

Slow Read DoS Attack and Classification

Enroll. No. (s) - 17103231, 17103248, 17103250

Name of Student (s) - Mayank Srivastva, Harshal Tyagi, Gourav Garg

Name of Supervisor - Dr.Gagandeep Kaur



November – 2019

Submitted in partial fulfillment of the Degree of

Bachelor of Technology

in

Computer Science Engineering

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING
JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

TABLE OF CONTENTS

Chapter No.	Topics
Chapter-1	Introduction <ul style="list-style-type: none">1.1 Problem Statement1.2 Introduction1.3 Real World Application1.4 Dependencies1.5 Work Approach
Chapter-2	Development <ul style="list-style-type: none">2.1 Explanation of algorithms2.2 Maths Used2.3 Code Implementation2.4 Diagram
Chapter -3	Testing <ul style="list-style-type: none">3.1 Outputs3.2 Drawbacks
Chapter-4	Result, Conclusion, Future work <ul style="list-style-type: none">4.1 Result4.2 Conclusion4.3 Future Work4.4 References

ACKNOWLEDGEMENT

It is our privilege to express our sincerest regards and thanks to our project supervisor, **Dr. Gagandeep Kaur**, for their valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of our project.

We deeply express our sincere thanks to our Head of Department Dr Prof. Vikas Saxena for encouraging and allowing us to present the project on the topic “**Slow Read DoS Attack And Classification**” at our department premises for the partial fulfillment of the requirements leading to the award of B-Tech degree.

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature:

Mayank Srivastva (17103231)

Harshal Tyagi (17103248)

Gourav Garg (17103250)

CERTIFICATE

This is to certify that the work titled “**Slow Read DOS Attack And Classification**” submitted by "MAYANK SRIVASTAVA , HARSHAL TYAGI , GOURAV GARG” in partial fulfillment for the award of degree of B. Tech of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor

Name of Supervisor

Dr. Gagandeep Kaur

Designation

Assistant Professor (Senior Grade)

Date

Chapter -1

(INTRODUCTION)

1.1 Problem Statement:

Sometimes when working on a system we may feel following problem. The Problem can be slow access to files either locally or remotely which may increase time and decrease efficiency of computer also a long-term inability to access a particular website, Internet disconnection which is the biggest problem at this time which can lead to several big as well as small problems, Problems accessing all websites and Excessive amount of spam emails which may increase the waste size of your computer and decrease the computational power of your computer.

These are the symptoms faced when a network is facing a Denial of Service attack.

1.2 Introduction

DoS (Denial of Service) attacks are evolved and consolidated as severe security threats to network service. Earlier DoS attacks use flood-based high-bandwidth approach and exploit the resource of network and transport protocol layers. Since DoS attacks are simple, they can be prevented by filtering the source IP address. In order to break through this countermeasure, DDoS (Distributed DoS) attacks deliver a DoS attack by many attackers. However, there is a problem using high-bandwidth in DoS/DDoS attacks, and many solutions are proposed. One of such solution is low-bandwidth approach. The latest attack method using such approach is low bit-rate type which exploits vulnerabilities of application layer protocols to accomplish DoS attacks . This attack is that an attacker basically sends a legitimate HTTP request to the Web server and then very slowly reads the response. If an attacker sends many legitimate requests, the Web server quickly reaches its maximum capacity and becomes unavailable for new connections by legitimate clients. Moreover, it is very hard to detect these attacks if we do not monitor the network layer, because those requests are indistinguishable from other legitimate clients .

1.3 Real world application:

Attackers using Slow Read DoS method use HTTP GET requests to encroach the HTTP connections allowed on a server, preventing other users from accessing and given the attackers an opportunity to open up multiple connections on the same server. Slow DoS attacks are trickier to deal with as they don't require many resources or planning. The attacker can launch the attack using just one system and sending 4 to 5 HTTP requests, unlike DoS or Distributed DoS attacks.

Without a doubt, a slow DOS attack is a potential threat for a server and one that has to keep an eye out for. In this age of cyber world, data safety is really important. Therefore to detect the source of attacker, this project can ultimately help in securing the system.

1.4 Dependencies:

- CentOS
- Slowhttptest tool
- Wireshark
- Vmware
- Jupyter Notebook
- Apache Server

1.5 Work Approach and Plan

Denial of service attack (DoS)

A **Denial-of-Service (DoS) attack** is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic, or sending it information that triggers a crash. In both instances, the DoS attack deprives legitimate users (i.e. employees, members, or account holders) of the service or resource they expected.

Victims of DoS attacks often target web servers of high-profile organizations such as banking, commerce, and media companies, or government and trade organizations. Though DoS attacks do not typically result in the theft or loss of significant information or other assets, they can cost the victim a great deal of time and money to handle.

