UDP Amplification: Data Based Mitigation

Michael Woodham, William Hupp, Neet Patel

I INTRODUCTION

Denial of Service attack (DoS) is a type of a network attack where an attacker is able to significantly slow down, or even crash a victim network service. The attacker is able to achieve this by sending an unusually large number of network packets. The victim's service, as a result, is unable to handle these many packets due to hardware limitations. These packets, which create a flood of network traffic on the victim's side, cause a slow down or crash. In a Distributed Denial of Service attack (DDoS), an attacker is able to establish a network of infected computers, also referred to as a Botnet, which further act as nodes that can inflict DoS attacks. For our research, we examine a form of DoS attack called a Distributed Reflective Denial of Service Attack (DRDoS).

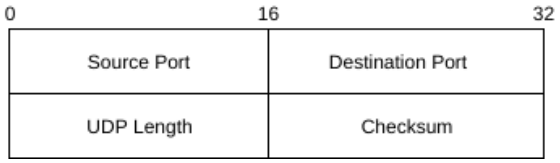
UDP is a connectionless protocol that does not validate source Internet Protocol (IP) addresses. Unless the application-layer protocol uses countermeasures such as session initiation in Voice Over Internet Protocol, an attacker can easily forge the IP packet datagram (a basic transfer unit associated with a packet-switched network) to include an arbitrary source IP address. When many UDP packets have their source IP address forged to the victim IP address, the destination server (or amplifier) responds to the victim (instead of the attacker), creating a reflected denial-of-service (DoS) attack.

Certain commands to UDP protocols elicit responses that are much larger than the initial request. Previously, attackers were limited by the linear number of packets directly sent to the target to conduct a DoS attack; now a single packet can generate up to 100 times the original bandwidth. This is called an amplification attack, and when combined with a reflective DoS attack on a large scale, using multiple amplifiers and targeting a single victim, DDoS attacks can be conducted with relative ease.

To understand UDP amplification attacks you first have to remember the differences between TCP and UDP traffic. At a high level, TCP traffic requires a full two-way negotiation between both devices before any real communications can begin. This negotiation is called the three-way handshake. As for DDoS attacks, this means that you can't really spoof TCP-based source address (with the exception of TCP negotiation attacks like SYN floods)**.** In other words, a TCP attack usually has to come from a computer the attacker controls. UDP, on the other hand, is a connectionless protocol. It doesn't require a prenegotiation, nor necessarily a response from the receiving device. There are other connectionless protocols as well, such as ICMP, which have a couple of advantages for DDoS attacks that we'll talk about later.

DRDoS attacks take advantage of response amplification. These amplification attacks work very well with networks that use UDP protocols since UDP doesn't require further validation of source or destination IP addresses. See diagram below for UDP packet structure. Using this protocol, an attacker is able to send relatively small amount of network packets to victim nodes, typically servers, and these servers in turn amplify the response significantly,. The response is send to another victim whose IP is spoofed by the attacker.In doing so, DRDoS attacks aim to exhaust the victim's bandwidth.

For our research, we first examined previous work done on using DRDoS attacks on UDP protocols to get a grasp on the protocols that we wanted to test on. Then we followed the experiment model that the previous research used in order to replicate their results. The experiment model simply illustrates how the DRDoS attack is carried out and will be shown in the next section. Thirdly we perform the experiment using our own machines and using the experiment model as guide. Lastly we mention some limitations we encountered in our experiment.



