T.C.
ALTINBAŞ UNIVERSİTY

Institute of Graduate Studies
Electrical and Computer Engineering

DETECTION OF DOSS ATTACKS AND ABNORMALITIES WITHIN THE NETWORK

Mohammed Hashim Mohsin ALHAMDI

Master Thesis

Supervisor

Asst.Prof. Dr. Oguz KARAN

Istanbul 2021

DETECTION OF DOSS ATTACKS AND ABNORMALITIES WITHIN THE NETWORK

By

Mohammed Hashim Mohsin ALHAMDI

Electrical and Computer Engineering

Submitted to the Graduate School of Science and Engineering
in partial fulfillment of the requirements for the degree of
Master of Science

ALTINBAŞ UNIVERSITY
2021

The thesis titled “Detection of DoSS Attacks and abnormalities within The Network “prepared and presented by “Mohammed Hashim Mohsin“ was accepted as Master of Science Thesis in Electrical and Computer Engineering.

Asst. Prof.Dr. Oguz KARAN

Supervisor

Thesis Defense Jury Members:

Asst.Prof.Dr. Oguz KARAN	School of Engineering and Natural Sciences, Altinbas University	
Asst.Prof.Dr. Abdullahi Abdu IBRAHIM	School of Engineering and Natural Sciences, Altinbas University	
Asst. Prof.Dr.Zenap ALTAN	School of Engineering and Natural Sciences, Beykent University	

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Approval Date of Institute of Graduate Studies

____/____/____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Mohammed Hashim ALHAMDI

Signature

DEDICATION

First I would like to Allah Almighty for the power of mind , health, strength , guidance knowledge and skills to complete this study .This thesis is wholeheartedly dedicated to my beloved father, who have been my source of inspiration, he tells me that" every success in your life will be best gift for me". To my mother, who have been supporting me with the kind and pure love.

ACKNOWLEDGEMENTS

I would like to thank my supervisor professor Prof. Dr. Osman Nuri Ucan for all support during my study. It's great pleasure to express my deepest gratitude to my friends who have shared with me best moments during my study for the Master degree.

ABSTRACT

Detection of DoSS Attacks and abnormalities within The Network

ALHAMDI Mohammed

M.Sc, Electrical and Computer Engineering, Altinbas University

Supervisor: Asst.Prof.Dr. Oguz KARAN

Date: 04/2021

Pages: 60

The issue of unambiguous identification of the user and his device on the Internet is a very important and addressed topic today. The reasons are mainly prevention against an increasing number of different types of attacks, but also the localization and adaptation of content to a specific type of user.

This implies the need to recognize and identify the users who interact with the application. There are several different approaches to identification, from network layer IP address mapping to application management of user accounts. However, each of them has its advantages and disadvantages.

The aim of this work is to create a unique identifier based on information available mainly from HTTP or TCP protocol. However, before building the algorithm itself, it is important to describe some key areas and procedures. The following paragraphs of the introduction will therefore briefly deal with the operation of client-server applications and the architecture of the work domain.

Keywords: HTTP level, Network Traffic, Network Administrators, HTTP Protocol.

TABLE OF CONTENTS

LIST OF FIGURES

Pages

LIST OF ABBREVIATIONS

HTTP	:	Hypertext Transfer Protocol
QoS	:	Quality Of Service
TCP	:	Transmission Control Protocol
IP	:	Internet Protocol
TLS	:	Transport Layer Security
FTP	:	File Transfer Protocol
SMTP	:	Simple Mail Transfer Protocol

1. INTRODUCTION

With the continued development of new technologies, the Internet and the Web are becoming an increasingly important part of our lives. The web as such is no longer limited to browsing on computers. It must adapt to different technologies, such as mobile or other client multimedia devices.

The issue of unambiguous identification of the user and his device on the Internet is a very important and addressed topic today. The reasons are mainly prevention against an increasing number of different types of attacks, but also the localization and adaptation of content to a specific type of user.

This implies the need to recognize and identify the users who interact with the application. There are several different approaches to identification, from network layer IP address mapping to application management of user accounts.

HTTP has client–server architecture. Communication is based on request–response. When a client makes a request, a server sends response back. It operates on TCP and offers TLS/SSL in extension protocol also called HTTPS. Quality of service (QoS) is limited and only provided by TCP. This protocol is mainly used for viewing websites and it is designed for devices where power and other resources are not limited.

With the continued development of new technologies, the Internet and the Web are becoming an increasingly important part of our lives. The web as such is no longer limited to browsing on computers. It must adapt to different technologies, such as mobile or other client multimedia devices.

The issue of unambiguous identification of the user and his device on the Internet is a very important and addressed topic today. The reasons are mainly prevention against an increasing number of different types of attacks, but also the localization and adaptation of content to a specific type of user.

This implies the need to recognize and identify the users who interact with the application. There are several different approaches to identification, from network layer IP address mapping to application management of user accounts. However, each of them has its advantages and disadvantages. They are discussed in detail in Chapter 4.

The aim of this work is to create a unique identifier based on information available mainly from HTTP or TCP protocol. However, before building the algorithm itself, it is important to describe some key areas and procedures. The following paragraphs of the introduction will therefore briefly deal with the operation of client-server applications and the architecture of the work domain.

2. BACKGROUND AND LITERATURE

Network layers

To better grasp the issue of user identification, it is first necessary to describe the individual network layers of the TCP / IP model and their protocols (2.1). The following text therefore focuses mainly on structures, the understanding of which is important for other modules of the theoretical and practical part of the work. It is therefore specific information of individual protocols, some of which were subsequently used in the implementation of the identification algorithm.

Giant. 2.1: Visualization of TCP / IP model layers, its most used protocols and transmission structures, source: own processing

Application layer

At the top of the hierarchy, the application layer is an abstraction that specifies the specific protocols and methods used by the host nodes on the network. This layer is defined both in TCP / IP and in the OSI (Open Systems Interconnection) model of network communication.

This work deals with the TCP / IP model, in which the application layer defines protocols and interface methods for communication between individual processes of connected parties in the network. However, the application layer itself only standardizes the form of communication, leaving the establishment of a comprehensive data connection and data transfer management for client-server and peer-to-peer applications on the protocols of the next transport layer. As the application layer does not specify in any way specific rules for the form of the data to be transmitted, the applications themselves must contain logic to ensure that both communicating parties are compatible with each other in this respect [Bra89b].

The application layer defines a large number of protocols used, such as:

- HTTP / HTTPS - for hypermedia transmission,
- TLS / SSL - providing secure remote communication of a given connection in the network,
- FTP - used to transfer files,
- SMTP - designed for email communication,

DNS - implementing a system of hierarchy of domain names, and other ... However, when creating an identifier, we will pay particular attention to information
from HTTP family protocols, specifically HTTP and HTTPS headers and attributes 1.

HTTP

HTTP (Hypertext Transfer Protocol) is a protocol designed for cooperation and transfer of information between individual systems for working with hypermedia. This protocol is stateless and generally usable not only for hypertext transmission, but also for communication with DNS servers, or for managing more complex objects, in which it applies a wide range of its headers and error messages. One of its characteristic features is the possibility to choose the representation of data, which gives the system great independence from the transmitted format.

HTTP works on the principle of request-response, based on the functioning of the client-server model of network communication. The client can be, for example, a web browser, and the server can be an application running on the host node device. In this case, the web browser sends a message in the form of an HTTP request to the server. Ideally, the server processes the request and generates a response (such as an HTML page) that it returns to the client as an HTTP response message. The response for the client always contains a so-called status, which informs about the completion of the request or a specific error and a possible body containing the response message [Bra89b].

Thus, an entire HTTP connection session between two nodes is a sequence of several such (request-response) transactions. However, the data transfer itself is not realized at the HTTP protocol level, but via the TCP protocol at the transport layer level. The HTTP client initiates the establishment of a TCP connection for a specific server port (typical ports are 80, 443 or 8080). The HTTP server listens on the given port and waits for the client request message. After receiving the message, the server processes it by default and returns a response.

The client and server communicate by sending text messages. The request consists of the following information:

● requirement definition

HTTP headers (for example: Accept-Language: en),

● blank line,

● possible message body.

An example of a requirement is shown in Figure 2.2. Also, the response message has a defined format similar to the request format, which consists of the following items:

● request processing status and reason,

● HTTP response headers (for example: Content-Type: text / html),

● blank line,

● The possible body of the response message.

The first line of the definition and each of the headers must end with characters <CR> <LF> (carriage return character followed by line feed character). At the same time, the blank line must consist exclusively of <CR> <LF> characters. From the point of view of identification, HTTP headers will be very important later. For HTTP version 1.1, the only mandatory header is Host defining the server to which the request was sent [BL10].

Transport layer

In the context of the TCP / IP network communication model, the transport layer is second in order, between the application layer and the network layer. It is a set of protocols and methods that provide so-called process-to-process communication of individual network nodes by the protocol of the application layer described above. Specifically, it implements the functionality of the data transfer itself in the data stream, ensures its reliability, control and multiplexing.

The best-known protocols in this layer include, for example, TCP (Transmission Control Protocol), whose name is also borne by the network model itself. This protocol is focused on the transmission of more complex connections, while the second of the protocols - UDP (User Datagram Protocol), is suitable and used more for simpler messages and connections. Protocol

TCP is more complex mainly due to its design, which preserves the status of individual transactions and ensures the reliability of delivery of all data messages [Bra89a].

Other important transport layer protocols are DCCP (Datagram Congestion Control Protocol) used, for example, in multimedia transmission, or SCTP (Stream Control Transmission Protocol), which

deals with the transmission of telephone signaling.

TCP protocol

TCP is one of the most important and most used protocols in the entire network model. Its most important characteristic is probably the guarantee of complete delivery of all packets in the correct order. Therefore, TCP also attaches a so-called TCP header (TCP Header) to each packet to the data of the application layer itself, which is supplemented by information 2.4 to achieve this functionality [FID08].

Some of the information from these headers is quite often used in existing user identification tools, which are discussed in detail in Chapter 4. Their use in the identifier algorithm of this work is then described in Chapter 5.

In order for the required data to be sent itself, a TCP connection must be established between the client and the host. Both the establishment and the termination of this connection have a precisely defined form of communication called a three-way handshake, which is described in diagram 2.4.

UDP protocol

Unlike TCP, UDP is a protocol that provides so-called unreliable data transmission. This unreliability lies in the possible loss of individual packets during data transmission, without the possibility of automatic forwarding of missing information. At the same time, UDP does not guarantee any order of their delivery [Bra89a].

Thanks to these features, this protocol is very fast and simple, while delivering packets independently and in a relatively short time. Another of its typical features is statelessness, thanks to which it is used mainly by systems that send a large number of small packets to multiple recipients. Examples are: DNS servers, IPTV media, online games and the like [FID08].

The UDP protocol header is relatively simple and consists of only the necessary information, such as the sender's port number, the packet length, and the recipient's port specification. Therefore, the UDP packet itself is visibly smaller than the TCP packet, which also helps its transmission speed and processing. The detailed structure of the header is shown in Figure 2.5. However, because UDP provides a small amount of information, its use in identification is minimal.

Although the two protocols are relatively different and serve different purposes, their common functionality is the use of ports to identify communicating party applications. The ports for TCP

and UDP are independent of each other, which brings the possibility of communication of one device in both TCP and UDP contexts.

Network layer

The network layer is a group of Internet definitions, protocols and specifications, which in the TCP / IP network model is used (transport layer) to transmit datagrams (packets) from the sender, through individual network interfaces, to the recipient. It thus implements the so-called host-to-host data transmission. Each of the nodes in the network is identified at the network layer by a so-called IP (Internte Protocol) address [SC81]. This layer derives its name from its main functionality, which is mainly the formation and creation of a connection to the Internet network through interconnected nodes.

