Stevens Institute of Technology

CS-573-Fundamentals of Cyber Security Project Report

DDoS Attacks: Detection, Mitigation Approaches and impact in Cloud Computing

Group 6

Anuja Phadnis

Jeet Patel

Maitrey Prajapati

Sumit Oberoi

Vidya Maiya

Yangyang Liu

To: Professor Edward Amoroso

Date: 2 ***Dec 2019***

Abstract

DDoS security for cloud computing will be a very complicated issue in the following years as it has been one in the last few years. The main reason is that cloud computing is based on distributed computing resources that are usually shared and uses different types of virtualization computing technologies which all contribute to make the DDoS security framework a lot more complex and harder to control. The main reason is that all resources can be very dynamic, they can move or migrate from one physical host to another and they can be extended upon demand. All of that creates a higher complexity of maintaining DDoS security policies for this environment. Due to the nature of cloud computing, the methodologies for preventing or stopping DDoS attacks are quite different compared to those used in traditional networks. Part of the effort to address these complexities in the DDoS security is by trying to develop new virtualized security technologies that will cause the security network manager in the end to control the security policy in a way that will move with the resource or will be extended if the resource is being extended as well. By that it will ease the way that DDoS security can be controlled in a cloud computing environment.

1. Introduction

In this section we will go over

1. Cloud Computing and types
2. DOS and DDOS Attacks General Overview
3. Types of DDOS Attacks
4. A brief introduction to DDOS Attack on Cloud

1.1 Cloud Computing:

If you are running a website or an app of a fair scale you need to deploy your codebase somewhere, provide a mechanism for your end-user to see it and you also want to protect it against hackers and other nuts and bolts to screw before the app is ready to be showcased. You have two options either to build everything on your hire IT team, buy servers and more servers if you are a growing business and other pieces of equipment or you can approach Amazon to rent their architecture and use it to deploy your app. Renting architecture from Amazon, Microsoft or Google and using various services that come along with it is cloud computing.

Cloud computing is the utilization of hardware and software combined to provide services to end users over a network like the internet. It includes a set of virtual machines that simulate physical computers and provide services, such as operating systems and applications

1.1.1 Deployment Models and architecture:

A cloud computing structure relies on three service layers: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS)

IAAS: IaaS gives users access to physical resources, networks, bandwidth, and storage eg. AWS, Azure, Google Cloud Platform and others

PAAS: PaaS builds on IaaS and gives end users access to the operating systems and platforms necessary to build and develop applications, such as databases. A common example for this is Heroku. Major IAAS providers has smaller subset which is bundled as PAAS

SAAS: SaaS provides end users with access to software applications. This is what needs to be deployed

An analogy to explain these three layers is food. SAAS is like a restaurant or home delivery service you just tell the waiter what food you want. IAAS is like you have a rented kitchen but you have rent oven, utensils and bring ingredients to cook your food. PAAS is you give someone your ingredients to cook food for you.

DDOS attack can originate at any of these layers.

1.2 DDOS (Distributed Denial of Service) Attacks:

Continuing over our food analogy. Imagine you are running a restaurant and food industry is competitive space to be in so you have a competitor who hires a group of young teenagers to go to your restaurant and not order anything but keep messing with you by asking about your menu and various items on it to keep you occupied. These teenagers are creative and start calling your restaurant to conduct similar activities. Now you won't be able to serve your real customers as you are dealing with these teenagers. It affects your genuine users as they cant come to your restaurant as it is always filled by these teenagers and they can't place orders via phone as well. Due to lack of order you don't make profit and have to close the shop or apologise to genuine users for not handling them. This is DDOS Attack. Attackers are teenagers, Cloud or your infrastructure is your restaurant and benign users are people coming to your app or website.

To understand DDOS we need to first understand DOS (Denial of Service Attacks):
DoS attackers target the server, which is providing a service to its consumers. Behaving like a legitimate customer, DoS attackers try to flood active server in a manner such that the service becomes unavailable due to a large number of requests pending and overflowing the service queue.

1.2.1 There are three ways to conduct DDOS Attacks

1. Buffer Flow Attacks: The most common DoS attack. The concept is to send more traffic to a network address than the programmers have built the system to handle.
2. ICMP Flood: It leverages misconfigured network devices by sending spoofed packets that ping every computer on the targeted network, instead of just one specific machine. The network is then triggered to amplify the traffic. This attack is also known as the smurf attack or ping of death.
3. SYN flood: In this attacker sends a request to connect to a server, but never completes the handshake. Continues until all open ports are saturated with requests and none are available for legitimate users to connect to

Now we need to understand D(Distributed) in DDOS Attacks. A DDoS attack occurs when multiple systems orchestrate a synchronized DoS attack to a single target. The essential difference is that instead of being attacked from one location, the target is attacked from many locations at once.

Distributed denial-of-service (DDoS) attacks constitute one of the largest threats faced by Internet users and cloud computing services. DDoS attacks target the resources of services, lowering their ability to provide optimum usage of the network infrastructure. The objective of DDoS attacks is to consume resources, such as memory, CPU processing space, or network

bandwidth, in an attempt to make them unreachable to end users by blocking network communication or denying access to services.

1.3 Types of DDOS Attacks:

1.3.1 Infrastructure Level Attacks:

Attacks at Transport and Network layer in OSI Model , are typically categorized as Infrastructure layer attacks. These are also the most common type of DDoS attack and include vectors like synchronized (SYN) floods and other reflection attacks like User Datagram Packet (UDP) floods. These attacks are usually large in volume and aim to overload the capacity of the network or the application servers. But fortunately, these are also the type of attacks that have clear signatures and are easier to detect

1.3.2 Application Level Attacks:

Attacks at Presentation and Application layer in OSI Model, are often categorized as Application layer attacks. While these attacks are less common, they also tend to be more sophisticated. These attacks are typically small in volume compared to the Infrastructure layer attacks but tend to focus on particular expensive parts of the application thereby making it unavailable for real users. For instance, a flood of HTTP requests to a login page, or an expensive search API

1.4 DDOS Attacks on Cloud:

According to Cloud Security Alliance, DDoS is one of the top nine threats(14%) to cloud computing environment. Cloud environment when the workload increases on a service, it will start providing computational power to withstand the additional load(called Auto Scaling).

