PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

MOHAMED BOUDIAF UNIVERSITY - M'SILA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

COMPUTER SCIENCE DEPARTMENT

DISSERTATION

Submitted in partial fulfillment of the requirements for the Degree of MASTER

Domain: Mathematics and Computer Science

Branch: Computer Science

Specialty: Networks

By: AICHAOUI TARIQ

TOPIC

Framework for detecting DoS/DDoS Attacks against Web servers

Publicly defended: / /2016 before a Jury composed of :

Ms. SAOUDI LALIA	**University of M'sila**	**Supervisor**
……	**University of M'sila**	**Chair**
……	**University of M'sila**	**Examiner**
……	**University of M'sila**	**Examiner**

Academic Year: 2015 /2016

ACKNOWLEDGEMENTS

Foremost, I am grateful to the almighty ALLAH for helping me to complete this research paper.

I would like to express my sincere gratitude to my advisor Ms. Saoudi Lalia, for all her guidance, and for her timely and highly up to the mark feedback and support.

I would like to thank my loved ones, who have supported me throughout the entire process, both by keeping me harmonious and helping me putting pieces together. I will be grateful forever for your love. My great friends , and all the doctors and professors for teaching us through our college years. I also place on record, my sense of gratitude to all those who, directly or indirectly, have lent their helping hand in this research paper. And to all my family for standing by me in every day of my life To all of you, my love and respect.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

General Introduction

1 Context:

Web applications are becoming more popular and widely being used in all aspects of work and social activities, unfortunately, with the growth of web technologies and availability of resources over internet number of attacks on servers providing these services, resources are increased.

One of the most popular attack targets are web-servers and web-based applications. A web server is a program that serves a request using the HTTP protocol. Initially web servers were using static Hyper Text Markup Language (HTML) pages to provide information. But nowadays the web servers provide dynamic services using database queries, executable script,etc. for providing information. The web is used for different kinds of services and various applications such as emailing,banking applications, real-time communication, etc.

2 Statement of the Problem:

Denial of service (DoS) attack against web server is one of the most dangerous attacks. Which attempts to make the web server unavailable to serve up the web sites they host to legitimate visitors. Usually, the users of web-servers request and send information using queries, which in HTTP traffic are strings containing a set of parameters having some values. Attackers are able to manipulate these queries and create requests which can corrupt the server.

Denial of Service attacks can result in significant loss of service, money and reputation for organizations. Typically, the loss of service is the inability of a particular network service, such as e-mail, to be available or the temporary loss of all network connectivity and services. An HTTP Denial of Service attack can also destroy programming and files in affected computer systems. In some cases, HTTP DoS attacks have forced Web sites accessed by millions of people to temporarily cease operation.

3 Objectives:

3.1 General Objectives

In this work, we propose an anomaly based system developed to detect DoS/DDoS attacks against web servers. The detection focus on two most attack target protocols TCP and HTTP.

3.1 Specific Objectives

The design of the framework handles all aspects of HTTP and TCP based DDoS attacks through the following three subsequent framework's layers:

- Firstly, an outer detector blocks attacking IP source if it is listed on the black list table.
- Secondly, the IP spoofed detector to validate whether the incoming request is launched by true IP source or a spoofed IP.
- Thirdly, two classifier modules are proposed to detect HTTP/TCP DDoS attacks, for this modules we have to :
    - Select the relevant features of the HTTP protocol, to calculate a new set of features to classify the HTTP traffic as normal or DoS attack.
    - Select the relevant features of the TCP protocol, to calculate a new set of features, to classify the TCP traffic as normal or DoS attack.

4 Thesis Outline:

To meet our objective this thesis is structured as follows:

The First chapter considers the content of our work. Firstly, we provide an overview of Hypertext Transfer Protocol and how HTTP client and server communicate, then we introduce the denial of service attack, its different concepts, techniques and tools.

The second chapter provide an overview of intrusion detection systems and taxonomy of defense mechanisms against DoS/DDoS attacks.

The third chapter presents the conceptual aspect of our DoS/DDoS attack detection framework and its different components, which are a packet sniffer, IP spoofing detector and TCP/HTTP classifier modules.

The fourth chapter shows the experiments to demonstrate the effectiveness of our proposed framework.

Finally, we conclude this thesis by a general conclusion, recommendations and different perspectives.

CHAPTER 1

Hypertext transfer protocol and Denial of service attacks

CHAPTER 1

HYPERTEXT TRANSFER PROTOCOL AND DENIAL OF SERVICE ATTACKS

1.1 Introduction

Many prominent web sites face so called Denial of Service Attacks (DoS).these attacks occur when an attacker attempts to make the web server, or servers, unavailable to serve up the web sites they host to legitimate visitors. For some time, it was thought that these types of attacks were generally used against large corporations, government sites, and activist sites as a form of protest to disrupt their web presence [05] .

However, more small and medium businesses are beginning to see their online presence disrupted by this type of attack.

Application Denial of Service attacks have rapidly become a commonplace threat for doing business on the Internet - more proof that Web application security is more critical now than ever. Denial of Service attacks can result in significant loss of service, money and reputation for organizations. Typically, the loss of service is the inability of a particular network service, such as e-mail, to be available or the temporary loss of all network connectivity and services .

In this chapter ,Before going more into details on DoS attacks we will talk about HTTP protocol that are typically used in the most dangerous DoS attacks.

1.2 Hypertext Transfer Protocol

The HyperText Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers[01] .

A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred. HTTP has been in use by the World-Wide Web global information initiative since 1990[01] .

Basically, HTTP is a TCP/IP based communication protocol that is used to deliver data (HTML files, image files, query results, etc.) on the World Wide Web. The default port is TCP 80, but other ports can be used as well. It provides a standardized way for computers to communicate with each other. HTTP specification specifies how clients' request data will be constructed and sent to the server, and how the servers respond to these requests[03] .

1.2.1 Protocol parameters

1.2.1.1 HTTP version

HTTP uses a "<major>.<minor>" numbering scheme to indicate versions of the protocol. The protocol versioningpolicy is intended to allow the sender to indicate the format of a message and its capacity for understanding further HTTP communication, rather than the features obtained via that communication. No change is made to the version number for the addition of message components which do not affect communication behavior or which only add to extensible field values.

The <minor> number is incremented when the changes made to the protocol add features which do not change the general message parsing algorithm, but which may add to the message semantics and imply additional capabilities of the sender. The <major> number is incremented when the format of a message within the protocol is changed[01] .

The version of an HTTP message is indicated by an HTTP-Version field in the first line of the message[01] .

*HTTP-Version = "HTTP" "/" 1*DIGIT "." 1*DIGIT*

The HTTP version of an application is the highest HTTP version for which the application is at least conditionally compliant.

Proxy and gateway applications need to be careful when forwarding messages in protocol versions different from that of the application. Since the protocol version indicates the protocol capability of the sender, a proxy/gateway MUST NOT send a message with a version indicator which is greater than its actual version. If a higher version request is received, the proxy/gateway MUST either downgrade the request version, or respond with an error, or switch to tunnel behavior[01] .

1.2.1.2 Uniform Resource Identifiers

URIs have been known by many names: WWW addresses, Universal Document Identifiers, Universal Resource Identifiers, and finally the combination of Uniform Resource Locators (URL) and Names (URN). As far as HTTP is concerned, Uniform Resource Identifiers are simply formatted strings which identify--via name, location,or any other characteristic--a resource [01] .

General Syntax :

URIs in HTTP can be represented in absolute form or relative to some known base URI , depending upon the context of their use. The two forms are differentiated by the fact that absolute URIs always begin with a scheme name followed by a colon[01] .

Http URL :

The “http” scheme is used to locate network resources via the HTTP protocol. This section defines the scheme specific syntax and semantics for http URLs[01] .

http_URL = "http:" "//" host [":" port] [abs_path ["?" query]]

If the port is empty or not given, port 80 is assumed. The semantics are that the identified resource is located at the server listening for TCP connections on that port of that host, and the Request-URI for the resource is abs_path. If the abs_path is not present in the URL, it MUST be given as “/” when used as a Request-URI for a resource [01] .

1.2.1.3 Date/Time Formats

HTTP applications have historically allowed three different formats for the representation of date/time stamps [01] :

Sun, 06 Nov 1994 08:49:37 GMT ; RFC 822, updated by RFC 1123

Sunday, 06-Nov-94 08:49:37 GMT ; RFC 850, obsoleted by RFC 1036

Sun Nov 6 08:49:37 1994 ; ANSI C's asctime() format

The first format is preferred as an Internet standard and represents a fixed-length subset of that defined by RFC 1123. The second format is based on the obsolete RFC 850 date format and lacks a four-digit year.

HTTP/1.1 clients and servers that parse the date value MUST accept all three formats (for compatibility with HTTP/1.0), though they MUST only generate the RFC 1123 format for representing HTTP-date values in header fields.

1.2.1.4 Character Sets

We use character sets to specify the character sets that the client prefers. Multiple character sets can be listed separated by commas. If a value is not specified, the default is the US-ASCII. Following are the valid character sets[03] :

US-ASCII orISO-8859-1or ISO-8859-7

1.2.1.5 Content Encodings

A content encoding value indicates that an encoding algorithm has been used to encode the content before passing it over the network. Content coding are primarily used to allow a document to be compressed or otherwise usefully transformed without losing the identity. All content-coding values are case-insensitive[03] .following are the valid encoding schemes:

Accept-encoding: gzip or Accept-encoding: compress or Accept-encoding: deflate

1.2.1.6Media Types

HTTP uses Internet Media Types in the **Content-Type** and **Accept** header fields in order to provide open and extensible data typing and type negotiation. All the Media-type values are registered with the Internet Assigned Number Authority (IANA). The general syntax to specify media type is as follows [03] :

*Media-type = type "/" subtype *(";" parameter)*

The type, subtype, and parameter attribute names are case--insensitive. Example :

Accept: image/gif

1.2.1.7Language Tags

HTTP uses language tags within the **Accept-Language** and **Content-Language** fields. A language tag is composed of one or more parts: a primary language tag and a possibly empty series of subtags[03] :

*language-tag = primary-tag *("-" subtag)*

White spaces are not allowed within the tag and all tags are case- insensitive. Example tags include:

en, en-US, en-cockney, i-cherokee, x-pig-latin

Where any two-letter primary-tag is an ISO-639 language abbreviation and any two-letter initial subtag is an ISO-3166 country code[03] .

1.2.2HTTP Message

HTTP client and server communicate by sending text messages. The client sends a *request message* to the server. The server, in turn, returns a response message[04] .Both types of message consist of a start-line, zero or more header fields (also known as "headers"), an empty line indicating the end of the header fields, and possibly a message-body[01] .

1.2.2.1 HTTP Requests

An HTTP client sends an HTTP request to a server in the form of a request message which includes following format [03] :

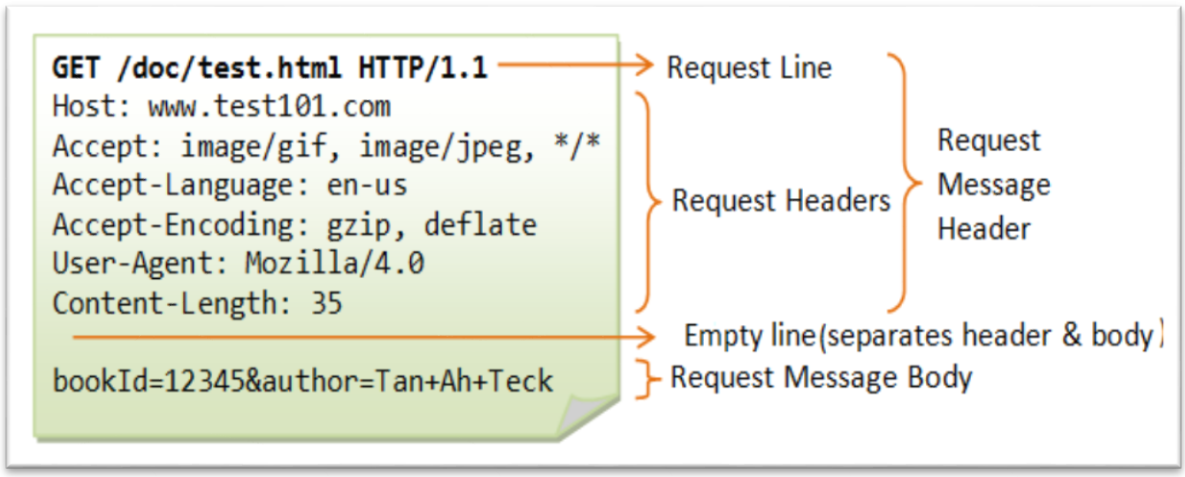


Figure 1.1: sample HTTP request message [04].

A. Request Line

The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF. The elements are separated by space SP characters [03] .

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

B. Request Header Fields

The request-header fields allow the client to pass additional information about the request, and about the client itself, to the server. These fields act as request modifiers, with semantics equivalent to the parameters on a programming language method invocation[01] .Given below is a list of the most common request header fields [02] :

Header field name	Description
Accept	tells the server what kinds of content the client is willing to accept, such as image types, office document formats, and so on.
Accept-Encoding	tells the server what kinds of content encoding the client is willing to accept.
Accept-Language:	tells the server what kinds of human languages the client is willing to accept.
User-Agent	Provides information about the browser or other client software that generated the request.
Referer	Specifies the URL from which the current request originated.
Connection	tells the other end of the communication whether it should close the TCP connection after the HTTP transmission has completed or keep it open for further messages.
Content-Encoding	specifies what kind of encoding is being used for the content contained in the message body, such as gzip, which is used by some applications to compress responses for faster transmission.
Content-Length	Specifies the length of the request body.
Content-Type	Specifies the type of content contained in the message body, such as text/html for HTML documents
Host	Specifies the hostname that appeared in the full URL being requested.

Transfer-Encoding	Specifies any encoding that was performed on the message body to facilitate its transfer over HTTP. It is normally used to specify chunked encoding when this is employed.
Authorization	submits credentials to the server for one of the built-in HTTP authentication types.
Cookie	submits cookies to the server that the server previously issued.
If-Modified-Since	Specifies when the browser last received the requested resource. If the resource has not changed since that time, the server may instruct the client to use its cached copy, using a response with status code 304.

Table 1.1: Request Header Fields**.**

1.2.2.2 HTTP Responses

After receiving and interpreting a request message, a server responds with an HTTP response message[01] .

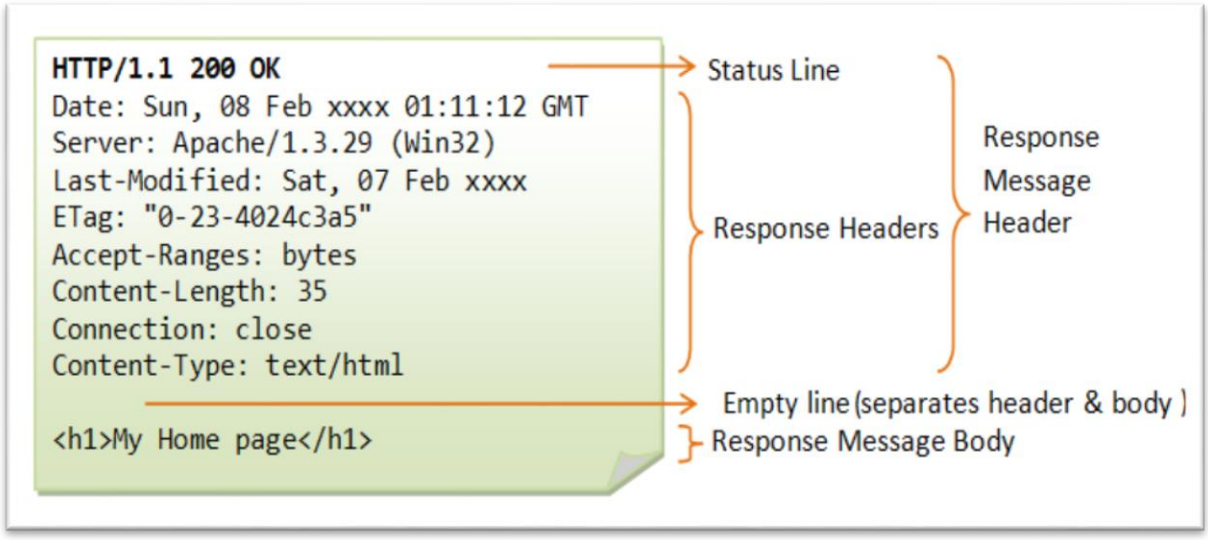


Figure1.2: sample HTTP response message [04].

A. Status-Line

The first line of a Response message is the Status-Line, consisting of the protocol version followed by a numeric status code and its associated textual phrase, with each element separated by space SP characters [01].

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

B. Status Code and Reason Phrase

The Status-Code element is a 3-digit integer where first digit of the Status-Code defines the class of response and the last two digits do not have any categorization role. There are five values for the first digit as it shown in table 2.1.

HTTP status codes are extensible and HTTP applications are not required to understand the meaning of all the registered status codes. Given in table1.3 a list of the most common Status Codes [03].

S.N.	Code and Description
1	**1xx: Informational** It means the request was received and the process is continuing.
2	**2xx: Success** It means the action was successfully received, understood, and accepted.
3	**3xx: Redirection** It means further action must be taken in order to complete the request.
4	**4xx: Client Error** It means the request contains incorrect syntax or cannot be fulfilled.
5	**5xx: Server Error** It means the server failed to fulfill an apparently valid request.

Table 1.2: classes of response[03].

Status Code and Reason Phrase	Description
100 Continue	Only a part of the request has been received by the server, but as long as it has not been rejected, the client should continue with the request.
101 Switching Protocols	The server switches protocol.
200 OK	The request is OK.
201 Created	The request is complete, and a new resource is created .
202 Accepted	The request is accepted for processing, but the processing is not complete.

204 No Content	A status code and a header are given in the response, but there is no entity-body in the reply.
300 Multiple Choices	A link list. The user can select a link and go to that location. Maximum five addresses .
301 Moved Permanently	The requested page has moved to a new url .
302 Found	The requested page has moved temporarily to a new url .
303 See Other	The requested page can be found under a different url .
400 Bad Request	The server did not understand the request.
401 Unauthorized	The requested page needs a username and a password.
402 Payment Required	You can not use this code yet.
403 Forbidden	Access is forbidden to the requested page.
404 Not Found	The server can not find the requested page.
500 Internal Server Error	The request was not completed. The server met an unexpected condition.
501 Not Implemented	The request was not completed. The server did not support the functionality required.
502 Bad Gateway	The request was not completed. The server received an invalid response from the upstream server.
503 Service Unavailable	The request was not completed. The server is temporarily overloading or down.

Table 1.3: List of Common HTTP Status Codes [03].

C. Response Header Fields

The response-header fields allow the server to pass additional information about the response which cannot be placed in the Status-Line. These header fields give information about the server and about further access to the resource identified by the Request-URI[01] .Given below is a list of the most common response header fields [02] :

Header field name	Description
Access-Control-Allow-Origin	indicates whether the resource can be retrieved via cross-domain Ajax requests.
Cache-Control	Passes caching directives to the browser (for example no-cache).
ETag	Specifies an entity tag. Clients can submit this identifier in future requests for the same resource in the If-None-Match header to notify the server which version of the resource the browser currently holds in its cache.
Expires	tells the browser for how long the contents of the message body are valid. The browser may use the cached copy of this resource until this time.
Location	is used in redirection responses (those that have a status code starting with 3) to specify the target of the redirect.
Server	provides information about the web server software being used
Set-Cookie	issues cookies to the browser that it will submit back to the server in subsequent requests.
WWW-Authenticate	is used in responses that have a 401 status code to provide details on the type(s) of authentication that the server supports.
X-Frame-Options	indicates whether and how the current response maybe loaded within a browser frame
Connection	tells the other end of the communication whether it should close the TCP connection after the HTTP transmission has completed or keep it open for further messages.

