IT Security Master Program
Department of Economic Informatics and Cybernetics
Faculty of Cybernetics, Statistics and Economic Informatics
Bucharest University of Economic Studies

Dissertation Thesis

Coordinator
Conf. univ. dr. Toma Andrei

Master Program Student
Santa David

Bucharest 2023

IT Security Master Program
Department of Economic Informatics and Cybernetics
Faculty of Cybernetics, Statistics and Economic Informatics
Bucharest University of Economic Studies

A Self Learning Anomaly Detection Based Web Application Firewall

Dissertation Thesis

Coordinator
Conf. univ. dr. Toma Andrei

Master Program Student
Santa David

Bucharest 2023

Statement regarding the originality of the content

I hereby declare that the results presented in this paper are entirely the result of my own creation unless reference is made to the results of other authors. I confirm that any material used from other sources (magazines, books, articles, and Internet sites) is clearly referenced in the paper and is indicated in the bibliographic reference list.

Abstract

Web application firewalls – WAFs for short – are a crucial component of contemporary web security. They are designed to protect websites from a multitude of attacks, ranging from exploits of the well-documented OWASP Top Ten Vulnerabilities to more sophisticated attacks targeting different architectures, design decisions or business flows. Traditional WAFs, as we have known for the past 30 years, rely on basic rule-based systems that we call signatures. Due to their high granularity and their stiffness signature-based WAFs can easily be circumvented by attackers with a wide range of techniques. To address this issue, this paper presents a modern solution – using machine-learning to develop an anomaly-detection based WAF.

The proposed system is designed to self-learn the normal patterns of traffic and then identify, alert and even block deviations from those patterns which we call anomalies. To achieve this, the system uses unsupervised machine learning to analyze the requests and create a model that scores each request with a probability of it being an attack. The benefit of unsupervised learning is that once the initial model is deployed in production, it can continue the process by constantly learning and adapting to new attack patterns, including the great benefit of protecting an application against zero-day attacks.

The machine-learning model is evaluated on a real-world dataset of both normal web traffic and attack-vector requests in order to prove its efficiency. The two metrics that matter the most in the evaluation are the detection rate and the false-positive rate.

Overall, the proposed system represents an advancement in web application security, building on top of the knowledge of decades of traditional security systems. Using a modern technology provides a more robust and effective defence against the ever-growing risk of cyber-attacks.

Keywords: Web Application Firewall × Web Security × OWASP × Cybersecurity × Next Generation Firewall × Deep Learning

Table of Contents

1. Introduction

It is impossible in today's age to find a business, no matter the size, that does not have an online presence. The rise of web sites and web applications in the past years, however, has also caused an ever-growing risk of cyber-attacks and security breaches [1]. Cyber criminals are working around the clock to find new ways of attacking anything and anyone to gain profit or notoriety. Such attacks range from gaining unauthorised access to the website to exfiltrate data to overloading the server with requests making it unable to deliver the service required and even encrypting the files with the intention to extort the victim into paying to regain access to their data [2].

Although research in web security is conducted, few papers are trying to tackle the problem by exhibiting a solution [3]. This paper, however, presents part of the solution, more precisely, an implementation of one of the many pieces to the puzzle we call cyber defence – a modern web application security system. Although traditional firewalls have been deployed in production since the 1980s [4], web application firewalls have entered the market in the late 1990s [5], machine learning powered web application firewalls are at the dawn of their existence, being brought to public attention just in the past decade.

The main purpose of this paper is to establish a baseline technical knowledge for future research and development in this area of ML-powered Web Application Firewalls, and even by extrapolation be adapted for other cybersecurity products such as SIEM, Antimalware, EDR/XDR, IDS, NIDS, IPS etc. In unison, this paper also documents the journey of actually implementing a machine-learning powered web application firewall, further named ML-WAF, from the ground up using open-source technologies in the .NET Framework.

The following chapters take the reader through a brief history and summary of the research conducted in the past regarding the junction of cybersecurity and machine learning, then present an overview of the components and the structural design and architecture of those components diving into the deeply technical subjects. Towards the end, the thesis discusses the results obtained in the study conducted while writing this paper and ends with a glance into the near future of this research topic.

The motivation behind this subject came when I was working as a SOC (Security Operations Center) Analyst. More accurately I was an L1 (Level 1) SOC Analyst, which meant my main objective was to triage from thousands of alerts generated by multiple cybersecurity detection solutions only the ones that were true-positives and an actual threat to the security of the company I was defending. From all the around 20 IDS (Intrusion Detection System) solutions I was monitoring at that point in time, the one that I enjoyed the most as well as the most efficient one was Darktrace. Darktrace is a tool that employs mathematical methodologies to identify security breaches [6]. The reason I liked Darktrace the most was that it did not use the traditional static rules system, on the contrary, it was working on the behavioural level which I found a far smarter and modern way of tackling cybersecurity.

2. State of the art

The purpose of this chapter is to provide a level playing field by bringing readers who have never delved into subjects as technical as cybersecurity or machine learning to a level of context good enough to read the following chapters without needing to search terms or concepts on google every couple rows. The following sub-chapters will display a brief crash-course into the context of cybersecurity and machine learning on the high-level non-technical side. Experienced technical individuals are also advised to walk through the chapter for a quick refresher.

2.1. Cyber-attacks landscape

The increase in cybercrime is undeniable. According to an article published by AAG – an IT company which regularly centralizes cybercrime statistics – the numbers are astonishing. Malware attacks increased by 358% compared to 2019, a colossal increase that can be explained by placing it in the context of the pandemic which took by surprise the network and IT sides of businesses just as much as any other department. The need to work remotely forced companies to open themselves to more risk and those that had no prior experience with VPNs or did not have the personnel skilled to ensure the security of such a vital component suffered great losses, both reputational and financial. Other striking statistics are more than 50% of US citizens were affected by cyber-attacks in the beginning half of 2022, 39% of UK companies reported suffering at least one attack in 2022, and one of the most dangerous and feared attacks – the ransomware attack – reached an aggregate of 236.1 million reports [7].

The European Union Agency for Cybersecurity (ENISA) issues an annual report named ENISA Threat Landscape (ETL) on the state of the cybersecurity threat landscape. The key conclusions from the 2022 report [8] are: 60% of organisations hit by a ransomware attack may have paid the ransom demands; 66 zero-day vulnerabilities were observed in 2021; phishing remains one of the most popular techniques used in cyber-attacks; the largest ever DDoS (Distributed Denial of Service) attack was recorded in Europe in July of 2022.

2.2. Cybersecurity Research

Although those figures spark fear in the mind and heart of every cybersecurity professional, should also motivate them alongside researchers to invest more and more time in understanding the trend and landscape of cyber-attacks and cyber defence. A research published by IEEE in 2019 showed that between 2015 and 2019, the community has collectively published 201 papers that had the words “cybersecurity” or “cyber security” in the title [9]. Considering that a substantial amount of papers that tackle cybersecurity topics do not even contain those keywords in the title, I decided to check ResearchGate, a very popular website and social network that indexes published papers. ResearchGate returns 11.382 publications composed of articles, preprints, conference papers and literature reviews, however checking the “Cyber

Security" topic reveals 106.346 publications [10]. Despite the fact that it can be a very challenging task to accurately compute the amount of papers addressing cybersecurity or adjacent subjects, it is not as difficult to observe that the interest of academia in this matter is at an all-time peak and still growing.

2.3. Web Defence Mechanisms

The wide variety of cyber-attacks is counter-attacked by a diversity of defence solutions, tools, and mechanisms [11]. They play distinct roles in different layers that when working together compose the concept of "Defence in Depth" – also known as the Castle Approach [12]. The most frequent roles a security solution ensures are detection, prevention, reconnaissance, management, proxy, hardening, patching etc. The strategy behind this concept is that when one, or even multiple layers, fail, the others serve as a back-up, greatly increasing the chances of the attack to be at least mitigated and in the best-case scenario, altogether stopped. The Web however is inherently a predominant layer 7 protocol, as such this paper will only focus on the application layer security, starting from the assumption that every layer below it is properly configured and protected.

With this assumption in mind, a 2012 paper by Yousaf, Iftikhar, Javed and Tahir states that all web applications are divided into three inter-related phases. Those phases are: 1) Authentication phase, 2) Session Management phase and 3) Data Access-Control phase [13]. I, however, do not think this model is optimal, because, even though it encompasses most of the attacks, it is impossible to map some attacks or vulnerabilities to this model. Such incompatible vulnerabilities are: Cryptographic Failures which are independent of phases or Security Misconfigurations, which happen way before the first phase is initialised, such vulnerabilities being prone to an attack without even accessing the application. One such attack is a man-in-the-middle (MITM) attack exploiting a missing HSTS header. An attacker can intercept a legitimate request and redirect it to a cloned website acting as a credential harvester. As it can be seen from this example, the original application did not even get a chance to apply the first defence mechanisms which would have been mapped in the Authentication Phase. The two incompatible vulnerabilities were not randomly chosen, as they are numbers two, and five respectively in the OWASP Top 10 [14].

