University of Piraeus

School of Information and Communication Technologies

Department of Digital Systems

Postgraduate Program of Studies

MSc Digital Systems Security

Firewall & WAF – Analysis & Implementation of a Machine Learning Integrated Solution

M.Sc. Digital Systems Security Thesis

Supervisor Professor: Christos Xenakis

Name-Surname	E-mail	Student ID.
Georgios Angelakis	gaggelakis@ssl-unipi.gr	MTE1901

Piraeus

25/02/2022

This document reflects the results of a study that has been prepared on behalf of the Postgraduate Program “Digital Systems Security” at University of Piraeus. The information and conclusions contained in this thesis express the author’s personal opinion and arguments, and therefore should not be interpreted that they represent the official concepts of University of Piraeus.

Abstract

In response to the increased challenges the modern network & web application security landscape arises, the development of Firewalls and Web Application Firewalls systems taking advantage of the Machine Learning technology and the intelligence it provides has already been initiated by multiple vendors investing money and manpower on it. Motivated by the above statement, a security solution making use of machine learning technology will be examined. Main objective of this thesis, except from a holistic analysis of concepts such as network, web application, security and machine learning, will also attempt the development, implementation and evaluation of a machine learning – integrated WAF. More specifically, it will begin from the phase of parsing HTTP Requests, extract the characteristics that will assist on their classification, import them on a proposed classifier and build/train the corresponding model which will properly evaluated. Finally, the resulting model will be integrated with a reverse proxy and through the classification process will detect and mitigate malicious-identified requests.

Keywords: Network, Web Application, Security, Machine Learning, Firewall, WAF, Classification

Contents

List of Figures

1 Introduction

Nowadays the constant growth of IT systems, commercial services and apps along with the continuously connection of new users and devices to the Internet increased significantly the complexity and needs of modern networks. In addition, the often non-secure by design software development has led to web applications with multiple security flaws and vulnerabilities. At the same time, since now users' financial and sensitive data is constantly being exchanged online, the attraction of adversaries trying to intercept it is a fact. In this highly demanding and upscaling security landscape, the traditional network and application security systems are, as for now, considered outdated and inadequate. In response to this phenomenon, adoption of modern technologies is required. Machine Learning is considered the next big thing, in terms of technology and capabilities, across the IT community especially due to its applicability on various sectors of everyday operations.

This thesis will proceed with an overview of the current networks and web applications ecosystems, the analysis of the Firewall systems having the role of protecting those environments and their evolution over time with the extension of their features to more efficiently secure web applications. Afterwards, a deep dive presentation of Machine Learning structure, components and capabilities will follow. As last part of the project, the development and evaluation of a web application firewall system will take place which, by properly analysing the traversing data and taking advantage of the Machine learning intelligence, should be capable of identifying malicious HTTP requests and prevent them from reaching the target web server.

2 Theoretical Background

Purpose of the current chapter is the analysis of all the key components required to better understand the role and the operation of Firewall and Web Application Firewall systems inside the ecosystem of an organization's network. Moreover, an introduction to machine learning sector and techniques will be conducted to examine how it can reinforce the aforementioned systems in enhancing network and application security.

2.1 Network

The first prerequisite that needs to be fulfilled is to understand what the exact definition of the term "network" is. A network, in short terms, is a group of devices (workstations, servers, "smart" devices, etc.), also referred as nodes, which are connected through a shared medium and exchange data and resources.

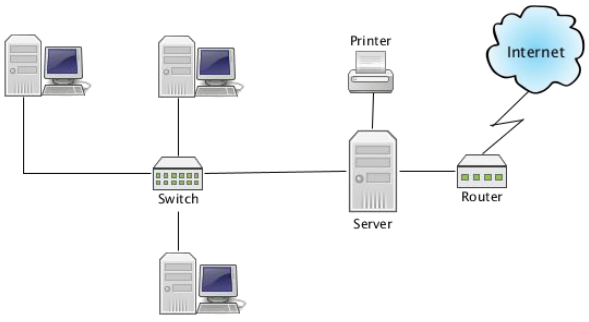

Figure 1: Simple Network Diagram

As presented in *Figure 1*, a network can consist of different device types based on different Operating Systems and Hardware. In order all these devices to properly communicate specific protocols, also known as network protocols, should be followed. These protocols provide multiple services such as identifying each network node by proper addressing, establish and maintain connection between nodes, control the data flows of the established connections and ensure that the transmitted data is sent/received correctly in the appropriate format.

Networks are divided into three different types based on their size and structure, Local, Metropolitan and Wide Area Networks. Local Area Networks, or LANs, constitute networks where devices are located in a relatively small geographical area and are interconnecting through ethernet cables or wirelessly through Wireless Access Points (WAPs). Figure 1 is a characteristic example of a typical LAN topology. Metropolitan networks, or MAN, cover a larger geographical area such as a city or a university campus interconnecting different LANs

through point-to-point connections. Finally, a Wide Area Network (WAN), as presented in Figure 2, serves the communication between the different MANs and LANs across different cities, countries and regions. Internet constitutes the biggest WAN being used in nowadays.

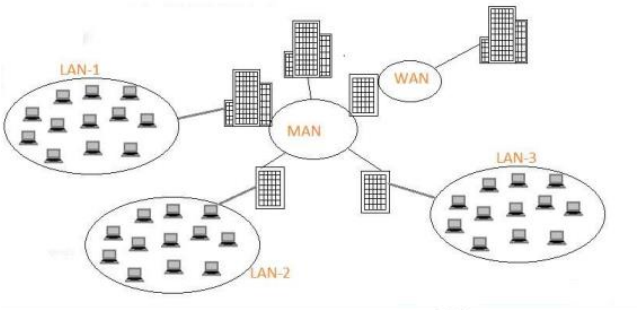

Figure 2: LAN, MAN & WAN topology

2.2 Network Security

2.2.1 Security Requirements

Recent studies[1] pointed that data is currently the most valuable resource, surpassing oil, in global economy and, since networks serve the data exchange and storage, their security acquires major importance. By referring to a network's security we imply the provision of measures and controls to protect three key security objectives, Confidentiality-Integrity and Availability. These security objectives are also referred as the CIA Triad. Confidentiality defines the requirement that the exchanged data in a network environment is accessed only by appropriate authorized entities. The requirement of Integrity demands the non-unauthorized process of data whereas Availability requires that the provided data or service should be accessible any time needed to the corresponding authorized entities.

Despite the above key security objectives, further concerns are raised when referring to a network's security such as accountability, meaning the need to be able to trace an entity's actions, and authenticity, meaning the capability of properly identifying each entity in a network environment to avoid impersonation incidents.

2.2.2 Attack Types & Threats

After analyzing the principles of network security is mandatory to report how they could be compromised by possible adversaries. Attacks, threating a network's security, are divided into two main types:

Passive attacks

[1] https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data

In these types of attacks, the attacker aims on gathering information of interest regarding the network and the data transmitted among it. This is achieved by using tactics such as eavesdropping and traffic analysis where the attacker, after successfully gaining access to the network, with the use of packet sniffing tools interprets the traffic. If network's security is poor, confidential information disclosure, as shown in Figure 3, can be achieved or network reconnaissance could get easier giving the attacker the capability to conduct an active type of attack which will be analyzed next. The paradox of passive attack technics is that their interaction with the network is considered as normal by the rest of the nodes making them hard to detect and mitigate.

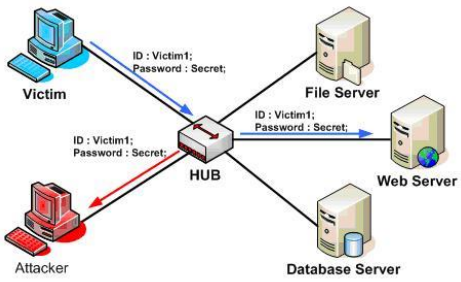

Figure 3: Eavesdropping could lead in credentials disclosure

Active Attacks

After the adversary gained all the available information needed regarding the victim network most probably will proceed to a more aggressive interaction with it. There are several active attack type technics, although four are the most common to be conducted.

i. Masquerade

In this attack technic the adversary spoofs the identity of another node residing on the network. Devices that their identity is more likely to be spoofed is the network's default gateway, the DHCP and the DNS server.

ARP spoofing tools mostly used for this purpose are Arpspoof, Cain & Abel and Arpoison. Masquerade, most of the times, is the stepping stone of a Man in the Middle Attack (MITM) which consists a serious challenge to the network's data confidentiality and integrity.

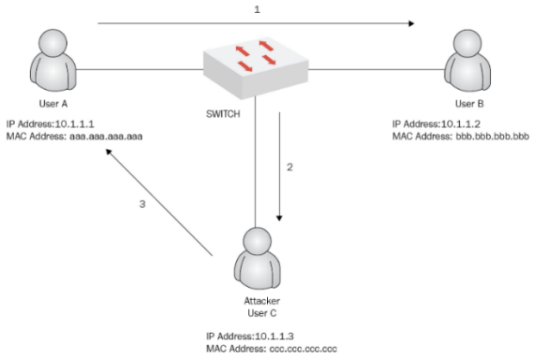

Figure 4: ARP Spoofing Attack

ii. Replay

In Replay attacks the adversary monitors the network traffic, captures network packets and is able of retransmitting them. This attack is used to conduct session hijacking attacks by retransmitting previously-capture valid authentication requests to a, for example, web-banking portal.

iii. Modification

A common factor of the two previous active attack types is that they did not alter the transmitted network packets. In the contrary, a modification attack involves a packet's modification either by changing its TCP header in order to redirect it to another destination or the packet's payload.

iv. Denial of Service

Denial of Service attacks, also known as DoS, are attacks that target the objective of availability of a service by attempting exhausting its resources such as the available bandwidth or the host's available memory and CPU. These attacks can be conducted by several OSI network layers such as Layer 3 (IP session floods), Layer 4 (TCP SYN floods, ICMP floods) and Layer 7 (HTTP Requests floods).

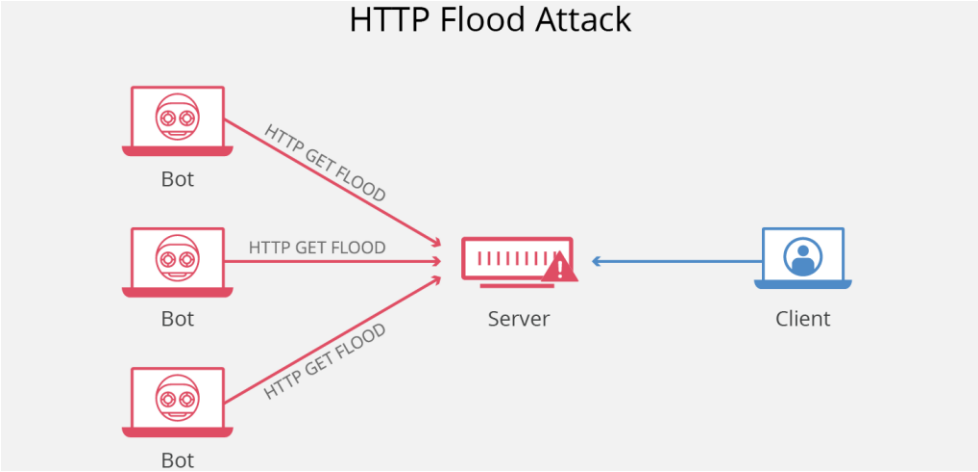

Figure 5: DoS Attack – HTTP Request Flood

2.3 Firewall

2.3.1 Overview

Firewall consists the main guardian of an enterprise's network security as it provides the first layer of protection related to both traffic from untrusted sources as the Internet and internal traffic from different virtual local area networks (VLANs) wishing to intercommunicate. To achieve that, the firewall is most commonly configured as the organization's network gateway in order all its traffic pass through it. To apply network security, a firewall supports several functions and features such as packet filtering, application control, logging and many more who will further analyzed later on this report. A firewall can be implemented as a software in general purpose hardware, as a dedicated hardware appliance or as virtual appliance hosted by a supervisor. The firewall implementation and its specifications are related to the organization's network needs and requirements (number of hosts, network throughput, bandwidth, etc.).

The first firewall concepts appeared in the late 80s[2] offering a basic packet filtering operation by Jeff Mogul of Digital Equipment Corp., continued their development with the addition of stateful capability on the early 90s by the AT&T Bell Labs and started to evolve at their final form on 2004 where Unified Threat Management (UTM) features and tools begun to be included to their arsenal. After 2015, several vendors introduced the concepts of the Next-Generation Firewalls expanding the UTM features and adding Intelligence and Machine Learning capabilities on their offerings.

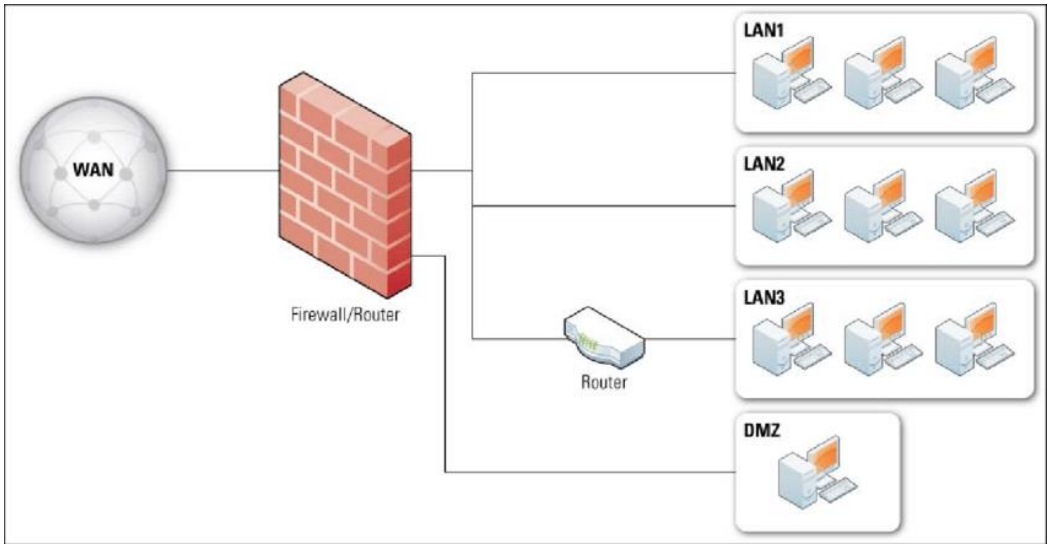

Figure 6: A typical firewall-protected network topology

[2] https://ostec.blog/en/perimeter/firewall/

2.3.2 Traditional Firewall Functions

First, let's examine the functions and services a common firewall can provide:

Packet Filtering

The most basic and vital function of a firewall system is the ability of traffic management and control which is conducted through packet filtering. To apply packet filtering a firewall makes use of a set of configured security policies or rules. Each packet, passing thought the firewall's packet filtering mechanism, is examined according criteria like source IP/Subnet, destination IP/Subnet, protocol (HTTP, FTP, DNS, etc.) and if a security policy exists matching those criteria, then the configured action (permit, drop, discard) is applied. If a security policy does not exist for the packet's criteria, then according to network security best practices, an "implicit deny" rule is matched and the packet is dropped.

Stateful Inspection

An important characteristic of modern firewalls is the ability of stateful packet inspection, meaning the categorization of packets into sessions based on same characteristics such as source/destination IPs, port and protocol. Stateful packet inspection consists on the saving of firewall's resources because only the first packet of each session is evaluated to determine if the session will be permitted or dropped. Also, there is no need of defining a separate security policy allowing the traffic from, the initially, destination IP to the source.

Network Address Translation

The arising issue of the available IPv4 addresses shortage, because of the constantly incrementing level of devices connected to the Internet, led to the development of a mechanism to limit IPv4 addressing allocation. In order for this mechanism to function, the total available IPv4 addresses were divided in two big categories, the addresses that can be freely allocated on Local Area Network devices as described by IETF (Internet Engineering Task Force) in RFC-1918 [3]and the addresses, also known as Public IPv4 Addresses (RFC-1366[4]), that are allocated by the ISPs to enterprise and home networks to provide Internet access since they are routable through the Internet in contrary with the private ones. NAT is an IP masquerade technique used to hide private IP addresses range behind a routing device (such as a firewall) which converts the source local IP addresses into certain public IP address/es for the establishment of connections and services with the rest of the Internet. There are 3 different types of NAT as described below:

[3] https://tools.ietf.org/html/rfc1918
[4] https://www.rfc-editor.org/rfc/rfc1366.txt

- **Port Address Translation (PAT)**

In PAT, the firewall:
i. Receives the packets of the internal host
ii. Checks the packet for host's IP address (e.g. 192.168.1.5) and port number (e.g 8321)
iii. Replaces the host's IP address with the public address provided by the ISP
iv. Adds the host IP address and port to a NAT table to keep a record of all the processed IPs and ports
v. Sends the packet, which now has as source address the public IP address, to the destination address as was originated from the internal host
vi. When the destination address responds the routing device checks its NAT table, replaces its public IP address with the host's IP address and forwards the response packet to the internal network.

- **Dynamic NAT**

Dynamic NAT requires more than one public IP to have been purchased from the ISP, also known as public IP Address Pool.
When the firewall receives a packet from the internal network the same procedure, as in PAT, is followed with the difference that the internal IP address is substituted with one of the available IP addresses from the Address Pool. This substitution can be performed either automatically or manually with configured rules.

- **Static NAT**

Finally, Static NAT requires from the network administrator to configure exactly at the routing device which private address and which port will be substituted from specific public IP address and port. In other words, in static NAT, the NAT table of the firewall is manually configured by the user.

Application-Level Gateway

A common firewall is able of relaying application level (OSI Layer 7) requests from hosts of the protected (internal) network addressing to hosts or servers of untrusted networks such Internet to deliver the proper function of an application or service. In this case, the firewall acts as a proxy server between the application client and the application server, establishing separate connections with them and applying packet-forwarding decisions providing and extra security layer of the internal hosts.

2.3.3 Next-Generation Firewalls

As the application and threat landscape gradually increased along with the modern enterprise network and business needs, the necessity of, beyond port-based, firewalls development arisen. From the current network security landscape, the leaders for the commonly known as *Next-Generated Firewalls (NGFW)*, are considered vendors like Palo Alto,

Juniper Networks, Fortinet, Checkpoint and Cisco according Gartner's yearly report[5]. All of them, exploited several new techniques and tools to develop features and capabilities in order to address the aforementioned security challenges:

Application identification & control

A NGFW should be able to classify incoming traffic, identify the corresponding application and apply the defined security policies that are set. To achieve that, proper Layer 7 packet inspection is applied and is being compared with pre-installed databases of multiple application signatures residing on firewall's appliance memory. Application decoders and heuristic processes are also deployed. Since newly developed applications are constantly come up update of firewall's application signatures is conducted when the ability of custom signature creation is also provided.