Content-Encoding	specifies what kind of encoding is being used for the content contained in the message body, such as gzip, which is used by some applications to compress responses for faster transmission.
Content-Length	specifies the length of the response body

Table 1.4: Response Header Fields**.**

1.2.3 HTTP Methods

HTTP protocol defines a set of methods. A client can use one of these methods to send a request message to an HTTP server [04] . The methods are:

Method	Description
GET	The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.
HEAD	Same as GET, but transfers the status line and header section only.
POST	A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.
PUT	Replaces all current representations of the target resource with the uploaded content.
DELETE	Removes all current representations of the target resource given by a URI.
CONNECT	Establishes a tunnel to the server identified by a given URI.
OPTIONS	Describes the communication options for the target resource.
TRACE	Performs a message loop-back test along the path to the target resource.

Table 1.5:HTTP Methods [03].

1.3 Denial of service

A DoS attack is an attack on a computer system or network that causes a loss of service to users, typically the loss of network connectivity and services by consuming the bandwidth of the victim network or overloading the computational resources of the victim system[06] .Using available tools, it is relatively easy to mount DoS attacks against remote networks.

1.3.1 Modes of DOS attack

Denial-of-service attacks come in a variety of forms and aim at a variety of services. There are three basic types of attack [08]:

1.3.1.1 Consumption of Scarce Resources

Computers and networks need certain things to operate: network bandwidth, memory and disk space, CPU time, data structures, access to other computers and networks, and certain environmental resources such as power, cool air, or even water.

A. Network Connectivity :

Denial-of-service attacks are most frequently executed against network connectivity. The goal is to prevent hosts or networks from communicating on the network. An example of this type of attack is the **"SYN flood"** attack. In this type of attack, the attacker begins the process of establishing a connection to the victim machine, but does it in such a way as to prevent the ultimate completion of the connection. In the meantime, the victim machine has reserved one of a limited number of data structures required to complete the impending connection. The result is that legitimate connections are denied while the victim machine is waiting to complete bogus "half-open" connections [08] .

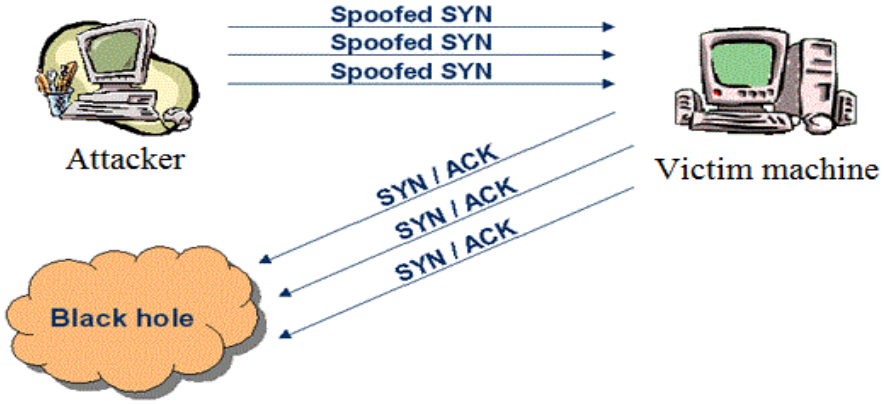


Figure1.3: SYN Flood attack.

B. Using Your Own Resources Against You:

An intruder can also use your own resources against you in unexpected ways. An example of this type of attack is **"UDP Bomb"** attack. An attacker can use the UDP and one of several services that echo packets on receipt to create Service-denying network congestion by generating a flood of UDP packets between two target systems. For example, the UDP chargen service on the first computer (which is a testing tool that generates a series of characters for every packet that it receives) sends packets to another system's UDP echo service, which echoes every character it receives. By exploiting these testing tools, an endless flow of echoes goes back and forth between the two systems, congesting the network [09] .
In addition to port 7 the echo port, an attacker can use port 17, the quote-of-the-day service , or the daytime service on port 13.These services also echo packets they receive. Disabling unnecessary UDP services on each computer (especially those mentioned) or using a firewall to filter those ports and services will protect you from this type of attack[09] .

C. Bandwidth Consumption :

An intruder may also be able to consume all the available bandwidth on your network by generating a large number of packets directed to your network. Typically, these packets are **ICMP ECHO** packets, but in principle they may be anything[08] .An example of this type of attack is **"Smurf attack"** attack.

The *Smurf attack* is a form of "brute force" attack that uses the same method as the ping flood but that directs the flood of ICMP echo request packets at the network's router. The destination address of the ping packets is the broadcast address of the network, which causes the router to broadcast the packet to every computer on the network or segment. This can result in a very large amount of network traffic if there are many host computers, creating congestion that causes a denial of service to legitimate users[09] .

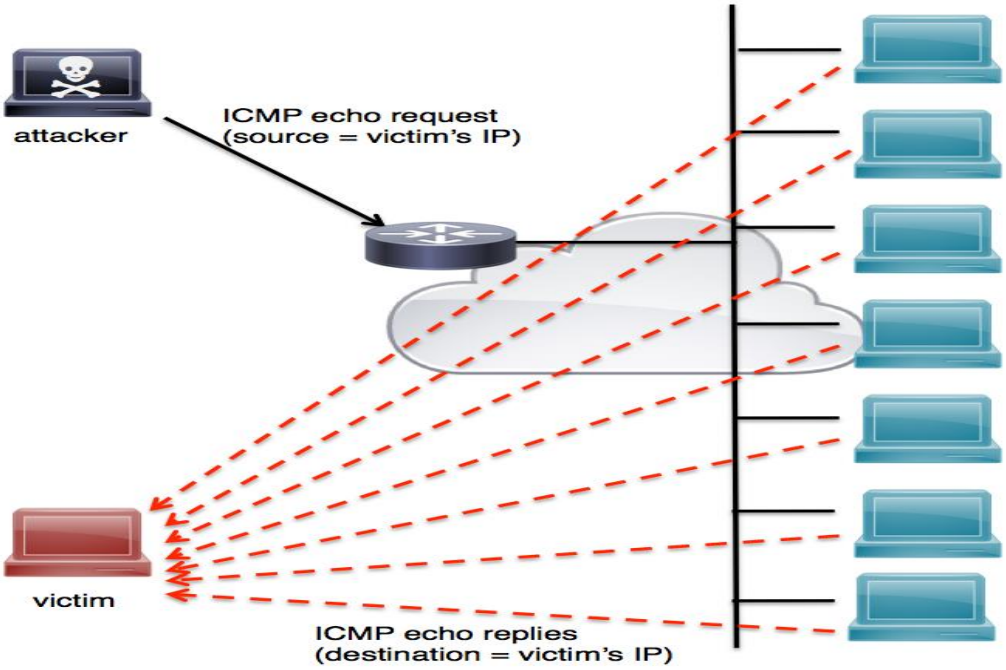


Figure1.4: Smurf attack [10]

D. Consumption of Other Resources:

In addition to network bandwidth, intruders may be able to consume other resources that your systems need in order to operate. For example, in many systems, a limited number of data structures are available to hold process information (process identifiers, process table entries, process slots, etc.).

An intruder may be able to consume these data structures by writing a simple program or script that does nothing but repeatedly create copies of itself. Many modern operating systems have quota facilities to protect against this problem, but not all do. Further, even if the process table is not filled, the CPU may be consumed by a large number of processes and the associated time spent switching between processes. Consult your operating system vendor or operating system manuals for details on available quota facilities for your system[08].

1.3.1.2 Destruction or Alteration of Configuration Information

An improperly configured computer may not perform well or may not operate at all. An intruder may be able to alter or destroy configuration information that prevents you from using your computer or network.

For example, if an intruder can change the routing information in your routers, your network may be disabled. If an intruder is able to modify the registry on a Windows NT machine, certain functions may be unavailable[08] .

1.3.1.3 Physical Destruction or Alteration of Network Components

The primary concern with this type of attack is physical security. We should guard against unauthorized access to computers, routers, network wiring closets, network backbone segments, power and cooling stations, and any other critical components of your network. Physical security is a prime component in guarding against many types of attacks in addition to denial of service[08] .

1.3.2 Distributed Denial of Service

A Distributed Denial of Service (DDoS) attack uses many computers to launch a coordinated DoS attack against one or more targets. Using client/server technology, the perpetrator is able to multiply the effectiveness of the Denial of Service significantly by harnessing the resources of multiple unwitting accomplice computers, which serve as attack platforms [06].

Typically, a DDoS attack use two types of components: agents, which run on compromised hosts and generate the actual attack messages; and a handler, which is a program that controls the agents, telling them when to attack, what to attack, and how to attack. Agents are also referred to as bots, and a collection of hosts that are running bots that are controlled by a single attacker is called a botnet. The Figure 1.5 illustrates the steps of a typical DDoS attack. First, an attacker compromises vulnerable hosts in the Internet and deploys attack tools (agents) on them. Next, the attacker disseminates an attack command from the handlers to the agents, instructing the agents on what to attack, when to attack and how to attack. Starting at the instructed attack time, agents generate attack traffic towards to the target to carry out the attack [07].

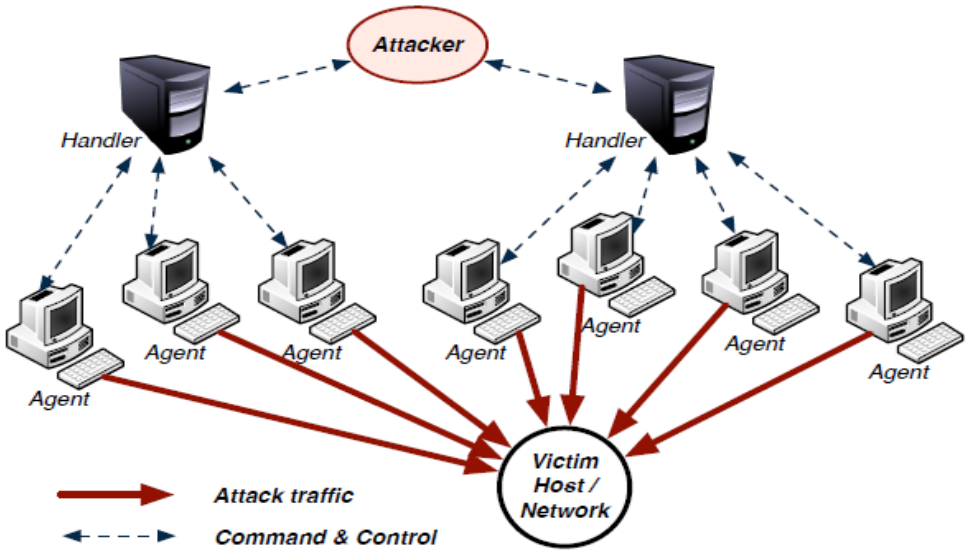


Figure1.5: Distributed Denial of Service Attack [07]

1.3.3 Application layer Distributed Denial of Service attacks

Application layer Distributed Denial of Service (DDoS) attacks are among the deadliest kinds of attacks that have significant impact on destination servers and networks due to their ability to be launched with minimal computational resources to cause an effect of high magnitude. Commercial and government Web servers have become the primary target of these kinds of attacks, with the recent mitigation efforts struggling to deaden the problem efficiently [11] .

Most application layer DDoS attacks can successfully mimic legitimate traffic without being detected by Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS). Application layer DDoS are among the most dangerous form of network attacks that are launched by an attacker using specific and well-written scripted tools or through thousands of compromised networked hosts (botnets) through Command and Control(C&C) or Internet Relay Chat (IRC) servers. These kinds of attacks prevent legitimate users from accessing server resources due to service and connection unavailability. Companies also lose a sizeable amount of revenue due to the application layer DDoS attacks as attacks on commercial Web servers can successfully damage business-critical services and lead to a serious loss within a short period of time[11] .

The average financial cost that small-to-medium-sized businesses incur per DDoS attack incident is $52, 000 and $440, 000 for larger businesses. Most companies today do not have adaptive filtering capability to filter the typical DDoS attacking methods like SYN and HTTP GET floods [11].

1.3.3.1 Application layer DDoS features

Application layer DDoS attacks are more worrying than the attacks at other layers because of following reasons [12] :

- High obscurity: These attacks follow Legitimate TCP or UDP connections, so it is very difficult to differentiate them from legitimate users.
- Highly efficient: These attacks require very less number of connections.
- Affect multiple applications: They may affect many different applications. Any one of the protocols examined above may be subject to a DDoS attack. Many of them targets HTTP to exhaust a web server’s vitality.
- Highly targeted: These attacks are highly-targeted oriented. Generally, these attacks are tailored to aim a specific application. For example, web servers that run a combination of Java, PHP5, and ASP.NET may be targeted by specially

crafted HTTP requests, which may collide with the web server´s hashing operation "when unique requests return non-unique and overlapping responses".

- Simplicity in exploitation: These attacks may exploit the simplicity in the application layer. For example, a simultaneous refresher of browsers by thousands of users may collapse the server soon or later.

- Multiple effects: These attacks may affect many victims directly or indirectly. For example, a DNS attack at a single DNS provider may affect all of its customers.
- Requirement of limited resources: These attacks require limited resources, so limited investment by attackers can result a successful attack.
- Follow normal traffic rules: Traffic involved in these attacks seems to be legitimate as it follows the protocol rules, rate and complete TCP handshake process. It follows all the basic requirements that a normal traffic flows.

1.3.3.2 Application level DOS categories

Application layer DOS attacks can be categorized into following categories [12] :

A. High-Rate distributed denial of service attack

A high rate distributed denial of service attack is a synonym for the traditional DDoS attacks when attackers exceed and violate the adopted threshold value[13] .

B. Low-Rate distributed denial of service attack

In the last years, a new category of DDoS threats has emerged, known as Low-Rate DDoS Attack or Slow DDoS Attack[14] .They make use of a tiny amount of network bandwidth to perform their malicious activity. They often directly target the application layer of the victim, instead of the transport or network layers because it requires less computational and network resources. While attacks at the communication layers typically require to flood the victim with a continuous stream of packets, attacks at the application layer send quite a few packets with appropriate timing. The malicious timing and content induces the server to maintain connections alive for long periods. The result is an allocation of the communication channels from the attacker[14] .

The resulting limitation prevent legitimate clients to communicate with the server. The malicious data transfer may remain silent for several seconds or even minutes before

proceeding with another stage of the attack. More specifically, these threats usually work by first seizing all the available connections on the targeted listening daemon and then make use of a Wait Timeout to temporarily interrupt the attack. After the timeout expiration, the attack is re-activated, otherwise, some other server-side timeout would close the connections[14] .

This on-off behavior may be repeated indefinitely. In this way, the attacking bandwidth is significantly reduced and the malicious traffic appears so similar to a legitimate one.[14] .

1.3.3.3 HTTP flood attacks :

HTTP flood is a type of Application layer DoS/DDoS attacks in which the attacker exploits seemingly-legitimate HTTP GET or POST requests to attack a web server or application. A sophisticated Layer 7 attack, HTTP floods do not use malformed packets, spoofing or reflection techniques, and require less bandwidth than other attacks to bring down the targeted site or server. As such, they demand more in-depth understanding about the targeted site or application, and each attack must be specially-crafted to be effective. This makes HTTP flood attacks significantly harder to detect and block [19].

The attack is most effective when it forces the server or application to allocate the maximum resources possible in response to each single request. Thus, the perpetrator will generally aim to inundate the server or application with multiple requests that are each as processing-intensive as possible. For this reason HTTP flood attacks using POST requests tend to be the most resource-effective from the attacker's perspective; as POST requests may include parameters that trigger complex server-side processing. On the other hand, HTTP GET-based attacks are simpler to create, and can more effectively scale in a botnet scenario [19].

1.4. Denial of Service Tools

In this section, we describe some of the DoS tools required in our study.

1.4.1 LOIC

Low Orbit Ion Cannon (LOIC) Tool was built by Praetox Technologies as a stress testing application.

The tool performs a simple DoS attack, by sending a sequence of TCP (Transmission Control Protocol), UDP (User Datagram Protocol) or HTTP (Hyper-Text Transfer Protocol) requests to a target host. In Figure 1.6, we show the initial screen of LOIC[15] .

LOIC tool allows the user to select a target host, a method of attack (TCP, UDP or HTTP), and some other parameters to customize the requests that will be sent. For example, the user can control the destination port number, the content of the messages (package payload), the number of concurrent threads, request timeout, etc.

The version 1.1.1.4 added an option that allows the tool to be remotely controlled, using the IRC (Internet Relay Chat) protocol. In this case, the user machine becomes part of a botnet.

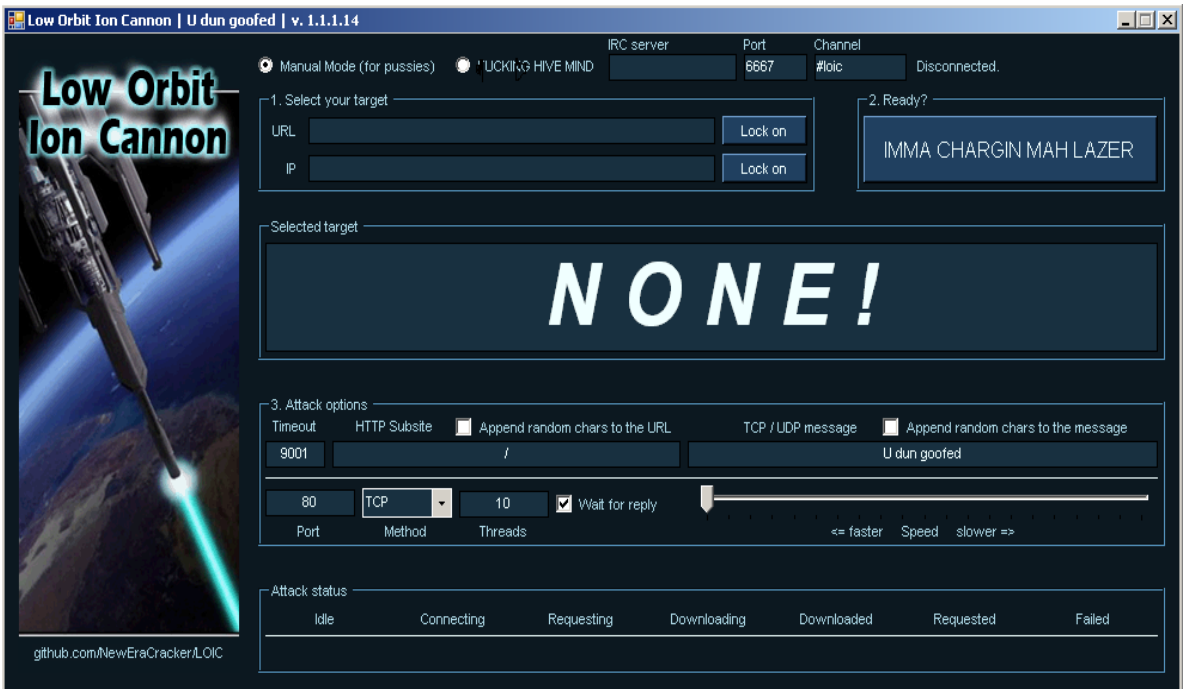


Figure 1.6 : Initial screen LOIC –version 1.1.1.4[15].