Which means Cloud system works against the attacker, but to some extent it supports the attacker by enabling him to do the most possible damage on availability of service, starting from single attack entry point.

Cloud service consists of other services provided on the same hardware servers, which may suffer by workload caused by flooding. Thus, if a service tries to run on the same server with another flooded service, this can affect its own availability.

Another effect of a flooding is raising the bills for Cloud usage drastically. The problem is that there is no “upper limit” to the usage.

And one of the potential attacks to cloud environment is neighbor attacks i.e. VM can attack its neighbor in same physical infrastructures and thus prevent it from providing its services. These attacks can affect cloud performance and can cause financial losses and can cause harmful effects in other servers in the same cloud infrastructure

1.4.1 How are DDOS Attack carried out in Cloud:

This is a very general overview. We will go in detail later:

- DDOS Attack on cloud can do more damage as compared to traditional deployment thanks to Autoscaling
- Once a VM gets deployed, it starts as a “Normal load VM”. Now, let us assume that the DDoS attack has started and consequently VM gets overloaded
- The overload condition triggers auto-scaling features of cloud resource allocation
- Overloaded VM may be given some more resources or migrated to a higher resource capacity server or may be supported by another instance started on another server
- This situation may last till service provider can pay or cloud service provider consumes all the resources. Finally, it will lead to “Service Denial”.

.

2. Infrastructure Layer Attacks

The infrastructure level DDoS attacks utilize weaknesses in layer 3- Network layer and layer 4-Transport layer of the OSI model to render the target inaccessible. The attackers typically flood the victim with a high volume of packets or connections, overwhelming servers, or bandwidth resources and ultimately cause a service disruption by consuming all the available capacity of the application servers. As a result, these are also known as Volumetric DDoS attacks/state exhaustion attacks.

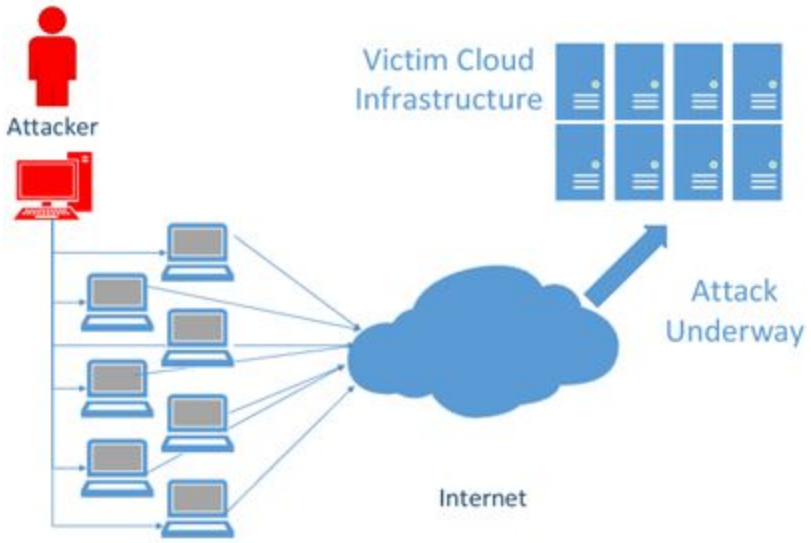

Diagram 2.1 Infrastructure attack

2.1 How does an infrastructure layer attack work?

As mentioned above, infrastructure DDoS attacks mainly target the network and transport layers of communication system. The so called “transport” layer of the network stack specifies the protocol (e.g., TCP or UDP) by which two hosts on a network communicate with one another. The attacks are designed to flood a network interface with attack traffic in order to overwhelm its resources and deny the ability to respond to actual traffic.

Layer 3 and 4 attacks are difficult—if not impossible—to mitigate. If an attacker can send more traffic than a network link can handle, no amount of additional hardware resources will help to mitigate such an attack. For example, if you have a router with a 10Gbps port and an attacker sends you 11Gbps of attack traffic, no amount of intelligent software or hardware will allow you to stop the attack if the network link is completely saturated.

Because an attacker launching a layer 3/4 attack doesn’t care about receiving a response to the requests they send, the packets that make up the attack do not have to be accurate or correctly formatted. Attackers will spoof all information in the attack packets, including the source IP, making it look as if the attack is coming from a virtually infinite number of sources. As packet data can be fully randomized, techniques like upstream IP filtering become virtually useless.

Infrastructure attacks nearly always originate from a number of sources. These many sources each send attack traffic to a single Internet location creating a tidal wave that overwhelms a target’s resources. In this sense, the attack is distributed. The sources of attack traffic can be a group of individuals working together, a botnet of compromised PCs, a botnet of compromised servers, misconfigured DNS resolvers or even home Internet routers with weak passwords.[12]

2.2 Types of infrastructure DDoS attacks

Common DDoS attacks at infrastructure level are:

1. **TCP Syn Flooding Attacks**: This attacks exploit part of the normal TCP three-way handshake to consume resources on the targeted server and render it unresponsive.[13]
2. **UDP Flood Attacks**: Its a type of denial-of-service attack in which a large number of User Datagram Protocol (UDP) packets are sent to a targeted server with the aim of overwhelming that device's ability to process and respond.[13]
3. **DNS Amplification Attacks**: DNS amplification attack uses DNS queries. The attacker uses a compromised endpoint to send UDP packets with spoofed IP addresses to a DNS recursor. The spoofed address on the packets points to the real IP address of the victim. Each one of the UDP packets makes a request to a DNS resolver, often passing an argument such as "ANY" in order to receive the largest response possible. After receiving the requests, the DNS resolver, which is trying to be helpful by responding, sends a large response to the spoofed IP address. The IP address of the target receives the response and the surrounding network infrastructure becomes overwhelmed with the deluge of traffic, resulting in a denial-of-service.[13]
4. **ICMP Flood Attacks:** In ICMP flood attacks, the attacker overwhelms the targeted resource with ICMP echo request (ping) packets, large ICMP packets, and other ICMP types to significantly saturate and slow down the victim's network infrastructure.[13]

2.3 TCP Syn Flood Attacks

SYN flood attacks work by exploiting the handshake process of a TCP connection. [14]

Under normal conditions, TCP connection exhibits three distinct processes in order to make a connection.