II BACKGROUND

As Rossow , laid out in his paper "Amplification Hell" the basic structure of amplification attacks. UDP amplification attacks rely on the fact that UDP protocols do not validate the packet sender's IP. This allows for any attacker to spoof a victim's IP and send them unusually large network track using intermediary systems as amplifiers. Amplifiers are systems which send a large amount of network traffic back to the victim, responding to a relatively small amount of request packets sent by the attacker. Reflective means that the attacker doesn't directly send the network traffic to the victim, but instead uses the mentioned intermediary systems to do so. Rossow explains that many of the past attacks routinely abused DNS servers because modern DNS servers use newer specifications, i.e DNSSEC, which increases the response size, making them good amplifiers. In particular the attackers use the ANY request to send to DNS servers, with the server than sending back the response to a victim machine. We selected this as one of the protocols to use for our experiment.[1]

Rossow examined other UDP based protocols such as SNMP, NTP, NetBios, SSDP, which are network services, QOTD and Chargen, which are legacy services, and 8 others which range from services that are P2P networks to file sharing networks. Using these protocols and their "threat model", Rossow used the bandwidth amplification factor (BAF) to quantify the level of amplification done for each protocol in a DRDoS attack. BAF is calculated as the percentage of the ratio of the length of the payload received by the victim to the length of the payload send by the attacker. Using this, they found that NTP had on average the highest BAF with QOTD being third and DNS falling in between the range along with the rest of the examined protocols.

NTP servers are excellent amplifiers because they support the monlist request. This request sends as a response the list of the last 600, sometime more, clients that connected to the NTP server. The request is only no more than 8 bytes long however, the response can be larger than 440 bytes for each packet the server sends back, totaling around 100 packets for the complete response. We therefore chose this protocol as one of the protocols for our own experiment. Lastly, we selected the QOTD service as another protocol to use for our experiment because the service can send an amplified response to a request as long as a single byte.

III METHOD

For our experiment, we used VMWARE and set up a virtual machine that used Ubuntu 16.04 server image as the operating system. This VM acted as our victim which shall receive all of the responses from the intermediary servers. For our intermediary servers, we used FSU's DNS server for our DNS ANY request experiment as well as for our NTP monlist request since the FSU DNS server is also an NTP server. For QOTD server, we established our own server on a different machine, using BSD sockets. Lastly our attacker machine used a Python script along with the Scapy module to craft a packet for a particular protocol and execute a DRDoS attack using the experiment model.

For DNS ANY request we first established FSU's IP name using Linux's dig command as follows:
dig +short NS fsu.edu
Which gave us the following result:
dnsa.fsu.edu.
dns1.fsu.edu.
trantor.umd.edu.
nsx.lbl.gov.
dns2.fsu.edu.
We further used the nslookup command, to get the authoritative name server, as follows:
nslookup
> set querytype=soa
> fsu.edu

This gave us the result:
Server: 128.186.120.179

[1] Amplification Hell: Revisiting Network Protocols for DDoS Abuse Christian Rossow VU University Amsterdam, The Netherlands Horst Gortz Institute for IT-Security, Ruhr University Bochum, Germany

Address: 128.186.120.179#53

Non-authoritative answer:
fsu.edu

origin = dns1.fsu.edu
mail addr = hostmaster.fsu.edu
serial = 2018111623
refresh = 3601
retry = 1200
expire = 604800
minimum = 86400

Authoritative answers can be found from:
dns1.fsu.edu internet address = 128.186.6.103

We further confirm this using the dig commands and their results on the dns1.fsu.edu and dns2.fsu.edu domain names:
dig @8.8.8.8 +short dns1.fsu.edu
128.186.6.103

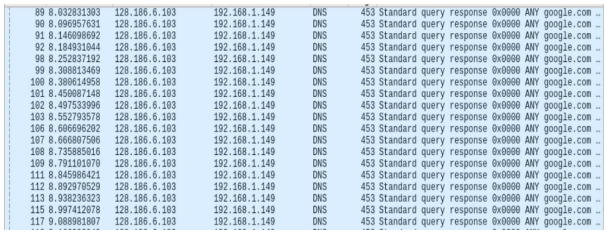
dig @8.8.8.8 +short dns2.fsu.edu
128.186.8.8

We used the ifconfig command on our victim vm to obtain its IP.