1.4.1.1 Tool Description

LOIC has two modes of operation: the manual mode, where the target address and all other parameters have to be filled in by the user; and the automatic mode, where the attacks are remotely controlled. The automatic mode or Hive Mind, as it is called in the tool, can be seen as an option to voluntarily join a botnet. When using this mode, all parameters of an attack (including the target) are set up remotely via IRC. IRC is a network protocol designed to provide real-time group chat. However, it is often (mis)used to control botnets.

To use this mode, the user has to fill in the address of the IRC server and the name of the channel ("#loic" by default), through which the attack commands will be sent[15] .Once the tool is started in the automatic mode, the commands are set via the topic of the IRC channel. They have a straight-forward plain-text format:

!lazor args start

!lazor stop

As it could be expected, the first command starts an attack while the second stops it again. The variable "args" contains a list of parameters, which includes the target host, the target port (when applicable), the attack method and a message to be included in the attack packets, among others. At the time of writing, the following was the topic in one of the channels associated with "Operation Payback"[15] :

!lazor default targethost=www.moneybookers.com subsite=/ speed=3

threads=15 method=tcp wait=false random=true checked=false

message=Sweet_dreams_from_AnonOPs port=80 start

1.4.2 HOIC

This DoS attack tool is very new and is the next-generation of LOIC. While LOIC uses TCP, UDP and HTTP attacks, HOIC uses only HTTP attacks. HOIC's goal is to disrupt multiple target services simultaneously by flooding them with HTTP POST and GET requests. HOIC differs from LOIC in two more areas. First, it can attack up to 256 different web addresses simultaneously, or alternatively multiply its attack strength to one target 256 times. Second, it can dynamically change its attack signature, making detection extremely difficult[16].

HOIC's boosters are used to tailor the HTTP requests sent by HOIC to the target for a specific type of attack. "HOIC is pretty useless," the documentation file that comes with the code says, "unless it is used in combination with 'Boosters'" And that's putting it mildly.

The attack code is generated based completely on what's in the booster file. When an attack is launched, HOIC compiles the booster to create the HTTP headers to be sent, and sets the mode of the attack[17] .

1.4.3 Slowhttptest

SlowHTTPTest is a highly configurable tool that simulates some Application Layer Denial of Service attacks. It implements most common low-bandwidth Application Layer DoS attacks, such as slowloris, Slow HTTP POST, Slow Read attack by draining concurrent connections pool, as well as Apache Range Header attack by causing very significant memory and CPU usage on the server.

Although its official intention is legitimate, testing the tool's availability online makes it useful for black-hat hackers as well. Without modifications, the tool is easy to identify because all the requests' Referer headers direct to the tool's development site[17].Slowhttptest implements three DoS methods:

1.4.3.1 Slowloris:

The Slowloris attack represents one of the most-known slow DoS threats. It may be seen as the SYN flood attack working at the application layer. Indeed, Slowloris sends legitimate but incomplete HTTP requests to the victim. In this way, the attack takes forces the web server to wait for the end of the requests, which will never be sent by the attacker, thus resulting in an endless wait. In particular, after a connection is established, Slowloris sends to the server a first part of the HTTP request, thus triggering the Wait Timeout. At the timeout expiration, an additional HTTP parameter is sent to the server, and the timeout is triggered again[14] .

1.4.3.2 Slow HTTP POST (SlowPost)

Slow HTTP POST method is very similar to Slowloris. Here, the requests are HTTP POST requests, in which all the headers are sent correctly, including the Content-length. After the headers are sent and received, the POST message body is sent at a very low rate, thus keeping the connection open for a prolonged time. The server has to wait until all content arrives according to the declared Content-length. Unlike Slowloris, Slow HTTP POST can't be mitigated by load balancers[18] .

1.4.3.3 Slow Read

The Slow Read attack works by sending legitimate HTTP requests to the server at usual rate. The purpose of the attack is to maintain connections alive by slowly reading the replies received by the server. This is possible by specifying a small client side reception buffer in the initial SYN packets sent to the victim during the connections establishment. In turn, this fictitious delay will slow down the responses of the server, too. In this way, the connections are kept alive by the server itself, thus eluding traditional protection systems. In case of the request of a large file, connections may be kept alive for very long times[14] .

1.4.4 More tools

Table1.6 summarizes the features of some other freely available DDoS tools.

Tool name	DDoS method	Accept Headers	Fingerprint	Evasion Techniques
Are-You-Dead-Yet	Slow POST	None		
Tor's Hammer	Slow POST	None	Content Length= 10,000	Random User-Agent
Simple Slowloris Flooder	Slowloris	None	Content Length= 5,235	Simple Slowloris Flooder
Nuclear DDoSer	Slowloris Slow POST	Accept Language, Accept Charset	data= nuclear ddoser	Changing Proxies
Dirt Jumper	HTTP flood SYN flood POST flood and more	None		Random User-Agent

Table 1.6: A sample of DDoS tools and their characteristics [18].

1.5 Conclusion

Denial-of-service is simple attack, it rarely use any sophisticated mechanism or complicated and covert actions (like viruses, worms or intrusion tools do). The difficulty in handling DDoS and low DoS attacks lies exactly in their simplicity. Because they misuse legitimate protocols to perform denial-of-service, it is extremely difficult to separate attack traffic from legitimate one, IP spoofing additionally complicates the problem.

This chapter has provided an overview of attack methods, on both TCP and HTTP protocol level, and popular attack tools. The next chapter provides I an overview of intrusion detection systems, then a taxonomy of defense mechanisms against DoS/DDoS attacks.

CHAPTER 2

Intrusion detection systems and DDoS Defense mechanisms

CHAPTER 2

INTRUSION DETECTION SYSTEMS

AND

DDOS DEFENSE MECHANISMS

2.1 Introduction

A DDoS is nothing more than a never-ending stream of requests from a large number of sources. The only way to protect against this is by having a system to identify the DDoS source and block it.

This is easier said than done. Identifying the source of a DDoS attack can be tricky and, in most cases, involves tweaking an intrusion detection system (IDS) to differentiate between legitimate requests and attacks.

In this chapter, we first talk about Intrusion Detection Systems and what they are, then we classify the defense mechanisms against DDoS attacks using three criteria. The first criterion for classification is the location where the defense mechanism is implemented (i.e., Deployment location). The second criterion is the protocol level on which defense mechanism works, and the last is the point of time when the DDoS defense mechanisms should act in response to a possible DDoS attack.

2.2 Intrusion Detection Systems

2.2.1 History of Intrusion Detection Systems

The idea of an automated Intrusion detection system goes back to 70's but wasn't introduced publicly until the 80's of the past century where James P. Anderson published a study outlining ways to improve computer security auditing and surveillance at customer sites. He introduced the notion that the traceability and audit data may contain vital information for understanding and analyzing user behavior. The original idea behind automated intrusion detection (ID) is often credited to him for his paper on "How to use

accounting audit files to detect unauthorized access". This ID study paved the way as a model of misuse detection for mainframe systems.

Between 1984 and 1986, Dorothy Denning and Peter Neumann researched and developed the first model of real-time IDS. This prototype was named the Intrusion Detection Expert System (IDES). This same system has been refined and enhanced to form what is known today as the Next-Generation Intrusion Detection Expert System (NIDES) [21].

The report published by James P. Anderson and the work on the IDES was the start of much of the research on IDS throughout the 1980s and 1990s. During this period, the U.S. government funded most of this research. Projects like Discovery, Haystack, Multics Intrusion Detection and Alerting System (MIDAS), Network Audit Director and Intrusion Reporter (NADIR) were all developed to detect intrusions. In the mid-1990s, commercial products surfaced for the masses. Two of the most popular IDS in the mid- 1990s were Wheelgroup's, Netranger and Internet Security Systems RealSecure. Both of these companies started out with network based IDS (NIDS).

Nowadays, more vendors are advertising they can process at gigabit speed. Internet Security Systems (ISS), Snort, Prelude, Cisco Systems and Intrusion.com advertise that they can analyze and alert on gigabit traffic [21].

Paul Anderson document written for a North American governmental organization, introduced the notion that the traceability and audit data may contain vital information for understanding and analyzing user behavior. Assumptions Anderson helped develop the foundation for future intrusion detection systems, and particularly the HIDS [21].

2.2.2 Some Definitions

2.2.2.1 Intrusion

Intrusions in computer systems are occurring at an increasingly alarming rate. Some sites report that they are the targets of hundreds of intrusion attempts per month. Moreover, there are numerous different intrusion techniques used by intruders [22].

2.2.2.2 Intrusion Detection

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network. Intrusions are caused by attackers accessing the systems from the Internet,

authorized users of the systems who attempt to gain additional privileges for which they are not authorized, and authorized users who misuse the privileges given them [23].

2.2.2.3 Intrusion Detection System

Intrusion Detection System (IDS) is software, hardware or combination of both used to detect intruder activity. An IDS may have different capabilities depending upon how complex and sophisticated the components are. IDS appliances that are a combination of hardware and software are available from many companies. As mentioned earlier, an IDS may use signatures, anomaly-based techniques or both [24].

There are three concepts to classify intrusion-detection systems, which are summarized in Figure 2.1.

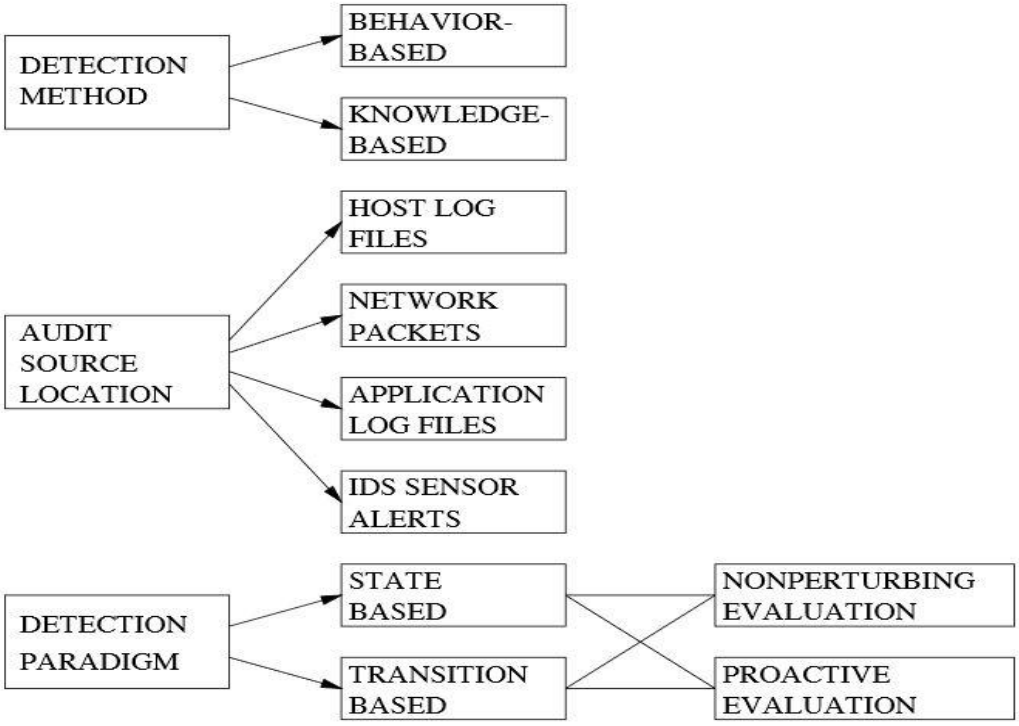


Figure 2.1: Characteristics of intrusion-detection systems [25].

2.2.3 Types of IDS

There are several types of IDS that can be deployed to aid security administrators in their endeavors. Two types, network-based intrusion detection systems (NIDS) and host-based intrusion detection systems (HIDS) are most prevalent in modern security deployments. There are other types of IDS, however, which include file integrity and log file checkers, and decoy devices known as honeypots. Additionally, there exist hybrid systems that combine some of the different functionalities mentioned earlier. We'll discuss each of these IDS in this section.

2.2.3.1 Host IDS

Host-based intrusion detection systems (HIDS) are systems that sit at service endpoints rather than in the network transit points like NIDS. The first type of IDS that's widely implemented, Host IDS, is installed on servers and is more focused on analyzing the specific operating system and application functionality residing on the HIDS host. HIDS are often critical in detecting internal attacks directed towards an organization's servers such as DNS, mail, and web servers. HIDS can detect a variety of potential attack situations such as file permission changes and improperly formed client-server requests [26].

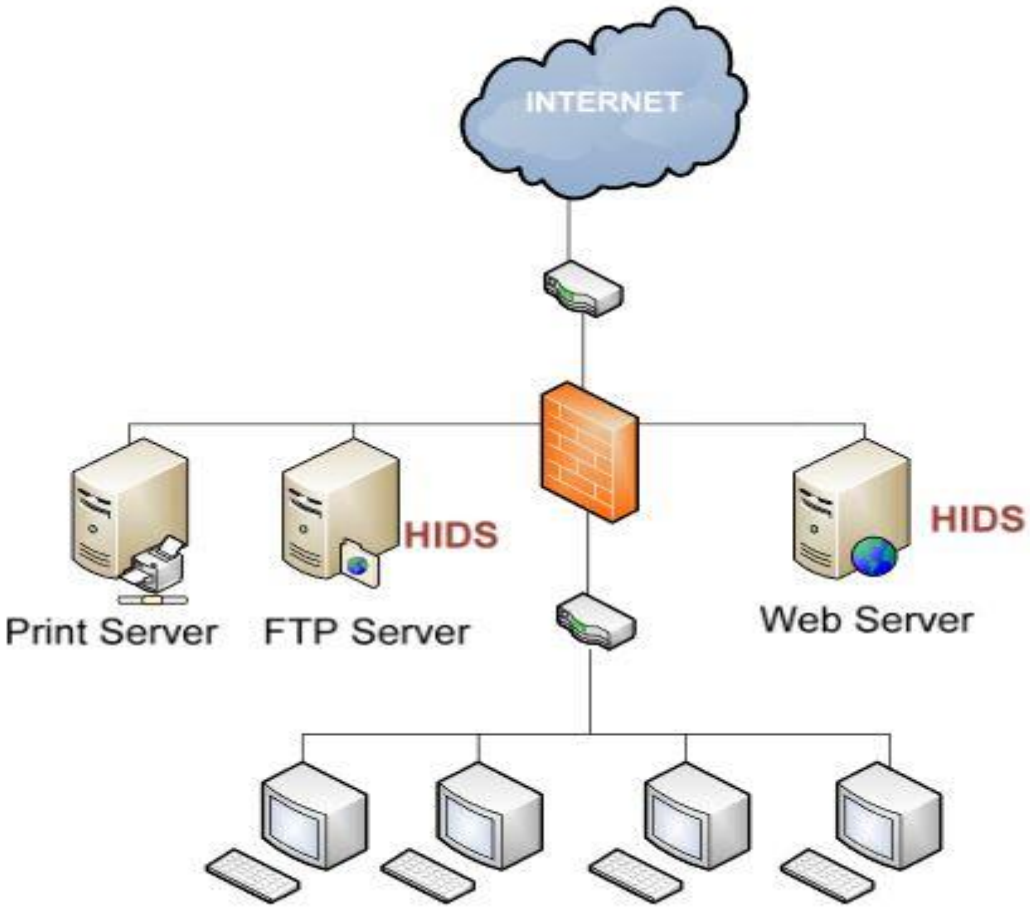


Figure 2.2: Host-Based IDS [27].

2.2.3.2 Network IDS

Network-based intrusion detection systems (NIDS) are devices intelligently distributed within networks that passively inspect traffic traversing the devices on which they sit. NIDS can be hardware or software-based systems and, depending on the manufacturer of the system, can attach to various network mediums such as Ethernet. Oftentimes, NIDS have two

network interfaces. One is used for listening to network conversations in promiscuous mode and the other is used for control and reporting [26].

With the advent of switching, which isolates unicast conversations to ingress and egress switch ports, network infrastructure vendors have devised port-mirroring techniques to replicate all network traffic to the NIDS. There are other means of supplying traffic to the IDS such as network taps. In some network equipment, includes NIDS components directly within the switch [26].

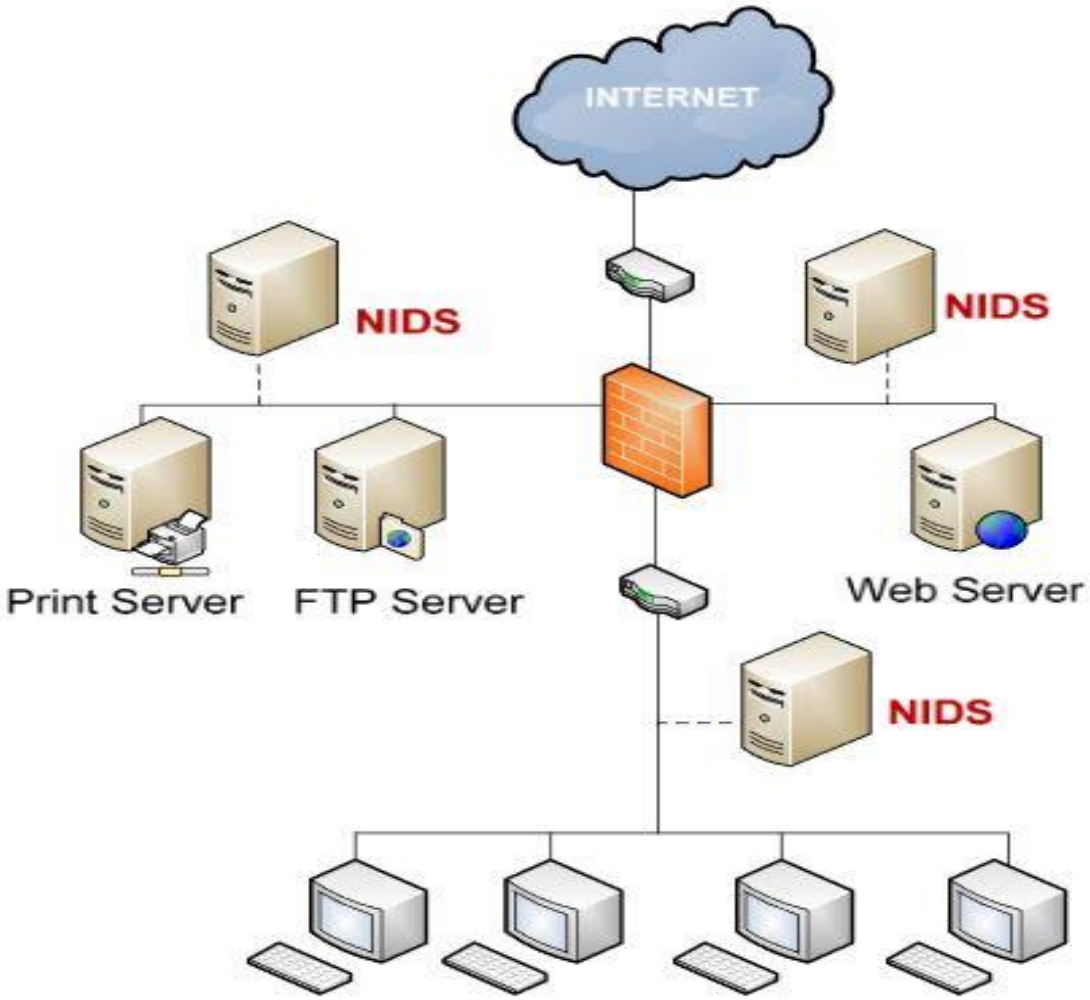


Figure 2.3 : Network-Based IDS [27].