2.4. Introduction to Web Application Firewalls

Dr. A Shaji George and A.S.Hovan George define a WAF in their study as "a dedicated firewall intended for filtering and controlling HTTP traffic through internet traffic between the web clients as well as application servers" **[15]**. WAFs operate at the OSI Layer 7, the top-most layer, offering application-level security using packet-inspection. In order for it to be able to prevent attacks, such a solution must be implemented in-line, meaning it must also serve the

function of a proxy that is able to forward the packets to the destination, or drop them if they are detected as malicious.

The main purpose of the WAF is to safeguard a web application from web requests that aim to exploit web vulnerabilities, however in recent times, those firewalls have evolved to offer multiple functions such as load balancing, reverse proxy, SSL inspection, DDoS protection and autonomous intrusion prevention. Considering all the functionalities previously mentioned, the benefits of using a Web Application Firewall in front of publicly exposed web applications are clear: they offer an organization security by ensuring the CIA triad (Confidentiality, Integrity, and Availability), reliability, compliance by fulfilling the necessities required by industry regulations and standards, risk reduction, improved performance and real-time monitoring and reporting. Overall, Web Application Firewalls have become a necessity for every company that has public websites, or web applications served over the internet, and their use will increase as long as the internet becomes more complex, and attackers become more sophisticated in their methods of targeting the web.

2.5. Machine Learning Research

Machine Learning (ML) is a branch of Artificial Intelligence (AI) that focuses on the development of algorithms and models that enable computers to learn from labeled or unlabeled data called training data without being explicitly programmed to do a particular task or to obtain a desired result [16]. In recent years, machine learning has become a valuable tool in various domains, including computer vision, natural language processing, speech recognition, and nonetheless cybersecurity [17]. ML algorithms are usually mainly grouped into three broad categories: supervised learning, unsupervised learning, and reinforcement learning.

In supervised learning, the algorithm is trained on a labelled dataset, where each data point is associated with a target output value. The goal is to learn a function that maps input data to the correct output values. Common supervised learning algorithms include linear regression, logistic regression, decision trees, and neural networks [18]. For the scenario proposed in this paper, this technique of learning without supervision is the desired one, as a normal web application can have up to hundreds of requests per second. If this number sounds high to you, think about all the bots and the crawlers scattering towards all the edges of the World Wide Web.

In the context of cybersecurity, machine learning can be used to detect and prevent several types of attacks, including malware, network intrusions, and web-based attacks. For example, machine learning algorithms can be used to analyse network traffic and identify anomalous behaviour that may indicate a cyberattack [19].

Overall, machine learning is a powerful tool in the field of cybersecurity, having the potential to enhance the accuracy and efficiency of various cyber-security activities. However, there are also challenges and limitations associated with the use of machine learning in security, including the need for enormous amounts of labelled data, the risk of adversarial attacks, and the potential for bias in the algorithms. Addressing these challenges will be a key area of future research.
One particular application of machine learning in cybersecurity is the development of machine learning-based web application firewalls (ML-WAFs). These systems use machine learning algorithms to analyse web traffic and detect anomalous behaviour that may indicate a web-based attack, such as SQL injection or cross-site scripting (XSS). ML-WAFs have been shown to be effective in detecting and preventing attacks, including zero-day attacks that are not detected by traditional signature-based approaches. Such a particular implementation of machine learning will be the central topic of this paper.

2.6. Self-Learning and Anomaly Detection in Web Application Security

Since a rule-based approach is no longer able to keep the pace with the unlimited ingenuity of attackers, the blue-teams have started applying machine learning techniques in order to fight the attacks of the black-hat hackers. Rule-based defences are obsolete because they are easily bypassed and circumvented with low-skilled obfuscation techniques such as: case toggling, encoding, junk characters, wildcards, line breaks, token bearers and many more. A rule-based tactic is losing the fight because it is reactive by-design, you need to know what the payload looks like to create a rule that detects it. This opens the door for the obfuscation techniques mentioned above and to a much bigger threat that the community calls “zero-day attacks”. Zero-day attacks are attacks that are not detected by any defence solution because it was until that point not learned by the developer. In other words, if the attacker discovers a new vector attack that did not cross the minds of the security and development teams, there is no rule created to detect that threat and it will easily go past any defence hops put in place.

By introducing machine learning and anomaly detection techniques in our security products, we change the whole paradigm from detecting what is known to be malicious to detecting everything that is outside the normal behaviour. Having in mind this new paradigm the reason machine learning powered security solutions combat zero-day attacks considerably more effective is straight-forward: a zero-day attack is by definition an outlier to the expected conduct. In the research conducted for this paper, I have found the work presented in **[20]** to be a great resource, centralizing the research made in the Deep Learning Network Traffic Monitoring and Analysis area. Also, the article in **[21]** is a great resource for understanding how Neural Networks can be trained to detect web attacks embedded in HTTP requests. It also uses the CSIC 2010 Dataset that will also be a part of the learning data used by this paper.

3. Architecture

The following sub-chapters will display multiple specialized diagrams that will also be talked on to provide technical details into the solution's architectural design.

3.1. Overview

Figure 1 shows the flow of the implementation on a high-level together with the main components built in the system architecture.

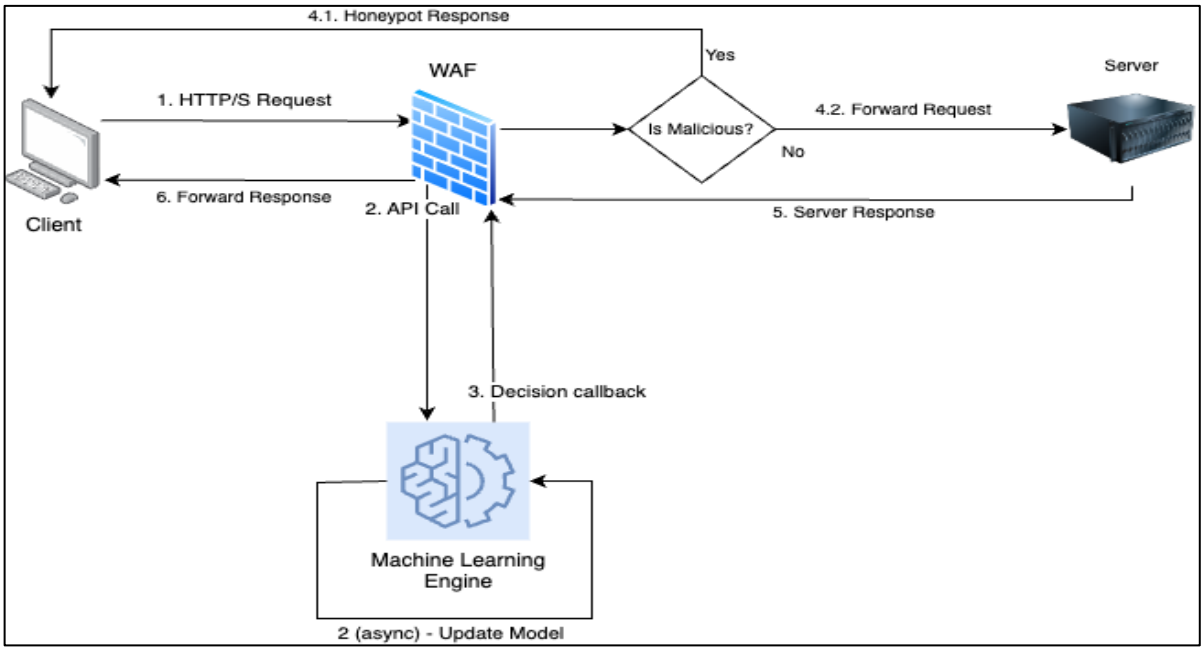

Figure 1 - Overview Diagram

First, we can establish the fact that there are two decoupled systems that communicate over an API, the WAF component and the ML component. The reason for which those two are decoupled is because they can work without each other and they serve two different purposes, however, it is decisive that the two cooperate to serve together as a Machine Learning powered Web Application Firewall (ML-WAF). The WAF component acts as the central piece in the architecture providing all the necessary services for the Client-Server communication to be both functional and secured. The Machine Learning Engine on the other part is totally decoupled from the Client-Server communication, playing however the crucial role of deciding if the web request initiated by the client will get to the server or not. This component is exactly the particularity separating an ML-WAF from a traditional WAF, meaning if the Machine Learning Engine fails to provide a decision (block or allow the traffic), the WAF component only functions as a proxy, or a reverse proxy and the only security features provided being that it hides the true IP Address of the web-server and it enforces encryption.

The second key point that stands out from Figure 1 is the fact that step 4 is split depending on if the MLE (Machine Learning Engine). If the request is not considered malicious, the process flows as if there was nothing in-between the client and the server, however, there is a design choice for the case in which the request is detected to be anomalous. The design choice to be made is what the WAF should return in this case. One path that can be taken is simply dropping the whole connection, showing the user an error screen or a warning, this however comes at the consequence that an attacker can learn about the technology behind the security solution he is fighting. Although, of course, security by obscurity is not a good concept, neither is being too transparent. Following this stream of thought, a new concept is being explored in the cybersecurity industry which is called "deception". The purpose of deception is to fight-back the attacker, attempting to make it harder for enemies to circumvent security solutions and controls put in place by the defenders. Such technologies deceive malevolent users by setting traps and decoys to imitate genuine assets or behaviours. The main advantage of using such techniques is logging and monitoring the attack vectors utilized throughout the engagement to learn more about the indicators of compromise, the TTPs (Tactics, Techniques and Procedures) and the novelties of the black-hats and improve defences. Other important advantages are reducing alert fatigue, increasing the chances of catching an attacker after the initial compromise and access to the network and increasing the time window for incident response actions such as isolation, quarantine etc. Going back to the WAF in question, the path chosen for the case of an anomalous request is to throw in a deceptive response. That can be configured to either have cached deceptive responses that are sent back to the client, or it can be arranged to redirect the communication to a honeypot. A honeypot is another cybersecurity deception technology represented by an application that is designed to look like a vulnerable system or service in production. The catch of course is that this honeypot is isolated from the real production environment, and it only stores bogus data.