Network layer protocols work with packets based on the IP addresses of individual devices. By far the best known protocol of this layer is the already mentioned IP (Internt Protocol). However, the network layer also defines other important and used protocols, such as: ICMP (Internet Control Message Protocol), which is used to send error messages.
messages by operating systems, or Internet Group Management Protocol (IGMP) to support multicast routers. However, it does not contain any protocols that define communication at the local network and physical connection level [Bra89a].

Network interface layer

The network interface layer works at the lowest level in the TCP / IP model. This layer describes the network architecture itself and its communication at the physical connection level. It is a group of methods and communication protocols that therefore operate on the physical connection of two nodes.
Some of its important protocols are: Address Resolution Protocol (ARP), Reverse Address Resolution Protocol (RARP), Neighbor Discovery Protocol (NDP), and others [Bra89b].
However, the information in these protocols operates at too low a level of network infrastructure, so it will not be possible to use it for identification. Thus, the network interface layer is shown in the report for completeness only.

Denial of Service attacks

One of the main reasons and motivations for user identification is prevention against attacks. One of the best known attacks that can be defended in this way is the so-called Denial of Service (DoS) attack. In general, an DoS attack is considered to be an attempt by an attacker to prevent authorized users from accessing the information or services of the provider [McD09]. Graphically, this type of attack is shown in. The attacker's effort is to disrupt the connection by constantly disrupting the server service or network infrastructure, as a result of which the host connection may be partially or completely lost.

Basic types and techniques There are many different types and techniques of performing DoS attacks. The following paragraphs therefore describe the most commonly used ones and identify the means by which they can be prevented.

Distributed DoS attacks

We speak of distributed DoS attacks when several devices flood the entire bandwidth or resources of the target system, which is usually one or more servers (an example is Figure 3.2). Such an attack is often the result of the use of various systems and devices (such as a so-called botnet) that try to exploit the target system. The botnet is an extensive virtual network of artificial (zombie) computers, the aim of which is to take orders without the owner's knowledge. When the target system consumes all free connections, no more can be established [ZJT13].
The main advantages of an attacker in using a Distributed DoS attack are the fact that multiple devices can generate more load than one, while the use of a number of systems provides

takes care of much more difficult detection of his identity. At the same time, the behavior of each of these devices is less observable, which makes it difficult to defend against the attack itself.
These advantages also lead to the development of defense mechanisms. On the target server side, a simple increase in bandwidth above the current attack size will no longer suffice, as an attacker may, for example, increase the number of devices involved, which would also cause a load and system outage [ZJT13].

Semantic DoS attacks

Semantic attacks use specific functionality or an implementation error of the application, or the protocol of the victim's device, to exploit a certain amount of its resources. For example, in the case of a TCP SYN attack, this exploited functionality is the allocation of a significant amount of space waiting for connections immediately after the TCP SYN request is acknowledged. An attacker opens multiple connections that he never closes, flooding the server. In a CGI attack, the attacker's goal is to flood the processor with multiple CGI requests in this way. One of the most dangerous attacks is the NAPTHA attack, which focuses on the TCP protocol. Initiates many TCP connections, which then fill all available server resources [MR04].

Reflection and amplification

In a Distributed Reflection Denial of Service (DRDoS) attack, any host IP address is considered to be a reflecting device that returns the packet when a packet is sent. Attack groups purposefully organize the host servers they control to send fake data simulating an infected node to a reflecting server [Buk15]. As a result, the victims are attacked from a significantly larger number of source points, extensively scattered in the network, whereby attackers effectively cut off their connection from the rest of the Internet. The layout of the DRDoS attack is shown in Figure 3.3.
An amplification attack is a type of reflective attack where the reflecting server sends many times more responses than it receives. In this way, it also multiplies the load on the connection between the victim and the source node [Pax01].

DoS tools and DoS as a Service

A typical method of transmitting the mechanisms of DoS and DDoS attacks is malverer. One example was the so-called MyDoom [BBC04]. It is a DoS mechanism that started at a pre-scheduled time. This attack involved setting the value of the target system's IP address to deploy the malware, and no user interaction was required to launch the attack.
Another way to exploit the system for a DDoS attack is to use a hidden piece of third-party software that allows an attacker to download a zombie agent, or the software itself already contains it. An attacker could also penetrate the system by using automated tools that exploit bugs in programs

that listen to remote access.

connections. This scenario affects primary systems that act as web servers. A typical example of a DDoS tool in this area is the so-called Stacheldraht [Dit99]. Stacheldraht uses a layered structure in which an attacker uses a client program to connect to handlers
that are misused to transmit commands to zombie agents performing the DDoS attack itself. Agents are exploited by attackers through service functions by using automated algorithms to find vulnerabilities in programs that accept remote connections. Each of the functions is able to control up to a thousand agents.

DDoS tools such as Stacheldraht still use classic DoS methods aimed at amplifying and spoofing IP addresses, such as bandwidth congestion. Another option is to flood resources - SYN Flood attack. Newer tools also use DNS servers for DoS attacks [Dit99].

Tools like MyDoom can be used against any IP address. Less experienced attackers use them to block the availability of popular and well-known web servers. On the contrary, more sophisticated attackers use these tools to blackmail, for example, against their business adversaries [KPM15].
In some cases, however, the device may become part of a DDoS attack intentionally - with the consent of the owner. An example is the distributed Operation Payback attack organized by Anonymous [Ley10].

DoS – Summary

As can be seen from the previous paragraphs, there are a really large number of variations of DoS and DDoS attacks. At the same time, their number is growing over time and therefore it is necessary to develop new tools to defend against them. However, the most difficult part is to detect and identify such an attack, especially its perpetrator. Therefore, one of the uses of the algorithm described in Chapter 5 and the implementation part is this functionality.connections. This scenario affects primary systems that act as web servers. A typical example of a DDoS tool in this area is the so-called Stacheldraht [Dit99]. Stacheldraht uses a layered structure in which an attacker uses a client program to connect to handlers that are misused to transmit commands to zombie agents performing the DDoS attack itself. Agents are exploited by attackers through service functions by

using automated algorithms to detect vulnerabilities in programs that receive.

Encryption of web traffic and its monitoring

Over the last two decades, the practice of encrypting web traffic changed drastically. Twenty years ago, HTTPS was used almost exclusively in banking and retailing [3]. Ten years ago (2009), even some of the major sites like

Facebook still have had users enter their usernames and passwords on a HTTP page [4]. Today, the consensus of the IT security community is that even static sites should have traffic encrypted to protect their customers, for example, from Man-In-The-Middle attacks adding cryptomining scripts or unwanted ads [5].

Currently all 100 top non-Google sites support HTTPS and 96 of them default to it. By Googles estimates, "these 100 sites account for approximately 25% of all website traffic worldwide" [1]. As of March 2020, 60.93% of websites from Tranco Top 1 Million were using HTTPS [6]. The following graph shows the percentage of pages loaded over HTTPS in Chrome over the last 5 years.

Figure 2.1: Percentage of pages loaded over HTTPS in Chrome by platform [1]

Ten years ago, getting a certificate that could be used for HTTPS involved manual requests, paying a significant amount of money (for example, Comodo SSL certificate cost $64.95 and a wildcard certificate $334.95 at 2012 [7]), and manual deployment. Today they can be acquired for free from issuers like Let's Encrypt[4] and their deployment and renewal can be nearly completely automated

for using utilities like Certbot, which uses the ACME protocol (Automated Certificate Management Environment).

Automation is certainly a good thing. It mitigates a lot of the potential for a human error, however software can have faults as well. Today it is already considered common practice to have a monitoring setup that can alert you in case of a problem. However, a lot of these systems are reactive, not proactive.

3. NETWORK MONITORNING

Network monitoring is essential in today's world. It is used for pricing, usage analysis and last but not least also for security control. There are several approaches to network monitoring, each with its advantages and disadvantages. We can divide network monitoring into two basic sections, *active* and *passive* monitoring.

Active monitoring is mainly focused on active network scanning. Tools like Nmap or ping are used here. The disadvantage is that active monitoring increases network traffic. The advantage, on the contrary, is to get far more detailed data using this method. This type of monitoring is not very often used to monitor the network continuously because it increases network traffic too much. Active monitoring methods can be used for network topology discover or network asset management.

Also can be used by attackers to scan the target network. Another option where active monitoring can be used is a self network scan within an organization to determine the status of hardware or software.

Passive monitoring methods do not increase network traffic as much as active monitoring methods. For this reason, they are better suited to continuous network monitoring. Passive monitoring methods can be, for example, Deep Packet Inspection (DPI) or network flows monitoring.

DPI is a technique of seeing the whole packets including payload [1]. For this reason, this method is very time-consuming and computationally intensive. Still, on the other hand, it often provides valuable information that cannot be obtained in any other way. Network flow monitoring is a useful technique even for large networks with massive traffic as it focuses only on the analysis of communications, rather than the payload of individual packets. [2]. This method is more effective in comparison to DPI because only packet headers are processed. In my thesis is used network flows monitoring method as a data source. Hence, the description of network flows in detail is in the following sections.

3.1 MONITORING OF ENCRYPTED NETWORK TRAFFIC

Network traffic can be monitored either actively or passively. Active monitoring requires probing techniques to scan the network hosts. Passive monitoring, on the other hand, observes the existing communication and does not generate any of its own. We cover only passive network monitoring techniques, as they are the focus of the thesis. Possible techniques for passive network traffic monitoring are deep packet inspection (DPI) and flow monitoring. DPI methods analyze the content of each packet. Therefore, the DPI is considered too slow for large networks, and the content of encrypted packets is not readable without decryption. For this reason, we focus solely on the progress in the field of monitoring of encrypted network flows. Monitoring of general network flows had already been exhaustively described in the past [3].

Traffic Classification

The first attempts at monitoring encrypted traffic were aimed at encrypted flow classification. Flow classification allows assigning each flow to a specific protocol or application, thus providing information about who is communicating with whom and what is the purpose. When the flow is encrypted, the patterns that identify the specific service or protocol are lost. Therefore, an accurate classification of encrypted traffic is currently a challenge [4]. A survey by Velan et al. [5] categorizes classification methods for encrypted traffic based on the used data inputs and provides their comparison. The survey lists two types of data inputs for classification that were used at the time – payload-based and feature-based. Dainotti et al. [4] mention one more possible data input – port-based.Port-based classification uses well-known port numbers to assign network flow to a protocol, so it is not affected by encryption. It is also fast, so it is suitable for the use in firewalls and similar devices processing a large volume of traffic [6]. However, ports can be often scram- bled by NAT, obfuscation, or random port assignment. Moore [7] and Madhukar state that only 30% to 70% of the Internet traffic could be classified using port-based classification methods. Payload-based classification requires some insight into the transferred content, so it is primarily aimed at unencrypted protocols. How- ever, some systems are also able to classify encrypted flows. They use packet format specifics of each protocol and the classification is being carried out by regular expression matching algorithms (Deri et al. [8]). Feature-based classification focuses instead on

specific communication protocol patterns. Moore et al. [9] identified almost 250 such features. For classification, feature-based methods employ supervised machine learning methods (Okada et al. [10]), semi- supervised machine learning methods (Bar-Yanai et al. [11]). A new approach for traffic classification based on deep learning had been proposed recently by Lotfollahi et al. [12]. Their Deep Packet framework is able to automatically extract features from network traffic using deep learning algorithms to classify it. According to the results, this method was faster and more accurate than the traditional methods mentioned above. It follows that deep learning might be used for traffic classification and network analysis tasks more frequently in the future.

Content Inspection

While the classification methods are able to distinguish different types of communication protocols and applications, they do not reveal much about the content being transferred. Investigating ways of how the content of the encrypted traffic could be analyzed or at least how to differentiate malicious from benign traffic was the next step for researchers. Three different approaches that use different data inputs are currently used to gain some visibility into encrypted traffic. The first approach is based on interception, decryption, and pay- load inspection, which in theory reduces the problem into monitoring unencrypted content. The inspection is accomplished with a device that acts as a proxy, situated between the user and server, where it can intercept, decrypt, inspect, then re-encrypt and forward on the user's traffic to its original destination [13]. This method is sometimes used by institutions to inspect the traffic of users and employees for malware. O'Neill et al. [14] conducted a study on how is this approach perceived by users and also expressed concerns about such proxies being set up by attackers. The approach raises not only privacy concerns but also may have unpredictable effects on the behavior of malware using encrypted traffic, as argued by Erquiaga et al. [15]. The negative impact of TLS interception and decryption on communication security was in-depth investigated by Durumeric et al. [16].

