

**Department of Computer Engineering**

**Spring 2018**

**CMPE-208: Net Architecture and Protocols**

**Hypertext Transfer Protocol**

**Group Lab 3 Report**

**By**

**Team 4**

**Virat Mathur**

**Shivangi Gupta**

**Tejas Chumbalkar**

**Vishal Govindraddi Yarabandi**

Table of Contents

[Objective 3](#_2et92p0)

[Brief Overview Internet Control Message Protocol (ICMP) 3](#_tyjcwt)

[ICMP Message Format 4](#_3dy6vkm)

[Lab Setup](#_1t3h5sf) 5

[ICMP Messages](#_4d34og8) 7

ICMP Tunneling 15

[ICMP Attacks 1](#_17dp8vu)6

ICMP Spoofing...................................................................................................................................... 16

Conclusion……………………………………………………………………………………………………………………………………… 17

[Contribution 1](#_3znysh7)7

References............................................................................................................................................17

## **Objective**

# **Brief Overview of Hypertext Transfer Protocol (HTTP)**

HTTP (Hyper Text Transfer Protocol) is protocol allows to fetch resources like HTML/ Web pages. It provides the foundation for any type of data exchange on Web and a client-server protocol which means request are initiated by recipient usually the web browser. HTTP is an application layer protocol, usually used for communication between web browser and web server but can be used for other purposes as well.

HTTP generally follows the client-server model where clients sends the request to the server and waits until it receives the response from the server. It’s a stateless protocol which means server doesn’t keep any data between two requests. It is generally based on TCP/IP layer and work with any reliable transport layer so that it doesn’t lose data packets silently unlike UDP.

HTTP uses default port TCP 80 and provide standardize way for communication for multiple systems.

# 

# **Basic Characteristics of HTTP**

# · **HTTP is media independent** – HTTP is independent of the type of media sent along the client- server model as long as client and server can handle it properly.

# · **HTTP is connectionless –** HTTP is connectionless as client sends the request and disconnects from the server after sending the request and waits for the response.

# · **HTTP is stateless –** Since, HTTP is stateless none of the client and server can retain any information about the any request other than the current request because both of them forget each other after the current request.

# 

# **HTTP Architecture**

# Basic architecture of HTTP relies on Server and Client.

# · Server – HTTP server respond to the client include message’s protocol version and a success or error code, MIME-like messages containing server information, entity meta information and entity body content.

# · Client – Client communicate with server in the by sending request method to send request to the server, URI and protocol version followed by MIME-like message containing request modifiers, client information and the rest body data content. It uses TCP connection.

# 

# 

# **HTTP Requests**

# HTTP request is sent by client to the server with all type of information required by the server.

# Information included in HTTP requests are-

# · Request line

# · Source IP address, port and proxy

# · Destination IP address, port, protocol and host

# · Requested Uniform Resource Locator (URI)

# · User- agent header

# · Request method and content

# · Cache and connection control headers

# 

# **HTTP Methods**

# There are various HTTP request methods used to indicate the methods to be performed on the requested URI. These are case-sensitive and should be in Upper Case.

# **GET** - Used to retrieve data using requested URI from the server.

# **POST** – It works opposite of GET method i.e. send information to the server example using HTML form.

# **HEAD** – It is similar to GET method. It transfers only status line and header section.

# **PUT** – Uploading content replaces the current representation of the target source.

# **DELETE** – It deletes all the current representation of the target.

# **CONNECT** – It establishes a tunnel to the server identified by the URI given.

# **OPTIONS** – It represents all the communication options provided for the target.

# **TRACE** – This method traces the path by performing loop back test along the path.

# 

# **HTTP Versions**

# HTTP Protocol has evolved over the years a lot varying with number and area of the server-client communications. Various versions of HTTP evolved overtime.

# Since 2016, default for HTTP servers are HTTP/1.1. If we have any libcurl with HTTP/2 built-in abilities then curl with attempt to use HTTP/2 or fall down to HTTP/1.1 if the negotiation fails. Non-HTTP/2 capable curls get 1.1 over HTTPS by default.

