# Linux Firewall Exploration Lab

**35 Points**

**Due Date: Start of Lesson 12**



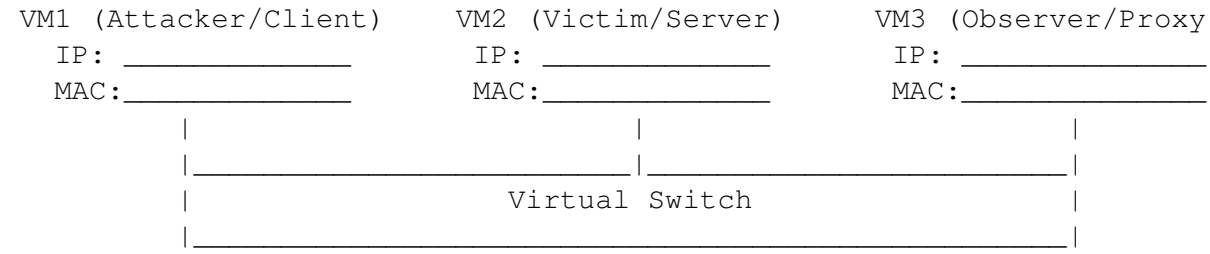
## 1 Overview

The learning objective of this lab is for students to gain insight into how firewalls work by working with firewall software and implementing a simplified packet filtering firewall. Firewalls have several types; in this lab, we focus on *packet filter* or *stateless* firewalls.

Packet filters act by inspecting the individual packets; if a packet matches the filter's set of rules, the packet filter will either drop the packet or foward it, depending on what the rules state. Packet filters are usually *stateless*; they filter each packet based only on the information contained in that packet, without paying attention to whether a packet is part of an existing stream of traffic. Packet filters often use a combination of the packet's source and destination address, protocol, and port numbers.

## 2 Lab Setup

You should start by loading up your three Ubuntu systems. Log in and verify the IP and MAC address of each. Write each system information below for future reference. Based on the set-up from lab 3, Attacker's IP should be 10.172.x.12, Victim should be 10.172.x.10, and Observer should be 10.172.x.11.




Since our the roles of ourVMs are different in this lab, we are going to change their names to alleviate confusion. We will be renaming Attacker to VM1_Client, Victim to VM2_Server, and Observer to VM3_Proxy. Make sure to prefrence each VM name with your first initial last name (so John Doe's VM1_Client name would be jdoe_VM1Client). **Also, make sure to snapshot your vms prior to starting this lab so you can revert them back to their current state in future labs.** In case you forgot, you change the hostname of your VMs by editing both `/etc/hosts` and `/etc/hostname`. After editing these files, restart your computer with `sudo shutdown -r now`. Alternatively, you can issue `sudo hostnamectl set-hostname <New hostname>` and edit `/etc/hosts`, and then open a new terminal. **NOTE: Failure to edit** `/etc/hosts` **is the reason why some students' VMs hang while issuing the** `sudo` **command.**

### 2.1 Before You Begin

This lab is much shorter than the DNS lab by design. The majority of the points will come from how well you documented each task. To earn full credit, submit a detailed lab report to describe what you have done and what you have observed:

- Ensure you describe all processes and results, showing that you completed the tasks, in a detailed lab report. This should also include descriptions explaining the "why" behind these actions
- Provide snippets of Wireshark captures that you implement to demonstrate firewall operation

**Look at the Rubric and Submission requirements to get a sense of what is important!**

## 3 Lab Tasks

### 3.1 Task 1a: Using Firewall - Installing Services

In order to test our firewall functionality, we need to have some additional services on our VM2_Server to test. Using VM2_Server, download and install telnet using the command:

```
sudo apt-get install telnetd
```

After this command completes, restarting your networking services by typing:

```
sudo service network-manager restart
```

Repeat these steps on VM1_Client.
Finally, you also need to download and install openssh on VM3_Proxy using the command:

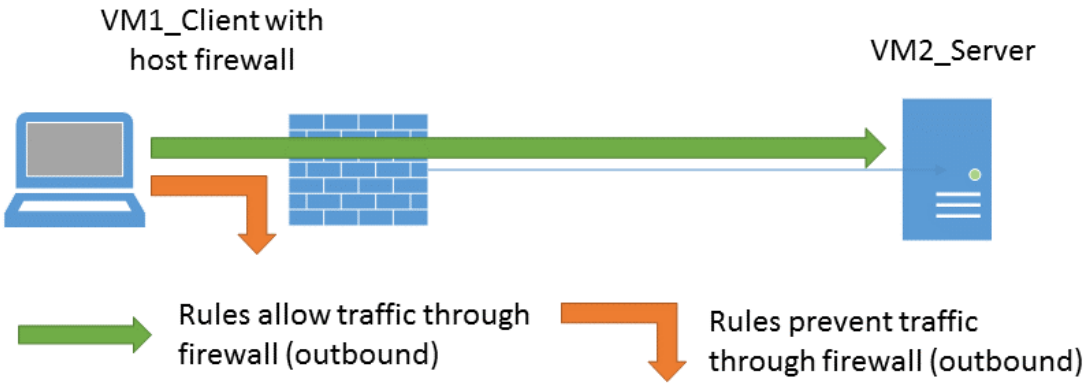
```
sudo apt-get install openssh-server
sudo service network-manager restart
```

### 3.2 Task 1b: Using Firewall - Testing Services

`Linux` has a tool called `iptables`, which is essentially a firewall. It has a nice front end program called The Uncomplicated Firewall, or `ufw`. In this task, the objective is to use `ufw` to set up some firewall policies, and observe the behaviors of your system after the policies are implemented. You need to start your first two VMs (VM1_Client & VM2_Server). You run the firewall on your VM1_Client. Basically, we use `ufw` as a personal or *host-based* firewall in this task. You can find the manual of `ufw` by typing `"man ufw"` or search for it online. It is pretty straightforward to use. Please remember that the firewall is not enabled by default, so you should run `sudo ufw enable` to specifically enable it. Order of rules is important! The firewall will act on the first match. For this task, you will use VM1_Client and VM2_Server. VM1_Client will host the firewall so that your network connection will look like the following setup:

1. First, check the default firewall configuration on VM1_Client: `sudo iptables -L -n`

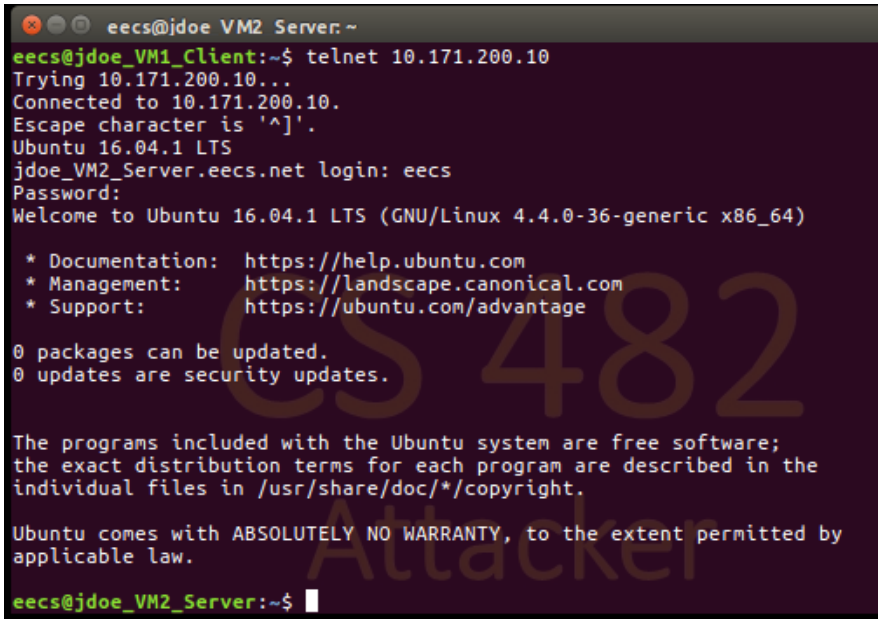
> **Question 1:** What are your default iptables firewall rules? Explain the difference between input, forward and output rules.