Unified Threat Management

Modern NGFWs deliver multiple security features addressing different types of threats such as malware, spam, phishing, intrusion attempts and DoS attacks. Use of Threat Intelligence Cloud solutions are often recruited since all of the above threats constitute a constantly changing ecosystem. For example, Juniper's NGFW solution SRX[6] provide the following UTM features:

- **Anti-spam**

Identification and handling of possible harmful emails through the use of, web or local, based Spam Block Lists (SBLs)

- **Anti-virus**

Use of deep packet inspection scanning engine and virus signature databases to protect against Trojans, worms and other malware through different protocols (HTTP, FTP, SMTP, etc.)

- **Content Filtering**

Block of certain MIME types of files (.jar, .exe, .xls, etc.) to provide Dara Loss Prevention (DLP) capabilities.

- **Web Filtering**

URL filtering based on URLs category (e.g gambling, proxy, spam, etc.) as defined by third-party web sources, reputation-based again on the rating of third-party service or custom definitions

[5] https://www.fortinet.com/solutions/gartner-network-firewalls
[6] https://www.juniper.net/uk/en/products-services/what-is/utm/

each enterprise may want to apply filtering to.

Intrusion Prevention System

Since the firewall consists the primary layer of a network's security, it should monitor the incoming traffic and identify/mitigate any intrusion attempt or malicious activity. A NGFW performs real-time packet inspection and through the use of an IPS signature database is capable of matching traffic patterns corresponding to malicious incidents such as DoS attacks, virus-infected transmitted files, etc. and apply the preconfigured actions such as packet-drop and file removal. Also, via the constant network monitoring, a network traffic baseline is created and if any abnormal behavior is detected the firewall will proceed the current traffic's block. The above capability is used for addressing zero-day attacks meaning newly developed attacks for which may not be created signatures.

User Identity Awareness

To deliver enhanced and more flexible application control NGFWs support all major authentication protocols (LDAP, RADIUS, SAML, etc.) and integration with authentication servers (Active Directory). As a result, security administrators are able of allowing user-specific application and service control depending on each user's needs and role.

Advanced Logging & Reporting:

In nowadays' network security, the logging of network's traffic and security sessions is of equal importance of its protection for security and regulatory purposes (GDPR). In response to the above challenge, NGFWs deliver enhanced logging capabilities of the traffic traversing them and support detailed report generation for further analysis by the system administrators.

2.4 Application Security

Web Applications are the gateway to data handling and delivery across all the entities authorized to access it. They may consist from a single server to a complete server stack including application, database and web servers (multi-tier application architecture). Applications often process data from various sources and provide multiple services (financial, healthcare, commercial, etc.).

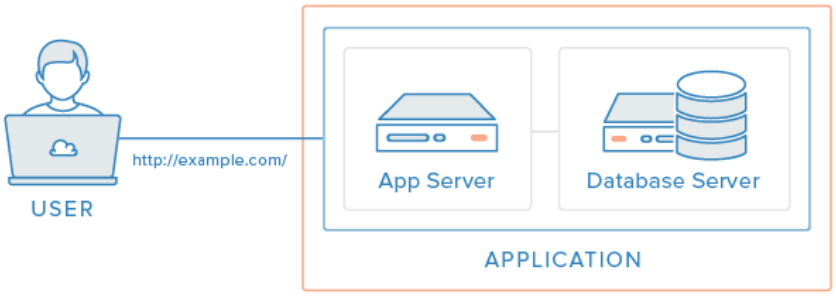

Figure 7: Typical Web Application Server Stack

Today's need for access to data from different locations in a fast and easy manner or the application's itself operating nature (e.g., Facebook, Twitter, E-Banking applications, etc.) forced their exposure to the Internet. The more valuable the data they handle and serve is, the higher the possibility to be targeted by adversaries attempting its extraction. The successful unauthorized acquisition of an application's data is commonly known as a data breach and dependent on the data's nature (e.g., financial, sensitive, etc.) may have significant consequences on an organization's financial or reputational status. According to Verizon's Data Breach Investigation Report (DBIR[7]), the 43% of data breaches conducted on 2020 was related to web application attacks. As a result, the security, a web application provides to its users and data, is elevated to major importance.

In favor of enhancing web application security awareness, the Open Web Application Security Project (OWASP), a non-profit foundation, published a report[8] of the 10 more critical web application security risks to encourage the industry of applying the appropriate countermeasures to mitigate these risks. The security threats presented in the report are:

i. Injection
   The injection of SQL queries, OS-level commands, etc. leads to execution of non-intended commands that may disrupt the application service or result in data exposure
ii. Broken Authentication
   The exploitation of poorly developed authentication mechanisms and the compromise of user credentials or user session hijacking

[7] https://enterprise.verizon.com/en-gb/resources/reports/dbir/
[8] https://owasp.org/www-project-top-ten/

iii. Sensitive Data Exposure
A Web Application's low-level security development such as the existence of weak encryption techniques threatens the protection of sensitive data like financial, healthcare or PII

iv. XML External Entities
The exploitation of vulnerable XML processors on XML-based applications may allow the adversary to conduct remote code execution, DoS attacks and data theft

v. Broken Access Control
The absence of adequate authorization mechanisms on an application's authenticated users may result in adversaries acquiring unauthorized administrative rights or access to other user's accounts data

vi. Security Misconfiguration
Unpatched software, the unintended public share of files or directories, the use of default or weak access credentials are a sample of misconfiguration incidents significantly reducing the application's security level

vii. Cross-Site Scripting (XSS)
The successful injection of malicious, client-based language, scripts (e.g., JavaScript) to web applications due to no input validation that allow attackers to hijack user sessions or force victims to execute malicious activities

viii. Insecure Deserialization
The non-proper validated deserialization of user input by a web application may lead to the injection of malicious data on application code

ix. Known Vulnerable Components
The adoption of vulnerable libraries or frameworks during the application development whose exploit may result in the violation of data confidentiality or web service's availability

x. Insufficient Logging/Monitoring
The lack of adequate user, system or network traffic logging/monitoring enhances the risk of non-detected application flaws or undergoing data breaches

2.5 Web Application Firewalls

2.5.1 Overview

In addition to the protection a firewall applies to the network, the level of protection a modern organization seeks, due to the wide use of web applications and services published to the internet, often requires the adoption of more advanced protection mechanisms. Web

Application Firewalls (WAFs) address these challenges by providing enhanced monitoring and protection of the Application Layer (Layer 7 of OSI Model) of these applications from, out of the organization, threats when logging & reporting is also applied for compliance and analytics purposes.

Figure 8: WAF – Reverse Proxy Deployment

As the diagram of Figure 8 displays, WAFs are residing in front of the organization's Web Servers, inspecting the incoming untrusted traffic and, by carrying the role of a reverse proxy, delivering it to the appropriate destination node. They can be delivered in different formats, as they can be deployed in a dedicated hardware appliance, in a virtual appliance or as a service offered by a third-party vendor. According to Gartner's Magic Quadrant for Web Application Firewalls 2020[9], Akamai and Imperva are considered leader vendors on the current sector where Cloudflare, F5, Barracuda and Fortinet are following as Challengers.

Figure 9: Top WAF Vendors

[9] https://www.gartner.com/doc/reprints?id=1-24F0FLTE&ct=201021&st=sb

2.5.2 Security Features

Web Application Firewalls are delivering a wide variety of features to form an adequate enterprise security solution.

Input validation

User input consists the greater mean of an application's vulnerabilities' exploitation as it permits a level of interaction between the user and the application. This is the reason the Injection attacks, as covered in 1.4, are ranked as the number one web application security risk according to OWASP. WAF applies a certain procedure upon receiving incoming data traffic.

The procedure consists of three steps:

i. Decryption (if needed)
ii. Normalization using the appropriate functions (hexDecode, lowercase, urlDecode)
iii. Validation against preconfigured policies and signatures

Cookie Protection

Cookies are small pieces of data assigned from web servers to users mostly used for session management, personalization and user tracking. The data they contain may correspond from a user's site preferences to login credentials and other user sensitive information. As a consequence, their compromise may lead to critical information disclosure. WAFs provide cookie security by either encrypting, signing or completely replacing cookies, through the use of Cookie Stores, transmitted by the protected web server to the end user.

URL Protection

Another common application security threat are the URL-related attacks where malicious scripts and malformed parameters are inserted along with the URL request. Buffer overflow, CSRF, directory traversal and remote file inclusion are a sample of the attacks that may be exploited by non-protected URL requests. WAFs make use of techniques such as URL encryption, character length limitation and parameters number restriction to address these types of attacks.

DoS Prevention

Denial-of-Service (DoS) attacks are widely deployed by adversaries to harm a web application's availability by exhausting its available resources (Bandwidth, CPU, memory or disk storage). When a DoS attack is conducted by multiple sources, often by a network of compromised hosts named botnet, then it is elevated to a Distributed DoS (DDoS). There are different types of DoS attacks depending on the layer of the OSI model they are targeting. The most frequently conducted attacks are those targeting the Application layer (Layer 7 DoS) through repetitive HTTP requests leading to resource starvation, Transport layer (Layer 4 DoS)

through SYN or UDP flooding and Network layer (Layer 3 DoS) through ICMP flood targeting the consumption of the bandwidth available for the for the serving of the legitimate traffic.

In response to these attacks, WAFs proceed in setting thresholds corresponding to the maximum ICMP, UDP packets allowed per minute when at the same time they set specific limits on accepted SYN packets permitted per second or per source IP address. If the number of SYN packets exceeds the preconfigured thresholds the source IP address is either entirely or periodically blocked from the WAF.

SSL

Modern WAFs are also equipped with SSL capabilities allowing them to conduct "SSL termination" at the incoming, encrypted, traffic and be able to validate it, in clear text, through their policy engine and signatures database. In addition, they are able to enforce the encryption of the connection between the users and the applications by redirecting the, initially, HTTP traffic to HTTPS.

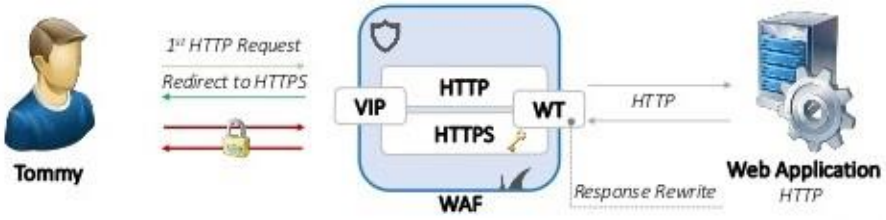

Figure 10: WAF – HTTPS Enforcement

Cloaking

At the primary stages of an attack the adversary attempts to collect information regarding the application structure such as the web or database server (OS, version, etc.) in order to search for associated vulnerabilities. To achieve it, he probes the application with input that will force it to generate error messages from which the desired information will be gathered. With the use of cloaking, WAFs block these error messages (e.g., 4xx/5xx error codes) from reaching the adversary. Also, cloaking removes certain HTTP headers from the response of the application servers that would include valuable information for the attacker (e.g., "set-cookie" header sends the session cookie from the server to user-agent).

Data Loss Prevention

Data Loss Prevention (DLP) is, in general, a set of security countermeasures to mitigate the disclosure of sensitive data such as Personally Identifiable Information (PII), Credit card numbers, social security numbers, etc. DLP is implemented on different sectors of an organization's environment due to data lifecycle. WAFs provide an application-based DLP

solution by inspecting the outbound traffic, recognizing via certain patterns whether there is sensitive data transmission and they either block or mask the corresponding data.

Virtual Patching

Virtual Patching is referred as a WAF feature that temporarily protects the underlying application from a known vulnerability until an actual patch is released from the development team and installed by the application administrator. Although is generally recommended software patches to be installed as soon as the vulnerability is detected, virtual patching consists a valuable countermeasure since it provides the required time to software engineers to develop and test the patch and schedule a maintenance window to install it guarantying the least possible downtime of the service. At the same time the security of the application is not downgraded significantly despite the vulnerability existence.

Logging & Reporting

A complete WAF solution should be able to provide advanced logging capabilities. Attack and traffic logs provide visibility upon application's proper protection by verifying the block of malicious requests and also assist on detecting false-positive events so the administrators to be able to proceed with the appropriate configuration fine tuning. In addition, logs are of high importance for auditing and regulatory purposes in case a security incident or data leak occur. Finally, logging enhances the monitoring of the firewall appliance itself by generating alert events when hardware-level anomalies or errors occur (High CPU/memory usage, low free disk space, etc.) giving the opportunity to administrators take the appropriate measures to ensure firewall's stable operation.

Reporting is the graphic representation of the data extracted by captured logs presented in a user-friendly manner. Reports are offered for advanced analysis of security threats (e.g., generated data rate of a possible DDOS attack), application insights (HTTP requests rate per hour, bandwidth usage per hour, etc.) and system auditing.

2.5.3 WAF Security Models

The logic a Web Application Firewall responds to inbound traffic differs, forming two separate security models the administrators are called to choose upon its configuration. The positive security model follows a whitelisting logic where every request to the web application is blocked except the ones defined as "permitted". This model presupposes that the administrator has complete knowledge of the protected web application's logic, operations and expected input which results in significantly increasing the configuration and administrative overhead. To ease administrator's effort, modern WAF's, upon their initial set up, run on a "training" mode for a certain period of time where they monitor the incoming requests, without applying preventive actions, to form a baseline regarding the legitimate traffic passed to the application.

In contradiction, a negative security model can be followed where all incoming packets are inspected and if evaluated as malicious, they are blocked from reaching the web application servers. To accomplish legitimate input verification, WAF's mostly use signature-based detection mechanisms where a blacklist of known-attack signatures is evaluated against the input and if a match occurs the corresponding packets are being dropped. While this method addresses efficiently most known web-based attacks as the one referred in OWASP Top-10, it provides minimum protection against zero-day attacks since no relevant signatures would have been implemented. Also, WAF bypassing could be easily conducted through use of several methods such as character obfuscation or multiple encoding. In response to the above, another deployment method of negative security model can be recruited called anomaly-based detection. According to this method, a series of anomaly detection processes can be followed to assist WAF in determining whether an attack is undergoing. Most anomaly detection processes are based on Machine Learning Models (further analysed in following chapter) used to learn and define application-expected traffic patterns so, eventually, to be able to classify incoming requests as legitimate or not. Although this approach appears to protect against a wider attack landscape compared to signature-based approach, it requires more system resources to work and also leads to higher times of response from the application servers to the clients since all requests need to be evaluated and classified. In continuance, use of non-adequate learning models may result in high number of false-positive events where legitimate traffic is evaluated as malicious and being dropped affecting application's proper functionality. Taking into consideration all the above, a hybrid approach is recommended to be followed where both anomaly and signature-based methods are used to gain the maximum protection with minimal resource utilization and false-positive occurrence rate.

2.6 Machine Learning

2.6.1 Overview

The sharp and constant increase of the network and application security threat landscape over the past years forced the IT community to develop and add to their cyber defense toolkit new technologies with Artificial Intelligence (AI) and especially Machine Learning (ML) to be the one gathering the most interest and growth. It is no surprise that AI in Cybersecurity market is forecasted to perform an increase of almost 30 billion USD in the next 5 years (38.2 billion USD in 2026 & 8.8 billion USD in 2019[10]). This chapter is going to present an introduction to the basic concepts of Machine Learning technology along with the capabilities it can provide at the network and application security sector.

Learning is a process that receives information from the external environment and by proper processing elevates it into knowledge which, in turn, will be used to perform multi-complexity tasks. Machine learning is a constantly learning system that makes decisions based on the knowledge it gains and not based on pre-configured commands or rule-sets. Also, is a continuous improving procedure since it is able of including its decision output, whether is wrong or right, on its learning process to enhance the accuracy of its next executions.

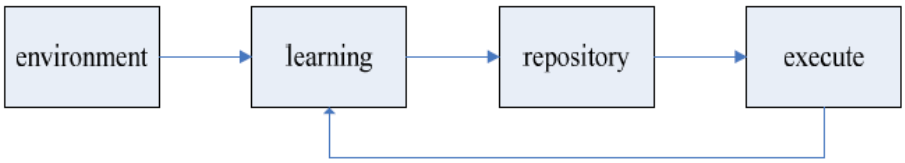

Figure 11: Machine Learning System Model

The unique capability a Machine Learning system provides is the simple identification of patterns through building and training models via processing huge multi-variant datasets from different external sources. The term "Model" is defined as the specific set of principles on which the data characteristics are based. This is achieved in an automated way since no further code development is required after releasing the machine learning algorithm. The only additional requirement is the provision of the appropriate datasets and examples to build and train each model. The scalability and the accuracy of the algorithm is directly related to the range and variance of examples provided. The nature and operating model of ML systems make them applicable, except of cybersecurity that the current project focuses, to several other sectors such as health care, marketing and finance.

However, as any newly developed technology, there are several challenges and concerns that ML systems need to address. Their effectiveness is related, except to the amount, to the quality of data it processes, so it should be as unbiased and integral as possible to avoid poorly designed models which, in turn, will result in a high error rate. In addition to quality, this huge volume of data requires a significant amount of time, compute and power resources (RAM,

[10] https://www.marketsandmarkets.com/Market-Reports/artificial-intelligence-security-market-220634996.html

CPU and GPU) for a proper model to be processed and trained. Finally, the selection of the right ML algorithm, according to the desired task, is of utmost importance along with the adequate revision of its results to proceed to possible errors detection and fine tuning.

2.6.2 Definitions

As with any technology and science sector, Machine Learning makes use of a certain glossary in order to define and describe its concepts, algorithms and tasks along with their proper evaluation. Before further analysis to ML's basic concepts and algorithms is made, a referral to its basic definitions is mandatory to fully comprehend the relevant content.

- Algorithm:
  A set of rules followed to achieve a certain objective. An algorithm describes all the necessary steps in order to get from the inputs provided to the desired output.
- Label:
  The "answer" for a given example. In a classification task, it will be the tag applied to the provided data. In terms of network security, a possible set of labels would be "malicious" or "legit" to describe a certain network traffic which would be filtered by a NGFW.
- Features:
  All the properties that characterize an input in the form of variables
- Example:
  A specific input containing all the relevant features. If the corresponding label is also included then it is termed as a "labeled" example.
- Dataset:
  A group of provided examples. If a dataset is used to train/evaluate/test a model is mentioned as training/validation/test dataset correspondingly.
- Model:
  The generated learned program that is able to provide predictions for any given input. Depending on the task it executes it can be named either as "classification" or "regression" model.
- Prediction:
  A model's output corresponding to a specific input
- Regression:
  A supervised learning technique where the output is a continuous value and is evaluated based on the accuracy of the predicted values when compared to the actual values.
- Classification:
  A supervised learning technique where the output is a discrete value such as "true" or "false", "malicious" or "legit", "normal" or "abnormal".
- Clustering:
  An unsupervised learning technique where grouping of data points is conducted in order a defined group or "cluster" to present common characteristics that differentiate it from other clusters.