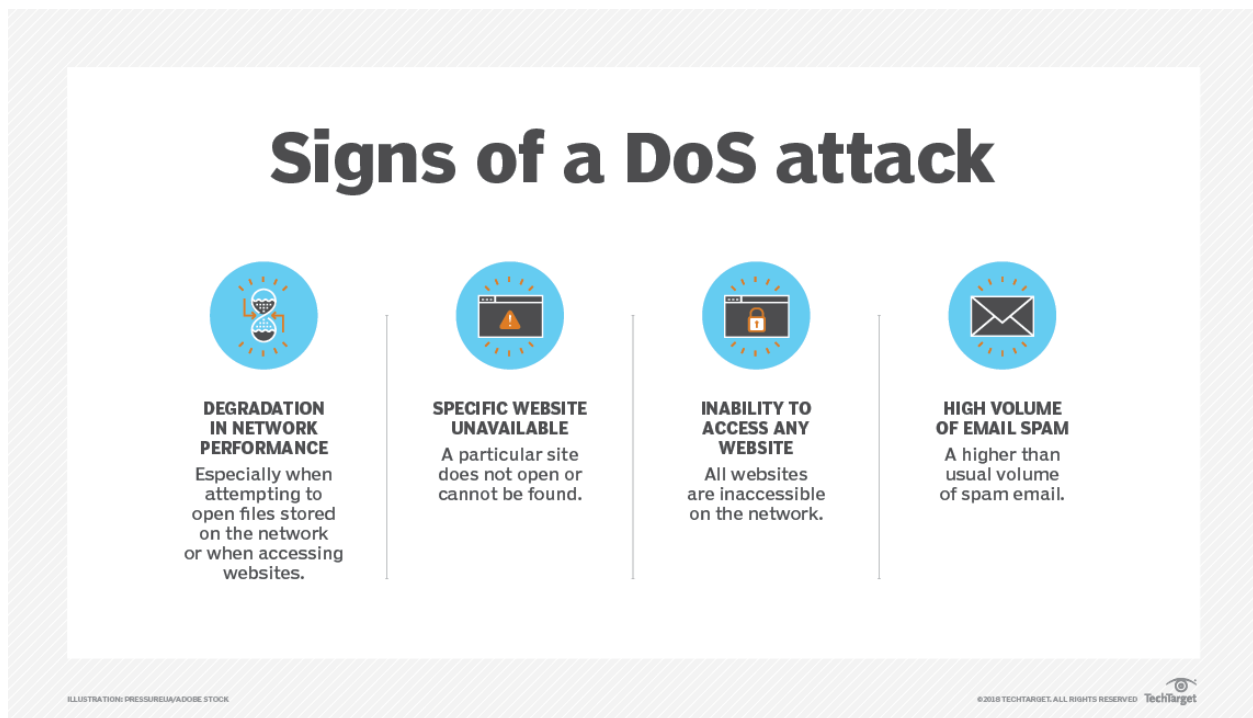


Figure 1 signs of DOS attack

Slow Read DOS attack

The Slow Read DoS Attack is one of Slow HTTP attacks. Slow HTTP attacks do not aim at the network layer like DoS / DDoS attacks, but exploit the application layer. If a HTTP request is not complete, or if the transfer rate is very low, the Web server keeps its resources busy waiting for the rest of the data. Slow HTTP attacks are based on this fact. Thus when the Web server keeps too many resources busy, this situation becomes like DoS attacks. To realize this malicious condition, the attacker can take following two types of techniques.

- 1) The technique of sending request slowly
- 2) The technique of reading response slowly

Type 1 is well-known technique and Slowloris (also known as Slow Headers or Slow HTTP GET) or Slow HTTP POST attacks are famous. The Slow Read DoS Attack is categorized into type 2 and this is the latest technique.

Environment Setup:

Two virtual machines are to be set up over a host machine. We've installed CentOS on both the virtual machines. One machine is used for attacking the server and second to create the Apache server as shown in figure 2.

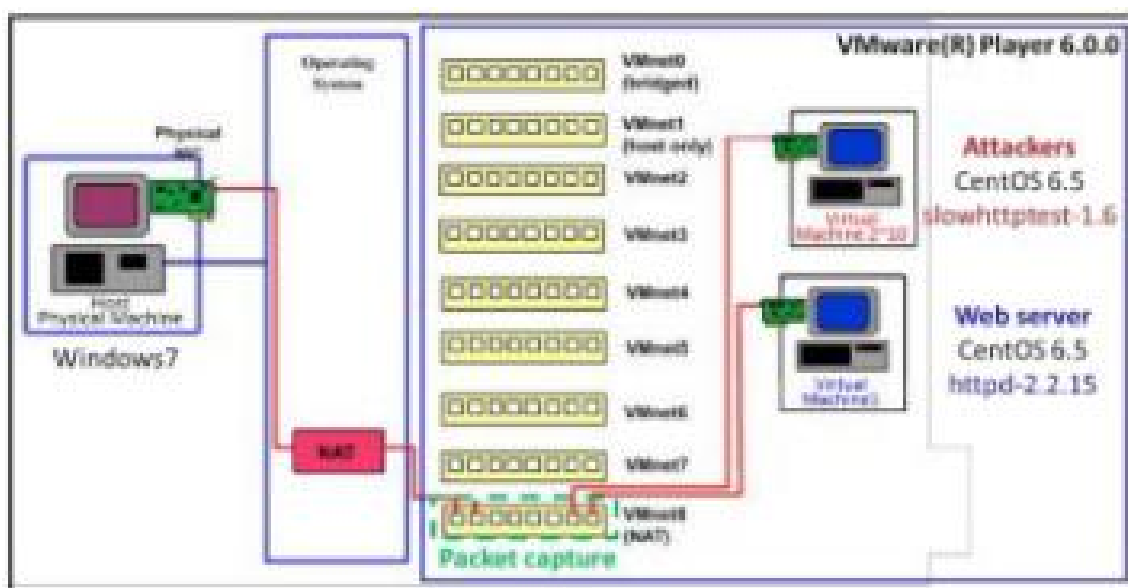


Figure 2 : Experiment Environment

Attack Strategy:

The attacker can deliver the Slow Read DoS attack by exploiting the flow control of TCP. First, the attacker sends a legitimate request after 3-way-handshake. After that, the attacker advertises the window size smaller than usual to make the HTTP response operation slow down. If the attacker advertises window size with zero, the Web server will stop sending data with holding the connection. As a result, the attacker succeeds in Web server making resource waste.

