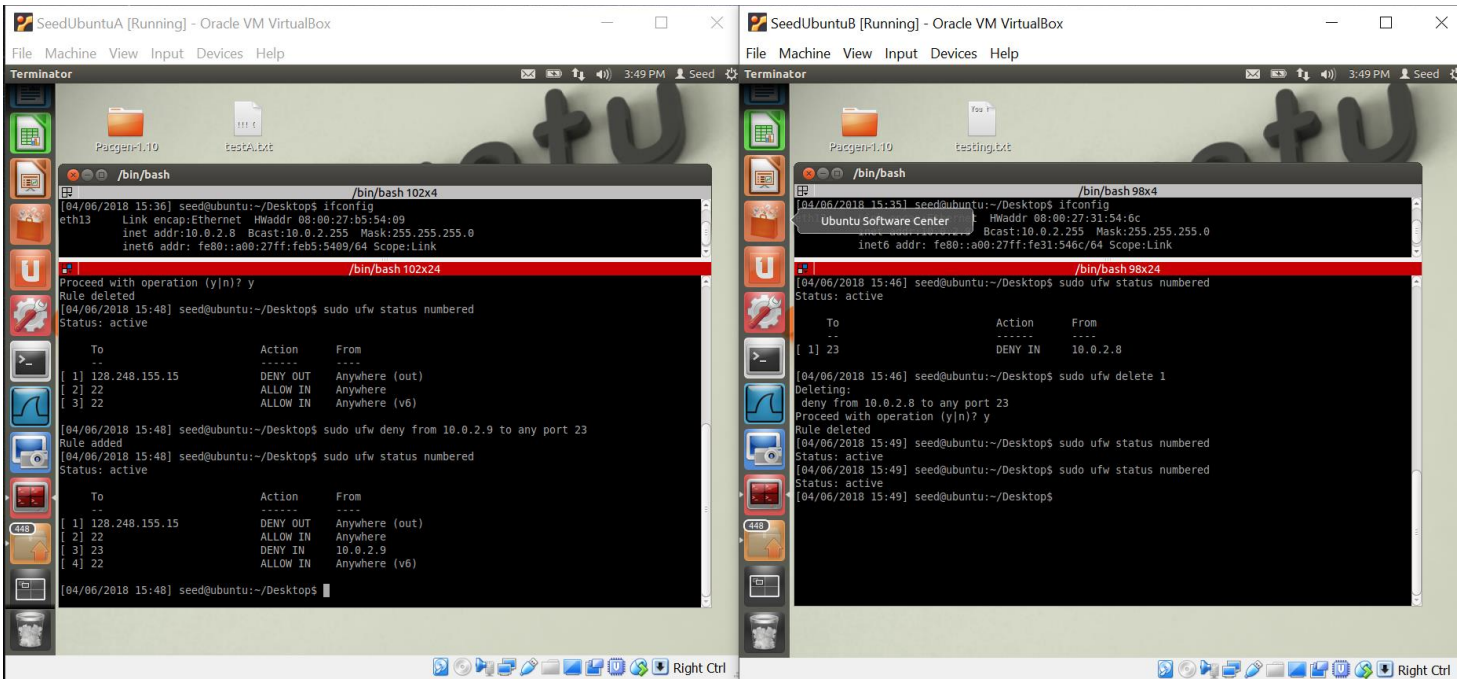
Obs: We can observe that ufw (Uncomplicated Firewall) have been installed on both the machines and is working fine. Machine B has rule to stop machine A from telnetting into it and machine A has no such rule.



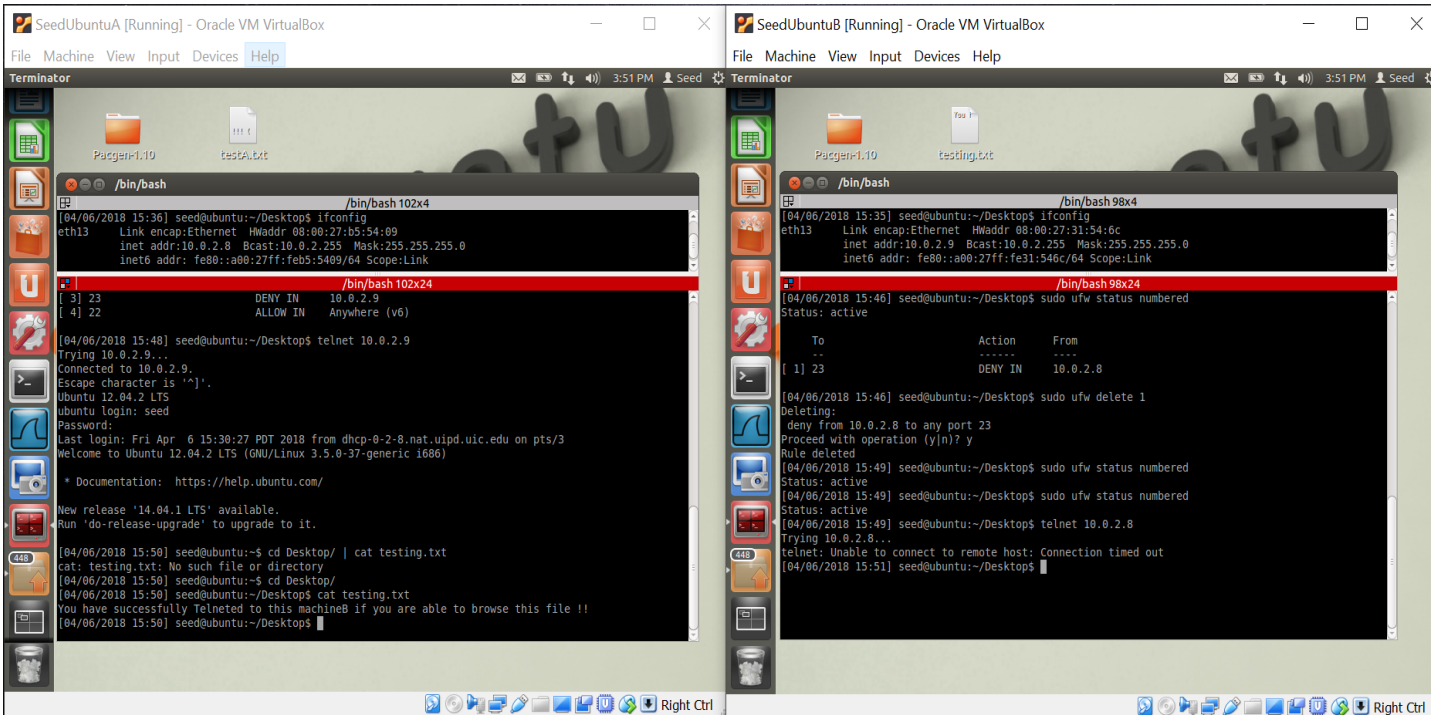
Obs: We can observe that machine B however was able to telnet into machine A because there was no rule blocking it. We can verify it by accessing the file that is saved on machine A's Desktop.



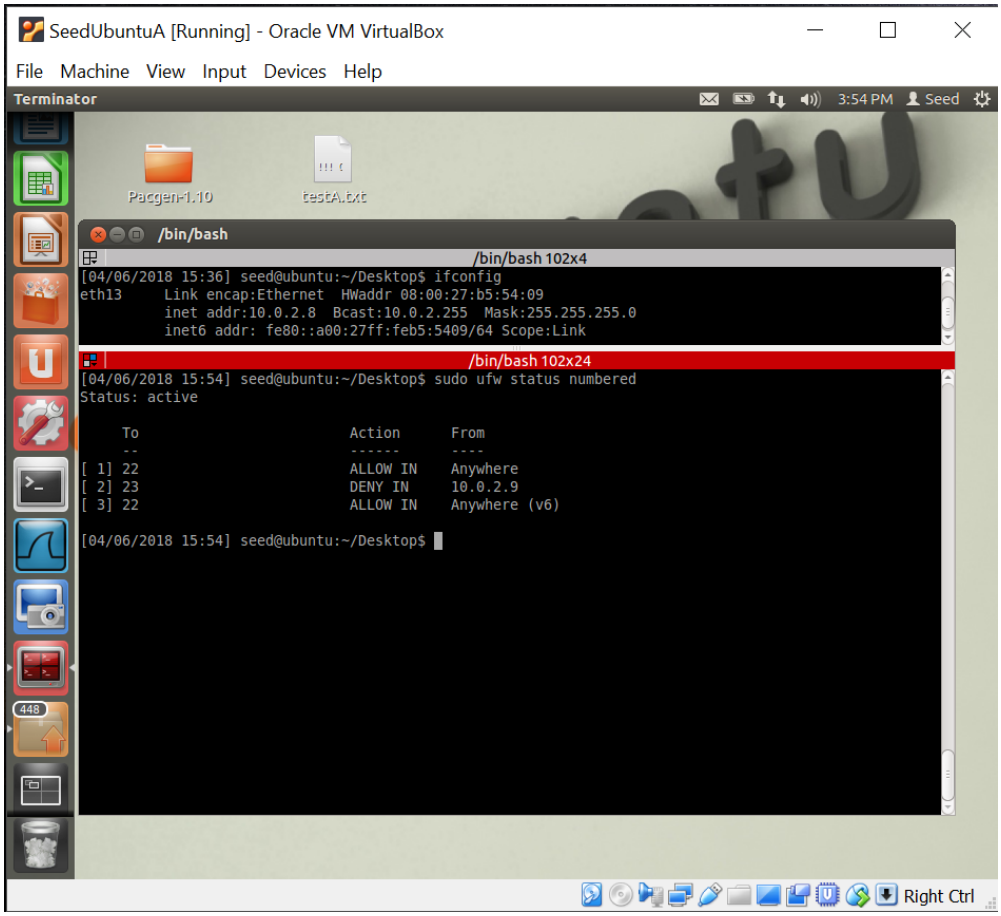
Obs: We can observe that telnet times out on machine A when it tried to connect to machine B. ufw rule on machine B is blocking it from doing so.



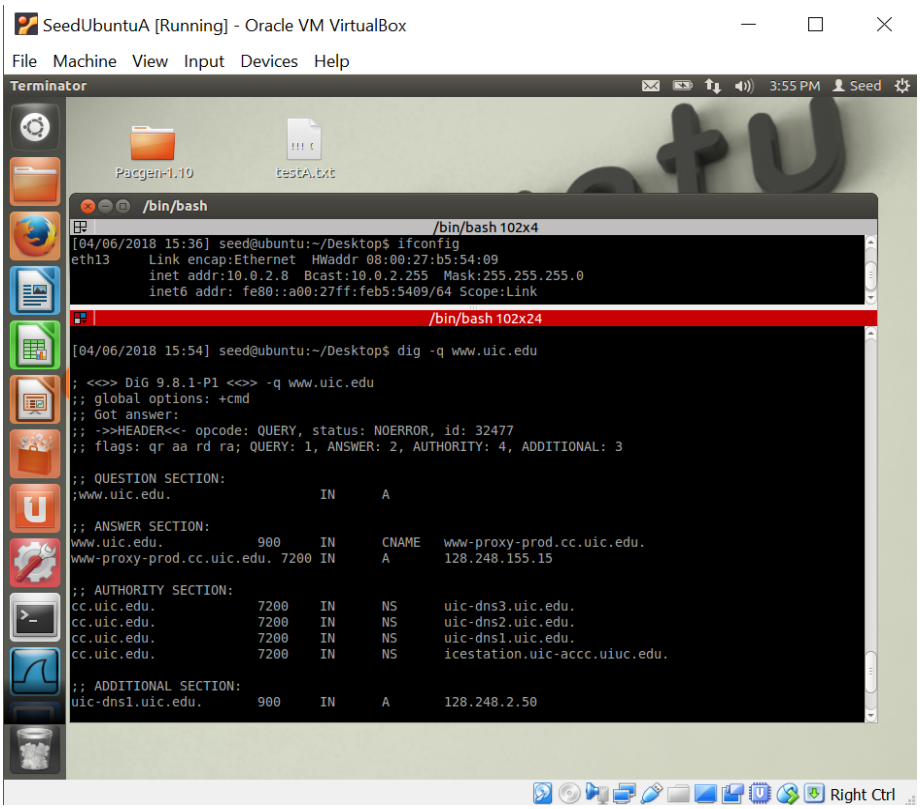
Obs: We have changed the rules now. Machine A is blocking telnet from machine B and rule that was blocking machine A to telnet has been removed from machine B.



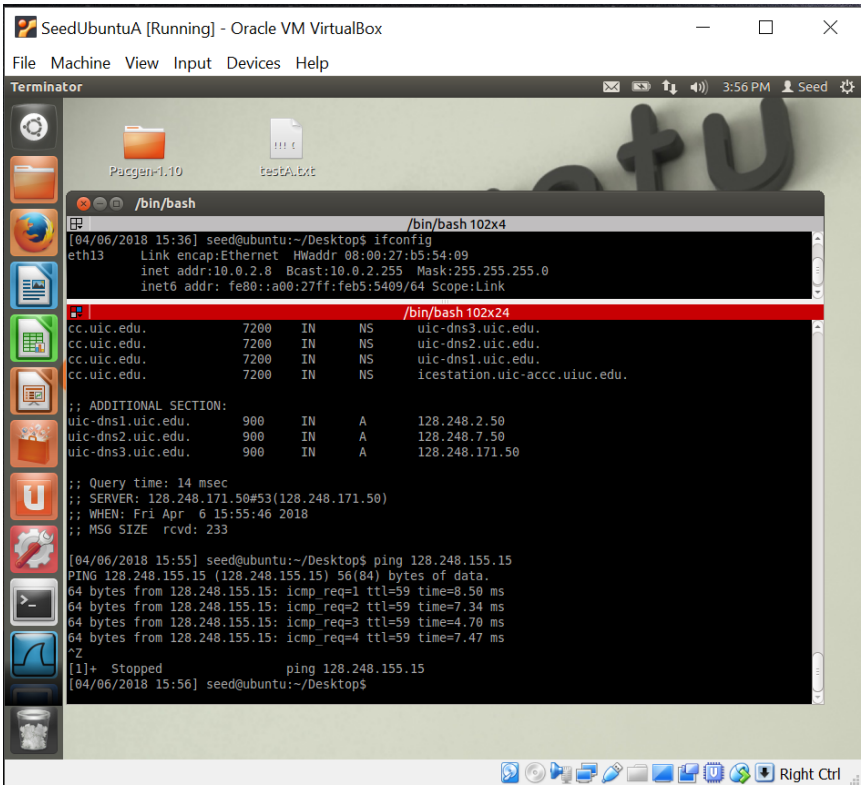
Obs: We can observe that machine A is able to telnet into machine B as it is able to open a file in machine B saved on it's desktop. On the other hand, we can observe that machine B is not able to telnet into machine A because ufw is blocking it now.



