DNS Rebinding Attack Lab

Table of Contents:

1 Abstract:

DNS rebinding is a way of influencing domain name resolution that is often used as a type of computer attack. A malicious web page induces users to launch a client-side script that assaults machines elsewhere on the network in this attack. we going to Learn in this lab how to utilize the DNS rebinding method to target IoT devices firsthand. We have a simulated IoT

device (thermometer) in the setup that can be operated via a web interface. Many IoT devices lack a solid protective system, and if attackers can communicate with them directly, they can quickly hack these devices.

2 Introduction:

a lot of IoT devices include a simple designed web server, allowing users to interact with these devices via web APIs. These IoT devices are typically protected by firewalls and cannot be accessed directly from the outside. Many IoT devices lack a strong authentication mechanism as a result of this type of security. It is simple for an attacker to compromise their security if they can find a way to interact with them.

An attacker uses a custom DNS server to fake the victim's IP address and thereby get read access to the victim's server in a DNS rebinding attack. DNS rebinding attacks have existed for over 15 years (Dean et al., 1996). Because it is low-cost and relatively rapid to execute, this assault has the potential to jeopardize both the victim and the intranet to which he or she is connected.

The attacker conducts a DNS rebinding attack using a malicious website, such as attacker.xyz, and a custom DNS server. When the victim visits attacker.xyz, the website sends a request to its server in the background. As a response, the malicious server returns the victim's private IP address. As a result, SOP in the victim's browser is duped into believing that the two IP addresses belong to attacker.xyz, and read access to the malicious website is granted.

In fact, an attacker uses DNS rebinding to circumvent firewalls, obtain access to internal computers in the victim's private network or to the victim's resources, retrieve sensitive resources, and exploit susceptible internal workstations.

The DNS rebinding attack has severe consequences.According to one research, major home routers like as Linksys, Thompson, Belkin, Dell, and Asus are vulnerable to DNS rebinding, putting millions of customers' privacy at risk (Heffner, 2010). Tatang et al. investigated four Internet of Things (IoT) devices and discovered that three of them are susceptible to DNS rebinding attacks (Tatang et al., 2019). They underlined that most smart home gadgets rely largely on the router's safeguards to identify and stop such assaults. Similarly, Dorsey revealed in a whitepaper that commonly used smart gadgets such as Google Home mini, RokuTV, Sonos speakers, and home thermostats may be controlled by an attacker from out of the victim's private network.

This lab report covers the following topics:

• DNS server setup

• DNS rebinding attack

• Attacks on IoT devices

• Same Origin Policy

3 Objective:

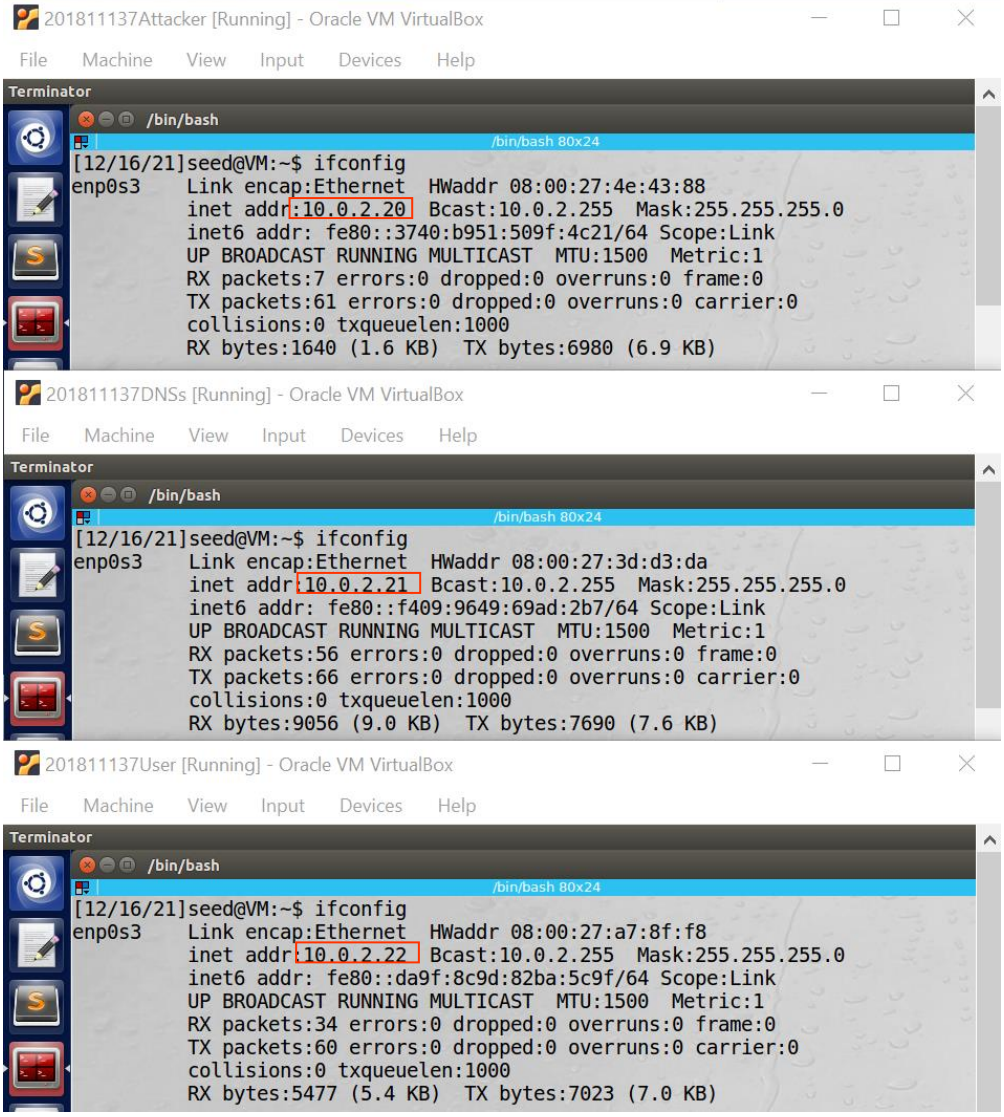
The objective of this attack is an IoT device (thermometer) protected by a firewall. This IoT device cannot be accessed from the outside. Our goal is to convince internal users to run our

JavaScript code so that we can interact with IoT devices via DNS rebinding attacks. The main purpose for DNS rebinding attacks is poison local DNS resolver cache

4 Tasks:

The lab's environment are as follows (Network Setup):

Name	Role	IP Address	MAC Address
201811137Attacker	Attacker	10.0.2.20	08:00:27:4e:43:88
201811137DNSs	Local DNS Server	10.0.2.21	08:00:27:3d:d3:da
201811137User	User Machine	10.0.2.22	08:00:27:a7:8f:f8

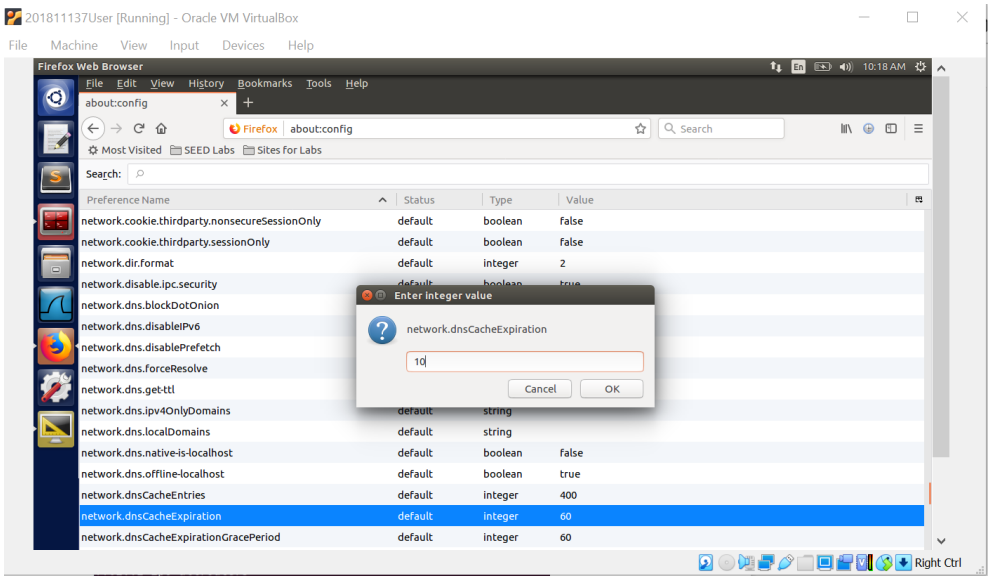


4.1 Task 1: Configure the User VM.

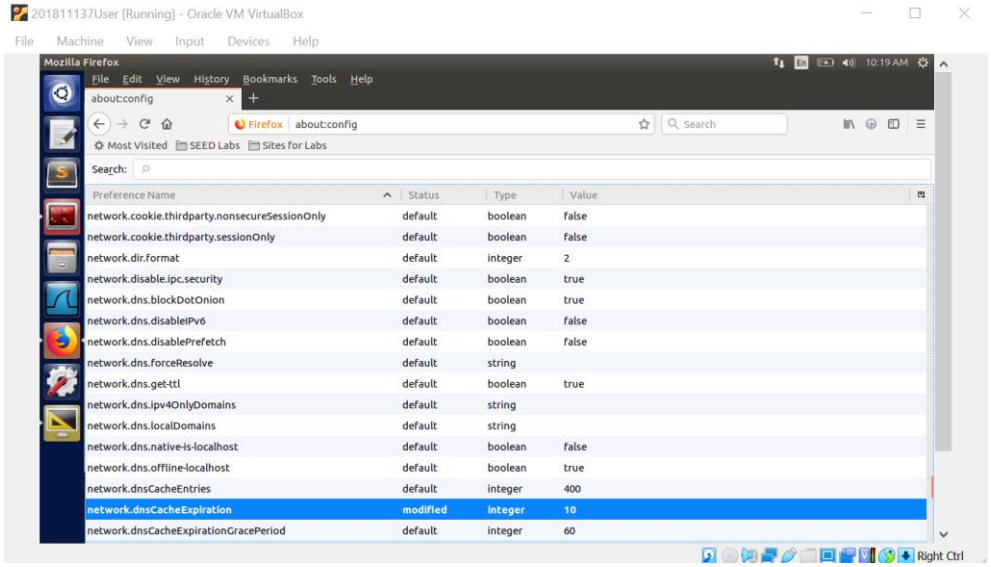
#Step 1. Reduce Firefox's DNS caching time:

Shorten the time it takes Firefox to cache DNS information. The Firefox browser caches DNS results in order to lessen the burden on the DNS server and improve response time. The cache expiry period is set to 60 seconds by default. This means that our DNS rebinding attack must wait at least 60 seconds before proceeding. We lowered the time to 10 seconds or less to make our experiment easy. In the URL

bar, type about:config. We'll get a list of preference names and values once we click on the warning page. dnsCache should be searched for, and the following entry's value should be set to 10:



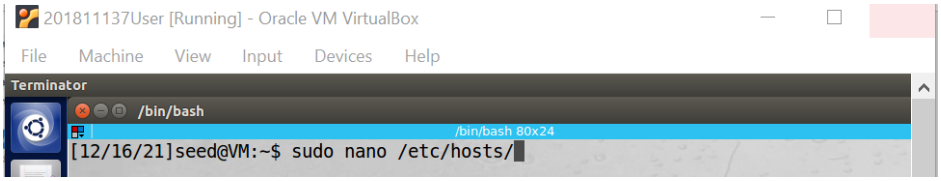
To speed up our attack, we lower the Firefox DNS caching duration to 10 seconds.