- Overfitting:
  A state where a model attempts to capture and adapt all features of the training data resulting to not be able to generalize to new observation.
- Underfitting:
  The opposite state of Overfitting where the model fails to efficiently capture all the necessary details of the provided training set and as a result no accurate predictions can be conducted.

2.6.3 Machine Learning Concepts

Depending on the task to be accomplished there are different approaches that can be followed. The main Machine Learning approaches are the following:

Unsupervised learning

An unsupervised learning algorithm accepts as input non-classified data and its primary goal is to discover given data's structure and the principles hiding behind it. Since the procedure of labeling data is not needed the available data sets are significantly increased and cost-effective. Popular problem the current approach can solve is clustering, meaning the grouping of dataset based on objects with similar properties. Clustering is mainly used for optimization in manual labeling of new samples. Another problem being addressed is the one of association. Association or recommendation is defined as the procedure of finding rules for association between given and new data to provide recommendations.

Supervised learning

Contrary to the previous approach, this algorithm receives labeled data and, given that information, builds and trains a model able to classify further new data. In order to work properly, it consists of two phases:

i. Training phase:
   Development and training of a model based on the available classified dataset. During training, the model structure and parameters are constantly changing according to the right answers of the labeling task.
ii. Evaluation phase:
   Use of the trained model to classify again labeled data but without given the label information. In this phase, no changes are made on the model's properties so as to be evaluated based on the error-rate it will produce. If the error-rate is considerably high then the model rolls back to the previous phase for further training.

Supervised algorithms are mostly used for classification tasks, called to split data into certain categories, and regression tasks where, given certain data, the output has to be a numerical value within a given valid range.

2.6.4 Machine Learning Algorithms

After examining the main ML concepts and the problems they address, an analysis to the most widely used algorithms each approach can employ is required.

2.6.4.1 Supervised Learning Algorithms

- **Linear regression**

Linear regression constitutes one of the simplest Machine Learning algorithms and is the foundation prior to a deeper dive in more complex algorithms as the ones described above. It is a statistical method to create models outlining the dependency between a dependent variable and several independent variables. Its main appliance is predictive analysis regarding continuous, real and numeric values which can be met in the sectors of finance and real estate. The mathematic formula defining linear regression is $y= a0+a1x+ \varepsilon$, where y is the dependent variable, x the independent variable, $a0$ the line interception, $a1$ input's scale factor and ε a possible random error.

The number of independent variables defines the type of the linear regression. If one independent variable exists then the algorithm is termed as a "Simple Linear Regression algorithm". If more than one independent variable exists then a "Multiple Linear Regression" algorithm is defined.

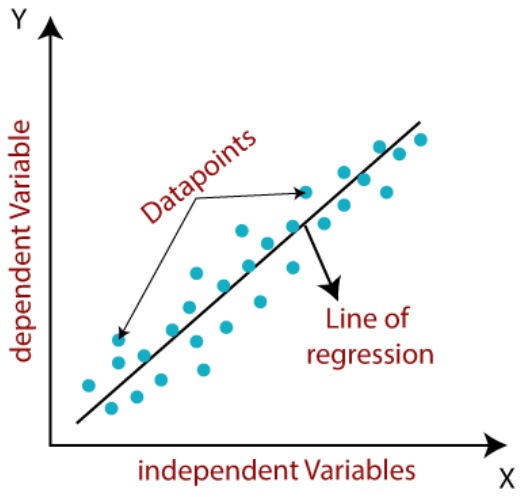

Figure 12: Linear Regression

Advantages:

i. Simple implementation and interpretation of output
ii. The best algorithm to use for linear relationship (dependent and independent) variables
iii. Ability to avoid Overfitting with regularization techniques

Disadvantages:

i. Susceptible to underfitting
ii. Inability to proper address real-world tasks since the linear dependency between dependent and independent variables is rare
iii. The existence of extreme values on a given data set may reduce significantly the algorithm's performance and the results' accuracy

- **Logistic regression**

Like linear, logistic regression is a widely popular supervised learning algorithm which, however, provides probabilistic predictions, in the form of discrete or categorical values as output, for dependent variables. A major difference, when compared to linear, is that the predicted values are selected from a continuous value space and by setting certain thresholds a classification task can be conducted. This difference is represented on a x,y-axis diagram by the introduction of a curved line indicating the likelihood of a datapoint to be classified among a set of specific labels.

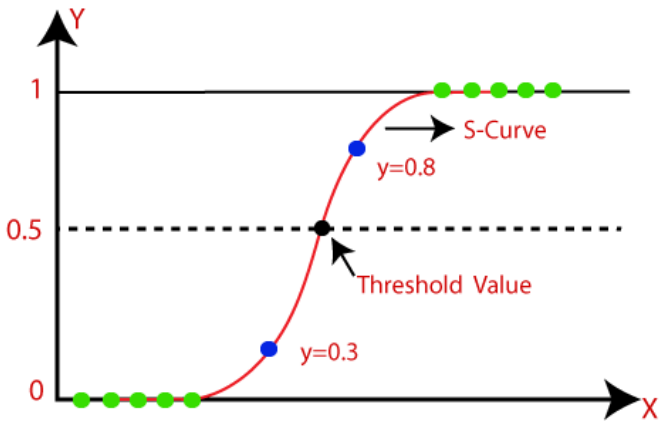

Figure 13: Logistic regression diagram

Given the above diagram, a threshold value of *0.5* has been set, meaning that if a prediction for a certain datapoint is for example *y=0.8* it will be assigned the value 1. This hypothesis of logistic regression is represented by the following mathematic formula:

$$h\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

Based on the number of the possible labels for the dependent values three different types of logistic regression algorithms are being defined:

- Binomial: Only two different possible values (e.g., True/False, 0/1, Pass/Fail)

- Multinomial: Three or more unordered possible values (e.g., “Red”, “Green”, “Orange”)
- Ordinal: Three or more ordered values (e.g., “High”, “Medium “, “Low”)

Advantages:

i. Ease implementation and output interpretation
ii. Quite extensible due to the ability of classification between several labels (multinomial regression)
iii. High accuracy given relatively simple datasets
iv. Less prone to overfitting compared to linear regression

Disadvantages:

i. Assumption of linear relationship between dependent and independent variables
ii. The dependent variable is bound to a discrete set of values
iii. Overfitting may occur if number of examples is lesser compared to the number of features

- **Decision Trees**

As the name indicates, Decision Trees algorithm offers a tree-like model used for the representation of decision analysis and is capable for both classification and regression tasks. The algorithm receives a multi-feature dataset and splits it in a multi-level tree based on certain conditions, features and attributes.

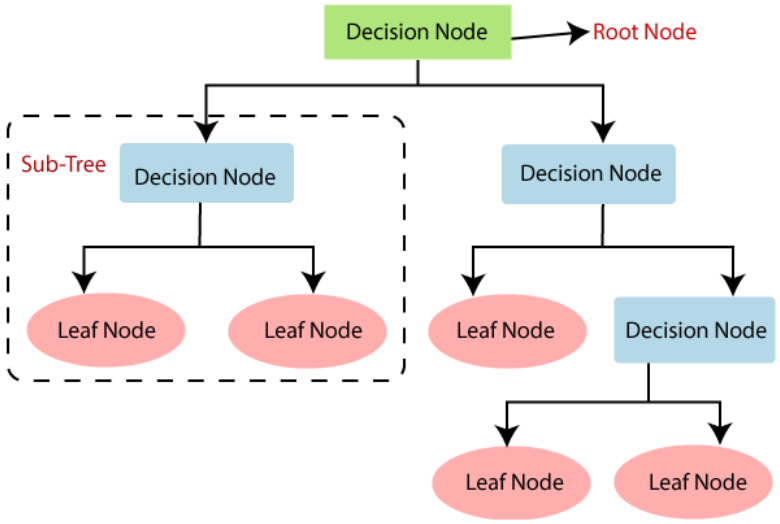

Figure 14: Decision Tree Model

Each condition represented on the tree is also referred as an internal or decision node when the edge of each node which is the corresponding decision is referred as a leaf. The length of the tree is directly dependent on the features of the imported dataset and the decision nodes derived from it.

The selection of the decision nodes from a given dataset is the main challenge for a Decision Tree algorithm. The method used for this selection is called Attribute Selection Measure (ASM). The most widely used formulas, *Information Gain*, *Gini Index* and *Gain ratio*.

When long-length trees are generated the risk of overfitting occurs and, in the opposite, short-length trees produce highly inaccurate decisions since not all dataset features may not been evaluated and included. To address these issues the process of pruning is used by removing internal nodes of low importance to reduce tree complexity without, however, decreasing accuracy of the algorithm. Two popular methods of pruning are used:

- Reduced error pruning: Starting from the leaf level and deleting the class with highest popularity
- Cost complexity pruning: An *Alpha* metric is defined and used to weigh and delete nodes based on their sub-tree size.

Advantages:

i. Ease visualization and interpretation of high-complex datasets
ii. Ease and fast execution of decision-oriented tasks (e.g classification)
iii. Adequate data analysis with low effort
iv. Tree performance is not affected by nonlinear related variables

Disadvantages:

i. High risk of high-complex trees (overfitting) if pruning is not applied efficiently
ii. Small variations in provided data may lead in generating different trees
iii. No proper training of the model, by providing limited samples of datasets or limited training time, may lead to no optimal tree generation and non-accurate decision making

- **Random Forest**

Exploiting the capabilities of Decision Tree algorithm, Random Forest model makes use of several different and uncorrelated trees where each of them carries out a class prediction. After summarizing the predictions of all trees, the one with the most votes become the model's prediction. By using non-correlated trees, the model prevents the adoption of non-accurate predictions performed by individual sub-models. The less correlation exists between the different tree instances the more unbiased and accurate the final prediction for a future examined datapoint will be.

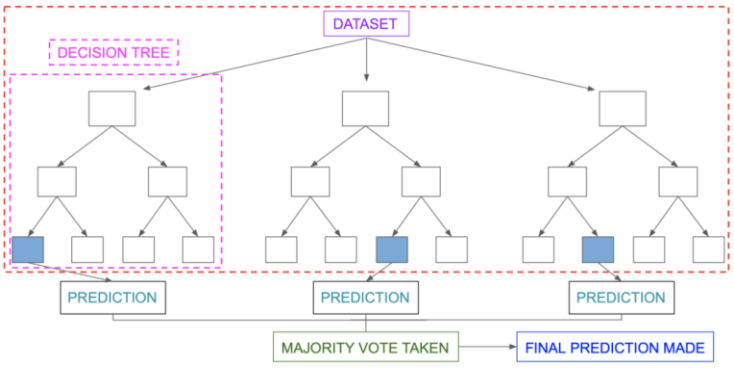

Figure 15: Random Forest Model

To achieve a low correlation level between decision tree instances, Random forests makes use of a technique called *Bagging* (Bootstrap Aggregation) where each instance randomly selects a sample-subset from the given dataset on where his training will take place. As already mentioned, small variations on the sample data lead on generation of different trees so the random selection of samples guarantees a low percentage of equivalence between the decision instances.

The advantages and disadvantages of Random Forest model are presented below:

Advantages:

i. Reduced risk for overfitting instances
ii. Improved accuracy compared with individual decision trees predictions
iii. Efficient process of both categorical and continuous values
iv. Adequate executing both classification and regression tasks

Disadvantages:

i. Higher resource consumption due to use multiple instances
ii. Higher training time need
iii. May lack of data interpretability due to its inconsistent process by many different instance models

- **Naive Bayes**

Naive Bayes classifiers is not an individual algorithm model but a set of different classification algorithms all based on Bayes' Theorem which calculates the probability of an event A given the probability of another, already occurred, event B and is described by the below formula:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

The event B is also referred as evidence when P(A) is termed as the priori probability of event A (probability of the event without given the evidence). The P(A|B) value is the posteriori probability after including the evidence in the calculation process. An evidence most of the times consists of more than on variables which, for simplicity purposes, are considered independent between each other. So, for an evidence X = (x_1, x_2, x_3, …, x_n) the Naive Bayes formula is expressed as:

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

P(y) is called class probability, P(x_i|y) is referred as conditional probability and P(y|x_1, …,x_n) consist the posterior probability of the target class. On a given dataset all the evidence variables x_n constitute the feature matrix of the model when the class variable y consists the response vector. The model, based on the calculation of the given evidence, will make an assumption that an unknown event belongs to the class with the higher posterior probability.

The advantages and disadvantages of Naive Bayes model are presented below[11]:

Advantages:

i. High performance algorithm
ii. Improved classification process due to removal of non-relevant attributes
iii. Short training time

Disadvantages:

i. Significant big datasets needed for accurate results
ii. Decreased accuracy compared with other classifier models for specific datasets

- **Support Vector Machines (SVM)**

Main objective of SVM algorithms is the definition of a hyperplane inside a space of N-dimensions, where N is the number of the different dataset attributes, which will be able to classify any given data point. The possible hyperplanes that can be produced are several and are presented the below figure:

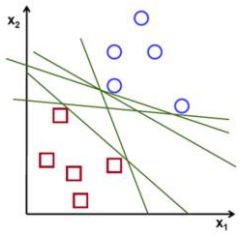

Figure 16: Possible Generated Hyperplanes

[11] Sayali D. Jadhav, H. P. Channe, "Comparative Study of K-NN, Naive Bayes and Decision Tree Classification Techniques", January 2016

In order to succeed the most accurate and less error-prone classification a hyperplane with maximum margin between the data points of the two different classes should be selected. This Hyperplane will be termed as the optimal Hyperplane for the current classification task.

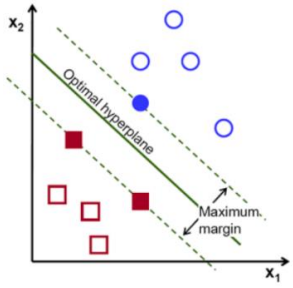

Figure 17: Optimal Hyperplane

The data points closer to the hyperplane are named support vendors and they basically define hyperplane's orientation. By manipulating support vendors, the hyperplane position is affected.

Advantages:

i. High accuracy algorithm when classes are coherently separated
ii. Highly effective in multi-dimensional spaces
iii. Not a significantly memory consuming model

Disadvantages:

i. Not efficient against large sets of data
ii. Not effective against overlapping classes
iii. No ease interpretation of classification results due to no probabilistic explanation

- **Artificial Neural Networks (ANN)**

Artificial Neural Networks constitute another set of algorithms whose main objective is pattern recognition through receiving and properly processing raw data. Neural networks assist on both classification and clustering tasks. The interpretation of data is conducted through multiple layers of nodes called "neurons". An ANN may consist of at least three layers of neurons, the input layer, where the data is received, one or more hidden layers, where the main data processing takes place, and the output layer which constitutes the final predictions of the model.

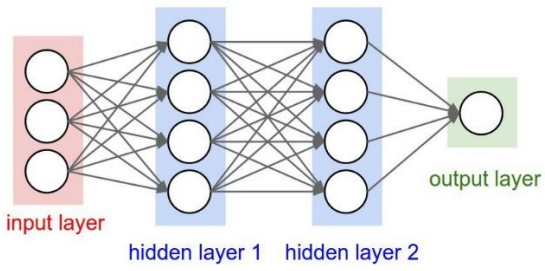

Figure 18: ANN Layer Structure

Data, while forwarded between layers through connections called channels, is assigned significance by added specific weight. Each neuron node has a certain bias value which is added to the input weighted-data. This sum is, then, passed through an “Activation Function” resulting in the predicted probability that will determine to what extent the specific signal will progress through the layers. All nodes of the hidden layers are designed to execute the same activation function. The nodes of the output layer will execute a different activation function depending on the type of the desired prediction. The continuous execution of the activation function on the weighted-data on each layer will lead the initial raw input to the final output of the model. This procedure is called “Forward Propagation”.

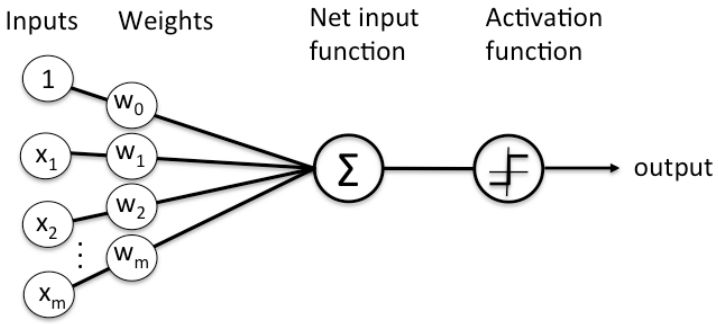

Figure 19:Input-data procession path

The activation functions that may be used by the hidden layers vary, however, the currently recommended for modern NNs, because of its simplicity, is the Rectified Linear Activation, also known as ReLU. The formula describing its function is max(0.0, x) meaning that if the input value “x” is negative then the value 0.0 is returned as output.

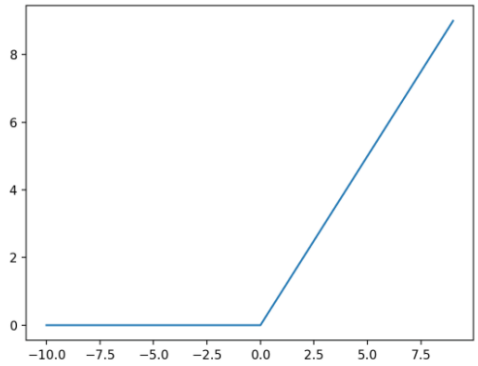

Figure 20: Input/Output of ReLU Activation function

Regarding the activation functions the output layer may use, depending on the prediction type, three are considered the major ones:

i. Linear:
   The linear function practically commits no change on the weighted-output of the hidden layer returning it as is. Linear activation function is solely used for regression problems.

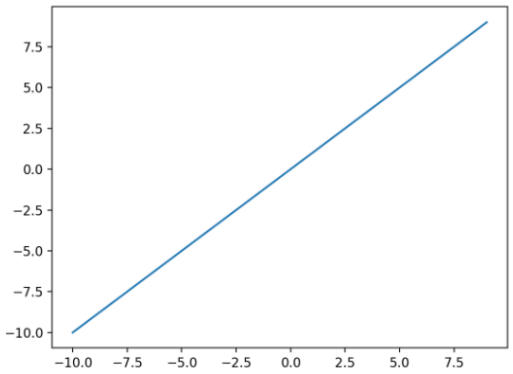

Figure 21: Input/Output of Linear Activation function

ii. Logistic:
The logistic activation function, also known as Sigmoid, is the same used in logistic regression classification tasks where any real value is inserted and the output fluctuates within the range 0 to 1. The prediction is calculated through the formula **1.0 / (1.0 + e^-x)** where x is the corresponding input. Sigmoid is mostly preferred as activation function for binary and multilabel classification tasks.