In the following chapter, Chapter 3 – Implementation, the steps from Figure 1 will be individually dissected and analysed to provide more context and technical details regarding the complete process. This first figure is thought and designed as a business-oriented diagram to help non-technical people understand the structure and purpose of the solution.

3.2. Technical Architecture

Figure 2 represents an activity diagram for the normal flow - the one where the request is determined to be legitimate; it is built on top of the overview diagram but explaining more in-detail the process. The observation that stands out the most in the activity diagram is the async box on the 2.2 step. It can be remarked that the async steps happen between the evaluation score being returned to the WAF and the request being proxied to the target server. This means that the machine learning model being updated and retrained is decoupled from the system. The decoupled-ness property can also be perceived directly from the diagram by noticing how step 2.2.4 ends the flow that does not rejoin the main flow that goes all the way back to the User Browser component ending with step 4.

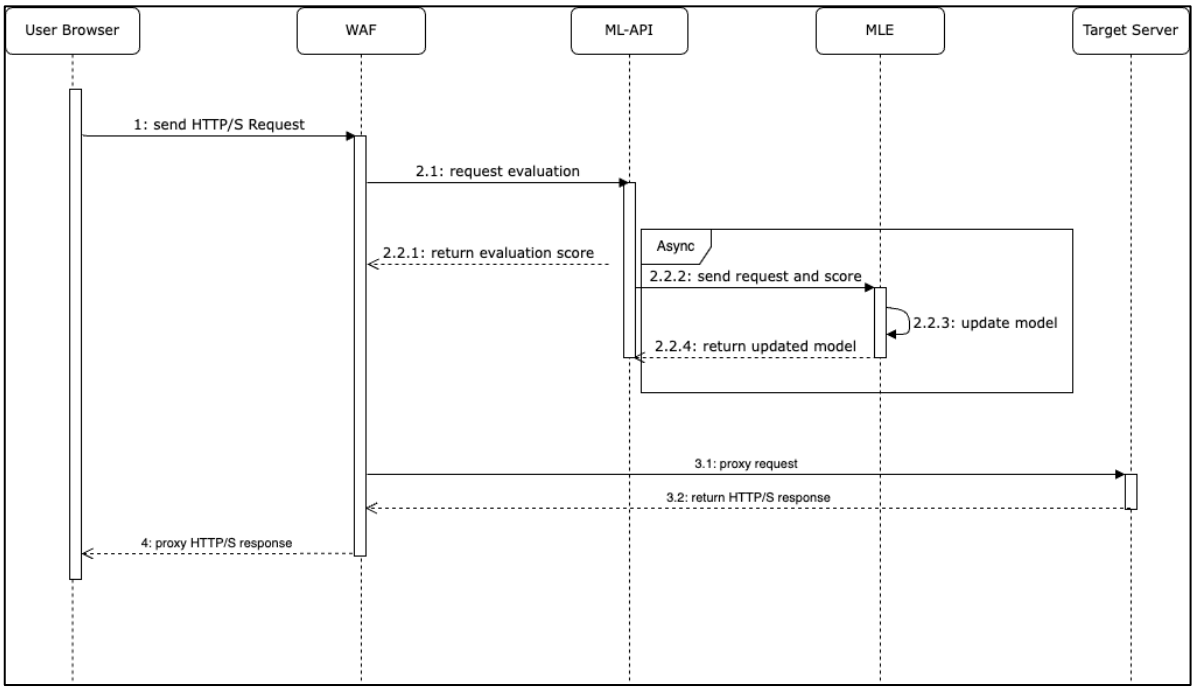

Figure 2 - Activity Diagram

Figure 3 signifies a Network Architecture Diagram for an enterprise implementation scenario. It depicts two clients connecting to two target servers hosting two web applications resulting from the TCP 80 / TCP 443 requests shown. Between the clients and the enterprise network are a Border Router (OSI Layer 3) and one Core Switch (OSI Layer 2). It is worth nothing the fact that between or around those elements may be other intermediary hops such as security solutions like border firewalls, NIDS/NIPS solutions or other network-related components such as additional switches, routers, VPN servers, AD Domain Controllers etc. The purpose of this diagram is not to show a complete network diagram, but to suggest an overview of the infrastructure needed to employ such a solution and how it fits in the already existing scenario of the network.

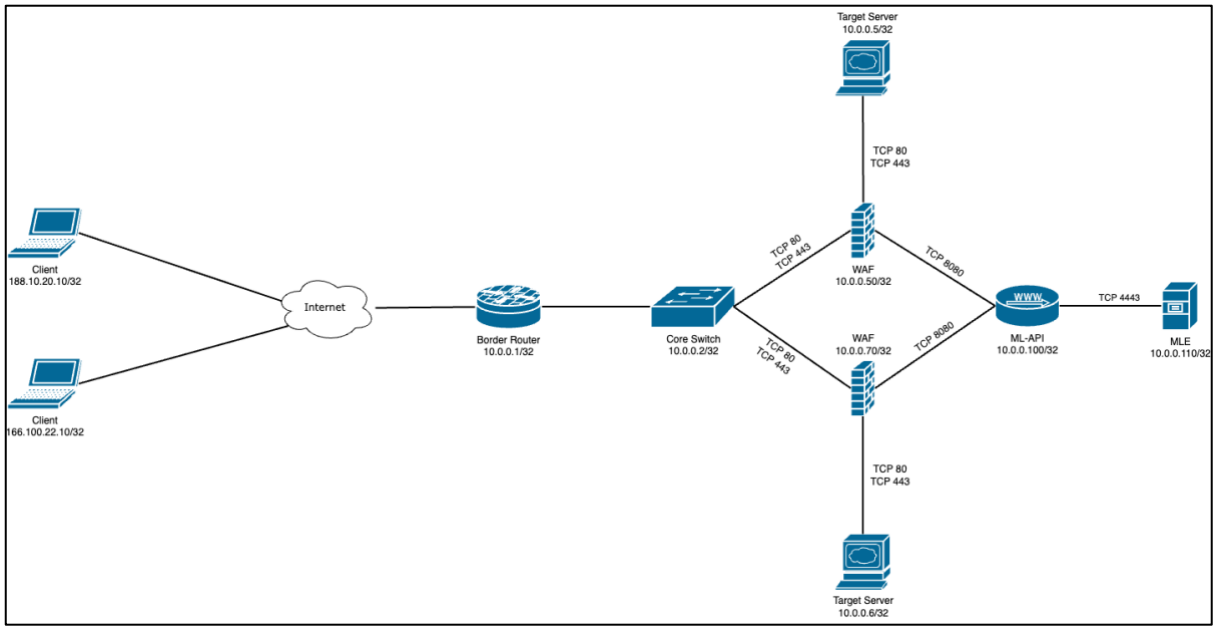

Figure 3 - Network Diagram

The more interesting discussion points arise from noticing the two distinct WAF instances deployed in the diagram. This is the best practice as having different hardware for each instance brings huge performance benefits, however it also means increasing the capital and the operational costs. Of course, those distinct WAFs do not necessarily need to be on different hardware, as using virtualization solutions the two different IP addresses might not mean different hardware, but different virtual machine or docker container instances. In the worst case two WAF applications could run on the same hardware, without virtualization even, but it would need to have different destination ports configured at the application level, that also match the destination ports of the targeted web server.

Another important observation is that the ML-API and the MLE are only hosted on one hardware each, no matter the number of applications thus WAF instances. That is because the connection to those two components is done asynchronously and in a decoupled manner, so the performance and latency does not affect the flow of the client-server communication. One justification that must be clarified is the need for only one MLE even though there are multiple applications with multiple use-cases and expected behaviors. The reasoning behind the only one MLE instance is that the distinction between the applications is made at the model level, not the MLE. The Machine Learning Engine is only concerned with data processing, so it can handle multiple applications if it is told to which model, i.e., which application to update aka re-train.

3.3. Used technologies

To get the best of a performance throughout the system I had to choose a technology stack that offered great coordination between the components and sub-systems. The optimal solution that I found was to use the .NET Framework to build every component, meaning, the WAF, the

ML-API and the MLE. Even though individually they are not the most efficient or well-performing, for example a far more efficient in regard to time and memory TCP proxy could have been built using plain C or C++ rather than C# and the .NET framework [22]. However, the big plus of the .NET choice is that it presents smooth communication between all components as they are built using the same programming language, framework, and common libraries. This saves a lot of time in development, post-development debugging and future component additions. For example, a web-based or even native application to act as a management console, or even a UDP equivalent proxy could be added seamlessly using the tools provided by the .NET Framework.