The second approach relies on the analysis of unencrypted data that are exchanged during the negotiation of the encrypted connection. Anderson et al. [17] provide a comprehensive study of malware's use of TLS by observing the unencrypted TLS handshake messages. They also identify handshake features, that can be used to cast some light on the data transferred by TLS, e.g., user agent, server name indication (SNI), and server certificate. The usage of SNI for flow analysis was further explored by Shbair et al. [18]. To analyze the traffic from the client side, Husák et al. [19]

proposed a client device fingerprinting algorithm based on the specific parameters of the TLS handshake. However, the current trend is to preserve user privacy and encrypt as much of the transferred data as possible. The new standard TLS 1.3, for example, transfers the server certificate encrypted [20]. A discussion was in progress to encrypt also the SNI, but at the end, only an optional protocol extension was proposed [21].The third approach focuses on the statistical data that can be de- rived from encrypted flows. Cisco in their white-paper specify such statistical flow features they use to identify malicious traffic on their firewalls [22]. These features include the length of the first *n* packets, their inter-arrival times and byte distribution of the flow during its active period. The statistical flow features are not influenced by any type of encryption, which is a definite advantage of this approach for current network monitoring.

While the current methods of network flow monitoring can provide some insight into the encrypted traffic, they will have to cope with further loss of input parameters due to encryption. This information lost in encryption need to be substituted by some other data source. The event logs often generated on the host devices might provide the data that is needed to fill this gap.

3.2 HOST-BASED MONITORING

Host-based monitoring focuses on the collection and analysis of logs. Typically, each service or application running on a host generates some kind of log. Every action executed by such software is registered, timestamped, and stored in the software's corresponding log. A host typically contains many logs, depending on the purpose of the host, e.g., a webserver will contain different logs than a database server. These logs can be used to monitor if the software works within its specifications, to check for any errors in software operation, and also for cybersecurity analysis. However, with the number of monitored hosts rising, the logs become too cluttered for manual analysis. This issue is exacerbated by the fact that there is not a single standard for the log event format, so logs between different software are very heterogeneous.

The initial research in host-based monitoring examined the statistical modeling of errors and failure prediction in complex systems. In 1990, Lin and Sieworek [23] proposed a technique for device fault prediction based on data mining. Data mining approaches like frequent pattern mining and sequential pattern mining have been used by Ma et al. [24] and Vaarandi et al. [25] to detect commonly occurring event patterns before a failure. While these techniques helped in the

management of complex systems, the logs remained too confusing and incomprehensible for manual analysis.

As far back as in 1994, Eick et al. [26] considered the logs cluttered and disorganized even though they contain valuable data, as those might be easily overlooked. They proposed a system for visualization of critical events and suppression of the unimportant ones. Takada and Koike [27] proposed a similar information visualization system for monitoring and auditing computer logs in 2002. These systems, however, only assisted system administrators to perform analysis tasks manually.Before any automated log analysis could take place, the log het- erogeneity issue had to be tackled. Currently, there is a number of standards that are used in various applications. The syslog standard is common for Unix based applications [28]. On the other hand, Microsoft Windows based applications may use a different proprietary standard, however, it is not public [30]. Other logging practices might include any format capable of structural data storage, being it csv, json, and xml format. The current systems for Security Information and Event Manage- ment (SIEM) are able to provide full infrastructure for log monitoring. That includes their collection to centralized storage, normalization, indexing for analysis, and finally visualization. Jayathilake [31] provides an overview of the proprietary SIEMs and proposes a frame- work developed for structured log analysis. The SPLUNK SIEM is considered the most popular by Jayathilake, with the alternatives being LogRhythm, ArcSight Logger, and loggly [32]. Aside from proprietary SIEMs, open-source SIEMs are also available. The Elasticsearch-Logstash-Kibana (ELK) stack is a widely used open-source system for log analysis. Its latest version supports functions specifically aimed at security monitoring, including integrated machine learning algorithms for anomaly detection, event correlation into complex alerts, and dashboard overview to keep track of the important events. There is a number of systems available for automated log monitoring and analysis. However, some devices might be unable or unwilling to share their logs, specifically BYOD and IoT devices. This represents a blind spot for host-based monitoring. On the other hand, logs are not affected by end-to-end encryption, so the creation of a correlated view on an event with network monitoring would be mutuallybeneficial.

3.3 TECHNIQUES FOR EVENT CORRELATION

The definition of event correlation is based on complex event processing method [26]. Events are

individual or aggregated messages or alarms describing or relating to activities in a network. Correlation is a connection or relationship between two or more entities. The event correlation is a process during which events are examined for com- mon parameters and are joined together if they appear to be caused by a single activity. Limmer and Dressler [34] define it as *"a process for consolidating events to increase their information quality while reducing the quantity of events. Meta data (such as time, location, network topology or administrative information) can be used to improve the quality of single or combined events."*

Event correlation is a critical process in monitoring complex systems that generate large amounts of operational data. It was originally used in the 1980s and 1990s for fault event correlation, especially in the management of telecommunication networks . Fault event correlation uses techniques to aggregate events, suppress low priority events, and generalize low-level events into higher-level ones to identify the exact point of failure. Possible fault event correlation approaches are listed by Tiffany [35] and include the rule-based, codebook, and artificial intelligence approach. The rule-based approach is relevant and often used even in current SIEMs.

With the swiftly rising amount of globally interconnected systems in the early 2000s, the event correlation systems needed to be modified to work with this new type of events and cope with a huge amount of events at once. A number of correlation algorithms were published to tackle these issues. Debar and Wespi [17] proposed a correlation algorithm to reduce false-positive alerts and prevent flooding of the operators. Krugel et al. [37] proposed a rule-based decentralized correlation system to enable better scaling over large networks. Abad et al. [38] improved the effectiveness of intrusion detection and reduced false-positive alerts by correlating log events from different devices.
date survey published by Mirheidari et al. [39] in 2014 – knowledge-based techniques, similarity-based, and statistical-based. The authors also categorize all at the time published correlation techniques into these categories.

Knowledge-based or rule-based techniques use predefined correlation rules to associate events. Because they rely on a static set of rules, they tend to create fewer false positives compared to other more dynamic methods. The reliability of the correlation, however, strongly depends on the set of

rules, as poorly specified rules degrade the effectiveness of the technique. Additionally, correlated will be only the events covered by a rule. When a new correlation pattern is discovered, a new rule needs to be defined and included in the system. Otherwise, the pattern will not be detected and correlated. Therefore, the set of rules needs to be updated on a regular basis. The similarity-based techniques compare the similarity of either two events. If the events have the specified similarity, they are correlated into one event. The most important advantage of these techniques is that there is no need for a precise definition of the correlation rules. Moreover, the correlation can be done only with the definition of similarity factors for event features.

The statistical-based correlation techniques search for similar statistical attributes of the event occurrence and tests if a categorization can be found. However, scientific results indicate that using these algorithms is possible only in very specific domains in which domain attributes are taken into account of designing algorithms. Otherwise, they will suffer from a high error rate [39].

All the techniques mentioned above use different parameters for correlation. However, the time of occurrence of each event is common for most of them. The time of occurrence itself is not a strong enough parameter for a correlation to be based upon it, but provides the necessary temporal constraint for a correlation, while other parameters provide the spatial constraints.

Henderson et al. [40] investigated possibilities of associating logged malicious events to a specific network flow in 2018. They proposed a time-based correlation algorithm and confirmed that this approach is viable. They also discussed the limitations that result from open issues of this new type of correlation. However, they investigated the correlation only from the malicious event standpoint and did not consider network flow features aside of time and source.

3.4 CYBER SITUATION AWARENESS MEASUREMENT

The definition of situation awareness is traditionally adopted from the work of Mica Endsley [41]. She defined the situation awareness as *"the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future."* In cybersecurity, the level of situation awareness has a direct influence over the duration of the incident response process [42]. Endsley dedicated her research to the field of

situation awareness in the late 1980s and has authored over 200 publications since, including works focused on finding methods for situation awareness measurement.

Endsley proposed the Situation Awareness Global Assessment Technique (SAGAT) for measuring the situation awareness in 1988 [43]. This technique requires that a simulation of the operational tasks is run. The simulation is momentarily paused at specified moments in order to query operators on their situation awareness. The technique is designed to measure to what extent a human operator is aware of the situation and how well he or she manages to maintain and develop this awareness as time progresses. SAGAT was originally designed to assess the situation awareness of fighter pilots in simulated combat situations. However, it is general enough so that it can be modified and applied to other fields.

Endsley published another research article in 1995, following her earlier proposal of SAGAT, where she argued the validity of the technique and demonstrated its application in other contexts [44]. She also argued in favor of immediate evaluation of situation awareness, even at the cost of pausing the simulation, as evaluation after the end of the simulation relies on human long-term memory that tends to be too inaccurate. Her extensive research into the field of situation awareness laid the foundations for its measurement in many different contexts. However, none of her works focused specifically on the situation awareness in the cybersecurity context.

Techniques for measurement of cyber situation awareness (CSA) were explored by other researchers. Doupé et al. [45] introduced two new metrics of CSA in 2011. However, these metrics are designed for use in a specific type of exercise – capture the flag. Franke and Brynielsson [46] surveyed the articles available in the field of CSA in a systematic literature review in 2014. The review notes that the majority of the available works describes specific aspects that are related to situation awareness. However the impact of the proposals on situation awareness is rarely measured. It also highlights that the situation awareness definition and SAGAT given by Endsley in 1988 can be used as a basis for such measurement.

The research in CSA measurement resulted in a number of techniques based on SAGAT being published. Giacobe [47] used a SAGAT questionnaire to evaluate two types of interfaces. He tested

the participants' understanding of what specific IP-addresses represent (e.g., workstation, internal server, external web server or external host), what is a reasonable value for a vulnerability scan, and if specific alerts should be reported. Huang [48] used performance measures and SAGAT to both provide post-training feedback to trainees and to receive feedback from trainees during system evaluations. Evan- gelopoulou and Johnson [49] applied SAGAT in a cybersecurity study. They conclude that the measure is difficult to use, and stated that there is still room for improvements. Most recently, Lif et al. [50] proposed the method freeze probe based on SAGAT in 2017 and published the questionnaires they used to measure CSA of log analysts.

The SAGAT proved sufficiently versatile to be used for CSA measurement in the past. However, the technique provides only a frame-work that needs to be modified to fit each individual case and requires expert knowledge of the field to specify input parameters.

3.5 NETWORK FLOWS

According to the RFC 7011, a network flow is defined as a set of IP packets passing an observation point in the network during a specified time interval. All packets belonging to a particular flow have a set of common properties [51]. These properties usually include the source and destination addresses, the source and destination ports, and the protocol used. Data such as the number of bytes transmitted, the number of packets sent or the duration of the flow can be then collected for each flow.

Packets are captured at the observation point. Generally, the source from which the data is collected and obtained is called the exporter. It is typically a probe that is connected with a device that duplicates traffic on the line (Traffic Access Point). However, also other network devices such as a router can do the probe function. Still, it is not customary to add a probe function to internal network elements (routers, switches) as it increases the performance requirements of the device. The probe then collects flow records and sends it to the collector. A collector is usually a device with large storage capacity that collects data from one or more usually several exporters. NetFlow v9 protocol is most commonly used to collect network flow records. It is the ninth version of the protocol developed by CISCO company. Other examples are NetStream, JFlow or IPFIX protocols[52]. For

sending data from exporter to the collector, TCP or UDP protocols are used. Once data are sent to the collector, they are deleted from exporter's cache.