2. Go to default policy file `/etc/default/ufw`. If `DEFAULT_INPUT_POLICY` is `DROP`, please change it to `ACCEPT`. Otherwise, all the incoming traffic will be dropped by default. After changing the `DEFAULT_INPUT_POLICY` to `ACCEPT`, reset your firefall with `sudo ufw reset`.

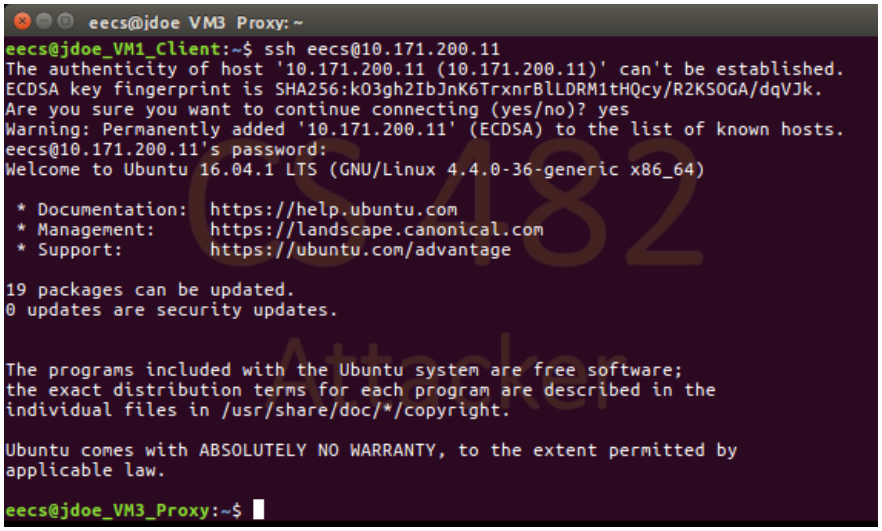
3. Now, enable the UFW: `sudo ufw enable`.

4. Try telnetting from VM1_Client to VM2_Server. To do this, at the command line use the command `telnet <target_IP>`. It will take a minute to connect. Eventually it will ask you to log in to the target system. Use your standard `eecs/scee` credentials. You should be able able to complete a telnet connection. Once you have telnetted to VM2_Server there are a few key things you should notice: If you successfully changed all the hostnames, you should now see that your command prompt displays eecs@`user`_VM2_Server. Typing in `whoami` should return `eecs` still as we have not changed any account name on any of the systems. To double verify that you are in fact telnetted into the correct system, run `ifconfig`. It should provide you with the ip information for VM2_Server. If this is not the case, you have failed somewhere in establishing your telnet connection and should try again. An example of a successful telnet session is shown below:



**Question 2:** Why is using telnet not recommended as a best practice? Be specific.

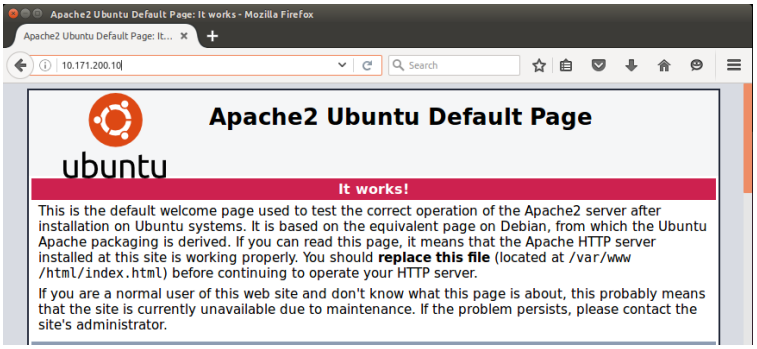
5. Now try SSHing from VM1_Client to VM3_Proxy. To do this, open a new `bash` terminal and at the command line use the command `shh eecs@<target_IP>`. After a couple seconds, you will be prompted that the authenticity of host can not be established and asked if want to continue connecting - type yes. If you paid attention during the earlier openssh server install, the install created several SSH keys including the SSH ECDSA key. A careful user would make sure the SSH key created by the server matches the key you receive when connecting to the server. After clicking yes, you will be prompted to enter the password for eecs. At this point, will be connected to an SSH prompt on VM3_Proxy very similar to the telnet prompt above. Run all the same tests as above to verify your connection. An example of a successful SSH session is shown below:




**Question 3:** Why is using ssh recommended in place of telnet?

6. The last check before implementing firewall rules is to check the webpage hosted at your VM2_Server. Inside of VM1_Client, open up Firefox. In the address bar, type in `10.172.xxx.yyy/index.html`, where xxx.yyy is the ip of your VM2_Server. You should have a page pop up that simply says "It Works!". If you do not get this, verify you have done everything correctly. You should also verify that apache2 is running on your VM2_Server (you disabled in on VM1_Client, but it should be on by default on VM2_Server). If it is disabled, re-enabled it using the steps in Section 7 of Lab 1 or by using the below command to set apache2 to start on startup and then restart your network manager.

```
sudo update-rc.d apache2 default
sudo service network-manager restart
```




### 3.3 Task 1c: Using Firewall - Implementing Rules

1. On your VM1_Client system, set up the firewall to prevent VM1_Client from telneting to VM2_Server. Use:
   `sudo ufw deny out from <Client_ip> to any port 23.`
   Now test to verify that you can no longer telnet out of VM1_Client to VM2 Server using the same steps you followed previously.

2. Now prevent VM2 from telneting to VM1. Use:
   `sudo ufw deny in from <Server_ip> to <Client_ip> port 23`. Again, check to verify that this rule has been applied by attempting to telnet to VM1_Client from VM2_Server.

3. Prevent VM1 from visiting an external website on VM2:

   `10.172.xxx.xxx/index.html`

   If you were to do this for a real website (like facebook.com), keep in mind that most modern web servers have multiple IP addresses so a simple IP based blacklisting would not work. Test this new rule and make sure it works. Note: You will likely have to clear the browser cache to verify your rule applied correctly if you have previously visited the website. To clear the cache, click the three bars in the top right corner (Open menu), then history, then clear recent history, and ensure Cache is selected (just leave all selected) then hit clear now.

**Question 4:** What was the command you used to block this webpage hosted at VM2_Server?

**Question 5:** Document your steps using wire capture or appropriate methods

### 3.4 Task 2: Evading Egress Filtering

In task 1 we blocked telnet and applied egress filtering to prevent users from accessing certain websites/applications. Many companies and schools enforce egress filtering, which blocks users inside of their networks from reaching out to certain websites or Internet services. They do allow users to access other web sites. In many cases, these types of firewalls inspect the destination IP address and port number in the outgoing packet. If a packet matches the restrictions, it will be dropped. They usually do not conduct deep packet inspections (i.e., looking into the data part of packets) due to performance reasons. In this task, we show how such

egress filtering can be bypassed using a tunnel mechanism. There are many ways to establish tunnels; in this task, we only focus on SSH tunnels. From task 1, you should have completed:

1. Firewall blocking Telnet from VM1_Client to VM2_Server
2. Firewall blocking Telnet from VM2_Server to VM1_Client
3. Firewall blocking web access to page hosted on VM2_Server at 10.172.xxx.xxx/index.html

In addition to setting up the firewall rules, the following commands will be useful for testing implementation going forward:

```
$ sudo ufw enable           // this will enable the firewall.
$ sudo ufw disable          // this will disable the firewall.
$ sudo ufw status numbered  // this will display the firewall rules.
$ sudo ufw delete 3         // this will delete the 3rd rule.
```