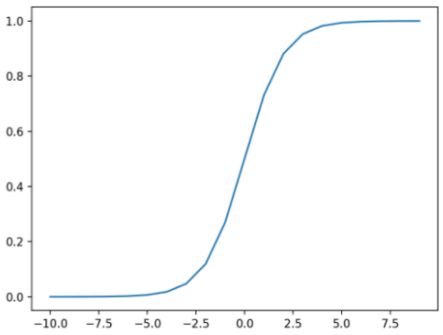

Figure 22: Input/Output of Sigmoid Activation function

iii. Softmax:
The Softmax Output activation function receives as input a vector of real values resulting to a vector of the same length but with a total sum of 1 which will correspond to the "predicted" label. The formula describing this function is **e^x / sum(e^x)** where x the input. Due to its architecture, Softmax cannot serve multilabel classification although it is widely recommended for classifications where multiple classes take place.

In order to train an ANN, a procedure called "Back Propagation" is mainly followed consisted of the below steps:

i. Define the initial weights and biases of each neuron for all layers respectively
ii. Insert labeled data to the ANN
iii. Depending on the predicted output calculate the error based on the expected output

iv. Propagate the error backwards to all neurons of each layer, adjusting the corresponding weights and biases until the "Forward Propagation" of the ANN produce the expected value

Advantages:

i. ANNs are not limited in addressing one particular type of task since their output may be either a real or discrete value
ii. ANNs are quite robust against noisy data since possible errors contained are not affecting the final prediction
iii. Ability to perform highly complex classification or clustering tasks through the use of multiple hidden layers
iv. Multiple tasks can be performed in parallel

Disadvantages:

i. Highly dependent on underlaying hardware since processors with parallel processing power are required
ii. Difficult interpretation of the provided output since no feedback is provided upon the followed process
iii. Extensive training is required

2.6.4.2 Unsupervised Learning Algorithms

- **K-means Clustering**

Clustering, in general, is widely used in the data science sector for data analysis purposes as it gives deep insights regarding data's structure by forming several subgroups, also known as clusters, composed by data points with similar characteristics.

K-means algorithm makes use of clustering to group multiple data points across a K-number of clusters. Necessary prerequisite is these clusters to be non-overlapping and each data point to belong to only one of these clusters at a time. K-means is considered as an unsupervised learning algorithm since it does not pass a training phase where certain input with the corresponding "correct" output is provided but, on the contrary, it has to analyze the input itself and based on each own perception to divide it into separate homogeneous subgroups.

Its operational logic could be described as below:

1. Randomly selects number K corresponding to the number of clusters the data points will be separated
2. From the given dataset picks a random K data points which will constitute the core of each cluster (centroids)
3. In continuance, assigns the remaining data points to their nearest centroid with which they share similar features. This results on forming a first version of clusters
4. Then, a recalculation is occurring on each cluster and based on the variance of the underlying data points, a new centroid is elected

5. This leads to a reform of the current clusters since now data points from different clusters may be closer to a centroid of another cluster.
6. If any cluster reform is conducted a recalculation will be triggered until no more reassignments occur. This would indicate that the model is ready and fully converged.

Main goal of the algorithm is the achievement of an ideal clustering that would result in the minimization of the distance between the data points and the centroid at each one of the formed clusters so the data segmentation to be as more distinct as possible.

Figure 23: Ideal Clustering example

Advantages:

i. Scalable to large sets of data
ii. Relatively ease at implementation
iii. Ease adaption of new input
iv. Due to the continuous recalculation of the variance between clusters and the appropriate reassignment of the data points convergence is always achieved

Disadvantages:

i. The selection of the K-value is hard to predict
ii. Different initial segmentation of data points may result in different final clusters
iii. Low performance for clusters with different size and density

- **Apriori algorithm**

Apriori is an iterative algorithm designed to process datasets containing transactions and based on the frequent itemsets builds certain association rules. With the term “frequent itemsets” is defined the high percentage of presence, determined by a user selected threshold, of specific items in a series of transactions. This percentage of presence is also termed as support.

Similar to K-means, Apriori algorithm follows a set of operational steps:

1. Records the support of itemsets from the imported transactions and defines a minimum support and confidence threshold. The term “confidence” describes the percentage an item A & an item B to belong to the same transaction
2. Next, it collects all the recorder supports which are higher than the predefined threshold
3. In continuance, identifies the rules of the collected supports that, again, have a higher confidence level from the predefined minimum confidence level
4. Finally, all the generated rules are sorted on a descending order

Advantages:

i. Relatively easy algorithm to comprehend
ii. Ease implementation of all the algorithm’s steps regardless the size of the dataset

Disadvantages:

i. Relatively slow algorithm
ii. Due to its iterative operation its total performance could be significantly decreased
iii. Datasets with multiple transactions between many different items majorly increase the algorithm’s time and space complexity

- **Principal Component Analysis**

Another popular example of unsupervised learning method is the one of Principal Component Analysis (PCA). Its main objective is the dimensionally reduction of large data sets meaning the limitation of variables while preserving the provided information in high levels. PCA’s operation constitutes from a series of certain steps:

1. Firstly, a standardization of the initial variables is needed tuning them at a comparable scale to avoid biased results. This is achieved by subtracting the mean from each feature’s value and divide the outcome with the standard deviation:

$$z = \frac{value - mean}{standard\ deviation}$$

2. Then, a covariance symmetric matrix is calculated to depict the correlation level between all the data set’s features. If a data set is consisted of 3 different variables x,y and z then a 3x3 matrix will be formatted:

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$$

Figure 24: 3x3 Covariance Matrix

The value of Cov(y,x) describes the correlation between y and x variable. If the value is positive then the two variables are considered correlated whereas a negative value would mean that they are inversely correlated (if one feature is incrementing the other one will be decreasing).

3. After the Covariance Matrix is formatted then the principal components are determined. They are constructed as a mix of the correlated features (as calculated on the Covariance Matrix), keeping that way the embedded information, in such a way so as to be uncorrelated with each other. The main objective is to compress the maximum possible information along the first principal components. The final principal components will be of the same number as the initial variables but the percentage of contained information will be on a descending order.

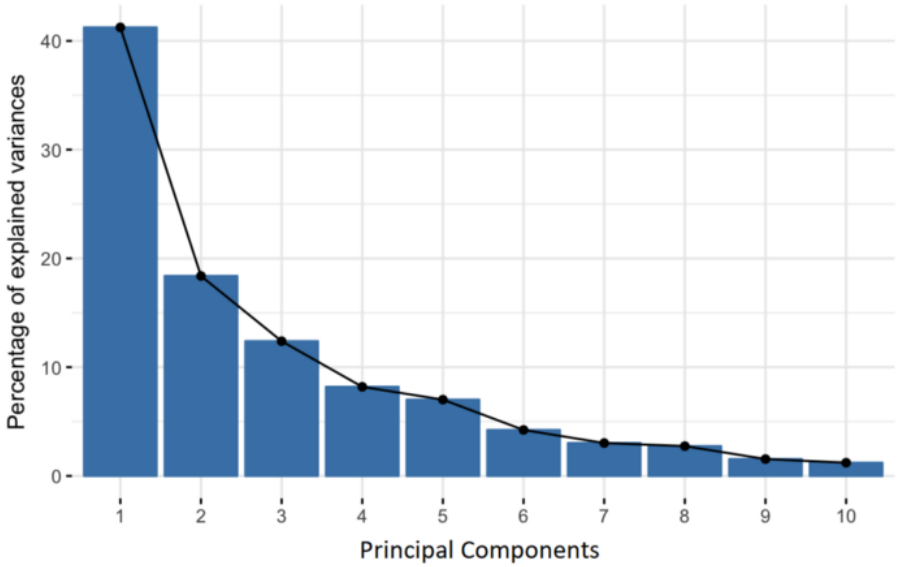

Figure 25: Percentage of contained information distributed along a data set's Principal Components

4. As a final step, a discard of the components with the less inherited information is applied to reduce data set's dimensionality making its interpretation and visualization more ease and less resource consuming.

Advantages:

i. Enhanced Algorithm Performance due to less feature processing
ii. Reduced Overfitting due to variable limitation
iii. Improved Visualization since data set's dimension is significantly decreased

Disadvantages:

i. Information Loss
ii. Decreased Accuracy
iii. Principal Components are less interpretable compared to the initial features

2.6.5 Performance Evaluation

Another significant part of the Machine learning technology except the available ML algorithms and their capabilities is their proper evaluation. Evaluation is critical in order to determine if an algorithm is capable of conducting a certain task efficiently along different criteria such as accuracy, time and resources. Depending on the task a Machine Learning model is designed to execute there are different metrics used in order to evaluate its efficiency.

Classification Metrics

- Confusion Matrix
  The task of classification can lead to four different possible outcomes:
  a. True positives: The labeling of the provided input corresponds to its true label
  b. True negative: The prediction that an input is not associated with a possible label is true
  c. False positive: The predicted label of the provided input does not correspond to its actual label
  d. False negative: The prediction that an input is not associated with a possible label is false

A Machine Learning Model is considered efficient if it keeps the True Positive & True Negative Rates as high as possible whereas the False Positive & False Negative Rates are kept on low levels.

The above four outcomes are easily represented on a confusion (NxN) matrix where N is the number classification's labels.

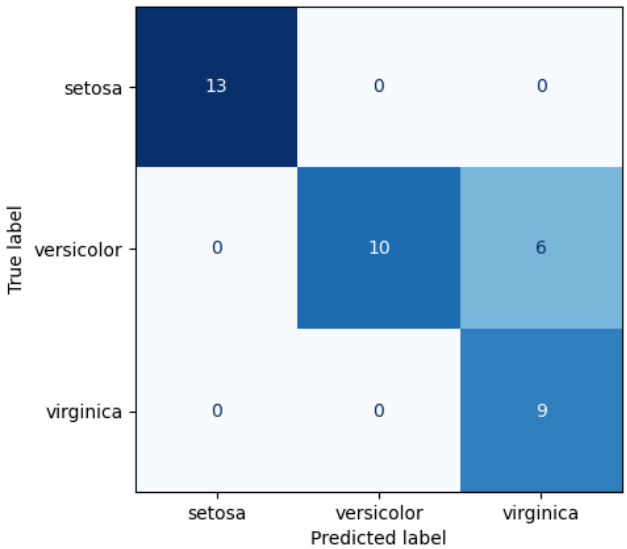

Figure 26: Confusion Matrix representing an Iris flower classification task

Based on the possible outcomes of the classification task and their representation on a confusion matrix, three classification evaluation metrics are defined:

- Accuracy
  The number of correct predictions compared to the total prediction attempts made

$$\text{accuracy} = \frac{\text{correct predictions}}{\text{all predictions}}$$

- Precision
  The percentage of true positives in comparison to the total true & false positives outcome

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

- Recall
  The fraction of the true positives' outcome among the total true positive & false negative outcomes

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

The individual use of one of the above metrics (e.g., accuracy) for the evaluation of a classifier is not considered as trustworthy since the possible classes among a given dataset may not be evenly distributed. To avoid this, an F-measure is defined where all three metrics are taken into consideration:

$$F_\beta = \left(1 + \beta^2\right) \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

The β parameter adjusts the weight between recall and precision so as the F-metric to adapt to each classification task's evaluation needs. To be more specific, if β<1 then more weight is given to precision metric whereas β>1 would add more value on recall.

- AUC – ROC Curve
  The most popular and descriptive way to visualize the evaluation of a binary classification algorithm is the AUC (Area Under the Curve) –
  ROC (Receiver Operating Characteristics) curve. ROC is a probability curve among the True Positive and False Positive Rates and AUC examines the level of separability between the True Positive and True Negative predictions of the classifier. A perfect model would have an AUC equal to 1 meaning that it totally separates the two classes correctly.

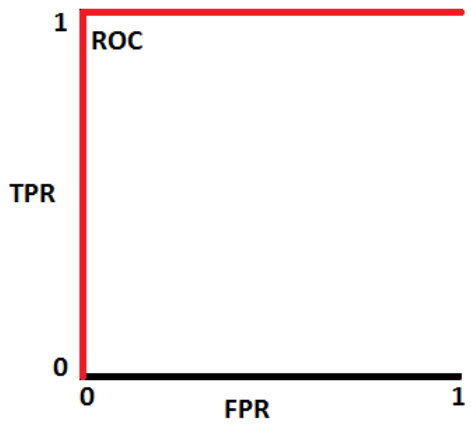

Figure 27: ROC curve of a perfect classifier

If a classifier presents an AUC equal to 0.7 that would mean that there is 70% possibility to distinguish correctly the two classes. This would lead to the below ROC curve:

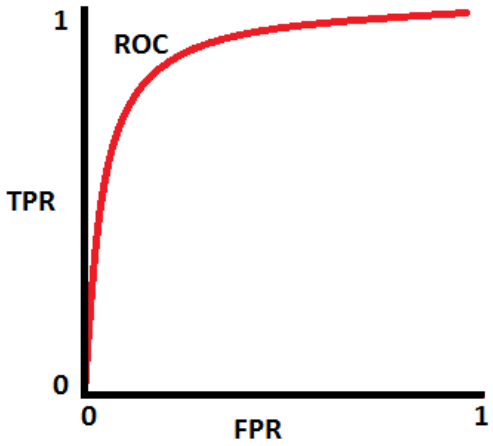

Figure 28: ROC curve of an AUC=0.7 classifier

Regression Metrics

Due to the different nature of regression tasks compared to the ones of classification where continuous values are predicted instead of discrete, different mechanisms are required for a model's performance evaluation.

- Mean Absolute Error (MAE)

  The sum of the absolute differences between the actual values and the corresponding predicted values

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

- Mean Squared Error (MSE)

  The average of squared differences between the actual values and the corresponding predicted values. MSE differentiates from the MAE since it does not focus on the distance spectrum between predicted and actual value, it just reports if the average of predictions were true or false.

$$\text{MSE}\left(y_{true}, y_{pred}\right) = \frac{1}{n_{samples}} \sum \left(y_{true} - y_{pred}\right)^2$$

- Root Mean Squared Error (RMSE)

  RMSE is defined by calculating the square root of MSE and it's preferred due to the fact that its output is more interpretable and easier to compare with relevant outputs of other evaluated models.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- Explained Variance (EV)

  Explained Variance metric states a model's outcome variance between the predicted and the expected outcomes.

$$EV\left(y_{true}, y_{pred}\right) = 1 - \frac{Var\left(y_{true} - y_{pred}\right)}{y_{true}}$$

- R^2 Score

  R^2 score constitutes another regression metric used to provide a measurement regarding how efficiently a model fits to a specific dataset. It is calculated with the division of the squared sum of prediction error along with the squared total sum.

$$R^2\left(y_{true}, y_{pred}\right) = 1 - \frac{\sum \left(y_{true} - y_{pred}\right)^2}{\sum \left(y_{true} - \bar{y}\right)^2}$$

  where:

$$\bar{y} = \frac{1}{n_{samples}} \sum y_{true}$$

2.7 Related Work

2.7.1 Creating firewall rules with machine learning techniques[12]

Main objective of the current thesis is the demonstration of more efficient and advanced firewall rules that can lead on significantly improved Intrusion Detection Systems, if enhanced with machine learning techniques. The machine learning algorithms used for the development of models, from where the firewall rules will be calculated, are the Random Forest and Neural Networks whereas SVM classifiers are used to categorize packets as legit or malicious.

The features to be extracted from each received packet are the following:

- The 256 possible bytes lead to 256 different features
- Total packet's length
- Total packet's header length
- Total amount of data stored (Caplen)
- Packet's identifier
- Flag (True or False)
- StdevTime: Time deviation between the last received packet and the previous seven ones received before it

Other feature-extracting methods used are:

- N-grams: The matching and grouping of features based on the defined n value
- Dimensionality of features: The number of resulting features after the n-gram grouping
- Clustering: Receives as input the frequency matric of n-grams, the number of dimensions, a value of tolerable information loss and outputs a "k" number of clusters
- Packet grouping: Since an attack is carried through a series of packets and not individually, several packets are grouped and then classified as malicious or normal

Below a performance evaluation of all the applied methods is displayed where N is the number of nodes per hidden layer, L is the number of hidden layers and D is the tree depth and "rr" stands for recognition rate.

[12] Roland Verbruggen, “Creating firewall rules with machine learning techniques”, Radboud University Nijmegen

nr	Method	data set	Vars excluded	parameter's	rr	TP	FP
0	Random tree	1	-	$D = 4$	94.099%	0.785%	0.035%
1	Random tree	2	-	$D = 9$	99.744%	0.997%	0.007%
2	Random tree	3	-	$D = 3$	93.210%	0.932%	0.223%
3	Random tree	3	-	$D = 4$	94.400%	0.944%	0.157%
4	Random tree	3	-	$D = 5$	95.910%	0.959%	0.156%
5	Neural network	66% of 3	-	$N = 133$, $L = 1$	85.600%	0.000%	0.055%
6	Neural Network	66% of 3	ident	$N = 133$,$L = 3$	98.650%	0.987%	0.077%
7	Random tree	4	-	$D = 3$	95.872%	0.987%	0.077%
8	Jripper	1	-	-	99.804%	0.998%	0.006%
9	Jripper	1	src and dest port	-	98.678%	0.987%	0.038%
10	Neural network	5	-	$N = 133$, $L = 3$	98.717%	0.947%	0.006%
11	Random tree	5	-	$D = 9$	98.975%	0.939%	0.002%
12	Jripper	5	-	-	99.835%	0.994%	0.001%
13	Random tree	6	-	$D = 9$	99.006%	0.956%	0.005%
14	Neural network	6	-	$N = 27$, $L = 4$	99.051%	0.95%	0.003%
15	Neural network	6	-	$N = 27$, $L = 7$	99.119%	0.957%	0.003%
16	Random forest	6	-	$D = 22$	99.825%	0.995%	0.001%
17	Neural network	6	-	$N = 27$, $L = 10$	99.149%	0.954%	0.003%
18	Jripper	6	-	-	99.704%	0.985%	0.001%

Figure 29: Performance Results

2.7.2 Network Firewall using Artificial Neural Networks[13]

Goal of the project is the analysis of the filtered, legit network traffic exiting a traditional rule-base firewall based on which a model will be built and trained. After properly developed and trained, the model should be able to classify the incoming network traffic as "allowed", "denied" or "accepted". To develop that model, an artificial neural network and more specifically a multi-layer perceptron (MLP) was used whereas its training was conducted based on the back-propagation algorithm combined with cross-validation to avoid overfitting.

A training example is consisted of certain attributes of a packet's header along with the corresponding firewall action:

- Source IP address (e.g., 192.168.1.1/24)
- Destination Port (e.g., 80)
- Protocol (e.g., TCP)
- Action (ALLOW, REJECT or DENY)

The MLP architecture used is consisted of the below components:

- Input: binary, IP + port + protocol (TCP, UDP), 32+16+1 = 49 bits (neurons)
- Hidden layers (to be found)
- Weights: [-0.1, 0.1]
- Output: three neurons (ALLOW, REJECT, DENY)

[13] Kristian Valentin, Michal Maly, "Network Firewall using Artificial Neural Networks ", Department of Applied Informatics Faculty of Mathematics, Physics and Informatics Comenius University, 2013

Below a graph representing the model's false positive classifications of the validation set for a certain time period during learning process is displayed (missing lines correspond to no false positive classifications).