Chapter-2

(Development)

2.1 SOFTWARE TOOLS AND ALGORITHM USED:

Slowhttpptest tool:

SlowHTTPtest is a highly configurable tool that simulates some Application Layer Denial of Service attacks. It works on majority of Linux platforms, OSX and Cygwin – a Unix-like environment and command-line interface for Microsoft Windows.

It implements most common low-bandwidth Application Layer DoS attacks, such as slowloris, Slow HTTP POST, Slow Read attack (based on TCP persist timer exploit) by draining concurrent connections pool, as well as Apache Range Header attack by causing very significant memory and CPU usage on the server.

VMware Workstation:

VMware Workstation is a hosted hypervisor that runs on x64 versions of Windows and Linux operating systems (an x86 version of earlier releases was available); it enables users to set up virtual machines (VMs) on a single physical machine, and use them simultaneously along with the actual machine.

Wireshark:

Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education.

Jupyter Notebook:

The **Jupyter Notebook** is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Random Forest Classification:

We have used this classification technique to distinguish between attacking and normal traffic. Random Forest Classification consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

2.2 CONDITION FOR ATTACK AND MATHS USED:

When conducting the Slow Read DOS Attack experiment, we define the status of attack as follows.

- Attack success: Unacceptable new legitimate connections.
- Attack Failure: Acceptable new legitimate connections.

“Attack success” means that the number of generated child processes is larger than or equal to the value of MC/SL (maximum client/server limit). On the other hand, “Attack failure” means that the number of generated child processes is less than the value of MC/SL (maximum client/server limit). The effectiveness of successful attack is estimated by time length (second) of maintaining service unavailable state.

Variables	Explanation
t0	The Value of Timeout
tz	Finish time of sending attack connections (tz=N / C)
N	The total number of attack connections
C	The number of attack connections which send per second
K	The number of disconnected connections per second after t0
M	The total number of connections that Web server can process (MC/SL)
A(t)	The total number of attack connections which the attacker sent at time

Table 1. Variables of calculation

Conditions for Attack Success Status

We can deduce the conditions of successful attack. Table 1 shows the variables. Let A be the total number of attack connections which connected to Web server. Then the condition of $M \leq A$ is necessary condition for successful attack. There are two cases of calculations for A under the conditions of t0 and tz. In the case of $t_0 \geq t_z$,

$$A(t) = C \times t \quad (1)$$

In the case of $t_0 < t_z$

$$A(t) = \begin{cases} C \times t & (t < t_0) \\ C \times t - K \times (t - t_0) & (t \geq t_0) \end{cases}$$

where $(t \geq t_0)$ denotes time progress after attack starts ($t=0$).

2.3 Implementations Details:

Step-1: Attack Environment Setup:

For performing the attacks we install CentOS 7 .After that we install GNOME GUI as two virtual machines in VMware.

Installing GNOME GUI in CentOS 7 :

```
# yum group install "GNOME Desktop" -y
```

After that in one of the virtual machine we install slowhttptest to perform the slow read dos attack. For installation of this we perform following steps:

Installing epel-release rpm:

```
# rpm -Uvh epel-release*rpm
```

Installing slowhttptest rpm package:

```
#yum install slowhttptest
```

After that we install apache web server in other virtual machine as a target on which we perform the attack using slowhttptest tool.

Installing Apache Server on other system:

```
#sudo yum install httpd
```

Figure 3. Showing the successful installing the Apache web server by checking it through entering the IP address of the server in search engine. Once we successfully setup the attack environment we are ready to perform slow read DoS Attack .After that we capture it using wire shark which we install on that virtual machine on which we make our server.



Figure 3. Apache web server

Figure 1 shows experiment environment. We use Wire shark to observe packet between the attacker and Web server. Table 2 shows the de fault directives of “httpd.conf” which controls a connection with a client in the configuration of the Web server which is the attack target. Table 3 shows the default configuration of “prefork MPM (Multi Processing Module)” which controls the generation of child processes when a connection is established.