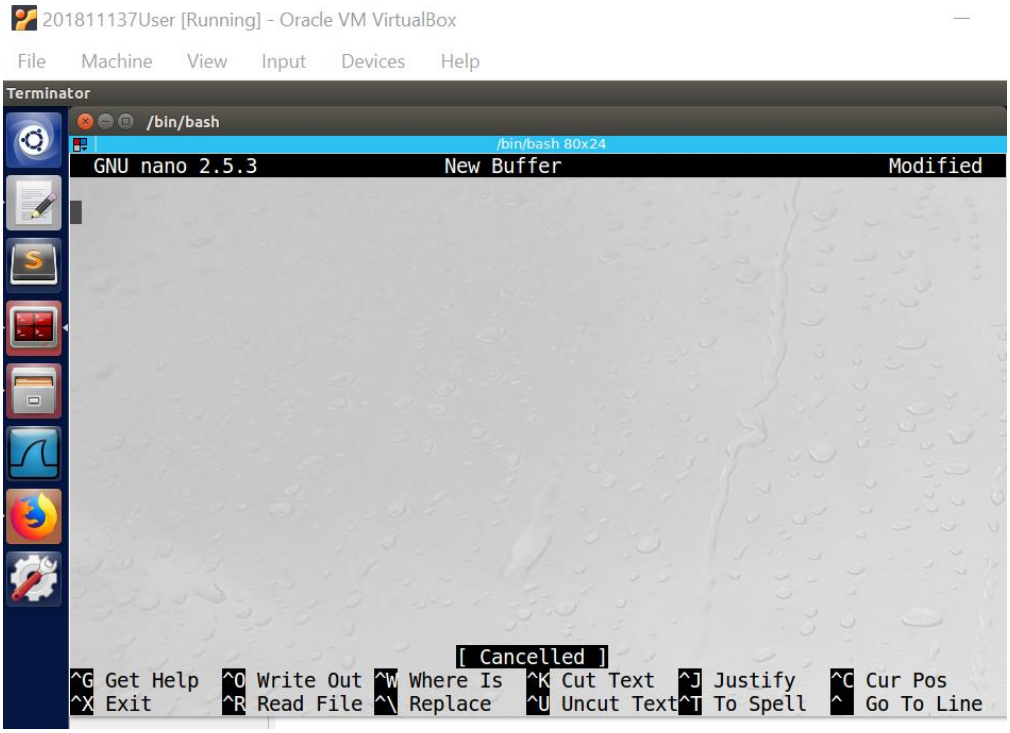
After making the modification, we must quit and restart the Firefox browser; otherwise, the change will not enter into force.

#Step 2. Change /etc/hosts:

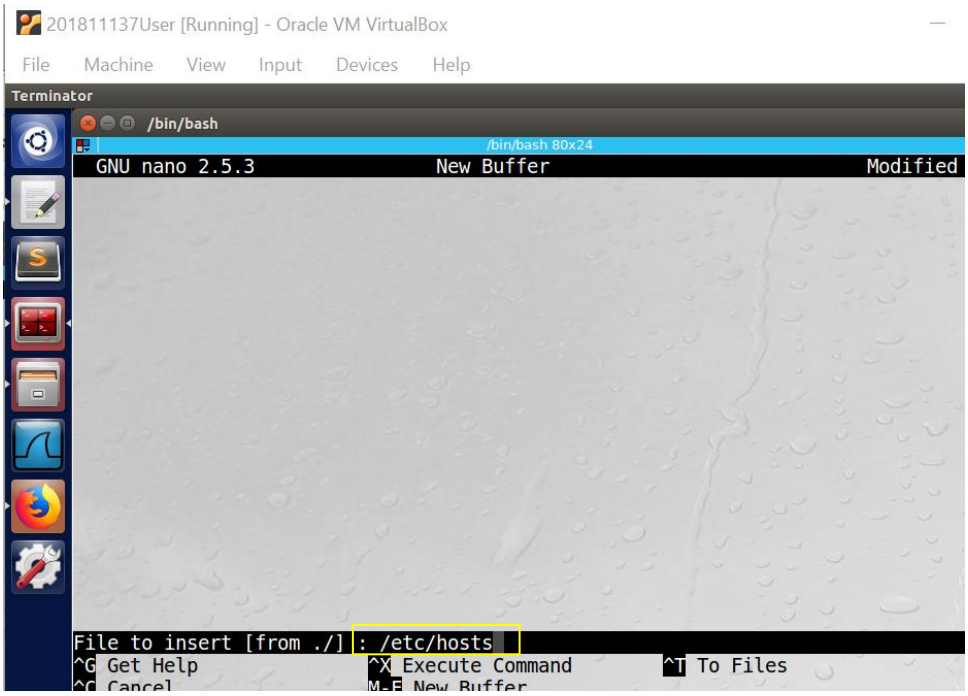
We going to modify the etc/hosts host file:



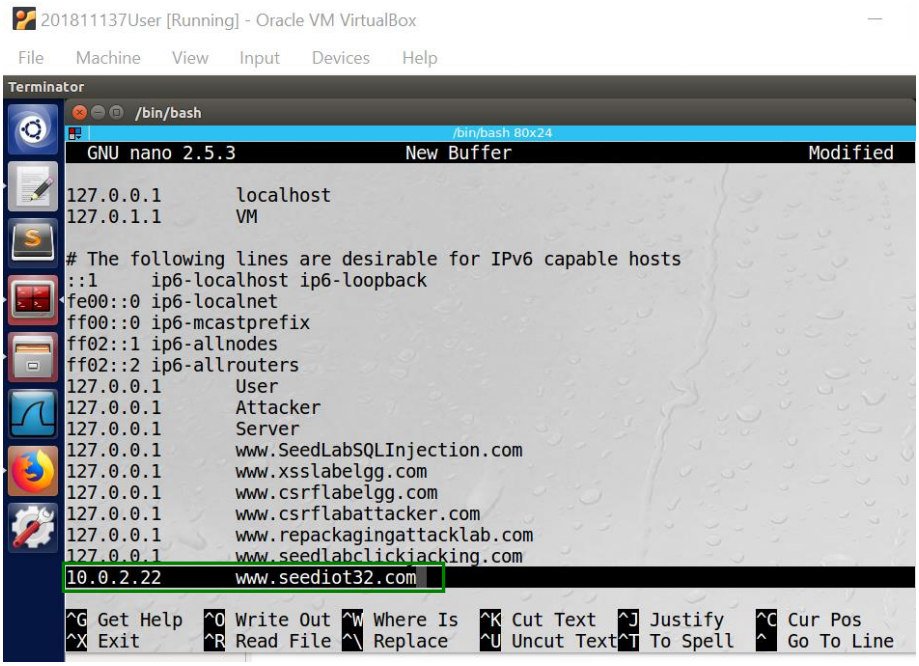
It will show this



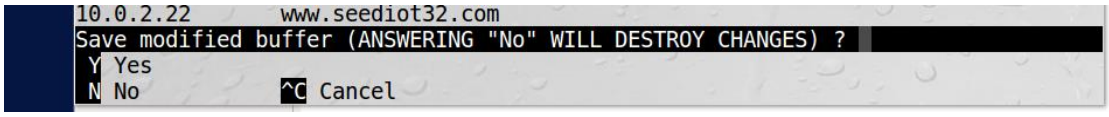
I will click ctrl+R



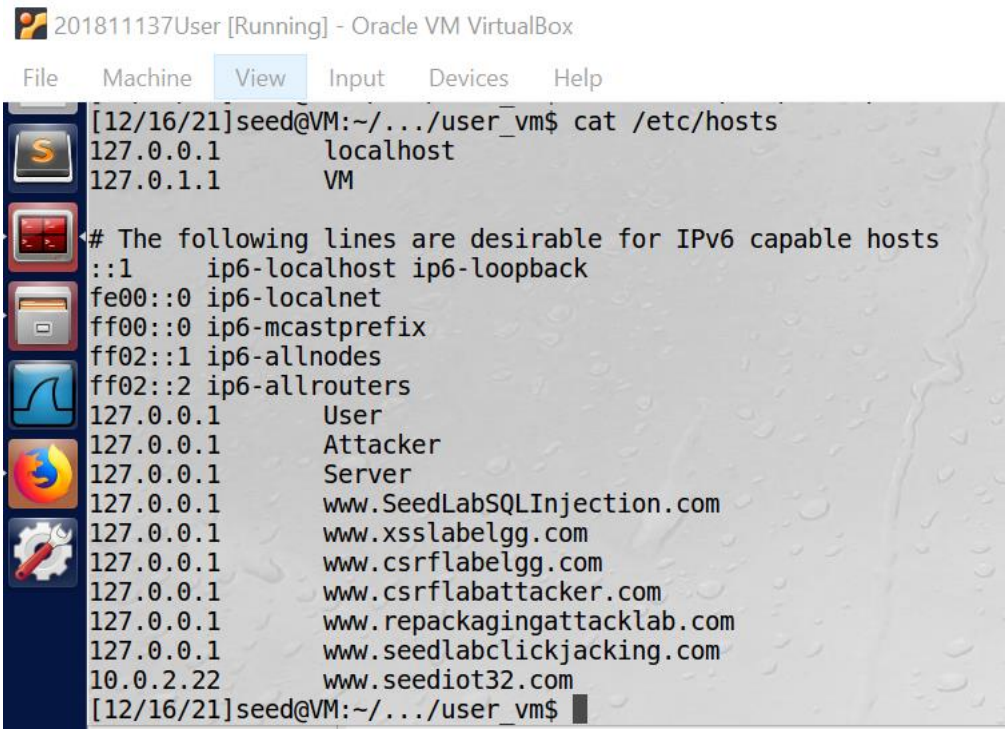
In this step we going to Modify **/etc/hosts.** The following items must be added to the **/etc/hosts** file. The name of the IoT web server will be **www.seediot32.com.** This server can operate on several VMs, however for the sake of simplicity, we run the IoT server on the user VM.



Now I will save it



On the User VM (10.0.2.22), I will launch the IoT web server with the name www.seediot32.com

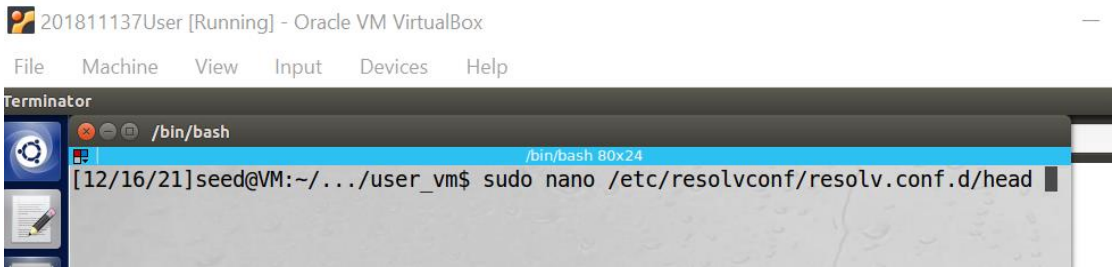


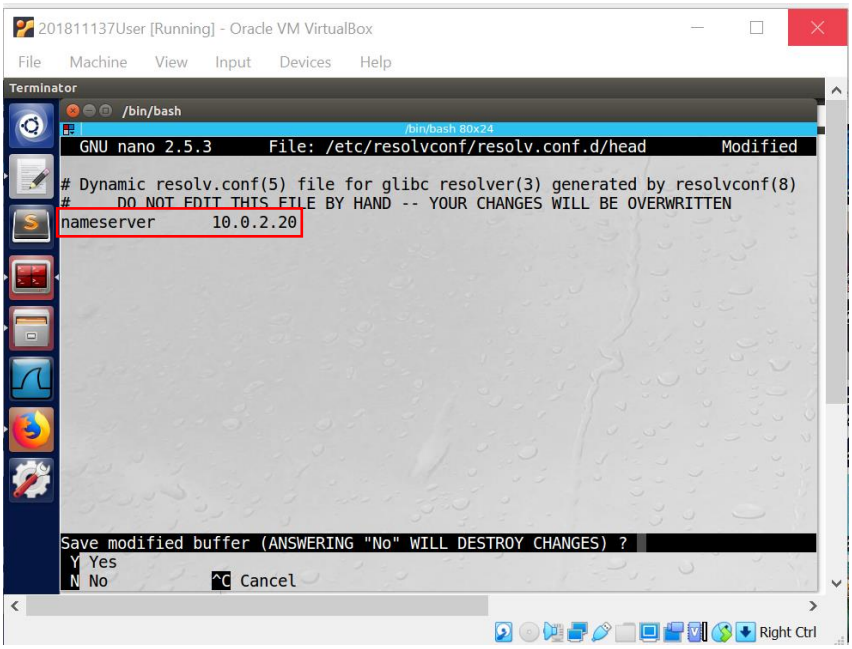
#Step 3. Local DNS Server:

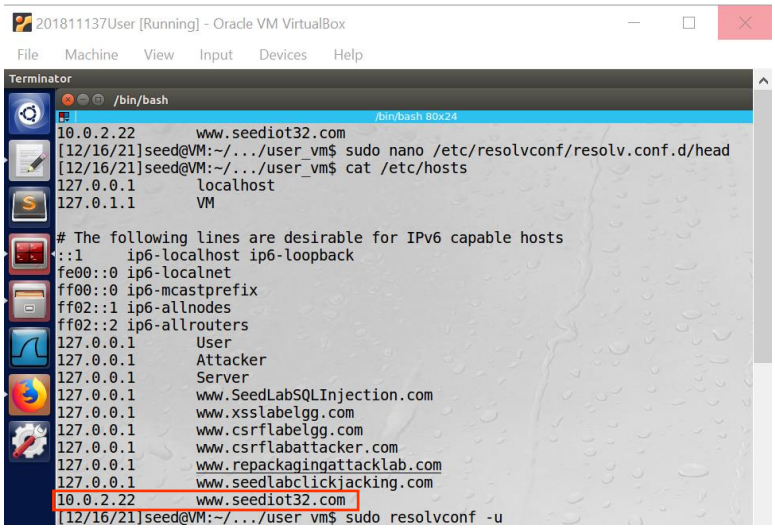
I must use 10.0.2.21 as the local DNS server on the user workstation 10.0.2.22. To avoid the issue of DHCP configuration altering the content of the /etc/resolv.conf file, we insert the nameserver in the /etc/resolvconf/resolv.conf. d/head file, which is prepended to the dynamically created resolver configuration file. To have the change take effect, I execute sudo resolvconf -u after making the change:

The userVM must be configured to utilize a specified local DNS server. Setting the local DNS server to the first name server entry in the resolver configuration file (/etc/resolv.conf) does this. One problem is that the offered VMs get network configuration settings such as IP addresses, local DNS servers, and so on through Dynamic Host Configuration Protocol (DHCP). The information given by the DHCP server will be overwritten in the /etc/resolv.conf file by the DHCP client.