3.6 CAPTURE OF NETWORK FLOWS

As already mentioned, a device called an exporter is used to capture data. It analyzes all packet headers and assigns them to flows. The flow is created when the exporter intercepts the first packet. Each additional packet that has the same set of common properties is then assigned to the same flow. The following data are usually recorded about the flow:

- Source and destination IP addresses
- Source and destination ports
- Flow duration
- Number of packets and bytes transmitted

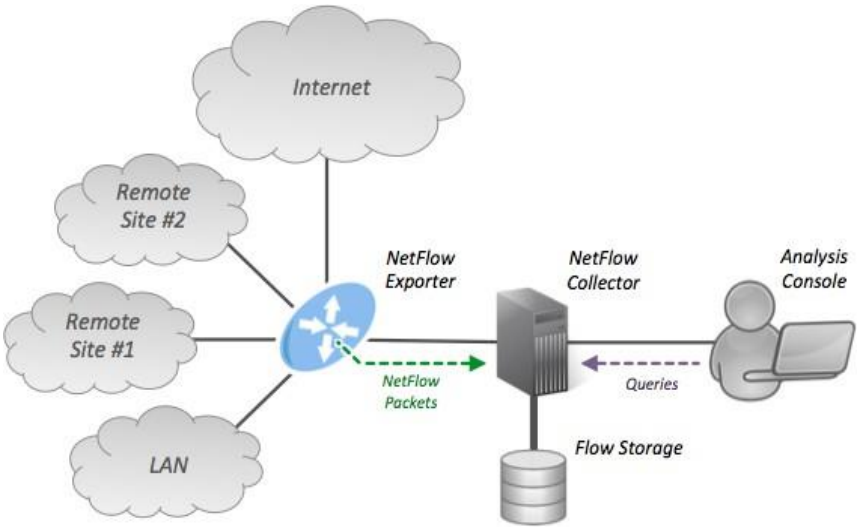

Figure 3.1: The schema of NetFlow architecture [53]

This approach of network monitoring does not interfere with net- work traffic at all. This is an advantage and disadvantage in once because anybody can not find out that the traffic is monitored. Advantage because if someone knew that the network is being monitored,

traffic could be directed differently and no undesirable behavior could have been detected. Disadvantage because of privacy issues.
When the flow is terminated, the record is sent to the collector.
There are three options of how the flow can be terminated.

Packet with FIN or RST flags is captured.

Inactive timeout - Flow is terminated after a specific time when the probe does not capture any packet belonging to the flow.

Active timeout - Active timeout is used in case of very long-lasting flows. The very long-lasting flows may never be exported, so usually after 300 seconds, every active flow is terminated and sent to the collector. Next incoming packet then starts the new flow. This ensures that each communication will be recorded and sent to the collector no matter how long the communication takes.

Table 3.1: Network flows example

Start	Duration	Proto	Src IP : Src Port	Dst IP : Dst Port	Packets	Bytes	Flows
2021-01-21 14:00:00:156	0.156	UDP	37.237.77.235:55864	239.36.36.52:80	50	560	1
2021-01-21 14:00:00:228	0.000	UDP	37.237.77.36:4859	239.36.54.12:78555	1	48	1
2021-01-21 14:00:00:548	0.255	UDP	37.237.77.52:80	239.36.55.235:55864	120	10080	1

3.6.1 Export of Records

Recorded flows are sent to the collector as soon as enough records to fill in packets is exported, or a predefined timeout is triggered. The collector is a device with a large storage capacity that receives data from an exporter and stores and process them. Usually, one collector receives data from several exporters. The NetFlow standard (RFC 3954) [54] does not specify a specific NetFlow listening port. However, the sending from the exporter to the collector is usually done on ports 2055 or 9555 or 9995 using User Datagram Protocol [55]. On the collector, the data can be further processed by tools such as nfdump,

nfsen, or others.

3.7 NETWORK SCANNING

The network must be monitored to detect anomalies. There are many ways to monitor the network. You need to realize exactly what needs to be monitored on the network. It is possible to monitor without an agent, but monitoring with an agent is a much more effective way, and for smaller networks it is best to configure it, as this will ensure a greater overview of events that take place in the network. Network monitoring methods are divided into three categories according to their packet processing capabilities: deep packet inspection, network flow monitoring, and extended flow monitoring. We evaluate the network monitoring methods by their usability in large-scale and high-speed networks. Broader monitoring range makes observation of network reconnaissance more convenient, while packet processing capability make the detection more accurate. Deep packet inspection (DPI) is widely used method of network traffic analysis. DPI processes both packet header and packet payload to identify used protocols, reconstruct transmitted data, search for predefined signatures, etc. Methods of DPI are often included in Network Intrusion Detection Systems (NIDS), systems designed to detect malicious network activity, and Intrusion Prevention Systems (IPS), which provide cooperation with firewalls to mitigate the malicious traffic. Widely-used examples of NIDS based on payload examination are Bro [37] and Snort [44]. Although methods of DPI are extensible and able to process almost any protocol as well as data in the packet payload, they suffer from several limitations. The main limiting factor of DPI is the processing speed which makes DPI unsuitable for large-scale monitoring of high-speed networks. Although combination of fast, low accurate methods with precise traffic analysis methods, the resulting increase in speed is still not sufficient for current network running at 10 Gbps and higher. DPI also faces law and privacy concerns due to processing of packet payloads. Last, encryption of the network traffic is naturally a limitation of DPI. Network flow monitoring aggregates the network traffic into flows, formalized network connections. A network flow is defined in the RFC 3954 [9] as a unidirectional sequence of packets with some common properties that pass through a monitoring device, the network probe. The common properties are source and destination IP address, source and destination ports, protocol number. Only the data from packet headers are processed and, in addition, aggregated by the common properties. Therefore, options of network traffic analysis are reduced. On the other hand, traffic processing is significantly faster, which makes flow monitoring suitable for

monitoring large-scale and high-speed networks. Flow representations of a network connection simplifies queries on communication partners, e.g. who contacted who. A survey describing fundamentals of flow-based monitoring focusing on NetFlow [9] and IPFIX [10] was published by Hofstede et al. [18]. Extended flow monitoring is a flow-based approach to network monitoring enhanced by application-level parsers. The exporter uses one or more parsers specific to an application-level protocol to process payload of a packet. Only a fixed-length part of a packet is processed, which reduces the computing time and memory requirements. Application-level data are added to a traditional flow record. Processing speed of extended flow monitoring approximates the speed of traditional flow monitoring. Therefore, extended flow monitoring is an interesting choice for large-scale and highspeed network monitoring that combines speed of traditional flow and capabilities of DPI. Current extended flow monitoring tools focus on parsing of widely-used protocols, such as HTTP [55], DNS [54], and others.

3.7.1 MONITORING AREAS

In general, the areas for monitoring are divided into:

- servers and their services
- active network elements
- network communication, operation
- security

Detailed specification of areas when monitoring network traffic:

- availability of servers, services, applications
- events on servers
- resource utilization
- line utilization
- network traffic statistics
- analysis of non-standard network behavior
- port information
- security incidents

So you need to realize exactly what needs to be monitored for different network events, because each attack is targeted to a different area. Based on these findings, safety measures must be chosen.

Monitoring technology

As already mentioned, it is therefore possible to monitor with or without an agent. Agentless monitoring tests the server's own services or retrieves data using standard protocols, such as SNMP (part of the Internet Protocol Suite). Not a very effective method for larger networks. Monitoring with an agent, ie the method described in this work, is based on the configuration of a client. This method allows you to obtain more detailed data and set up more comprehensive network protection. Network monitoring technologies are:

• availability of servers via ping
• service availability - establishing a TCP connection
• events from servers - syslog
• data acquisition by the client
• data acquisition using protocols
• network flow monitoring
• analysis of network protocols
• security in the IDS / IPS network

3.4 Ways to look for anomalies

Anomaly Monitoring and IDS systems are divided into 2 basic groups:

• HIDS - Host based IDS
• NIDS - Network IDS

The first mentioned method is only marginally, as this work focuses on NIDS. HIDS is therefore host-oriented. These systems use records generated by the operating system kernel, monitor ongoing processes in the context of running applications and file changes. NIDS process information obtained from network interfaces, their location is critical,
to capture as much network traffic as possible. Their advantages include

• with a good deployment, the possibility of monitoring a large network
• NIDS do not affect network operation in any way

Among their weaknesses are

• It may be difficult to process all packets during heavy traffic
• Encrypted traffic cannot be analyzed
• it is not possible to clearly determine whether the attack on the network was complete

4. DENIAL OF SERVICE ATTACKS

One of the main reasons and motivations for user identification is prevention against attacks. One of the best-known attacks that can be defended in this way is the so-called Denial of Service (DoS) attack.

In general, an DoS attack is considered to be an attempt by an attacker to prevent authorized users from accessing the information or services of the provider [McD09]. Graphically, this type of attack is shown in Figure 4.1. The attacker's effort is to disrupt the connection by constantly disrupting the server service or network infrastructure, as a result of which the host connection may be partially or completely lost.

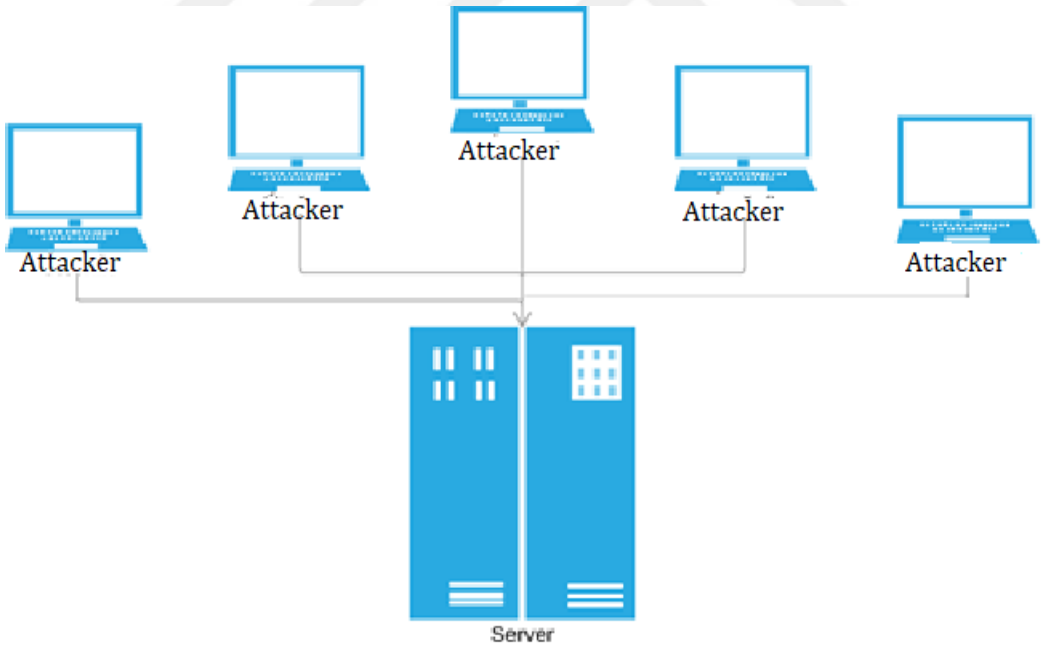

Figure 4.1: DoS attack scheme, source: own processing

4.1 BASIC TYPES AND TECHNIQUES

There are many different types and techniques of performing DoS attacks. The following paragraphs therefore describe the most commonly used ones and identify the means by which

they can be prevented.

4.1.1 Distributed DoS attacks

We speak of distributed DoS attacks when several devices flood the entire bandwidth or resources of the target system, which is usually one or more servers (an example is Figure 4.2). Such an attack is often the result of the use of various systems and devices (such as a so-called botnet) that try to exploit the target system. The botnet is an extensive virtual network of artificial (zombie) computers, the aim of which is to take orders without the owner's knowledge. When the target system consumes all free connections, no more can be established [ZJT13].