**Task 2.a: Telnet to VM2_Server through the firewall using a tunnel** To bypass the firewall, we could establish an SSH tunnel between VM's 1 and 2, so all the telnet traffic will go through this tunnel (encrypted), evading inspection. However, it is unlikely that you will have access to a random distant end server which would allow you to directly establish a encrypted tunnel with them. Usually, if someone is going to attempt to circumvent a firewall, they will connect to a system they can control outside of the network. This system will act as a proxy to the user, allowing them to appear and operate as if they were outside of the company's firewall. The following command establishes an SSH tunnel between the localhost's (VM1_Client) port 8000 and VM3_Proxy port 22. When packets come out of VM3's end, it will be forwarded to VM2's port 23 (telnet port). To the user at VM1_Client, it will appear as if they had just telnetted to VM2_Server!

```
$ ssh -L 8000:VM_2_IP:23  eecs@VM_3_IP
```

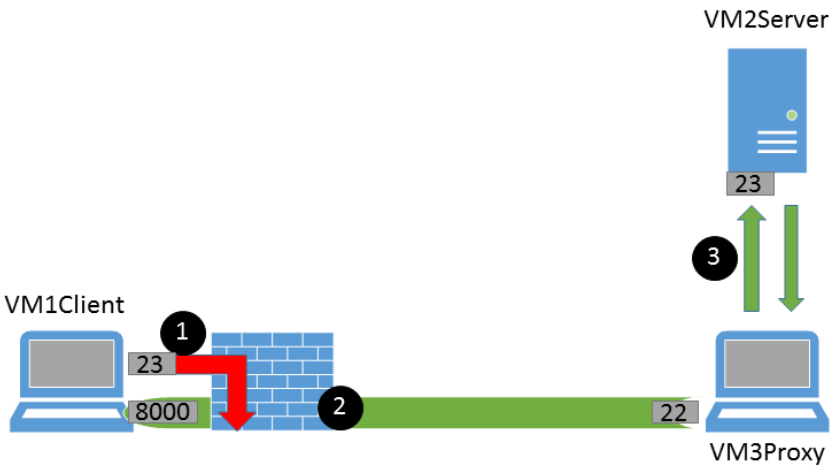



Figure 1: SSH Tunnel Example

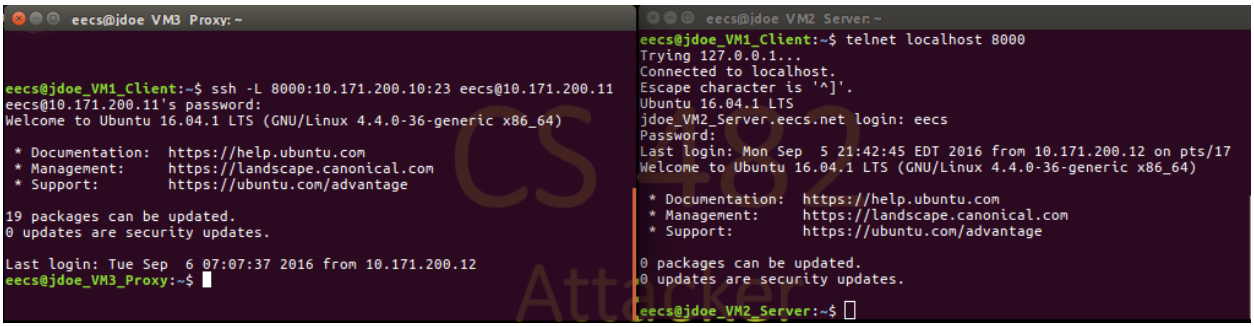
In bullet 1 in the figure, the user attempts to telnet to VM2_Server, but his company's firewall blocks his connection. In bullet 2, the same user has established an encrypted tunnel to a proxy server, which is set up to forward all requests to VM2_Server (Bullet 3). Now the user can telnet to VM2_Server, through the tunnel, effectively bypassing his company firewall.

After establishing the above tunnel, leave the command window open (note that this command window now has a prompt showing it belongs to the VM3_Proxy.) You will now have to open up a second command terminal in VM1_Client to execute your telnet to the VM2_Server:

```
$ telnet localhost 8000
```

SSH will transfer all your TCP packets from your end of the tunnel (localhost:8000) to VM3, and from there, the packets will be forwarded to VM2_Server:23. Replies from VM2_Server will take a reverse path, and eventually reach your telnet client. This results in you telneting to VM2_Server despite a firewall in place to block this action!