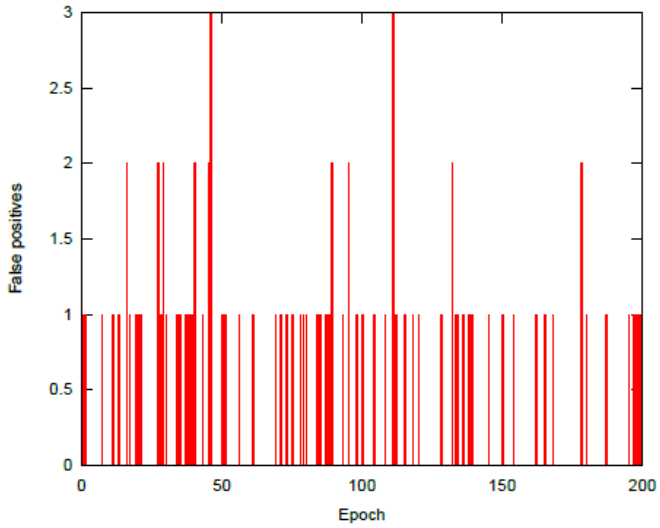

Figure 30: Learning Model FPR

2.7.3 A Machine Learning-based Approach to Build Zero False-Positive IPSs for Industrial IoT and CPS with a Case Study on Power Grids Security[14]

Current paper focuses on the enhancement of modern IDS systems used by next-generation firewalls through the development of a new type of classifiers referred as z-classifiers who make use of zero false-positive as metric for making decisions. This is achieved with the use of an iterative algorithm and several reduced datasets (which are generated from the initial dataset where some of the positive samples are removed in order to be more separable).

Before the analysis of the proposed algorithm, a modification of a classic SVM classifier is evaluated and resulted in non-acceptable results. For the set classification the CART (Classification And Regression Trees) algorithm is used as the core classifier since the project attempts to build a firewall specifically for a Power Grid Monitoring System. For the algorithm evaluation the KDD CUP'99 dataset is used.

[14] Mohammad Sayad Haghighi, Faezeh Farivar, Alireza Jolfaei, "A Machine Learning-based Approach to Build Zero False-Positive IPSs for Industrial IoT and CPS with a Case Study on Power Grids Security", IEEE, 23 July 2020

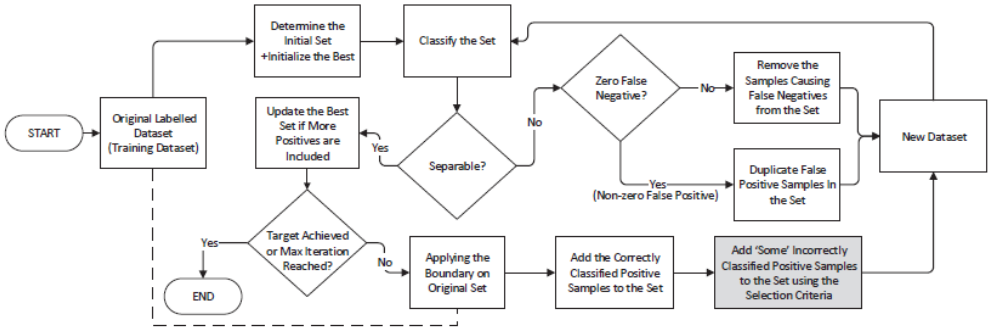

Figure 31: Flowchart of the proposed iterative algorithm

Below, a performance matrix is presented grouped by the number of iterations the algorithm used in order to classify the KDD CUP'99 dataset:

Iterations	TN	TP	FN	FP
10	289,542	138,869	65,589	0
50	289,542	204,167	291	0
100	289,542	204,425	33	0
500	289,542	204,432	26	0
1000	289,542	204,432	26	0

2.7.4 Web Application Firewall[15]

In this paper a simple Web Application Firewall is implemented filtering HTTP traffic based on pre-configured rules. In addition, an IDS system is developed which, through use of clustering algorithms, groups and categorizes packets among two different classes, "Normal" & "Intrusion". Finally, a 0-day attack detection technique is developed which is triggered based on the abnormal increase of the "intrusion" classified incoming packets.

The clustering algorithms used on the current project are the *Leonid Portnoy* algorithm, which uses a single-linkage clustering method to group intrusion packets, and the more widely-used *K-Means* algorithm which has been further analyzed previously on the current thesis. Both of the above algorithms were tested against the KDD Cup 1999 dataset where Leonid Portnoy presented a result of 65% TP classifications and K-Means a result of approximately 80% of TP classifications.

2.7.5 Web Application Attacks Detection Using Machine Learning Techniques[16]

Authors of the current thesis propose two different classification approaches based on different learning scenarios which are then compared against Modsecurity WAF configured with the OWASP Core rule set. The first approach consists of multi-class classifier when both valid

[15] Namit Gupta, Abakash Saikia, "Web Application Firewall", Department of Computer Science and Engineering Indian Institute of Technology, Kanpur, 30 April 2007

[16] Gustavo Betarte, Rodrigo Martinez, Alvaro Pardo, "Web Application Attacks Detection Using Machine Learning Techniques", Universidad de la Republica, Uruguay, 2018

and attack data are provided and the second approach is based on a one-class classifier which can be used when only valid data is provided.

The three different learning scenarios are:

i. Scenario 1: Valid application requests where legit and malicious requests are being labeled
ii. Scenario 2: Valid application requests with malicious classified requests which however are not specifically target the protected application
iii. Scenario 3: Only valid traffic requests are provided without being labeled as legit or malicious

The proposed multi-class classifier is applied for both Scenario 1 & 2 whereas the Scenario 3 is used for the testing of the single-class classifier.

The architecture of the learning process consists of the following components:

i. Parser: Analyzes and discriminates the parameters of the incoming requests (headers, query string, body, etc.)
ii. Tokenizer: Keeps track of the number of each feature's appearance
iii. Classifier: In scenario 1 the SVM, K-nearest neighbors and random forest classifiers are applied. Scenario 2 makes use again of the K-nearest neighbors and random forest classifiers whereas in scenario 3 the EM algorithm was used to estimate the parameters of a Gaussian Mixture Model (GMM).

Below a matrix is presented regarding the corresponding evaluation of each algorithm based on the scenario, algorithm and dataset used:

Scenario	Dataset	Algorithm	Precision	Recall	TPR	FPR
Baseline	DRUPAL	MODSECURITY W/OWASP CRS	0.11	0.76	76.22%	38.89%
	PKDD2007	MODSECURITY W/OWASP CRS	0.70	0.93	92.97%	57.21%
	CSIC2010	MODSECURITY W/OWASP CRS	0.50	0.34	34.32%	23.93%
*sc*1	DRUPAL	Random Forest	0.97	0.91	91.22%	0.05%
		KNN-3	0.97	0.89	88.97%	0.05%
		SVM	0.98	0.90	89.67%	0.04%
	PKDD2007	Random Forest	0.96	0.85	85.10%	1.67%
		KNN-3	0.96	0.70	70.39%	1.41%
		SVM	0.95	0.81	81.10%	1.91%
	CSIC2010	Random Forest	0.97	0.72	72.00%	1.47%
		KNN-3	0.95	0.65	65.03%	2.38%
		SVM	0.97	0.70	70.34%	1.26%
*sc*2	DRUPAL	Random Forest	0.95	0.48	47.57%	0.05%
		K-NN 3	0.90	0.07	6.99%	0.01%
	PKDD2007	Random Forest	0.96	0.23	22.56%	0.45%
		K-NN 3	1.00	0.12	11.70%	0.02%
	CSIC2010	Random Forest	0.91	0.32	32.06%	2.27%
		K-NN 3	0.66	0.50	50.43%	17.85%
*sc*3	DRUPAL	One-class w/λ=0.156 (test)	0.22	0.95	95.34%	22.15%
		One-class w/λ=0.156 (validation)	0.15	0.89	88.90%	26.83%
	PKDD2007	One-class w/λ=0.156 (test)	0.70	0.93	93.12%	58.30%
	CSIC2010	One-class w/λ=0.156 (test)	0.75	0.48	47.66%	11.15%

2.7.6 A comprehensive survey on machine learning for networking: evolution, applications and research opportunities[17]

The current survey conducts a deep dive research and provides a highly valuable insight upon the various Machine Learning techniques and their applicability on essential networking

[17] Raouf Boutaba, Mohammad A. Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar,

challenges with main objective the enhancement of traffic engineering, performance of future networks and network security. The networking sectors covered are listed as follows:

i. Traffic prediction
ii. Traffic classification
iii. Traffic routing
iv. Congestion control
v. Resource management
vi. Fault management
vii. QoS/QoE
viii. Network security

In each of these sectors the authors attempt a short analysis of the challenges that arise along with a brief report and comparison of the ML methodologies and algorithms being applied to address them. Below a sample table is displayed demonstrating a comparison of the ML techniques employed to conduct a network's traffic classification based either on the payload's (Haffner, Ma, Finamore) or the interconnecting hosts' (Schatzmann, Bermolan) behavior.

Ref.	ML Technique	Dataset	Features	Classes	Evaluation	
					Settings	Results
Haffner et al. [176]*	Supervised NB, AdaBoost, MaxEnt	Proprietary	Discrete byte encoding for first n bytes of unidirectional flow	FTP, SMTP, POP3, IMAP, HTTPS, HTTP, SSH	$n = 64 - 256$ bytes	Overall error rate <0.51%, precision > 99%, recall > 94%
Ma et al. [286]*	Unsupervised HCA	Proprietary: U. Cambridge, UCSD	Discrete byte encoding for first n bytes of unidirectional flow	FTP, SMTP, HTTP, HTTPS, DNS, NTP, NetBIOS, SrvLoc	$n = 64$ bytes, distance metric: PD = 250, MP = 150, CSG = 12%	Error rate: PD $\leq$ 4.15%, MP $\leq$ 9.97%, CSG $\leq$ 6.19%
Finamore et al. [146]*	Supervised SVM	Tstat [439]; NAPA-WINE [268]; Proprietary: ISP network	Statistical characterization of first N bytes of each packet a window of size C, divided into G groups of b consecutive bits	eMule, BitTorrent, RTP, RTCP, DNS, P2P-TV (PPLive, Joost, SopCast, TVAnts), Skype, Background	$C = 80, N = 12, G = 24, b = 4$	Average TP = 99.6%, FP < 1%
Schatzmann et al. [404]†	Supervised SVM	Proprietary: ISP network	Service proximity, activity profiles, session duration, periodicity	Mail, Non-Mail	N/A	Average accuracy = 93.2%, precision = 79.2%
Bermolan et al. [53]†	Supervised SVM	Proprietary: campus network, ISP network	Packet count exchanged between peers in duration ΔT	PPLive, TVAnts, SopCast, Joost	$\Delta T = 5$ s, SVM distance metric $R = 0.5$	Worst-case TPR $\approx$ 95%, FPR < 0.1%

Figure 32: Summary of Payload & Host Behavior-based Traffic Classification

2.7.7 fWaf - Machine Learning-driven Firewall[18]

In this project Faizan Ahmad, CEO of Fsecurity, attempted to develop and train a classifier to solve a binary classification problem where incoming web requests should be labelled either as “CLEAN” or “MALICIUS”. In order to train its classifier, logistic regression has been used, mainly due to its performance and ngrams as tokens with n=3. The dataset with the legitimate http requests (1000000 queries) was collected from SecRepo and the malicious queries (50000) from a certain Github repository .

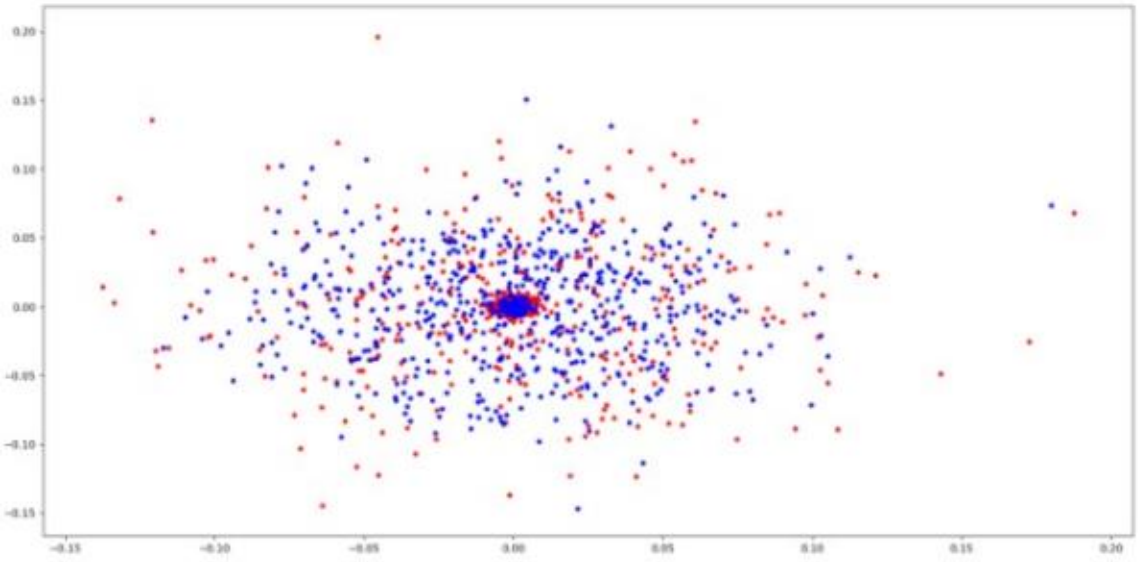

Figure 33: Visualization of legitimate (blue points) & malicious (red points) queries

In continuance, author used Tfidvectorizer to convert the data into tfidf values before employing the classifier so as to add certain weights on each ngram based on their probability to belong to a malicious query. The accuracy of the classifier is reported to be equal to 99% and sample of its results is listed below:

wp-content/wp-plugins ***(CLEAN)***
<script>alert(1)</script> ***(MALICIOUS)***
SELECT password from admin ***(MALICIOUS)***
"> ***(MALICIOUS)***
/example/test.php ***(CLEAN)***
google/images ***(CLEAN)***
q=../etc/passwd ***(MALICIOUS)***
javascript:confirm(1) ***(MALICIOUS)***
"> ***(MALICIOUS)***
foo/bar ***(CLEAN)***
fooooooooooooooooooooo ***(CLEAN)***
example/test/q=<script>alert(1)</script> ***(MALICIOUS)***
example/test/q= ***(MALICIOUS)***
fsecurify/q= ***(MALICIOUS)***
example/test/q= **(MALICIOUS)**

[18] Faizan Ahmad, “fWaf – Machine learning driven Web Application Firewall”, https://kdnuggets.com , GitHub repository link, February 2017

2.7.8 WAF-Brain[19]

WAF-Brain constitutes an open-source Web Application Firewall, developed by BBVA-Labs Security team, where ANNs are employed to monitor and filter incoming HTTP traffic. The payload of the HTTP requests is examined by the deployed model and if its content is marked as malicious the corresponding request is dropped by the WAF. The current models are trained to detect and mitigate only SQL Injection Attacks, however, there is the capability to utilize any custom model to extend its firewalling arsenal.

To evaluate WAF-Brain, a set of attacks were executed via the use of sqlmap, a popular SQL injection tool, and OWASP ZAP, an application vulnerability scanner. The same tests were deployed against Modsecurity WAF for terms of comparison. Below the corresponding results are being displayed:

Modsecurity:

Tool name	**Total Attacks**	**Blocked Attacks**	**Success Attacks**	**%**
Sqlmap	22.462	20586	1876	91.6
OWASP ZAP	57.652	47952	9700	83.2

WAF-Brain:

Tool name	**Total Attacks**	**Blocked Attacks**	**Success Attacks**	**%**
Sqlmap	22.458	21626	832	96.3
OWASP ZAP	57.254	49048	8206	85.7

[19] BBVA-Labs Security team, “WAF-Brain - the clever and efficient Firewall for the Web”, GitHub repository link , April 2018

3 Problem Statement

As indicated by the aforementioned related work, lots of different approaches have been designed, implemented and tested over time to address the challenge of developing security systems able to detect and prevent malicious payloads based on Machine learning techniques and not based on obsolete blocklists and static signature patterns. Since Machine Learning landscape is quite extensive, being able to apply in many aspects of everyday life, further investigation is needed to conclude on the best suitable algorithm qualified to address the objective of identifying a, for example, non-legit request made towards a public website.

In occasion of the above, the current thesis conducts research on the two big families of ML algorithms, those based on supervised and those based on unsupervised learning, to end up on the one that will build a model to be used by a web application firewall system.

To achieve this objective, the following main sections have been studied:

- **HTTP Request structure**: Before developing a classifier to mark a request as "malicious" or "legit", an analysis of a HTTP request status should be made to understand and conclude on the features of interest that should be used by the classifier.
- **HTTP Request parsing**: After noting these features, a tool to parse a HTTP request and extract those parts needs to be developed.
- **Machine Learning**: By knowing which features to extract, the creation of a dataset must take place. This dataset will be imported on several classifiers to build the corresponding models which will be, then, compared and evaluated.
- **Performance evaluation**: To decide which classifier is the most adequate for this thesis's' objective a performance evaluation will take place based on certain criteria and metrics already reported (chapter 1.6.5)
- **Web Application Firewall**: The most efficient classifier will be used to develop a web application firewall that will intervene the client requests, analyze and classify them accordingly. If the requests are considered as malicious, they will be dropped as intended.

The analysis and development of these sections are presented in the following approach and implementation chapters. The results are then reported in the corresponding chapter and the conclusion and potential future work are presented next.

4 Approach

In this chapter the approach method will be presented, which will be an analysis of the steps taken to collect the data sets of the HTTP requests, the selection of features to be extracted, the build of a classifier that will mark the requests as malicious or legitimate and its integration with a simple reverse proxy to protect a web application from injecting malicious payloads in real time.

4.1 HTTP Messages

Since the scope of the project is the classification of HTTP Messages, an analysis of their functionality and structure is considered of essential need.

4.1.1 HTTP Message types & protocols

HTTP Messages serve the purpose of data exchange between a client and a web server. When a client sends a HTTP Message to trigger an action from server's end this message is also referred as a "HTTP Request". The web server's reply is, accordingly, named as "HTTP Response". HTTP Messages are encoded in ASCII and are consisting from multiple lines. In HTTP/1.1, which is the most commonly used protocol, these messages were openly sent over the Internet whereas in HTTP/2, the once human-readable message is divided into HTTP frames before transmission for optimization and enhanced performance purposes.

4.1.2 HTTP Messages Structure

HTTP requests, and responses, share a similar structure as noted below:

- A start-line (HEAD) describing the transmitted requests (method & protocol) along with their status (successful or failed) represented by a certain code.
- A no-mandatory set of HTTP headers further specifying the message, or reporting characteristics of the body included in the message.
- A blank line containing request's metadata follows
- An optional body part carrying the data associated with the request (e.g., credentials of a HTML login page), or the document associated with a response. The presence of the body and its size is specified by the start-line and HTTP headers.