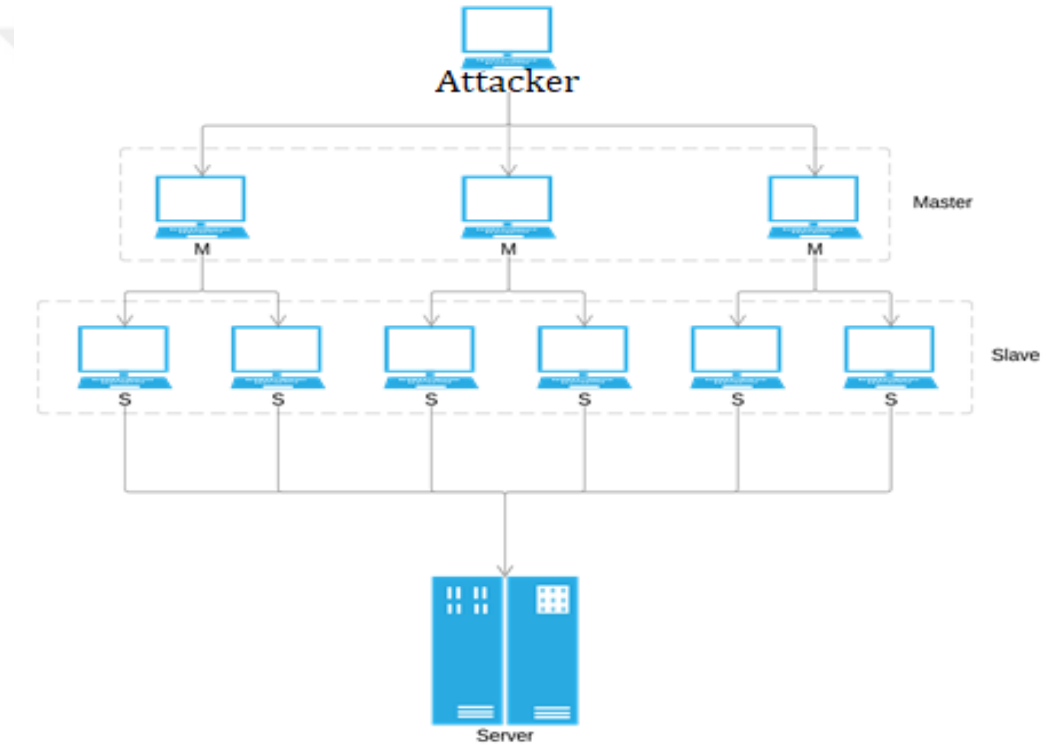

Figure 4.2: DDoS attack distribution scheme, source: own processing

The main advantages of an attacker when using a Distributed DoS attack are the fact that multiple devices can generate more load than one, while the use of a number of systems provides takes care of much more difficult detection of his identity. At the same time, the behavior of each of these devices is less observable, which makes it difficult to defend against the attack itself. These advantages also lead to the development of defense mechanisms. On the target server side, a simple increase in bandwidth above the current attack size will no longer suffice, for example, an attacker could increase the number of devices involved, which would also cause a load and

system outage [ZJT13].

4.1.2 Semantic DoS attacks

Semantic attacks use specific functionality or an implementation error of the application, or the protocol of the victim's device, to exploit a certain amount of its resources. For example, in the case of a TCP SYN attack, this exploited functionality is the allocation of a significant amount of space waiting for connections immediately after the TCP SYN request is acknowledged. An attacker opens multiple connections that he never closes, flooding the server. In a CGI attack, the attacker's goal is to flood the processor with multiple CGI requests in this way. One of the most dangerous attacks is the NAPTHA attack, which focuses on the TCP protocol. Initiates many TCP connections, which then fill all available server resources [MR04].

4.1.3 Reflection and amplification

In a Distributed Reflection Denial of Service (DRDoS) attack, any host IP address is considered to be a reflecting device that returns the packet when a packet is sent. Attack groups purposefully organize the host servers they control to send fake data simulating an infected node to a reflecting server [Buk15]. As a result, the victims are attacked from a significantly larger number of source points, extensively scattered in the network, whereby attackers effectively cut off their connection from the rest of the Internet. The layout of the DRDoS attack is shown in Figure 4.3. An amplification attack is a type of reflective attack where the reflecting server sends many times more responses than it receives. In this way, it also multiplies the load on the connection between the victim and the source node [Pax01].

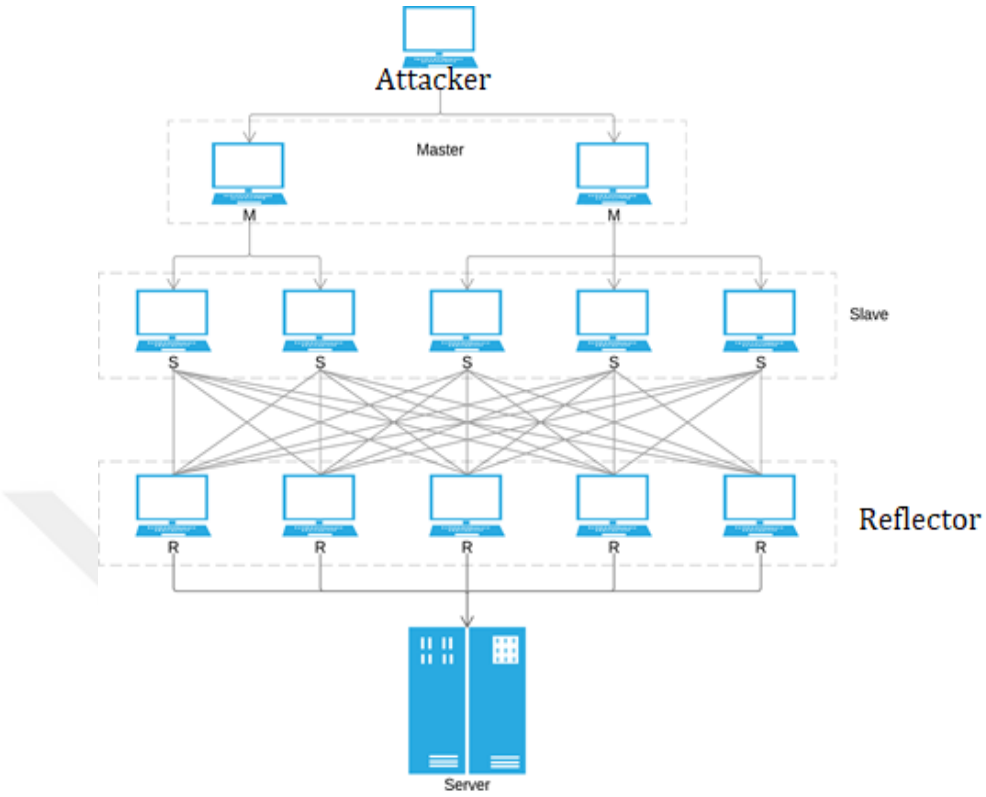

Figure 4.3 : DRDoS attack distribution scheme, source: own processing

4.2 DOS TOOLS AND DOS AS A SERVICE

A typical method of transmitting the mechanisms of DoS and DDoS attacks is malverer. One example was the so-called MyDoom [BBC04]. It is a DoS mechanism that started at a pre-scheduled time. This attack involved setting the value of the target system's IP address to deploy the malware, and no user interaction was required to launch the attack.

Another way to exploit the system for a DDoS attack is to use a hidden piece of third-party software that allows an attacker to download a zombie agent, or the software itself already contains it. An attacker could also penetrate the system by using automated tools that exploit bugs in programs that listen to remote access.

connections. This scenario affects primary systems that act as web servers. A typical example of a DDoS tool in this area is the so-called Stacheldraht [Dit99]. Stacheldraht uses a layered structure in which an attacker uses a client program to connect to handlers that are misused to transmit commands to zombie agents performing the DDoS attack itself. Agents are exploited by attackers through service functions by using automated algorithms to detect vulnerabilities in programs that accept remote connections. Each of the functions is able to control up to a thousand agents.

DDoS tools such as Stacheldraht still use classic DoS methods aimed at amplifying and spoofing IP addresses, such as bandwidth congestion. Another option is to flood resources - SYN Flood attack. Newer tools also use DNS servers for DoS attacks [Dit99].

Tools like MyDoom can be used against any IP address. Less experienced attackers use them to block the availability of popular and well-known web servers. On the contrary, more sophisticated attackers use these tools to blackmail, for example, against their business adversaries [KPM15].

In some cases, however, the device may become part of a DDoS attack intentionally - with the owner's consent. An example is the distributed Operation Payback attack organized by Anonymous [Ley10].

4.3 DOS - SUMMARY

As can be seen from the previous paragraphs, there are a really large number of variations of DoS and DDoS attacks. At the same time, their number is growing over time and therefore it is necessary to develop new tools to defend against them. However, the most difficult part is to detect and identify such an attack, especially its perpetrator. Therefore, one of the uses of the algorithm described in Chapter 5 and the implementation part is this functionality.

5. EXPERIMENT IMPLEMENTATION

The main goal and output of this work is an algorithm defining a unique user identifier. The next chapter deals with the description of its operation, structure, technologies used and subsequent implementation. The information from the HTTP and TCP protocols described in Chapters 2 and 4 was used to create it. Thanks to them, the algorithm is able to recognize and assign activity to each of the users of the host system.

The principle of operation of the algorithm is a trigger action, which is the acceptance of an HTTP request by the host server.1 Based on each such request, its analysis and comparison is then started. The output of the algorithm is the degree of similarity of the current request in terms of its distance from the nearest similar element from the database. Additional functionality is the ability to notify the host application when a large number of similar requests from one user are exceeded.

5.1 ALGORITHM

In general, the whole algorithm consists of five main, consecutive steps:

1. capturing the ongoing request before the actual processing of the host application logic,
2. collection of HTTP and TCP connection information, namely:
   - HTTP data is parsed from the intercepted request,
   - TCP data is provided by a separate network monitor,
3. processing and serialization of information into the form of an object of a unique imprint of the current request,
4. analysis and finding of the nearest similar identifier in the database using the function for searching for the nearest object,The host server is typically a back-end application that contains an algorithm implementation as an independent module.

5. processing of the result in the form of the distance of the impressions and the subsequent reaction:

if it is a new session, a new record is created for the identifier in the database with frequency one, if the degree of similarity exceeds the specified threshold, no new record shall be created; however, the frequency of the fingerprint in the database is increased, if the frequency of a particular fingerprint is very high, there is a possibility to notify the host application.

As shown in detail in Figure 5.1, the analysis is triggered by each new HTTP request, which is followed by identifier creation and comparison actions. At the same time, at the start of the application, a thread is started to perform a network analysis of the transport layer, which stores information from the last few connections. From this HTTP and TCP information, a unique Unique Footprint (UFoo) is created for each request. The current identifier is then compared with each fingerprint in the database. The algorithm for searching for the nearest neighbor will then find the nearest identifier using the comparison function. Depending on the degree of similarity, the algorithm further determines whether it is a request of one user, which causes an increase in its frequency, or it is a new fingerprint, for which it creates a new record. As described above, one of the additional functionalities is the notification of the host application when the frequency limit of requests from one user is exceeded, which could indicate, for example, a DoS or DDoS attack.

5.2 IDENTIFIER STRUCTURE

The fingerprint of user activity is thus a combination of specifically selected HTTP and TCP information. The object consists of two parts - static and relational data. Static data is used by the comparison function to determine the distance between the two identifiers. The distance calculated by this function can be further influenced by relational data relationships that address specific aspects of the selected information, such as the determination of less secure geo-zones by geolocation of the client's IP address.