To be more precise, the technologies within .NET used to develop the components are as follows: for the WAF, the System. Net namespace libraries [23] were extensively used to write a TCP proxy using sockets and to provide an HTTP middleware. The ML-API was developed using ASP.NET Core [24] and the MLE is based on the ML.NET [25].

As will be mentioned in the following sub-chapter ELK was also used to provide a presentation layer for marketing or technical PoC reasons.

3.4. Additional components

This sub-chapter's purpose is to approach the architecture of the additional components developed around the solution. The reason behind those components being named “additional” and being approached separately is that they are not in the scope of the ML-WAF solution, instead they are used as a control-center to visualize the data, store it to be data-retention audit approved etc.

The additional components are known as the Elastic Stack or ELK for short. According to Elastic, the developers, ELK is “a fast and highly scalable set of components – Elasticsearch, Kibana, Beats, Logstash, and others - that together enable you to securely take data from any source, in any format, and then search, analyze, and visualize it.” [26]. Elastic, this way, provides a full stack for creating a control center dashboard style web app. This is a fantastic opportunity for this paper’s use-case as this web-app is outside the scope of the study, but it offers a wonderful way to present the solution in a PoC (Proof of Concept) way. The data from the ML-WAF is imported in the Elastic Search ingest pipelines, indexed, and analyzed by the Elasticsearch and Logstash components and then presented in a web application by Kibana. Figure 4 presents the architecture flow and the role of each sub-component.

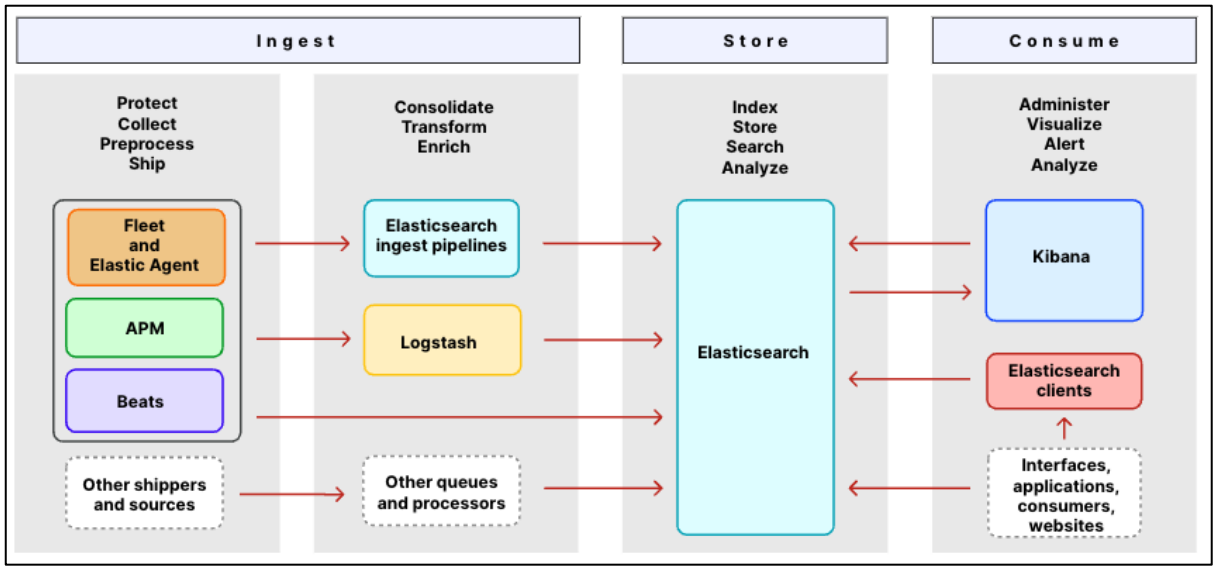

Figure 4 - Components of the Elastic Stack [26]

Elasticsearch is a NoSQL database, meaning it indexes and stores data in an unstructured way. More accurately the data is stored in the form of JSON strings. We will not go into the details of Elasticsearch's more advanced concepts such as indexes, mappings, shards, replicas, and nodes. The crucial point to be made is that the data is not stored in a SQL manner, so that SQL Queries will not work. Elastic provides a query DSL (Domain Specific Language) to be used in queries and data analysis on the JSON raw data stored. The ingestion of the data from the ML-WAF to the Elasticsearch database is made via the CSV import functionality. Elastic offers a File Data Visualizer feature that allows the uploading of different file formats and the transformation of that data into Elasticsearch compatible formats. Logstash and other processors can also optionally be used to enrich the data.

HTTP Request	
PK	_id
	(string) client_ip
	(string) request_method
	(string) request_line
	(string) request_headers
	(string) request_body
	(float) probability_score
	(int) status_code
	(bool) was_blocked

Figure 5 - HTTP Request Elasticsearch Document Schema

Figure 5 is an entity diagram for the only Elasticsearch document used. A document in Elasticsearch is the equivalent of a table in SQL databases.

The _id field is marked as PK (Primary Key) and is automatically assigned by Elasticsearch for each entry to uniquely identify the documents. This field is not a proper primary key, as we are not dealing with a SQL database so that is not a concept applicable in this context, however it does serve the role of uniquely identifying an entry and can be used in queries. The rest of the fields are related to the HTTP Request or adjacent information such as client IP, probability score assigned by the machine learning component and the status code returned by the server. The “was_blocked” field is necessary as the threshold can vary from appliance to appliance, so the field is not redundant.

4. Implementation

In the following subsections the steps from Figure 1 will be individually dissected and analysed in order to provide more context and details about the complete process of the development.

4.1. HTTP/S Request sent to WAF

The first step of the process of course is sending the client web request to the WAF application. The low-level networking details will not be covered in this paper as they are not relevant to the subject. For example, in the case of HTTP/S requests we will skip the TCP 3-way handshake process and all the other OSI layer 4 specific functionalities. We will concentrate our efforts on the OSI Layer 7 – the Application layer.

To go more in-depth with the technical details of the TCP proxy side of the WAF, a System.Net.Sockets.TcpListener was used to listen and accept incoming connections from the internet. The TcpListener can accept multiple async connections so it is scalable, and it can be multithreaded to improve performance. For every connection accepted by the TcpListener, a System.Net.HttpClient is instantiated for the WAF to proxy the requests to the target server. A separate package-level HttpClient is instantiated in the Main method, so at the boot of the WAF application that runs as the MLE API Caller. This client is responsible for sending API requests to the machine learning API to obtain the probability score for the said request.

The first problem the WAF might encounter arises as soon as the very first packet arrives. In the case of an HTTP request there are no problems, however if the request was sent using the HTTPS protocol, it means that the data is encrypted using TLS or SSL. It is obvious that the WAF cannot just proxy the encrypted packets to the server or try to send it to the MLE in its encrypted form because it would not be able to truly analyse the request and accurately decide regarding the legitimacy of the request. Considering this problem, the first functionality the WAF must implement is called “SSL Inspection”.