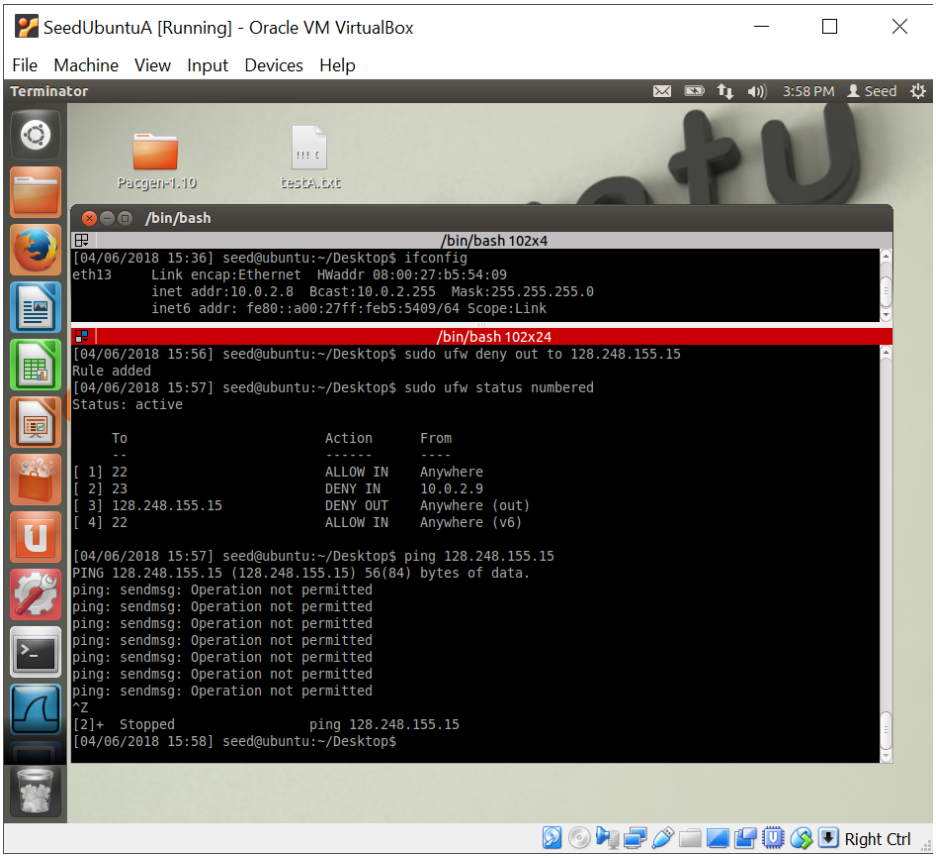
Obs: We can observe that ufw implemented on machine A is blocking telnet connection from machine B.



Obs: We are now trying to block access to a particular website from machie A. For our lab we are blocking "www.uic.edu". We found out it's address with the help of 'dig' command. Address of website to be blocked is: 124.248.155.15.



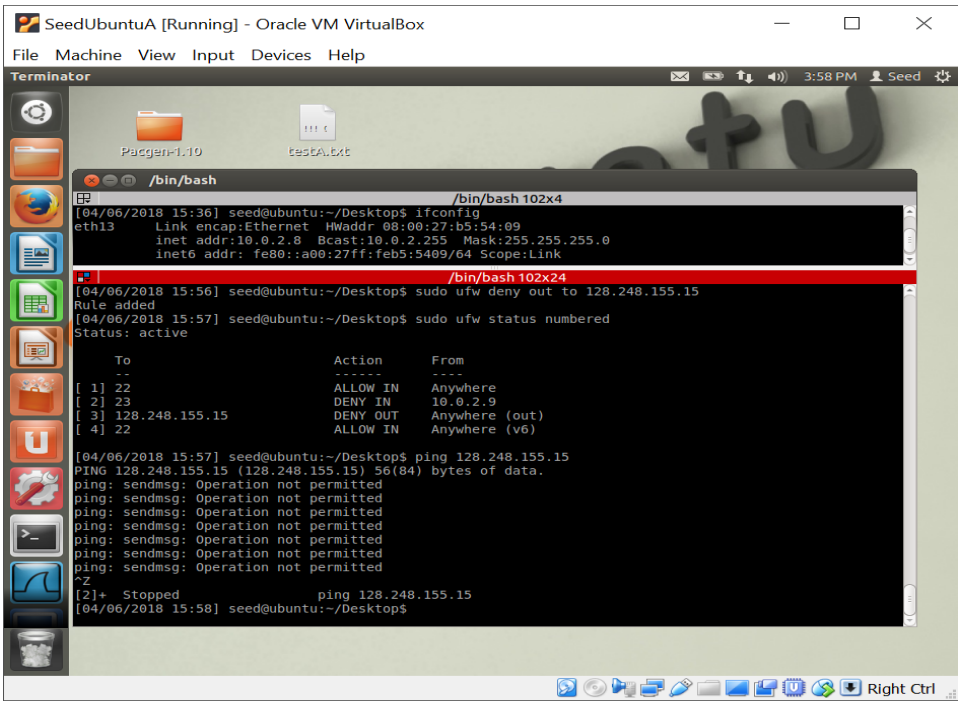
Obs: We can observe that we are able to ping "www.uic.edu" (124.248.155.15) with the help of the ping command. It shows that we able to access this website.



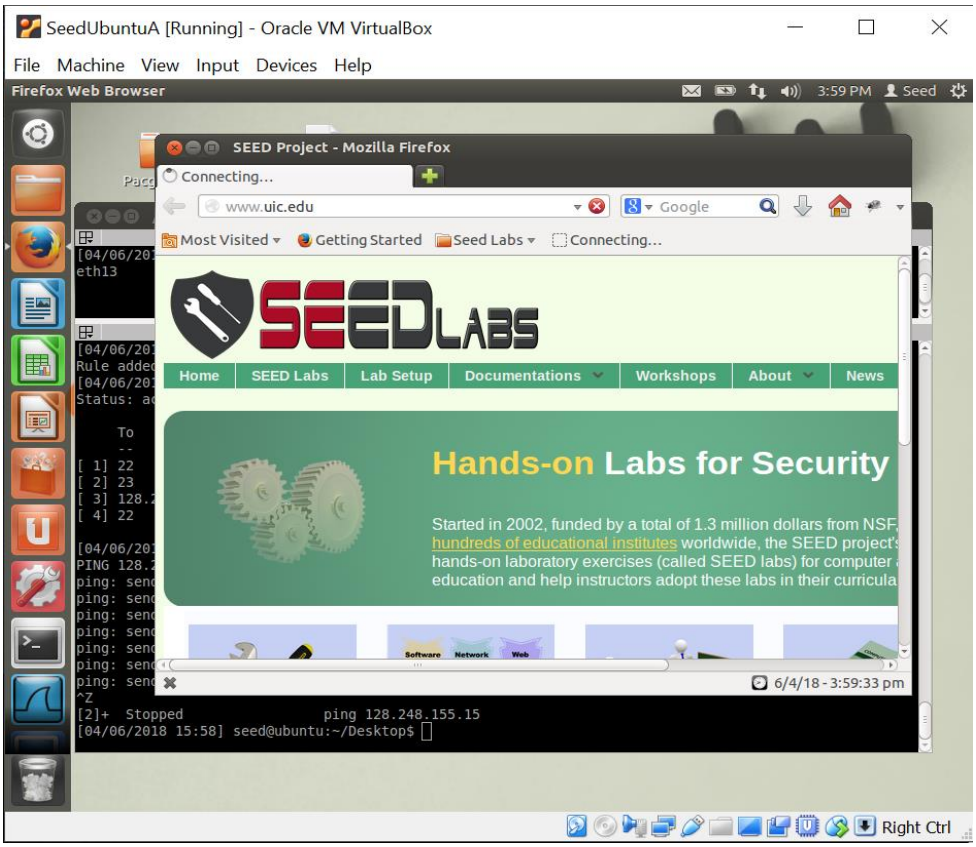
Obs: We can now observe that we are not able to ping “www.uic.edu” (124.248.155.15). This is because we have implemented a rule on ufw that prevents machine A from reaching that particular website. This can be seen as rule 3 in the ufw rule table.

Task 2: Evading Egress Filtering

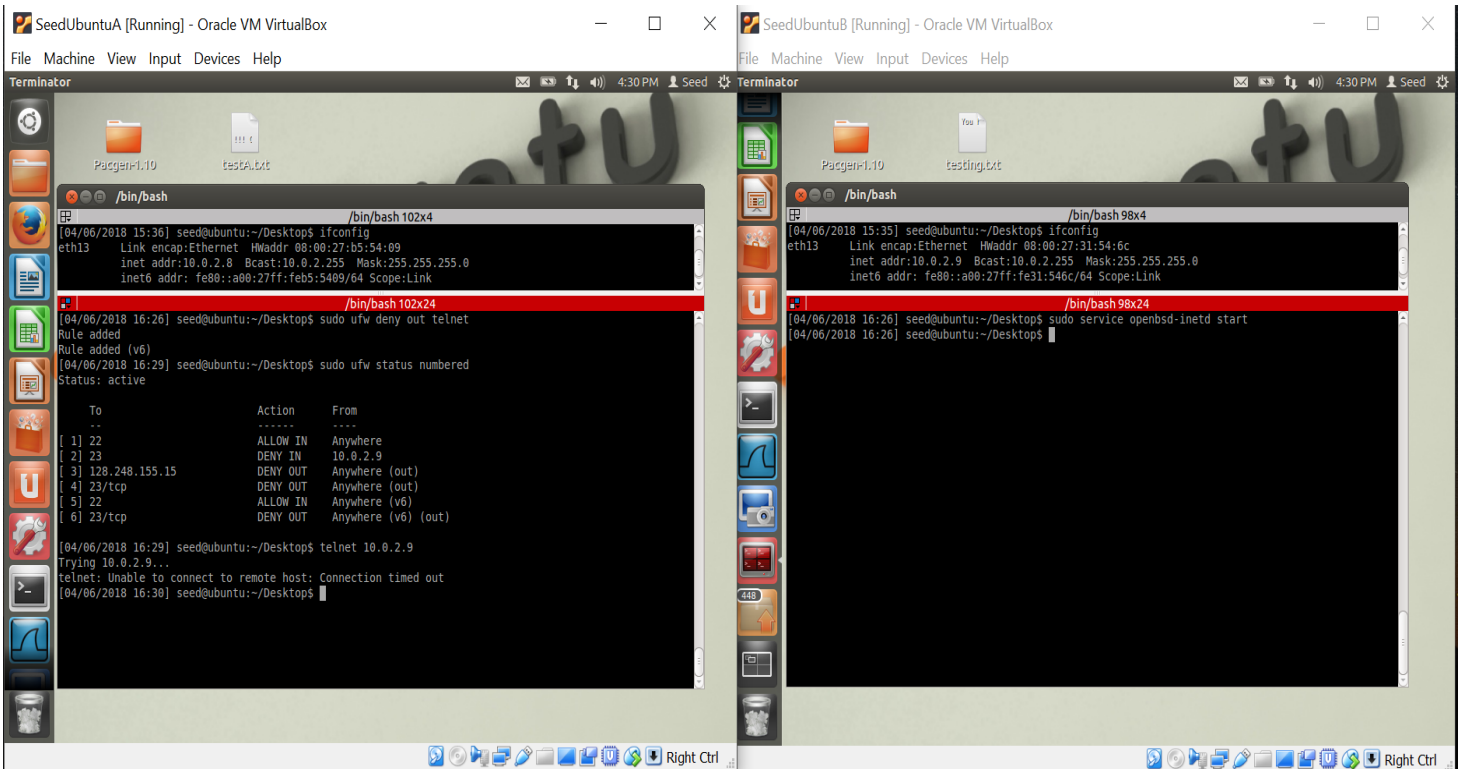
- Block all the outgoing traffic to www.uic.edu (using this website instead of “facebook” as it has only one IP address)
- Block all the outgoing traffic to external telnet servers



Obs: We can observe that we are not able to ping “www.uic.edu” (124.248.155.15). This is because we have implemented a rule on ufw that prevents machine A from reaching that particular website. Rule 3.