5.2.1 Static data

The static part of the identifier contains a larger amount of data, so it is stored and represented in the form of a string. It consists of the following three subsections of data for comparison:
static headers - specific preselected and sorted values of HTTP headers, such as: Authorization, Cookie, Content-Length, or User-Agent,
unknown headers - other headers of the given HTTP request, which were not used as static,

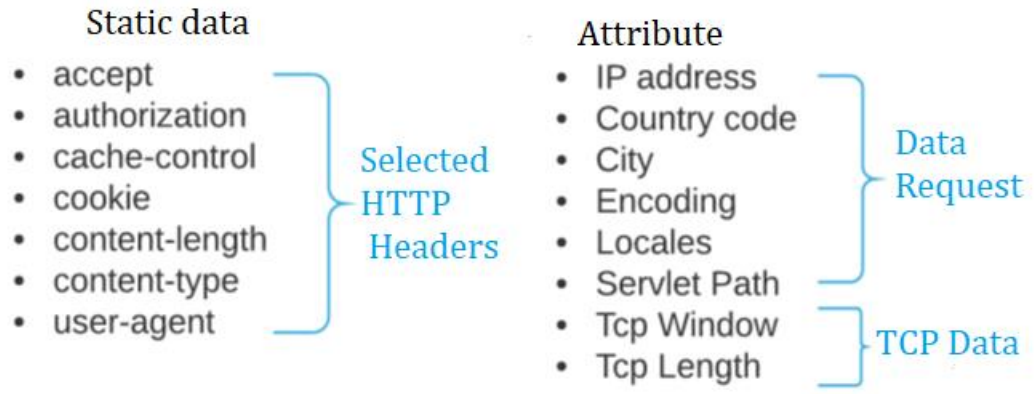

Figure 5.1: Detailed view of the structure and sections of the static data of the identifier representation

attributes - specific information from the HTTP and TCP protocol, such as IP address, geolocation, or for the size of the TCP window.

In summary, Figure 5.2 shows all the information used for each category of static data.

Each of this information, when used in a comparison function, has a purpose and a specific weight that determines its importance. For example, the content of static headers is very important. Conversely, the weight of TCP window values is lower compared to IP address and geolocation, but can play a role in a large number of requests. Subchapter 5.3.2 deals with the comparison function and the method of calculating the similarity of individual parameters in more detail.

5.2.2 Relational data

Relational data is not used to compare and search for similar values. Their purpose is to treat special cases with information that requires it. There is not much such information, so it is not necessary to compress and serialize it into a string as static data. It is therefore an object containing the following items:

● timestamp - the exact time of establishing the connection of the current session,

● geolocation - country of origin determined on the basis of the client's IP address,
relational headers - HTTP headers, such as forwarded, or x-csrf-token.
The individual items are evaluated separately and their results are applied to the distance returned by the similarity algorithm. For example, with a very small time difference between the compared requirements, their distance is additionally reduced. For geolocation, a change in the calculation parameter can be achieved after evaluating the client's request from the country with the most common origin of DoS attacks, such as: China, Korea, or Russia.

5.2.3 Variability of data structure

The structure of static and relational fingerprint data is calibrated in the application after testing and consultation to determine as accurately as possible the similarity of requests from one user. Of course, it is possible to change it in the future, or add new information to it.

5.3 SIMILARITY SEARCH ALGORITHM

As mentioned in the previous paragraphs, the main functionality of the identification process is an algorithm to determine the similarity and search for the nearest neighbor of the current identifier. The input for the algorithm is thus this fingerprint together with the set containing the fingerprints of the requirements analyzed so far. After initialization and reading of the input parameters, a function for calculating their distance is applied to each pair of fingerprints, which returns a value in the range from zero to one. For identical fingerprints, the distance is zero, on the contrary, for fingerprints that differ in many parameters, the value is close to one. The function gradually compares and takes into account the individual values of the static part of the identifier according to their weight and degree of similarity. A detailed description of the operation of this function is dealt with in subchapter 5.3.2.

After processing the results, the algorithm finds the nearest identifier, which together with its distance is returned as output for further processing and possible modification by relational data. Thus, the following pseudo-code formally describes this algorithm and its work with the results of the distance calculation function:

```
algorithm get_nearest_neigbour is
    input: UFooEntity uFooEntity
           UFooEntity[] allStockItems
    output: Result<int distance, UFooEntity nearestNeigbour>

    minDistance ← MAX_INT;
    nearestneigbour ← null;

    for each stockUFooEntity in allStockItems do

        acctualDistance ← getDistance(uFooEntity.staticData,
                          stockUFooEntity.staticData);

        # Relation Data analysis - can affect distance

        if acctualDistance < minDistance then
            minDistance ← acctualDistance;
            nearestneigbour ← stockUFooEntity;
        end if

    return Result<minDistance, nearestneigbour>
```

5.3.1 Procedures used

Before describing the function for calculating fingerprint distances, it is necessary to explain the techniques and concepts that the similarity search algorithm works with. Specifically, in addition to comparing compressed strings for simple attributes, it is mainly a matter of determining the degree of similarity of sets by calculating the so-called Jaccard index. The following paragraphs deal with the closer operation of both techniques.

Jaccard's index

The calculation of the Jaccard coefficient, also known as the penetration and unification ratio, is a function originally described by Paul Jaccard, which is used to compare the similarities and differences of the sets tested. The Jaccard coefficient thus measures the degree of similarity between two finite sets [Zez06]. It is mathematically defined as the ratio of the magnitudes of the unification and intersection of these sets:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \qquad (5.1)$$

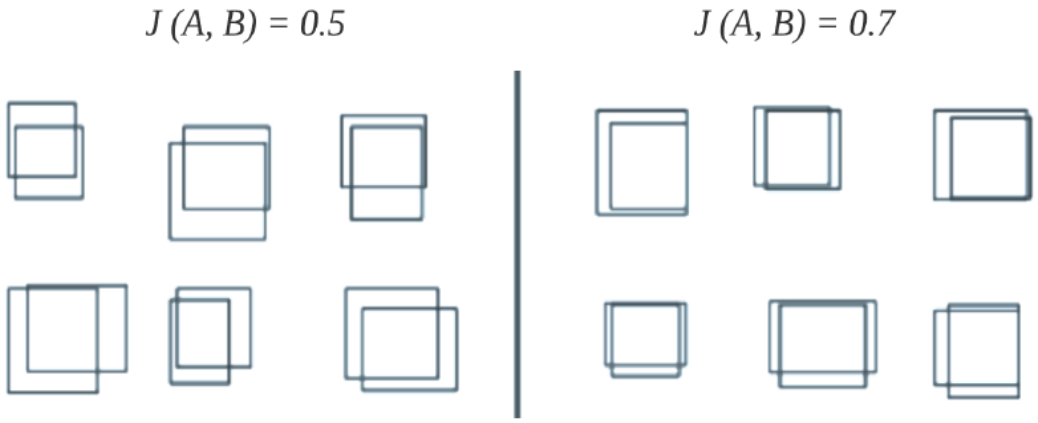

Figure 5.2: Representation of penetration and unification of sets for calculation of Jaccard coefficient, source: [ZD14]

If both sets are empty J (A, B) = 1. The Jaccard distance, which in turn determines the difference between the two sets, is a supplement to the Jaccard coefficient [Zez06; Kos16]. It is therefore calculated by subtracting its value from the unit, or as a proportion, the numerator of which is the difference between unification and penetration and the denominator is the unification of the following sets:

$$d_J(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} \qquad (5.2)$$

$$0 \leq J(A,B) \leq 1 \qquad (5.3)$$

When comparing specific strings for HTTP headers or attributes, such as servletPath or locales, these values are compressed before comparison. Specifically, it is about removing redundant white characters and symbols that could cause inaccuracies in the comparison. The same string compression rules are used when comparing items in the Jaccard index calculation.

5.3.2 Distance calculation function

The input for the distance calculation function is therefore always two representations of

identifiers, specific values in their static part.

The resulting distance is calculated as a supplement to the weighted average of the similarities of the three subsections of this data as follows:

● the Jaccard coefficient for static headers is calculated first,

then the algorithm will use Jaccard's coefficient to calculate the similarity of the remaining headers,

● Finally, the degree of similarity of the individual attributes is calculated.

Each of these parts is weighted by its own constant in the subsequent addition. At the same time, each of the attributes of the third part has its own weight. The specific calibration of the weights of individual items is shown in Figure 5.3.

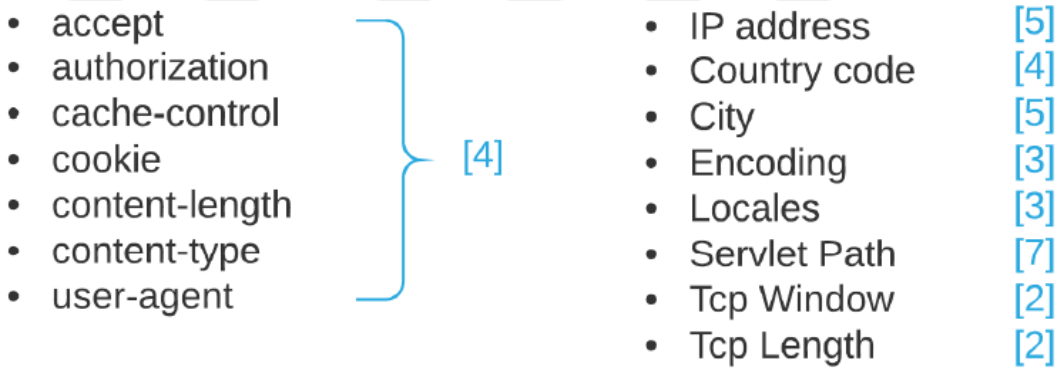

figure. 5.3: Example of calibration of weights of individual parameters of the static part of the fingerprint for distance calculation, source: own processing

Of course, the key calibration of weights and parameters plays a key role in the calculation. The following pseudo-code formally describes the distance calculation function and its operation:

```
algorithm get_distance is
    input: String actualData
           String stockEntityData
    output: double distance

    if actualData = stockEntityData
        return 0
    end if
```

```
# Compute weighted Jaccard for static headers
jaccardStaticHeaders
    ← jaccardIndex(actualData.staticHeaders,
                   stockEntityData.staticHeaders);
jaccardStaticHeaders
    ← jaccardStaticHeaders * SH_WEIGHT;

# Compute weighted Jaccard for unknown headers
jaccardUnknownHeaders
    ← jaccardIndex(actualData.unknownHeaders,
                   stockEntityData.unknownHeaders);
jaccardUnknownHeaders
    ← jaccardUnknownHeaders * UH_WEIGHT;

# Compare values of remaining attributes by similarValue
# function. Every of this attributes will have sub-weight
# which will be computed in attributesIndex.
isSameIp
    ← similarValue(actaulData.IP, stockEntityData.IP)
    * IP_SUB_WEIGHT;
isSameCountry
    ← similarValue(actaulData.country,
                   stockEntityData.country)
    * COUNTRY_SUB_WEIGHT;
...
isSameLength
    ← similarValue(actaulData.length,
                   stockEntityData.length)
    * LENGTH_SUB_WEIGHT;

attributesIndex
    ← ∑(isSameIp, isSameCountry, ..., isSameLength)
     / ∑(IP_SUB_WEIGHT, COUNTRY_SUB_WEIGHT,
         ..., LENGTH_SUB_WEIGHT);
attributesIndex
    ← attributesIndex * ATTR_WEIGHT

similarity
    ← ∑(jaccardStaticHeaders, jaccardUnknownHeaders,
        attributesIndex)
     / ∑(SH_WEIGHT, UH_WEIGHT, ATTR_WEIGHT)

return 1 - similarity;
```

The values of the similarities of the individual parts, which are always in the interval [0; 1], are

thus multiplied by the assigned weights and added. The ratio of the sums of these values and their possible weights is then calculated by the coefficient of similarity of the static data of the input identifiers. The output of the function is the distance, which is a supplement to this coefficient.

5.1 IMPLEMENTATION

In addition to the design of the algorithm, its implementation in Java is a part of this work. This is an accurate implementation of the functionality described in the previous subchapters. The library itself is as universal as possible and independent of a specific framework or technology. After inserting its dependency into the host project, it detects access methods itself and captures individual HTTP requests for further processing. At the same time, at startup, if possible, it starts monitoring the network to supplement the information from the TCP protocol.