Add the following addition to the /etc/resolvconf/resolv.conf.d/head file to import information into /etc/resolv.conf without bothering about DHCP.

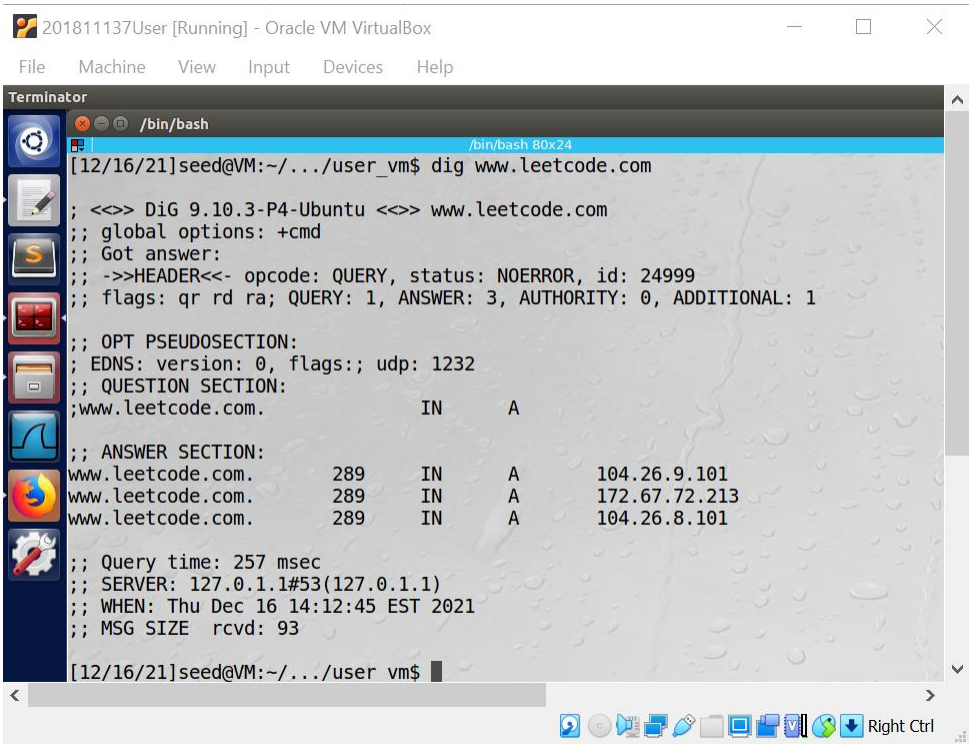






#Step 4. Testing:

After setting the user VM, use the dig command to get the IP address of your choosing from the hostname. Provide proof that this response came from your server based on this response. If no proof is discovered, the setup was deemed ineffective.



4.2 Task 2: Start the IoT server on the User VM.

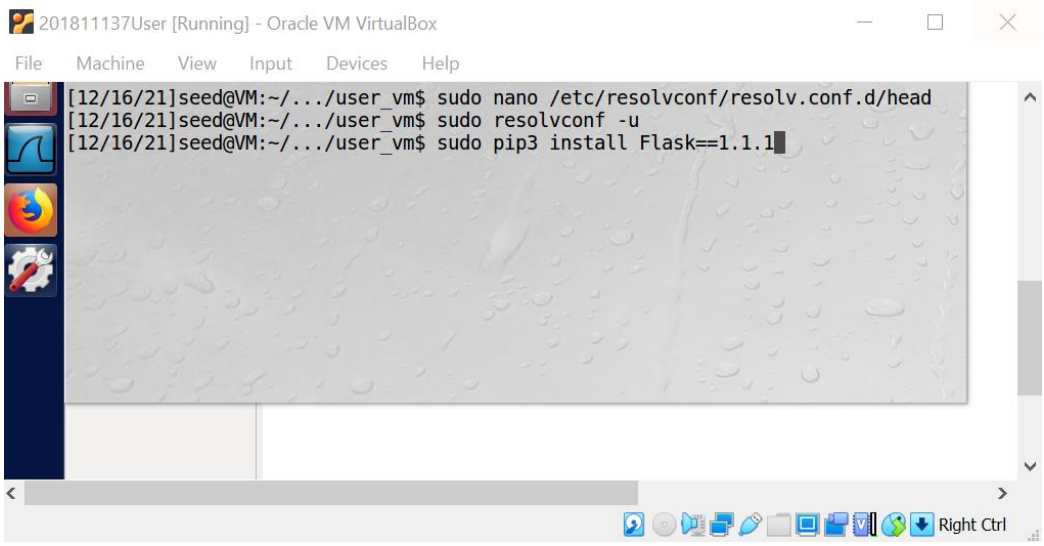
We will launch the IoT server on the user VM in this step. Users can connect with IoT devices via the web server.

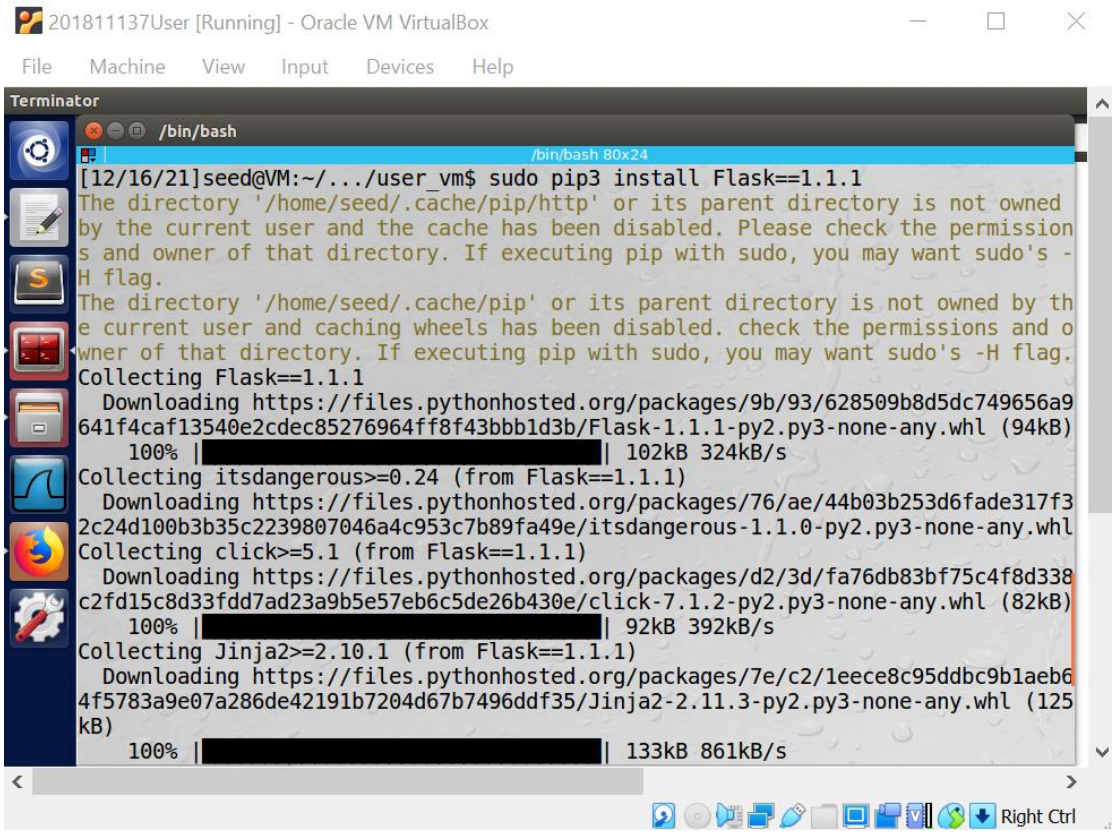
#Step 1. Install Flask:

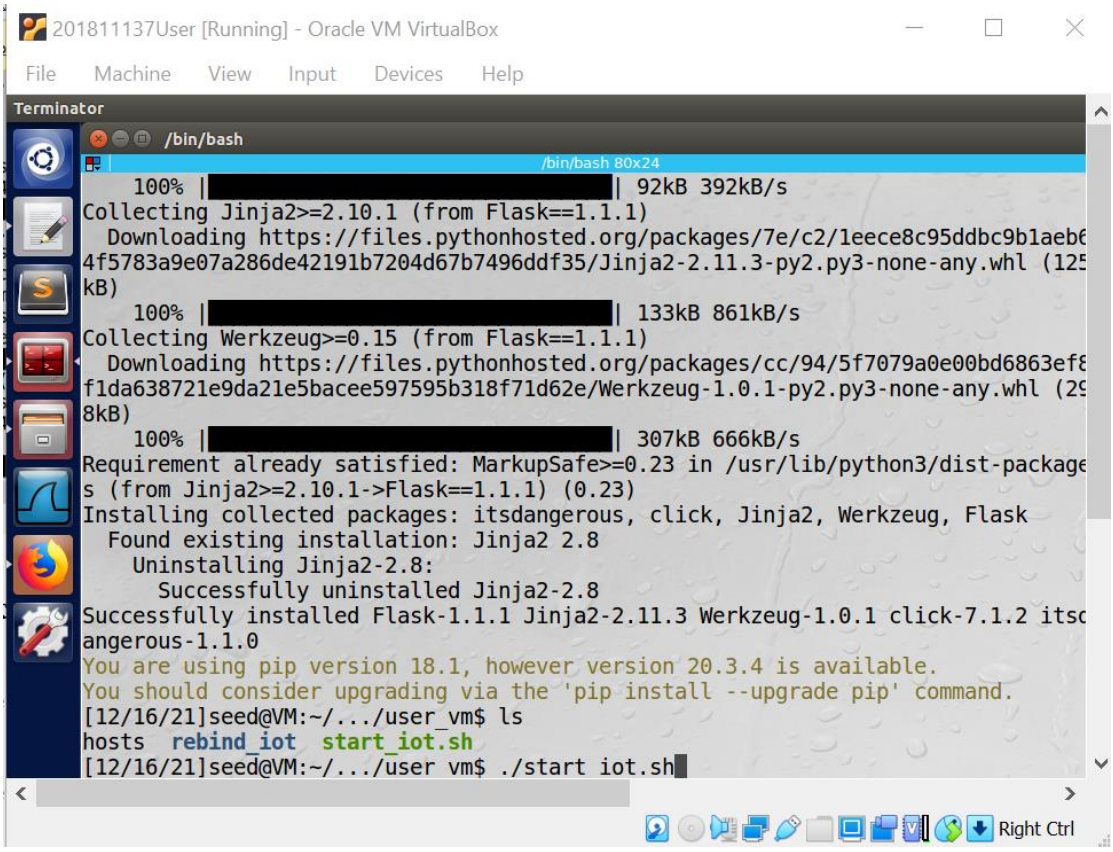
The header file's contents are pre-written into the dynamically created parser configuration file. This is normally only a remark line (the comment in