Directive	Value
Timeout	60
Keepalive	Off

Table 2. Directives of httpd.conf

Directive	Value
StartServers	8
MinspareServers	5
MaxspareServers	20
SeverLimits	256
Maxclients	256
MaxRequestsPerchild	4000

Table 3. Prefork MPM

Step 2: Performing Attacks:

For performing the attack we fix the parameter of attacker as shown in Table 4
For the same the command is:

```
#slowhttptest -X 500 -H -g -i 10 -r 200 -t GET - u
http://192.168.222.141 -x 24 -p 10
```

Table 4. Attack options

Option	Value
Number of connections	500
Receive Window Range	8-16 byte
Pipeline factor	1
Read rate from receive buffer	5byte/sec
Connection rate	50 connections/sec
Timeout for probe connection	10 sec
Using proxy	no proxy

Step 3: Classification of Attack traffic and Normal traffic using Random Forest Classification:

For classification of attack traffic and normal traffic we use random forest classification. Here is the code for the same:

```
import pandas as pd
import numpy as np
from sklearn import metrics

df= pd.read_csv('C:/Users/Mayank Srivastava/Desktop/Minor/cleandata3.csv')

df['Source_id'] = df.groupby(['Source']).ngroup()
df['Destination_id'] = df.groupby(['Destination']).ngroup()
df['Destination_Port'] = df.groupby(['DestinationPort']).ngroup()
df['Source_Port'] = df.groupby(['SourcePort']).ngroup()

df[df['SourcePort'] < 0] = 0
df[df['DestinationPort'] < 0] = 0
df['Source_Port'].fillna(0)

df.to_csv('C:/Users/Mayank Srivastava/Desktop/Minor/dataset.csv')

features= ['Source_id', 'Destination_id','Source_Port','Destination_Port']
X= df[features]
Y=df.Result

#Splitting the dataset
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
# Fitting Random Forest Classification to the Training set
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
classifier = RandomForestClassifier(n_estimators = 15 , max_depth= 1, criterion = 'gini', random_state = 0)
```

```
classifier.fit(X_train, Y_train)
```

```
y_pred = classifier.predict(X_test)
```

```
print(y_pred)
```

```
print("Accuracy:",metrics.accuracy_score(Y_test, y_pred))
```

Chapter- 3

(Testing)

3.1 Outputs:

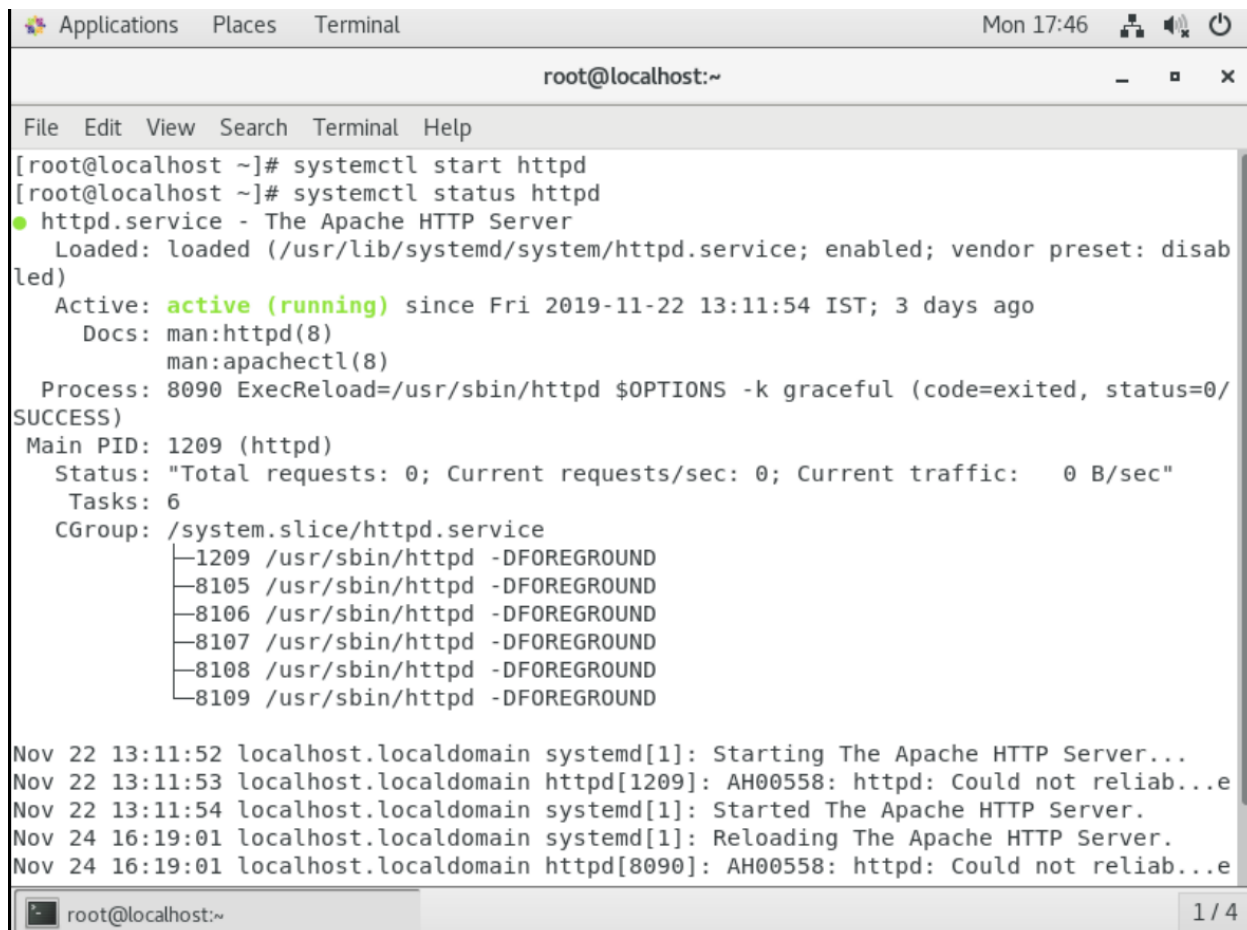
After completion the environment setup we first run the server by entering the command

```
#systemctl start httpd
```

```
#systemctl status httpd
```

We can also check the server status as shown in figure 4.