Diagram 2.2 TCP Three Way Handshake

1. First, the client sends a SYN packet to the server in order to initiate the connection.
2. The server than responds to that initial packet with a SYN/ACK packet, in order to acknowledge the communication.
3. Finally, the client returns an ACK packet to acknowledge the receipt of the packet from the server. After completing this sequence of packet sending and receiving, the TCP connection is open and able to send and receive data.

To create denial-of-service, an attacker exploits the fact that after an initial SYN packet has been received, the server will respond back with one or more SYN/ACK packets and wait for the final step in the handshake.

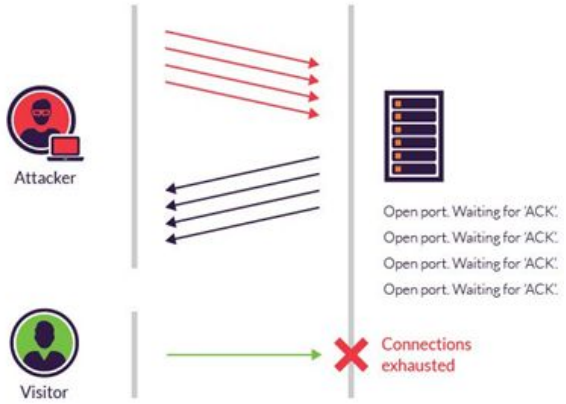

Diagram 2.3 TCP SYN Flood Attack Scenario

1. The attacker sends a high volume of SYN packets to the targeted server, often with spoofed IP addresses.
2. The server then responds to each one of the connection requests and leaves an open port ready to receive the response.

3. While the server waits for the final ACK packet, which never arrives, the attacker continues to send more SYN packets. The arrival of each new SYN packet causes the server to temporarily maintain a new open port connection for a certain length of time, and once all the available ports have been utilized the server is unable to function normally. [14]

The attacks can occur in three different ways:

1. **Direct Attack:** A SYN flood where the IP address is not spoofed is known as a direct attack. In this attack, the attacker does not mask their IP address at all. As a result of the attacker using a single source device with a real IP address to create the attack, the attacker is highly vulnerable to discovery and mitigation.
2. **Spoofed Attack:** A malicious user can also spoof the IP address on each SYN packet they send in order to inhibit mitigation efforts and make their identity more difficult to discover.
3. **Distributed attack (DDoS):** If an attack is created using a botnet, the likelihood of tracking the attack back to its source is low. For an added level of obfuscation, an attacker may have each distributed device also spoof the IP addresses from which it sends packets. If the attacker is using a botnet, they generally won't care about masking the IP of the infected device.[14]

2.4 UDP Flood Attacks

In a UDP Flood, DDoS attackers send highly-spoofed UDP (user datagram protocol) packets at a very high packet rate. The victim's network is overwhelmed by the large number of incoming UDP packets. This attack normally consumes network resources and available bandwidth, exhausting the network until the service is disrupted.

UDP is a networking protocol that is both connectionless and session-less. Unlike TCP, UDP traffic does not require a three-way handshake. Without an initial handshake to ensure a legitimate connection, UDP channels can be used to send a large volume of traffic to any host. There are no internal protections that can limit the rate of a UDP flood. As a result, UDP flood

DOS attacks are exceptionally dangerous because they can be executed with a limited amount of resources.[15]

A UDP flood works primarily by exploiting the steps that a server takes when it responds to UDP packet sent to one of it's ports. Under normal conditions, when a server receives a UDP packet at a particular port, it goes through two steps in response:

1. The server first checks to see if any programs are running which are presently listening for requests at the specified port.
2. If no programs are receiving packets at that port, the server responds with ICMP (ping) packet to inform the sender that the destination was unreachable.

In UDP Flood attack, the attackers send a big bunch of UDP packets to random ports of their target server. The server has to respond to all of them, as stated above. First, it needs to check if any of its applications are listening for communication at these ports. When it finds none, the server needs to send back information about the destination being unavailable. It does so through the internet protocol called ICMP which is used for sending out error messages.[15]
Every packet sent to the targeted server needs to be replied to. This eats through server's connectivity and sometimes other resources. This huge amount of traffic – that is in part generated by the server itself – makes a regular connection to the server impossible.

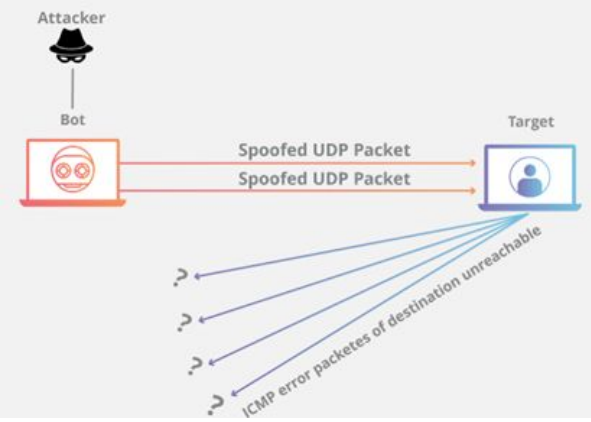

Diagram 2.4 UDP Flood Attack Scenario

3. Application Layer Attacks

SaaS model of cloud uses SOA and web services to present always accessible services. Users need to access Cloud from anywhere and this availability comes from presenting services as a

Web Service over the Internet. Web Services are intended to be accessible from different places and applications.

Application Layer Attacks occur at Layer 6 (Presentation) and Layer 7 (Application) in the OSI Model where common internet requests such as **HTTP GET** and **HTTP POST** occur. The **HTTP** and **XML (HX-DoS)** attacks can easily pass through firewalls and take down the server. In contrast to network layer attacks such as DNS Amplification, these layer 7 attacks are particularly effective due to their consumption of server resources in addition to network resources. For instance, a flood of HTTP requests to a login page, or an expensive search API, or DNS Query floods or even Wordpress XML-RPC floods (also known as Wordpress pingback attacks).