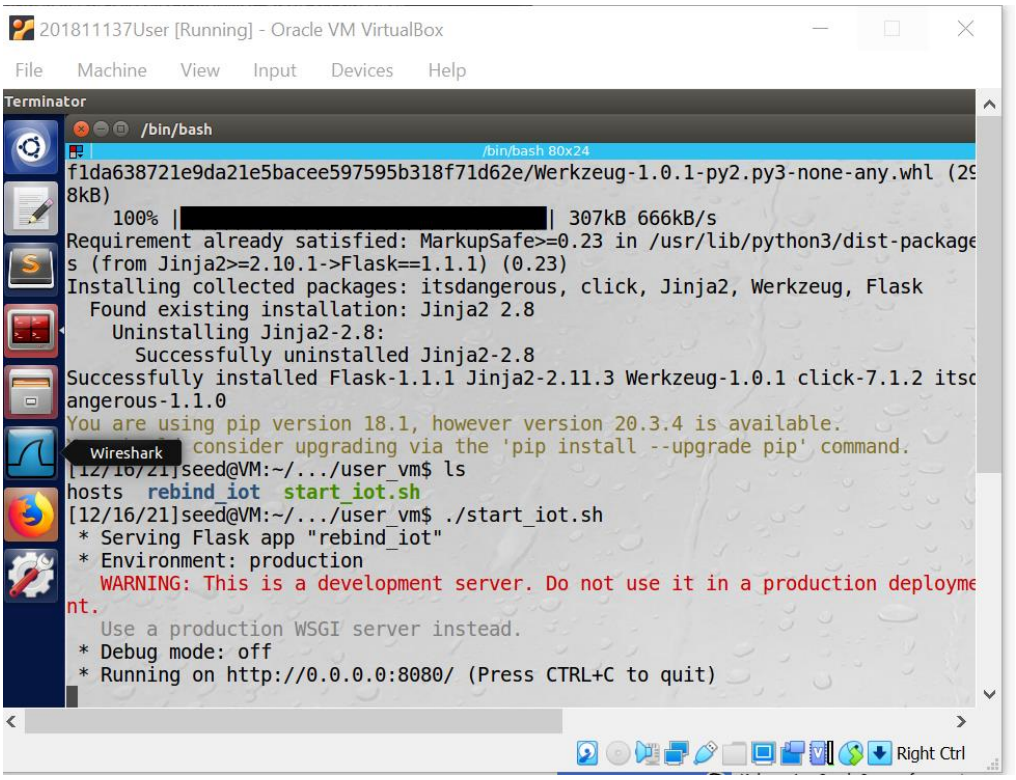
/etc/resolv.conf originates from this header file). To make the changes take effect, we must execute the following command after making the modifications:

Installing the FLASK web framework, which will be used to build the IoT server:



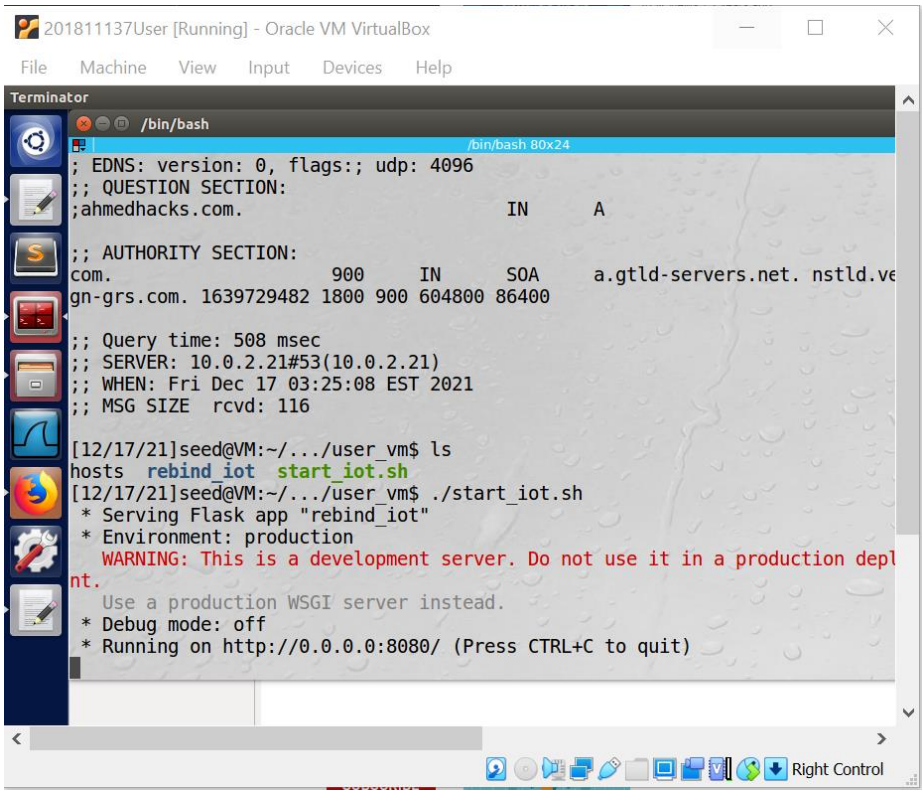






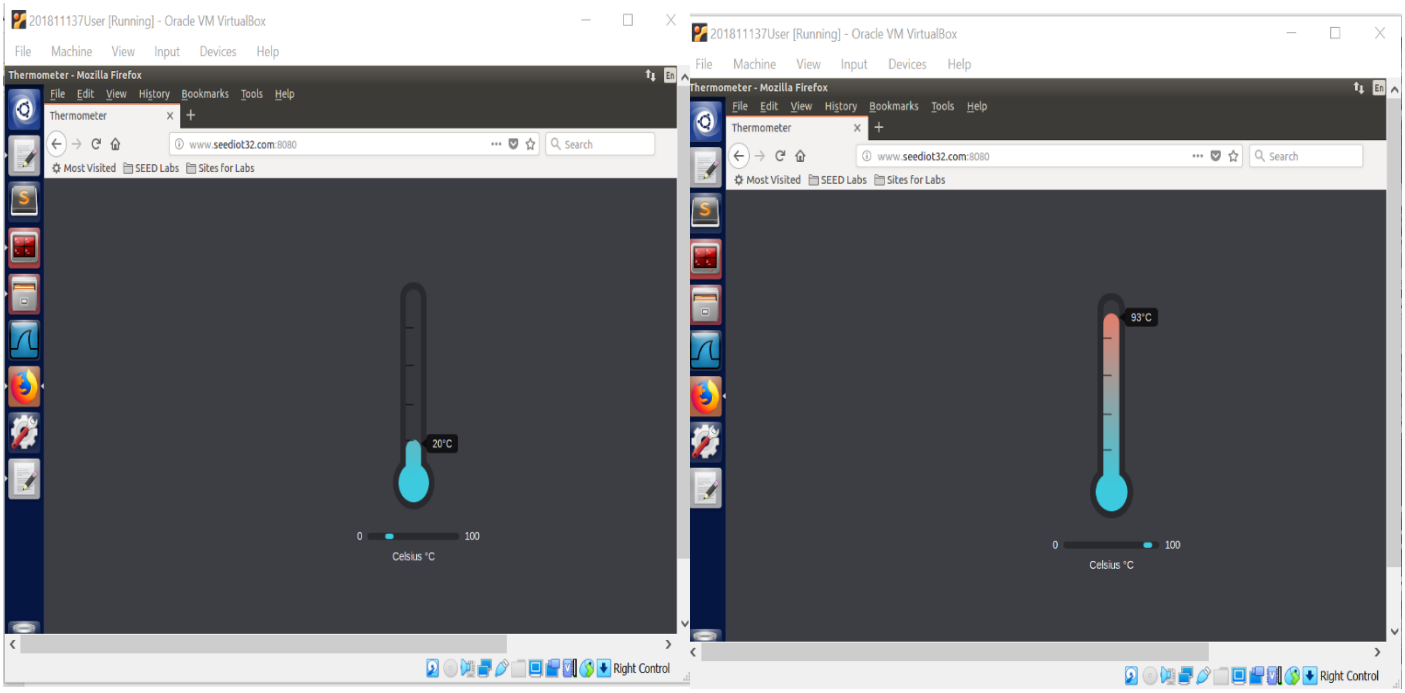
Now we will go to chick

#Step 2. Start the IoT server:



#Step 3. Testing the IoT server:

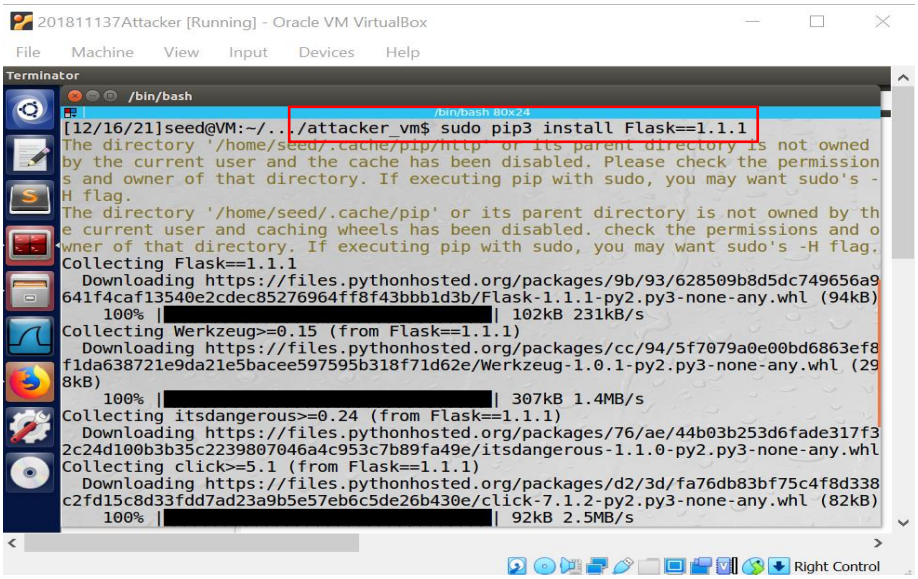
To check the IoT server, navigate to the following Link on the user VM. We should be able to see a thermostat if everything is properly set up. We may also adjust the temperature by sliding the slider.



4.3 Task 3: Start the attack web server on the Attacker VM.

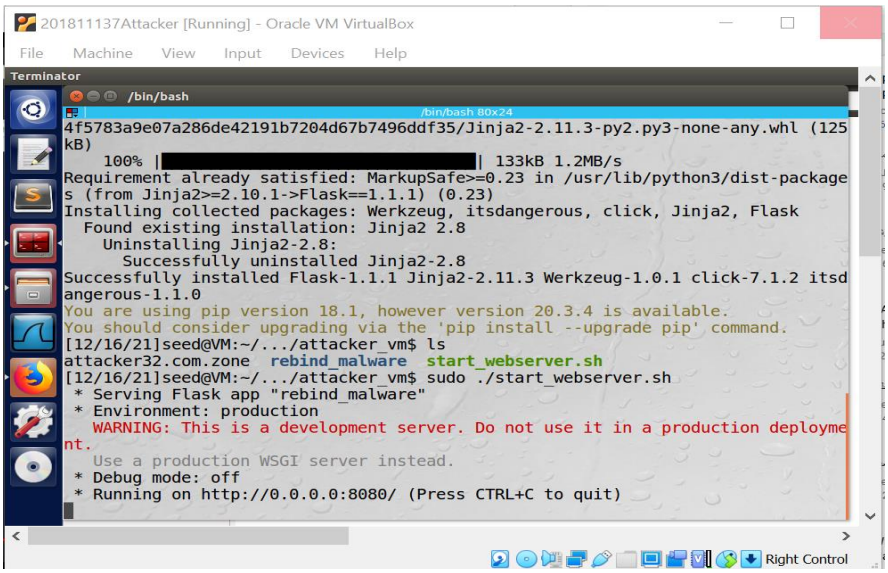
IoT devices in this Task can only be accessible from behind a firewall, i.e. from user VMs in the experimental configuration. Allowing the user to visit our website so that JavaScript code placed on a web page may reach the user's virtual machine is a common approach to transmit harmful code to the user's virtual machine. We will establish a web server to host these web pages in this assignment.

#Step 1. Install Flask:

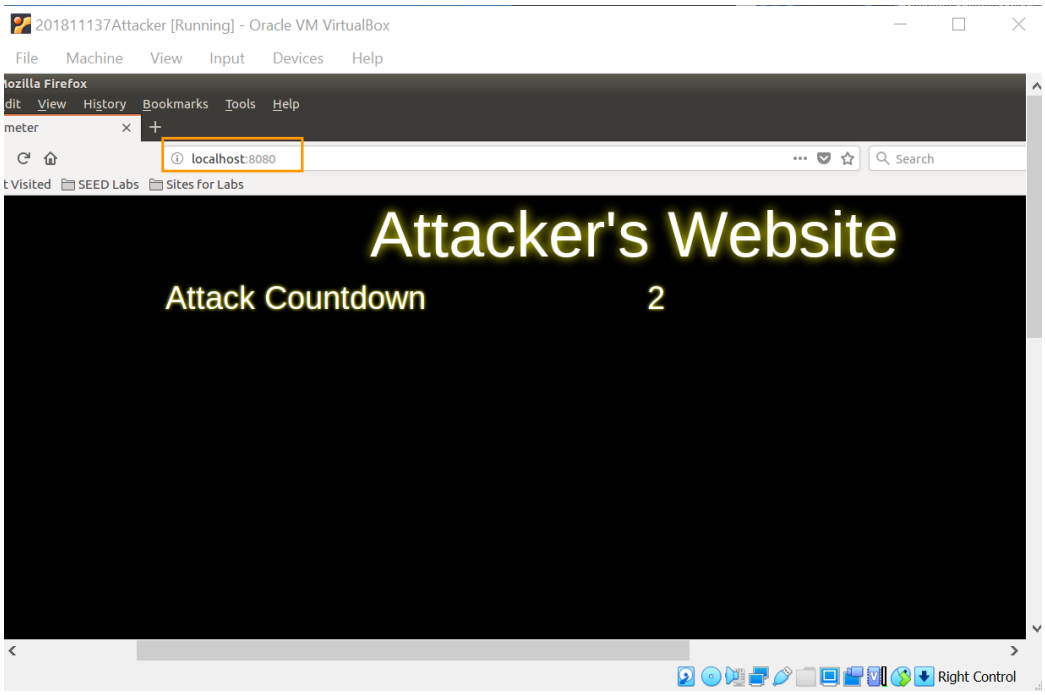


#Step 2. Start the attacker's web server:

Begin the web server by running the script start webserver.sh that has been created.