2.2.3.3 Hybrid IDS

Hybrid IDS are systems that combine both Host IDS and limited Network IDS functionality on the same security platform. A Hybrid IDS can monitor system and application events and verify a file system's integrity like Host IDS, yet because the monitoring network interface runs in a non-promiscuous mode, the Network IDS functionality

only serves to analyze traffic destined for the device itself. A Hybrid IDS is often deployed on an organization's most critical servers [26].

2.2.4 Approaches to Intrusion Detection

The two major approaches that are used by IDSs to detect intrusive behavior are called anomaly detection and misuse detection. The anomaly detection approach is based on the premise that an attack on a computer system (or network) will be noticeably different from normal system (or network) activity, and an intruder (possibly masquerading as a legitimate user) will exhibit a pattern of behavior different from the normal user [28]. So, the IDS attempts to char,acterize each user's normal behavior, often by maintaining statistical profiles of each user's activities. Each profile includes information about the user's computing behavior such as normal login time, duration of login session, CPU usage, disk usage, favorite editor, and so forth. The IDS can then use the profiles to monitor current user activity and compare it with past user activity. Whenever the difference between a user's current activity and past activity falls outside some predefined "bounds" (threshold values for each item in the profile), the activity is considered to be anomalous, arid hence suspicious [22].

In the misuse detection approach, the IDS watches for indications of "specific, precisely representable techniques of computer system abuse" [28]. The IDS includes a collection of intrusion signatures, which are encapsulations of the identifying characteristics of specific intrusion techniques. The IDS detects intrusions by searching for these intrusion signatures in the records of user activities.

Although there exist only the above two major approaches to intrusion detection, IDSs are nevertheless quite diverse in their designs. Different IDSs employ different algorithms, different criteria for identifying intrusive behavior, and so forth. Also, several IDSs use a combination of both detection approaches [22].

Often, the main source of information about user activity for an IDS is the set of audit records from the computer system. However, relying on audit records alone can be problematic. First, audit records may not arrive in a timely fashion. Some IDSs use a separate computer to perform the analysis of audit records. So, it may take a significant amount of time to transfer the audit information from the monitored computer to the computer which performs the analysis. Second, the audit system itself may be vulnerable. Intruders have been known to be able to turn off the audit system or to modify the audit records to hide their intrusions. Finally, the audit records may not contain enough information to detect certain intrusions [22].

2.2.5 The architecture of an IDS

Intrusion detection systems are constructed from three components: a sensor, an analyzer and a user interface.

2.2.5.1 Sensors:

A sensor is the component that collect data such as network traffic, log files and system trace files. Once the data is collected it is then forwarded to the analyzer.

The sensor can simply transmit the captured data directly, but in general a pre-processing is perfumed before the transmission. This pre-processing could be, for example, a data filtering based on data fields pertinence which limits the amount of information to be analyzed subsequently. In addition, the sensor generally performs a formatting on the acquired data raw in order to present them to the analyzer using a data event format [21].

2.2.5.2 Analyzers:

An analyzer or a detection engine in IDS's is the part responsible of determining if there was an intrusion/anomaly among the data. After an intrusion is detected the analyzer’s output is either an alarm or action. The sensor and the analyzer can be a single system, as known as a probe, or they can be separated into individual components depending on how the IDS is constructed.For example, one analyzer might get traffic data from multiple sensors or the sensor might be embedded into- the analyzer. The user interface provides the means for the administrator to monitor the output of an analyzer and configure analyzer and sensor operations [21].

2.2.5.3 User interface:

The user interface provides the means for the administrator to monitor the output of an analyzer, a probe, and configure analyzer and sensor operation.

2.3 DDoS Defense mechanisms :

The seriousness of the DDoS problem and the increased frequency, sophistication and strength of attacks have led to the advent of numerous defense mechanisms. In this chapter, we classify the defense mechanisms against DDoS attacks using three criteria. The first criterion for classification is the location where the defense mechanism is implemented (i.e., Deployment location). The second criterion is the protocol level on which defense mechanism

works, and the last is the point of time when the DDoS defense mechanisms should act in. :response to a possible DDoS attack.

2.3.1 DDoS defense mechanisms based on location deployment

This classification is based on the location of the implemented defense mechanism. This can further be categorize as source based, destination based, network based and hybrid DDoS defense .

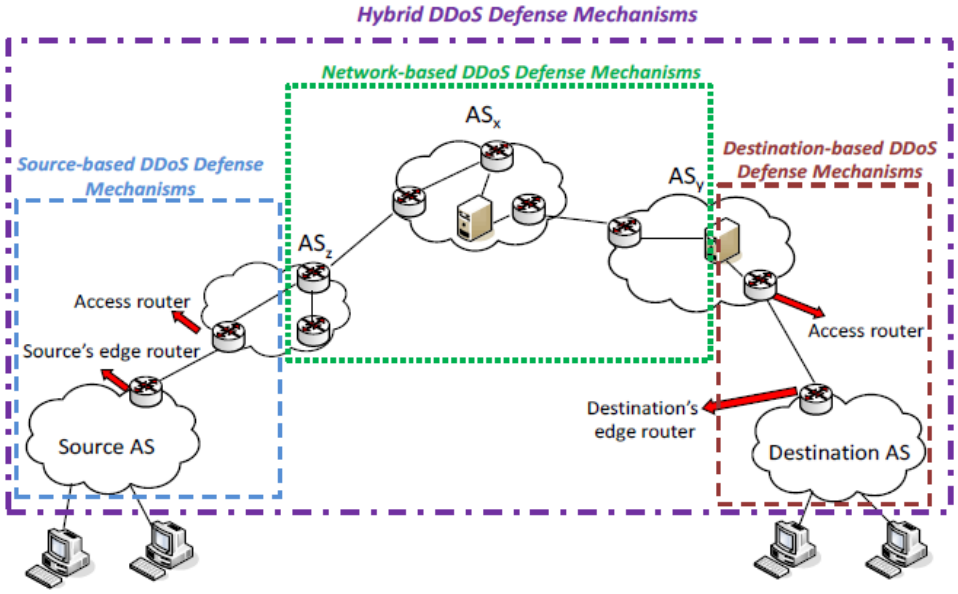


Figure 2.4: DDoS defense mechanisms based on their deployment location [31].

2.3.1.1 Source-based mechanisms

Here the mechanisms are deployed near the sources of attack. These mechanisms basically focus on restricting the network customers from generating DDoS attacks. There are various mechanisms that are source based, some major one's are:

A. D-WARD

Mirkovic et al[29]. Proposed a system called D-WARD that does DDoS attack detection at the source based on the idea that DDoS attacks should be stopped as close to the sources as possible.

D-WARD is installed at the edge routers of a network and monitors the traffic being sent to and from the hosts in its interior. If an asymmetry in the packet rates generated by an internal host is noticed, D-WARD rate limits the packet rate. The drawback of this approach is that there is a possibility of numerous false positives while detecting DDoS conditions near the source, because of the asymmetry that there might be in the packet rates for a short duration. Furthermore, some legitimate flows like real time UDP flows do exhibit asymmetry.

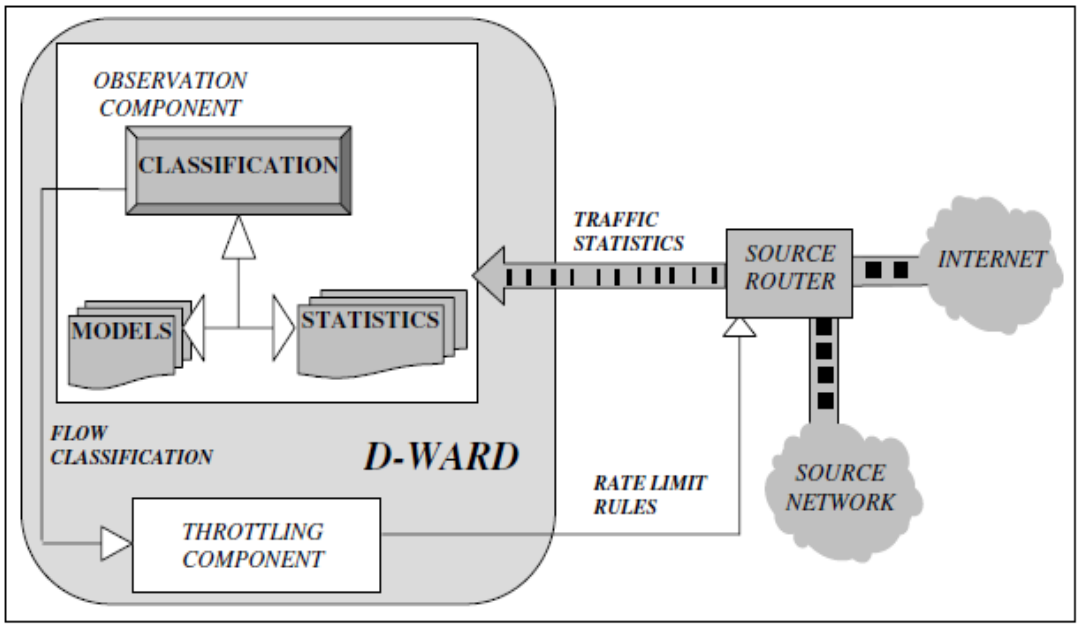


Figure 2.5: D-WARD architecture [29].

B. Ingress/Egress Filtering

The purpose of ingress/egress filtering is to allow traffic to enter or leave the network only if its source addresses are within the expected IP address range. Ingress filtering refers to filtering the traffic coming into a network, and egress filtering refers to filtering the traffic leaving the network[07] .

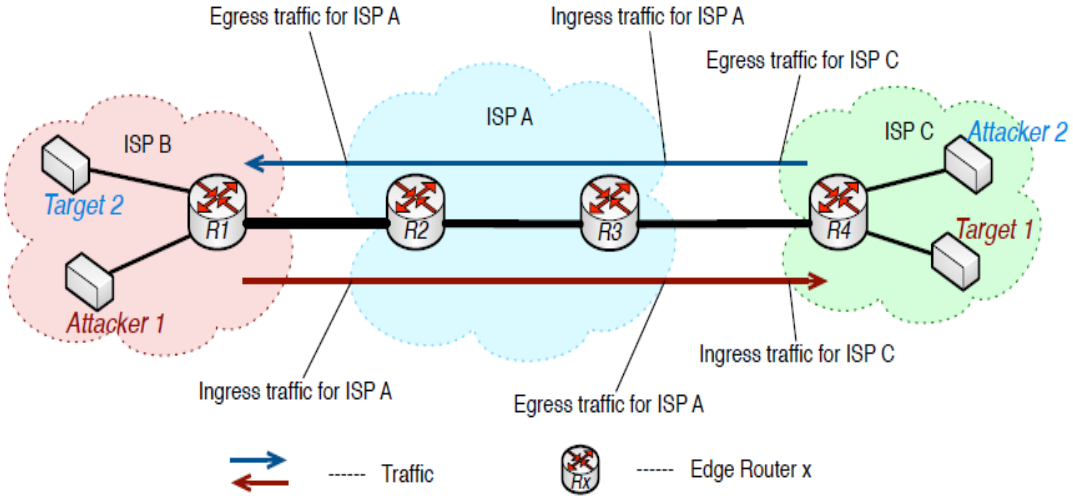


Figure 2.6: An example of ingress/egress filtering [07] .

The concepts of ingress filtering and egress filtering are illustrated in Figure 2.2 . In the figure, attacker 1 resides within the network 204.69.207.0/24, which is provided Internet connectivity by Internet service provider (ISP) A. An input traffic filter on the ingress (input) link of edge router R2, which provides connectivity to the attackers' network, restricts traffic to allow only traffic originating from source addresses within the 204.69.207.0/24 prefix, and prohibits an attacker from using "invalid" source addresses which reside outside of this prefix range. This is an ingress filtering. If edge router R1, instead of R2, provided the same filtering function, then that will be called egress filtering.

The key requirement for in ingress or egress filtering is to know the expected IP addresses at a particular port. For some networks with complicated topologies, it is not easy to obtain this knowledge. Additionally, ingress/egress filtering does not provide a strong deployment incentive to Internet service providers, and it is currently partially deployed. Hence, attackers can carefully choose a network without ingress/egress filtering to launch DoS attacks that use spoofed source ad-dresses. Moreover, ingress/egress filtering does not preclude an attacker using a forged source address of another host within the permitted prefix filter range [07] .

C. MULTOPS

Gil and Poletto [32] proposed MUltiLevel Tree for Online Packet Statistics (MULTOPS), a tree data structure that network devices, such as routers, can use to detect bandwidth flooding attacks. MULTOPS tree contains packet rate statistics for subnet prefixes at different aggregation levels, and it expands and contracts within a fixed memory budget.

A network device using MULTOPS detects ongoing bandwidth attacks by the significant, disproportional difference between packet rates going to and coming from the attack agent or the attack target.

When the packet rate to or from a subnet reaches a certain threshold, a new sub-node is created in a MULTOPS tree to keep track of more fine-grained packet rates. This process can go till per IP address packet rates are being maintained. Therefore, starting from a coarse granularity one can detect with increasingly finer accuracy, the exact attack source or destination addresses.

However, MULTOPS detection assumes that incoming and outgoing packet rates for a host is proportional, which may not be true for certain traffic types. Furthermore, MULTOPS fails to detect attacks that use randomly spoofed IP addresses to proportionally distribute attack traffic across multiple IP addresses. Lastly, the MULTOPS tree itself may be the target of a memory exhaustion DoS attack [32] .

2.3.1.2 Destination-based mechanisms

Under this category, mechanisms are deployed near the victim [33] . Historically, most of the systems for combating DDoS attacks have been designed to work on the victim side, since this side suffered the greatest impact of the attack. The victim has the greatest incentive to deploy a DDoS defense system, and maybe sacrifice some of its performance and resources for increased security [30]. Examples of these systems are:

A. IP Traceback mechanisms

The process of tracing back the forged IP packets to their true sources rather than the spoofed IP addresses that was used in the attack is called traceback. These mechanisms can be classified into two main categories [31]:

Packet marking mechanisms :

Usually routers in the path to the victim mark packets (i.e., add routers' identification to each packet) so that the victim can identify the path of attack traffic and distinguish it from legitimate traffic after the detection.

However, storing the entire path in the IP identification field of each packet needs certain coding schemes and these schemes sometimes are not able to assign each mark to a unique path; hence, false positive rates of these mechanisms are still high. In other words, legitimate packets could be treated as attack packets [31].

Link testing mechanisms :

In These mechanisms the traceback process usually starts from the router closest to the victim and iteratively tests its upstream links until it can be determined which link is used to carry the attacker's traffic (i.e., the traceback process is recursively repeated on the upstream router until the source is reached).

All of the traceback mechanisms have serious deployment and operational challenges. One of the fundamental deployment and operational challenges is ensuring a sufficient number of routers that support traceback before it is effective. Moreover, attackers can also generate traceback messages; consequently, some form of authentication of traceback messages is necessary. Furthermore, most of the traceback mechanisms have heavy computational, network or management overheads [31].

B. ICMP traceback message

Bellovin [34] proposed a new type of ICMP message called iTrace packet, to help receiver reconstruct the path that packets take through the Internet. Each router generates a traceback

message with a very low probability for each packet it forwards, and sends the message to the receiver. To avoid burdening the receiver with too much extra traffic, the probability of generating the traceback message should be adjustable, and should not be greater than 1/1000. If enough traceback messages are gathered at the receiver, the source of traffic can be found by constructing a chain of traceback messages. The advantage of this scheme is that it is simple and easy to implement. However, it needs digital signatures to protect the integrity of the information contained in a traceback message. Such digital signatures are expensive to generate and verify, and requires a key distribution infrastructure. Moreover, compared with other traceback schemes, this scheme creates extra traffic in the network .

C. Rate Limiting mechanisms

Rate limiting mechanisms limit the rate of packet arrivals which match the criteria for DDoS attacks, It is important that rate limiting mechanisms only limit the rate of malicious packets and do not harm legitimate flows. Furthermore, these rate limiting mechanisms should not incur a lot of extra overhead and they shouldn't become a source of denial of service attacks themselves. Rate limiting attacks can also be thought of as a less severe form of packet filtering. If it is known that the attack detection mechanism can come up with many false positives, it is better to go for rate limiting rather than packet filtering. Rate limiting can also be thought of as packet shaping using which all the internet sources are forced to respect some constraints imposed on their forwarding rate [35].

2.3.1.3 Network-based mechanisms

These mechanisms are deployed inside networks and mainly on the routers of the autonomous systems. Detecting attack traffic and creating a proper response to stop it at intermediate networks is an ideal goal of this category of defense mechanisms [31]. Some of the main network-based DDoS defense mechanisms are as follows:

A. Route-based packet filtering

Route-based packet filtering extends ingress filtering to the routers at the core of the Internet. The traffic on each link in the core of the Internet usually originates from a limited set of source addresses. Hence, if an unexpected source address appears in an IP packet on a link, then it is assumed that the source address has been spoofed, and hence the packet can be filtered. However, this mechanism is ineffective against DDoS attacks if attackers either use

genuine IP addresses instead of spoofed ones or spoof with carefully chosen source IP addresses that are not going to be filtered [31].

B. Detecting and filtering malicious routers

Routers are continuously targeted and compromised. They can be leveraged to empower DDoS attacks. A range of specialized anomaly detection protocols have been proposed to detect malicious routers involved in packet forwarding between routers. For instance, Watchers detects misbehaving routers that launch DDoS attacks by absorbing, discarding or misrouting packets. It uses the conservation of flow principle to examine flows between neighbors and endpoints.

Watchers can only detect compromised routers and it is vulnerable to misbehaving hosts. It also assumes that every router knows the topology of the network, which is not a feasible assumption for large networks [31].

2.3.1.4 Hybrid mechanisms

In most of the previously discussed categories of DDoS flooding defense mechanisms (source-based, destination-based, and network-based), there is no strong cooperation among the deployment points. Furthermore, detection and response is mostly done centrally either by each of the deployment points (e.g., source-based mechanisms) or by some responsible points within the group of deployment points (e.g., network-based mechanisms). Hence, we call these categories of DDoS defense mechanisms centralized. As opposed to centralized defense mechanisms, hybrid defense mechanisms are deployed at (or their components are distributed over) multiple locations such as source, destination or intermediate networks and there is usually cooperation among the deployment points. For instance, detection can be done at the victim side and the response can be initiated and distributed to other nodes by the victim. Some of the hybrid DDoS defense mechanisms are as follows [31]:

A. COSSACK

This mechanism is built on all of the border routers of the edge networks, with the core software system called watchdog. This mechanism is based on a number of assumptions. First, the border routers are assumed to have ingress/egress filtering mechanisms implemented. Second, border routers can prevent IP spoofing by employing ingress/egress filtering. The final critical set of assumptions Papadopoulos et al. consist of the existence of an attack signature, the capability of border routers to filter packets based on the signature,

and the connection availability between watchdogs . Therefore, with all the supports from different implemented components that they assume, the application layer watchdog could multicast attack notifications from the victim side to the source side and stop DDoS attack flows around the source employing the ingress/egress filtering mechanisms implemented on the border routers. COSSACK is, however, unable to handle attacks from legacy networks that do not deploy COSSACK defense mechanism [31].