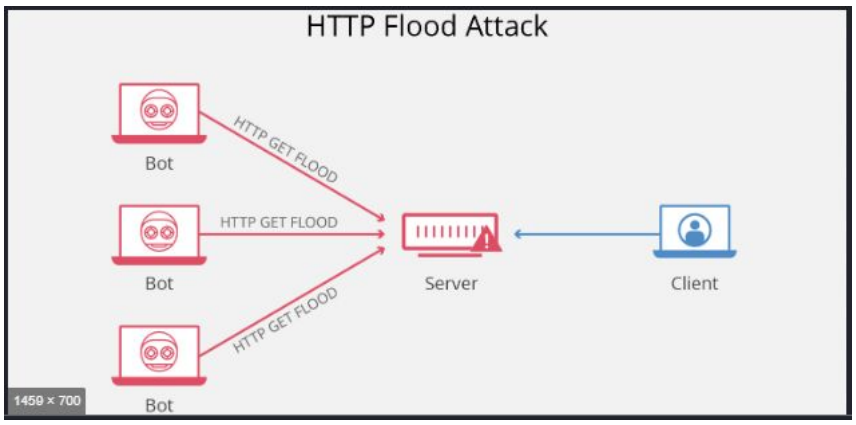

Diagram 3.1 HTTP Flood Attack

3.1 How does Application Layer Attack work?

Total bandwidth required to cause disruptive effect on the targeted server and the network is comparatively less. Because of this, application layer attacks are less expensive for threat actors to carry out and are perceived as harder for security solutions to detect than attacks aimed at the network layer.

Let's take a look at the difference in relative resource consumption between a client making a request and a server responding to the request. When a user sends a request logging into an online account such as a Gmail account, the amount of data and resources the user's computer utilize are minimal and disproportionate to the amount of resources
consumed in the process of checking login credentials, loading the relevant user data from a database, and then sending back a response containing the requested webpage.

Most of the times, a server receiving a request from a client must make database queries or other API calls in order to produce a webpage. When many devices target a single web property like during a botnet attack, the effect can overwhelm the targeted server resulting in denial-of-service to legitimate traffic. In many cases, simply targeting an API with Layer 7 attack is enough to take the server offline.

3.2 Why is it difficult to stop?

Distinguishing between attack traffic and normal traffic is difficult, especially in the case of an application layer attack. When a botnet is performing an HTTP flood attack against a victim's server, each bot in a botnet makes seemingly legitimate network requests. The traffic is not spoofed and may appear "normal" in origin.

Even if the defender tries to analyze identifiable attack patterns, a determined attacker will monitor the results of his attack and modify it to baffle a skilled and determined defender. Since active attackers are known to continually modify payload patterns to avoid simplistic DDoS mitigation, maintaining an ongoing list of known attack patterns quickly becomes impractical due to scale issues and the rate at which this list must be updated.

3.3 Types of Application Layer Attacks

Common application level attacks include :

1. **X-DOS**: X-DoS attack uses an XML message that is sent to a Web Server or Web Service. The attacker sends a SOAP message with an unlimited amount of opening tags in the SOAP request body, exhausting CPU usage. When an attacker launches the X-DoS attack, this causes to flood the network with XML message rather than packets, thus legitimate users will be deprived to use network communications. Also, when the web server is flooded with XML requests, its services would not be available to users.
2. **H-DOS:** H-DoS attacks are using a **HTTP Flooder** that starts up 1500 threads, it is able to send randomly a huge HTTP requests to a web server to use up all its communication channels. There is no method to differentiate between legitimate and illegitimate HTTP requests.
3. **Slowloris:** Slowloris is an application layer DDoS attack which uses partial HTTP requests to open connections between a single computer and a targeted Web server, then keeping those connections open for as long as possible, thus overwhelming and slowing down the target.
4. **Slow Post:** In a Slow Post DDoS attack, the attacker sends legitimate HTTP POST headers to a Web server. In these headers, the sizes of the message body that will follow are correctly specified. However, the message body is sent at a painfully low speed(1 byte/2min). When attackers launch hundreds or even thousands Slow POST

attacks at the same time, server resources are rapidly consumed, making legitimate connections unachievable.

5. **Slow Read:** A slow read DDoS attack involves an attacker sending an appropriate HTTP request to a server, but then reading the response at a very slow speed - the server assumes the client is actually reading the data and therefore keeps the connection open. This has the cumulative effect of consuming server resources, thus preventing legitimate requests from going through.

Let's explore 2 major Application Layer attacks - HTTP Flood and DNS Flood in depth in following 2 sections.

3.4 Dynamic HTTP Flood Attack

The HTTP protocol is an Internet protocol which is the basis of browser-based Internet requests, and is commonly used to send form contents over the Internet or to load web pages.

HTTP floods are designed to overwhelm web servers' resources by continuously requesting single or multiple URLs from many source attacking machines, which simulate HTTP clients such as web browsers (though the attack analyzed here does not use browser emulation). An HTTP Flood may consist of either GET (images and scripts), POST (files and forms) or combined GET and POST requests.

"**Dynamic**" HTTP flood is a DDoS attack which continuously changes the suffix of the HTTP request (i.e adding to the base URL 'learn.mydomain.com' randomly generated suffix '/nowqerwr/21dsa'). This forces services like CDNs to generate a request to the originating web server.

When the servers' limit of concurrent connections is reached, the server can no longer respond to legitimate requests from other clients attempting to connect, causing a denial of service.

Since HTTP flood attacks use standard URL requests, hence it may be quite challenging to differentiate from valid traffic.