After that we perform the attack for probe connection timeout is 3 sec and connections is 1000 and connections per second is 200 and test duration is 240 seconds which is perform using slowhttptest as shown in figure 5.



```
Applications  Places  Terminal  Mon 17:46
root@localhost:~

File Edit View Search Terminal Help

[root@localhost ~]# systemctl start httpd
[root@localhost ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2019-11-22 13:11:54 IST; 3 days ago
     Docs: man:httpd(8)
           man:apachectl(8)
  Process: 8090 ExecReload=/usr/sbin/httpd $OPTIONS -k graceful (code=exited, status=0/SUCCESS)
 Main PID: 1209 (httpd)
    Status: "Total requests: 0; Current requests/sec: 0; Current traffic:  0 B/sec"
     Tasks: 6
    CGroup: /system.slice/httpd.service
            └─1209 /usr/sbin/httpd -DFOREGROUND
              └─8105 /usr/sbin/httpd -DFOREGROUND
                └─8106 /usr/sbin/httpd -DFOREGROUND
                  └─8107 /usr/sbin/httpd -DFOREGROUND
                    └─8108 /usr/sbin/httpd -DFOREGROUND
                      └─8109 /usr/sbin/httpd -DFOREGROUND

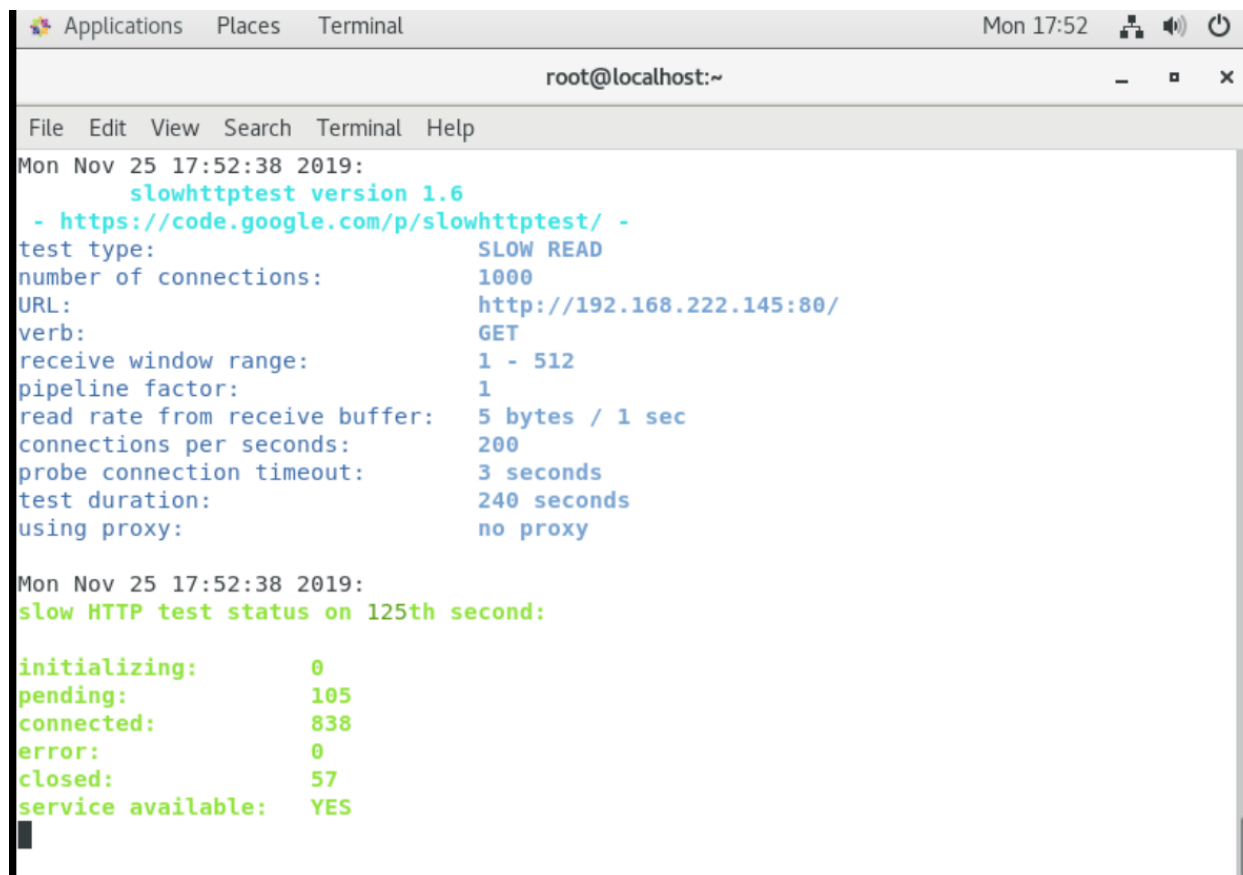
Nov 22 13:11:52 localhost.localdomain systemd[1]: Starting The Apache HTTP Server...
Nov 22 13:11:53 localhost.localdomain httpd[1209]: AH00558: httpd: Could not reliab...e
Nov 22 13:11:54 localhost.localdomain systemd[1]: Started The Apache HTTP Server.
Nov 24 16:19:01 localhost.localdomain systemd[1]: Reloading The Apache HTTP Server.
Nov 24 16:19:01 localhost.localdomain httpd[8090]: AH00558: httpd: Could not reliab...e

root@localhost:~ 1 / 4
```