# 

# **Message Response**

# Response is sent by server after receiving and interpreting a request message. The response massages consist of the following:

# · Status line

# · Zero or more header

# · An empty line which indicates the end of the header

# · Optionally a message-body

# 

# **Status Code**

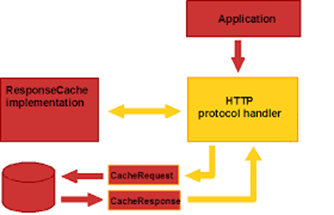
|  |  |
| --- | --- |
| **S.N.** | **Code and Description** |
| 1 | **1xx: Informational** - Request received and the process is continue. |
| 2 | **2xx: Success** - Action received successfully, understood, and accepted. |
| 3 | **3xx: Redirection** - Action must be taken in order to complete the request. |
| 4 | **4xx: Client Error** - Request contains incorrect syntax or cannot be fulfilled. |
| 5 | **5xx: Server Error** - Server failed to fulfill an apparently valid request. |

# 

# **HTTP Caching**

# Fetching some data over internet requires many round trips between the client and the server which delays if they are available and when the browser can process them This increases the cost and slows down the network. Hence, to optimize the performance, caching is one of the critical aspects tp fetch resources.

# Every browser is with implementation of HTTP cache already. We only need to ensure that each server provides the correct HTTP header directives to instruct the browser on for how long and when can the browser cache the response.



# 

# 

# **Lab Setup**

We observed the behavior of HTTP protocol on a single linux machine (Ubuntu 16.04). Study of various HTTP packets is done using following tools:

* Web browser
* Telnet
* Wireshark tool

# **Lab Execution and Observations**

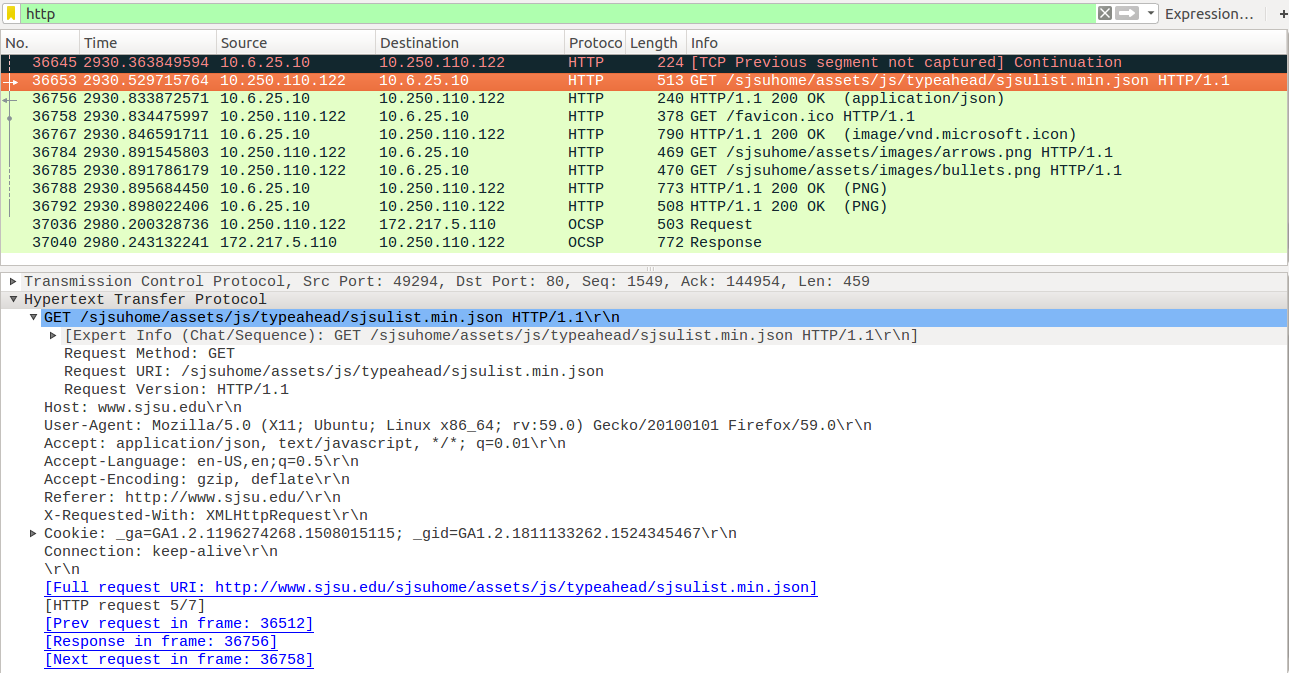
**HTTP Methods**

**HTTP GET**

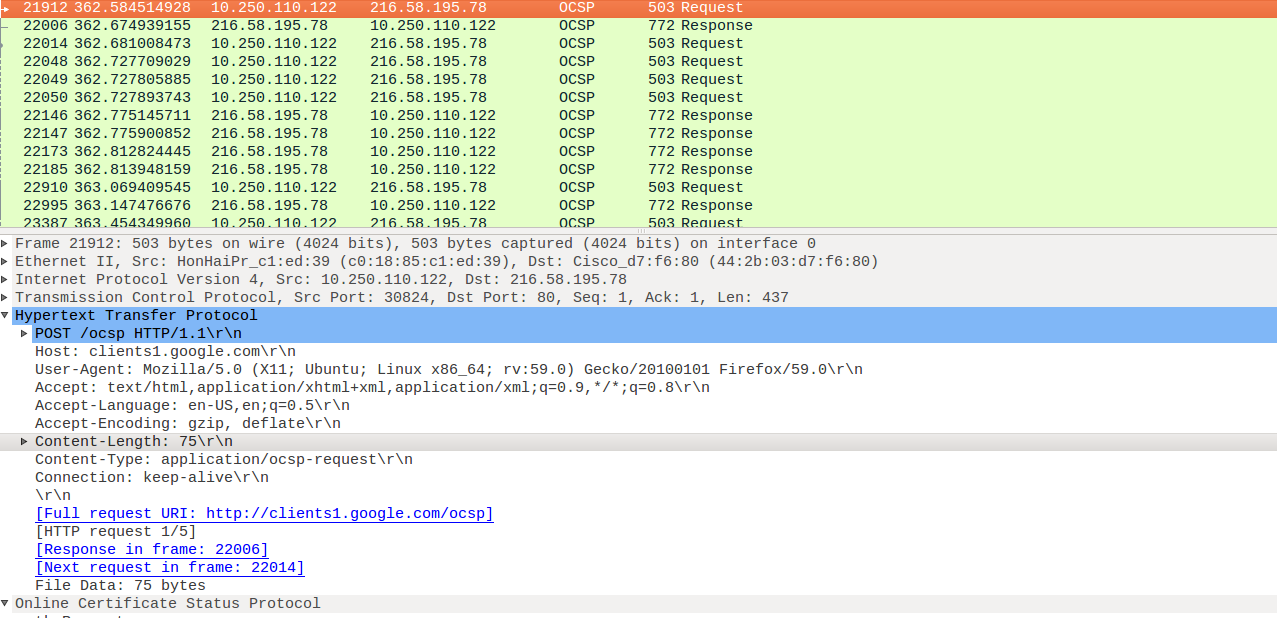
HTTP GET method is used to retrieve data from the target server. A bookmark of HTTP GET request can be done. It is not advisable to use HTTP GET while dealing with sensitive data. GET

requests have length restrictions.

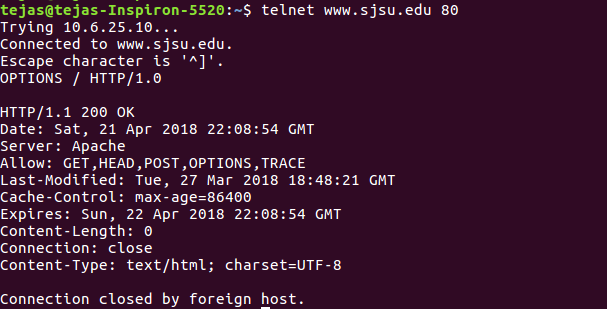
Following is the HTTP GET packet captured in wireshark while visiting [www.sjsu.edu](http://www.sjsu.edu) in Mozilla Firefox browser.