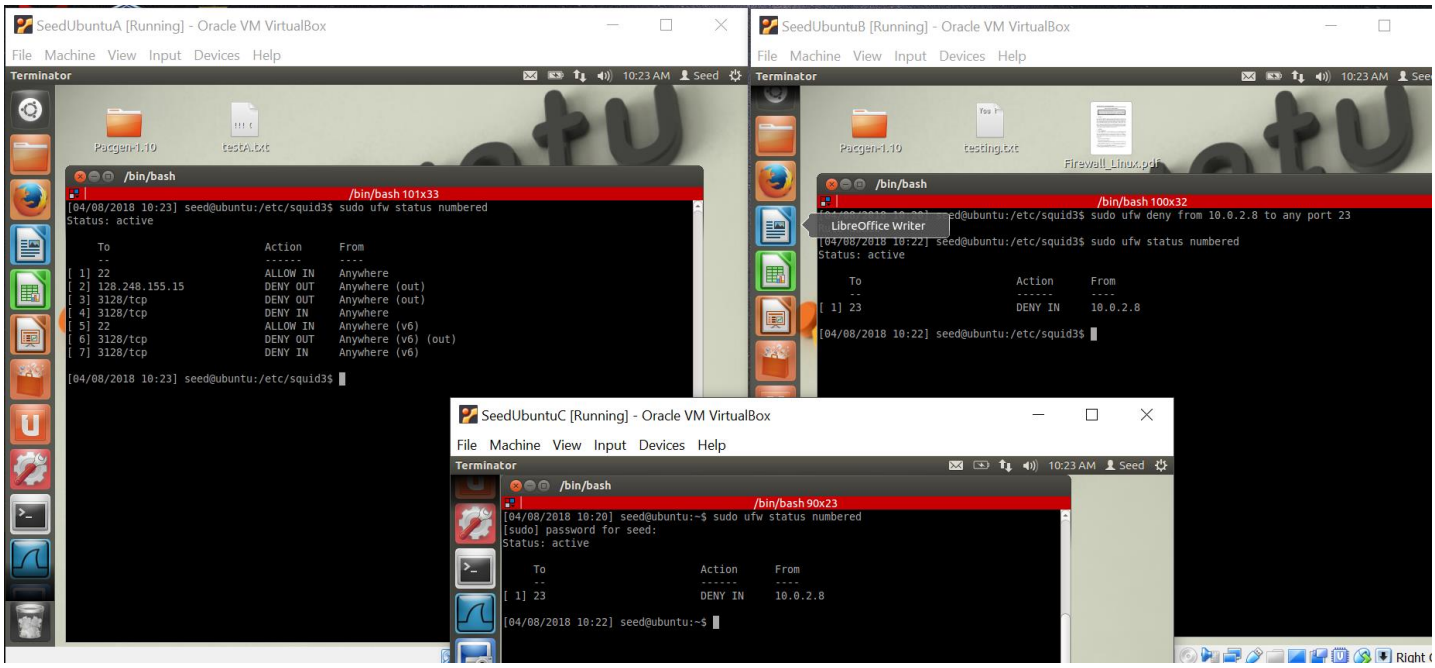
Obs: When we tried to reach out to www.uic.edu, we can clearly see that we are not able to reach that site. Thus this confirms that ufw has blocked any access to www.uic.edu



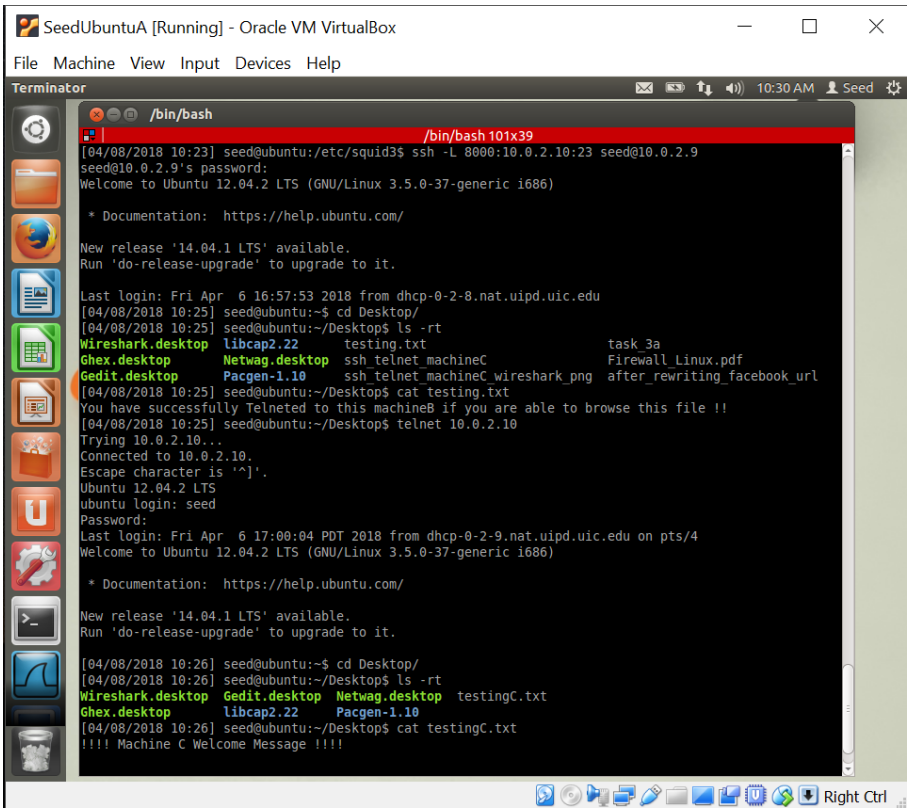
Obs: We are running telnet server on machine B and is trying to connect to it via machine A but we can oberve that our attempt to telnet to machine B fails. This is because of the rule 4 in the ufw firewall that is preventing all the outgoing connections via port 23. (Rule 4: 23/tcp DENY OUT)

Task 2a: Telnet to Machine B through firewall
Task 2b: Connecting to UIC's website using SSH Tunnel

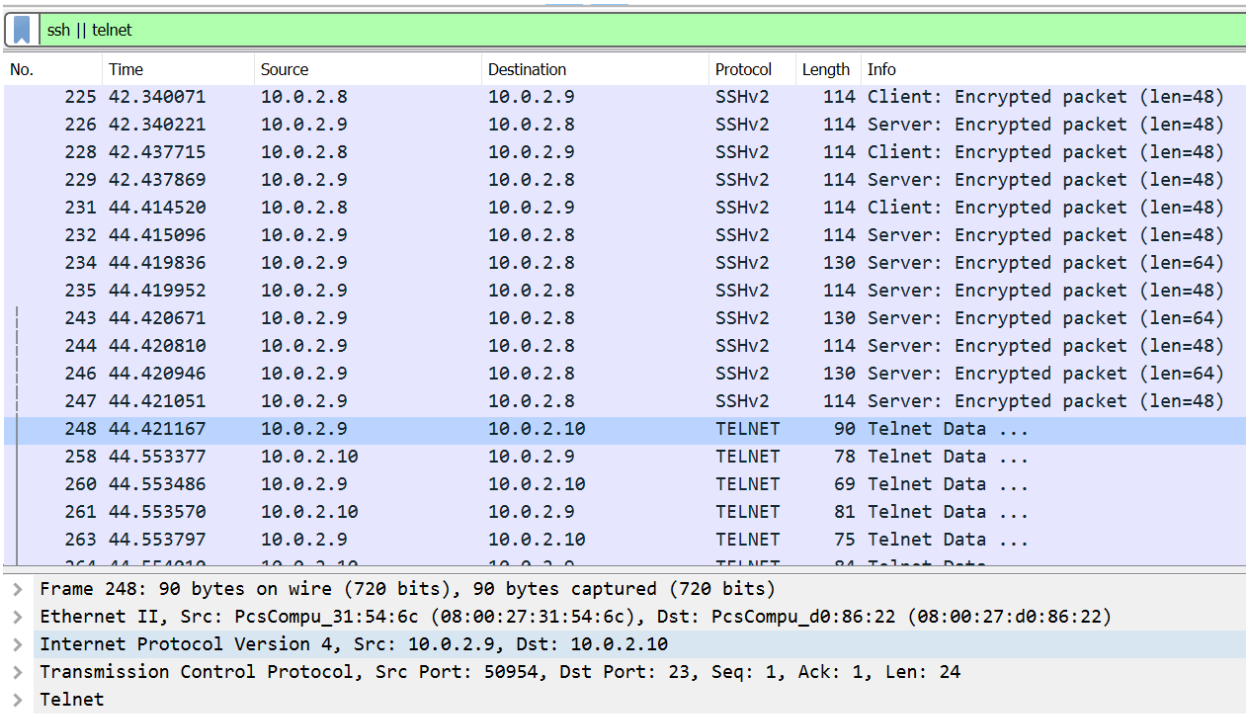
The combined task of 2a and 2b in a nutshell is to bypass the firewall without modifying the existing firewall policies. Though we will block the incoming telnet request from machine A to machine B and Machine C and have blocked the access to www.uic.edu on machine A, but still we will be able to access this site with the help of SSH tunnel.



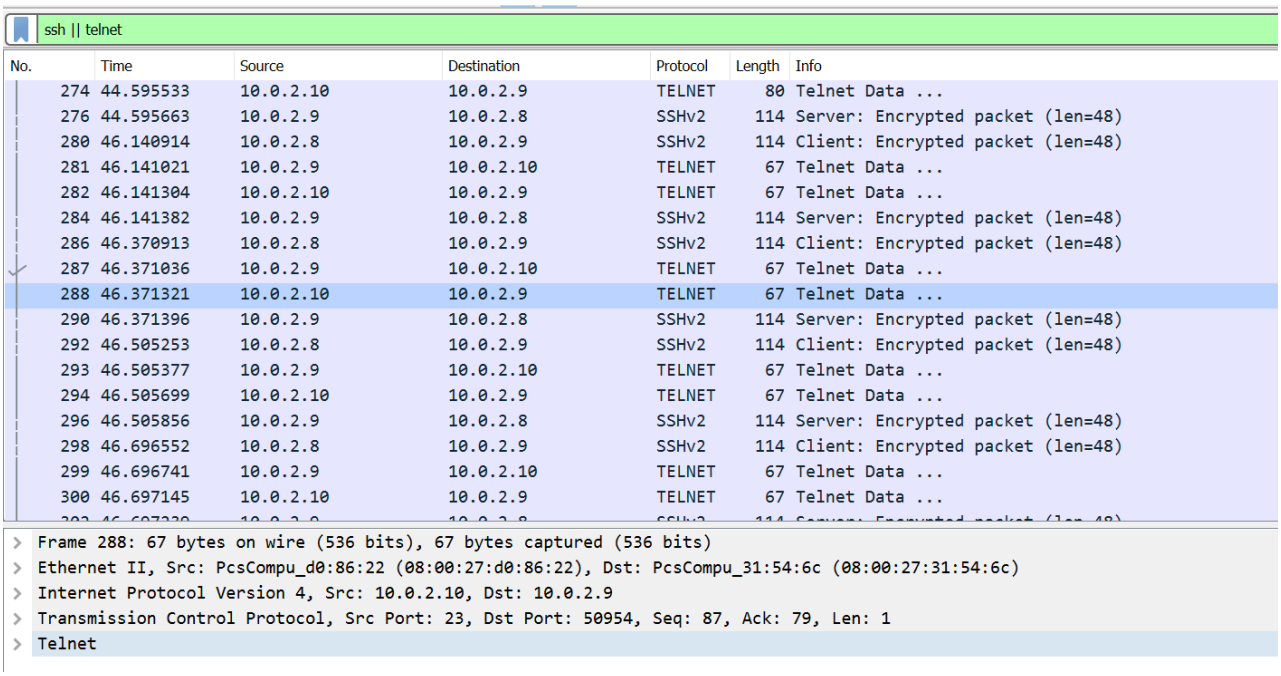
Obs: In this image we can observe that we have blocked any telnet connection request from machine A to machine B and machine C. This can be observed from machine's B and machine's C ufw rule 1. Also, access to www.uic.edu is blocked on machine A and can be seen in the ufw rule 2.



Obs: From this image, we can observe that we have successfully bypassed the firewall by implementing SSH from machine A to machine B and now it can perform everything that was being blocked on it's firewall.



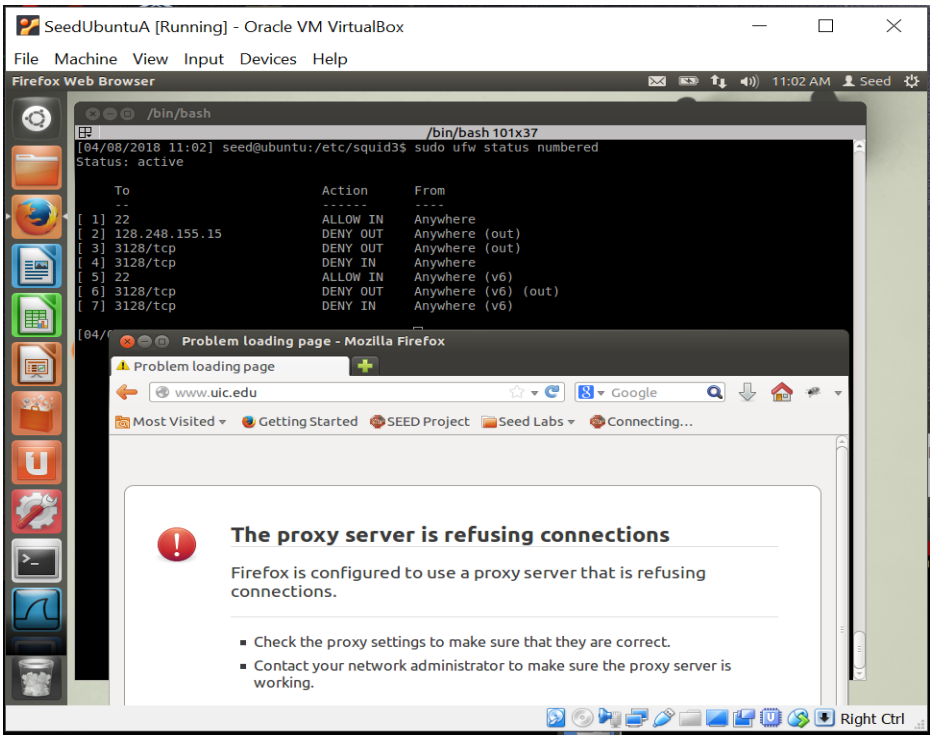
Obs: The above packet capture shows us that first a ssh tunnel is being created between machine A (10.0.2.8) and machine B (10.0.2.9) and then a telnet session was created between machine B and machine C (10.0.2.10).



Obs: From the above capture image, we can actually figure out how the communication is taking place between machine A, machine B and machine C. The packets are being requested from machine A to machine B through ssh and then these requests are forwarded to machine C via telnet. The same path is being followed to revert packets back to machine A. Machine C communicates to machine B via telnet and then machine B communicates packets to machine A via ssh.

We can ssh with the following command:
$ ssh -L 8000:10.0.2.10:23 seed@10.0.2.9
Here, -L represents binding the address, 10.0.2.10 is the host machine and 23 is the host port and seed@10.0.2.9 is to ssh to machine B. After entering in the machine B we can telnet to machine C with the help of "$ telnet 10.0.2.10" and from the figure we can oberve that we have successfully done that and had evaded egresss filtering. You can see the machine C welcome message at the bottom most line in the figure.

Obs: Applying the proxy settings as stated in the problem statement to make www.uic.edu work on machine A.