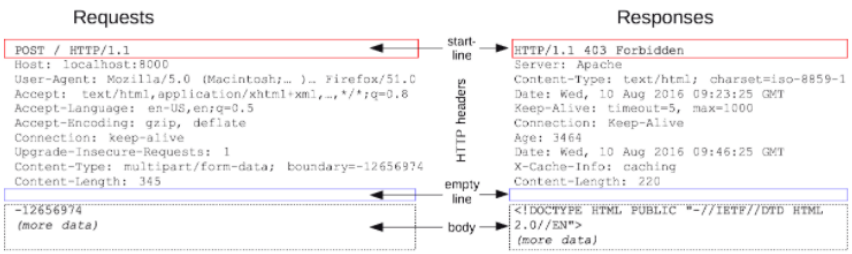

Figure 34: HTTP Messages Structure

4.1.3 HTTP Request

Since the messages originating from the client with the server as recipient are the ones of interest for the current project, their distinct characteristics are presented below:

Start-line consists of:

- a HTTP Method (like GET, PUT, POST, HEAD, etc.). The implementation that will take place focuses on the GET & POST methods where GET indicates that information should be retrieved from the server and POST sends data to the server
- The request target (URL, path, etc.)
- HTTP version (HTTP/1.0, HTTP/1.1, etc.)

HTTP Headers follow a form of *Header: value* (one per line) and their structure vary according the header type. Different requests may include different HTTP Headers which may be divided in a set of groups:

- General headers like “Connection” which apply to the whole message
- Request headers (Host, User Agent, Accept, etc.)
- Representation headers (Content-type, Content-length) that describe the “body” part of the message. No existence of these headers in a request imply that no data is transmitted on the “body” section

Body, as already mentioned, contains data destined to the web server in order to update it and is mostly found on POST-method requests. A body part can either be single-resource or multiple resource depending of the number of parts it consists from.

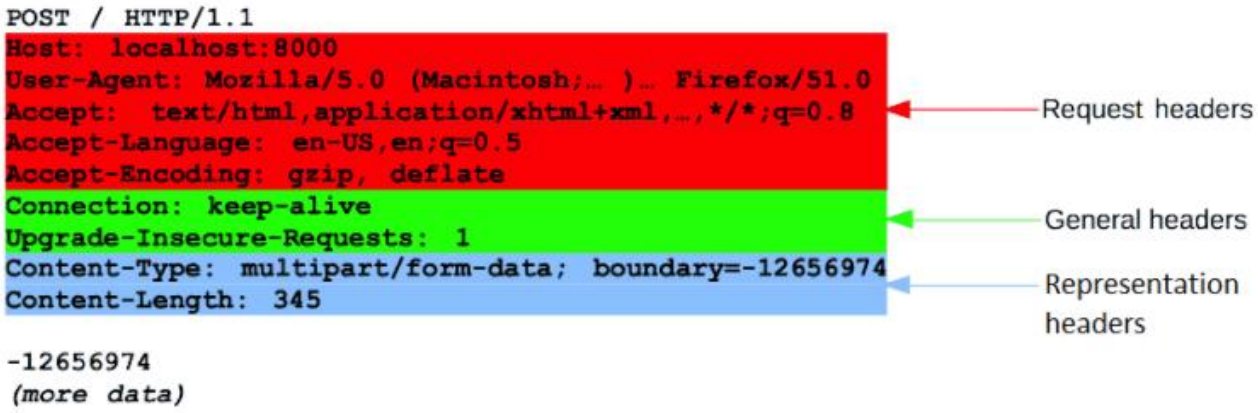

Figure 35: HTTP POST Request

4.2 Feature Selection & Extraction

4.2.1 Feature Selection

Feature selection is the process of defining the attributes of most interest for the classification task of data and consist the basis on which the Machine Learning model will be built and trained.

Since the current objective is the development of a web application firewall system to detect and prevent known web application attacks, the decisions and conventions that should be taken were the following:

- Scope of the web application attacks that the implemented WAF system should provide mitigation to. Taking into consideration the OWASP Top 10 Web Application Security Risks[20], the selection of Injection-based attacks has been made and more specific the attacks:
  - Cross-site Scripting (XSS)
  - SQL-Injection
  - Command Injection

  The above selection has been based on two main pillars which are the severity (No3 of OWASP's aforementioned list) and the occurrence frequency of these types of attacks on the today's enterprise web application security landscape.

- Following the definition of the above attacks, and since the security control against them is the intensive analysis and accurate classification of the incoming HTTP requests, there should be decided which attributes of a HTTP Request should be extracted and examined to provide the desired result. Based on relevant researches[21], the number of special characters on the target URL path will be extracted as attributes of interest and more specific the characters:
  - Single quotes – " **'** "
  - Double quotes – " **"** "
  - Dashes – " **-** "
  - Braces – " **(** "
  - Spaces

  In addition, in order to reduce the training time and the performance of the developed model, a set of keywords which characterize the selected attacks will be used as displayed below:

  sql_badwords *= ['drop','select','where','from','table','if', 'if (1=1) then', 'if (1=1) select', 'concat', 'char', 'union', 'group by', 'having', 'order by', 'insert', 'exec', 'limit', 'waitfor', 'delay', 'sleep']*

  xss_badwords *= ['script', 'alert', 'prompt', 'eval', 'onclick', 'onerror', 'onpropertychange', 'onresize', 'onload', 'onmouse', 'onblur', 'onkey', 'onfocus', 'fromCharCode', 'ontoggle', 'expression', 'foo']*

[20] https://owasp.org/www-project-top-ten/

[21] Sara Althubiti, Xiaohong Yuan, Albert Esterline, "Analyzing HTTP requests for web intrusion detection", DigitalCommons@Kennesaw State University, 2017

ci_badwords *= ['wget', 'etc', 'passwd', 'cmd', 'cat', 'system', 'bin', 'curl', 'dir', 'echo', 'whoami', 'ifconfig', 'ipconfig', 'netsh', 'netstat', 'net use', 'perl', 'phpinfo', 'reg add', 'print', 'echo']*

The datasets based on which the SQL injection and XSS keyword sets have been defined were collected from the Kaggle repository while the Command Injection keyword set was defined based on the datasets created specific for the purposes of the project with the use of the open source command injection tool "Commix".

4.2.2 Feature Extraction

In order to extract the defined features from raw HTTP requests and represent them as numeric-type values, the development of a HTTP Request Parser has been conducted (Deliverable: "HTTP Request Parser.py") in programming language Python and its functionality will be further analysed on the "Implementation" chapter.

4.3 Data Sets Creation Tools

To create the labelled data sets needed, the generation of legitimate and malicious payloads corresponding to the defined attack scenarios (SQL Injection, XSS, Command Injection) should take place. The tools used for this process will be presented on the "Implementation" chapter on section "Implementation Tools".

4.4 Machine Learning Classification Architecture

The overall process of classifying the incoming HTTP Requests as legitimate or malicious can be analysed as below:

The classification is supervised because the classification labels are appended. The requests will be classified as "SQL Injection", "Command Injection", "XSS" & "Legit". The parsed requests' dataset (Deliverable: "Labelled Dataset.csv") will include a column named "Class" which will define the corresponding label of request so that the ML algorithm can adapt and built its model with the best possible effort.

As already stated, the numeric features that the dataset will include and will characterize the request will be:

- Number of single quotes
- Number of double quotes
- Number of braces
- Number of spaces
- Number of SQL Injection – keywords (referred as "SQL Badwords")
- Number of XSS – keywords (referred as "XSS Badwords")
- Number of Command Injection – keywords (referred as "Command Injection Badwords")

The Request parts “Method”, “Path” & “Body” will also be included but they will be ignored from the evaluation process to avoid complexity increase and unbiased results.

A	B	C	D	E	F	G	H	I	J	K	L
metho	path	body	single_q	double_q	dashes	braces	spaces	SQL Badwords	XSS Badwords	Command Injection Badords	class
GET	/		0	0	0	0	0	0	0	0	Legit
GET	/index.jsp?content=inside_careers.		0	0	0	0	0	0	0	0	Legit
GET	/index.jsp?content=personal_othe		0	0	0	0	0	0	0	0	Legit
GET	/index.jsp?content=business_lendi		0	0	0	0	0	0	0	0	Legit
GET	/index.jsp?content=inside_press.ht		0	0	0	0	0	0	0	0	Legit
GET	/index.jsp?content=personal_loans		0	0	0	0	0	0	0	0	Legit
GET	/index.jsp?content=personal_savin		0	0	0	0	0	0	0	0	Legit
GET	/robots.txt		0	0	0	0	0	0	0	0	Legit

Figure 36: Sample output of Labelled dataset

The summary of the processes included on the current object’s Machine Learning architecture are properly displayed on the below diagram:

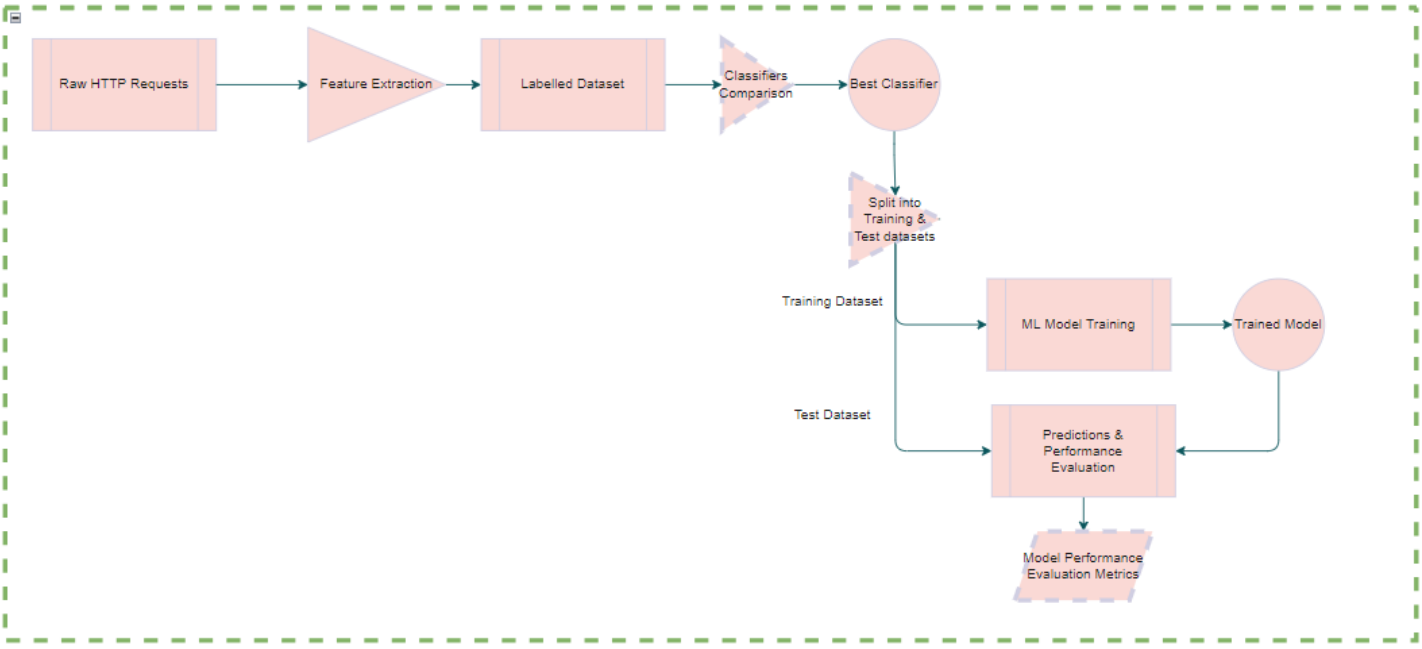

Figure 37: Machine Learning Architecture Processes

More analytically, the raw HTTP requests are entering the pipeline and the “Feature extraction” process occurs along with the labeling of each request based on whether is a legit, sql injection, xss or command injection payload. The labeled dataset is then imported, normalized and used for the training of 18 different classifiers. The performance metrics of all the trained models are then compared, cross evaluated and the one with the best total metrics (Accuracy, AUC, Recall, Precision, etc.) is chosen as the one more efficient, based on the defined features, to be used for the classification task.

Since the dataset is already imported, is then split in Training & Testing sets for the training and the test/prediction phase accordingly. After the completion of both phases the model is finalized and its performance metrics are available for evaluation.

4.5 Machine Learning WAF System Architecture

Now that the best effective classifier is selected, a new actions pipeline is initiated where the model is again built, trained, tuned and finalized. Finally, a reverse proxy in Python will be developed (Deliverable: "ML-WAF.py"), intercepting the incoming HTTP requests destined to a preconfigured host, and via its integration with the previously prepared ML Model should be able to detect and address malicious traffic by proper classification of http packets. The detailed summary of processes is attached below:

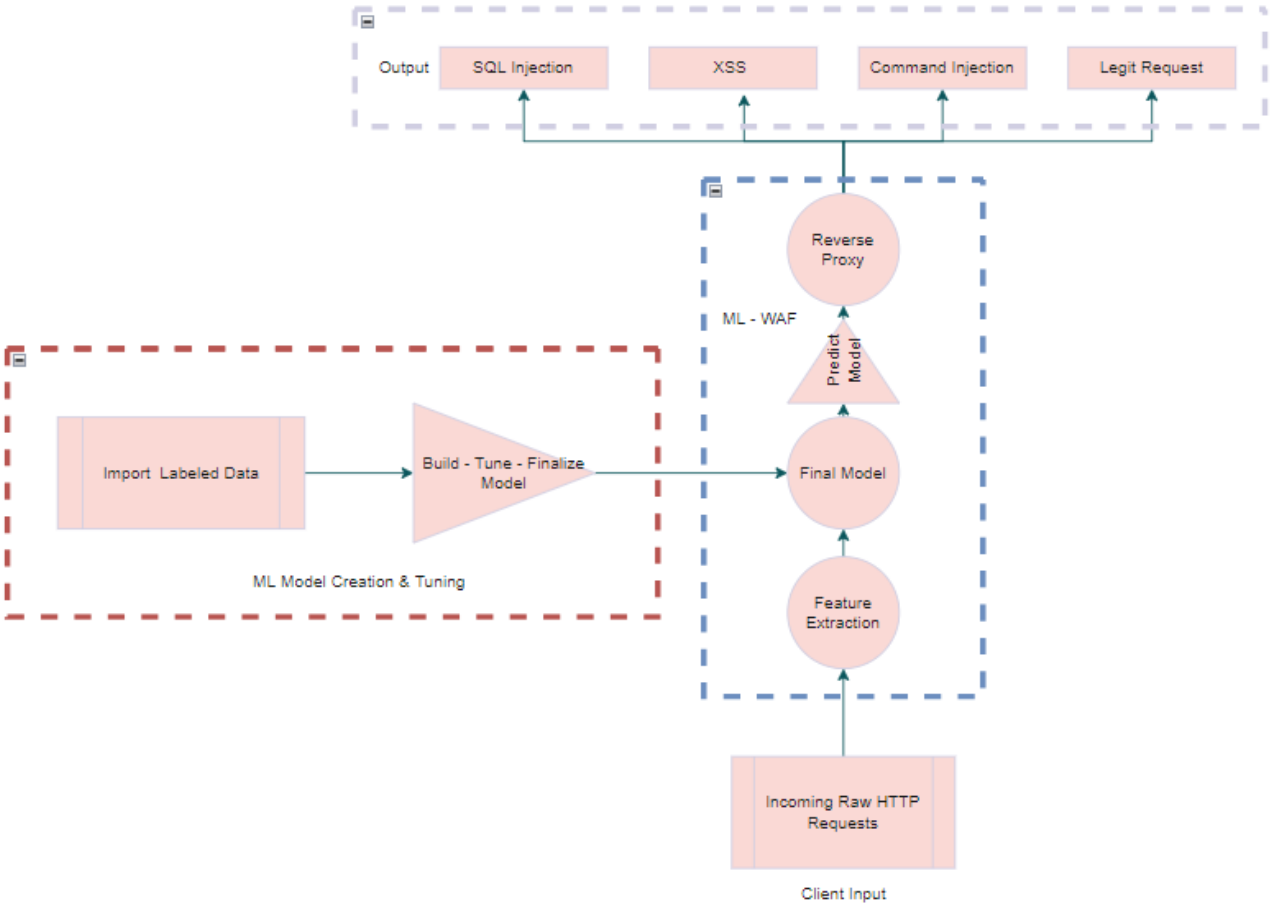

Figure 38: WAF System Process Summary

5 Implementation

In this chapter the implementation of the previously shared architectures will be presented. The analysis of the implementation will be based on the flow charts already shown above. The relevant code for every part will be presented and explained. More specifically, there will be a description of the implemented code's parts, including functions, classes and the way it operates. Any information about included libraries and modules will be provided at the parts of code in which they were used. All the mentioned deliverables can be found at the project's Github repo.

5.1 Implementation Tools

Before proceeding on the implementation's states analysis, the complete list of tools used should be presented.

5.1.1 Programming Language

First and foremost, the programming language used for the development is the Python programming language. More specifically the version of Python that was used is the 3.7.9 version released on August 17 of 2020. Python was selected over other programming languages due to various factors:

- Comes with a wide variety of libraries and frameworks for provisioning, managing and visualizing Machine Learning models
- Extensive documentation and resources
- Easy to read code
- Fast development due to user-friendly syntax
- Multiple platforms for code review and testing

5.1.2 Dataset Creation tools

In order to create the labelled dataset consisting of malicious and legitimate HTTP requests a series of penetration testing tools have been used and presented:

- **Burp Suite**: A Java-based application for assessing and analysing a web application's security. Its main features include a proxy server, a spide, an intruder and a repeater for request automation. Through the use of Burp's proxy server, the capture and export of raw HTTP requests has been handled.
- **Screaming Frog SEO Spider:** Industry leading website crawler with a free-tier available. The use of the crawler assisted in quick and efficient generation of legitimate traffic to be used for the labelled dataset.
- **XSStrike:** Cross Site Scripting scanner with intelligent payload injection capabilities in order to generate malicious XSS-related requests for the labelled dataset.
- **SQLmap**: Open-source penetration testing tool exploiting SQL injection flaws and automates detection of relevant vulnerabilities.

- **Commix**: Open-source penetration testing tool automating detection and exploitation of command injection vulnerabilities.

5.1.3 Machine Learning tools

Afterwards, the selection of the tools, that will assist on the classifier selection as long as model building, tuning, plotting and finalizing is considered essential:

- **Anaconda (Python distribution)**: Distribution of Python (and R programming) language for scientific computing such as machine learning applications and large-scale data processing that simplifies package management and deployment.
- **Jupyter Notebook**: Web-based interactive computational environment for creating notebook documents.
- **Visual Studio Code**: Source-code editor supporting a wide variety of programming languages, including Python, with enhanced support for code development such as syntax highlighting, bracket matching and code folding.
- **PyCaret**: Open source, low-code, machine learning Python library allowing from preparing data to rapidly deploying models from various notebook environments.