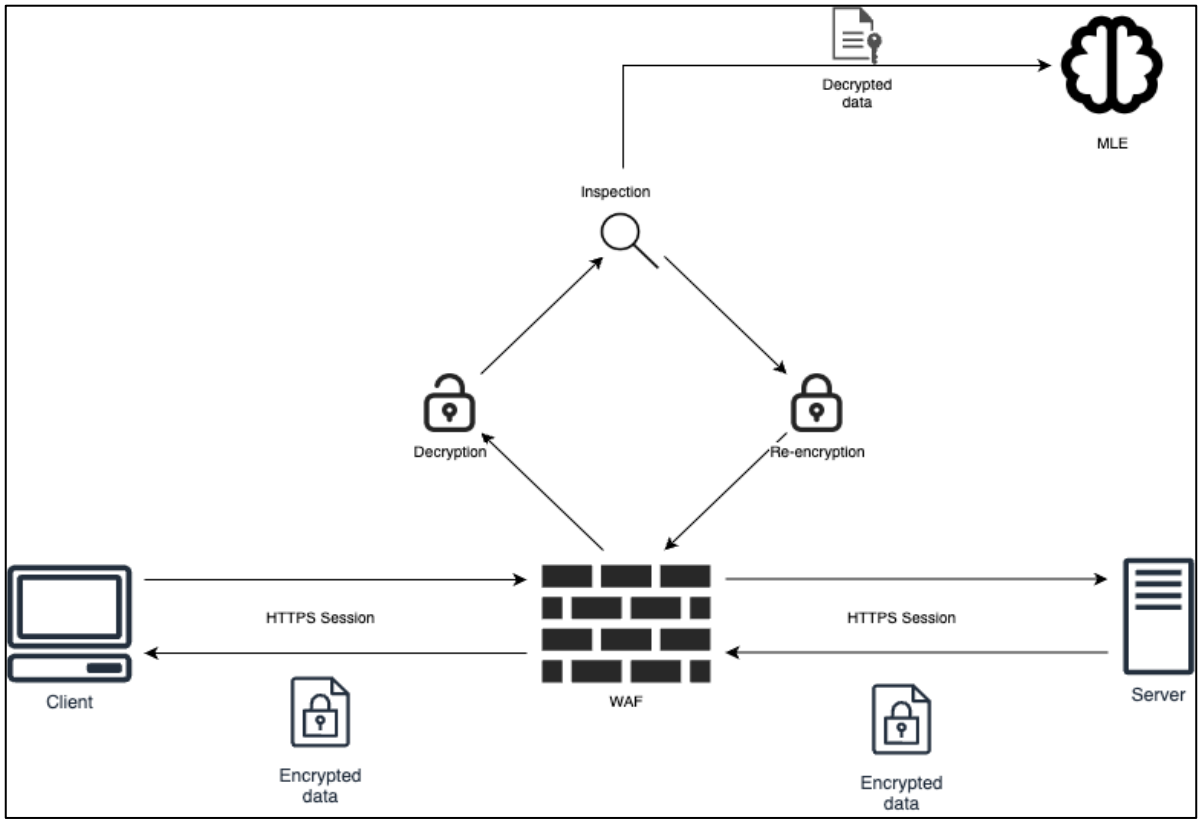

Figure 6 - SSL Inspection

SSL Inspection as it can be seen in Figure 6 is a cybersecurity technology that allows solutions to examine traffic encrypted using SSL/TLS protocols. Attackers can use the encryption provided by those protocols to either hide malicious ingress actions such as malware downloads, command and control communications or egress traffic such as data exfiltration or lateral movement payloads. The SSL inspection process involves intercepting the encrypted traffic, decrypting it, analysing the clear-text data, and then re-encrypting the data for it to be sent securely to the destination. The content inspection can take many forms including content filtering, data loss prevention, malware detection, compliance enforcements etc. This technology, even though it provides a very-much needed security level, it also raises some privacy and legal concerns involving GDPR, CCPA or other applicable laws and regulations **[27]**.

The SSL inspection is possible because the connection is not directly between the client and the server, on the contrary, there is no connection between the two machines. This client-server connection is simulated by the proxy component of the WAF. There are, in fact, two separate TCP connections, one between the client and the WAF and another one between the WAF and the server. There is however a method to implement SSL inspection in the configuration where the TCP connection is between the client and the server directly in which the WAF only relays the packets. The way to implement SSL inspection in this scenario is by having a known server key, meaning the WAF has a copy of the private key and the server certificates. This

implementation however creates complications in the transport, storage and securing of those secrets.

The SSL/TLS session keys used to decrypt and reencrypt HTTP/S can be obtained on Windows/Linux/Mac by using the SSLKEYLOGFILE environmental variable. The SSLKEYLOGFILE is a text file generated by web browsers and web servers where session keys are logged for future decryption in network traffic or security analysis. A step-by-step procedure for logging those session keys on each of the platforms can be found on this website [28]. In order to provide improved security those session keys will not be stored more than strictly needed. The keys will be deleted from the WAF after the session is terminated as they are no longer needed considering a new session will not re-use the previous key.

4.2. API call to the Machine Learning Engine

The decrypted request must now be sent to the core function of the ML-WAF, which is the anomaly detection done by the Machine Learning Model. Of course, machine learning models cannot understand HTTP/S requests as they are either words or binary representations which means that the data must be transformed into a format that a machine learning engine can understand and process. This transformation will be covered and explained in the following sub-chapter where the whole machine learning engine will be discussed, the only detail relevant to this step is that the payload will be transformed into a vector [29] that will be sent to the MLE.

One point worth mentioning is that the WAF and the MLE do not necessarily need to run on the same machine or appliance. Depending on the load of the target web application, the network it is installed on, the budget and many other factors a design choice must be made when configuring such a solution. The WAF and MLE can run on the same machine and communicate on the loopback interface using different ports, which reduces latency and networking issues, but it also increases the CPU and memory loads significantly. The pros and cons are reversed if opting to run the two components on separate machines.

One might notice on the Overview diagram showed in Figure 1 that step two has also an async action which happens inside the MLE component. This async activity refers to adding the knowledge from the request that is sent for analysis to the machine learning model in a feedback loop manner. This ensures continual learning to keep the WAF up to date with the newest attacks but also for it to always have an accurate view of what the normal or expected behaviour is. As web applications are constantly being updated new functionalities are being released or existing ones are modified for better user experience or performance. This constant change must be mirrored in the WAF, so this persistent re-learning is crucial for the solution on the long-term.

The format of the API call is an HTTP/S request to the following URI format:

https://MLE_IP:MLE_PORT/predict/model_id

The model_id is an integer value uniquely identifying the machine learning model that is being used for the particular application being defended. The response of the API is a simple HTTP Response with status code 200 with the body being a single float number encoded in ASCII representing the probability of the request that was send for prediction to be malicious.

4.3. Decision callback

This section will skip the actual work of the MLE as it will be discussed in the machine learning chapter. For now, we will consider the MLE as a black-box that generates yes or no – anomalous or normal – responses to our API call that contains the initial app client request.

The anomaly detection machine learning model is a binary or continuous score that indicates the level of anomalousness for the given input sample. A threshold can be decided upon depending on the requirements, needs or preferences of the implementer. This threshold refers to the value of the anomalousness beyond which a request will be classified as malicious.

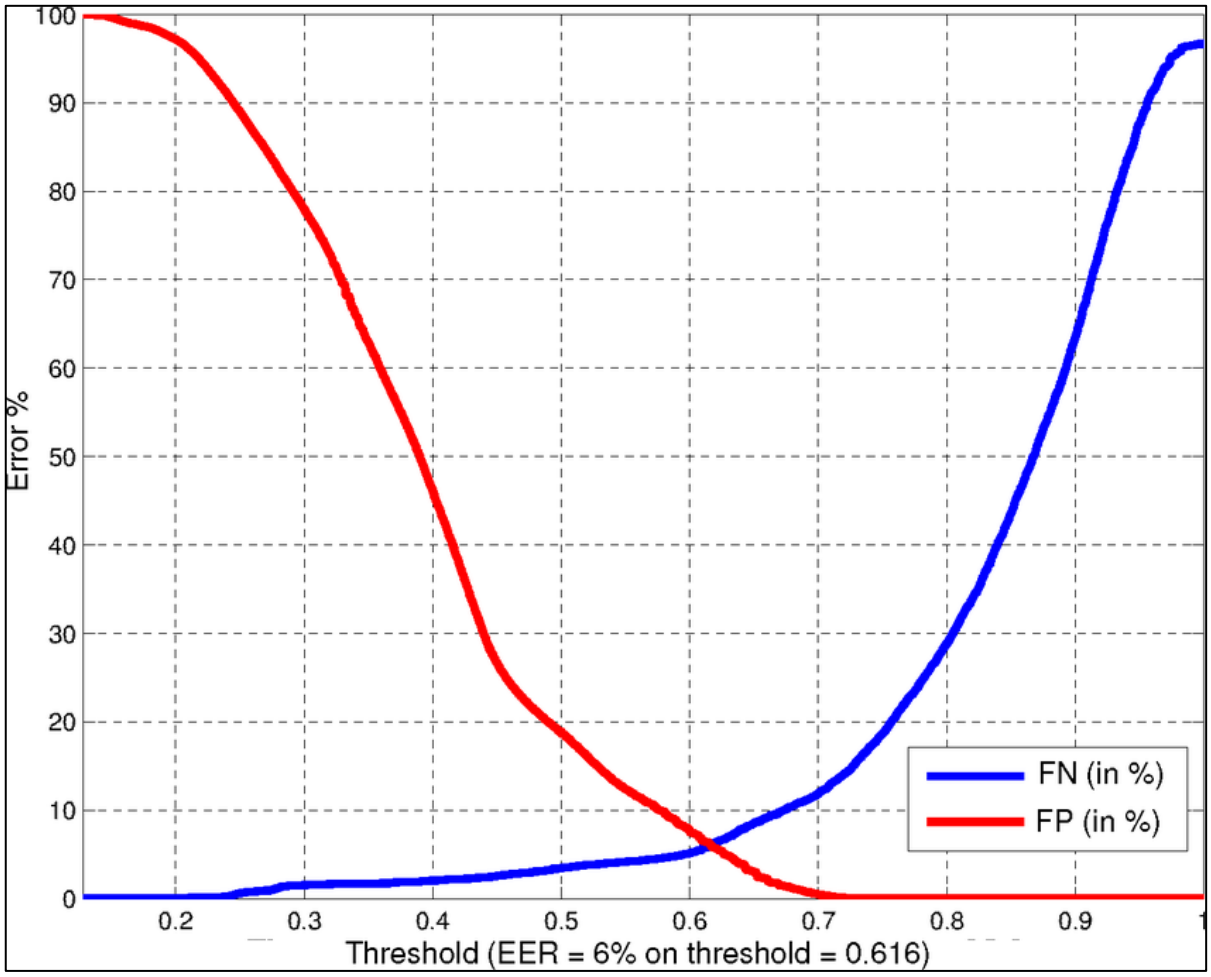

Figure 7 - FPR and FNR

This implementation decision can be understood by studying a graph such as the one seen in Figure 7 **[30]**. In short, the higher the value of the threshold, the more outlying the request must be from the expected pattern. A high threshold results in more False Negatives (FN) – malicious requests that the WAF will let through which increases the risk of a security breach. A low threshold on the other hand will impact production and the end-user experience as any slight, minuscule change from the usual request will be blocked. We call those legitimate non-malicious requests that are blocked False Positives (FP). Finding the balance between FP and FN is an exceedingly challenging task especially as there is no correct value for the threshold. A risk assessment must be conducted to apply the threshold value taking into consideration many factors. A banking application will most of the times have a much lower threshold than a web-based game as banks are often more regulated and have a much lower risk appetite not wanting to endanger money or reputation.