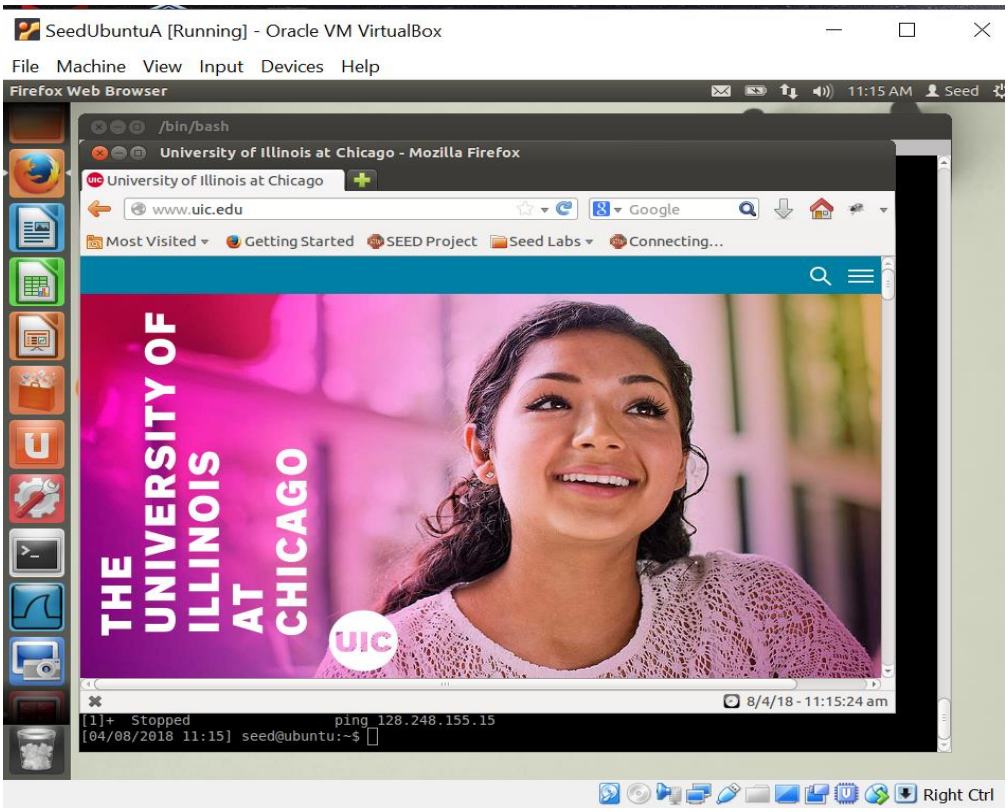
i) Run firefox and visit www.uic.edu . Can you see that page? Describe your obervation.

- From the above figure we can observe that we are not able to reach our destination because it has been blocked by one of the rule in machine's A ufw. (ufw rule 2)

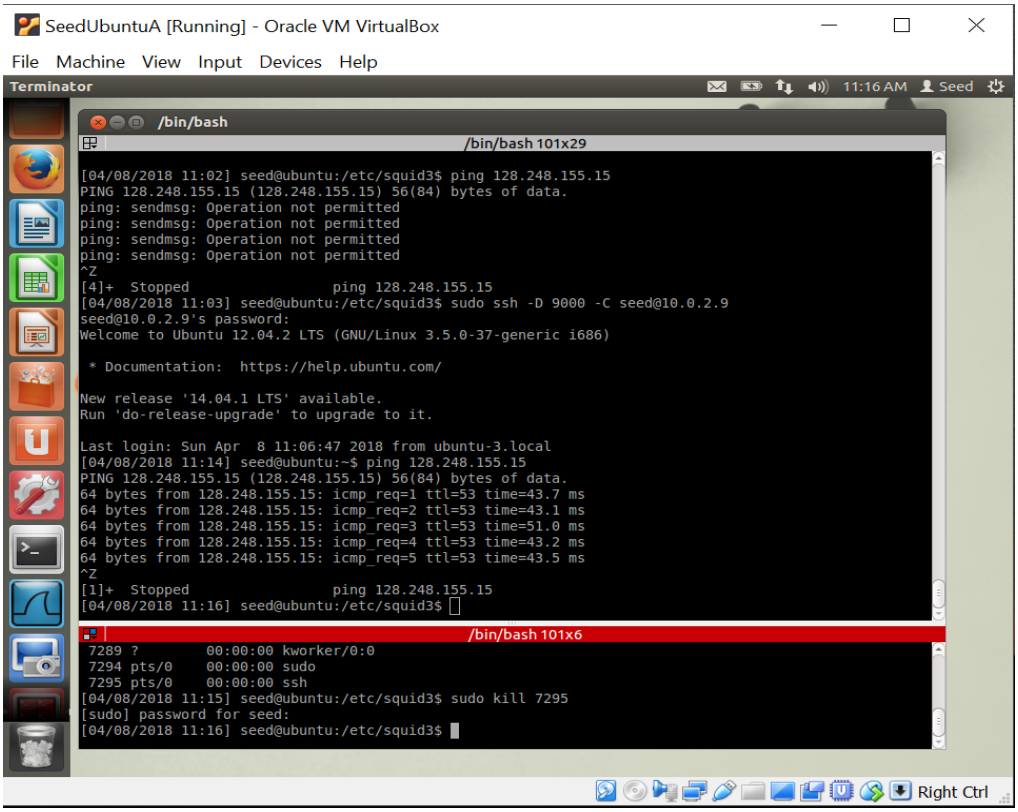
Connecting www.uic.edu via ssh, creating a tunnel between machine A and machine B with the help of the specified command.
$ ssh -D 9000 seed@10.0.2.9
This command is establishing the tunnel between machine A and machine B with the help of port #9000.
-D represents that this uses dynamic port forwarding on machine B.

ii) After you get the UIC's page, break the SSH tunnel, clear the firefox cache and try connecting again. Describe obs.

- In the following figure, we can oberve that once we are able to connect to machine B from machine A via ssh, we are able to connect to www.uic.edu . As we establish the connection, we can observe a 3 way handshake between both the machines and they establishes the connection using port 22. After connection establishment, they communicate via TCP protocol.



Obs: Able to reach out to www.uic.edu once we are able to ssh in to machine B from machine A.



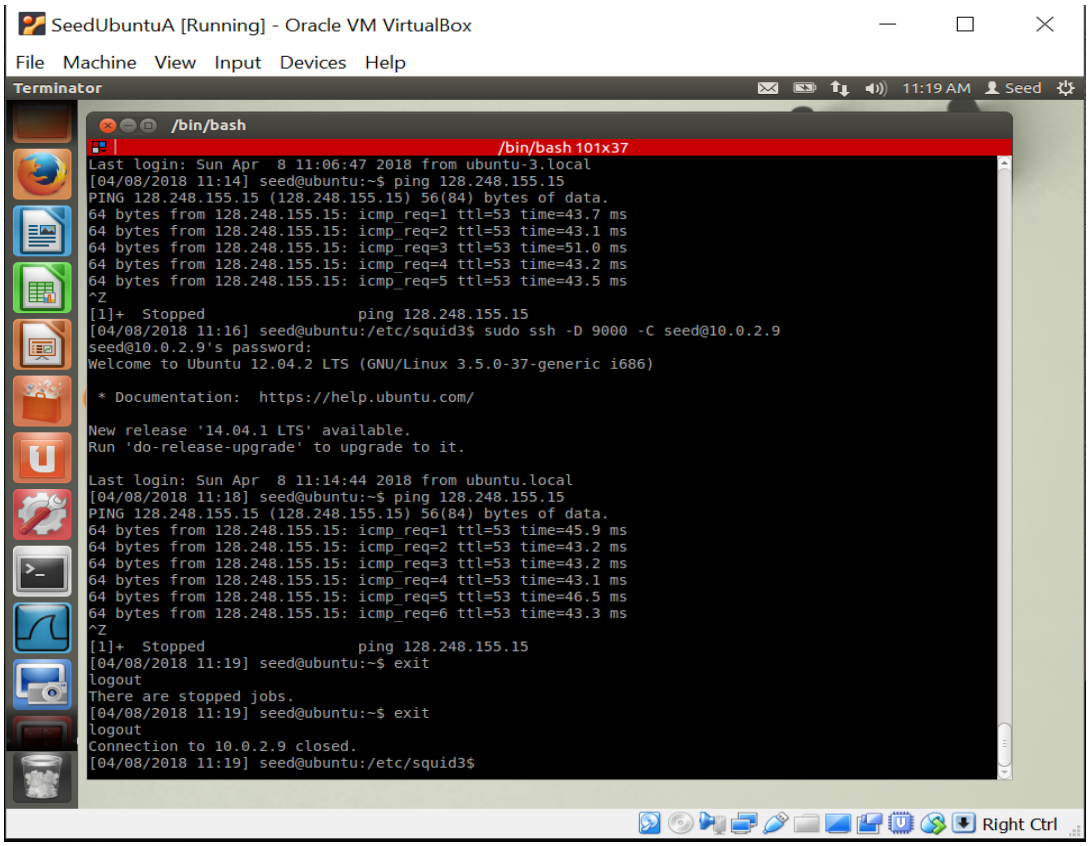
Obs: We can observe from command line too that we were able to ping the UIC's website once we are connected to machine B via ssh. Then we broke the ssh tunnel with the help of the kill command.
Once we break the tunnel, we are not able to reach the website again. Port 9000 refused the connection.

```
3171 39.747250   128.248.155.15   10.0.2.9        TCP     1514 80 → 50380 [PSH, ACK] Seq=232359 Ack=406 Win=32363 Len=1460 [TCP segment of a reassembl…
3172 39.747405   10.0.2.8         10.0.2.9        TCP       66 46308 → 22 [ACK] Seq=38146 Ack=1244954 Win=72704 Len=0 TSval=7799396 TSecr=7791721
3173 39.747413   128.248.155.15   10.0.2.9        TCP     1514 80 → 50380 [PSH, ACK] Seq=233819 Ack=406 Win=32363 Len=1460 [TCP segment of a reassembl…
3174 39.749374   10.0.2.8         10.0.2.9        TCP       66 [TCP ACKed unseen segment] 46308 → 22 [ACK] Seq=38146 Ack=1262650 Win=72704 Len=0 TSval…
3175 39.749800   10.0.2.8         10.0.2.9        SSHv2    130 Client: [TCP ACKed unseen segment] , Encrypted packet (len=64)
3176 39.749884   10.0.2.9         128.248.155.15  TCP       54 [TCP ACKed unseen segment] 50380 → 80 [FIN, ACK] Seq=406 Ack=235329 Win=65535 Len=0
3177 39.750086   10.0.2.9         10.0.2.8        SSHv2     98 Server: [TCP Previous segment not captured] , Encrypted packet (len=32)
3178 39.750182   128.248.155.15   10.0.2.9        TCP       60 [TCP Previous segment not captured] 80 → 50380 [ACK] Seq=235329 Ack=407 Win=32362 Len=0
3179 39.789504   10.0.2.8         10.0.2.9        TCP       66 [TCP ACKed unseen segment] 46308 → 22 [ACK] Seq=38210 Ack=1262682 Win=72704 Len=0 TSval…
3180 40.085675   10.0.2.8         10.0.2.9        SSHv2    114 Client: Encrypted packet (len=48)
3181 40.087694   10.0.2.8         10.0.2.9        SSHv2    834 Client: Encrypted packet (len=768)
```

Obs: Killing the connection via the kill command.

iii) Establish the connection again and connect to www.uic.edu and describe your obeservations.

- Following figures shows that we have established the connection again. It again performs a 3 way handshake and both the machines can now communicate via ssh.



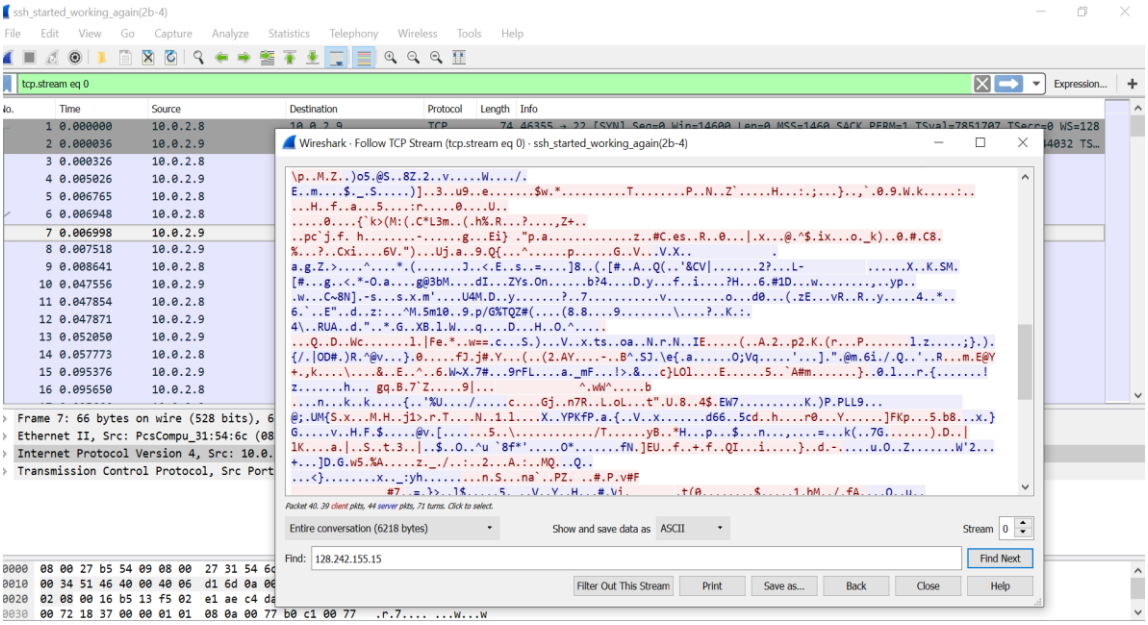
Obs: We are again able to connect from machine A to machine B via ssh.

```
3187 40.134661   10.0.2.9        10.0.2.8        SSHv2    642 Server: Encrypted packet (len=576)
3188 40.134893   10.0.2.8        10.0.2.9        TCP       66 46308 → 22 [ACK] Seq=39026 Ack=1263258 Win=72192 Len=0 TSval=7799493 TSecr=7791817
3189 40.136479   128.248.155.15  10.0.2.9        TCP       60 443 → 38197 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1460
3190 40.136506   10.0.2.9        128.248.155.15  TCP       54 38197 → 443 [ACK] Seq=1 Ack=1 Win=14600 Len=0
3191 40.136615   10.0.2.9        10.0.2.8        SSHv2    114 Server: Encrypted packet (len=48)
3192 40.136812   10.0.2.8        10.0.2.9        TCP       66 46308 → 22 [ACK] Seq=39026 Ack=1263306 Win=72320 Len=0 TSval=7799493 TSecr=7791818
3193 40.137405   10.0.2.8        10.0.2.9        SSHv2    162 Client: Encrypted packet (len=96)
```

Obs: Performing a 3-way handshake to get connected again. Once the connection was broken and then ammended again, 3 way handshake is again required.