Figure 4.server status

Command used for attack:

```
# slowhttptest -X 1000 -H -g -i 10 -r 200 -t GET - u  
http://192.168.222.141 -x 24 -p 3
```



```
Mon Nov 25 17:52:38 2019:  
    slowhttptest version 1.6  
- https://code.google.com/p/slowhttptest/ -  
test type:                SLOW READ  
number of connections:    1000  
URL:                     http://192.168.222.145:80/  
verb:                    GET  
receive window range:    1 - 512  
pipeline factor:         1  
read rate from receive buffer: 5 bytes / 1 sec  
connections per seconds: 200  
probe connection timeout: 3 seconds  
test duration:           240 seconds  
using proxy:             no proxy  
  
Mon Nov 25 17:52:38 2019:  
slow HTTP test status on 125th second:  
  
initializing:            0  
pending:                 105  
connected:               838  
error:                   0  
closed:                  57  
service available:       YES
```

Figure 5. Attack options

After that we capture the packets of normal traffic and attack traffic using wire shark into that virtual machine in which server is made as shown in figure 6.

After capturing the packet we can check that the attack is happened or not and Successful installation of slowhttptest as well by using TCP stream feature which is shown in figure 7.

We have saved the .pcap file and created a dataset of more than 96,000 rows and saved .csv file. Before classification we applied some techniques to make a clean dataset so that we can use random forest.

After that we classify the normal and attack traffic using random forest classification with the help of machine learning the outputs for the complete

Classification process in jupyter notebook by taking features source IP, destination IP ,source port and destination port as shown in figure 8 , figure 9 , figure 10 .