3.5 Technical Analysis by importing TCP dump in WireShark:

1. **Example of TCP Connection**

   Before sending an HTTP request a TCP connection between a client and a server is established, using 3-Way Handshake (SYN, SYN-ACK, ACK), as seen in packets 3,12,13 in the example below:

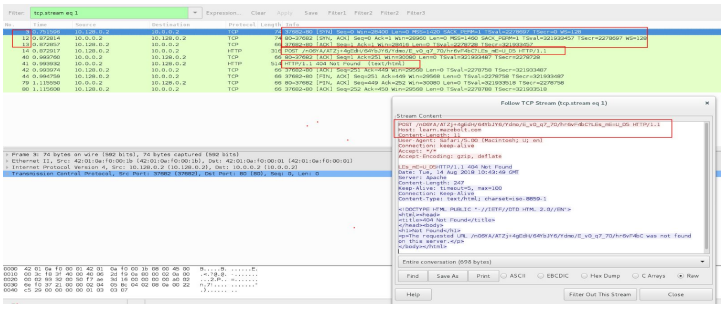

Diagram 3.2 Example of TCP Connection

2. **Example of Dynamic HTTP packets**

   An attacker (IP 10.128.0.2) sends GET and POST requests, changing the URL suffix dynamically, while the target (10.0.0.2) responds with HTTP/1.1 404 Not Found as can be seen below:

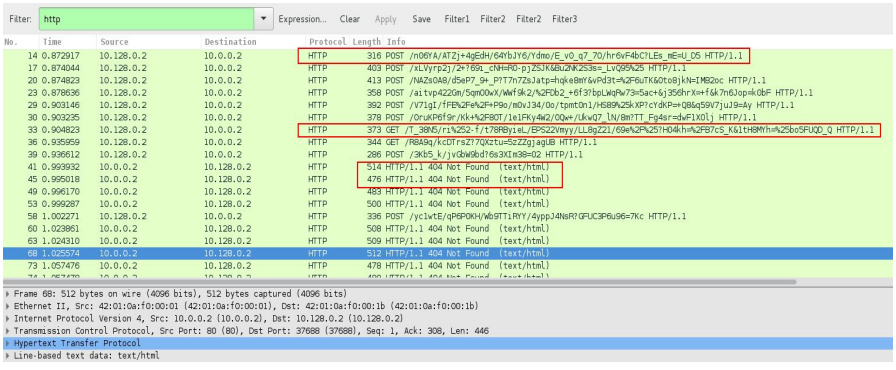

Diagram 3.3 Example of Dynamic HTTP packets

3. Example of Dynamic HTTP flood stats

The capture analyzed is around 8.7 seconds long and the average number of packets per second is around 80, with a rate of around 0.091 Mbps (considered low, actual attack rates could be significantly higher).

Traffic	Captured	Displayed	Displayed %	Marked	Marked %
Packets	702	702	100.000%	0	0.000%
Between first and last packet	8.723 sec				
Avg. packets/sec	80.481				
Avg. packet size	141 bytes				
Bytes	98787	98787	100.000%	0	0.000%
Avg. bytes/sec	11325.518				
Avg. MBit/sec	0.091				

Diagram 3.3 Example of Dynamic HTTP flood stats

3.6 DNS Flood Attack

To attack a DNS server with a DNS flood, the attacker runs a script , generally from multiple servers. These scripts send malformed packets from spoofed IP addresses. Since Layer 7 attacks like DNS flood require no response to be effective, the attacker can send packets that are neither accurate nor even correctly formatted.

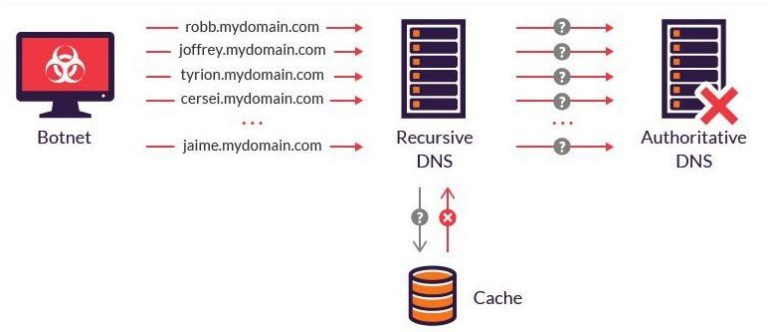

Diagram 3.3 DNS Flood Attack

Another common type of DNS flood attack is DNS NXDOMAIN flood attack, in which the attacker floods the DNS server with requests for records that are nonexistent or invalid. The

DNS server expends all its resources looking for these records, its cache fills with bad requests, and it eventually has no resources to serve legitimate requests.

4. Case Study 1 - DDos Attack on Github(2018)

On Wednesday, February 28, 2018 GitHub.com was unavailable from 17:21 to 17:26 UTC and intermittently unavailable from 17:26 to 17:30 UTC due to a distributed denial-of-service (DDoS) attack.

Between 17:21 and 17:30 UTC on February 28th Github identified and mitigated a significant volumetric DDoS attack. The attack originated from over a thousand different autonomous systems (ASNs) across tens of thousands of unique endpoints. It was an amplification attack using the **memcached**-based approach described above that peaked at **1.35Tbps via 126.9 million packets per second**. This was the largest seen successful DDos attack until the day.

GitHub briefly struggled with intermittent outages as a digital system assessed the situation. Within 10 minutes it had automatically called for help from its DDoS mitigation service, Akamai Prolexic. Prolexic took over as an intermediary, routing all the traffic coming into and out of GitHub, and sent the data through its scrubbing centers to weed out and block malicious packets. After eight minutes, attackers relented and the assault dropped off.

The first portion of the attack peaked at 1.35Tbps and there was a second 400Gbps spike a little after 18:00 UTC. This graph provided by Akamai shows inbound traffic in bits per second that reached their edge:

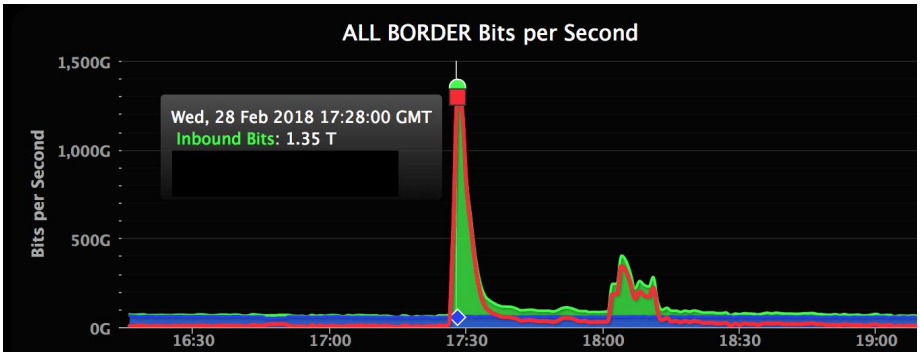

Diagram 4.1 All Border Bits per second

4.1 What actually happened?

On 27th of February a new attack was discovered which included the usage of memcached servers for **reflection** and **amplification** of packets. Memcached is meant to cache data and reduce the strain caused by memory intensive services.