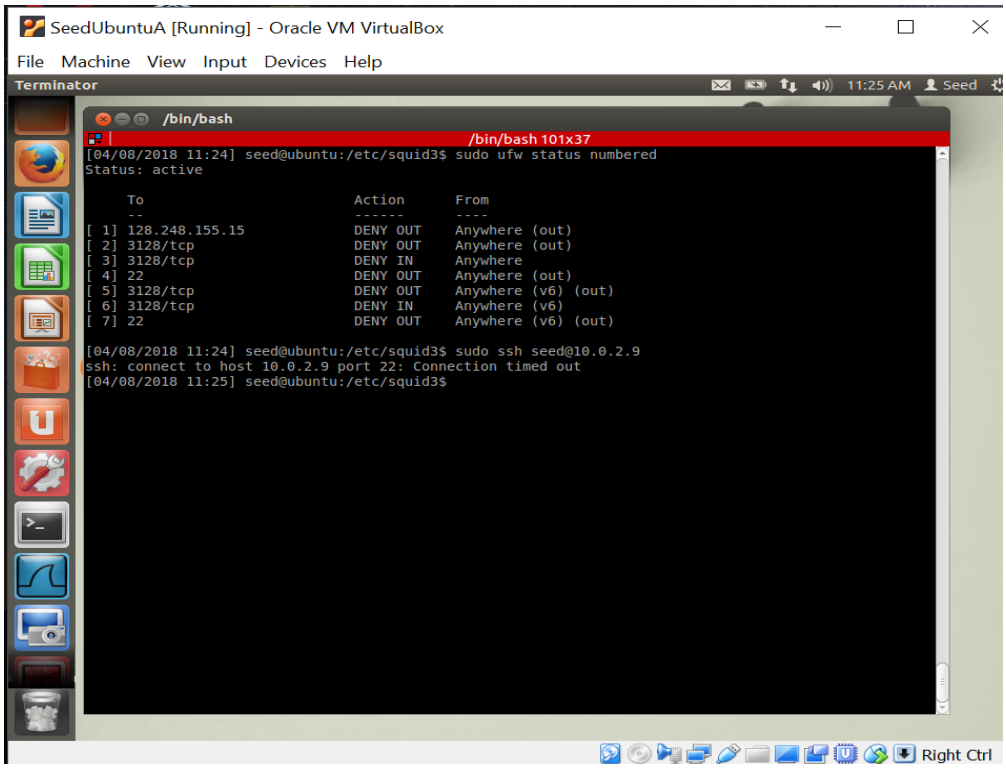
iv) Explain what exactly is happening. How you can bypass the egress filtering with the help of the ssh tunnel.Explain using the packets captured from the wireshark.

- From the following image, it is clear that how we are able to bypass the egress filtering. When we are creating a ssh tunnel, machine A is only sending the information about the L2, ip address information (L3), and TCP (L4) information. Everything else is encrypted because of which it is able to bypass the ufw rule.

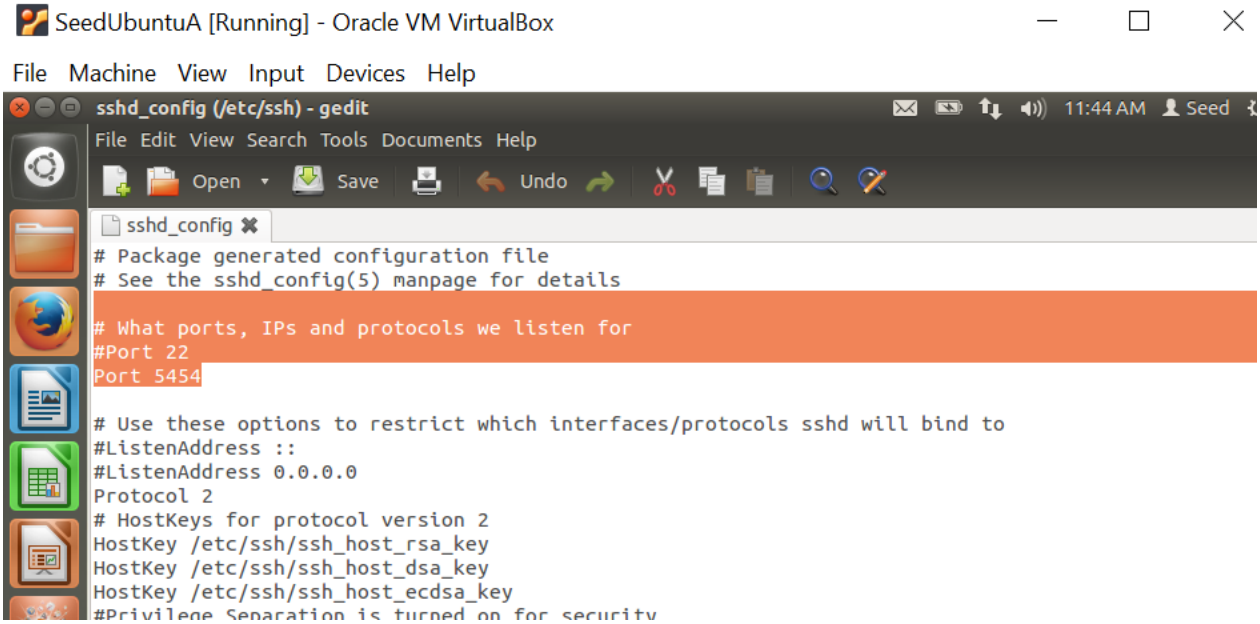


v) If ufw blocks TCP port 22, which is the port used by SSH and can you still set up SSH tunnel to evade egress filtering ?

- We can change the default port number inside the “sshd_conf” in order to still evade the egree filtering. Following images will depict how to do that.



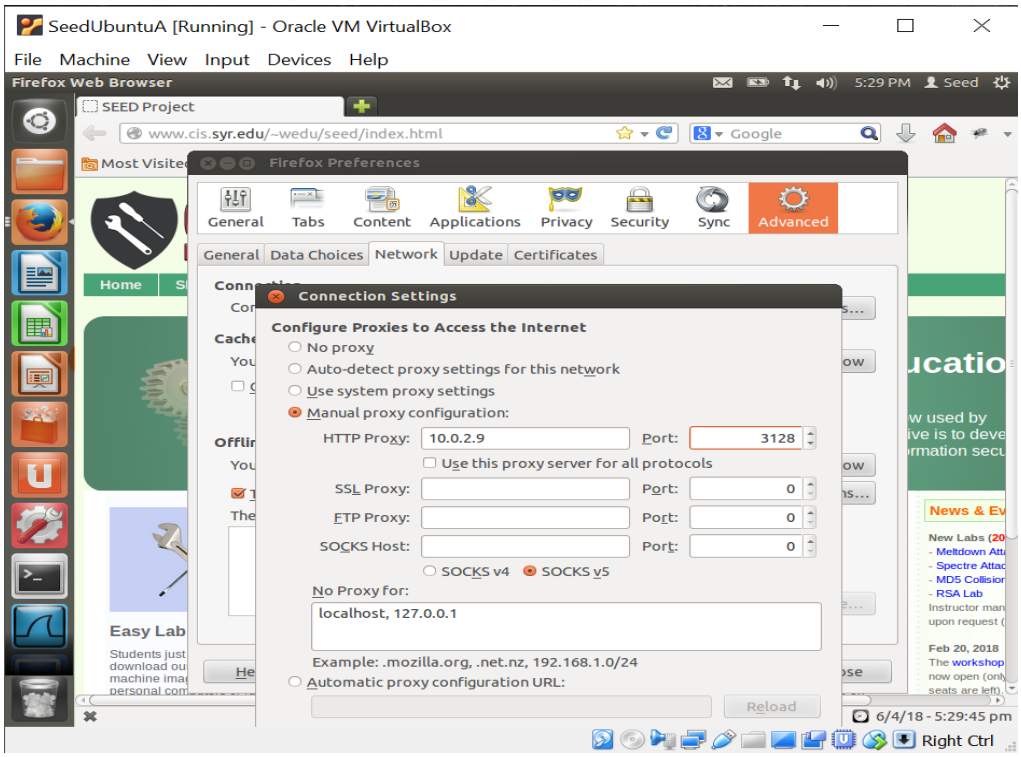
Obs: First we define a rule in ufw on machine A to block ssh to machine B. (refer to Rule #4) . Then we tried to ssh into machine B from machine A. As expected, connection timed out error confirms that we won’t be able to ssh to B.



Obs: Above image shows you how and where to change the default port inside the sshd_conf file. You can see the path too. In my case, I have change to some random port 5454 and then opened that port in the terminal. After following these steps, I was able to ssh again into the machine B. This confirms that we can ssh into machine B by changing the port # thereby evading the egress filtering.

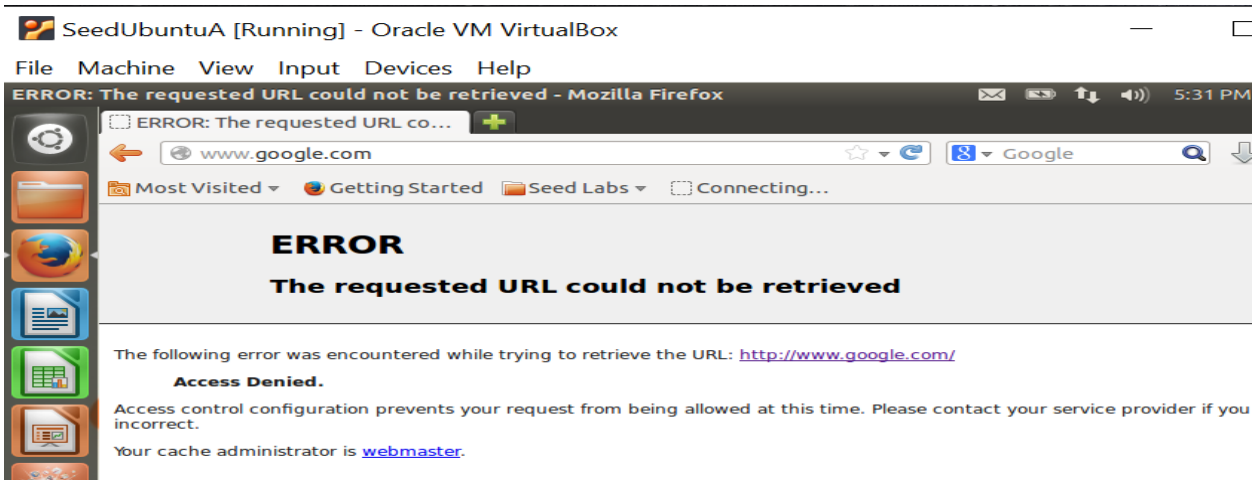
Task 3: Web Proxy (Application Firewall)
ufw (Uncomplicated Firewall) that was being used in the Task2 inspects the packets for the protocols that are used upto Layer 4 (Transport Layer). Web Proxy or the application layer firewall also inspects the contents of the packets. For this task, we are going to implement web application firewall with the help of “Squid3”- a free web proxy firewall in Linux.

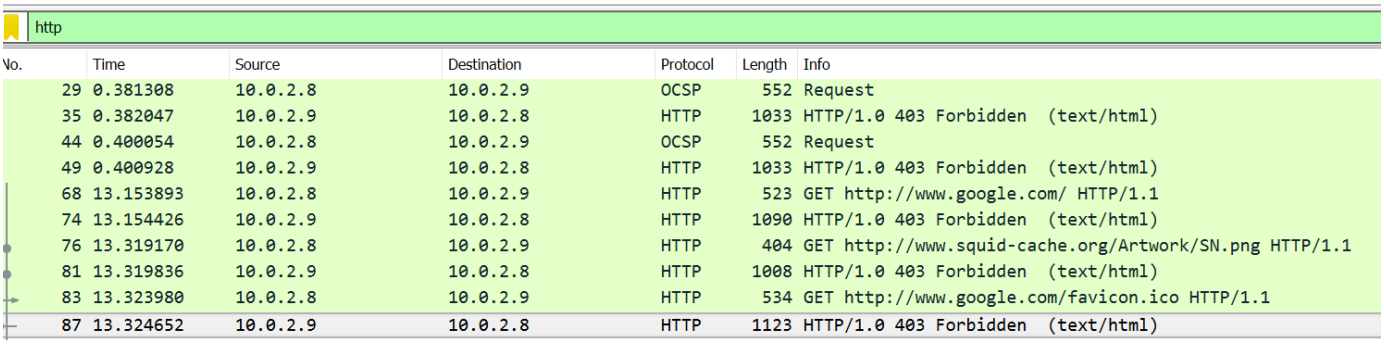
Setup: 2 Virtual machines have been set up for this task. Machine A’s browsing behavious has to be restricted and machine B is where we will run the web proxy. Machine A’s browser should always use the web proxy server on machine B. Following are the settings for that.

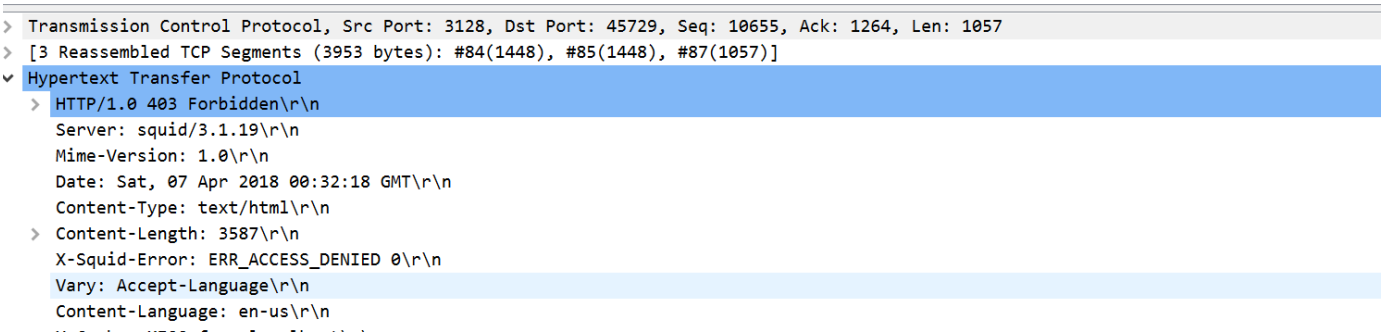


i) Try to visit some web sites from machine A's firefox browser and describe your observation.

- On trying to visit websites after enabling squid3 on our virtual machine B, we can observe that we were not able to open any website. This is so because by default, all the external websites are blocked by the web proxy. This can be checked after browsing through the squid config file.