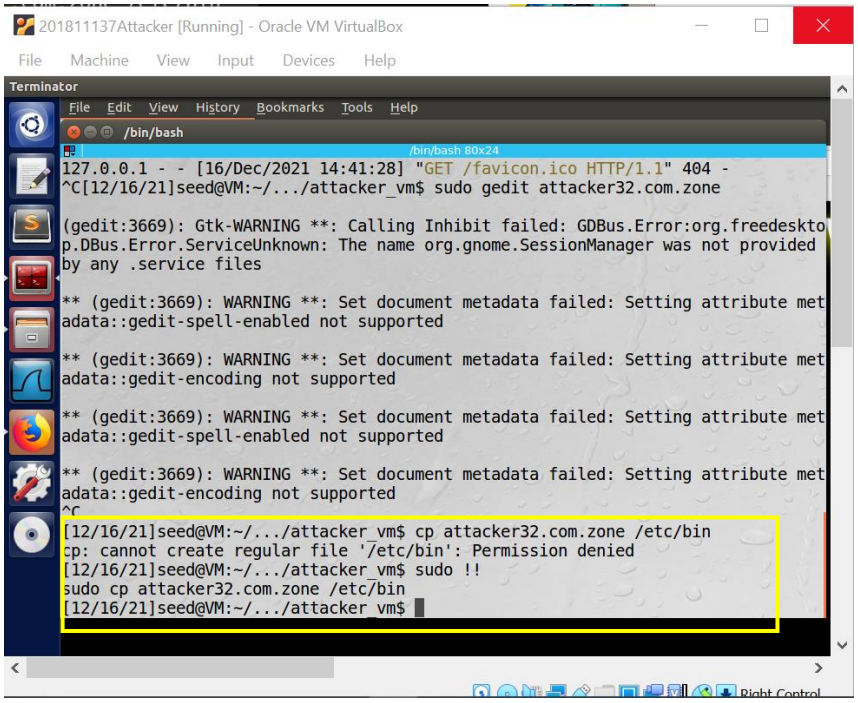


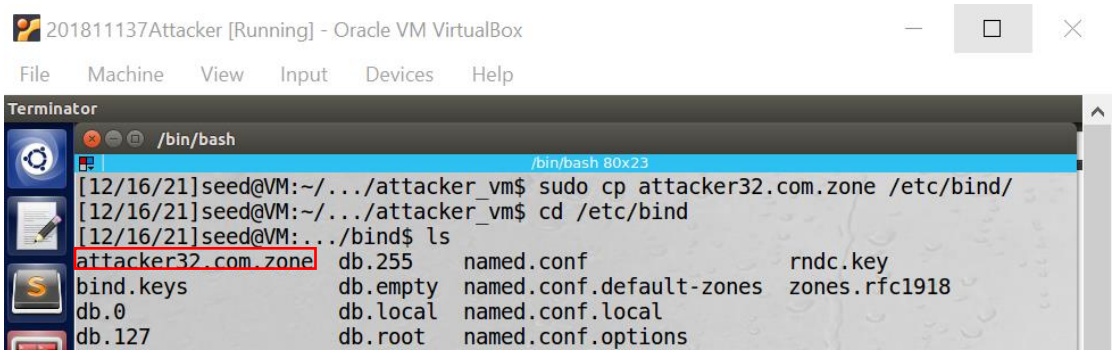
#Step 3. Testing the Attacker's web server:



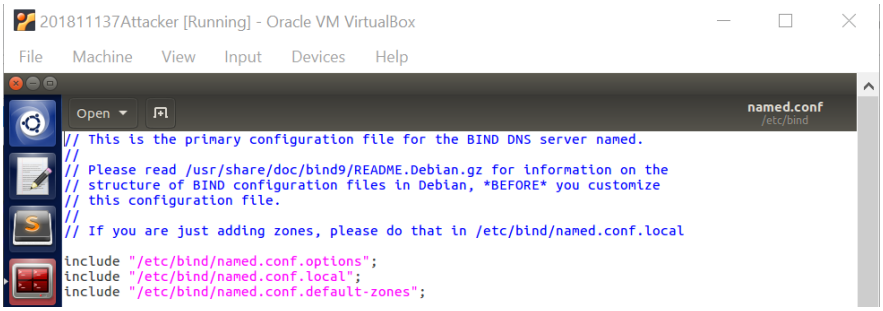
4.4 Task 4: Configure the DNS server on the Attacker VM.

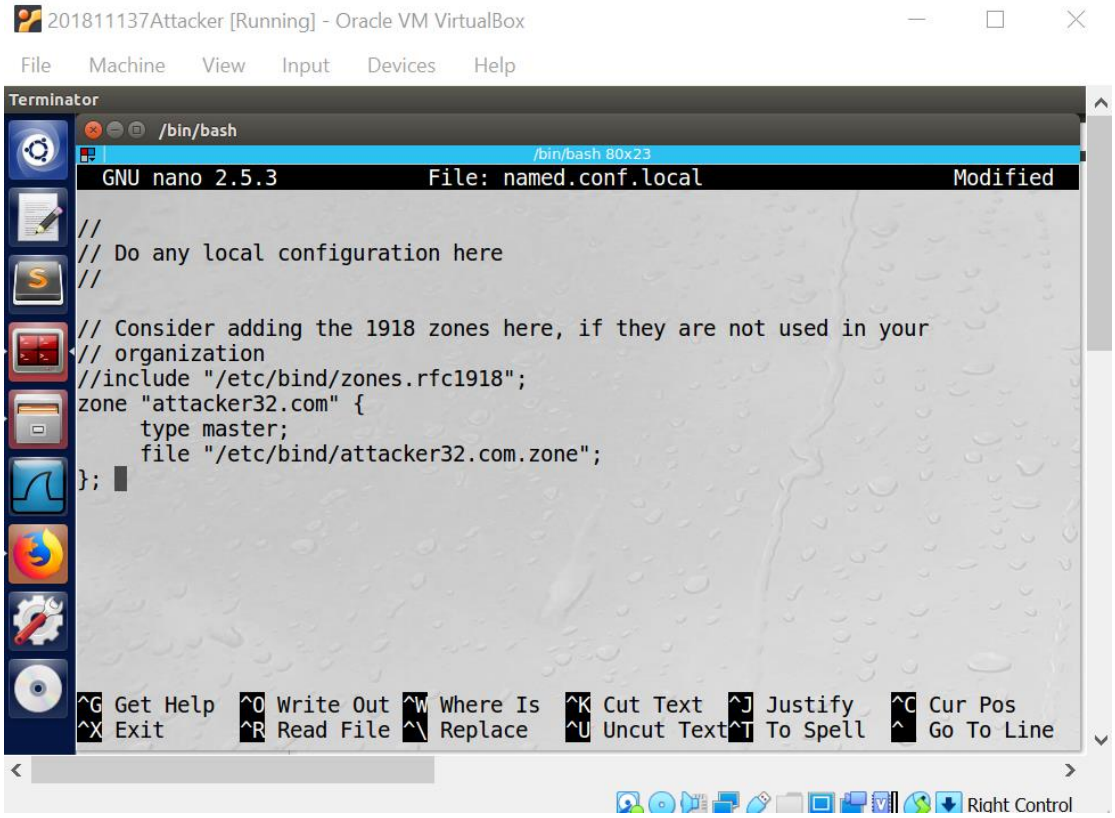
The attacker VM also serves as a naming server for domain names ending with attacker32.com. The BIND9 server is already up and running on the attacker's VM, and I need to create a zone file for it. In the attackr vm folder, you'll find an example zone file. I should make the necessary changes to the zone file and copy it to the /etc/bind folder.



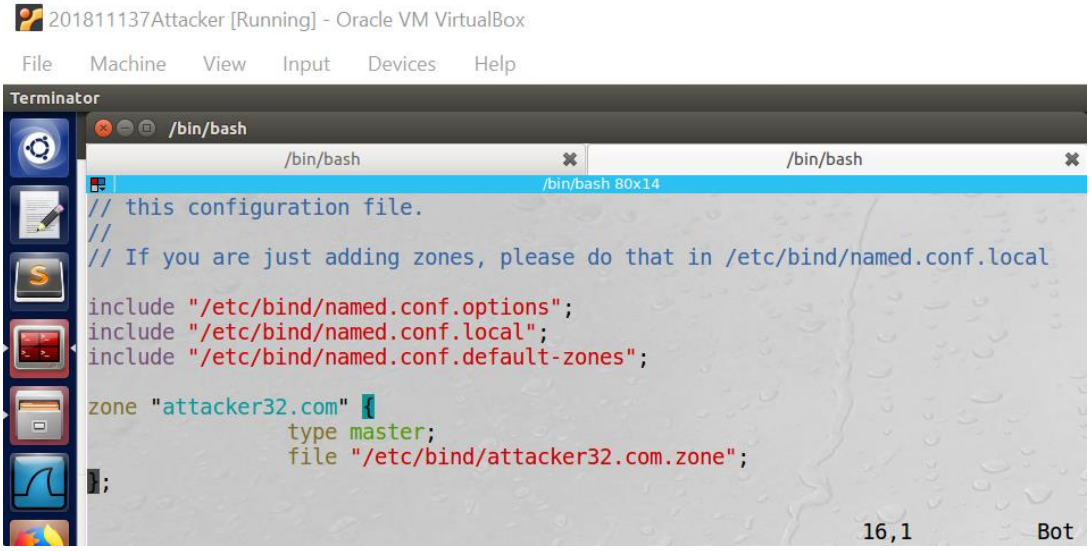


Put the following area entry to /etc/bind/name. As a result, the aforementioned zone file will be utilized by the BIND9 server.