B. DEFensive Cooperative Overlay Mesh (DEFCOM)

DEFCOM is a distributed framework to enable information and service exchange among all of the defense nodes. It attempts to shift from isolated defense architectures towards a distributed framework of heterogeneous defense nodes in which all the nodes collaborate and cooperate to achieve an effective defense. For instance, since attack detection is best done near the victim and response is most effective at the source of the attack, defense nodes should be specialized for different aspects of the defense; hence, different nodes in their proposed distributed architecture are responsible for their specialized defense ability and they must be able to communicate with others in order to successfully detect and respond to the attacks. Each node in their proposed framework must at least support the followings [31]:

- *Attack alerts* generated from the alert generators should be sent to the rest of the network.
- *Rate-Limit requests* should be sent upstream.
- *Resource requests* that each node issues should be sent to its downstream neighbors.
- *Traffic Classification* nodes must communicate with their downstream neighbors to ensure that the bulk of legitimate traffic will not be dropped.

DEFCOM is comprised of heterogeneous and distributed defense nodes organized into a P2P network where the nodes are communicating with each other to achieve cooperative defense. The P2P structure and topology construction of the DEFCOM defense nodes allows for the approximation of underlying topology; hence, when an attack alert is raised, it will be possible to discover victim-rooted traffic tree. Then, upstream and downstream relationships among peers is identified and proper rate limits to control the attack traffic is created and placed as close as possible to the sources of the attack. At the same time, classifier nodes will differentiate legitimate traffic from attack traffic.

Classifier nodes in the DEFCOM require an in-line deployment and their malfunction deters a wide deployment of classifier functionality in the network. Therefore, there will be no means to verify if the traffic which has been received by a node is legitimate or attack and rate

limiters severely rate-limit all the traffic coming from these sources. If it is assumed that a significant portion of networks in the Internet will be legacy networks, this results in a DoS attack to the legitimate clients from legacy networks during attacks; this is the main disadvantage of the DEFCOM framework [31].

2.3.2 DDoS defense mechanisms based on protocol

Under this category the defense mechanisms can be classified as the mechanisms to defend against the TCP/IP level DDOS attacks and mechanisms to defend against APPLICATION level DDOS attacks [31].

2.3.2.1 TCP level defense mechanisms

These types of mechanisms are basically deployed to defend against DDoS attacks where TCP protocol is exploited. Some of the common defenses are:

A. SYN flood detection

Wang et al [36]. proposed flooding detection system (FDS), a SYN flood detection method based on the protocol behavior of TCP SYN/FIN, SYN/RST pairs. It is assumed that FDS is deployed at the first-mile or last-mile edge routers. SYN flood detection in FDS is based on the fact that a normal TCP connection starts with a SYN and ends with a FIN or RST packet, whereas in SYN floods there will be more SYN packets than FIN and RST packets. FDS models the difference between the number of SYN packets and FIN/RST packets for normal traffic as a stationary ergodic random process, and uses a non-parametric Cumulative Sum (CUSUM) sequential change point detection algorithm to detect attack traffic. The drawback of FDS is that an attacker can avoid detection by sending low Time to Live value FIN or RST packets that will be dropped after passing the detection point .

B. Recycling the Oldest Half-Open TCB

Once the entire backlog is exhausted, some implementations allow incoming SYNs to overwrite the oldest half-open TCB entry. This works under the assumption that legitimate connections can be fully established in less time than the backlog can be filled by incoming attack SYNs. This can fail when the attacking packet rate is high and/or the backlog size is small, and is not a robust defense [37].

C. Reducing SYN-RECEIVED Timer

Another quickly implementable defense is shortening the timeout period between receiving a SYN and reaping the created TCB for lack of progress. Decreasing the timer that limits the lifetime of TCBs in SYN-RECEIVED is also flawed. While a shorter timer will keep bogus connection attempts from persisting for as long in the backlog, and thus free up space for legitimate connections sooner, it can prevent some fraction of legitimate connections from becoming fully established. This tactic is also ineffective because it only requires the attacker to increase the barrage frequency by a linearly proportional amount. This timer reduction is sometimes implemented as a response to crossing some threshold in the backlog occupancy, or some rate of SYN reception [37].

D. Firewalls and Proxies

Firewall-based tactics may also be used to defend end hosts from SYN flooding attacks. The basic concept is to offload the connection establishment procedures onto a firewall that screens connection attempts until they are completed and then proxies them back to protected end hosts. This moves the problem away from end hosts to become the firewall's or proxy's problem, and may introduce other problems related to altering TCP's expected end-to-end semantics. A common tactic used in these firewall and proxy products is to implement one of the end host based techniques discussed above, and screen incoming SYNs from the protected network until the connection is fully established. This is accomplished by spoofing the source addresses of several packets to the initiator and listener at various stages of the handshake [37].

2.3.2.2 IP level defense mechanisms

These defense mechanisms are used as countermeasure to IP-Level DDoS attacks. Following are some defense mechanisms.

A. Probabilistic Packet Marking

Probabilistic packet marking (PPM) was originally introduced by Savage et al . Who described efficient ways to encode partial route path information and include the traceback data in IP packets. It is an approach that can be applied during or after an attack, and it does not require any additional network traffic, router storage, or packet size increase. Even though it is not impossible to reconstruct an ordered network path using an unordered collection of router samples, it requires the victim to receive a large amount of packets. The advantage of

this approach is that no extra traffic is generated, since the extra information is bound to the packets. Furthermore, there is no interaction with ISPs and this mechanism can be used to trace attacks after an attack has completed [30].

On the other hand, there is a backward incompatibility as IP marking on the ID field conflicts with IPsec in which the Authentication Header encrypts the identification header. Moreover probabilistic ID-field marking requires modifications of Internet routing devices to generate such marks on the fly. The reconstruction of an attack path by the victim demands a high computation overhead. High false positives are generated when multiple attackers initiate DDoS.

This approach is not robust against a compromised router. Ioannidis and Bellovin argue that even though the attack path has been identified, it is not clear what are the next tasks that must follow [30].

B. Martian Address Filtering and Source Address Validation

Martian address filtering and source address validation are defined in the Requirements for IP Version 4 Routers. Martian address filtering specifies that a router should not forward any Martian packet (or a packet from Mars), where a Martian packet is a packet whose source or destination specifies an IP address designated by the Internet Assigned Numbers Authority (IANA) as reserved or special-use from or to which packets cannot actually originate or be delivered. Latest special-use IPv4 addresses are defined in IETF RFC 5735, and other examples of invalid IP addresses include IP addresses from as-yet-unallocated range (Bogon addresses) and a destination address of 255.255.255.255/32 etc [07] .

Source address validation specifies that a router should implement the ability to filter traffic based on a comparison of the source address of a packet and the for-warding table for a logical interface on which the packet was received. If this filtering is enabled, the router must silently discard a packet if the interface on which the packet was received is not the interface on which a packet would be forwarded to reach the address contained in the source address. In simpler terms, if a router wouldn’t route a packet containing this address through a particular interface, it should not believe the address if it appears as a source address in a packet read from this interface.

Martian address filtering eliminates the possibility of spoofing for a small set of addresses. Attacker can simply avoid spoofing any Martian addresses. Source address validation can eliminate majority of source address spoofing. However, with the number of asymmetric routes in the Internet, it is quite possible that the return path to a given packet’s source address

may not flow out the same interface as that packet arrived upon. Hence, using such technique to filter packets causes collateral damage to legitimate users' traffic [07] .

C. Hop-Count Filtering

Jin et al [38]. introduced filtering packets with spoofed IP addresses using a method called Hop-Counter Filtering (HCF). They argue that although an attacker can forge any field in the IP header, he or she cannot falsify the number of hops an IP packet takes to reach its destination. They propose a method to infer this hop-count information from Time to Live (TTL) value in the IP header. Using the TTL-based hop-count computation method, a victim builds a hop-count to source IP address mapping table. When a victim receives a packet, it computes a hop-count for its IP address and compares it to the hop-count stored in the mapping table to identify address-spoofed packets. HCF stays in alert state by default, in which it monitors the trend of hop-count changes without discarding packets. Upon detection of a flux of spoofed packets, HCF switches to action state to examine each packet and discard spoofed IP packets.

The advantage of HCF is that it requires deployment at the victim, which is much easier to deploy compared with network based filtering approaches. More-over, a potential victim has a much stronger incentive to deploy defense mechanisms that the intermediate network service providers. However, HCF suffers from high false positives and false negatives. HCF hop counting method relies on the initial TTL value for computing the hop-count, and initial TTL values for different operating systems (OSs) are different. HCF hop count method fails to work when the difference between initial TTL values for different OSs are less than the average hop-counts between Internet end hosts, and initial TTL value difference for some OSs are indeed less that the average hop-count (for examples, initial TTL values 30, 32, 60, and 64). Legitimate packets may be identified as spoofed due to inaccurate IP to hop-count mapping or delay in hop-count update. Even assuming the hop-count computation is precise, attackers can still spoof IP addresses with the same hop-count as their machines do. Lastly, as with any other victim-based filtering approaches, HCF cannot prevent DDoS attack traffic that is overwhelming the link coming into the machine that is enforcing the filtering [07] .

2.3.2.1 Application level defense mechanisms

These defense mechanisms are implemented to defend against application level attack. Because http level attack is more difficult to trace due to its legitimate behavior. Amount of traffic used to successfully carry out application level DDoS is much less than to carry out a

TCP or IP level DDoS attack. That is why the techniques used to detect TCP or IP level DDoS attacks are incapable to detect application level DDOS attacks [33] . Some of the common defenses are:

A. DDoS-Shield

This mechanism uses statistical methods to detect characteristics of HTTP sessions and employs rate-limiting as the primary defense mechanism.

DDoS-Shield consists of a suspicion assignment mechanism and a DDoS-resilient scheduler. The suspicion assignment mechanism assigns a continuous value as opposed to a binary measure, which have been assigned by previous mechanisms, to each client session. The DDoS-resilient scheduler acts as a rate-limiter and utilizes continues values, assigned by the suspicion assignment mechanism, to determine if and when to schedule a session's requests. However, it is not clear if a legitimate session is given another chance to receive the service if it is dropped by the DDoS-resilient scheduler [31] .

This model explores the vulnerability of systems to sophisticated application layer DDoS-attacks which are both protocol-compliant and non-intrusive. A framework is developed to classify these resource attacks as one of request flooding, asymmetric workload, repeated one-shot attacks or combinations thereof, on the basis of the application workload exhibit. Since these resource attacks are undetectable via 41 application layer techniques, they developed DDoS-Shield, a counter-mechanism which assigns a suspicion measure to a session in proportion to its deviation from legitimate behavior and uses a DDoS-resilient scheduler to decide whether and when the session is serviced. Using a web application hosted on an experimental test bed, they demonstrated the potency of these attacks as well as the efficacy of DDoS-Shield in mitigating them [39].

B. Speak-up

The way this mechanism tries to decrease the number of malicious requests is to encourage all the clients to automatically send higher volumes of traffic. The reasoning behind this approach is that attackers are already using most of their upload bandwidth so cannot react to the encouragement. On the other hand, good clients have spare upload bandwidth and will react to the encouragement by increasing their traffic volumes drastically. The goal of this mechanism is that the good clients crowd out the bad ones, thereby capturing a much larger fraction of the server's resources than before. Speak-up is applicable mainly against session flooding attacks and it is not applicable in case of request flooding attacks and asymmetric

attacks. It is also assumed that server will somehow detect whether or not it is under attack [31].

C. DOW (Defense and Offense Wall)

This mechanism employs the encouragement method which is presented in the Speak-up mechanism with an anomaly detection method based on K-means clustering to detect and filter session flooding attacks, request flooding attacks, and asymmetric attacks. Like Speak-up, DOW's encouragement model (currency model) encourages legitimate clients to increase their session rates so that they get a chance to be served. In other words, their detection model drops suspicious sessions while their currency model encourages more legitimate sessions. They believe that if these two models collaborate with each other, normal clients could gain higher service rates and lower delays in their response times. However, the main question that remains to be answered is with regards to the complexity and performance of this approach. This mechanism at this stage is too resource consuming to be implemented [31].

D. CALD

Application layer attacks utilize HTTP requests to overwhelm server. These kinds of attacks are more undetectable. It is even more difficult to detect these attacks when they occur during flash crowd event. CALD filters legitimate traffic and blocks the attack traffic. This model is concerned with three types of attacks namely, Repeated request DDoS, Recursive request DDoS, Repeated Workload DDoS. MyDoom , Code Red belongs to these kinds of DDoS attacks.

It has three main functions namely, Abnormal traffic detection, DDoS attack detection, Filter [39].

2.3.3 DDOS defense mechanisms based on time of action

Based on the time of action, defense mechanisms can be of following types:

2.3.3.1 Before the attack (attack prevention)

These attack mechanisms are basically deployed to prevent the attack from happening [33] . The prevention mechanisms can be deployed at the attack sources, intermediate networks, destinations or a combination of them. Most of the prevention mechanisms aim to fix security vulnerabilities (e.g., insecure protocols, weak authentication schemes, and vulnerable computer systems) that can be exploited to launch DDoS attacks.

There are some general prevention mechanisms that should be employed almost everywhere (e.g., servers, hosts, and intermediate networks) and in as many places as possible by both end hosts and service providers. Some of these general prevention mechanisms are as follows [31]:

A. Server-side specific security considerations

One of the main problems regarding application-level flooding attacks is that there is a lack of security mechanisms or security policies in place to address the servers vulnerabilities against application-level DDoS flooding attacks. Such security mechanisms or policies can protect servers from various attacks. For instance, Shekyan suggest the following policies as the best protections for the servers in handling the write readiness for active sockets:

- Do not accept connections with abnormally small advertised window sizes.
- Do not enable persistent connections and HTTP pipelining unless performance really benefits from it.
- Limit the absolute connection lifetime to some reasonable value.

Similar security mechanisms or security policies for different servers, such as, Web servers, application servers, database servers, etc. should be defined and should be used by considering current vulnerabilities of the servers against various application-level DDoS flooding attacks [31].

B. Secure Overlay Services

SOS is an architecture in which only packets coming from a small number of nodes, called servlets, are assumed to be legitimate client traffic that can reach the servlets through hash-based routing inside an overlay network. All other requests are filtered by the overlay. In order to gain access to the overlay network, a client has to authenticate itself with one of the replicated access points (SOAPs). SOS is a distributed system that offers excellent protection to the specified target at the cost of modifying client systems, thus it is not suitable for protection of public servers [30].

C. Changing IP address

Is another simple solution to a DDoS attack in order to invalidate the victim computers IP address by changing it with a new one. This is called moving target defense. Once the IP address change is completed all Internet routers will have been informed, and edge routers will drop the attacking packets. Although this action leaves the computer vulnerable because the attacker can launch the attack at the new IP address, this option is practical for local DDoS

attacks, which are based on IP addresses. On the other hand, attackers can render this technique a futile process by adding a domain name service tracing function to the DDoS attack tools [30].

2.3.3.2 During the attack (attack detection)

The next step in defending against DDoS attacks is attack detection, which happens during the attack. The detection mechanisms can also be deployed at sources, intermediate networks, destinations or a combination of them.

There are various mechanisms to detect DDoS attacks. Some of the detection mechanisms detect attack flows when the network links are congested to a certain level. Other mechanisms detect DDoS flooding attack traffic (not vulnerability attacks) when anomalous patterns are discovered in both the network/transport-level traffic and application-level traffic [31].

There are many IDPSs that are based on these detection mechanisms. They employ data mining and artificial intelligence techniques for more accurate detection. These mechanisms monitor some features/headers of the traffic flows at various locations and, points in time. Basically, they learn the normal behavior of either the network/transport-level or application-level traffic. Then, based on the information they have monitored and collected they can detect any changes on the traffic patterns and usage patterns of the resources. Based on the analysis , anomaly detection algorithms to detect a DDoS attacks can be classified depending on either the monitored parameters or statistical techniques used (e.g., change point detection , wavelet analysis) or granularity level of the analysis[31] . Some of attack detection mechanisms are as follows:

A. Scalable network monitoring system (NOMAD)

This system is able to detect network anomalies by making statistical analysis of IP packet header information. It can be used for detecting the anomalies of the local network traffic and does not support a method for creating the classifier for the high-bandwidth traffic aggregate from distributed sources [30].

B. MIB Traffic Variable Correlation

Cabrera et al[40]. investigates how to correlate Management Information Base (MIB) traffic variables that are exchanged through Simple Network Management Protocol (SNMP) to generate statistical sig-natures of attacking behavior, and how to use such signatures to detect DDoS attacks before they cause damage. In typical SNMP use, one or more

administrative computers called managers have the task of monitoring or managing a group of hosts or devices on a computer network. A SNMP agent that runs on a managed device periodically reports local MIB information to the network management system (NMS), a software that runs on the manager .

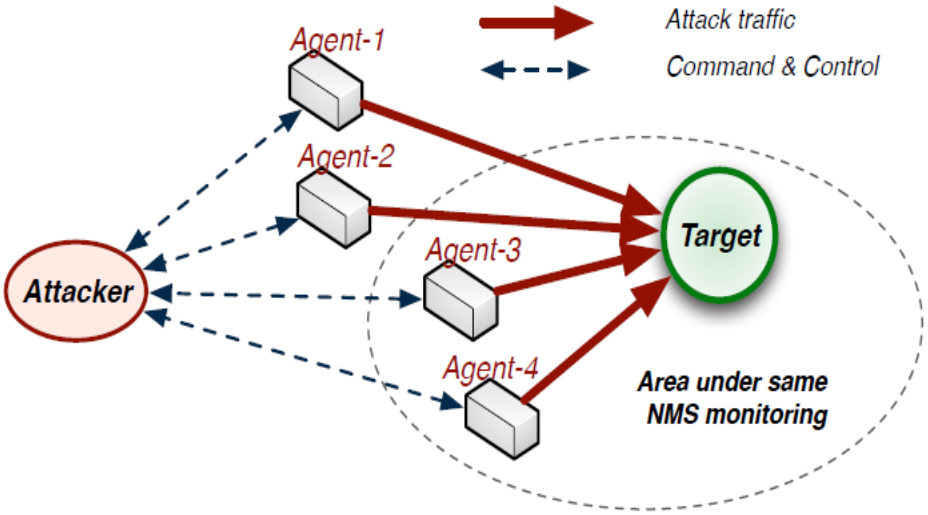


Figure 2.7: MIB variable correlation approach assumes that the attack target and some of the attacking agents are under the supervision of the same NMS [07].

Cabrera DDoS detection approach assumes that the attack target and some of the attack agents are under the supervision of the same NMS, as shown in Figure 2.9. To detect DDoS attacks, a set of rules or signatures are extracted off-line, and the extracted signatures are matched against monitored traffic online to determine the attack traffic. The signature extraction process includes three steps. First, a set of key MIB variables that characterize the occurrence of an attack are identified from attack target. Next, MIB variables at the agents that are causally related to the key MIB variables from step one are determined through statistical analysis. The assumption here is that any causal relationship between variables at potential attack agents and the key variables at the target is to be inferred as a link between the agent and the target machine. Following the determination of causal correlations between the key MIB variables at the target and key MIB variables at the attack agents, the last step is to determine particular features of the attack agent's key MIB variables that precede the attack. The extracted features that are indicative of an incoming attack is used as signatures [07].

Cabrera's detection approach may be applicable to mapping TCP, UDP, and ICMP packet statistical abnormalities to specific DDoS attacks. However, the evaluation of the approach in

a tightly controlled traffic environment did not show a high accuracy when detecting attacks, and it is unclear how it performs in realistic network environments. Furthermore, this approach assumes that the target and some attack agents are managed by the same NMS, which tremendously limits the applicability of this approach to all parts of the Internet [07].

C. Bro

Bro is a network intrusion detection system (NIDS) for detecting network intruders in real-time by passively monitoring a network link over which the intruder's traffic transits.