To clarify this even further, for the use-case in Fig. 5 the most effective threshold value mathematically calculated is 0.616 (61.6%). This threshold value will on average produce an error of 6% or one error in every 17 decisions. Even though this is the best value for the threshold to achieve balance and minimize the error rate, this might not be the best value to choose. Following the online banking application scenario, a one-in-seventeen chance of being hacked will probably be considered too risky and so a higher value for the threshold will be chosen. For example, for a threshold of 0.7 the false-positive rate will be close to 0, however the false-negative rate will also grow to 12%.

The actual threshold value is set in the WAF component not at the machine learning engine level; thus, the response is the probability. This design choice is made to provide more flexibility and opportunity to develop around in adjacent components. The other side of this coin would have been simply returning a binary or boolean value, such as block or do not block aka. Malicious or non-malicious. The additional advantages of returning the probability instead of the result of the prediction is that other components can use this value. Examples of usages are statistical analysis – a simple boolean result would not have provided much insight into how dangerous the request is. We can agree that there is a difference between a request getting the score of 98% malicious versus a request that is considered 51% malicious. In a scenario where the threshold is set to 50%, they would both be considered equally dangerous. Another advantage is that the probability could be used in human analysis triage. Let us assume that every morning a human security analyst looks through all the requests blocked by the WAF over the course of a set period of time. This action could be done for threat hunting, OSINT publishing or simply for reporting purposes. A human will need to triage and prioritise those events in a scenario where there are too many to require attention. This prioritization could prove quite challenging when not having available a numerical value to sort upon. The probability being made “public” throughout the components gives such a scenario a field to be used by analysts in a tier-1 operations centre. The probability score can also be used in other security solutions in the network such as a SIEM or an EDR that is running on the same machine as the WAF or even on the target server.

4.4. Honeypot response

If the callback from the MLE resolves as malicious than following the flow described in Figure 1 the WAF will send a deception response back to the initial client. Deception response is a strategy used in cybersecurity to actively mislead and deceive detected attackers. As it was previously mentioned there are multiple ways to assemble such a response however the details are not relevant to the subject at hand. One common implementation that is also facile to our implementation is the integration with a honeypot. The implementation is as simple as changing the destination IP from the real web server to the IP of the honeypot web server. It is recommended that the honeypot emulates the real application as well as possible for optimised results. The best way to implement a honeypot is to share the same frontend and backend, but to be connected to a separate database with bogus data and isolated in a separate VLAN to mitigate the risks of lateral movement.

The advantages of using such a strategy include: the longer the WAF can mislead the attacker into believing they are still attacking a real target the more telemetry it can gather which can be used to further reinforce its defence knowledge. The best scenario, for example, is catching a zero-day attack being used against a honeypot as it can be used to write rules for traditional defence solutions without suffering any damages. The concept of zero-day attacks was explained in sub-chapter 1.2.6. OBJ Self-Learning and Anomaly Detection in Web Application Security.

Moreover, all the resources the attacker spends on striking a fake application are resources that are not used on the real threat. In addition to those great benefits the deception also comes in to help the operational component of a company, providing more time for the security team to respond and contain the possible incident.

This honeypot idea can be levelled-up even further by expanding to the honeynet concept. A honeynet is simply put an isolated part of the network containing one or more honeypots. A honeynet however is much more expensive in terms of time to set-up and maintain and in terms of resources needed. Usually, honeynets are only used in very cybersecurity mature organizations that have already deployed all the other simpler cybersecurity solutions and services such as SIEMs, EDRs, network sniffers and a fully functioning SOC.

4.5. Sending the response back to the client

The last two steps will be covered in this one subchapter as it is only technically split into two separate flows because the response must be hopped through the WAF on its way back to the client. This hop however can be a good thing in the end as the WAF can use its proxy function to do proxy caching. Proxy caching is a mechanism in which a proxy server stores copies of content previously requested by other clients and responds on behalf of the origin server without consulting the server. This saves CPU cycles, memory and disk I/O operations on the origin server and bandwidth and latency at the network level. The connection ends with the

WAF sending the response, either cached or obtained from the origin server to the client that initiated the request.

One important implementation detail that needs to be documented is the method used to be able to return the response to the correct client. The WAF operates as a reverse proxy, thus it hides the real server IP, but that also means that the web server "sees" the client as being the WAF even though the real client is behind that WAF. To be more explicit, if the web server only relies on the TCP data sent by the WAF, the response stops at the WAF and the WAF will not have stored the real client – which now should be the destination of the response. To solve this problem, the proxy inserts a new header into the HTTP/S headers. This header is called X-Forwarded-For (XFF), and it stores the originating IP address of the client connecting to the web server through the proxy. It is also particularly important to note that any intermediary device that processes the request must append its own IP address to this header. Additional such devices might include NIPS (Network Intrusion Prevention Systems), normal internal proxies, etc.

Another relevant security point is that the header can be easily spoofed. This is a well-known problem in the industry, but it cannot be mitigated by the WAF. The vulnerability appears when the network is not properly defended, and a man-in-the-middle attack might be happening unrelated to the WAF flow.

4.6. Machine Learning Engine

The technology behind the Machine Learning Engine is the ML.NET, an open source, cross-platform machine learning framework built on top of the .NET backbone that enables developers to easily build machine learning models using existing C# and .NET skills. To simulate the initial "learning stage" period, a TSV (Tab-Separated Values) file containing more than 38000 requests manually labelled in a binary-classifier manner (0 for not malicious, 1 for malicious) was used to train the initial model. Additionally, the WAF can be left running while browsing through the application simulating normal behaviours can be used to further help the machine learning algorithm create the baseline model that then can be deployed in production.

As it was previously touched in the earlier sub-chapters, one of the most important processes is transforming the HTTP/S request into a format that can be used by a machine learning algorithm. The HTTP/S requests naturally come in either ASCII text format for HTTP version 1 and 1.1, and in binary format for HTTP version 2. None of these formats are understood and can be used by machine learning algorithms, so the process of transforming text into machine learning friendly language is called Text Featurizing. The larger concept of machine learning models understanding and analysing textual data is NLP or Natural Language Processing. Text featurizing is a technique in which text is transformed into numerical features in the form of vectors which can then be processed using mathematical and statistical operations to analyse, classify, find patterns, and extract insights from the input. Text featurizing can be done in a

multitude of ways, such as Bag-of-Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), Word Embeddings, etc.

The default method used in ML.NET and the way the machine learning in our example was built is using n-grams. N-grams are contiguous sequences of a number n of terms extracted from the initial string. They are great at capturing local-level context and can help machine learning models to understand the connections and links between nearby words. The features are made of such n-grams. The result of the FeaturizeText function is a vector of float values representing normalized counts of n-grams.

The machine learning algorithm that powers the MLE behind the WAF is known as SDCA Logistic Regression, where SDCA stands for Stochastic Dual Coordinate Ascent. SDCA Logistic Regression is an optimization algorithm used for solving large-scale linear and logistic regression problems [31]. Logistic regression has been used for many years in binary classification problems where the goal is to predict the probability of a statement being true or false. The SDCA algorithm works by minimizing a convex loss function by iteratively by randomly selecting a coordinate, updating its dual variable and them the primal variable accordingly in a stochastic fashion. The iterations are being repeated until convergence is achieved or until a maximum number of iterations calls a timeout. One of the advantages of SDCA logistic regression is that it is well-suited for large-scale datasets because it processes one data point at a time, which reduces memory requirements. Additionally, it can handle both sparse and dense data efficiently. One exceptionally good article [32] was written by Zhang and Shalev-Schwartz that goes very deep into the mathematics and statistics of the SDCA and has been a great aid in choosing and understanding the algorithm that is the engine of the ML-WAF. The details of the algorithm implementation and variables that must be minded are very well presented in the Microsoft official documentation [33].