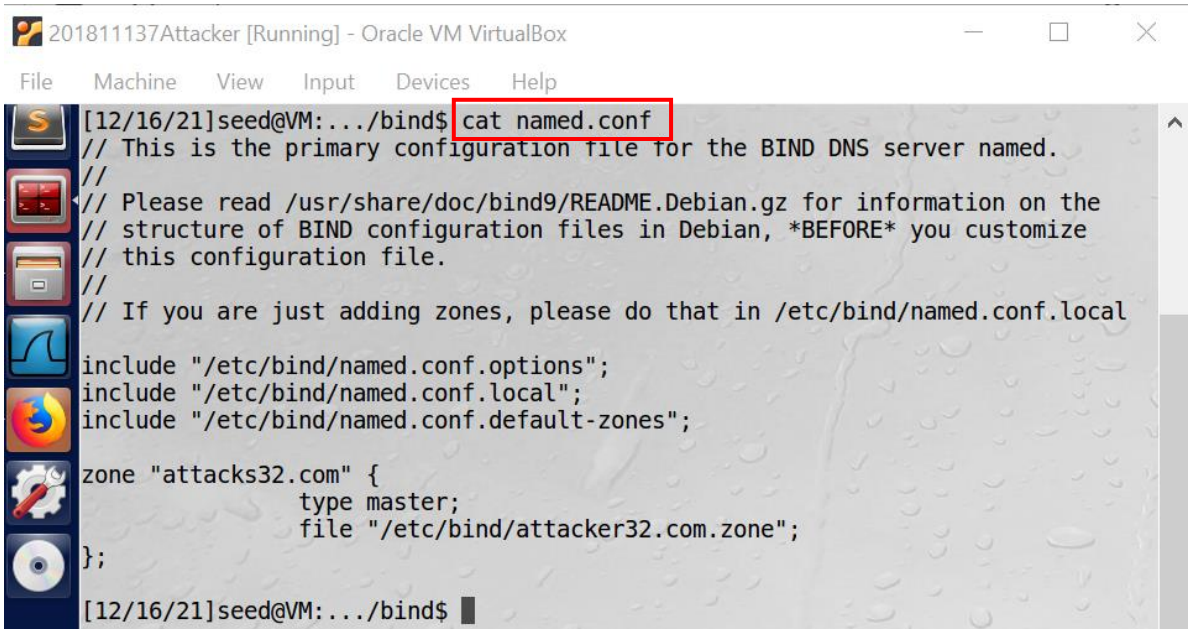




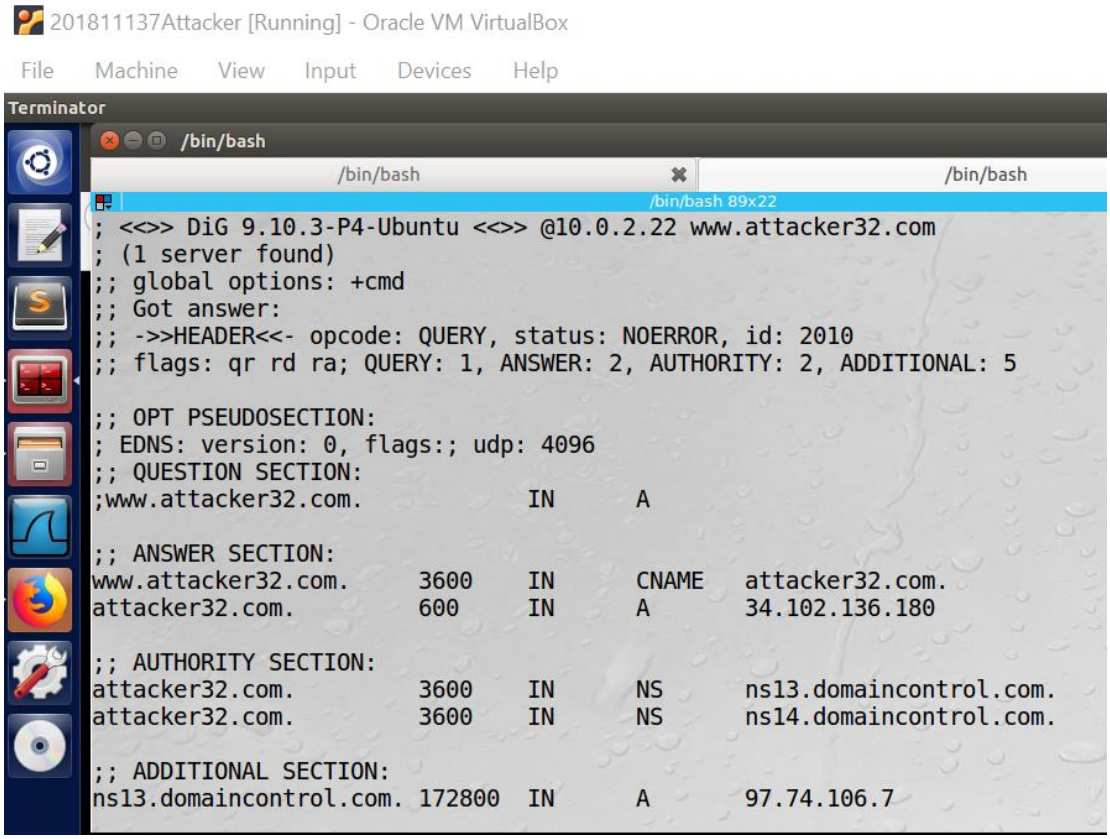
To create a record like this to the local DNS server, add the following line to /etc/bind/named.conf. To create a record like this to the local DNS server, add the following line to /etc/bind/named.conf.



After I change & save it, I prefer to chick by using cat named.conf command.



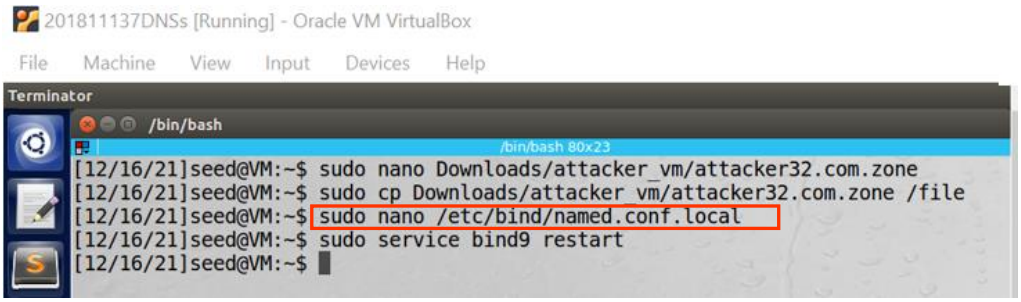
Whether everything is in order, we can attempt the following dig command to test if the response we get matches the one in the region file.

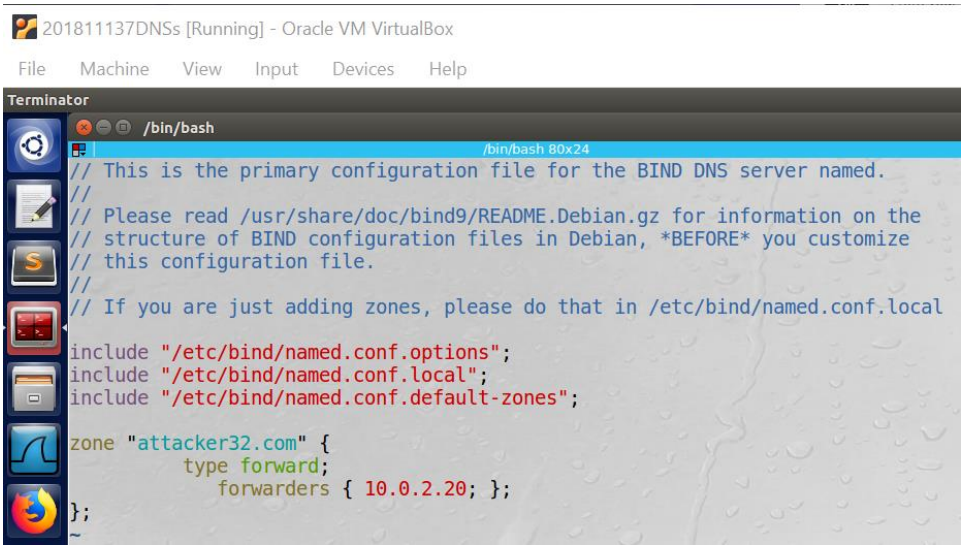


4.5 Task 5: Configure the Local DNS Server.

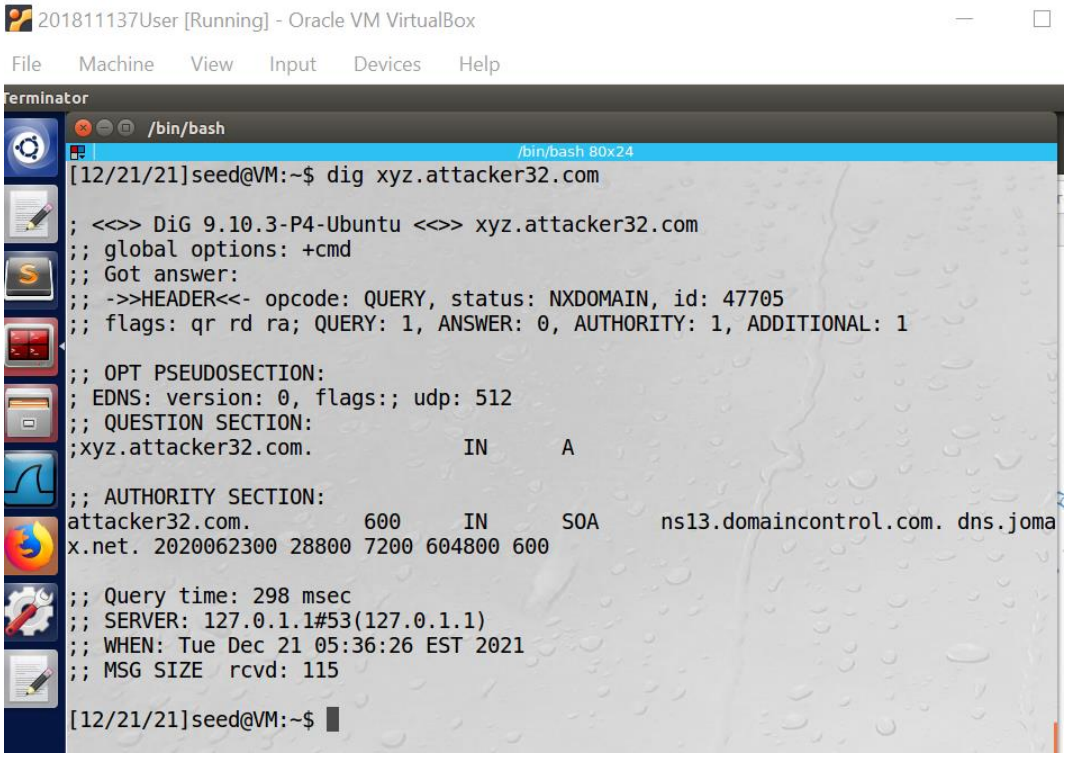
We set up forwarding records for the attacker32.com domain on the local DNS server, so that whenever the local DNS server receives a DNS query from a host within this domain, it simply sends the DNS query to the IP address of the specified forwarding record, rather than going to the root server and the.com server to determine the location of the attacker32.com domain's name server.

To put such a record to the local DNS server, I need to put /etc/bind/named.conf.local





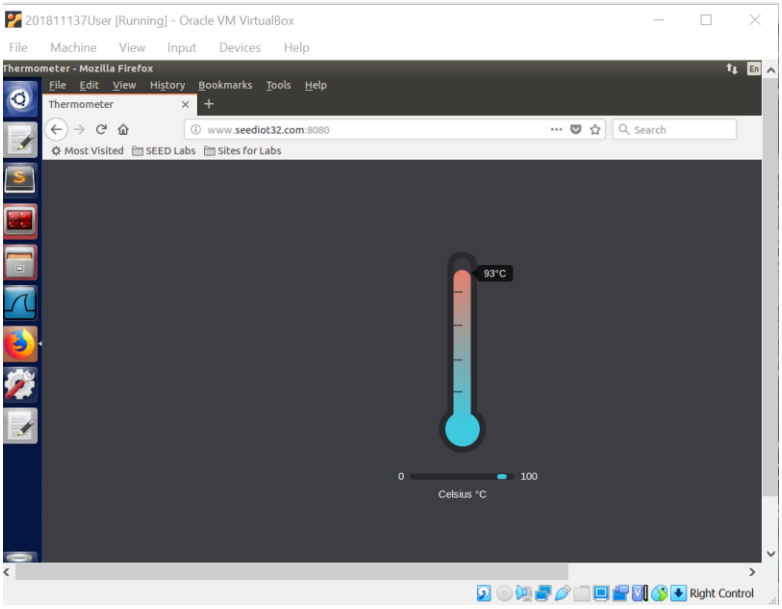
dig command tested on the user machine:



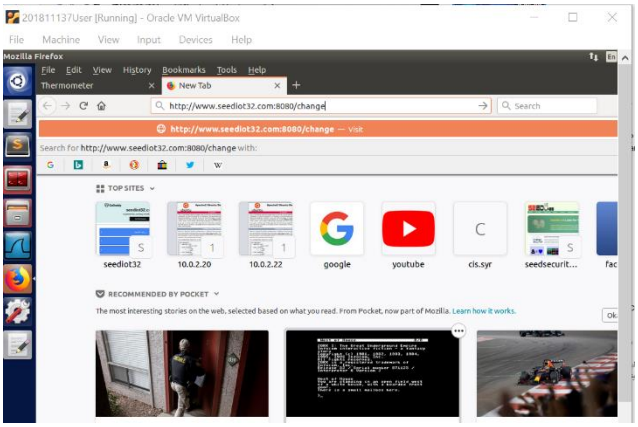
4.6 Task 6: Understanding the Same-Origin Policy Protection.