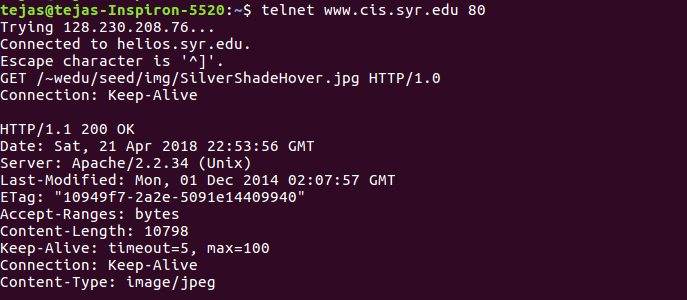
**HTTP POST**



**HTTP OPTIONS**

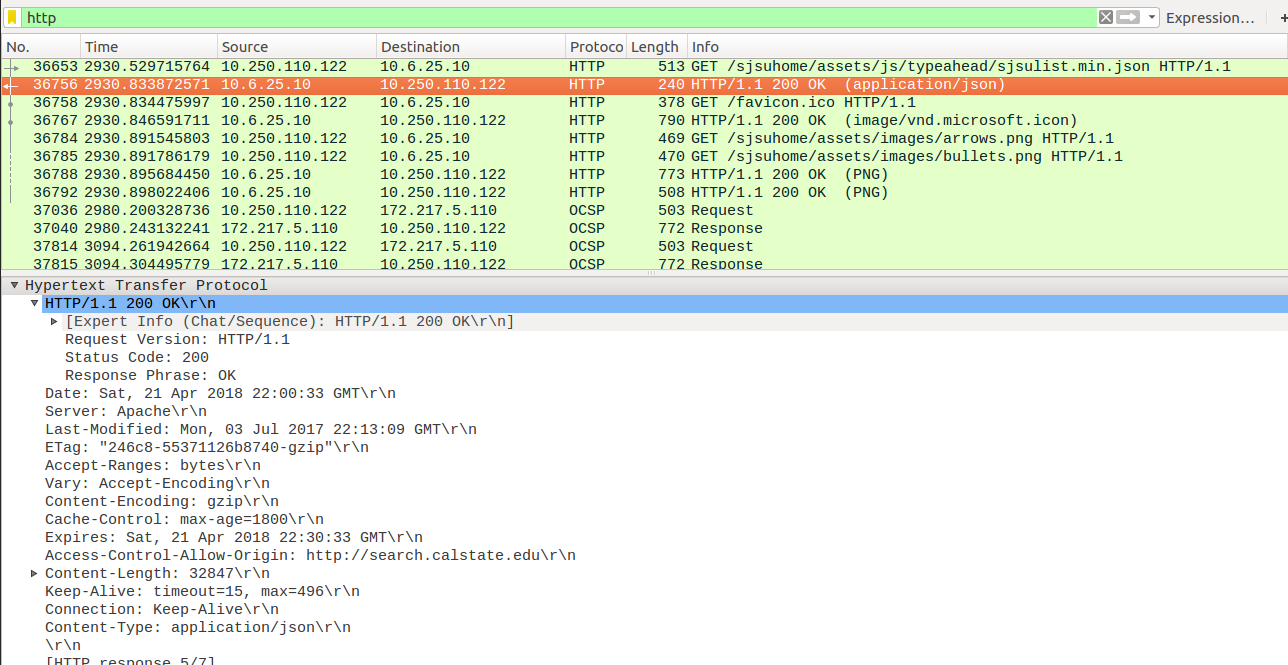
****

# **HTTP Keep Alive**

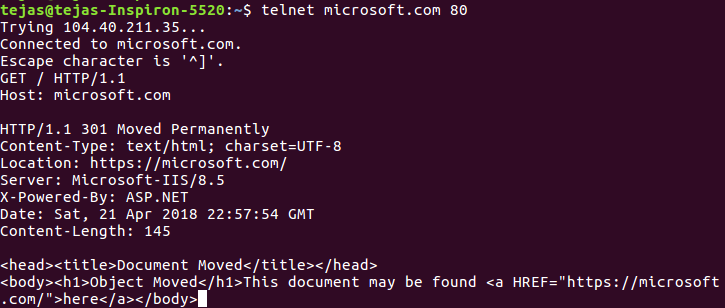


# **HTTP Response codes**

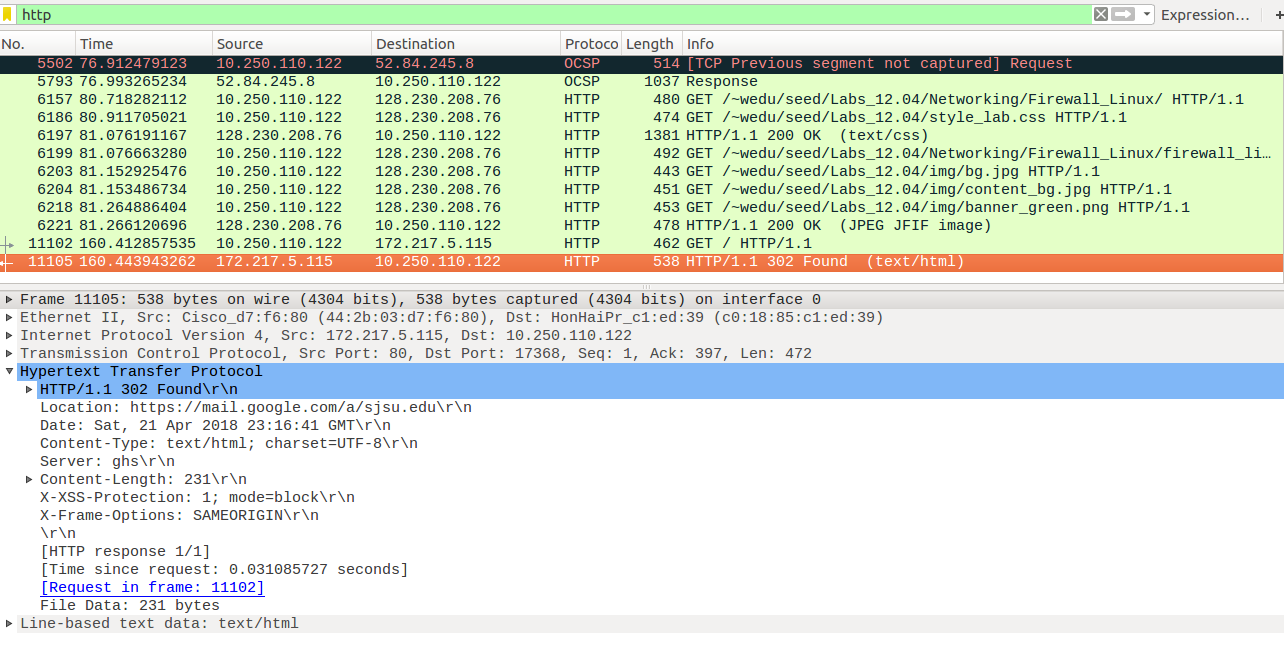
**HTTP 200 OK**

****

**HTTP 301 moved**

****

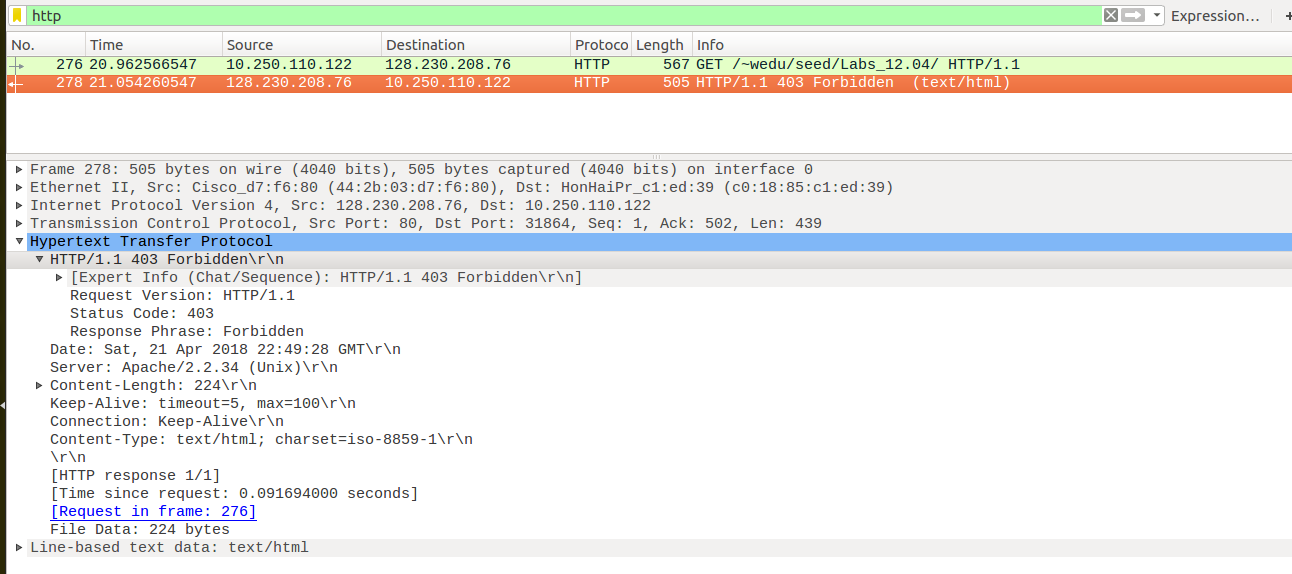
**HTTP 302 Found Request**

****

**HTTP 400 Bad request**

# **208/drive-download-20180422T040918Z-001/400%20bad%20request.png**

**HTTP 403 Forbidden**



**HTTP 404 Not Found**



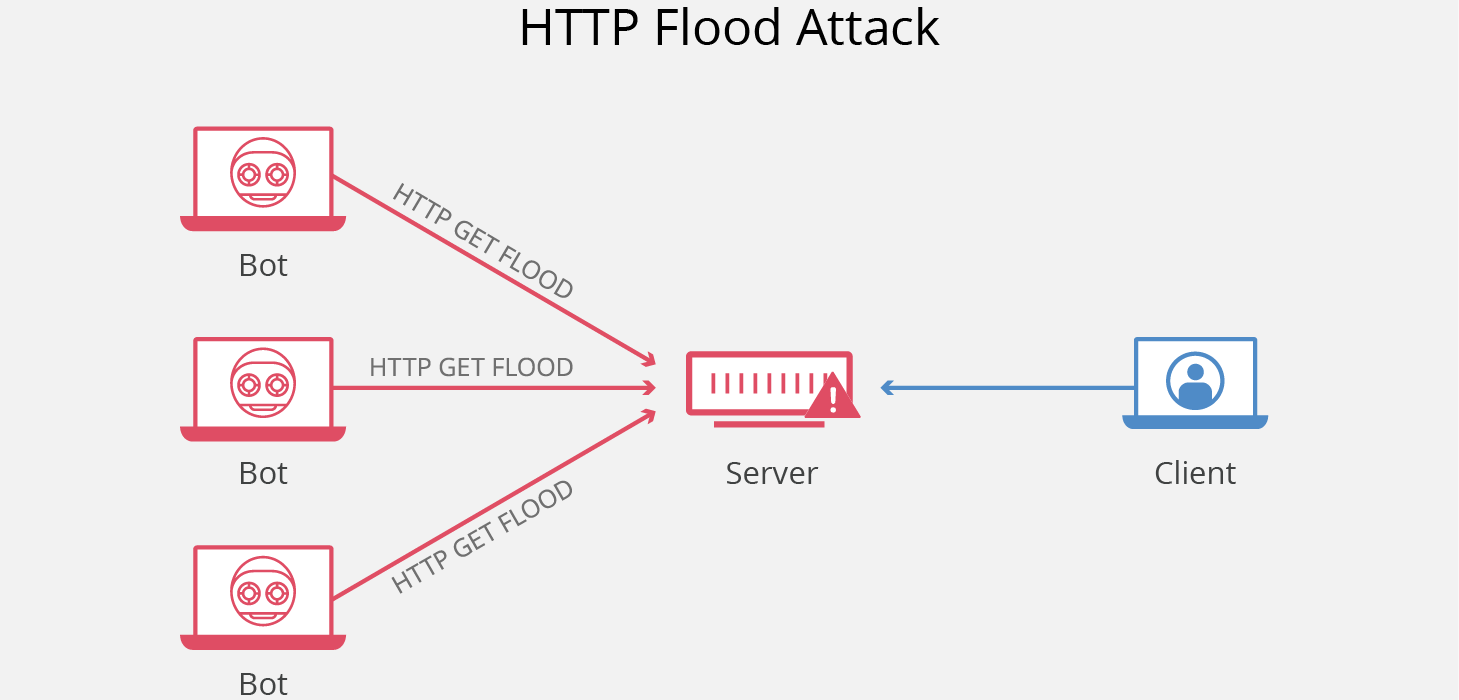
# **HTTP Attacks**

As http runs over TCP, it can get attack by the TCP related attacks. Therefore, when securing for http attack it is important to keep a broader security perspective in mind other than just HTTP protocol.