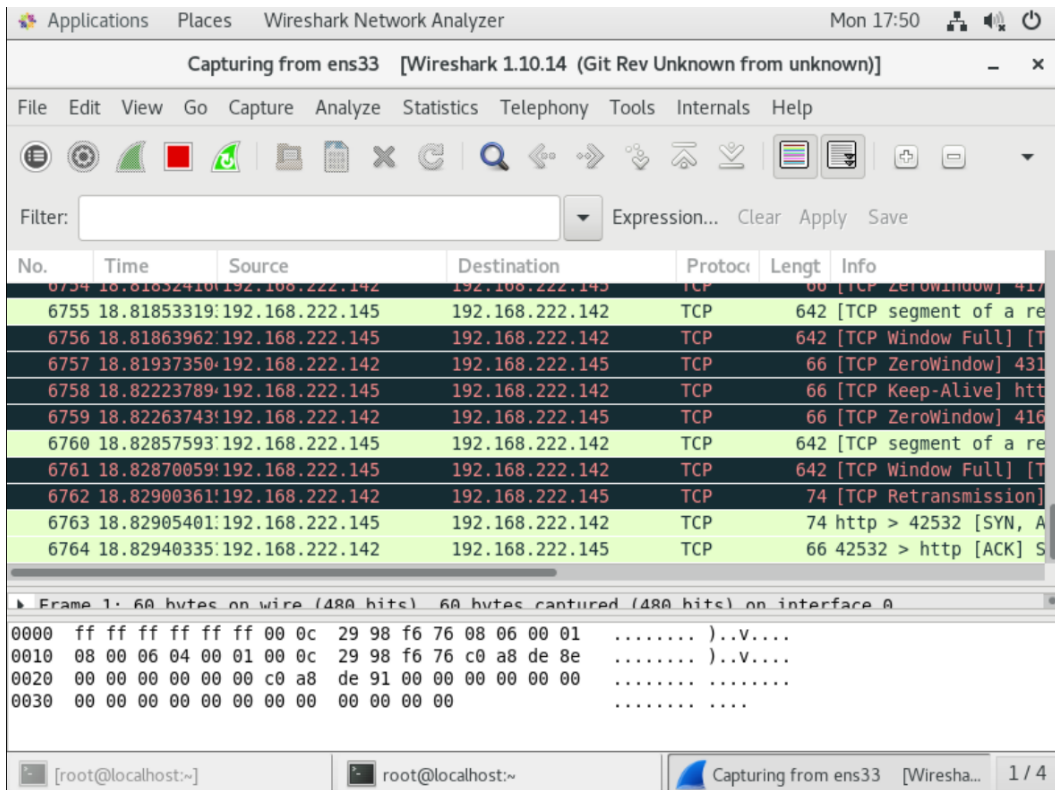


Figure 6.capturing of packets

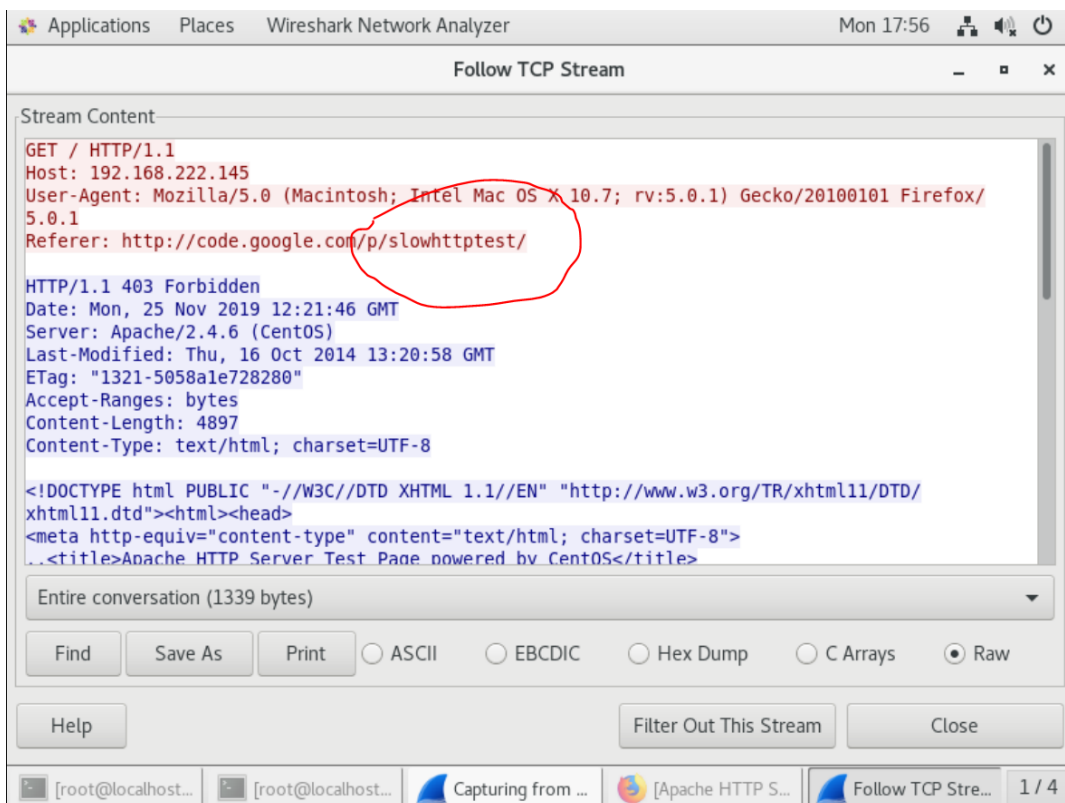


Figure7.(TCP stream)

```
In [190]: import pandas as pd
import numpy as np
```

```
In [191]: df= pd.read_csv('C:/Users/Mayank Srivastava/Desktop/Minor/cleandata3.csv')
```

```
In [192]: df.head()
```

Out[192]:

	Unnamed: 0	No.	Time	Source	Destination	Protocol	Length	Info	Result	Src_Port	Deest_Port	SourcePort	DestinationPort
0	0	1	0.000000	192.168.222.141	youtube- ui.l.google.com	TLSv1.2	708	Application Data	0	46978	https	46978.0	443.0
1	1	2	0.000259	youtube- ui.l.google.com	192.168.222.141	TCP	60	https > 46978 [ACK] Seq=1 Ack=655 Win=64240 Len=0	0	https	46978	443.0	46978.0
2	2	3	0.330808	youtube- ui.l.google.com	192.168.222.141	TLSv1.2	337	Application Data, Application Data, Applicatio...	0	https	46978	443.0	46978.0
3	3	4	0.340927	192.168.222.141	youtube- ui.l.google.com	TLSv1.2	93	Application Data	0	46978	https	46978.0	443.0
4	4	5	0.341198	youtube- ui.l.google.com	192.168.222.141	TCP	60	https > 46978 [ACK] Seq=284 Ack=694 Win=64240 ...	0	https	46978	443.0	46978.0

```
In [193]: df['Source_id'] = df.groupby(['Source']).ngroup()
df['Destination_id'] = df.groupby(['Destination']).ngroup()
df['Destination_Port'] = df.groupby(['DestinationPort']).ngroup()
df['Source_Port'] = df.groupby(['SourcePort']).ngroup()
```

```
In [194]: df.head()
```

Out[194]:

	Destination	Protocol	Length	Info	Result	Src_Port	Deest_Port	SourcePort	DestinationPort	Source_id	Destination_id	Destination_Port	Source_Po
	youtube- ui.l.google.com	TLSv1.2	708	Application Data	0	46978	https	46978.0	443.0	1	48	6	244
192.168.222.141		TCP	60	https > 46978 [ACK] Seq=1 Ack=655 Win=64240 Len=0	0	https	46978	443.0	46978.0	31	4	2439	
192.168.222.141		TLSv1.2	337	Application Data, Application Data	0	https	46978	443.0	46978.0	31	4	2439	