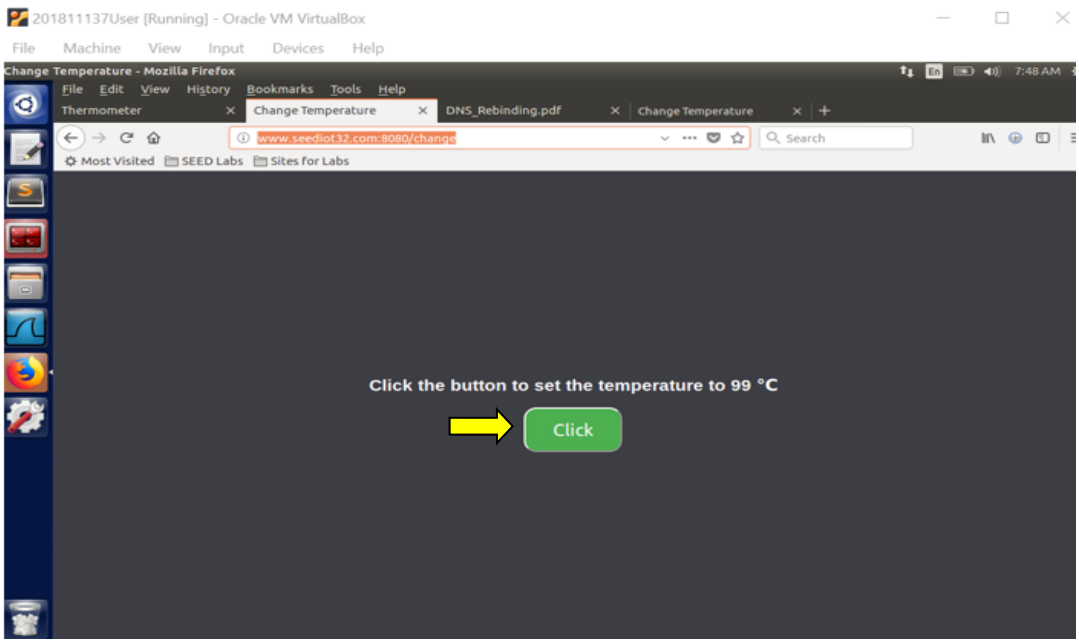
A same-origin policy is a security convention that is the browser's most fundamental and fundamental security feature. The same-origin policy of the browser prevents "document" or script from different sources from reading or modifying specific attributes on the current document The "source" is influenced by the following factors: host (either a domain name or an IP address; if an IP address), it is considered a root domain name), subdomains, ports, and protocols. In the browser, tags like script>, image>, iframe>, and link> can load resources from other domains without being prohibited by the same-origin policy, but browsers limit JavaScript's capabilities so that it cannot read or write the returning content.

I will try the first URL before.

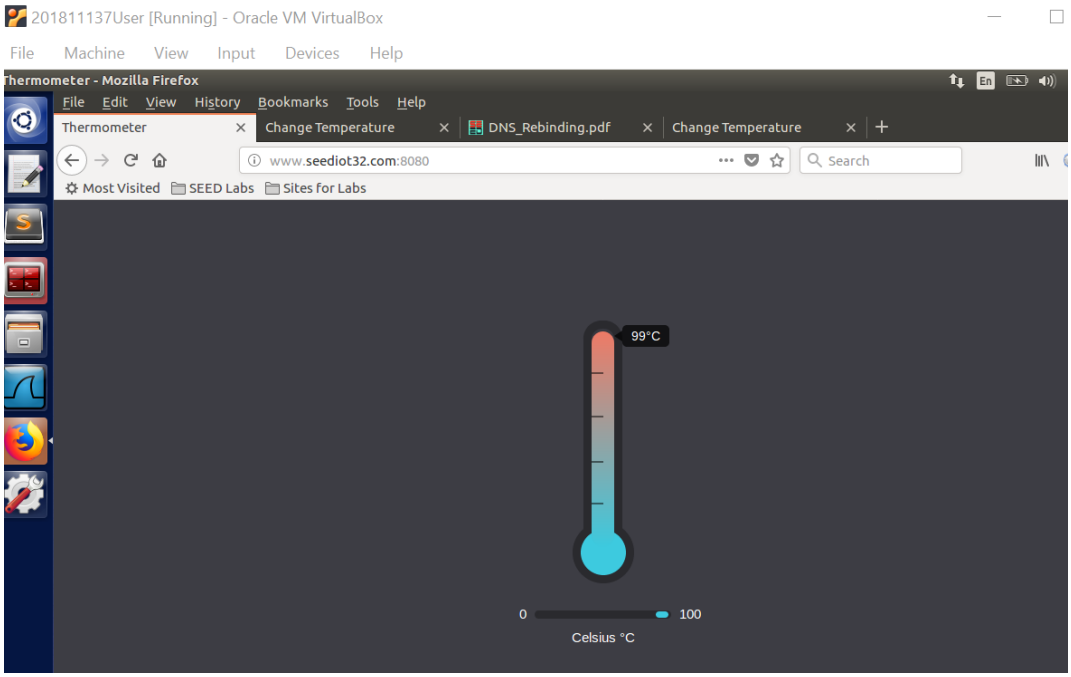


The sconed URL:

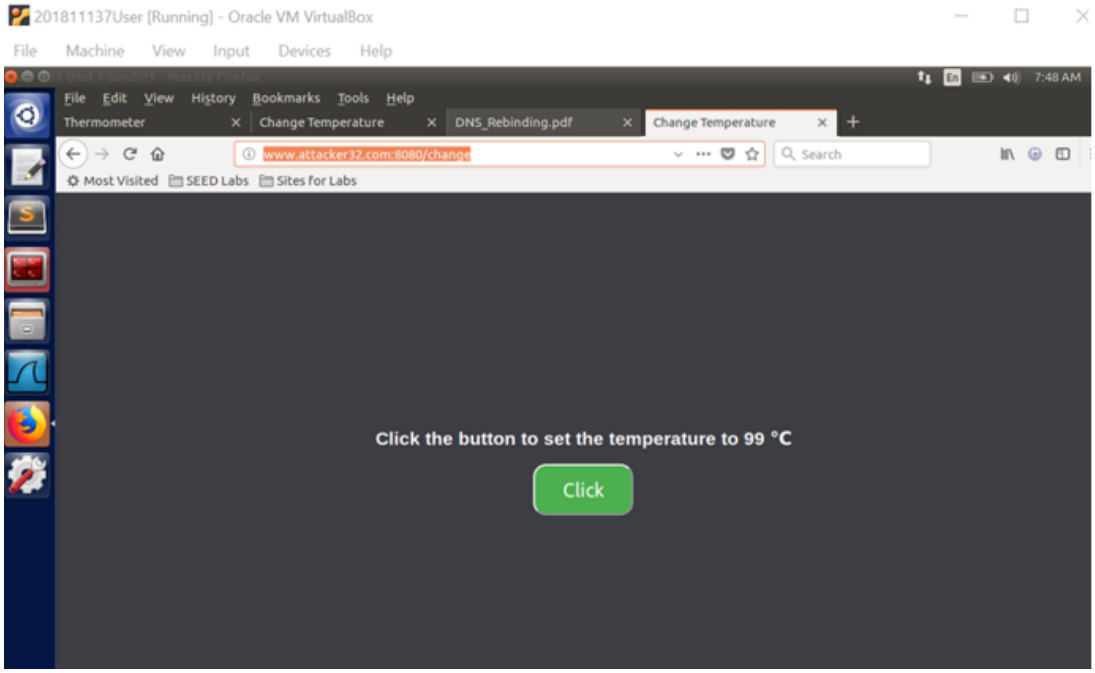




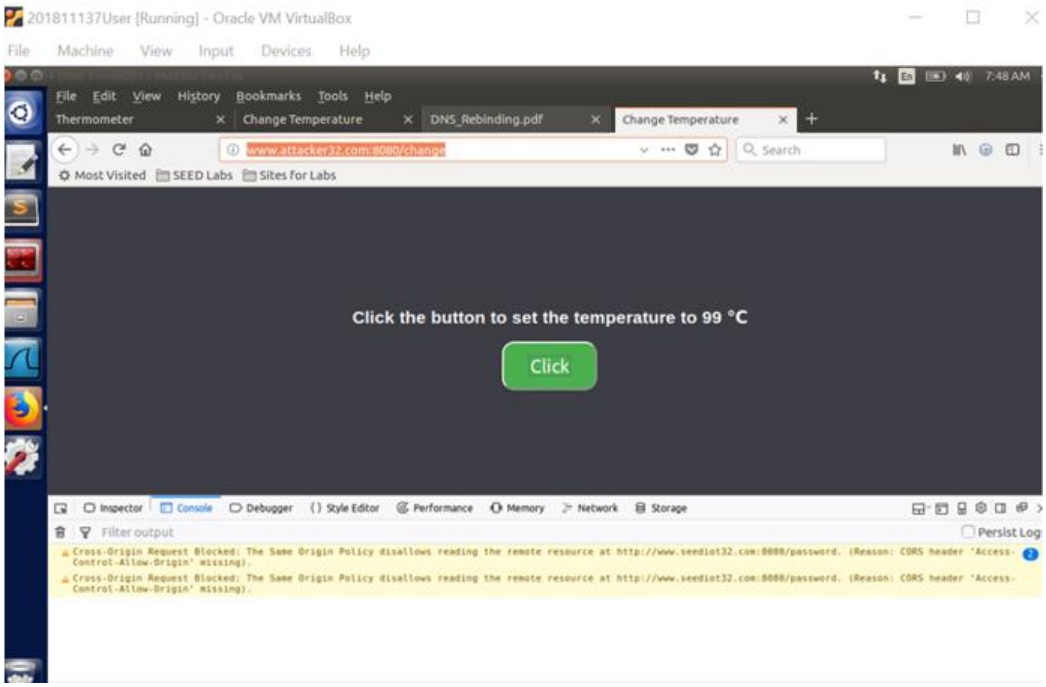
When I clicking "click" in the sconed URL it was change successfully



The third URL:



The temperature of the user vm does not change after clicking click when I was trying the third URL



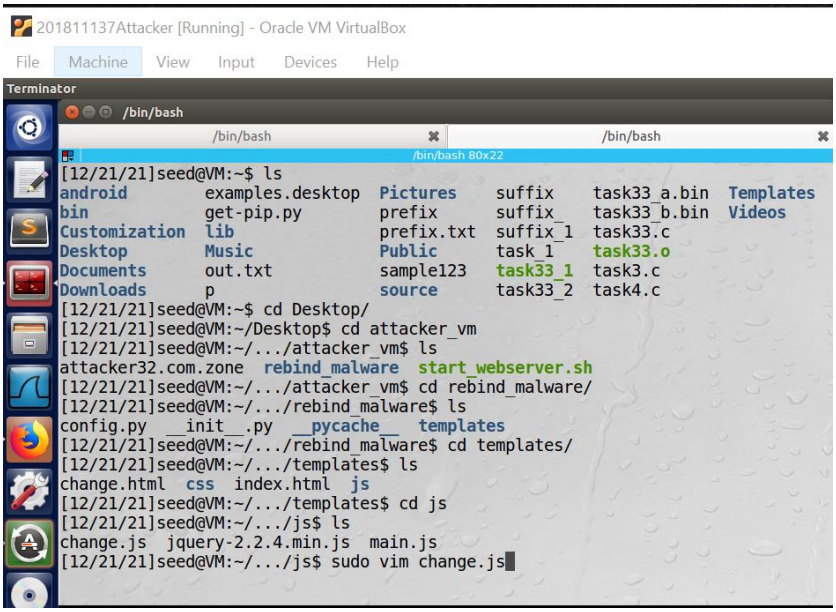
Despite the fact that the fundamental logic for both sites was the same, the attaker32 web page failed but the seediot32 web page succeeded. This is because the browser uses the same origin policy.

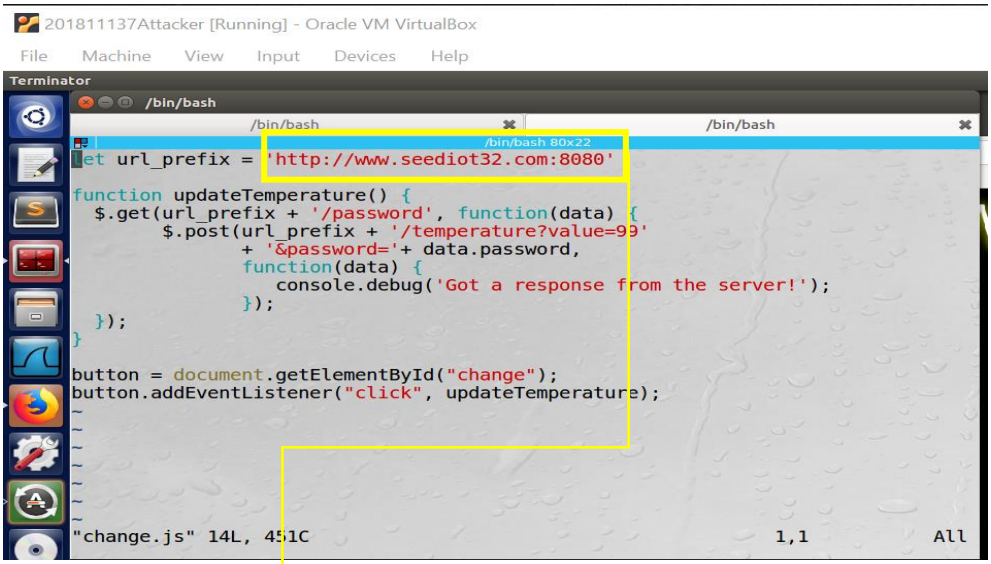
The browser allows only requests to a domain that originate from the same domain under the same origin policy (as in the second URL). Because the request on the second webpage comes from the attacker32.com domain to seediot32.com, the browser considers it a cross-origin request and so blocks it.