5.2 Implementation Phases Analysis

Now that all the tools used for the project's purposes have been reported, the analysis of all the implementation phases, which are presented at high level on Figures 37 & 38, will follow.

5.2.1 Labelled Dataset Creation

As for any classification task, a labelled dataset should be provisioned in order all the proposed algorithms to be trained by and compared accordingly. To meet this objective, as already mentioned, a "HTTP Request Parser" has been developed.

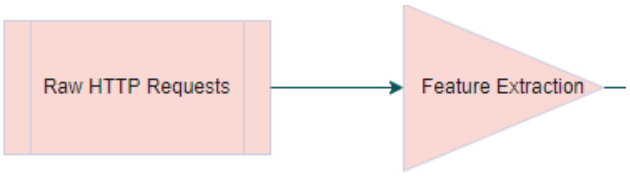

Figure 39: Labelled Dataset Creation Phase

To generate the malicious payloads for each of the injection attacks of interest, the corresponding penetration testing tool has been used against public web applications specifically used for security testing (e.g. http://demo.testfire.net/, https://brutelogic.com.br/blog/). All the malicious payloads' exploitation has been delivered

through the Burp Suite's proxy to capture and log the requests towards the targeted web servers as displayed below:

Figure 40: SQL Injection Vulnerability scan via Proxy

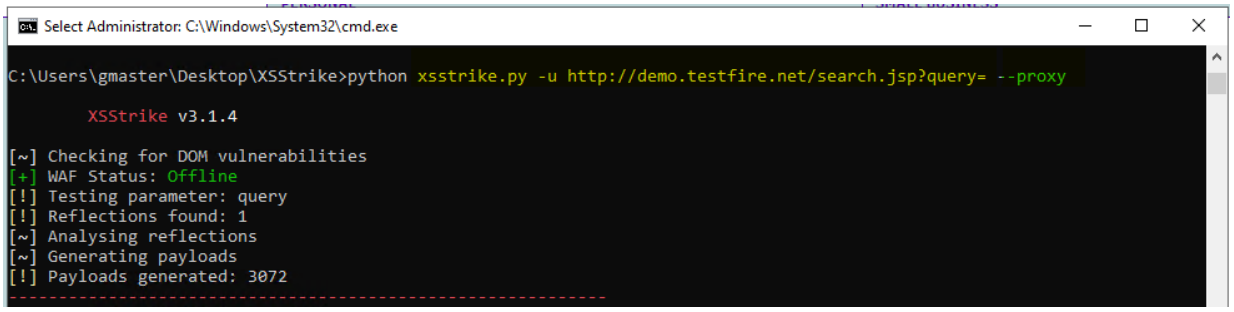

Figure 41: XSS Vulnerability Scan via Proxy

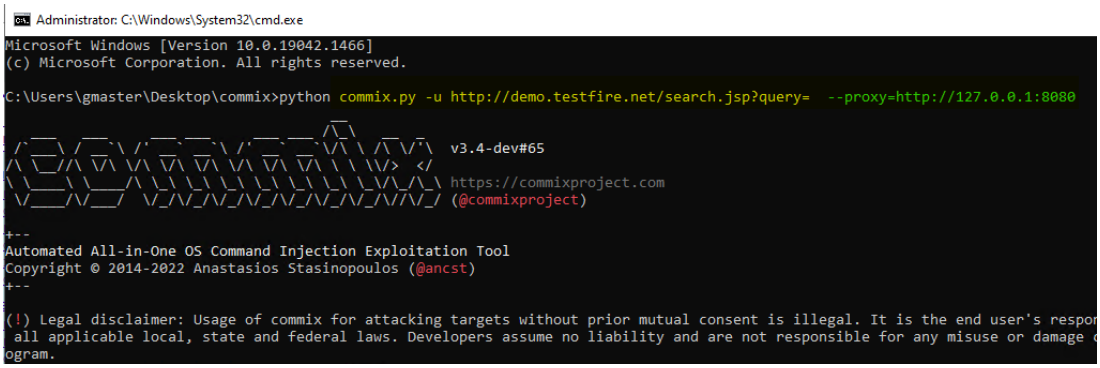

Figure 42: Command Injection Vulnerability Scan via Proxy

Except from the collection of HTTP requests regarding malicious activity, legitimate requests needed to also been captured for the complete dataset. For that purpose, Screaming Frog SEO Spider has been preferred to simply crawl target websites to collect the needed data.

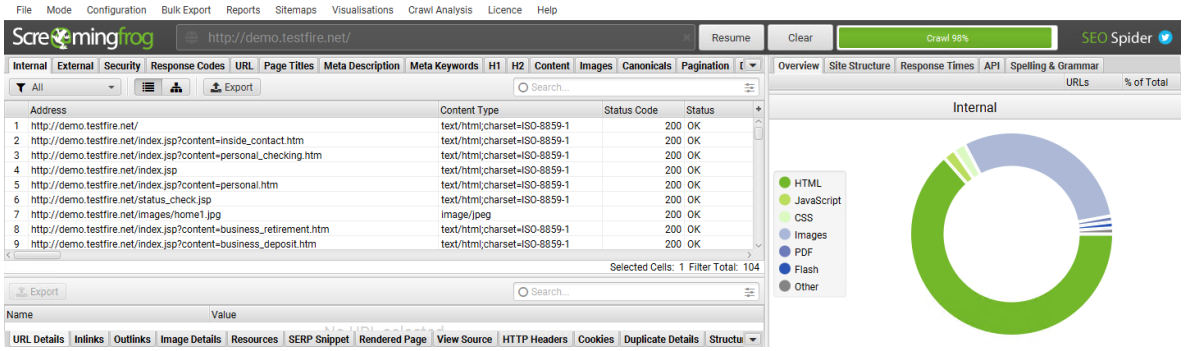

Figure 43: Website crawling

After the collection of raw data stage is completed, its parsing & labeling should take place. Both of these tasks are carried out from the "HTTP Request Parser" tool, part of its code is displayed:

HTTP Request Parser.py

```
5
6   log_path = 'xss_requests' #Raw HTTP Requests location
7   output_csv_log = 'xss_requests.csv' #Exported Parsed HTTP Requests
8   class_flag = "XSS Injection" #Label definition
9
10  #HTTP Request Parsing
11 class LogParse:
59
60
61  sql_badwords = ['drop', 'select' ,'where' ,'from' ,'table', 'if (1=1) th
62  xss_badwords = ['script', 'alert', 'prompt', 'eval', 'onclick', 'onerror
63  ci_badwords = ['wget', 'etc', 'passwd', 'cmd', ' cat ', 'system', 'bin',
64
65  #Enumerate & Export Request's special Chars & Badwords
66 def ExtractFeatures(method,path_enc,body_enc,headers):
```

Figure 44: HTTP Request Parser source code

As presented, the parser receives as input the raw data file and provides as output a csv file containing the parsed requests with the corresponding label (Figure 36).

5.2.2 Best Classifier Determination

In continuance, the determination of the more efficient classifier needs to be determined.

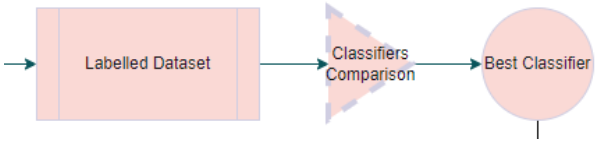

Figure 45: Best Classifier Determination Phase

To achieve this, the Pycaret libraries (pycaret.datasets, pycaret.classification) will be used and developed with the assistance of Jupyter Notebook from the Anaconda environment (Deliverable: "Classifier Determination & Modeling.ipynb"). Firstly, the labelled dataset will be imported (via the pandas library) and normalized. Then, the compare_models() function will compare all 18, Pycaret supported, classifiers to evaluate their performance (Accuracy, AUC, Recall, Precision, etc.)

Based on the subject Dataset, the ExtraTreesClassifier has presented the most optimal performance evaluation results and, therefore, will be selected for the current project's classification tasks:

```
In [7]: print(best_classifier)

ExtraTreesClassifier(bootstrap=False, ccp_alpha=0.0, class_weight=None,
                     criterion='gini', max_depth=None, max_features='auto',
                     max_leaf_nodes=None, max_samples=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=-1,
                     oob_score=False, random_state=123, verbose=0,
                     warm_start=False)
```

Figure 46: ExtraTreesClassifier Characteristics

5.2.3 Build & Train Model

Now that the classifier has been defined, the Model building and training should follow. First, a part of the total dataset (90%) will split to be used for the learing phase (use of method dataset.sample(frac=0.9, random_state=786)) and the remaining data (10%) for predictions (method: dataset.drop).

```
In [4]: #Split data for model build and data for predictions
        data = dataset.sample(frac=0.9, random_state=786)
        data_unseen = dataset.drop(data.index)

        data.reset_index(drop=True, inplace=True)
        data_unseen.reset_index(drop=True, inplace=True)

        print('Data for Modeling: ' + str(data.shape))
        print('Unseen Data For Predictions: ' + str(data_unseen.shape))

        Data for Modeling: (3412, 12)
        Unseen Data For Predictions: (379, 12)
```

Figure 47: Data Split

Afterwards, through the create_model() function, the initial model will be built and trained.

```
In [8]: #Build Model
        et_model = create_model('et')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1	0.9916	0.9977	0.9844	0.9917	0.9916	0.9882	0.9882
2	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
3	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
5	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
6	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
7	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
8	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
9	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Mean	0.9992	0.9998	0.9984	0.9992	0.9992	0.9988	0.9988
SD	0.0025	0.0007	0.0047	0.0025	0.0025	0.0035	0.0035

Figure 48: Model Training

Pycaret, except of model training, offers the capability of tuning the deployed model, enhancing accuracy, by optimizing its hyperparameters through the Random Grid algorithm. To take advantage of this capability, the tune_model() function is available to use.

```
In [9]: tuned_et_model = tune_model(et_model)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1	0.9916	0.9960	0.9844	0.9917	0.9916	0.9882	0.9882
2	0.9958	0.9987	0.9881	0.9959	0.9958	0.9941	0.9941
3	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
4	0.9958	1.0000	0.9881	0.9959	0.9958	0.9941	0.9941
5	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
6	0.9958	1.0000	0.9881	0.9959	0.9958	0.9941	0.9941
7	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
8	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
9	0.9916	1.0000	0.9762	0.9917	0.9914	0.9881	0.9882
Mean	0.9971	0.9995	0.9925	0.9971	0.9970	0.9959	0.9959
SD	0.0033	0.0012	0.0082	0.0032	0.0033	0.0046	0.0046

Figure 49: Model Tuning

Following the model's tuning, the prediction phase takes place (method: predict_model()) using a dataset sample of the initial dataset, also known as "test / hold-out Sample", to gain insights on the resulted model's performance metrics.

```
In [12]: predict_model(tuned_et_model)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Extra Trees Classifier	0.9922	0.9997	0.9845	0.9922	0.9921	0.9891	0.9891

Out[12]:

	single_q	double_q	dashes	braces	spaces	SQL Badwords	XSS Badwords	Command Injection Badords	class	Label	Score
0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	XSS	XSS	0.9846
1	1.0	0.0	0.0	0.0	4.0	2.0	0.0	0.0	SQL Injection	SQL Injection	1.0000
2	0.0	0.0	0.0	2.0	0.0	0.0	0.0	1.0	command injection	command injection	0.9524

Figure 50: Model Predictions

Lastly, the finalization of the model occurs (method: finalize_model()) and a prediction upon the initial split prediction-dataset will take place. The performance evaluation of the final model will be further analyzed on the "Results" section.

```
In [13]: final_et = finalize_model(tuned_et_model)