5.1.2 Technologies used

For the correct implementation of the functionality of the whole algorithm, it was necessary to use many specific technologies, from the library for monitoring TCP packets, to the geolocation of IP addresses. The following paragraphs describe the most important ones.

Aspect-oriented programming

Aspect-oriented programming (AOP) is a paradigm designed to increase code modularity without disrupting the natural running of the application. It is about adding new functionality to existing code, without having to modify the original source. Instead, a so-called access point is specifically defined, which adds the required functionality to a predetermined location [Kic97].

In the implementation part, the AOP, specifically the AspectJ library, is used for injecting individual access methods of the host application and subsequent capture of information from their HTTP requests. It is thus possible to create access points for any of the frameworks or approaches used [Lad09]. Currently, for testing purposes, the implementation includes an access point for methods with Spring annotation @RestController. However, it is not a problem to extend the interface with any existing or own access point. Another advantage of this approach is the asynchrony of the whole calculation. After capturing the information from the HTTP request, the

process of its processing is performed in a new thread. The logic of the host application is thus not delayed and can continue in the standard way.

In order to avoid system congestion when possible slowing down the operation and analyzing a large number of new requests, the implementation uses a constant number of threads for processing them. Currently, this number is set to eight. However, the value can be easily adapted to the conditions of a particular host server. In the future, it is possible to optimize this functionality, for example, by including requirements in the line and subsequent gradual processing.

Network analysis

When the host application is started, monitoring, analysis and parsing of TCP packets is started by default as one of the modules in a separate thread. The native libpcap library and the tcpdump program are used to access this information. Of course, these options must be enabled on the host server in advance, so there is a possibility to use the application without monitoring network activity. However, calculations of the distances of the identifiers that contain this information are more accurate and produce fewer false-positive detections.

IP Geolocation

As described in Section 5.2, one of the attributes that are considered when evaluating the similarity of identifiers is the client's geolocation. Specifically, it is a country determined by its IP address. The source of this information is the database provided by the MaxMind GeoIP service.

In addition to this database, the system also uses a static list of less secure zones. These are specific countries that are the most common cause of DoS and DDoS attacks. Their detection can result, for example, in a lowering of the threshold for notifying the host application when a large number of similar fingerprints are exceeded.

5.1.2 Functionality and code structure

The following section describes the entire course of the implemented request similarity analysis, which is also graphically illustrated in the sequence diagram in Appendix B.

The request processing action begins in the Injector class. This class contains AOPs for each method of the host application. It also serves to manage dependencies and initialize single-ton threads of the PacketStream class. PacketStream and other helper classes of the endpoint.collector.tcp package provide, if possible, parsing and management of received TCP packets. These are then available as a map of the last n intercepted packets in the form of a set of objects of the TCPFootprint class and their IP addresses.
From the Injector, the data always continues in a separate thread for processing by the RequestHandler class and the other classes of the endpoint.collector.http package. Their purpose is to extract the necessary HTTP information, which is then represented by the HTTPFootprint class.

The UFooProcessor class also works with the HTTPFootprint and TCPFootprint representations (available in Appendix A). This class handles the main logic of analyzing the similarity of the currently processed request with other identifiers in the database. UFooProcessor uses the methods of the Serializer class to process the information into the final form of the UFoOEntity object, which is a representation of the unique identifier itself. UFooEntity contains all extracted static and relational data.

However, the most important part of the whole process is the similarity analysis performed by the FootprintSimilairtyService class, which implements the nearest neighbor search algorithm formally described in section 5.3. Its getNearestNeighbour, computeDistance, and jaccardIndex methods therefore compute the final result in the form of a distance and the nearest similar identifier from the database. Based on this result, UFooProcessor then decides whether it is an activity of an unknown user and creates a new record in the database for the current fingerprint, or a recurring session, thereby increasing its frequency or alerting the host application to possible exceeding its limit.

For the purposes of testing and calibration of algorithm parameters, the practical part of the work contains, in addition to the library itself, also an example of the host application in which the library is used. This server also provides the ability to view simple statistics from similarity analysis.

6. EXPERIMENTAL RESULTS

The algorithm and its implementation described above were tested and calibrated to simulate a real DDoS attack. 1 The following paragraphs describe both the course of this attack, as well as the course and results of its simulation.

6.1 ATTACK PARAMETERS

The attack on which the subsequent simulation is based was relatively large and well organized. It lasted a total of 56 minutes and was attended by clients from 136 different IP addresses. During these 56 minutes, the server received a total of 1,345,446 requests, all directed to the same URL and one specific action. Another feature was the relatively low diversity of HTTP header content. The individual variants of headers differed only in some values, or slightly changed their subset of items. Examples of these headers for some of the requirements are shown in the following diagram:

```
[
    "Connection": "keep-alive",
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; )...",
    "Content-Type": "application/json; charset=UTF-8",
    "Host": "sample-domain.com",
    "Accept": "application/json, text/javascript, q=0.01",
    "Origin": "http://sample-domain.com",
    "Cookie": "__gvf=DN-24; _be=DV1.2; SampleTool...",
    "Accept-Encoding": "gzip, deflate",
    "Referer": "http://sample-domain.com/pages/homepage...",
    "X-Requested-With": "XMLHttpRequest",
    "Accept-Language": "en-US,en;q=0.8"
]
```

Thus, a large number of devices were used for this attack, which gradually sent the required data. However, all requests contained a similar group of HTTP headers, with various variations, which helped to better identify it.

6.2 SIMULATION PARAMETERS

The simulation of the attack described above tried to simulate it as faithfully as possible. Of

course, it was not possible to do it in full. Therefore, a standard load of 25,000 requests was created for the test server. These requirements tried to imitate real users and their activity as closely as possible in their content. In parallel, content simulating the mentioned DDoS attack in the range of 13,454 requests from 13 different IP addresses was sent to the server within a few tens of minutes. Other parameters of the original attack and the content of HTTP headers have been preserved.

The purpose of this simulation was to verify the correct operation of the algorithm, in particular its ability to identify and distinguish the activity of DDoS attack users from standard sessions.

6.3 INTERPRETATION OF RESULTS

The following graphs show the results of the simulated attack compared to the activity of regular users. All results, as well as their graphical representation, were produced by an application using the implementation of the library for the purpose of testing the algorithm, which is part of the practical part of the work.

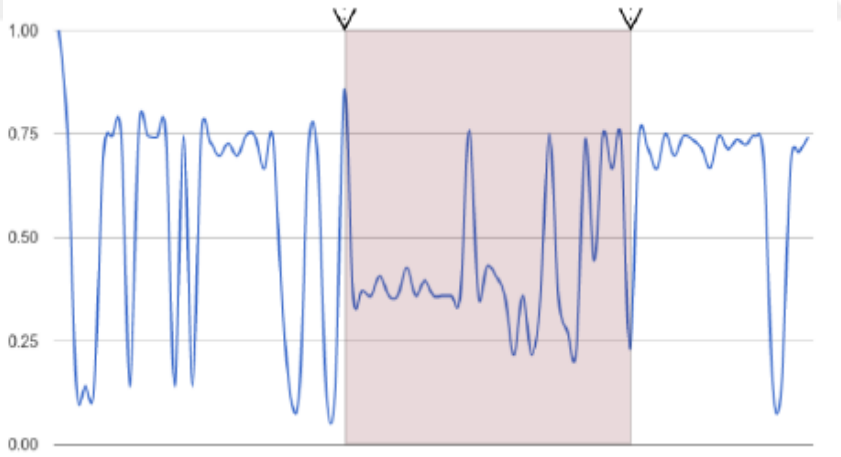

Figure 6.1: Distance of processed simulation requests with limitation of DDoS attack duration in time, source: implementation part of the work

Figure 6.1 shows the results of the calculation of the distances of successive requirements2 over time. The marked section indicates the beginning and end of the simulated DDoS attack. Histogram 6.2 further shows the exact distribution of the number of distances of the whole simulation. At the same time, it color-codes the distances calculated for DDoS attack

requirements from the distances of real users.

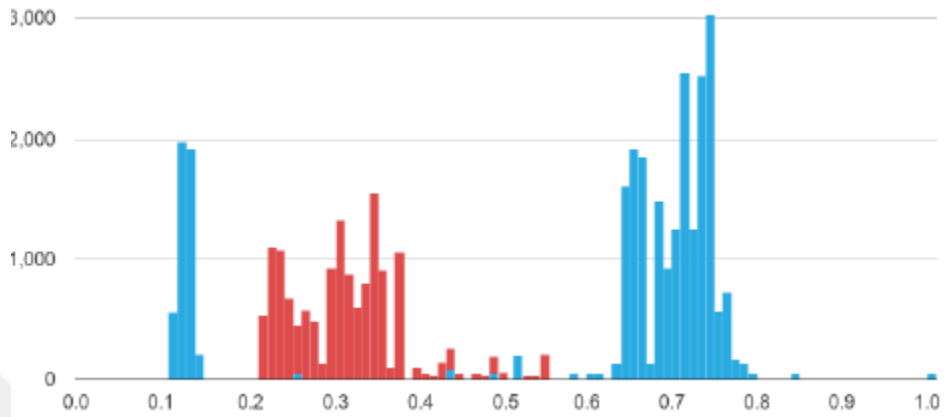

Figure 6.2: Histogram of the number of distances of simulation requests.

The blue color represents the distances of real users, the red the activity of the DDoS attack, the source: the implementation part of the work

6.4 SUMMARY

From the simulation results, especially from the results displayed in the histogram 6.2 it can be seen that the activity of DDoS attack requests can be distinguished from the activity of real users by measuring the distance of individual requests. While the simulation distances of real user activity are in most cases in the intervals [0.1,0.15] and [0.6,0.8], most of the simulation distances of DDoS attack activity are in the interval [0.2,0.4]. There is also an interval [0.4,0.6]

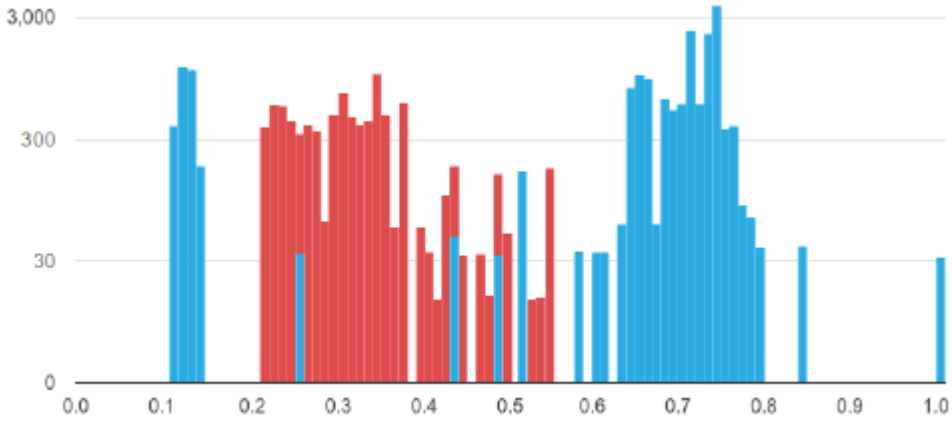

Figure 6.3: Histogram of the number of distances of simulation requests using the logarithmic y-

axis, source: implementation part of the work

in which the two types intersect in smaller quantities. Histogram 6.3 uses the logarithmic y-axis to better represent the mean interval. According to these data, it is therefore possible to calibrate the threshold for alerting the host application to a possible threat.

7. CONCLUSION

The results of tests with real data proved the functionality of the algorithm and its ability to identify users even during the simulation of an ongoing DoS attack. The original idea of detecting DoS attack requirements accounted for a large number of resulting distances in the middle of the interval for the attacker and a large number of marginal distances for normal loads. In reality, as can be seen in histogram 6.2, the simulation of real user requests did indeed show maxima at the distance margins and the simulation of DoS attack requests at the mean distance values. This layout is probably due to the way in which the attacker, unlike regular users, communicated with the server. Ordinary users sent relatively different requests to each other, while the requests of one user were similar in their session. This caused them to be distributed over the marginal values of the distances. However, the attacker used one type of request all the time, but he kept changing it. Therefore, the distribution of its distances is in the mean values of the scale.