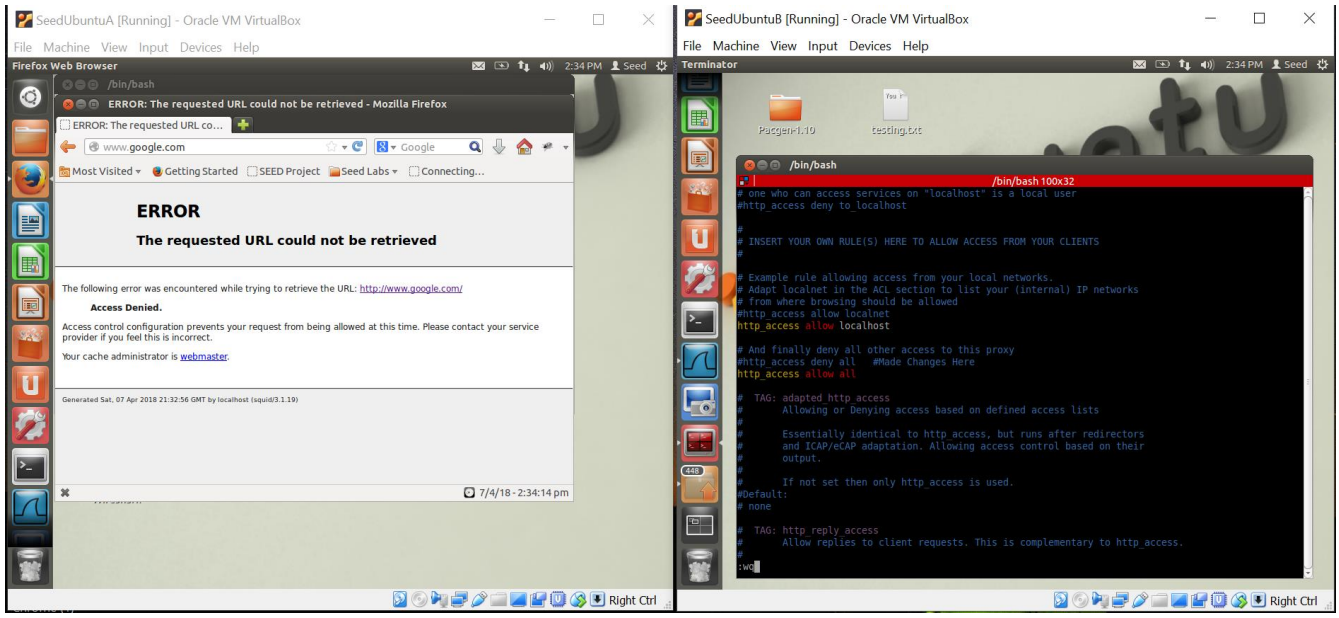




Obs: From this packet capture, it is clear that every website that we require to visit has been blocked by default and is forbidden by squid. Error: ERR_ACCESS_DENIED.

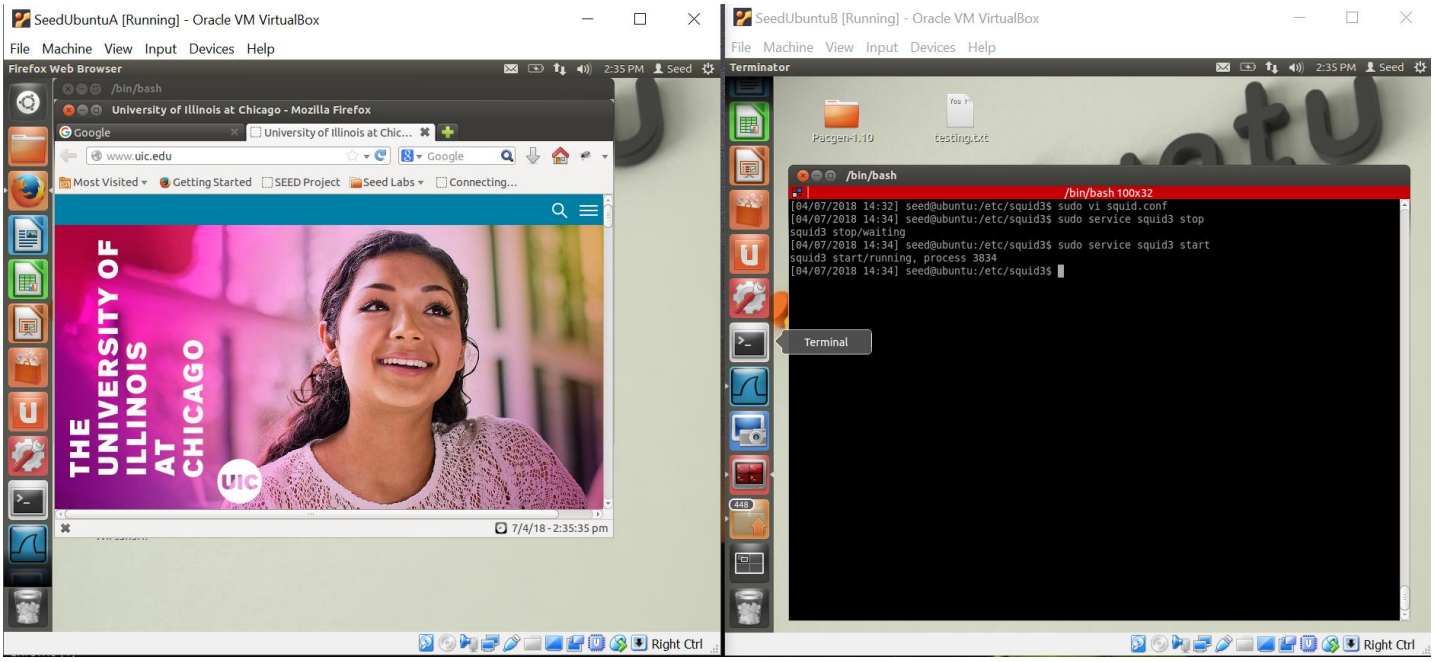
ii) Take a look at the config file and describe what rule must have caused that.

- After openinig the config file in the “vi” editor, we can observe that the “http_access deny all” was the rule that was denying all the websites from getting open in the machine A.



iii) Make changes to the config file, so that all the web sites are allowed.

- We can change the rule from “http_access deny all” to “http_access allow all” in order to allow the web proxy running on machine B to let open the websites on machine A.

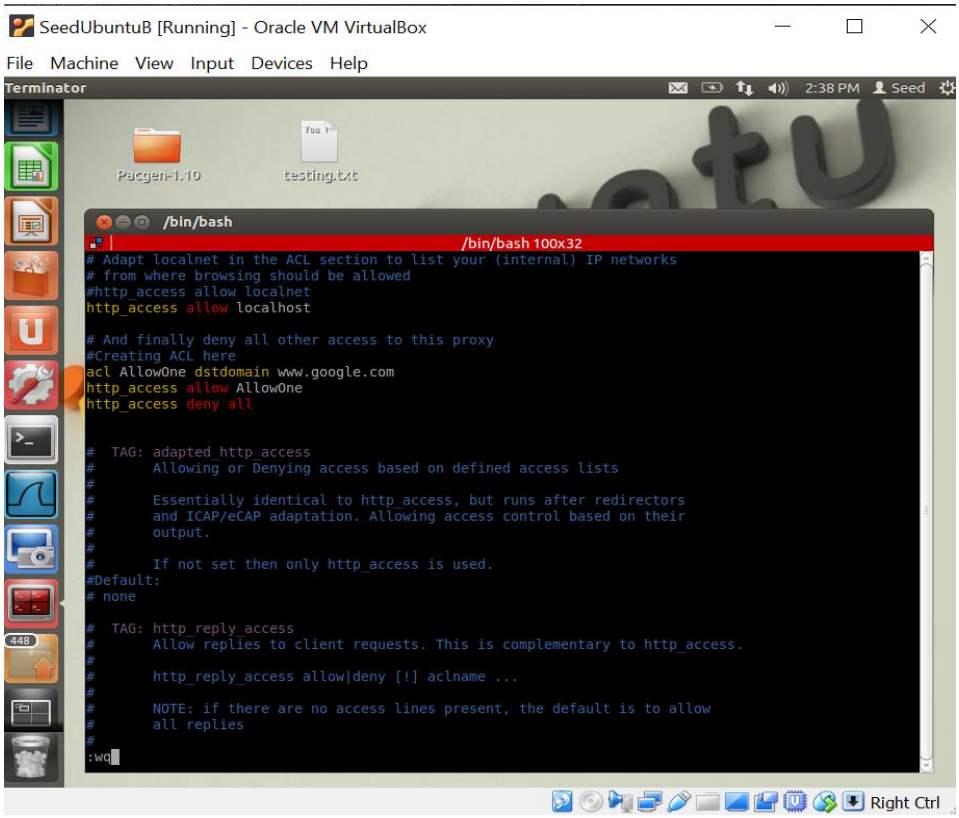



```
417 163.925037    10.0.2.9          128.248.155.15    TCP      54 48633 → 80 [ACK] Seq=381 Ack=16104 Win=40880 Len=0
418 163.925086    128.248.155.15    10.0.2.9          TCP      60 80 → 48633 [FIN, ACK] Seq=16104 Ack=381 Win=32388 Len=0
419 163.925192    10.0.2.9          128.248.155.15    TCP      54 48633 → 80 [FIN, ACK] Seq=381 Ack=16105 Win=40880 Len=0
420 163.925285    10.0.2.9          10.0.2.8          HTTP    722 HTTP/1.0 200 OK  (text/html)
421 163.925527    128.248.155.15    10.0.2.9          TCP      60 80 → 48633 [ACK] Seq=16105 Ack=382 Win=32387 Len=0
422 163.925541    10.0.2.8          10.0.2.9          TCP      66 44386 → 3128 [ACK] Seq=791 Ack=16951 Win=42368 Len=0 TSval=2307225 TSecr=2299537
423 163.943130    10.0.2.8          10.0.2.9          HTTP    415 GET http://www.uic.edu/_assets/005954d95b0b1aaf329517897038b779.jpg HTTP/1.1
```

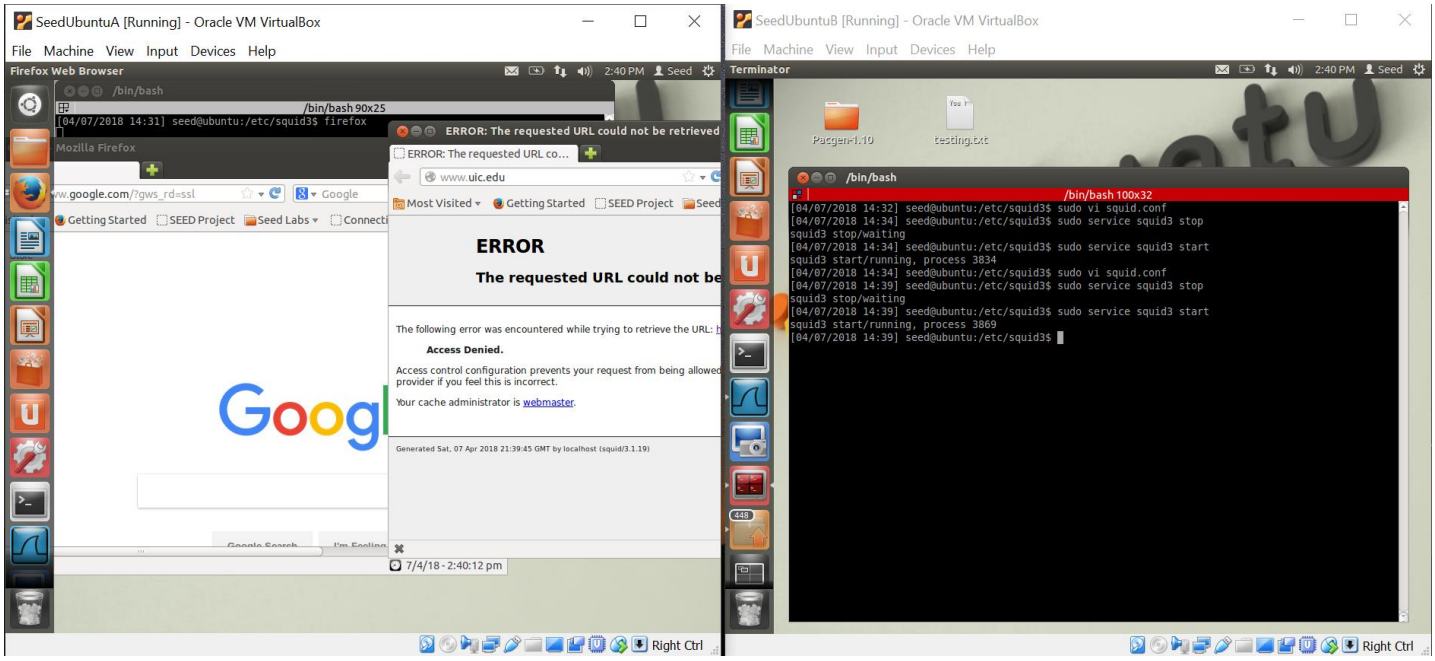
Obs: Packet captures shows that http connection has been established again and every website is now allowed to be visited by the user.

iv) Make changes to the configuration file, so only access to the google.com is allowed.

- This can be achieved with the help of the access list. The procedure of how to define this acl rule is mentioned in the squid3 config file itself. First you define a “name” for your acl. Then in our case we want to allow a particular website to be accessible, so “dstdomain” – destination domain and then you allow that rule and you can insert deny all at the end.



Obs: We can define an access rule in the squid config file itself to allow machine A to visit only google.com.



Obs: We can clearly observe that machine A is able to visit google.com but is unable to visit any other website such as uic.edu. Once you have set an access rule in your squid3 config file, don’t forget to restart squid service. It is necessary for the new rule to start working.