The main design goals of Bro are high-speed, large volume monitoring, real-time attack notification, separation of mechanisms from policy, and extensibility with regard to new attacks. Bro is conceptually divided into an event engine that reduces a stream of filtered packets to a stream of higher-level network events, and an interpreter for a specialized language that is used to express a site's security policy. The event engine maintains state for each connection based on source IP, source port, destination IP and destination port. Network administrators use the Bro language to write policy scripts [40].

Bro has a powerful signature language that allows the use of regular expressions, association of traffic going in both directions, and encoding of attacks that comprise multiple stages. The event driven attack detection approach of Bro greatly reduces the number of packets that need to be processed, and helps Bro towards achieving its high-speed traffic monitoring goal. Separation of event generation and event handling helps Bro achieve separation of security mechanisms from policy. Such separation also make it easy to add new protocol analyzers and new event handlers, hence provide good extensibility [40].

However, similar to many other network intrusion detection systems, Bro is susceptible to evasion attacks and DoS attacks on the monitor.

2.3.3.3 After the attack (attack source identification and response)

After a DDoS attack is detected, the defense system should identify the source of the attack and block the attack traffic. Today, most of the DDoS response mechanisms cannot completely prevent or stop DDoS attacks. Therefore, minimizing the attack impact and maximizing the availability of services is the main focus of all after the attack mechanisms. Moreover, law enforcement agencies must collaborate and cooperate with each other in order to gather and submit evidences that could be used to prosecute attackers.

It is necessary for all the Internet providers to understand that even if a particular provider would be able to secure its own assets, it does not secure itself against DDoS attacks as other

compromised hosts of other providers could still be used to launch attacks on it. Therefore, without collaborating with others to make sure their assets are also secured, defending against DDoS attack is almost impossible. There are two main categories for most of the after the attack mechanisms [31]:

A. Attack source identification

The first category of after the attack mechanisms is responsible to identify the source of the attack. For instance, an attacker uses host x to launch an attack by representing the spoofed source address of host y, IP traceback mechanism must find out the real source address of the attacker which is host x. This can be accomplished if there is a way of traversing all the routers from x to the victim in the reverse order or marking the legitimate paths or packets so that spoofed or illegitimate ones are identifiable [31].

B. Initiating a proper response

The second category of after the attack mechanisms is responsible for initiating a proper response to the attack. Most of the DDoS defense mechanisms apply throttling (rate limit) or packet filtering on upstream routers and hosts for the traffic coming from those identified attack flows (e.g., spoofed IP addresses) after identifying the source of the attack [31].

2.4 Conclusion

In this chapter, we have presented Intrusion Detection Systems then a comprehensive classification of various DDoS defense mechanisms along with their advantages and disadvantages based on where and when they detect and respond to DDoS attacks. In the next chapter we will present our proposed mechanism to protect web server against DoS/DDoS attacks.

CHAPTER 3

DoS/DDoS detection framework

CHAPTER 3

FRAMEWORK FOR DETECTION OF HTTP-BASED AND TCP-BASED DOS/DDOS ATTACKS

3.1 Introduction

We have seen in the second chapter that despite many researchers' efforts, no optimal solution that addresses all sorts of HTTP DoS/DDoS attacks is on offer. Therefore, our framework aims to propose an alternative solution which works on collaborative, multilayer's manner.

The innovative design of the framework handles all aspects of HTTP and TCP based DoS/DDoS attacks through the following three subsequent framework's layers:

- Firstly, an outer detector blocks attacking IP source if it is listed on the black list table.
- Secondly, the IP spoofed detector to validate whether the incoming request is launched by true IP source. Or a spoofed IP.
- Thirdly, the classifier module is to eliminate high rate /HTTP/TCP DDoS attacks.

3.2 Proposed Framework

In our work we propose an anomaly based system developed to detect DDoS attacks using a decision tree classifier, coupled with IP spoofed detector. The detection focus on two most attack target protocols TCP and HTTP.

Our framework is summarized in the diagram **Figure 3.1** that shows the steps followed to implement it.

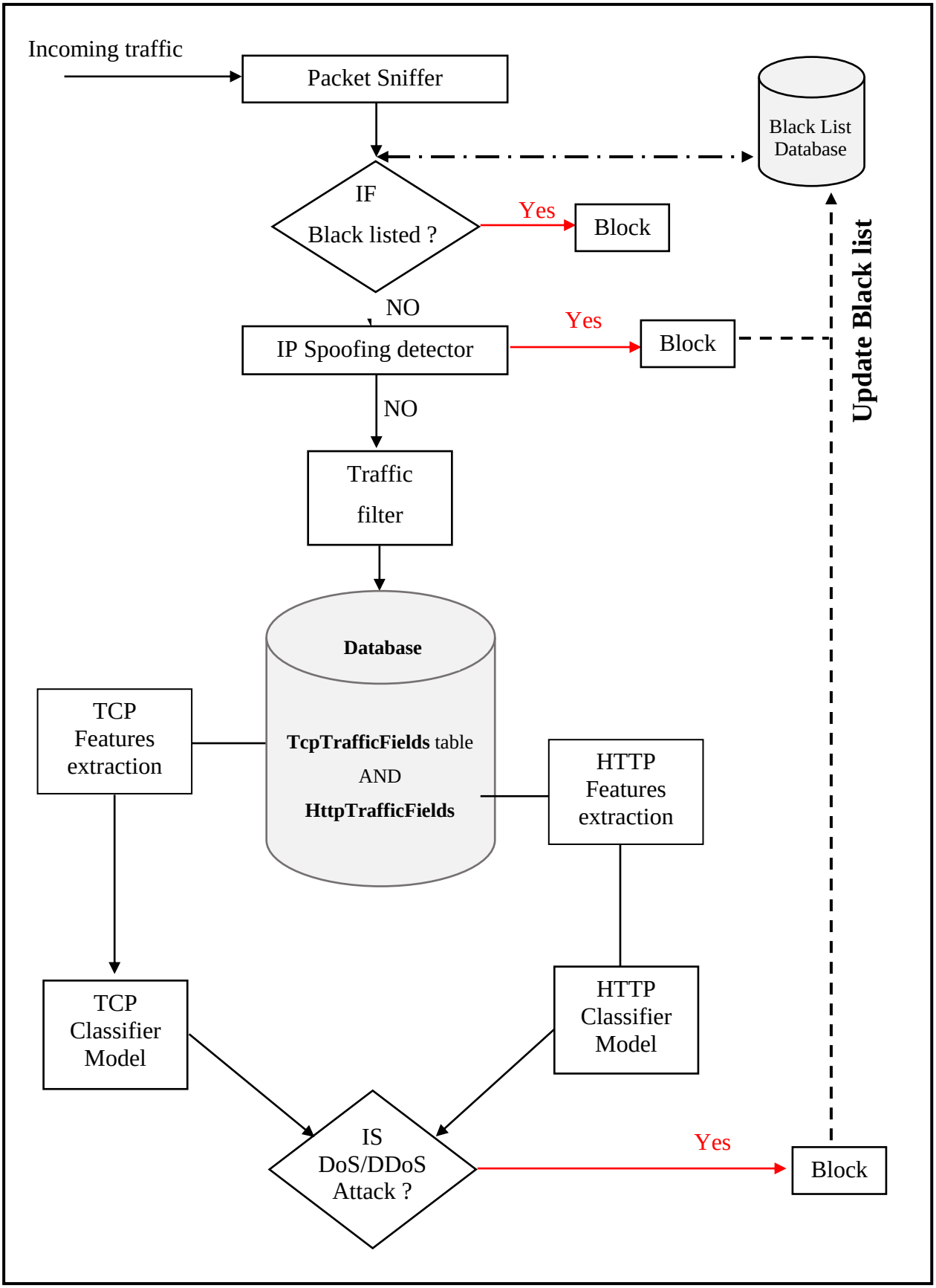


Figure 3.1: DoS/DDoS detection framework.

3.2.1 Packet sniffer

The information running through networks is a valuable source of evidence for our framework to fish out anomalous connections. The need to capture this information has lead to the development of packet sniffers.

Packet sniffers are software applications that use a network adapter card in promiscuous mode to capture all network packets. In our work we need packet sniffer to capture all TCP and HTTP packets form the network traffic.

3.2.2 IP spoofing detector

IP Spoofing is the act of creating an IP packet with a forged source IP address for the purpose of hiding the true source IP address, usually for the purpose of launching special types of distributed denial-of-service (DDoS attacks). By forging the source IP address of a packet; the individual sending it can direct the target IP address' machine to send its reply packet somewhere other than the real IP address of the source machine. Those wishing to launch DDoS attacks without large botnets can therefore send packets with random spoofed source IP addresses in order to both conceal their own identity and make the attack harder to block.

The methods used to detect spoofed IP can be generally classified as either passive or active [42]. We use the term active to mean the host must perform some network action to verify that the packet was sent from the claimed source. Passive methods require no such action, however an active method may be used to validate cases where the passive method indicates the packet was spoofed [43].

3.2.2.1 Spoofed Packets Detection Methods

A. Direct TTL probes

By sending a packet to the claimed host that will cause a reply we can check to see if the TTL in the reply is the same as the packet being checked. If they are of the same protocol, they generally have the same TTL. Because different protocols use different initial TTLs, when the probe packet is of a different protocol, we must infer the actual hop count. Only a few initial TTL values are commonly used. For TCP/UDP, 64 and 128 are most commonly seen. ICMP commonly uses 128 and 255 as the initial value. By subtracting the observed TTL from the supposed initial value we can estimate the number of hops. For example, for an ICMP packet with an observed TTL of 241, we get 255-241 or 14 as the estimated number of hops. If we are checking a TCP packet with an observed TTL of 50, we get 128-50=78 and

64-50=14. Because 14 is the expected value, we can assume the packet was not spoofed. If we knew the actual initial TTL for the host this would be more certain. Using information about a particular host is discussed in the section on passive methods, but it should be noted here that combined methods are feasible and will result in better detection [43].

If the attacker happened to be the same number of hops from the target as the spoofed source, this method would result in a false negative. Similarly, if the attacker knew the number of hops between the spoofed host and target, it may be possible to spoof the TTL field as well [43].

B. IP Identification Number

The sending host increments the Identification Number (ID) in the IP header with each packet sent. Because this is a value that is easily probed and changes in its value are predictable, we can use it to determine if a packet is spoofed. Unlike TTL values, IP ID numbers can be used to detect spoofed packets even when the attacker and the target are on the same subnet.

If we send probe packets to the claimed source and we receive a reply, the ID values should be near the value of questionable packets recently received from the host. Also, the ID values observed in the probe should be greater than the ID values in the questionable packets. If not the packets were likely not sent by the claimed source. If the host associated with the claimed source is very active, the ID values may change rapidly. To be effective, the probes must be done very close in time to receipt of the questionable packets. Some systems change initial ID values using more sophisticated method than increment by or some other constant value. To avoid violating RFC 79I, for fragmented packet assembly, ID numbers only need be sequential for the fragments of a particular datagram. This allows for more complicated ID number usage. Two common alternatives are to use a separate counter for each packet stream, or to use pseudo-random values. The implementation challenge is to prevent overlapping existing IP data streams. In cases where sophisticated ID number assignments are implemented, using ID numbers to detect spoofed packets may be problematic. However, if the attacker's computer does not use the same ID number creation method, probes to classify the ID numbering method used would readily show a difference. Also, some OSs exhibit quirky ID number assignment for certain protocols or services. For example, the Linux (kernel 2.4.0-2.4.4) ICMP echo request/reply packets always set the ID to zero. This defeats the simplest ID number probes, but it does facilitate more sophisticated probes [43].

C. TCP Flow Control Method

The TCP header includes a window size field. This is used to communicate the maximum amount of data the recipient can currently receive. This can also be interpreted as the maximum amount of data the sender can transmit without an acknowledgement from the recipient. This is the TCP flow control method. If the window size is set to zero, the sender should not send more data. If the packets we are receiving are spoofed, then the sender will never see the recipient's ACK-packets. This implies that the sender will not respond to flow control. If the recipient does not send any ACK-packets, the sender should stop after the initial window size is exhausted. If it does not, it is likely the packets are spoofed.

One way of implementing this check is to always send an initial window size that is extremely small. If packets received exceed this threshold, we can infer the packets are spoofed. Because spoofing replies with the correct sequence number to multiple TCP packets may be challenging, most spoofed TCP connections do not progress past the first ACK-packet. This implies that the best chance to detect spoofed packets requires it be done in the handshake. Fortunately the TCP handshake requires the host sending the initial SYN wait for the returned SYN-ACK prior to sending its first ACK packet. By setting the window size in the SYN-ACK to zero, we can determine if the sender is receiving (and responding to) our packets. If the sender sends an ACK-packet with any data, we know the true source is not responding to our packets, and was likely a spoofed packet [43].

3.2.2.2 Method used in our approach

In our approach we use TCP Flow Control Method to detect the spoofed packets, TCP Flow Control Method are considered a better choice for the following reasons:

- TCP Flow Control Method is successful when attacker is in the same subnet and in a different subnet. On the other hand, the Direct TTL Probes method is successful when attacker is in a different subnet and the IP Identification Number method is successful even if the attacker is in the same subnet.
- TCP Flow Control method Have the lowest percentage of falling in false negative or false positive.

3.2.3 Traffic filter

In this step of our approach, we extract essentials TCP and HTTP Fields and put them in database. The objective of monitoring protocol headers is to evolve a distinguishing pattern in

them, which will subsequently help detecting DDoS attacks on the protocol. In addition to TCP and HTTP Fields we extract IP address from the ip header for every packet received.

3.2.3.1 TCP Extracted Fields

No.	Field	Fields description
01	Source Port	identifies the sending port
03	SYN Flag	Synchronize sequence numbers. Only the first packet sent from each end should have this flag set.
02	ACK Flag	Indicates that the Acknowledgment field is significant. All packets after the initial SYN packet sent by the client should have this flag set.
04	RST Flag	Reset the connection
05	FIN Flag	No more data from sender
06	Window Size	which specifies the number of window size units (by default, bytes)that the sender of is currently willing to receive
07	Header length	specifies the size of the packet header (on bytes)
08	Payload length	specifies the size of the packet data (on bytes)

Table 3.1: TCP Extracted Fields.

3.2.3.2 HTTP Extracted Fields

No.	Field
01	Request Method
02	Response Status Code
03	Connection
04	Content-length
05	User-Agent
06	Referer
07	Accept-Encoding
08	Accept-language

Table 3.2:HTTP Extracted Fields.

Note :HTTP Fields in the table 3.2 are described in the first chapter.

3.2.3.3 Traffic database

The implemented database has two tables:

- **TcpTrafficFields** table: is needed to record TCP header fields of captured packets.
- **HttpTrafficFields** table: is needed to record HTTP header fields of captured HTTP requests and responses.

3.2.4 Features preprocessing

This process is divided into two phases: windowing phase and features extraction phase.

3.2.4.1 Windowing

Windowing essentially means splitting input traffic into traffic subsets, which fit into logical entities called windows. Analysis is done on every single window (traffic subset) to help arrive at a conclusion on the network state. Windowing may be done in two ways [45]:

- Time Windows: Windows based on time slices, occurring every n time units.
- Packet count Windows: Windows based on packet count, occurring for every n network packets.

The important factor that drives the choice among the two is the fact that analysis during learning should help analysis to be done real time. Packet count windows are considered a better choice of the two for the following reasons [45]:

- Packet count windows provide smaller reaction times during an DDoS attack situation, because of the fact that the system may have to wait wait for the time window to complete before deciding to flag an attack(or anomaly).
- Packet count windows may provide for more accurate modeling, since the number of possible events to be considered is bound to be limited, whereas the number of events to be considered may be large in time windows, since there is no control on how many packets could be received within a time window.

3.2.4.2 Features extraction

In order to detect attacks against web server, two types of features are extracted from the Traffic database , TCP features and HTTP features. We extracted 11 features from TCP fields table and 13 features from HTTP fields table, these features represent main characteristics of TCP and HTTP DDoS attacks.

A. TCP features

We preprocess the captured TCP header fields into windows of data where each window contains 11 features as shown in Table 3.3 .

NO.	Feature	Description
1	Number of packets	The number of packets per flow
2	Average number of port source used by host	number of distinct source port / number of distinct source IP.
3	Number of SYN	The number of packets with SYN flag
4	Number of ACK	The number of packets with ACK flag
5	Number of SYN_ACK	The number of packets with SYN+ACK flag
6	Number of RST	The number of packets with RST flag
7	Number of FIN	The number of packets with FIN flag
8	Header size	Average of packets header size
09	Payload size	Average of packets payload size
10	Duration	Duration of flow
11	Window size	Average window size of packets

Table 3.3: TCP Features.

B.HTTP features

We preprocess the HTTP requests and response headers fields captured into windows of data where each window contains 13 features as shown in Table 3.4 .

NO.	Feature	Description
1	Number of HTTP messages	The number of HTTP messages per flow
2	Number of HTTP requests	The number of HTTP requests per flow
3	Number of HTTP response	The number of HTTP response per flow
4	Number of GET	The number of HTTP GET requests per flow
5	Number of POST	The number of HTTP POST requests per flow
6	Content length	Average content length of HTTP requests
7	Number of keep-alive Connections	Number of requests with keep-alive Connections
8	Requests with referer Header Field	Number of requests where referer header field is defined

9	Requests with User-Agent Header Field	Number of requests where User-Agent header field is defined
10	Requests with Accept-Language Header Field	Number of requests where Accept-Language header field is defined
11	Requests with Accept-Encoding Header Field	Number of requests where Accept-Encoding header field is defined
12	Success Response	Number of requests successfully received, understood, and accepted.
13	Error Response	Number of requests not successfully received, understood, and accepted.

Table 3.4: HTTP Features.

3.2.5 TCP and HTTP traffic classifiers:

Classification is a supervised machine learning technique which assigns one of several predefined class labels to data objects. The classifier needs to be trained with labeled input examples, so that it could understand the characteristics of different classes, and then, it could be able to map new data items to different classes. There are many classification algorithms in data mining [44]. In our work we are used Decision Tree (DT) algorithm because of their well-known good performance in classification of vast amount of data.

3.2.5.1 Decision Tree (DT)

Decision Tree is a common method used in statistics, data mining and machine learning, where It is an efficient method for producing classifiers from data. It is considered as a tree-structured plan of a set of attributes to be tested in order to predict the output. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Moreover, it is a type of tree-diagram used in determining the optimum course of action, in situations having several possible alternatives with uncertain outcomes. A decision tree classifier is modeled in two phases: tree building and tree pruning. In tree building, the decision tree model is built by recursively splitting the training data set and assigning a class label to leaf by the most frequent class. Pruning a sub tree with a leaf or a branch if lower training error obtained. **Figure 3.2** presents decision tree algorithm [44].

Input:
- Data partition, **D**, which is a set of training tuples and their associated class labels;
- *attribute list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of *a splitting attribute* and, possibly, either *a split point* or *splitting subset.*

Output: A decision tree.
Method:
(1) create a node ***N***;
(2) **if** tuples in ***D*** are all of the same class, ***C*** **then**
(3) return ***N*** as a leaf node labeled with the class ***C***;
(4) **if** *attribute_list* is empty **then**
(5) return ***N*** as a leaf node labeled with the majority class in ***D***; // majority voting
(6) apply **Attribute selection method**(**D**, *attribute-list*) to **find** the "best" *splitting criterion*;
(7) label node ***N*** with *splitting_criterion*;
(8) **if** *splitting_attribute* is discrete-valued **and** multiway splits allowed **then** // not restricted to binary trees
(9) *attribute_list* ← *attribute_ list – splitting_attribute*; // remove splitting attribute
(10) **for each** outcome **j** of splitting criterion
// partition the tuples and grow subtrees for each partition
(11) let $\mathbf{D}_j$ be the set of data tuples in **D** satisfying outcome **j**; // a partition
(12) **if** $\mathbf{D}_j$ is empty **then**
(13) attach a leaf labeled with the majority class in **D** to node **N**;
(14) **else** attach the node returned by **Generate decision tree**($\mathbf{D}_j$, attribute list) to node **N**;
end for
(15) return **N**;

Figure 3.2: Basic structure of Decision Tree algorithm [44].