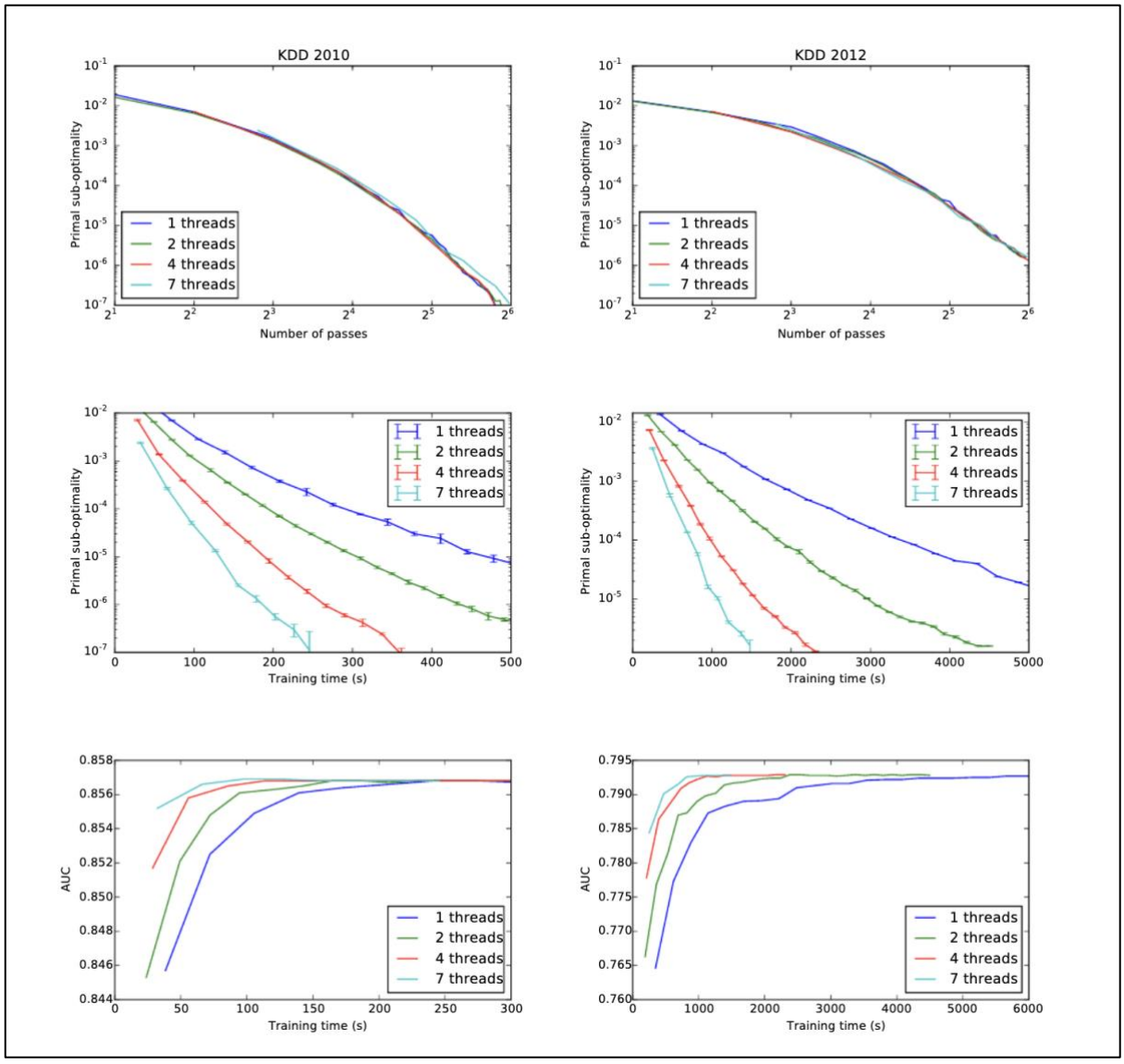

Figure 8 - Convergence of SA-SDCA for different number of threads [34]

Another great advantage of the SDCA algorithm is the fact that it was proved to handle scaling very well. The article at [34] provides an amazingly well-done study and experiment that demonstrates the impressive performance boost obtained by multi-threading the algorithm. Figure 8 taken from the same article is displaying the training type and number of passes improvements going from a single-threaded approach to multi-threading and increasing the number of threads. OBJ

5. Results

5.1. Machine Learning Model Results

The ML.NET Framework that was used to build the Machine Learning offers a built-in class for metrics that are used to evaluate trained models. It calculates statistical values such as

Accuracy, F1 score and Recall but also offers an amazingly fast way to obtain and see the Confusion Matrix. Figure 4 shows the results obtained from training the MLE model on 38297 entries from the CSIC 2010 HTTP Dataset [35]. Figure 4 presents the results returned by the Evaluate function in ML.NET after training the model splitting the dataset into 80% training data and 20% testing data.

```
Model quality metrics evaluation
--------------------------------
Accuracy: 99,25%
AUC-PR: 100,00%
AUC: 100,00%
Negative Precision: 98,97%
Negative Recall: 100,00%
Positive Precision: 100,00%
Positive Recall: 97,23%
F1Score: 98,60%
Confusion Matrix:
TEST POSITIVE RATIO:    0,2723 (2094,0/(2094,0+5597,0))
Confusion table
          ||========================
PREDICTED || positive | negative | Recall
TRUTH     ||========================
 positive ||    2.036 |       58 | 0,9723
 negative ||        0 |    5.597 | 1,0000
          ||========================
Precision ||   1,0000 |   0,9897 |
```

Figure 9 - Model quality metrics evaluation

Figure 9 shows that the model correctly classified 99.25% of the test data HTTP requests, indicating a remarkably high performance. Moreover, by analysing the Confusion Table we can see that the percentage is given by 58 out of 7691 requests that were mistakenly classified as negative instead of positive. It is important to state before discussing more on this confusion matrix that the positive values are malicious requests and negative values are legitimate requests.

One thing to notice is that all the failed predictions were false negatives meaning all the mistakes classified the request as being legitimate when it was a malicious request. This should give an indication that the model tends to let malicious requests fly under the radar rather than blocking legitimate requests that are suspicious. This false negative allowing bias can be a

problem to certain scenarios such as banking applications that may want to be stricter and risk blocking legitimate transactions rather than allowing frauds to happen.

The model bias can be however easily countered in the WAF component by tinkering with the probability threshold. By changing the threshold, the WAF can require a higher or a lower probability of maliciousness to block a request. The area under curve metrics shows an exceptionally good discrimination power and an excellent precision-recall trade-off balancing identification of positive or malicious requests with minimizing false positives. The F1 Score is also confirming the good balance between precision and recall.

Overall, the model has high performance across multiple metrics, indicating its effectiveness in correctly identifying both positive and negative samples. However, it is important to evaluate the model's performance on external data to ensure its generalizability. When it comes to the positive-negative discussion there is one limitation of the model that must be mentioned. In the context of cybersecurity, a benign result is where a security solution detects as malicious a legitimate activity. The difference between a benign and a false positive is that the false positive case is triggered by a misconfiguration, a wrong rule or bad training, a benign result is both a legitimate request and an activity that the security solution should detect as malicious.

One simple example to help understanding benign results is a “drop all tables” query that was legitimately executed by a network administrator on a test database schema. Dropping all the tables in a database is a malicious activity as it can lead to data loss and application denial of service, however in the context of a network administrator clearing space in a test environment that is a legitimate behaviour. This problem is one of the hardest to tackle in cybersecurity as it requires an incredibly good understanding of context, which machine learning tend to struggle with especially in low quantity training data scenarios. This limitation for the WAF will have maximum impact in the training stage and in the first period of usage, the backpropagation and continuous learning will in time help reduce the problems caused by benign results.

5.2. Performance Results

Another discussion of great interest in the case of implementing a Web Application Firewall, especially if it implies content analysis using machine learning, is the impact this new component has on the performance of the network. Forever since the development of security countermeasures the tradeoff between performance and security was a very debated topic. While providing a secure environment for the consumer, so is keeping the user experience as close to untouched as possible, for if a website is very secure but takes too long to load and operate, the business will be impacted. This is as true as ever considering we are living in a consumerism-based age and a period when not even gigabit speed internet is fast enough anymore.

Website speed tools such as Google PageSpeed Insights or Pingdom and many others were developed in order to test the performance of the website and of the infrastructure (web server, network etc.). Those tools calculate a score based on an average of timed tests such as page speed, page load time, blocking time, time to interactive etc. According to a New York Times study from 2012 – user experience is affected even by loading times lower than 400 milliseconds which is the time a human eye takes to blink. The study goes even further to say that a website slower than a competitor by more than 250 milliseconds will be in danger of losing ground to the competition [36]. Considering that this study was published in early 2012, we can only imagine that those times are lower in 2023, the time this paper is being written. If those arguments are not enough to convince someone of the importance of a website performance, search engines, including Google, give significant importance to site speed when determining the order of appearance in the search engine results page. This has opened a new job position and area of expertise called SEO - standing for Search Engine Optimization. A SEO-expert's job description is to make sure that the website they are representing appears first in the search engines for as many relevant queries as possible.

With all the arguments for the importance of site speeds and performance in mind, any cybersecurity solution provider, especially web-related, must be careful of their product's impact. Thus, the performance of the ML-WAF developed during this study was also subject to those tests and to optimizations – with the particularly important notice that the security the product offers was not traded off for performance by skipping checks or coding shenanigans to bypass certain functions in isolated cases. The optimization was limited to extracting every millisecond of performance while checking and taking the needed time to provide the most accurate machine learning generated probability for every query analyzed.