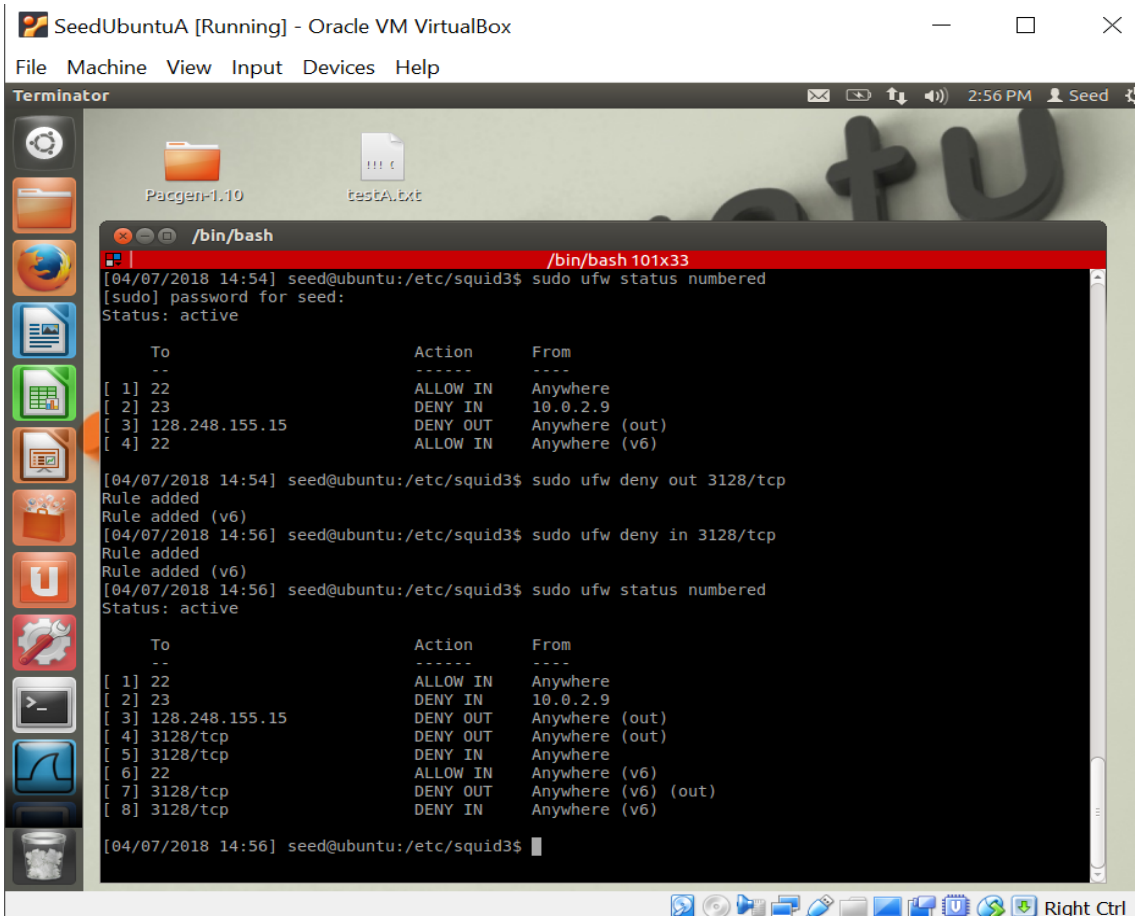
```
7096 420.335495    10.0.2.8          10.0.2.9          HTTP      445 GET http://www.uic.edu/ HTTP/1.1
7099 420.335932    10.0.2.9          10.0.2.8          HTTP     1013 HTTP/1.0 403 Forbidden  (text/html)
7103 420.546804    10.0.2.8          10.0.2.9          HTTP      401 GET http://www.squid-cache.org/Artwork/SN.png HTTP/1.1

> HTTP/1.0 403 Forbidden\r\n
  Server: squid/3.1.19\r\n
  Mime-Version: 1.0\r\n
  Date: Sat, 07 Apr 2018 21:39:45 GMT\r\n
  Content-Type: text/html\r\n
> Content-Length: 3477\r\n
  X-Squid-Error: ERR_ACCESS_DENIED 0\r\n

0000  48 54 54 50 2f 31 2e 30  20 34 30 33 20 46 6f 72   HTTP/1.0  403 For
0010  62 69 64 64 65 6e 0d 0a  53 65 72 76 65 72 3a 20   bidden.. Server:
```

Obs: From this packet capture, it is clear that only google.com is now allowed to visit. Access to rest of the websites have been blocked.

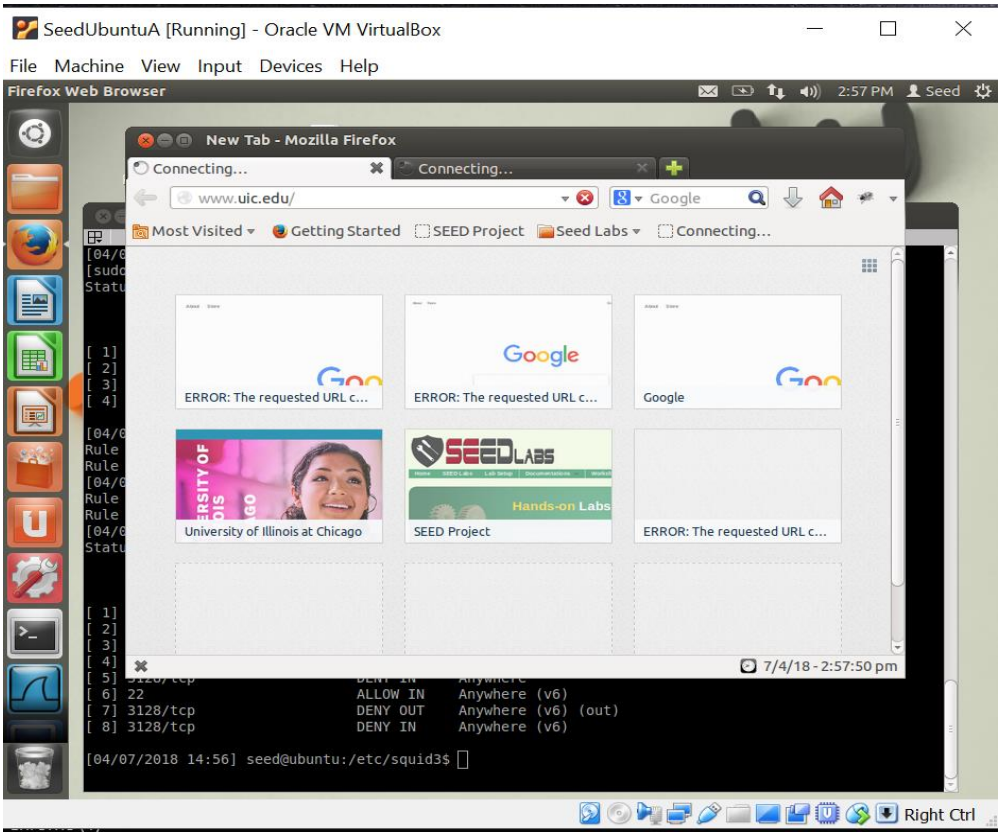
Task 3b: Using web proxy to evade Firewall

If ufw blocks the TCP port 3128, can you still use the web proxy to evade the firewall ?

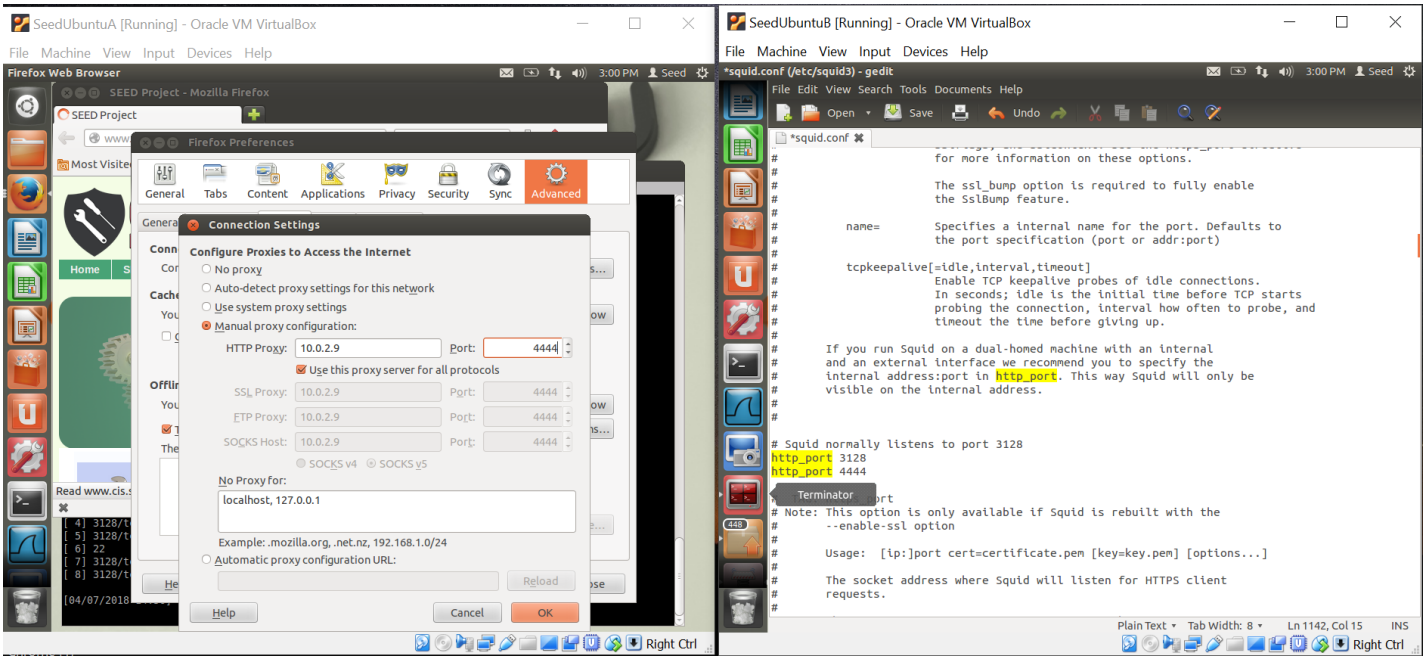
- Lets block port 3128 and try to evade the firewall.



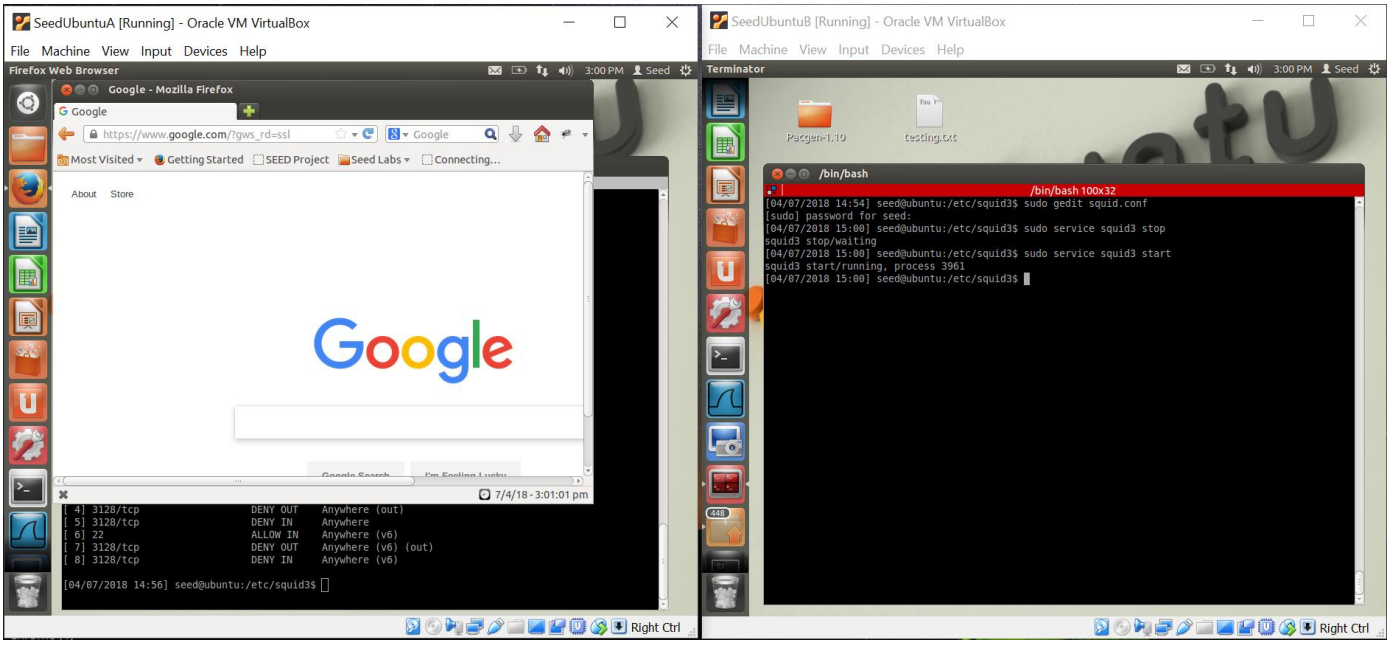
Obs: Port 3128 has been blocked with the help of ufw using command: “sudo ufw deny out 3128/tcp”.



Obs: We can observe that we are not able to reach either uic.edu nor google.com. This means we have successfully blocked port 3128.



Obs: Squid web proxy can listen to more than one ports. Thus we can evade the firewall by changing the port number in the proxy settings as well as in the squid config file. In my case, I have changed it to 4444.

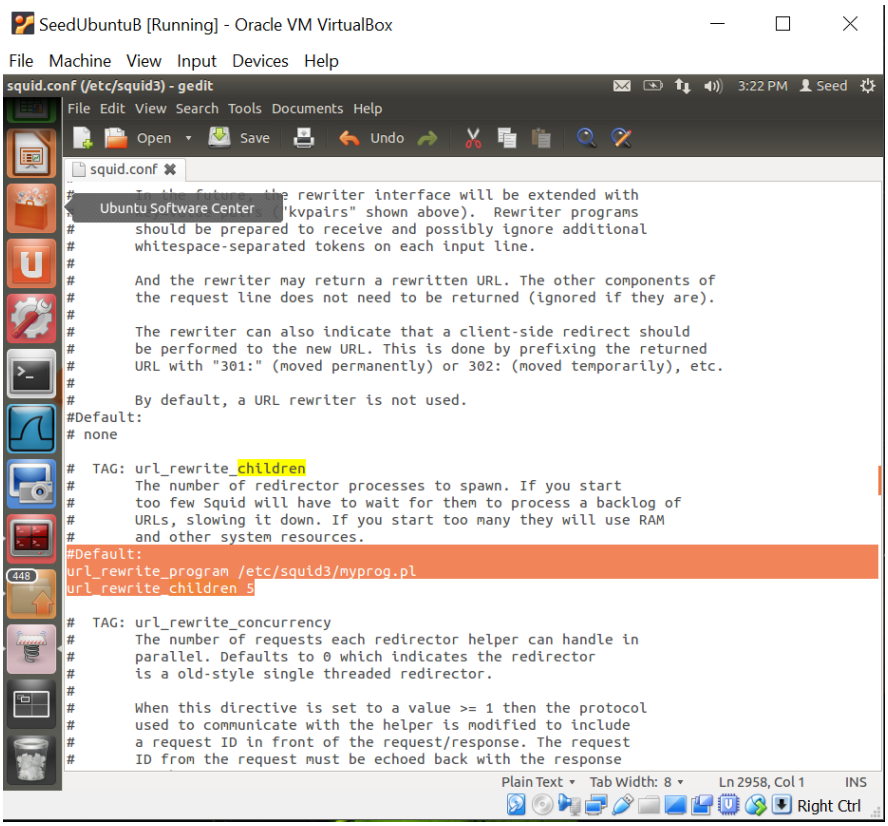


Obs: When we restarted squid, we could see that we are able to communicate via port # 4444. So even though port # 3128 was blocked by ufw firewall, we were able to evade it by entering a different port number.