In [14]: unseen_predictions = predict_model(final_et, data=data_unseen)
         unseen_predictions.head()
```

Out[14]:

	method	path	body	single_q	double_q	dashes	braces	spaces	SQL Badwords	XSS Badwords	Command Injection Badords	class	Label	Score
0	GET	/index.jsp?content=personal_loans.htm	NaN	0	0	0	0	0	0	0	0	Legit	Legit	0.9977
1	GET	/index.jsp?content=personal_deposit.htm	NaN	0	0	0	0	0	0	0	0	Legit	Legit	0.9977
2	GET	/index.jsp?content=business_other.htm	NaN	0	0	0	0	0	0	0	0	Legit	Legit	0.9977
3	GET	/index.jsp	NaN	0	0	0	0	0	0	0	0	Legit	Legit	0.9977
4	GET	/index.jsp?content=business_insurance.htm	NaN	0	0	0	0	0	0	0	0	Legit	Legit	0.9977

Figure 51: Model Finalize & Prediction Phase

5.2.4 Machine Learning WAF Development

The last phase of implementation presents the main objective of the current thesis, which is the significant security enhancement & modernization of a traditional Web Application Firewall system added by its integration with a properly trained ML model. The predictive capabilities of the model simplify the administration effort a WAF requires, making it at the same time less prone to zero-day attacks and less dependent from multiple static attack signatures.

The overall proposed architecture of the developed ML-WAF is displayed on the below diagram:

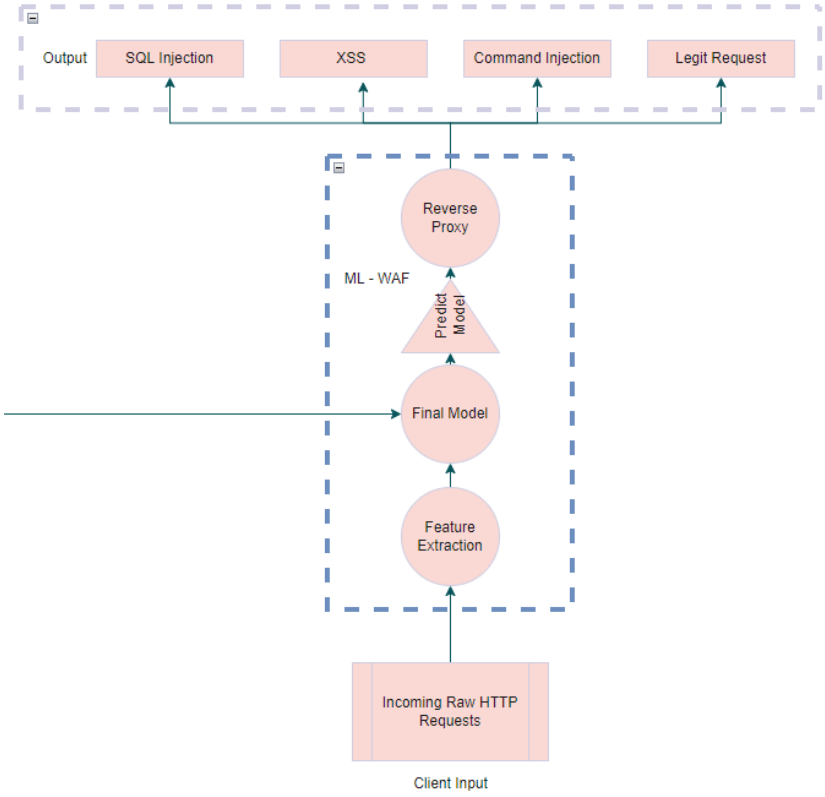

Figure 52: Proposed ML-WAF Architecture

For the project purposes, a reverse proxy system has been developed (Deliverable: ML-WAF.ipynb) which will take as input the HTTP Requests destined to the protected web application which at this case will be the website http://demo.testfire.net . The security system will run on a virtual machine hosted on Microsoft Azure cloud.

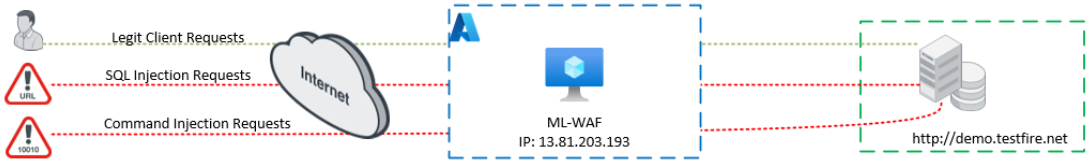

Figure 53: ML-WAF Project Topology

Depending on the predictions of the previously built model, will either forward the requests to their destination or, if the request is identified as malicious payload, will revert with a “403 – Forbidden error” alongside with an error message to the request’s origin while at the same time raise an alert to the system administrators including the detected malicious payload, the attack type it belongs to and its source IP.

Figure 54: ML-WAF Output

On Figure 55 a sample display of the ML-WAF’s output is presented. The green highlighted parts correspond to the HTTP Requests identified as legit resulting in a HTTP “200 - OK” status code. On the contrary, the pink highlighted part points to an alert that a malicious payload was identified. The HTTP Request triggered the alert is displayed on the “Malicious Request:” tab, the IP from which the Request was originated is also visible and finally the administrator is informed that a “SQL Injection” attack has been detected from the corresponding message. With orange is highlighted the part that logs the error message “Request blocked by ML-WAF” sent to the Request sender along with the 403 error.

Below the error presented to the client’s browser is also noted.

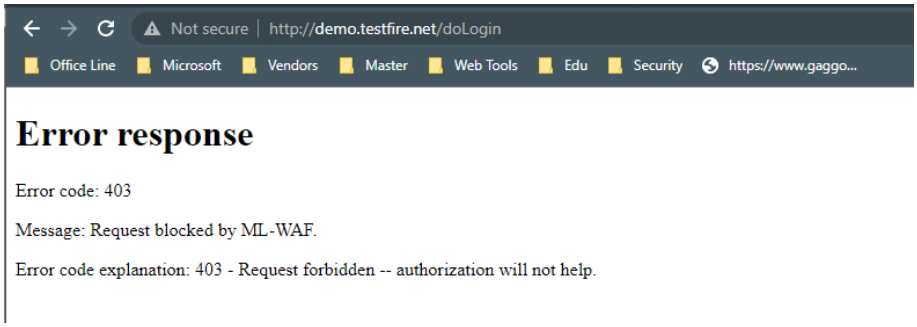

Figure 55: Client Browser - Error 403

6 Results

In this chapter the results of the implementation described before will be presented. More specifically the performance evaluation metrics that will be pointed out regarding the initially compared classifiers are:

- Accuracy
- AUC
- Recall
- Precision
- F1 Measure
- Kappa
- Matthews Correlation Coefficient (MCC)
- Training time for each of the compared classifiers

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
et	Extra Trees Classifier	0.9992	0.9998	0.9984	0.9992	0.9992	0.9988	0.9988	0.1920
rf	Random Forest Classifier	0.9987	0.9998	0.9972	0.9988	0.9987	0.9982	0.9982	0.2100
lightgbm	Light Gradient Boosting Machine	0.9983	0.9993	0.9961	0.9983	0.9983	0.9976	0.9976	0.0550
ada	Ada Boost Classifier	0.9979	0.9990	0.9949	0.9979	0.9979	0.9970	0.9971	0.0380
gbc	Gradient Boosting Classifier	0.9975	0.9997	0.9937	0.9975	0.9975	0.9965	0.9965	0.1790
dt	Decision Tree Classifier	0.9971	0.9979	0.9941	0.9971	0.9971	0.9959	0.9959	0.0060
svm	SVM - Linear Kernel	0.9962	0.0000	0.9926	0.9963	0.9962	0.9947	0.9947	0.0280
knn	K Neighbors Classifier	0.9954	0.9985	0.9918	0.9955	0.9954	0.9935	0.9935	0.0560
lda	Linear Discriminant Analysis	0.9954	0.9990	0.9943	0.9955	0.9954	0.9935	0.9936	0.0070
lr	Logistic Regression	0.9950	0.9996	0.9881	0.9950	0.9949	0.9929	0.9929	0.4830
ridge	Ridge Classifier	0.9933	0.0000	0.9883	0.9934	0.9933	0.9906	0.9906	0.0070
nb	Naive Bayes	0.9137	0.9985	0.7557	0.8915	0.8783	0.8746	0.8845	0.0140
dummy	Dummy Classifier	0.3568	0.5000	0.2500	0.1273	0.1876	0.0000	0.0000	0.0050
qda	Quadratic Discriminant Analysis	0.0871	0.0000	0.2500	0.0076	0.0140	0.0000	0.0000	0.0140

Figure 56: Classifiers' Performance Metrics Comparison

A worthy assumption that can be made based on the above table is that the Extra Trees Classifier, that has been used for the project's implementation, despite having the 3rd higher training time (0.1920sec) behind Random Forest and Logistic Regression classifiers manage to surpass the rest classifiers on all the other performance metrics with a significant high score on those of Accuracy and Recall (0.9992).

The performance results that are going to be examined next are the ones of the finalized model that was integrated with the deployed Web Application Firewall:

- **Confusion Matrix**: Comparison of the actual target values with those predicted by the machine learning model

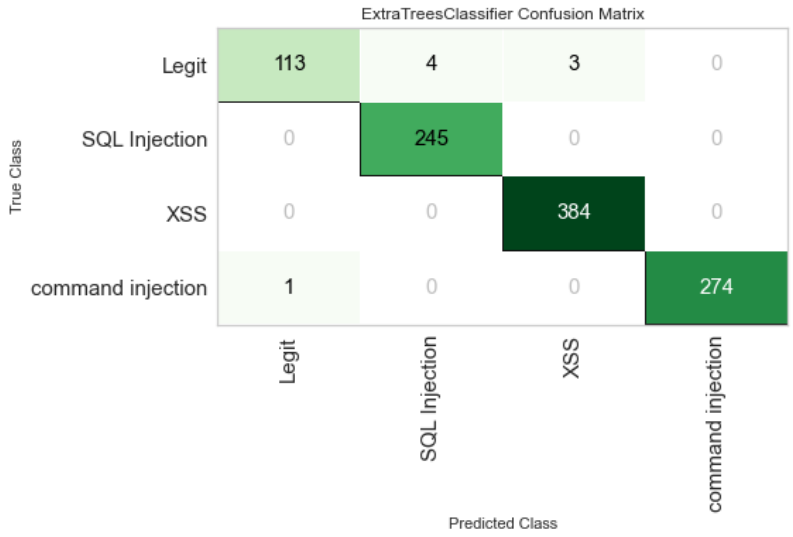

Figure 57: ML-WAF Model Confusion Matrix

- **Learning Curve**: Line plot of learning (y-axis) over experience (x-axis)

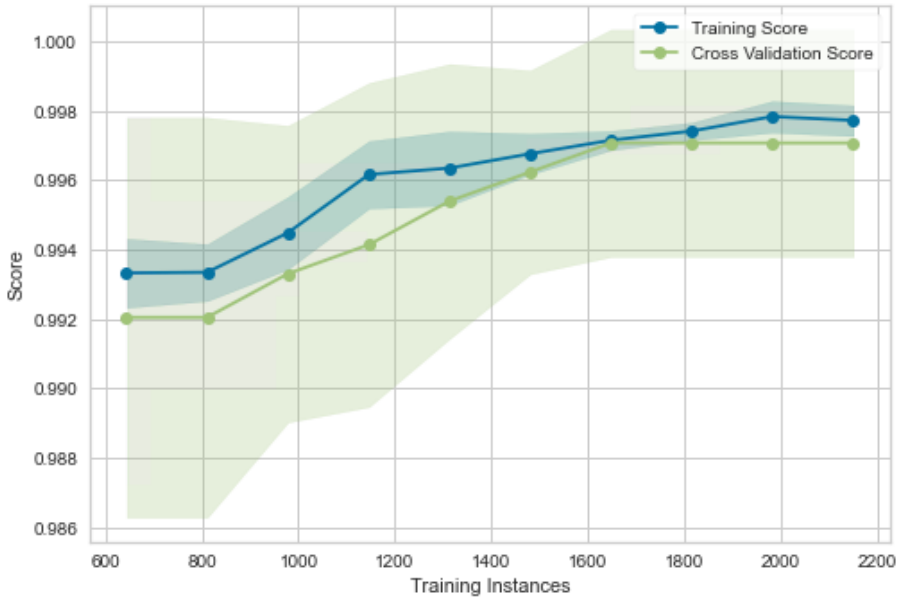

Figure 58: ML-WAF Model Learning Curve

- **Decision Boundary**: Set of hyper-planes that separate the data points into specific classes, where the algorithm switches from one class to another

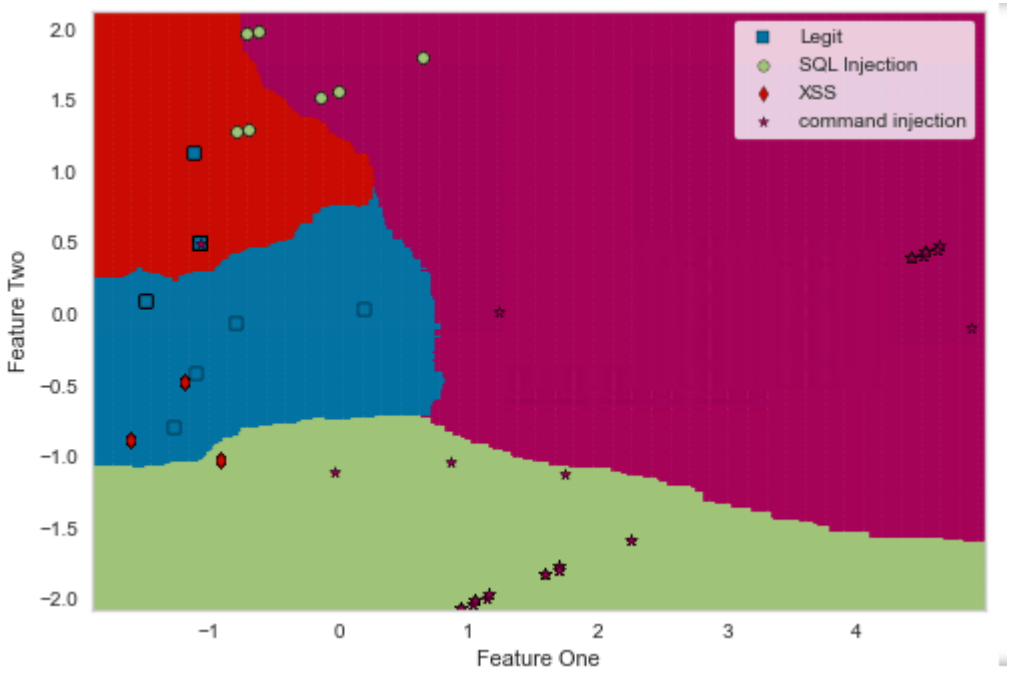

Figure 59: ML-WAF Model Decision Boundary

- **Feature Importance Plot**: Top features contributing on the model predictions

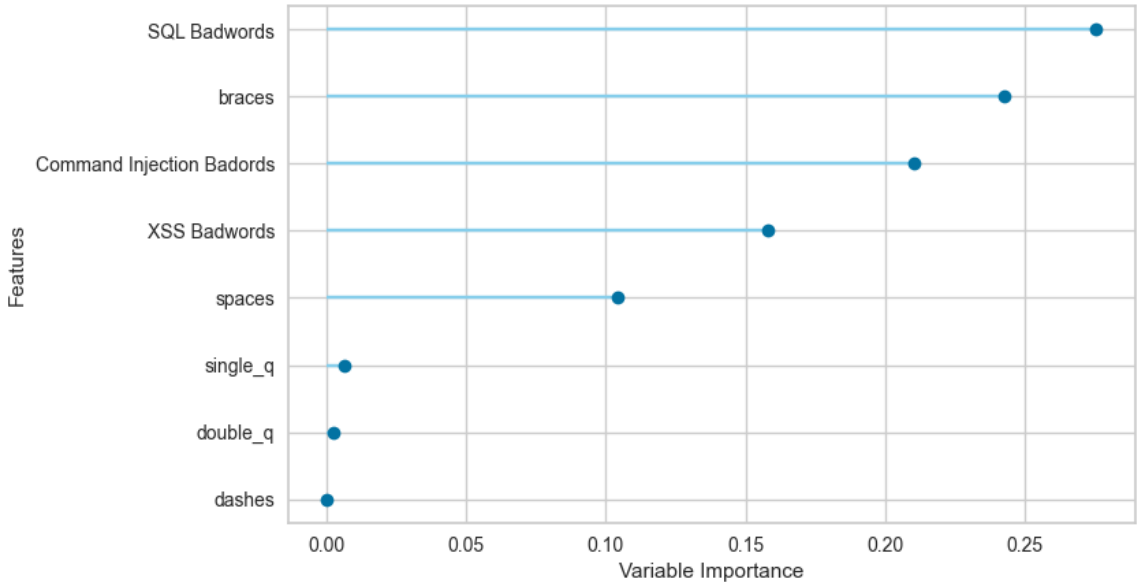

Figure 60: ML-WAF Model Feature Importance

- **Class Prediction Error**: Graphical representation of the mapping between predicted & actual classes

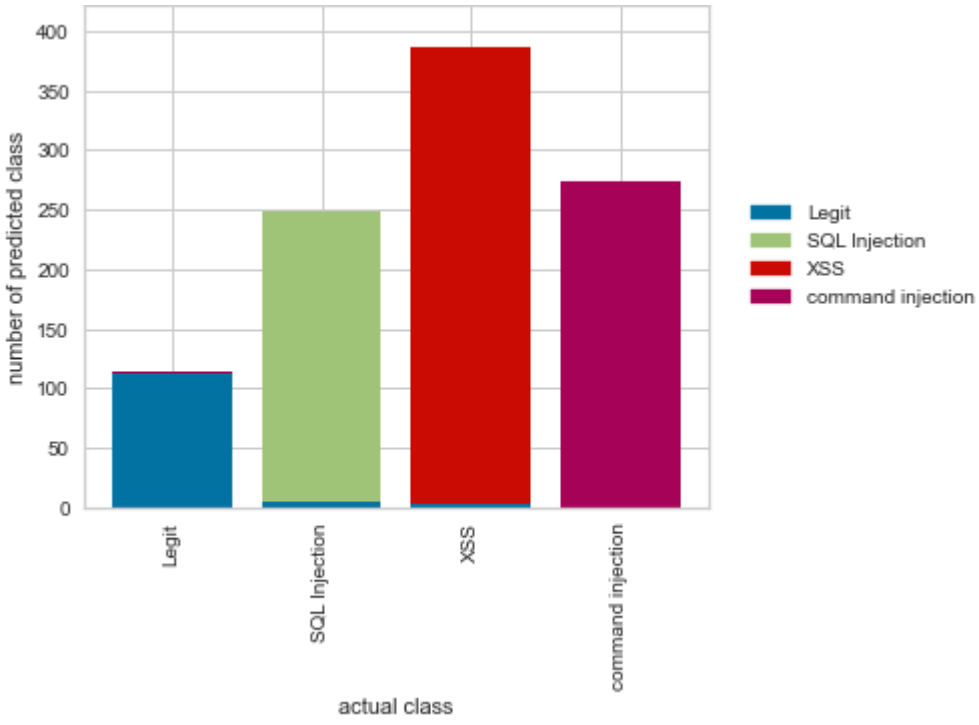

Figure 61: ML-WAF Class Prediction Error

7 Conclusion & Future Work

The current project attempted a holistic analysis of the security risks and exploits that consist a totally dynamic attack surface against modern networks ad web applications. As technology advances the malicious activities become more and more sophisticated turning traditional defending mechanisms such as firewalls inadequate of handling them. To address these continuously raised challenges, network and application security systems need to adopt state of the art technologies to improve the depth of their security controls. This is where Machine Learning takes over which through its unique capability of data exfiltration and analysis can conduct classification and clustering tasks that, in turn, may be used by the aforementioned security systems to efficiently identify threats and risks before any kind of compromise occur.

In this thesis, a proposed classifier has been examined and evaluated against a specific dataset containing certain malicious requests. As analytically presented, its proper training resulted in building a model which through its integration with a simple reverse proxy formed a web application firewall system able to detect and prevent attacks in real time.

Since this implementation just covered a small part of the iceberg's tip, additional suggestions follow that could be used for further research on how the IT security will achieve taking advantage of most of Machine Learning's potential.

7.1 Selected Features Extension

For sure, the addition of more features based on which the training of the model would take place, would lead in significant enhancement of the model's prediction capabilities despite the possible increase of the dataset's complexity. Features that could be added on the HTTP Parsing process are:

- Length of the HTTP Requests
- Length of the arguments
- Number of the arguments
- Path length
- Number of the arguments' letters
- Maximum request's byte value

7.2 Expansion of the identify attack surface

The import of more labeled datasets containing various non-specific payloads combined with the extension of the requests' extracted features could lead on building a model capable of detecting a higher range of attacks like buffer overflow, file disclosure, information gathering, CSRF, etc.

7.3 Evaluation of semi-supervised machine learning algorithms

Another scenario certainly worth to be, and already is, examined is to apply data along semi-supervised algorithms meaning to treat an unsupervised algorithm as a supervised one. This can be achieved by first using a clustering model to identify itself the most relevant samples of our data set and then, by providing labelled data as input, train and increase its precision. This scenario could be really helpful in cases where multi-dimensional data exists and labeling is not possible or when the labeled data is quite limited.

8 References

1. https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data
2. https://ostec.blog/en/perimeter/firewall/
3. https://tools.ietf.org/html/rfc1918
4. https://www.rfc-editor.org/rfc/rfc1366.txt
5. https://www.fortinet.com/solutions/gartner-network-firewalls
6. https://www.juniper.net/uk/en/products-services/what-is/utm/
7. https://enterprise.verizon.com/en-gb/resources/reports/dbir/
8. https://owasp.org/www-project-top-ten/
9. https://www.gartner.com/doc/reprints?id=1-24F0FLTE&ct=201021&st=sb
10. https://www.marketsandmarkets.com/Market-Reports/artificial-intelligence-security-market-220634996.html
11. Sayali D. Jadhav, H. P. Channe, “Comparative Study of K-NN, Naive Bayes and Decision Tree Classification Techniques”, January 2016
12. Kristian Valentin, Michal Maly, “Network Firewall using Artificial Neural Networks “, Department of Applied Informatics Faculty of Mathematics, Physics and Informatics Comenius University, 2013
13. Mohammad Sayad Haghighi, Faezeh Farivar, Alireza Jolfaei, “A Machine Learning-based Approach to Build Zero False-Positive IPSs for Industrial IoT and CPS with a Case Study on Power Grids Security”, IEEE, 23 July 2020
14. Namit Gupta, Abakash Saikia, “Web Application Firewall”, Department of Computer Science and Engineering Indian Institute of Technology, Kanpur, 30 April 2007
15. Gustavo Betarte, Rodrigo Martinez, Alvaro Pardo, “Web Application Attacks Detection Using Machine Learning Techniques”, Universidad de la Republica, Uruguay, 2018
16. Raouf Boutaba, Mohammad A. Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano and Oscar M. Caicedo, Journal of Internet Services and Applications, 21 June 2018
17. Faizan Ahmad, “fWaf – Machine learning driven Web Application Firewall”, https://kdnuggets.com , GitHub repository link, February 2017
18. BBVA-Labs Security team, “WAF-Brain - the clever and efficient Firewall for the Web”, GitHub repository link , April 2018
19. https://owasp.org/www-project-top-ten/
20. Sara Althubiti, Xiaohong Yuan, Albert Esterline, “Analyzing HTTP requests for web intrusion detection”, DigitalCommons@Kennesaw State University, 2017

9 Bibliography

1. Sayali D. Jadhav, H. P. Channe, “Comparative Study of K-NN, Naive Bayes and Decision Tree Classification Techniques”, January 2016
2. Kristian Valentin, Michal Maly, “Network Firewall using Artificial Neural Networks “, Department of Applied Informatics Faculty of Mathematics, Physics and Informatics Comenius University, 2013
3. Mohammad Sayad Haghighi, Faezeh Farivar, Alireza Jolfaei, “A Machine Learning-based Approach to Build Zero False-Positive IPSs for Industrial IoT and CPS with a Case Study on Power Grids Security”, IEEE, 23 July 2020
4. Namit Gupta, Abakash Saikia, “Web Application Firewall”, Department of Computer Science and Engineering Indian Institute of Technology, Kanpur, 30 April 2007
5. Gustavo Betarte, Rodrigo Martinez, Alvaro Pardo, “Web Application Attacks Detection Using Machine Learning Techniques”, Universidad de la Republica, Uruguay, 2018
6. Raouf Boutaba, Mohammad A. Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano and Oscar M. Caicedo, Journal of Internet Services and Applications, 21 June 2018
7. Sara Althubiti, Xiaohong Yuan, Albert Esterline, “Analyzing HTTP requests for web intrusion detection”, DigitalCommons@Kennesaw State University, 2017
8. Binh Nguyen, “Network Security and Firewall”, Helsinki Metrpolia – University of Applied Sciences, 29 April 2016
9. Resul Daş, Abubakar Karabade, Gurkan Tuna, “Common Network Attack Types and Defense Mechanisms”, Department of Software Engineering, Technology Faculty Firat Univ. , 2015
10. Emmanuel Tsukerman, “Machine Learning for Cybersecurity Cookbook”, Birmingham, November 2019
11. Carmen Torrano-Gimenez, Alejandro Perez-Villegas, Gonzalo Alvarez, “A Self-learning Anomaly-BasedWeb Application Firewall”, Madrid, Spain
12. Adem Tekerek, Cemal Gemci, Omer Faruk Bay, “Development of a Hybrid Web Application Firewall to Prevent Web Based Attacks”, Gazi University Ankara, Turkey
13. Victor Clincy, Hossain Shahriar, “Web Application Firewall: Network Security Models and Configuration”, Department of Computer Science, Department of Information Technology - Kennesaw State University, 2018