Based on these results, it is possible to set thresholds for determining the attack in the interval of medium distances of the attacker, according to the results of the simulation.

I was very interested in this issue, so I plan to further develop the work and add other interesting functionality from the possibility of sharing a database of fingerprints, to improve the monitoring and use of TCP information.

REFERENCES

[1]. BERNERS-LEE, Timothy. *Biography* [online] [visited on 2020-09-21]. Available from: www.w3.org/People/Berners-Lee/.

[2]. FIELDING, Roy; RESCHKE, Julian. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing* [RFC 7230]. 2014. ISSN 2070-1721. Available also from: tools.ietf.org/html/rfc7230.

[3]. FIELDING, Roy; RESCHKE, Julian. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content* [RFC 7231]. 2014. ISSN 2070-1721. Available also from: tools.ietf.org/html/rfc7231.

[4]. KUROSE, James F.; ROSS, Keith W. *Computer networking: A Top-Down Approach*. 6th ed. Addison-Wesley, 2013. ISBN 978-0-13-285620-1.

[5]. BERNERS-LEE, Timothy; FIELDING, Roy; MASINTER, Larry. *Uni- form Resource Identifier (URI): Generic Syntax* [RFC 3986]. 2005. Avail- able also from: tools.ietf.org/html/rfc3986.

[6]. GOURLEY, David; TOTTY, Brian; SAYER, Marjorie, et al. *HTTP: The Definitive Guidee*. O'Reilly Media, 2002. ISBN 978-1-56592-509-0.

[7]. RESCHKE, Julian. *The 'Basic' HTTP Authentication Scheme* [RFC 7617]. 2015. Available also from: tools.ietf.org/html/rfc7617.

[8]. SHEKH-YUSEF, Rifaat; AHRENS, David; BREMER, Sophie. *HTTP Digest Access Authentication* [RFC 7616]. 2015. Available also from: tools.ietf.org/html/rfc7616.

[9]. FIELDING, Roy T.; RESCHKE, Julian. *Hypertext Transfer Protocol (HTTP/1.1): Authentication* [RFC 7235]. 2014. Available also from: tools.ietf.org/html/rfc7235.

[10]. SURVEYS, World Wide Web Technology (ed.). *Usage of HTTP/2 for websites* [online] [visited on 2020-09-21]. Available from: w3techs. com/technologies/details/ce-http2/all/all.

[11]. MOGUL, Jeffrey; MASINTER, Larry M.; FIELDING, Roy T., et al. *Hy- pertext Transfer Protocol – HTTP/1.1* [RFC 2616]. 1999. Available also from: tools.ietf.org/html/rfc2616.

[12]. *Detailed GTA Online Network Analysis for PS3* [online] [visited on 2020-09-21]. Available from: www .reddit.com /r/gamedev /comments / 1pg59e/detailed_gta_online_network_analysis_for_ps3 On- line forum comment.

[13]. MICROSOFT CORPORATION. *WOW64 Implementation Details* [on- line] [visited on 2017-03-02]. Available from: msdn.microsoft.com/ en-us/library/windows/desktop/aa384274(v=vs.85).aspx.

[14]. LARDINOIS, Frederic. *Google Forks WebKit And Launches Blink, A New Rendering Engine That Will Soon Power Chrome And Chrome OS* [on- line]. 2013 [visited on 2020-09-21]. Available from: techcrunch.com/ 2013/04/03/google-forks-webkit-and-launches-blink-its-own-rendering-engine-that-will-soon-power-chrome-and- chromeos.

[15]. SILBEY, Marc. *Introducing IE9's User Agent String* [online]. 2010 [visited on 22020-09-21]. Available from: blogs.msdn.microsoft.com/ ie/2010/03/23/introducing-ie9s-user-agent-string.

[16]. ZAKAS, Nicholas C. *History of the user-agent string* [online]. 2010 [visited on 2020-09-21]. Available from: www . nczonline . net / blog / 2010/01/12/history-of-the-user-agent-string.

[17]. BARTOŠ, Jiří. *Windows 10: analýza probíhající komunikace* [online]. 2015 [visited on 2020-09-21]. Available from: www . root . cz / clanky / windows-10-analyza-probihajici-komunikace.

[18]. MICROSOFT CORPORATION. *Windows Push Notification Services (WNS) overview* [online]. 2017 [visited on 2020-09-21]. Available from: docs . microsoft . com / en - us / windows / uwp / controls - and - patterns / tiles - and - notifications - windows - push - notification-services--wns--overview.

[19]. *Tcpdump & Libpcap* [online] [visited on 2020-09-21]. Available from: www.tcpdump.org.

[20]. *Wireshark* [online] [visited on 2020-09-21]. Available from: www . wireshark.org.

[21]. *The Bro Network Security Monitor* [online] [visited on 2020-09-21]. Avail- able from: www.bro.org.

[22]. CLAISE, Benoit. *Cisco Systems NetFlow Services Export Version 9* [RFC 3954]. 2004. Available also from: tools.ietf.org/html/rfc3954.

[23]. CLAISE, Benoit; TRAMMELL, Brian; AITKEN, Paul. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information* [RFC 7011]. 2013. Available also from: tools.ietf.org/ html/rfc7011. RFC.

[24]. *NetFlow* [Wikipedia, the free encyclopedia] [visited on 2020-09-21]. Available from: en.wikipedia.org/wiki/NetFlow.

[25]. SVOBODA, Jakub; GHAFIR, Ibrahim; PRENOSIL, Vaclav. Network Monitoring Approaches: An Overview. *International Journal of Ad- vances in Computer Networks and Its Security*. 2015, vol. 5, no. I.

[26]. CERT NETSA SECURITY SUITE. *YAF* [online] [visited on 2020-09-21]. Available from: tools.netsa.cert.org/yaf.

[27]. NTOP. *Collecting Proprietary Flows with nProbe* [online] [visited on 2020-09-21]. Available from: www.ntop.org/nprobe/collecting- proprietary-flows-with-nprobe.

[28]. *Flowmon Probe* [online] [visited on 2020-08-21]. Available from: www. flowmon.com/en/products/flowmon/probe.

[29]. *NFDUMP - Netflow processing tools* [online] [visited on 2020-09-21]. Available from: sourceforge.net/projects/nfdump/.

[30]. *ipfixcol* [online] [visited on 2020-08-21]. Available from: github.com/ CESNET/ipfixcol.

[31]. VELAN, Petr. *fbitdump* [online] [visited on 2020-09-21]. Available from: github.com/CESNET/ipfixcol/tree/master/tools/fbitdump.

[32]. *Wordpress* [online] [visited on 2020-09-21]. Available from: wordpress. com.

[33]. KINABLE, Joris. Detection of network scan attacks using flow data. In: *9th Twente Student Conference on IT, 23th June*. 2008.

[34]. MEUCCI, Mateo; KEARY, Eoin; CUTHBERT, Daniel, et al. *Owasp testing guide v3* [OWASP Foundation]. 2008.

[35]. HUSAK, Martin; VELAN, Petr; VYKOPAL, Jan. Security Monitoring of HTTP Traffic Using Extended Flows. In: *Availability, Reliability and Security (ARES), 2015 10th International*

Conference on. 2015, pp. 258– 265.

[36]. TOORN, Olivier van der; HOFSTEDE, Rick; JONKER, Mattijs, et al. A First Look at HTTP(S) Intrusion Detection using NetFlow/IPFIX. *In- tegrated Network Management (IM), 2015 IFIP/IEEE International Sym- posium on*. 2015, pp. 862–865.

[37]. CID, Daniel. *New Brute Force Attacks Exploiting XMLRPC in WordPress*. Available also from: blog.sucuri.net/2014/07/new-brute-force- attacks-exploiting-xmlrpc-in-wordpress.html cit. 2020-08-21.

[38]. EVERITT, Brian. *The Cambridge dictionary of statistics*. Cambridge Uni- versity Press, 1998.

[39]. LUKO, Stephen N. *What Are Z-Scores?* Available also from: www.astm. org/SNEWS/MA_2011/datapoints_ma11.html cit. 2017-04-02.

[40]. CID, Daniel. *Brute Force Amplification Attacks Against WordPress XML- RPC*. Available also from: blog.sucuri.net/2015/10/brute- force-amplification-attacks-against-wordpress-xmlrpc. html cit. 2017-04-03.

[41]. ESET REASERCH. *Sathurbot: Distributed WordPress password attack* [on- line] [visited on 2020-08-21]. Available from: www.welivesecurity. com/2017/04/06/sathurbot-distributed-wordpress-password- attack.

[42]. FORRISTAL, Jeff. NT Web Technology Vulnerabilities. *Phrack Magazine*. 1998, vol. 8, no. 54.

[43]. *OWASP* [online] [visited on 2020-08-21]. Available from: www.owasp. org/index.php/Main_Page.

[44]. HALFOND, William G.J.; VIEGAS, Jeremy; ORSO, Alessandro. A Classification of SQL Injection Attacks and Countermeasures. In: *Proceedings of the IEEE International Symposium on Secure Software Engineering*. 2006, vol. 1, pp. 13–15.

[45]. NETWORK, YAHOO! developer. *Yahoo Query Language (YQL)* [online] [visited on 2020-08-21]. Available from: developer.yahoo.com/yql.

[46]. HANSEN, Robert. *XSS Filter Evasion Cheat Sheet* [online] [visited on 2020-08-21]. Available from: www.owasp.org/index.php/XSS_ Filter_Evasion_Cheat_Sheet#Null_breaks_up _JavaScript_ directive.

[47]. WESTERGREN, Randy. *Widespread XSS Vulnerabilities in Ad Network Code Affecting Top*

Tier Publishers, Retailers [online] [visited on 2020-09-21]. Available from: randywestergren.com/widespread-xss-vulnerabilities-ad-network-code-affecting-top-tier-publishers-retailers.

[48]. MANNERS, Darren. The user agent field: Analyzing and detecting the abnormal or malicious in your organization. *SANS Institute reading room site*. 2011.

[49]. KHEIR, Nizar. Analyzing HTTP User Agent Anomalies for Malware Detection. In: *Data Privacy Management and Autonomous Spontaneous Security*. Springer, 2013, pp. 187–200.

[50]. GRILL, Martin. Combining Network Anomaly Detectors. 2016.

[51]. *Emerging Threats* [online] [visited on 2020-08-21]. Available from: rules.emergingthreats.net.

[52]. *4G Series: The Ultimate User-Agent Blacklist, Featuring Over 1200 Bad Bots* [online] [visited on 2020-08-21]. Available from: perishablepress.com/4g-ultimate-user-agent-blacklist.

[53]. *Internet Archive WayBack Machine* [online] [visited on 2020-08-21]. Available from: archive.org.

[54]. *Alexa's Web and Site Audit Crawlers* [online] [visited on 2020-08-21]. Available from: support.alexa.com/hc/en-us/articles/200450194-Alexa-s-Web-and-Site-Audit-Crawlers.

[55]. ZAHARIA, Andra. *Why are Java's Vulnerabilities One of the Biggest Secu- rity Holes on Your Computer?* [online] [visited on 2020-08-21]. Avail- able from: heimdalsecurity.com/blog/java-biggest-security-hole-your-computer.

[56]. FADILPAŠIĆ, Sead. *NSA and GCHQ reverse-engineered Kasper- sky Labs* [online] [visited on 2020-08-21]. Available from: www.itproportal.com/2015/06/23/nsa-and-gchq-reverse- engineered-kaspersky-labs.

[57]. ANDERSON, Blake; PAUL, Subharthi; MCGREW, David. Decipher- ing Malware's use of TLS (without Decryption). *arXiv preprint arXiv:1607.01639*. 2016.