GitHub called in assistance from Akamai Prolexic, which rerouted traffic to GitHub through its "scrubbing" centers, which removed and blocked data deemed to be malicious. Following eight minutes of the assault, the attackers called it off and the DDoS stopped.

4.2 Reflection Process in the attack

Usage of reflection with amplification is pretty common in attacks but the attack could be scaled to the next level when something like memcached is used.

A Fake GET REQUEST that could be used to reflect to the victims server can have format like following.

GET DATA Source_IP = X.X.X.X

Here the *DATA* is the address to the packet that needs to be sent. The *Source_IP* is the IP address of the victim that the hacker wants to target. Since there is no authentication process during the memcaching and UDP packets are used the hacker puts the IP address of the victim, which the memcache server considers as source of the request and responds with the *DATA* file to that address. This is how the reflection technique is used to target the victim.

4.3 Amplification process in the attack

Memcache store the data in key-value format, the default max size of this pair is 1MB. But with the latest update of version 1.4 it can be changed easily with minor configuration. Since the memcache doesn't require authentication the hacker can easily put in large chunks of files on the server which he/she could use to send it to the victim's server.

Memcached can have both UDP and TCP listeners and requires no authentication. Since UDP is easily spoofable, it makes this service vulnerable to use as a reflector. Worse, memcached can have an **amplification factor of over 500,000**, meaning a **203 byte request results in a 100 megabyte response**.

During this attack the hackers did the same thing they sent several hundreds of thousands of requests from multiple endpoints to multiple memcached servers with the IP of Github with the intention to flood the website with traffic, which resulted in average 126.9 million packets per second.

4.4 Why did this happen?

The hackers targeted the memcache servers which were on public internet, at the time of attack there were about 100,000 memcache servers on the public internet. Since this servers are not meant to use authentication, they are usually put behind the firewall and thus away from public but servers used in this attack were not behind any protection so anyone with the internet connection and IP address of the server can GET and POST *DATA* from the server.

Due to this design flaws of these third-party memcache servers, attack on Github was possible.

4.5 How was the attack stopped?

After about 10 minutes into the attack all the traffic was rerouted by the website through Akamai's DDos prevention scrubbing center. These centers are specially designed to handle the traffic of this scale. Alongside this these scrubbing centers are used for the filtering of spam data from the actual one and this way in the next few minutes the attack was stopped.

Akamai uses cloud platform to handle DDos attacks which is why they are able to handle traffic of this scale. Further they closed the port **11211** on their website which is default memcache port resulting in complete stoppage from any kind of traffic from any memcache servers.

5. Case Study 2 - DDos Attack on Github(2015)

On March 26, 2015, a very well-coordinated distributed denial of service (DDoS) attack was waged on GitHub, the heir apparent to the now-closing Google Code. GitHub characterized this as the largest DDoS in its history.

The Electronic Frontier Foundation (EEF) and security researchers Netresec name the Chinese government as the culprits of the attack, which lasted until March 31, 2015.

5.1 Deconstruction of the DDoS attack

The manner in which the attack was carried out is the following:

1. An innocent user is browsing the internet from outside China.
2. One website the user visits loads a JavaScript from a server in China, for example the Badiu Analytics script that often is used by web admins to track visitor statistics (much like Google Analytics).
3. The web browser's request for the Baidu JavaScript is detected by the Chinese passive infrastructure as it enters China.
4. A fake response is sent out (3 packets injected) from within China instead of the actual Baidu Analytics script. This fake response is a malicious JavaScript that tells the user's browser to continuously reload two specific pages on GitHub.com. This malicious JavaScript send a HTTP GET Requests to the Github's Servers every 2 seconds for two specific pages.

However, not all users loading JavaScripts from inside China are attacked in this way. Our analysis shows that only about 1% of the requests for the Baidu Analytics script are receiving

the malicious JavaScript as response. So in 99% of the cases everything behaves just like normal.

After the JavaScript loaded, the following behavior was observed in the affected network traffic:

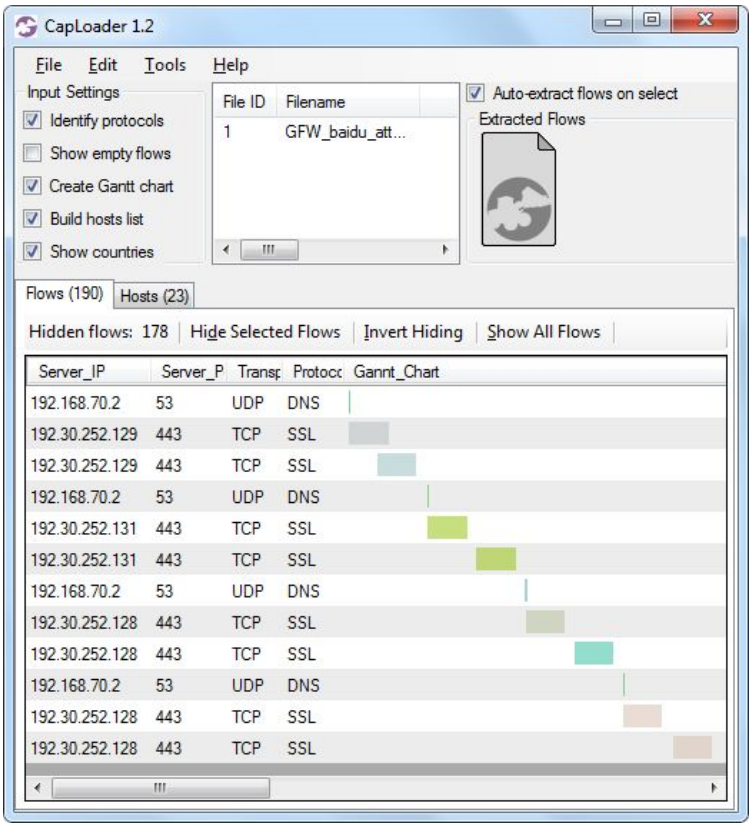

Diagram 5.1 affected network traffic

The script forced browser to connect to github.com (IP address 192.30.252.[128-131]) in an infinite loop.