A summary of the two command windows is shown below for reference:




**Question 6:** Please describe your observation and explain how you are able to bypass the egress filtering. You should use Wireshark to see what exactly is happening on the wire, and include your Wireshark capture (from VM3_Proxy) and use line numbers in your explanation. Close both sessions once complete.

**Task 2.b: Connecting to Facebook using SSH Tunnel.** To achieve this goal, we can use the approach similar to that in Task 2.a, i.e., establishing a tunnel between your localhost:port and VM3_Proxy, and ask VM3_Proxy to forward packets to a website like Facebook. Since our VM's do not have web access, we will use the webpage hosted on VM2_Server that we discussed earlier in the lab. To do this, you can use the following command to set up the tunnel: "`ssh -L 8000:WebpageIP:80 ...`". We will not use this approach, and instead, we use a more generic approach, called dynamic port forwarding, instead of a static one like that in Task 2.a. While it is not essential, we recommend you configure your browser to not cache anything (see Appendix A). To enable dynamic port forwarding, we only specify the local port number, not the final destination. When VM3_Proxy receives a packet from the tunnel, it will dynamically decide where it should forward the packet to based on the destination information of the packet.

```
$ ssh -D 9000 -C eecs@VM_3_IP
```

Similar to the telnet program, which connects `localhost:9000`, we need to ask Firefox to connect to `localhost:9000` every time it needs to connect to a web server, allowing traffic to go through our

SSH tunnel. To achieve this, we can tell Firefox to use `localhost:9000` as its `SOCKS` proxy. Clear the "HTTP Proxy", "SSL Proxy", "FTP Proxy" settings. The following procedure does this:

```
Open Menu (three bars) -> Preferences -> Advanced ->
Network tab -> Settings button.

Select Manual proxy configuration
SOCKS Host: 127.0.0.1      Port: 9000
SOCKS v5
No Proxy for: localhost, 127.0.0.1
```

After the setup is done, please do the following:

- Run Firefox and go visit the VM2_Server page.
- After you get the webpage, break the SSH tunnel, clear the Firefox cache, and try the connection again. Please describe your observation. Note: you are still setup to use the proxy. Since you just killed that connection, you should get a "Proxy server is refusing connections". Go disable the proxy config in Firefox and try again.

**Question 7:** Please explain what you have observed, especially on why the SSH tunnel can help bypass the egress filtering.

You should use Wireshark to see what exactly is happening on the wire. Describe your observations and a detailed explanation of the packets capture along with the Wireshark screenshot.

**Question 8:** If `ufw` blocks the TCP port 22, which is the port used by SSH, can you still set up an SSH tunnel to evade egress filtering?

# 4 Submission requirements

## 4.1 Rubric

1. Questions 1-4, 8 2 pts
2. Documentation questions
   (a) 5 pts question 5
   (b) 5 pts question 7
   (c) 5 pts question 6
3. 10 points for reflection (purpose of stateless firewalls, princples, lessons learned)

### 4.2 Partner Submission

Provide one written lab report, answering each question properly labelled with the number and original question, per partner team. Be sure to include the time spent on the lab and document any external resources used. Again good documentation:

1. clearly enumerates tasks with a description of you did and evidence.
2. shows the progress you were able to achieve.
3. explains your troubleshooting attempts.
4. accurately describes an issue and the potential solution (if really good, I will give near full credit).

### 4.3 Individual Submission

Each member needs to submit a detailed lab reflection. This includes

- approximately one half page that talks about the various security issues and princples.
- although we have demonstrated various evasion techniques to stateless firewalls, do they still have a purpose? How would you employ them?
- any challenging points or thoughts on what you found interesting during the lab
- time spent you personally spent and how much effort you put forth
- time your partner spent, and how much effort they put forth
- be sure document any external resources used.

# 5 Appendix A

These instructions will show you to disable caching on your web browser so that every time you reload a web page, the browser contacts your web server for it instead of using a local version of the page.