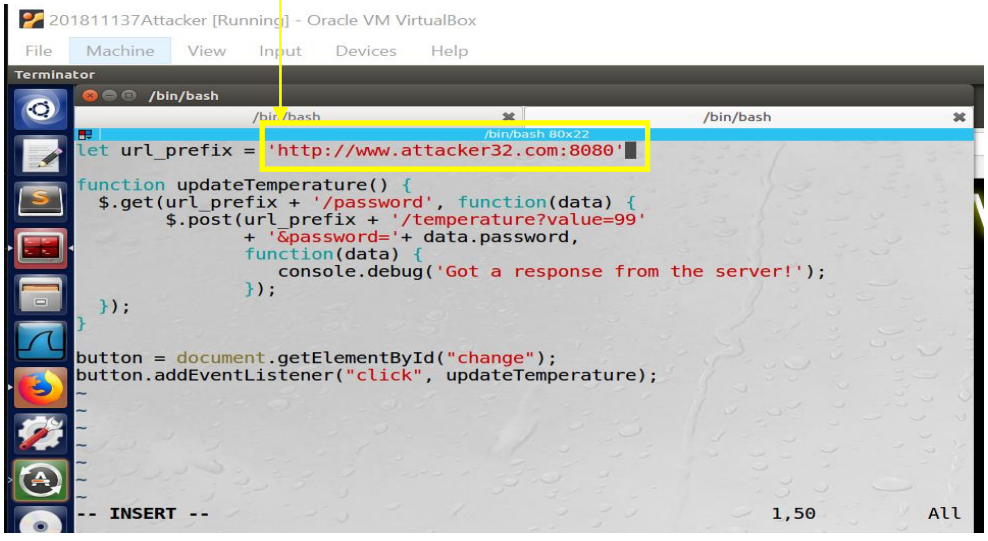
4.7 Task 7: Defeat the Same-Origin Policy Protection.

#Step 1: Modify the JavaScript code:

JavaScript code from www.attacker32.com:8080/change is saved in the attacker's virtual machine in the file attacker vm/rebind malware/templates/js/change.js. Because the page is served by the www.attacker32.com server, you can only interact with it using the same origin policy. As a result, instead of www.seediot32.com:8080, we must alter the first line of code to:

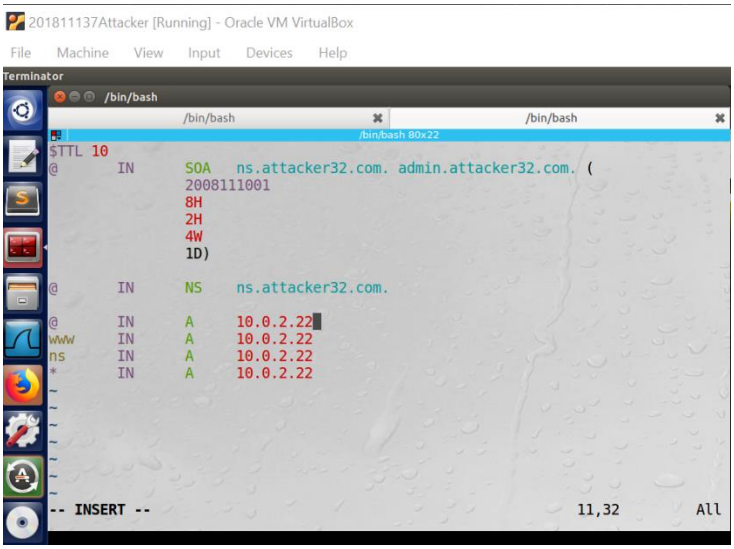

201811137Attacker [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
/bin/bash 80x22
[12/21/21]seed@VM:~$ ls
android examples.desktop Pictures suffix task33_a.bin Templates
bin get-pip.py prefix suffix_ task33_b.bin Videos
Customization lib prefix.txt suffix_1 task33.c
Desktop Music Public task_1 task33.o
Documents out.txt sample123 task33_1 task3.c
Downloads p source task33_2 task4.c
[12/21/21]seed@VM:~$ cd Desktop/
[12/21/21]seed@VM:~/Desktop$ cd attacker_vm
[12/21/21]seed@VM:~/.../attacker_vm$ ls
attacker32.com.zone rebind_malware start_webserver.sh
[12/21/21]seed@VM:~/.../attacker_vm$ cd rebind_malware/
[12/21/21]seed@VM:~/.../rebind_malware$ ls
config.py __init__.py __pycache__ templates
[12/21/21]seed@VM:~/.../rebind_malware$ cd templates/
[12/21/21]seed@VM:~/.../templates$ ls
change.html css index.html js
[12/21/21]seed@VM:~/.../templates$ cd js
[12/21/21]seed@VM:~/.../js$ ls
change.js jquery-2.2.4.min.js main.js
[12/21/21]seed@VM:~/.../js$ sudo vim change.js


201811137Attacker [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
/bin/bash 80x22
let url_prefix = 'http://www.seediot32.com:8080'
function updateTemperature() {
$.get(url_prefix + '/password', function(data) {
$.post(url_prefix + '/temperature?value=99'
+ '&password='+ data.password,
function(data) {
console.debug('Got a response from the server!');
});
});
}
button = document.getElementById("change");
button.addEventListener("click", updateTemperature);
"change.js" 14L, 451C 1,1 All


201811137Attacker [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
/bin/bash 80x22
let url_prefix = 'http://www.attacker32.com:8080'
function updateTemperature() {
$.get(url_prefix + '/password', function(data) {
$.post(url_prefix + '/temperature?value=99'
+ '&password='+ data.password,
function(data) {
console.debug('Got a response from the server!');
});
});
}
button = document.getElementById("change");
button.addEventListener("click", updateTemperature);
-- INSERT -- 1,50 All

#Step 2: Conduct the DNS rebinding:

Our JavaScript code sends a request to www.attacker32.com, which is then returned to the attacker VM. This is not what we want; instead, we want the request to be sent to the iOT server. DNS rebinding methods can be used to do this. We first map www.attacker32.com to the IP address of the attacker VM, so the user may access the real page at http: [www.attacker32.com/change. We remap www.attacker32.com before clicking the button on the website. The request generated by the button will be routed from the host name to the IP address of the iot server. This is precisely what we desire.



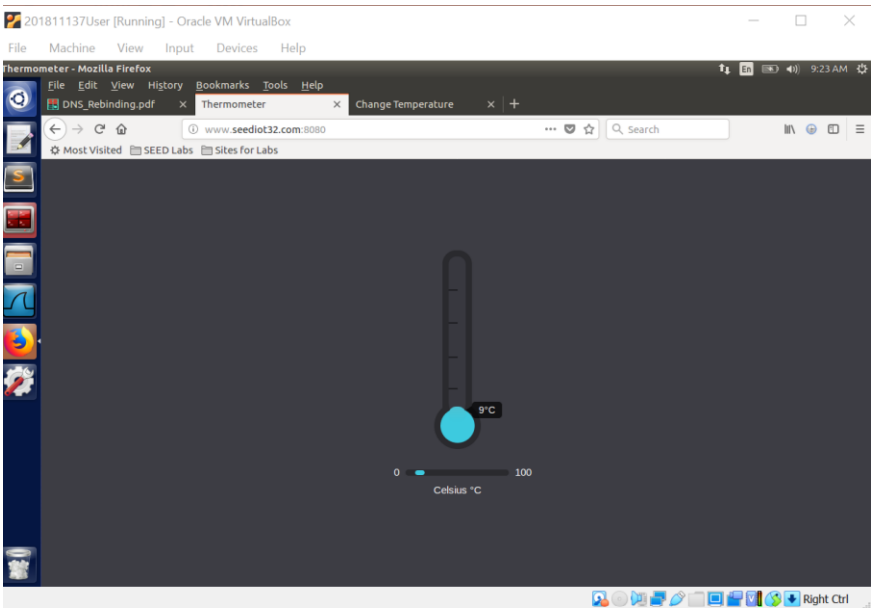
To refresh, change the relevant file:

The same-origin policy is a browser security feature that prevents client scripts from different origins from reading and writing to one other's resources without express permission. As a result, the js script on xyz.com employs ajax to read file data from abc.com and will be refused. Same-origin restrictions limit the ability of documents or scripts loaded from the same source to interact with resources from another source. This is a critical security feature for isolating possibly harmful files. The same-origin policy does not apply to links, redirects, and form submissions on sites that are not subject to the same-origin policy. 2. It is feasible to incorporate cross-domain resources. However, the loaded material cannot be read or written by JavaScript. For instance, in the page >/script >, script src="..."embeddedimg>,link>,iframe>

4.8 Task 8: Launch the Attack.

Access the following URLs using the user machine:

When the 10-second countdown reaches zero, the temperature is adjusted to 9 degrees:



6 Summary:

Task 1: I lower the Firefox DNS caching duration to 10 seconds in order to speed up our assault. The IoT web server was then started on the User VM (10.0.2.22) with the name - www.seediot32.com. I must use 10.0.2.21 as the local DNS server on the user workstation 10.0.2.22, finally I tested by using command dig.

Task 2: I download the FLASK web framework, which is used to develop the IoT server, and then I start the IoT server by having run the prepared script on port 8080 of the local machine. Finally, I test the IoT server by loading the URL: (www.seediot32.com:8080) and observing the web page of a thermostat (IoT device).

Task 3: I used sudo pip3 install Flask==1.1.1 to install Flask on the Attacker VM, then I started the Attacker's web server by executing the prepared script on port 8080 of the local machine, and finally, I tested the Attacker's server by browsing the URL: (localhost:8080) on the Attacker VM.

Task 4: I have the required zone file – attacker32.com – in the /etc/bind folder. I also added the zone entry to /etc/bind/named.conf, so that the aforementioned zone file is utilized by the BIND9 server. After applying these changes, I restarted the server and used the dig command to see whether I got the same result as in the zone file.

Task 5: I was editing the /etc/bind/named.conf file and added the following zone entry. This entry specifies that any requests for the attacker32.com domain on the Local DNS (10.0.2.22) should be routed to 10.0.2.20. The DNS server is then restarted using the following command: To restart the service, type sudo service bind9 restart. I ran the dig command on the User VM to see whether I got the same result as in the zone file on the Attacker Machine.

Task 6: I loaded the following two pages on the user machine, and when I clicked the "click" button, I saw that the temperature on the IoT device had changed. Then I lower the temperature to 50 degrees and click on the "click" button again, this time from a different homepage. The temperature does not change, and the online console displays an error message. Despite the fact that the fundamental logic for both sites was the same, the attaker32 web page failed but the seediot32 web page worked. This is because the browser uses the same origin policy. The browser allows only requests to a domain that originate from the same domain under the same origin policy (as in the first case). Because the request on the second webpage comes from the attacker32.com domain to seediot32.com, the browser considers it a cross-origin request and so blocks it.

Task 7: I was able to circumvent the same origin policy by taking advantage of the fact that SOP regulation is applied based on the host name rather than the IP address. To comply with SOP, we alter the following code to make the attacker32.com webpage connect with just the attacker32.com pages and not the seediot32 web page. I restarted the web server on the attacker's VM after making the update. in the following step Now, because I want the requests to go to the IoT server rather than the attacker's website, I used the DNS Rebinding method to first map the attacker32.com domain to the real IP address of the attacker VM and then connect the same domain to the IoT server's IP address, i.e., User VM.

Task 8: Similarly, I was carrying out the automated assault by utilizing the following code: (attacker vm/rebind malware/templates/js/change.js.), which sends the set-temperature request once the timer hits 0. I was now using the same procedure as previously, first linking the www.attacker32.com zone to the attacker, then loading the page on the User VM, and

then doing a DNS rebinding attack to link the www.attacker32.com zone to the IoT server (user machine).

7 Conclusion:

Finally, based on my notes, I can conclude that this SEED lab the DNS rebinding attacks, which may be applied to the links of Internet of Things devices, which we can be certain that no house will be devoid of them in our time or in the near future.

I reach the conclusion that this type of device cannot be accessed directly - despite the simplicity of the design of their sites - because I believe that most of their focus is on the device and how it works without focusing on the sites linked to these devices - but through such a laboratory we note that many IoT devices have a simple built-in web server, so users can communicate with these devices through web apis.

These internet of things devices are often protected by firewalls and cannot be accessed directly from the outside. Many IoT devices do not employ a robust authentication method as a result of this form of security. It is simple to compromise their security if an attacker can find a method to interact with them.

8 References:

- Secure Computer Networks course lectures / Moodle
- Attacking Private Networks from the Internet with DNS Rebinding https://medium.com/@brannondorsey/attackingprivate-networks-from-the-internet-with-dns-rebindingea7098a2d325
- Computer & Internet Security: A Hands-on Approach. Wenliang Du chapter 18: DNS rebinding attack page 415