5.2 The Injected JavaScript

The Injected JavaScript was somewhat obfuscated, but after some deobfuscation, the following code was obtained:

```
document.write("<script
src='http://libs.baidu.com/jquery/2.0.0/jquery.min.js'>\x3c/script>");
!window.jQuery && document.write("<script
src='http://code.jquery.com/jquery-latest.js'>\x3c/script>");
startime = (new Date).getTime();
var count = 0;
 function unixtime() {
   var a = new Date;
   return Date.UTC(a.getFullYear(), a.getMonth(), a.getDay(), a.getHours(),
a.getMinutes(), a.getSeconds()) / 1E3
}
url_array = ["https://github.com/greatfire", "https://github.com/cn-nytimes"];
NUM = url_array.length;

function r_send2() {
   var a = unixtime() % NUM;
   get(url_array[a])
}

function get(a) {
   var b;
   $.ajax({
      url: a,
      dataType: "script",
      timeout: 1E4,
```

```
        cache: !0,
        beforeSend: function() {
            requestTime = (new Date).getTime()
        },
        complete: function() {
            responseTime = (new Date).getTime();
            b = Math.floor(responseTime - requestTime);
            3E5 > responseTime - startime && (r_send(b), count += 1)
        }
    })
}
function r_send(a) {
    setTimeout("r_send2()", a)
}
setTimeout("r_send2()", 2E3);
```

As one can see, Line 10 displays the targeted URLS of Github.[10]

Also, these malicious JavaScript had a different Time-to-Live(TTL) values than the normal JavaScripts sent from the Baidu-search Engine. The normal JavaScripts have a TTL of 42 where as these malicious JavaScripts had a TTL value varying from 30 to 229. This made it difficult to traceback the location from where it was injected.

5.3 Conclusion of the Attack

The HTTP GET Requests from various infected web Browser's led to an overflow of Traffic in the Github Servers which could not handle this amount of GET Requests and were rendered unavailable for Github's legitimate users.Github was forced to render all its employees to try

and mitigate this attack. This attack was carried out for five whole days after which Github was able to deflect most of the attack traffic and render its service available to its legitimate users.

6. Future Direction and Conclusion

6.1 Lesson Learned

In conclusion, DDoS attack on cloud is a complex problem. The DDoS Attack happened in different layers need to be handled with different strategies.So, the lesson learned here is that we need to come up with the mitigation approach that can mitigate DDoS attack as well as make the infrastructure available at the same time, at least we need to minimize the downtime.

There are three things we need to do to combat DDoS attack on cloud. First, we need to suggest considering sustainability, collaboration, resource management, damage minimization, and availability while handling DDoS attacks in cloud computing. Second,we need to provide a multilevel collaborative solution framework that may be beneficial in designing efficient mitigation solutions.Finally, based on attack trend,we illustrate future surface learning.

6.2 Factors In DDoS Attacks

1. **Traffic filtering**: It is well-established that detection based on traffic filtering alone is not sufficient and foolproof. Modern sophisticated attacks evolve by varying their attack features to remain undetected by traffic filters.
2. **Managing costs**: DDoS attacks are compromising the finances of victim service owners. While designing mitigation solutions, the cost factor is important when managing the sustainability aspects.
3. **Services availability**: While mitigating DDoS attacks, a mechanism should exist to run services for benign users with minimum or no downtime.

4. **Collaboration**: Massive volumetric attacks, power attacks, and other sophisticated attacks are not fully detectable at the victim end. There are many other information/alert points in the cloud stack and Internet stack that may help in gaining important information about the likelihood of attacks. These alerts and subsequent actions based on these alerts may prove to be promising to combat attacks.

5. **Damage minimization**: DDoS mitigation should also provide for minimizing collateral damages as shown in the study by Somani et al.9 This can be ensured by isolating and monitoring the efforts at other components such as hypervisors and networks.

6. **Resource management**: DDoS attacks in the cloud has been evaluated as a resource management problem at the victim service end of several studies.17,18 The major idea behind these solutions is to provide a guarantee of resource contention-free execution of attack mitigation solutions in the presence of the attacks. These resource management-based methods are useful and cost-effective.[7]

6.3 Multi-level Solution Hierarchy

As the paper above has already covered that the DDoS attack on Cloud could happened in Application level or infrastructure level. And there are varieties of attacks. To make the infrastructure still remain sustainable and minimize the damage. The multi-level solution is the most efficient and useful solution for DDoS attack on cloud.

There are three level for this solution, which is application level system level and external level.Application Defense which is formed using attack prevention mechanisms. System Defense which is formed by three defense levels (VM/OS, hypervisor and cloud). External Defense which is formed using ISP level and third party defense. We can combine these different defense method to make this multi-level solution.

1. Application Defense : This is a design which considers defense at application level only. This level of defense is the most used and helps in the multi-tenant environment where each hosted VM should be isolated because of multiple virtualization and data security threats. Most

solutions in the literature follow this design but it is also clear that this design alone is not suitable to take care of aspects of cloud.

2. Application Defense + System Defense : If this design can be implemented with ease than it can be proven as one of the effective solutions as the defense mechanism would take advantage of the information from multiple sub-levels in “System Defense”. The information gathered and supplied by the Level “Application Defense” would be important in taking proactive decisions at level “System Defense”.

3. System Defense/System Defense + External Defense : Both of these solutions would work at the system level to defend the DDoS attack. The difference is that the later will use the ISP support. “System Defense” alone would be effective, however, identifying the “True positives” and “False Negatives” is the most important concern here. As without actual verification of attack traffic from “Application Defense’ level, this defense would lack effectiveness.