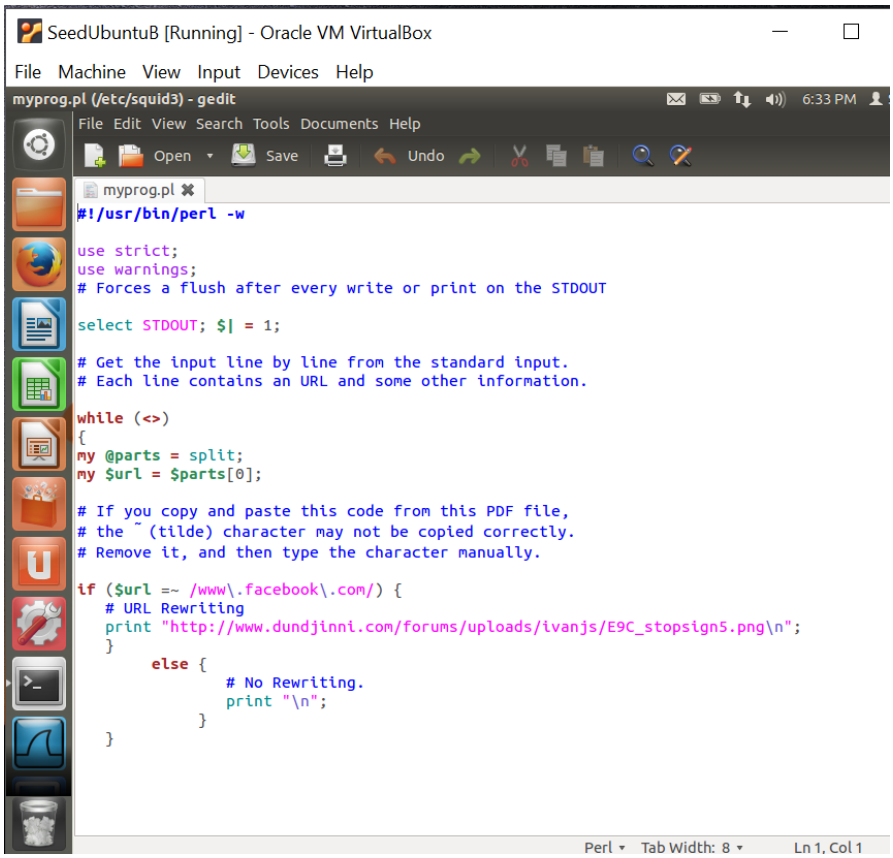
Task 3c: URL Rewriting/Redirection

Not only squid can block the websites but it can also re-direct them. Everytime user tries to visit facebook.com, try to redirect them to some other website or you can also block them.

Setting up the resources for this task as stated in the document.



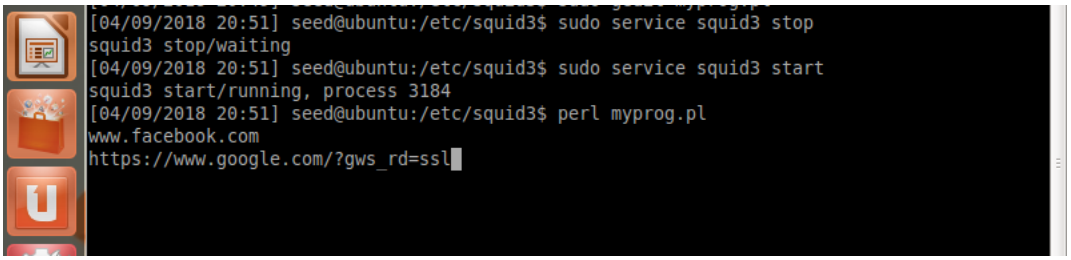
Obs: Include the name of the program that you have created for the redirection inside the squid config file so that squid gets connected to the redirective program.



Obs: Creating a “myprog.pl” program as stated in the document inside the same directory as that of squid config file.

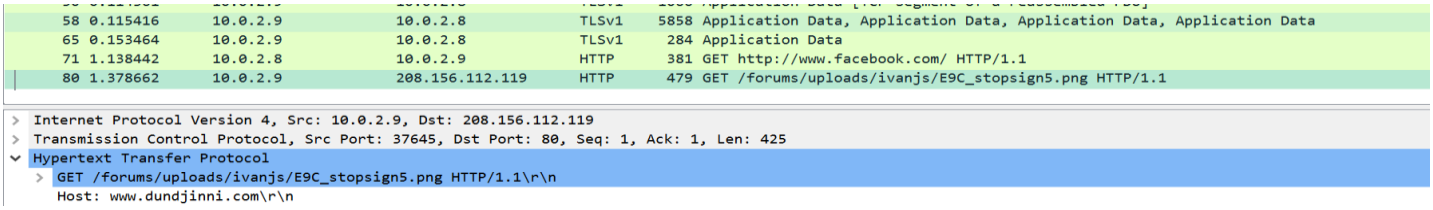
i) Please describe what the above program does ?

- After making “myprog” file, I tried to run it on terminal itself. On running the file, whenever I tried to write facebook.com, the output was the name of the rewritten url. In my case, it was google.com. Whenever I tried to input any other website’s name after running the perl command, the code was just showing a blank line as an output.

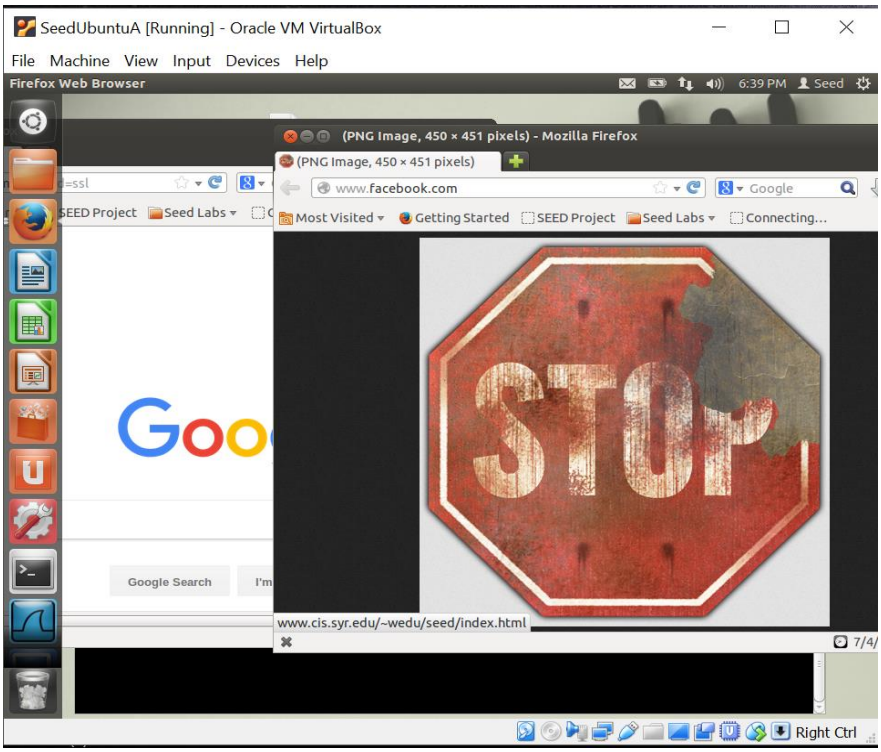


ii) modify the given program so it replaces all the Facebook pages with a page that shows a Stop sign.

- The above mentioned “myprog.pl” code is the one which can change the facebook.com with a page displaying a Stop Sign.



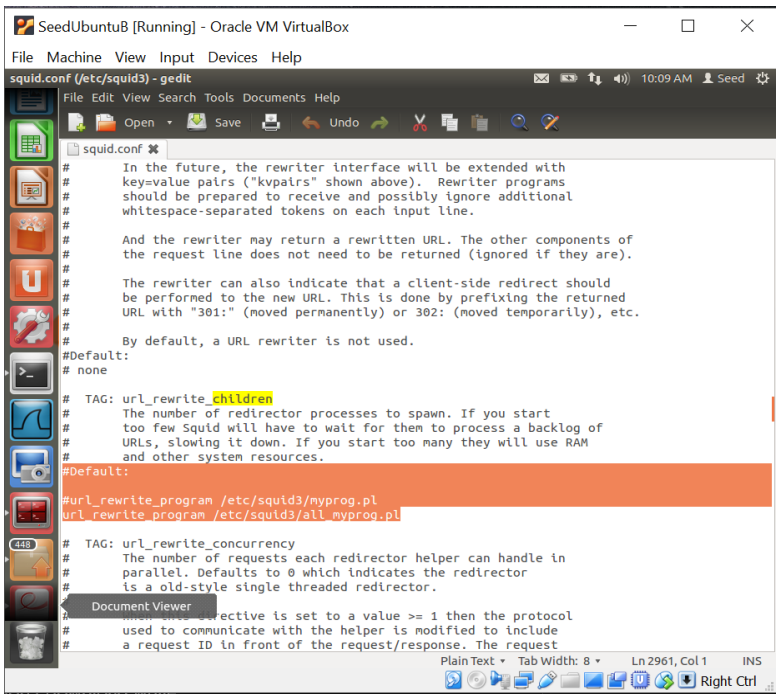
Obs: From this packet capture, we can observe that when we entered facebook.com, we got address of the stop sign image.



Obs: From the above mentioned image, we can see that facebook.com has been replaced by a big stop sign. However when we are trying to reach any other website, we are able to access that website normally. Only facebook.com has been redirected to a stop sign image.

iii) Modify the perl program so it replaces all the images (of any kind – gif, jpeg, png, etc) inside any page with a picture of your choice.

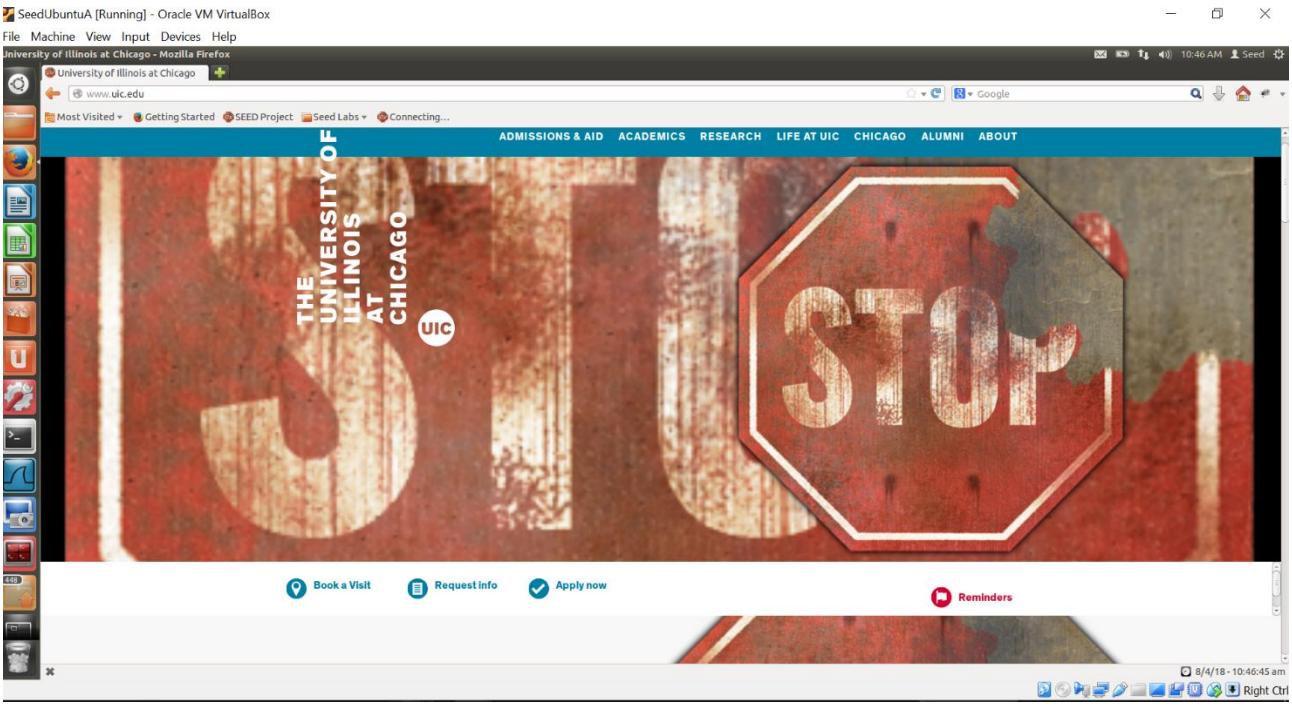
- To perform this task, you can either create a new file like the one you created for the previous case, or you can rewrite your code in the previous file only. In either of the case, you will need to restart the squid3 config file in order to make it work.



Obs: I have created an entirely new file to implement this part by the name “all_myprog.pl”. You need to insert the name of the file again in the squid3 config file and restart the service in order for it to work.

```
while (<>)
{
 chomp $_;
 if($_=~ /(.*\.(bmp|ico|jpeg|jpg|gif))/i) {
   print "$image\n";
  }}
```

Obs: There is a very minute change in the code from the previous file and it has been described in the above image.



Obs: In this image, we can observe that every image has been replaced with the stop sign image.

Task 3d: A real URL Redirector

- Download SquidGuard – compile and install
- Download Oracle Berkeley DataBase
- Download a blacklist

For this task, we just have to download, install and study about the SquidGuard. This has been done but no further exercise has been mentioned.