- In the Firefox address bar, type `about:config` to get a page of various preferences that you can change. You will first be presented with a warning, click "I'll be careful, I promise!".
- In the "Search:" at the top, type `browser.cache` to filter for the cache options (see Figure 2).

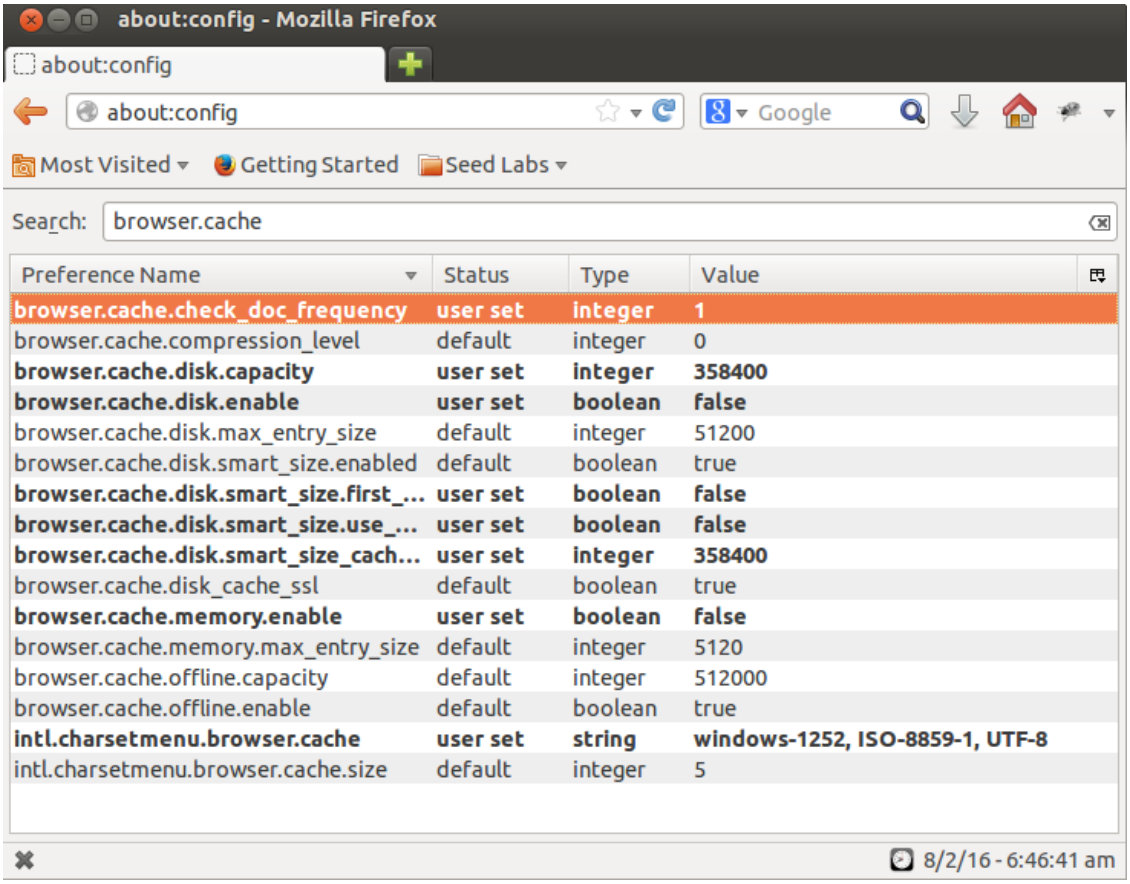



Figure 2: Configuring the Firefox browser cache

- Locate and change entries as follows:

  1. `browser.cache.memory.enable` – double-click to set the Value to "false". This will turn off browser caching in memory.
  2. `browser.cache.disk.enable` – double-click to set the Value to "false". This will turn off browser caching on the disk drive.
  3. `browser.cache.check_doc_frequency` – double-click to open a dialog box to change the frequency, set this to 1 and click "OK". This will force the browser to verify a page each time you load it.