4. Application Defense + System Defense + External Defense : This is a complete design with multi-level support and information or alert flow. After solving the data security and business logic theft issues among levels, it would be an ideal design solution for an Infrastructure cloud.[6]

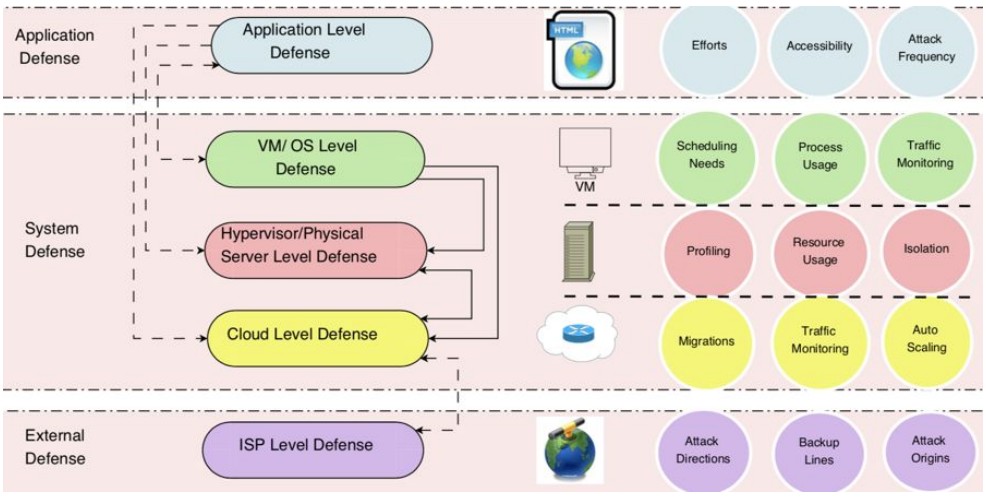

Diagram 6.1 Multi-level solution hierarchy[6]

6.4 DDoS Attacks in Future

Based on traditional volumetric DDoS attacks, we now see a trend where the attacks are becoming sophisticated and are variable rate based. This allows the attackers to remain undetected.

Now, we can provide an extreme example of DDoS attack for which detection is impossible by the methods available today, i.e.,a detection near impossible (DeNy) DDoS attack. The phrase "near impossible" has been specifically used to imply a hope for detection and direction for the whole cloud security community.[7]

Here is the algorithm theoretically is nearly undetected meanwhile can attack infrastructure successfully. It is named DeNy DDoS which is named by its detected near impossible feature.

Algorithm 1: DeNy DDoS
Data: N Attack Sources S_u, $u = 1 \text{ to } N$
Benign request distribution/pattern B_u for $u = 1 \text{ to } N$.
Result: Successful attack for duration T
while Attack!=Successful && Attack Duration<= T **do**
Learn B_u for N users;
Prepare N Attack sources;
for all the u from 1 to N **do**
S_u follows B_u;
end
end

Diagram 6.2 DeNy DDoS Algorithm[7]

It has two important properties. One of them is benignness which means the resultant traffic has no anomalies as per the attack detection rules. The other one is false alerts, which means any detection method that is traffic based would always give false positives even if the method was able to detect any patterns.

This DeNy attacks are mostly benign traffic from a very large number of cloud-driven computationally capable sources. They may not be completely similar to today's stealthy attacks, but they are benign requests that if detected by current methods would generate a huge number of false positives/alerts. We anticipate that the future solutions in mitigating

DDoS attacks in the cloud or in general (with respect to DeNy DDoS attacks) will require a thorough re-appraisal and shift in combating DDoS attacks efficiently.

6.5 Conclusion

DDoS attacks have important characteristics which play an important role while considering utility computing models. DDoS attacks in the cloud are different from the behavior of attacks on fixed on-premise infrastructure. We provide a detailed introduction to the attack methods, consequences, and attack dynamics. It is an attempt to analyze and gather the important requirements in designing DDoS mitigation solutions for cloud infrastructure. These requirements include optimization of five important factors governing the attack. These factors are sustainability/budget constraints, controlled autoscaling, minimization-based optimization of attack traffic,MTT(Mitigation Throughput Time), and service quality and availability. Thus, Collaboration-based multi-level solutions specifically designed for cloud and its features would surely perform better as compared to traditional DDoS solutions.

References

[1] DDoS Attacks in Cloud Computing: Issues, Taxonomy, and Future Directions: Gaurav Somani(a,b), Manoj Singh Gaur (b), Dheeraj Sanghi (c), Mauro Conti (d), Rajkumar Buyya (e) (a) Central University of Rajasthan, Ajmer, India (b) Malaviya National Institute of Technology, Jaipur, India © Indian Institute of Technology, Kanpur, India (d) University of Padua, Padua, Italy (e)The University of Melbourne, Melbourne, Australia

[2] https://kb.mazebolt.com/knowledgebase/dynamic-http-flood/

[3] Cloud-based DDoS Attacks and Defenses: Marwan Darwish, Abdelkader Ouda, Luiz Fernando Capretz Department of Electrical and Computer Engineering University of Western Ontario London, Canada

[4] https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/

[5] https://www.netscout.com/what-is-ddos/application-layer-attacks

[6] DDoS Attacks in Cloud Computing: Issues, Taxonomy, and Future Directions, Gaurav Somani, Manoj Singh Gaur, Dheeraj Sanghi, Mauro Conti, Rajkumar Buyya

[7] Combating DDoS Attacks in the Cloud: Requirements, Trends, and Future Directions, Gaurav Somani,Manoj Singh Gaur, Dheeraj Sanghi, Mauro Conti,Muttukrishnan Rajarajan,Rajkumar Buyya

[8] https://github.blog/2018-03-01-ddos-incident-report/ - February 28th DDos Incedent report

[9]https://techcrunch.com/2018/03/02/the-worlds-largest-ddos-attack-took-github-offline-for-less-than-tens-minutes/ - The world's largest DDos attack took github down for fewer than 10 minutes

[10]https://www.netresec.com/?page=Blog&month=2015-03&post=China%27s-Man-on-the-Side-Attack-on-GitHub

[11]A Cisco Guide to Defending Against Distributed Denial of Service Attacks
https://tools.cisco.com/security/center/resources/guide_ddos_defense

[12]Cloudflare Advanced DDoS Protection
https://www.cloudflare.com/media/pdf/cloudflare-whitepaper-ddos.pdf

[13]DDoS Attacks in Cloud and Mitigation Techniques

http://ijiset.com/vol2/v2s7/IJISET_V2_I6_77.pdf

[14]Cloudflare- TCP Syn Flood Attacks

https://www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/

[15]Cloudflare UDP Flood Attacks

https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/