3.2.5.2 DATASET Collection

A new dataset was collected in our work because there is no existing data sets that contain a modern DDoS attack such as (SlowPOST, SlowLoris,HTTP Flood), and furthermore, other available data sets may include a great deal of duplicate and redundant records, and that may result in an ultimate unrealistic outcome. Our collected dataset contains four types of DDoS attack as follows: (HTTP Flood, HTTP POST, HTTP GET and TCP based attacks) without redundant and duplicate records.

3.2.5.3 Training phase

To train our TCP and HTTP classifiers we use a large amount of DDoS and normal traffic, this traffic converted into TCP features records and HTTP features records, each record

represent 200 of TCP packets traffic. Then we use a TCP and HTTP classifiers to detect DDoS attack traffic.

3.2.5.4 Evaluation metrics

A confusion matrix (also known as a contingency table or an error matrix) is a table layout that allows visualization of the performance of a supervised learning algorithm. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class. All correct guesses are located along the diagonal of the table such that errors can be easily visualized by any non-zero values outside the diagonal [46].

For a classifier to have good accuracy, ideally most of the tuples would be represented along the diagonal of the confusion matrix (CM). Given two classes, we can introduce the notion of positive tuples and negative tuples. True positives (TP) refer to the positive tuples that were correctly labeled by the classifier, while true negatives (TN) are the negative tuples that were correctly labeled by the classifier. False positives(FP) are the negative tuples that were incorrectly labeled. Similarly, false negatives(FN) are the positive tuples that were incorrectly labeled [46].

To evaluate the performance of the classifiers applied in our work, we use primary performance indicators based on confusion matrix shown in Table 3.5.

		Predicted	
		Normal	Attack
Actual	Normal	TP	FP
	Attack	FN	TN

Table 3.5: confusion matrix[46].

.**True Positive (TP):** the number of benign packets correctly classified as benign.

$$TP\ Rate = \frac{TP}{TP + FN} \qquad (7)$$

False Positive (FP): the number of normal traffic falsely classified as malicious.

$$FP\ Rate = \frac{FP}{FP + TN} \qquad (8)$$

False Negative (FN): it occurs when the malicious traffic is classified as normal traffic.

$$FN\ Rate = \frac{FN}{FN + TP} \qquad (9)$$

.**True Negative (TN):** the number of the malicious packets correctly classified as malicious

$$TN\ Rate = \frac{TN}{TN + FP} \qquad (10)$$

Accuracy: refer the percentage of test set tuples that are correctly classified by the classifier.

$$Accuracy\ = \frac{|TP| + |TN|}{|TP| + |FP| + |TN| + |FN|} \qquad (11)$$

Classification error: refer to relative number of misclassified, and the rate was recorded on the training and testing data sets.

$$Classification\ error = \frac{FP + FN}{TP + TN} = 100 - Accuracy \qquad (12)$$

3.3 Conclusion

In this chapter, we presented a qualitative description of our DDoS detection approach and explained its components one by one. In the next chapter we will give our framework experimentations and discuss the obtained experimental results .

CHAPTER 4

Experimentation and Discussions

CHAPTER 4

EXPERIMENTATION AND DISCUSSIONS

4.1 Introduction

This chapter describes experimental results and performance of our DoS/DDoS detection framework, which are described in the previous chapter.

First, we introduce the various tools of development of our detection framework, then we describe how our dataset are collected, naturally this data are used for training and testing our classifiers modules. Finally, we show the results of multiple scenarios of experiments

4.2 Development environment and tools

Special tools and programs were used to realize this work :

NetBeans IDE: (version 8.0.1) A NetBeans module is a set of Java classes written to interact with the NetBeans APIs, for extending the IDE or for creating your own application on the NetBeans Platform.

JPCAP: Jpcap is an open source library for capturing and sending network packets for Java applications that is based on libpcap/WinPcap, implemented in programming languages C and Java. Using Jpcap, we can develop Java applications to capture packets from a network interface and analyze them. Jpcap can capture Ethernet, IPv4, IPv6, ARP/RARP, TCP, UDP, and ICMPv4 packets. Jpcap has been tested on Microsoft Windows (98/2000/XP/Vista), Linux (Fedora, Ubuntu), Mac OS X (Darwin), FreeBSD, and Solaris [41].

Jpcap is usually a group of java classes and interfaces that hides the low-level details of network packet capture by abstracting many network packet types and protocols into Java classes. Internally, Jpcap implements bindings to the libpcap system library through JNI (the Java Native Interface). Jpcap offers the following facilities to:

- Capture raw packets live from the wire.

- Save captured packets to an offline file, and read captured packets from an offline file.
- Automatically detect packet types and generate corresponding Java objects (for Ethernet, IPv4, IPv6, ARP/RARP, TCP, UDP, and ICMPv4 packets).
- Filter the packets based on user-specified rules before dispatching them to the application.
- Send raw packets to the network.

Jpcap enable us to develop many varieties of network applications including network and protocol analysers, network monitors, traffic loggers, traffic generators, user level bridges and routers, network intrusion detection systems (NIDS), network scanners, security tools etc. Jpcap captures and sends packets independently from the host protocols (e.g., TCP/IP). Consequently it does not block, filter or manipulate the traffic generated by other programs on the same machine. It just "sniffs" the packets that transit on the wire [41].

MySQL database: The MySQL is employed as being the relational database management system. It really used in our work to implement a relational database that stores essentials headers fields of TCP packets and HTTP Messages captured by the Jpcap sniffer. The implemented MySQL database has two database's tables structured like that :

A) HTTP traffic table structure:

Column	**Type**	**Null**	**Fault**
request_method	varchar(20)	Oui	*NULL*
Response_code	int(11)	Oui	0
connection	varchar(30)	Oui	*NULL*
content_length	int(11)	Oui	0
user_agent	text	Oui	*NULL*
referer	text	Oui	*NULL*
accept_encoding	text	Oui	*NULL*
accept_language	text	Oui	*NULL*
ipsrc	varchar(20)	Non	

Table 4.1: HTTP traffic table structure.

B) TCP traffic table structure:

Column	Type	Null	Fault
src_port	int(11)	Non	
SYN	varchar(10)	Non	
ACK	varchar(10)	Non	
FIN	varchar(10)	Non	
RST	varchar(10)	Non	
Window_size	int(11)	Non	
Header_length	int(11)	Non	
payload_length	int(11)	Non	
current_time	timestamp	Non	CURRENT_TIMESTAMP

Table 4.2: TCP traffic table structure.

WEKA: Weka (Waikato Environment for Knowledge Analysis) is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from our own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

WEKA has implementations of numerous classification and prediction algorithms. The basic ideas behind using all of these are similar. In our work, we use a Decision Tree algorithm to build our classification models.

4.3 Experiment results and Discussion

In this section, we show performance of our DDoS detection framework, exactly by testing our TCP and HTTP classifier modules with big amount of data.

4.3.1 Dataset Generation:

As we mentioned in the previous chapter, a new dataset was collected to train and test our own classifiers.

To collect this dataset we used two machines: Client machine and server machine.

Machines	Characteristics	Roles
Client machine	*CPU Capacity:* Pentium® dual-core 2.10 GHz. *Operating System:* Microsoft Windows 7 professional (32 bits). *Memory Capacity:* 4 GB RAM.	1) Navigate to the websites hosted on the http server using Webserver Stress tool. 2) Lunching DDoS attacks.
Server machine (HTTP server)	*CPU Capacity:* Intel(R) Core(TM) i5 CPU 2.40GHz. *Operating System:* Microsoft Windows 8.1 professional (64 bits). *Memory Capacity:* 4 GB RAM.	1) Play the role of HTTP server using WampServer. 2) capturing and converting traffic into dataset records contains essential features used to train classifiers .

Table 4.3: Essential characteristics and main roles of client and server machines.

4.3.1.1 Normal traffic generator:

Web server Stress Tool simulates any number of users accessing to a website at the same time. By simulating the HTTP requests generated by many simultaneous users for testing a websites and webservers.

In our case we needed the Web server Stress Tool to simulate many users accessing to the websites hosted in our HTTP server over few days, during that period the network traffic was captured and saved on MySQL database.

4.3.1.2 Malicious traffic generators:

we launched DDoS attacks tools (such as SlowLoris, HOIC, Slow-Post and HULK which was described in the first chapter) against our HTTP server. During that period, the network traffic was captured and saved on MySQL database.

Both of captured normal and DDoS traffics are converted into traffic records which represents datasets that are used for training and testing our TCP and HTTP classifiers.

TCP dataset collected are used for training and testing TCP classifier. and HTTP dataset used for training and testing HTTP classifier. Training dataset is divided into 60% of the entire dataset while the remaining 40% used for testing classifiers.

Category of class	Number of packets	Number of records	Training records	Testing records
Normal	862 000	4 310	2 626	1 684
HOIC	413 800	2 069	1 219	850
SlowLoris	127 600	638	365	273
SlowPost	39 000	195	117	78
SYN FLOOD	208 400	1 042	626	416
LOIC-TCP	235 200	1 176	715	461
Total of HTTP based DDoS attacks	580 400	2 902	1 701	1 201
Total of TCP based DDoS attacks	443 600	2 218	1 341	877
Total	1 886 000	9 430	5 668	3762

Table 4.4: global dataset.

4.3.2 Experimental Results

To show the performance of our DoS/DDoS detection framework , we use traffic of two types of attacks :HTTP based DoS/DDoS attacks and TCP based DoS/DDoS attacks.

4.3.2.1 HTTP based attacks scenarios

We used three different scenarios to test the performance of our classifiers models against HTTP based attacks.

A) First test scenario against HTTP-DDoS:

In the first test scenario, we divide our data into two classes which are normal class and HTTP-DDoS class. Tables 4.5 and 4.6 shows classification results using the confusion matrix in term of accuracy.

- *HTTP Classifier results:*

		Predicted	
		Normal	DDoS
Actual	Normal	1676	8
	DDoS	5	1196

Table 4.5: Confusion Matrix for HTTP classifier (Normal and HTTP-DDoS classes).

Accuracy detection= 99.55 % (Correctly Classified)

Classification error = 00.45 % (Incorrectly Classified)

- *TCP Classifier results:*

		Predicted	
		Normal	DDoS
Actual	Normal	1651	33
	DDoS	43	1158

Table 4.6: Confusion Matrix for TCP classifier (Normal and HTTP-DDoS classes).

Accuracy detection= 97,37 % (Correctly Classified)

Classification error = 02,63 % (Incorrectly Classified)

B) Second test scenario against HTTP-DDoS

In the second test scenario, we divide our data into four classes which are normal class, SlowLoris class, SlowPost class and HOIC class .tables 4.7 and 4.8 show classification results using the confusion matrix in term of accuracy.

- *HTTP Classifier results:*

		Predicted			
		Normal	Hoic	Slowloris	Slowpost
Actual	Normal	1677	1	6	0
	Hoic	1	849	0	0
	Slowloris	4	0	259	10
	Slowpost	3	0	29	46

Table 4.7 :Confusion Matrix for HTTP classifier (four classes of traffic).

Accuracy detection= 98.13 % (Correctly Classified)

Classification error = 01.87 % (Incorrectly Classified)

TCP Classifier results:

		Predicted			
		Normal	Hoic	Slowloris	Slowpost
Actual	Normal	1629	55	0	0
	Hoic	62	700	102	0
	Slowloris	1	122	135	1
	Slowpost	0	2	0	76

Table 4.8 : Confusion Matrix for TCP classifier (four classes of trafic).

Accuracy detection= 88.04 % (Correctly Classified)

Classification error: 11.96 % (Incorrectly Classified)

C) *Third test scenario against HTTP-DDoS*

In the third test scenario, we add 1305 unknown records (without training) to the DDoS classes, these records represent HTTP-LOIC DDoS attack. tables 4.9 and 4.10 show classification results using the confusion matrix in term of accuracy

- *HTTP Classifier results:*

		Predicted	
		Normal	DDoS
Actual	Normal	1676	8
	DDoS	13	2493

Table 4.9: Confusion Matrix for HTTP classifier (additional type of traffic).

Accuracy detection= 99.5 % (Correctly Classified)

Classification error = 00.5 % (Incorrectly Classified)

- *TCP Classifier results:*

		Predicted	
		Normal	DDoS
Actual	Normal	1636	48
	DDoS	52	1199

Table 4.10: Confusion Matrix for TCP classifier (additional type of traffic).

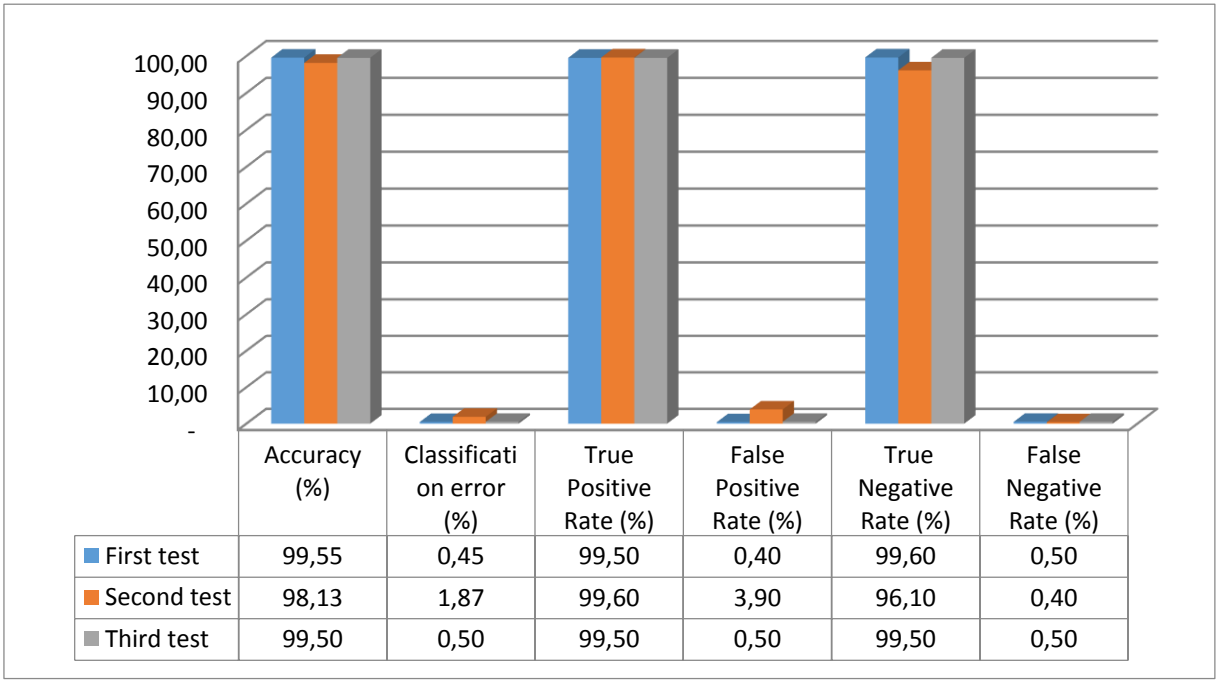
Accuracy detection = 96.60 % (Correctly Classified)

Classification error = 03.40 % (Incorrectly Classified)

4.3.2.2 HTTP-DDoS Results discussion

Figure 4.1 and 4.2 shows a summary for experiments in three HTTP-DDoS test scenarios for each classifier.

A) HTTP classifier Results discussion



	Accuracy (%)	Classification error (%)	True Positive Rate (%)	False Positive Rate (%)	True Negative Rate (%)	False Negative Rate (%)
First test	99,55	0,45	99,50	0,40	99,60	0,50
Second test	98,13	1,87	99,60	3,90	96,10	0,40
Third test	99,50	0,50	99,50	0,50	99,50	0,50

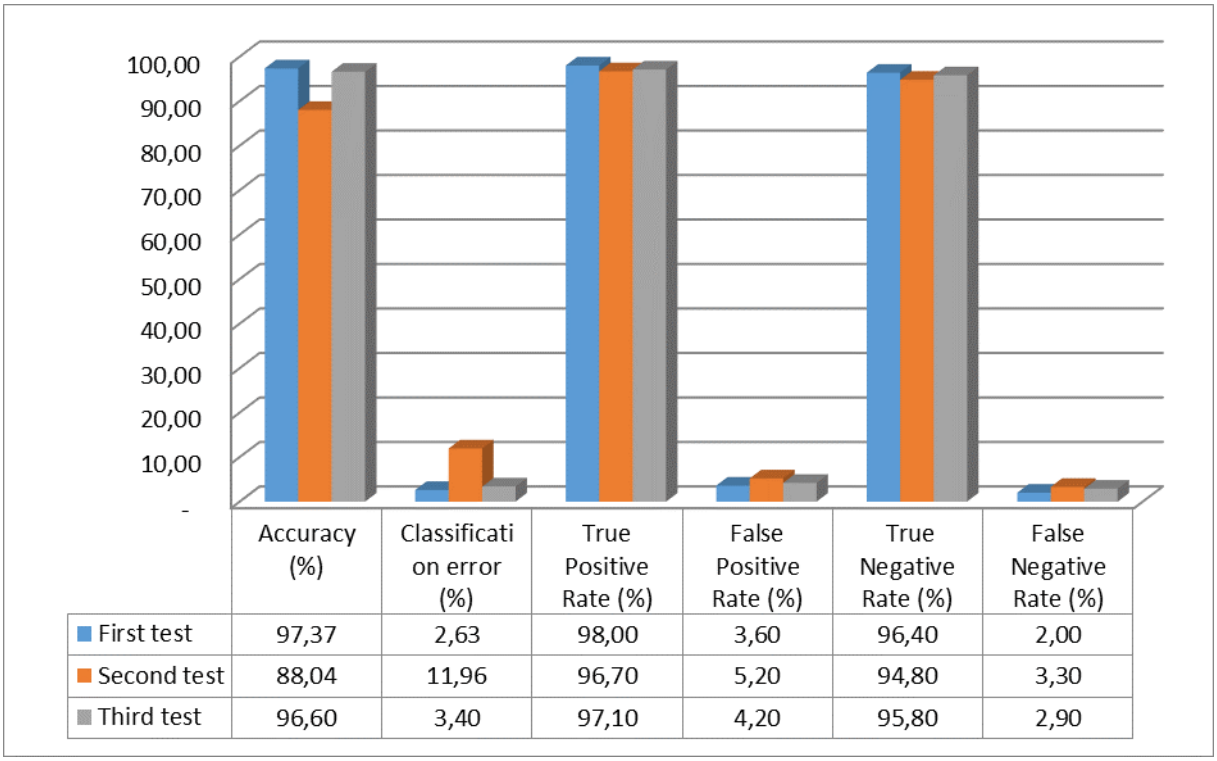
Figure 4.1: Summary for experiments in three HTTP-DDoS test scenarios (HTTP classifier).

It can be observed that our HTTP classifier model, can detect perfectly the HTTP-DDoS attacks with accuracy = 99,5% even after adding unknown malicious traffic ,but it can't

distinct carefully between used tools because SlowPost attack is very similar to Slowloris , As we mentioned in the first chapter.