SYN FLOOD ATTACK:

In this attack the attacker sends many SYN packets to the server. Since, every SYN opens a TCP stack and it needs to preserve memory to keep the session state intact. Therefore, sending a huge amount of SYN Packets will exhaust the memory.

The latest version of it is Tsunami SYN flood attack in which large packets are send with TCP SYN to block the internet pipe.



*Reference: https://www.cloudflare.com/le 1*

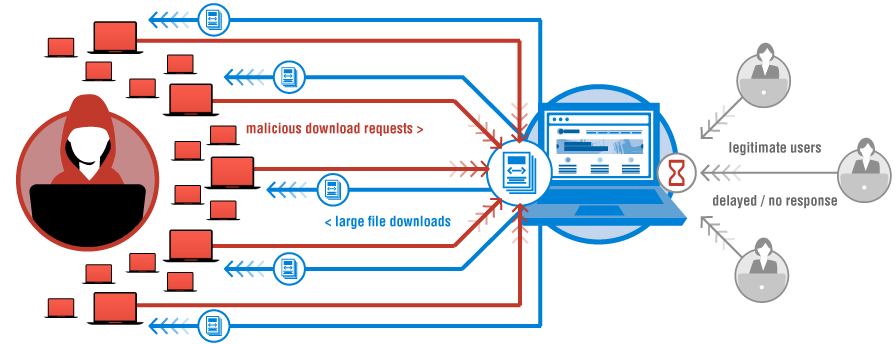
HTTP L7 attack and network attack differ in the way that HTTP requires valid IP, as the attacker can spoof the IP address but today with the use of botnet it is not a difficult task to spoof the IP and hence once the connection is allocated many attacks can be implemented:

* GARBAGE FLOOD:

In this attack attacker first open the connection to the HTTP ports and sends garbage data to it, this attack is overlooked in mitigation as the server and the firewalls protecting it usually expects a “valid” HTTP traffic. As here it floods internal buffers and queue in the server and also sometimes it blocks the internet connection.

* GET FLOOD:

GET request is the most used in the HTTP protocol and in this attack GET flood is used by sending it in high quantity. Here the attackers floods the server with high volume of GET request which blocks the server to answer the legitimate request.



*Reference: https://www.verisign.com/en\_U 1*

* REVERSE BANDWIDTH FLOODS:

This attack tries to saturate the internet link from behind : hence it makes server to send packets traffic which saturates up-link of the server. Server sends large packets continuously and saturates the link. This does not get detected by the devices as they do not check the up-link traffic.

* LOW AND SLOW ATTACK:

This attack consumes very less Bandwidth and PPS. It works in less traffic hence hard to detect. The attack sends legitimate traffic to the server but at very low pace with large time gap within packets of GET request. It sends like ‘G’ then after sometime ‘E’and ‘T’ and others so on. The connection is correct but it consumes a lot of resources of the server. It has to keep the connection open as it waits for complete request to arrive. Hence pool reaches saturation with little traffic.

* HTTP Flood Attack

It is a kind of volume/quantity metric distributed denial of service(DDOS) attack to exhaust the server resource with HTTP request.