Figure 8: Classification part 1/3

Here, IP addresses are in string datatype so we have pre processed the data by giving a unique id to each IP address and Port number.

```
In [195]: df[df['SourcePort'] < 0] = 0
df[df['DestinationPort'] < 0] = 0
df['Source_Port'].fillna(0)
```

```
Out[195]: 0      2440
1         5
2         5
3      2440
4         5
...
96446     3
96447    3258
96448    3258
96449     3
96450    3258
Name: Source_Port, Length: 96451, dtype: int64
```

```
In [196]: df.to_csv('C:/Users/Mayank Srivastava/Desktop/Minor/dataset.csv')
```

```
In [197]: features= ['Source_id', 'Destination_id', 'Source_Port', 'Destination_Port']
X= df[features]
```

```
In [198]: Y=df.Result
```

```
In [199]: X.head()
```

```
Out[199]:
```

	Source_id	Destination_id	Source_Port	Destination_Port
0	1	48	2440	6
1	31	4	5	2439
2	31	4	5	2439
3	1	48	2440	6
4	31	4	5	2439

```
In [ ]:
```

```
In [200]: #Splitting the dataset
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)
```

```
In [201]: # Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Figure 9: Classification part 2/3

We faced that there were some null values in the source port, so we processed those values to 0 and then moved forward towards splitting and scaling the dataset.

```

In [202]: X_train
Out[202]: array([[ 1.0970968 , -0.32569226, -0.7737275 ,  2.23989213],
 [-0.40639746, -0.2142472 , -0.77556195,  0.47226356],
 [-0.25604803, -0.32569226,  1.68994262, -0.73388299],
 ...,
 [-0.40639746, -0.2142472 , -0.77556195,  0.96412543],
 [-0.40639746, -0.2142472 , -0.77556195,  0.44875546],
 [-0.25604803, -0.32569226,  1.50007676, -0.73388299]])

In [203]: X_test
Out[203]: array([[ -0.25604803, -0.32569226,  0.52506509, -0.73388299],
 [-0.40639746, -0.2142472 , -0.77556195, -0.18957997],
 [ 3.05163933, -0.32569226, -0.7737275 ,  1.94513668],
 ...,
 [-0.40639746, -0.2142472 , -0.77556195,  0.63953276],
 [-0.40639746, -0.2142472 , -0.77556195,  1.80680053],
 [-0.40639746, -0.2142472 , -0.77556195,  1.33302182]])

In [204]: # Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 15 , max_depth= 1, criterion = 'gini', random_state = 0)
classifier.fit(X_train, Y_train)

Out[204]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                                max_depth=1, max_features='auto', max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=15,
                                n_jobs=None, oob_score=False, random_state=0, verbose=0,
                                warm_start=False)

In [205]: y_pred = classifier.predict(X_test)
           print(y_pred)

           [1 0 0 ... 0 0 0]

In [206]: from sklearn import metrics
           print("Accuracy:",metrics.accuracy_score(Y_test, y_pred))

           Accuracy: 0.9808719091804469

```

Figure 10: Classification part 3/3

By using the random forest classifier we got an accuracy of 98% approx.

Chapter-4

(Result, Conclusion, Future work)

4.1 Result:

We attacked from one virtual machine to the other and capture packets through Wireshark where we detected the slowhttptest attack. We then analysed the packets and classified them using Random Forest Classification with an accuracy of 98%.

4.2 Conclusion:

In this project, we perform Slow Read DoS Attack by computer simulations. From results, we concluded that the attack by a single attacker is limited to a very few times.

After that we capture the attack and normal packets using Wireshark and classified the attack traffic and normal traffic using machine learning random forest classification algorithm. However, from results, we can derive the improvement of the Slow Read DoS Attack which we will do in our future work.

4.3 Future Work:

Since the attack connection is compulsorily disconnected by passing the Timeout, the attack success status of Web server returns to service available state. Moreover, if the total number of attack connections is less than MC/SL, the attack cannot succeed. So, the effectiveness of attack by a single attacker is small. However, if another attacker sends new attack connections before attack connections are disconnected, it will be expected that the length of attack success status can be maintained efficiently longer. Thus, we consider the scenario with which two or more attackers collude and many more. This technique is called slow read ddos attack which we will do in our future project and extend our work and more efficiently and effectively do the DoS attack on the server and waste the resources of the server.

4.4 REFERENCES:

- [1] Cambiaso Enrico, Papaleo Gianluca, Chiola Giovanni and Aiello Maurizio, “Slow DoS attacks: definition and categorisation,” Int. J. Trust Management in Computing and Communications, Vol. 1, Nos. 3/
- [2] Sergey Shekyan, “Are you ready for slow reading?,”
<https://community.qualys.com/blogs/securitylabs/slow-read>
- [3] Kelly Jackson Higgins, “New Denial-Of-Service Attack Cripples Web Server By Reading Slowly, <http://www.darkreading.com/attacks-breaches/new-denial-of-service-attack-cripples-we/232301367>
- [4] Sergey Shekyan, “slowhttpstest,” <https://code.google.com/p/slowhttpstest>
- [5] Analysis of Slow Read DoS Attack and Countermeasures paper presented at (Proceedings of the International Conference on Cyber-Crime Investigation and Cyber Security, Kuala Lumpur, Malaysia).
- [6] VMware Player, <http://www.vmware.com/jp/products/player>
- [7] Apache, <http://httpd.apache.org>
- [8] W3Techs, “Most popular web servers,” <http://w3techs.com>
- [9] Wireshark, <http://www.wireshark.org>