These results indicate that our HTTP classifier model can distinct carefully between normal and malicious traffic.

B) TCP classifier Results discussion



	Accuracy (%)	Classification error (%)	True Positive Rate (%)	False Positive Rate (%)	True Negative Rate (%)	False Negative Rate (%)
First test	97,37	2,63	98,00	3,60	96,40	2,00
Second test	88,04	11,96	96,70	5,20	94,80	3,30
Third test	96,60	3,40	97,10	4,20	95,80	2,90

Figure 4.2: Summary for experiments in three HTTP-DDoS test scenarios (TCP classifier)

It can be observed that our TCP classifier model, can detect the HTTP-DDoS attack traffic with accuracy= 97,37 % and 96,6 % after adding unknown malicious traffic , but it can't distinct between HTTP-DDoS attacks types because these attacks have a common TCP features .

After these experiments, we observed that the HTTP classifier's performance is better than TCP classifier and that is because the HTTP-DDoS attacks are focuses on HTTP features .

4.3.2.3 TCP based attacks scenarios

A) First test scenario *against TCP-DDoS*

In the first TCP test scenario, we divide our data into two classes which are normal class and SYN FLOOD class. Tables 4.11 show classification results using the confusion matrix in term of accuracy.

		Predicted	
		Normal	DDoS(SYN FLOOD and TCP-LOIC)
Actual	Normal	1734	1
	DDoS(SYN FLOOD and TCP-LOIC)	5	872

Table 4.11 : Confusion Matrix for TCP classifier (normal and TCP-DDoS classes).

Accuracy detection= 99.77 % (Correctly Classified)

Classification error = 00.23 % (Incorrectly Classified)

B) Second test scenario *against TCP-DDoS*

In the second TCP based test scenario, we divide our data into three classes which are normal class, SYN FLOOD class and LOIC-TCP class .table 4.12 show classification results using the confusion matrix in term of accuracy.

		Predicted		
		Normal	SYN FLOOD	TCP-LOIC
Actual	Normal	1772	5	3
	SYN FLOOD	2	414	0
	TCP-LOIC	1	0	460

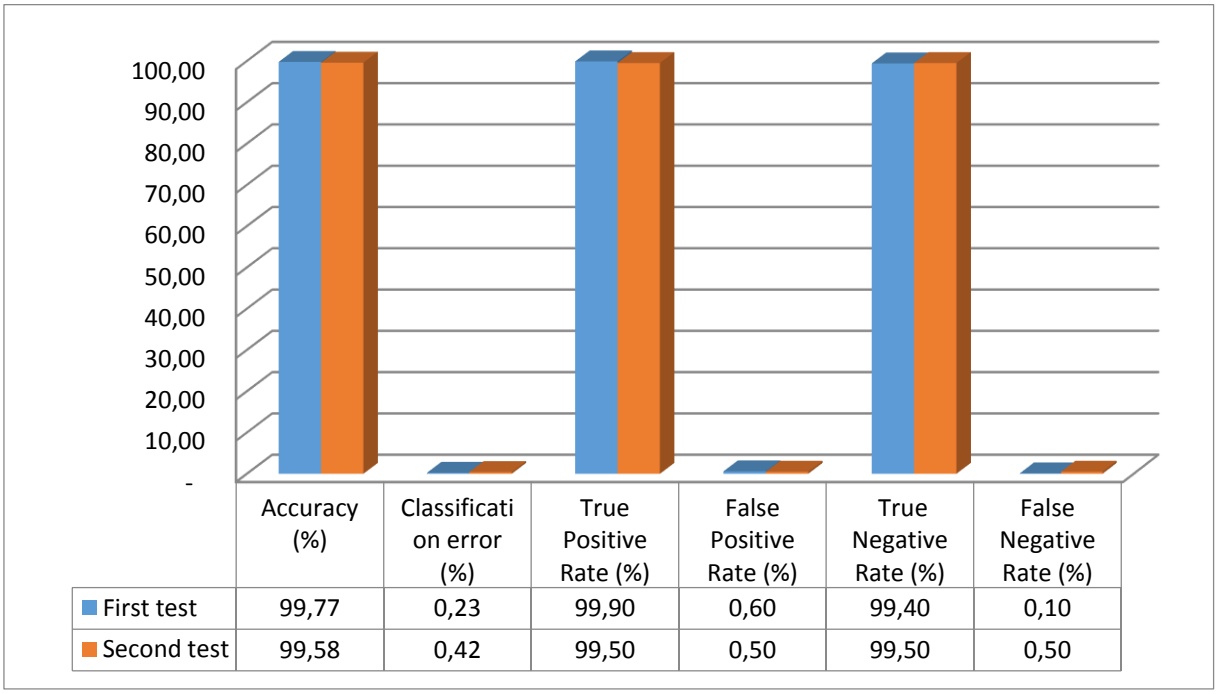
Table 4.12: Confusion Matrix for TCP classifier (normal , SYN flood and LOIC classes).

Accuracy detection= 99,58 % (Correctly Classified)

Classification error = 99,42 % (Incorrectly Classified)

4.3.2.4 TCP-DDoS Results discussion

Figure 4.3 show a summary for experiments in two TCP-DDoS test scenarios.



	Accuracy (%)	Classification error (%)	True Positive Rate (%)	False Positive Rate (%)	True Negative Rate (%)	False Negative Rate (%)
First test	99,77	0,23	99,90	0,60	99,40	0,10
Second test	99,58	0,42	99,50	0,50	99,50	0,50

Figure 4.3: Summary for experiments in two TCP-DDoS test scenarios (TCP classifier).

It can be observed that our TCP classifier model, can detect the TCP-DDoS attacks traffic with accuracy= 99,77 % and 99,58 % with two different classes of TCP-DDoS attacks traffic. These results indicate that our TCP classifier model can distinct carefully between normal and malicious traffic.

4.4 Conclusion

In this chapter we described the implementation and the experimentation of our proposed DDoS detection approach against web server.

The experimental study clearly shows the performance of our framework in detection of DDoS attack, with three different scenarios:

At the first we test our classifiers to recognize HTTP DoS attack through tree scenarios:

- detection of HTTP based DDoS attack
- detection of known HTTP based DDoS attack types
- detection of unknown HTTP based DDoS attack types

Then we test our classifiers to recognize the TCP DoS attack with two scenarios:

- detection of TCP based DDoS attack
- Detection of known TCP based DDoS attack types.

For the all cases of the two experiments, our classifiers prove their performance with an accuracy of 99%.

General Conclusion

Undoubtedly, DDoS attacks present a serious problem in the Internet and challenge its rate of growth and wide acceptance by the general public, skeptical government and businesses. As the use of internet increasing, the need for more efficient DDoS detection system becomes critical.

The great complexity of the DoS/DDoS problem suggests that its solution will require the use of multiple defenses .Our implemented framework proposes an alternative solution which work on collaborative, multilayers manner to detect HTTP and TCP based DoS/DDoS attacks. This solution starts by a simple traffic filter, which detect the black listed IP source, then pass to the second medium filter to detect the IPs spoofed, finally the traffic pass into two parallel filters: TCP filter and HTTP filter, for each detected anomaly, the traffic will be blocked.

In this work, we collected a new dataset that includes modern types of DoS/DDoS attack, which were not been used in previous researchs. The collected data has been recorded for different types of attack that target the Application and transport layers. Random Tree classifier algorithm was applied on the collected dataset to classify the normal traffic and DDoS types of attack namely: SlowPost, Slowloris, HOIC, LOIC-HTTP, LOIC-TCP, SynFlood.

To evaluate our DoS/DDoS framework we split our dataset into four type of traffic, which are HTTP-Normal, TCP-Normal, HTTP-DDoS and TCP-DDoS traffics. The both of TCP-Normal and TCP-DDoS traffics are used to test the performance of TCP classifier. Simultaneously, HTTP -Normal and HTTP-DDoS traffics are used to test the performance of TCP and HTTP classifiers. The evaluation proves that the proposed method can successfully identify DDoS attacks with very high detection rates.

As perspective we propose to integrate our framework with snort IDS to generate snort rules for each reported anomaly.

Bibliography

[01] R. Fielding, J. Gettys , J. Mogul , H. Frystyk, and T. Berners-Lee, "Hypertext Transfer Protocol --HTTP/1.1", RFC 2616, June 1999.

[02] Dafydd Stuttard, Marcus Pinto, The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws, Second Edition, John Wiley & Sons, Inc, Indianapolis, Indiana , October 2011.

[03] HTTP tutorial, http://www.tutorialspoint.com/http/http_pdf_version.htm ,visited 24/12/2015.

[04] Programming notes , HTTP (HyperText Transfer Protocol), https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html - visited 29/01/2016 .

[05] Applicure , Prevent Denial of Service (DoS) Attacks , http://www.applicure.com/solutions/prevent-denial-of-service-attacks , visited – 01-02-2016.

[06] L. Stein, J. Stewart , The World Wide Web Security FAQ , https://www.w3.org/Security/faq/wwwsf6.html - visited 04/02/2016.

[07] M. ABLIZ , Internet Denial of Service Attacks and Defense Mechanisms, Department of Computer Science, University of Pittsburgh, 2011.

[08] CERT , Denial of Service Attacks , 1997 https://www.cert.org/information-for/denial_of_service.cfm? -visited 06/02/2016

[09] Anne Carasik-Henmi, W. Shinder, Dr. Thomas, Robert J. Shimonski , Debra Littlejohn , Best Damn Firewall Book Period , Syngress ,2011 .

[10] Cisco, A Cisco Guide to Defending Against Distributed Denial of Service Attacks , http://www.cisco.com/c/en/us/about/security-center/guide-ddos-defense.html -visited 06/02/2016

[11] Jema D. Ndibwile, A. Govardhan , K. Okada, Y. Kadobayashi , Web Server Protection against Application Layer DDoS Attacks using Machine Learning and Traffic Authentication , IEEE 39th Annual International Computers, Software & Applications Conference , vol: 3, 2015, pp. 261-267.

[12] Gulshan Kumar , Understanding Denial of Service (Dos) Attacks Using OSI Reference Model , International Journal of Education and Science Research, vol: 1, 2004, pp. 10-17.

[13] Mohammed A. Saleh , Azizah A. Manaf , A Novel Protective Framework for Defeating HTTP-Based Denial of Service and Distributed Denial of Service Attacks , The Scientific World Journal,vol:2015 ,2015.

[14] M. Aiello, E. Cambiaso, M. Mongelli, G. Papaleo , An On-Line Intrusion Detection Approach to Identify Low-Rate DoS Attacks , 2014 International Carnahan Conference on. IEEE, 2014, pp. 1-6.

[15] A. Pras, A. Sperotto, Giovane C. M. Moura,I. Drago, R. Barbosa, R. Sadre,R. Schmidt and R. Hofstede , Attacks by "Anonymous" WikiLeaks Proponents not Anonymous , 2010.

[16] Check Point Whitepaper , DoS Attacks: Response Planning and Mitigation , August 2012

[17] Sean Gallagher , High Orbits and Slowlorises: understanding the Anonymous attack tools, Feb 16, 2012 , http://arstechnica.com/business/2012/02/high-orbits-and-slowlorises-understanding-the-anonymous-attack-tools/2/ - visited 10/02/2016.

[18] Imperva , Hacker Intelligence Initiative, Monthly Trend Report #12, September 2012.

[19] Impreva, https://www.incapsula.com/ddos/attack-glossary/http-flood.html, visited 16/02/2016.

[20] Verma Nischal, Francois Trousset, Pascal Poncelet, Florent Masseglia. Intrusion Detections in Collaborative Organizations by Preserving Privacy, Advances in Knowledge Discovery and Management,volume: 292, 2010, pp. 235-247.

[21] Chadouli Youssouf, Saoudi Lalia, "A New Feature Selection approach For Network Intrusion Detection Systems", Master thesis, University of Msila , 2014.

[22] Nicholas J.Puketza, Kui Zhang, Mandy Chung, Biswanath Mukherjee, Ronald A.Olsson . A Methodology for Testing Intrusion Detection Systems, IEEE Transactions on Software Engineering, volume:22, 1996,pp 719-729.

[23] Rebecca Bace, Peter Mell. Intrusion Detection Systems, National Institute of Standards and Technology,2001.

[24] Rafeeq Ur Rehman ,Intrusion Detection Systems with Snort: Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID, Upper Saddle River, NJ : Prentice Hall PTR, 2003.

[25] Herv´e Debar, An Introduction to Intrusion-Detection Systems, Proceedings of Connect, volume :2000, 2000.

[26] Michael Sweeney, C. Tate Baumrucker , James. D. Burton, Ido Dubrawsky, Cisco Security Professional's Guide to Secure Intrusion Detection Systems, 2003.

[27] Jacob .B, Automatic XSS detection and Snort signatures/ACLs generation by the means of a cloud-based honeypot system ,Doctoral dissertation, Edinburgh Napier University,2011.

[28] D.E. Denning, An Intrusion-Detection Model, IEEE Trans. Software Eng, volume:13, 1987, pp. 222-232.

[29] Jelena Mirkovic, Gregory Prier, Peter Reiher, Attacking DDoS at the Source, Network Protocols 10th IEEE International Conference, vol: 2002.pp. 312-321

[30] Christos Douligeris , Aikaterini Mitrokotsa , DDoS attacks and defense mechanisms: classification and state-of-the-art , Computer Networks, vol:44, October 2004,pp. 643-666.

[31] Saman Taghavi Zargar, James Joshi and David Tipper, A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks, IEEE COMMUNICATIONS SURVEYS & TUTORIALS,Vol:15, 2013, pp. 2046-2069.

[32] Thomer M.Gil ,Massimiliano Poletto , MULTOPS: a data-structure for bandwidth attack detection, 10th USENIX Security,2001,pp 23-38.

[33] Rajkumar , ManishaJitendra Nene , A Survey on Latest DoS Attacks:Classification and Defense Mechanisms , International Journal of Innovative Research in Computer and Communication Engineering , vol :1,October 2013.

[34] Steven M. Bellovin ,ICMP Traceback Messages, AT&T Labs Research, Florham Park,2003.

[35] Puneet Zaroo , A Survey of DDoS attacks and some DDoS defense mechanisms , Advanced Information Assurance (CS 626), 2002.

[36] Haining Wang, Danlu Zhang, Kang G. Shin, Detecting SYN Flooding Attacks, Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol: 3 ,2002,pp 1530-1539.

[37] W. Eddy, TCP SYN Flooding Attacks and Common Mitigations, RFC 4987 , August 2007.

[38] Cheng Jin, Haining Wang, Kang G. Shin, Hop-Count Filtering: An Effective Defense Against Spoofed Traffic , 10th ACM conference on Computer and communications security ,2003,pp 30-41.

[39] Naga Shalini Vadlamani , A Survey on Detection and Defense of Application Layer DDoS Attacks , Master Thesis - Science in Computer Science,*University of Nevada, Las Vegas*, December 2013.

[40] Joao B. D. Cabrera, Lundy Lewis, Xinzhou Qin, Wenke Lee, Ravi K. Prasanth, B. Ravichandran, Raman K. Mehra , Proactive Detection of Distributed Denial of Service Attacks using MIB Traffic Variables -A Feasibility Study , IEEE/IFIP International Symposium ,2001,pp. 609-622.

[41] Dileep Kumar G,CV Guru Rao,Manoj Kumar Singh,Farid Ahmad, Using Jpcap API to Monitor, Analyse and Report Network Traffic for DDoS Attacks , 14th International Conference on Computational Science and Its Applications , 2014.

[42] Opeyemi.A. Osanaiye, Short Paper: IP Spoofing Detection for Preventing DDoS Attack in Cloud Computing , International Conference on Intelligence in Next Generation Networks,2015,pp 139-141.

[43] Steven J. Templeton, Karl E. Levitt , Detecting Spoofed Packets, DARPA Information Survivability Conference and Exposition, vol:1, 2003, pp. 164-175.

[44] Hanaa A. Qeshta , Tawfiq S. Barhoom, Adaptive Worms Detection Model Based on Multi Classifiers , Thesis on Master of Science In Information Technology , Islamic University – Gaza ,2012.

[45] R Vijayasarathy , Balaraman Ravindran, S V Raghavann, A System Approach to Network Modeling for DDoS Detection using a Naive Bayesian Classifier , Third International Conference on Communication Systems and Networks ,2011,pp. 1-10.

[46] V.Mohan Patro, Manas Ranjan Patra , Augmenting weighted average with confusion matrix to enhance classification accuracy, ransactions on Machine Learning and Artificial Intelligence, vol:2, 2014,pp. 77-91.

ملخص: أصبحت العديد من المواقع الإلكترونية في وقتنا الحالي تواجه خطر ما يسمي بهجمات الحرمان من الخدمة. هاته الهجمات تظهر عندما يحاول المهاجم جعل موارد خادم الويب غير متاحة لمستخدميها مما يؤدي الى توقف الموقع عن العمل .وبالرغم من جهود العديد من الباحثين الى أنه لا يوجد حل أمثل للتصدي لجميع أنواع هجمات حجب الخدمة.

لذلك اقترحنا نظام مبتكر يعمل على كشف كل أنواع هجومات الحرمان من الخدمة التي تقوم على مبدأ استغلال البرتوكول TCP والبرتوكول HTTP. وذلك اعتمادًا على ثلاث طبقات من الكشف:

- أولاً، كاشف خارجي يوقف جميع التدفقات الشبكية القادمة من مصدر هو ينتمي أصلا الى القائمة السوداء الممنوعة من التواصل مع الخادم.
- ثانياً، كاشف للعناوين (IP) ذات المصدر المغشوش.
- ثالثاً، مصنِفين مقترحين لكشف هجمات الحرمان من الخدمة التي تعتمد على البرتوكولين HTTP وTCP. ولتصميم هذين المصنِفين قمنا بما يلي:
  - انتقاء اهم حقول البرتوكولHTTP، وذلك لحساب واستنتاج الخصائص التي تسمح لنا بتصنيف تدفق HTTP العادي وتدفق HTTP الخاص بهجمات الحرمان من الخدمة.
  - انتقاء اهم حقول البرتوكولTCP، وذلك لحساب واستنتاج الخصائص التي تسمح لنا بتصنيف تدفق TCP العادي والتدفقTCP الخاص بهجمات الحرمان من الخدمة.

الكلمات المفتاحية: الكشف عن هجوم الحرمان من الخدمة، خادم الويب، نظام الكشف عن التسلل، شجرة القرارات، كشف عنوان IP مغشوش.

Abstract: Recently many prominent web sites face so called Denial of Service Attacks (DoS).these attacks occur when an attacker attempts to make the web server, or servers, unavailable to serve up the web sites they host to legitimate visitors. Despite many researchers' efforts, no optimal solution that addresses all sorts of DoS/DDoS attacks is on offer.

Therefore, our framework aims to propose an alternative solution which handles all aspects of HTTP and TCP based DDoS attacks through the following three subsequent framework's layers:

- Firstly, an outer detector blocks attacking IP source if it is listed on the black list.
- Secondly, the IP spoofed detector to validate the source of incoming requests.
- Thirdly, two classifier modules are proposed to detect HTTP/TCP DDoS attacks, for this modules we :
  - Select the relevant features of the HTTP protocol, to calculate a new set of features to classify the HTTP traffic as normal or DoS attack.
  - Select the relevant features of the TCP protocol, to calculate a new set of features, to classify the TCP traffic as normal or DoS attack.

Keywords: Denial of service attack (DoS) detection, Web server attack, Intrusion detection system, Decision tree, IP spoofed detection.