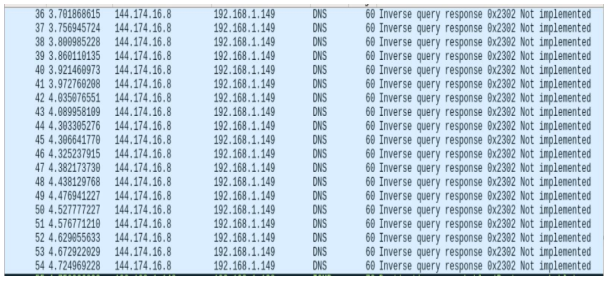
For NTP protocol and attack, we used the same FSU DNS server, 128.186.6.103. For QOTD, we created our own server using C and BSD sockets on a different machine that listened for incoming UDP connection on port 17. After establishing the requirement for each protocol, we constructed a single, interactive and threaded, Python script which allows the user to select the type of protocol they want to attack and the source and destination IP's. Currently, we use a fixed number of threads, each of which calls the Scapy's send method to send the packet indefinitely in a while loop. The packet is a global variable constructed after user input, so the threads simply have to call the send method.

IV RESULTS

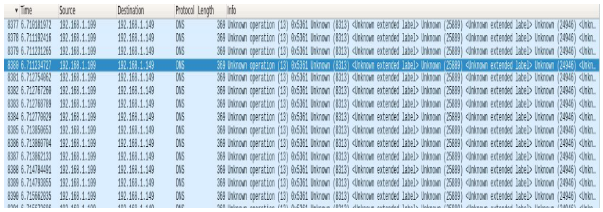
To examine our results, we used Wireshark to capture packets on the victim VM. For DNS ANY request our result are as follows:



For NTP our results are as follows:



For QOTD, our results are as follows:



As shown by Wireshark captures, our DNS ANY attack yielded a packet response of length 453 bytes to the victim when the request length was 81 bytes. For NTP we were not able to get a response back of any meaningful amplification, and we further explain this in the next section. For QOTD we were able to get 369 bytes in each response packet sent to the victim from only a single byte sent in the request packet.

V DRAWBACKS

We were unable to instantiate an amplification attack using Florida State's DNS servers outside of the Florida State network. This is due to a firewall issue preventing ANY requests to be instantiated outside of Florida State's network. When the dig command is issued we get a kickback with the following output:

dig @128.186.6.103 google.com ANY
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

NTP requests also do no longer work on Florida State's servers, whilst the monlist command is available to use on NTP servers prior to version 4.2.7 by default. By upgrading a NTP server to 4.2.7 or above, the command

is disabled. As a result when we send 98 bytes asking for the monlist command of NTP, the victim will only receive 60 bytes stating that the inverse query is not implemented, essentially the inverse of an amplification attack.

We were unable to find a server which still supports QOTD over UDP. While we were able to find servers which supported it over TCP, this would not work for our desired implementation for rather obvious reasons. As a result, we crafted our own UDP server in C++ which supports the protocol proving its viability.

In testing we were attacking dynamic IP addresses instantiated by a NAT protocol. In order to attack each other's machines, we had to be on the same router within the same subnet, so the server would know where to redirect the request when the source IP address was spoofed. By this, in a client based attack, we found that the IP address could not be dynamic. Therefore, attacking clients rather than servers would prove to be an ineffective use of this type of attack.

VI Conclusion

The general idea behind all amplification attacks is the same. To IP spoof, an attacker sends forged requests to a vulnerable UDP server. The UDP server, not knowing the request is forged, politely prepares the response. The problem happens when thousands of responses are delivered to an unsuspecting target host, overwhelming its resources - most typically the network itself.

We implemented three different UDP amplification attacks in a multi-threaded environment, DNS ANY, NTP, and QOTD. We proved the viability of all of these attacks and how they could be used within a distributed attack to compromise a server.

In the future we would like to implement a scenario where we were able to scan appropriate DNS and generalized servers to find attacks which work on these servers and compile a library of vulnerable of sources that can be used.