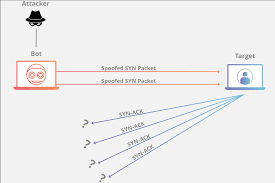
In this Distributed Denial of Service attack the attack disrupts the usual traffic of the server and the attacker floods the infrastructure with internet traffic.

Internet connected machines are infected with malware, and this way many bots are controlled as they give remote access. The attacker directs the instructions to every bot and they all target to a particular IP address by sending it continuous requests which causes the the victims systems resources to overflow and hence resulting in Denial of Service to the legitimate clients.

Various types of Denial of Service attacks:

1. Protocol Attack:

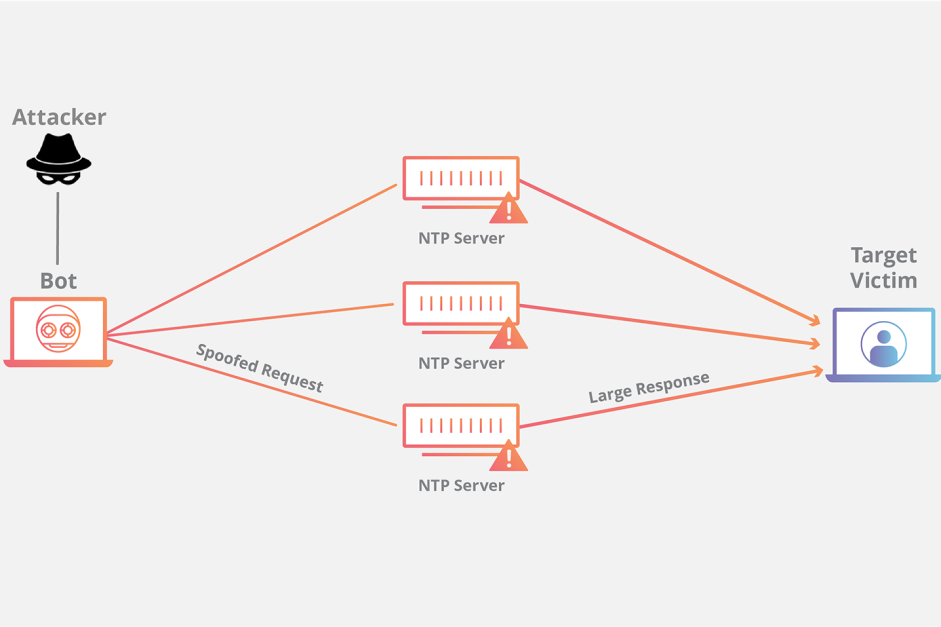
Also known to be as state-exhaustion attacks, and disruption to the actual legitimate client. Protocols uses the layer 3 and layer 4 weaknesses.



*Refrence: https://www.cloudflare.com/lea 1*

1. NTP Amplification attack

It is reflection based Distributed denial of service(DDoS) and it exploits a Network Time Protocol(NTP) server by putting amplified usage of UDP traffic.



*Reference:* [*https://www.cloudflare.com/le*](https://www.cloudflare.com/le) *2*

# 

# **Conclusion**

# **Contribution**

|  |  |
| --- | --- |
| Name | Contribution |
| Tejas Chumbalkar | * Study of HTTP methods * Execution of various commands on Linux CLI |
| Vishal Govindraddi Yarabandi | * Wireshark observation * Documentation and report formatting |
| Shivangi Gupta | * Study of HTTP response codes * Wireshark observations |
| Virat Mathur | * Study of HTTP attacks * Documentation and report formatting |

**References**

<https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/> <https://blog.radware.com/security/2017/11/http-attacks/>

# <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>

# <https://developer.mozilla.org/en-US/docs/Web/HTTP>

# <https://www.tutorialspoint.com/http/http_overview.htm>

# <https://www.javatpoint.com/http-requests>

# <https://www.tutorialspoint.com/http/http_methods.htm>

# <https://ec.haxx.se/http-versions.html>

# <https://www.tutorialspoint.com/http/http_responses.htm>

# <https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching>

# 