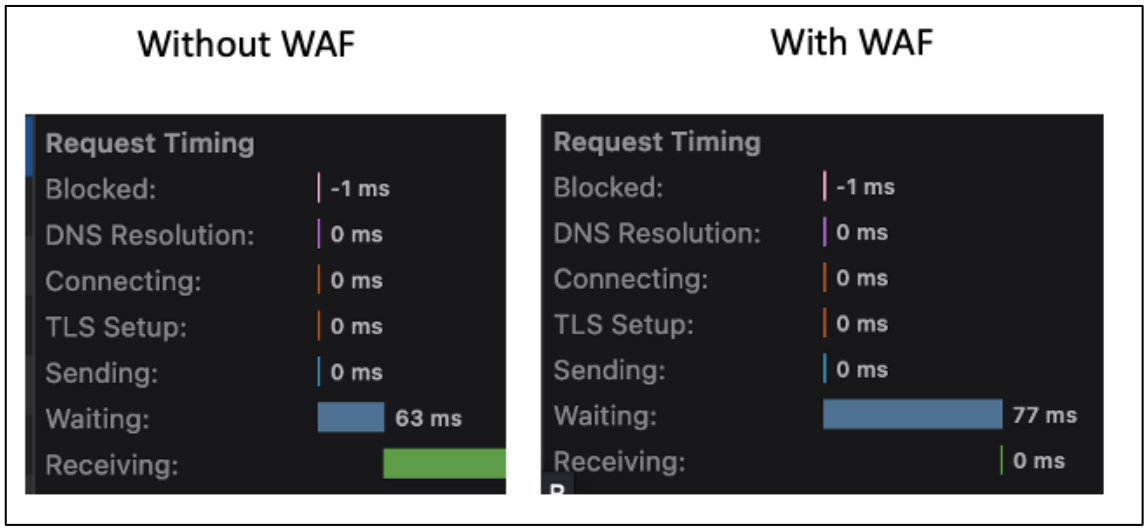

Figure 10 - WAF Performance Impact

Figure 10 is a compilation of two screenshots taken from the Developer Tools offered by the Mozilla Firefox web browser. The tests were performed on the same browser, in the same conditions (CPU temperature, RAM memory used, free network bandwidth, internet speed,

etc.) and on the same hardware (Apple M1 16GB MacBook Pro – running on macOS Ventura 13.1). A difference of only 14ms difference was detected when proxying a simple GET request for the example.com website – a website "reserved by the Internet Assigned Numbers Authority (IANA) at the direction of the Internet Engineering Task Force (IETF) as special-use domain names for documentation purposes" [37]. It is important to state that no SEO optimization techniques or development techniques were used in order to manipulate any of the times in the benchmarks. The techniques talked about are HTTP caching, using CDNs, optimizing images, minifying CSS and JavaScript files, prefetching etc.

Scenario/Times	Finish	DOMContentLoaded	Load
Without WAF	2570 ms	851 ms	852 ms
With WAF	2730 ms	950 ms	1040 ms
Difference	+160 ms	+99 ms	+188 ms

Figure 11 - YouTube Performance Impact

The test was also taken against a heavier website just to show that the WAF handles any type of web applications not just a simple HTML website. The heavy website tested was youtube.com and the differences were again insignificant. Figure 11 is a table that charts the times for both scenarios (with and without WAF) and the difference – meaning the impact of the WAF. The differences remain proportionally small as no difference passes the 22% margin. From those two tests we can affirm that the impact of the WAF grows linearly with the normal loading time. This is especially important as this means that very heavy applications or operations such as uploading and downloading large files will not suffer an exponentially growing waiting time when passing through the Web Application Firewall.

6. Conclusions

As it was mentioned in the introduction, the purpose of this paper was to offer a technical knowledge baseline to anyone who wants to build a cybersecurity solution that employs machine learning algorithms to boost its performance.

After reading this paper the added value the reader will have is that of a high-level understanding regarding the decisions that are to be taken along the development process, the architecture and components needed and the knowledge of evaluating and understanding the results obtained. This high-level understanding is obtained by following along the thought-process documented during this implementation and finding the problems encountered by me before encountering them themselves.

This architecture and technical design could be used in production at a below-average to average infrastructure (web application and network). Regarding the implementation, however, there are many improvements that can be made to the solution presented and certain notches that can be slightly altered to fit needs. I trust, however, that the presented configuration is a stable and trustworthy foundation to be built on to achieve reliable software production. The improvements that can be built upon are implementing a better machine learning training workflow, for example having a learning mode phase where the WAF ingests all the traffic and the requests are manually labeled as malicious or normal by actual humans (employees, developers, network engineers etc.). Another improvement that can be implemented is having a control center where humans can whitelist or blacklist certain requests, thus not relying only on autonomous learning and improving the machine learning engine.

The biggest limitation of the study conducted was the lack of high-quality training data. Arguably, the most important factor in building a machine learning model is the amount and quality of the training dataset, as there is a positive linear relationship between the data quality and the model quality.

I would like to end this paper by stating that it is our responsibility as researchers and cybersecurity professionals to not only keep up with the pace of threat actors but to also outsmart them by building better, more secure, more performant security solutions and I have faith that every single small improvement that we can bring in the community makes it stronger.

Bibliography

[1] M. I. Danish and G. Shyam, "A Survey on Web Application Security," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology,* vol. 6, no. 5, pp. 223-228, 2020.

[2] D. Pałka and M. Zachara, "Learning Web Application Firewall - Benefits and Caveats," *Lecture Notes in Computer Science,* vol. 6908, p. 295–308, 2011.

[3] H. Yulianton and S. Warnars, "Web Security and Vulnerability: A Literature Review," *Journal of Physics: Conference Series,* vol. 1477, p. 022028, 2020.

[4] K. ngham, K. Consulting and S. Forrest, "A history and survey of network firewalls," *ACM Journal Name,* pp. 1-42, 2002.

[5] Wikipedia, "Web application firewall," Wikipedia, 5 July 2016. [Online]. Available: https://en.wikipedia.org/wiki/Web_application_firewall. [Accessed 29 March 2023].

[6] A. Temotipe, T.-A. Aderonke, A. Femi and O. John, "Stop Cyber Attacks Before They Happen: Harnessing The Power Of Predictive Analytics In Cybersecurity," *Journal of Multidisciplinary Engineering Science and Technology (JMEST)* , vol. 10, no. 4, 2023.

[7] G. Charles, "The Latest 2023 Cyber Crime Statistics (updated March 2023)," AAG-IT, 06 March 2023. [Online]. Available: https://aag-it.com/the-latest-cyber-crime-statistics/. [Accessed 5 April 2023].

[8] European Union Agency for Cybersecurity, ENISA threat landscape 2022: July 2021 to July 2022., Publications Office, 2022.

[9] H. Suryotrisongko and Y. Musashi, "Review of Cybersecurity Research Topics, Taxonomy and Challenges: Interdisciplinary Perspective," *2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA),* pp. 162-167, 2019.

[10] "Review articles in CYBER SECURITY," ResearchGate, [Online]. Available: https://www.researchgate.net/topic/Cyber-Security/publications. [Accessed 5 April 2023].

[11] D. Hoang, N. Trang and N. Hưng, A Survey of Tools and Techniques for Web Attack Detection, 2022.

[12] C. Michael, "Defence in depth, protection in depth and security in depth: A comparative analysis towards a common usage language," in *The Proceedings of the 5th Australian Security and Intelligence Conference*, Perth, Western Australia, 2012.

[13] K. Yousaf, A. Iftikhar, A. Javed and A. Tahir, "Explore and Exploit Security Flaws in Web Applicationsfor Implementing Efficient Security Provision Techniques," *International Journal of Information and Education Technology,* pp. 143-148, 2012.

[14] "OWASP Top10:2021," OWASP, 2021. [Online]. Available: https://owasp.org/Top10/. [Accessed 5 April 2023].

[15] G. Dr. A.SHAJI and G. A.S.HOVAN, "A Brief Study on The Evolution of Next Generation Firewall and Web Application Firewall," *IJ ARCCE,* vol. 10, no. 5, 2021.

[16] M. Batta, "Machine Learning Algorithms - A Review," *International Journal of Science and Research (IJSR,* 2018.

[17] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys & Tutorials,* vol. 17, no. 4, pp. 2347-2376, 2015.

[18] G. Ian, B. Yoshua and C. Aaron, Deep Learning, Cambridge, MA, USA: MIT Press, 2016.

[19] T.-G. Carmen, P.-V. Alejandro and A. Gonzalo, "A Self-learning Anomaly-Based Web Application Firewall," *Advances in Intelligent and Soft Computing,* pp. 85-92, 1009.

[20] M. Abbasi, A. Shahraki and A. Taherkordi, "Deep Learning for Network Traffic Monitoring and Analysis (NTMA): A Survey.," *Computer Communications,* vol. 170, pp. 19-41, 2021.

[21] O. Victor and C. Analogbei, "Automatic Detection of HTTP Injection Attacks using Convolutional Neural Network and Deep Neural Network," *J. Cyber Secur. Mobil.,* vol. 9, pp. 489-514, 2020.

[22] "Programming Language and compiler Benchmarks," 4 May 2023. [Online]. Available: https://programming-language-benchmarks.vercel.app/c-vs-csharp. [Accessed 27 May 2023].

[23] Microsoft, "System.Net Namespace," Microsoft, [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/system.net?view=net-7.0. [Accessed 27 May 2023].

[24] Microsoft, "Create web APIs with ASP.NET Core," Microsoft, [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-6.0. [Accessed 27 May 2023].

[25] Microsoft, "ML.NET Documentation," Microsoft, [Online]. Available: https://learn.microsoft.com/en-us/dotnet/machine-learning/. [Accessed 27 May 2023].

[26] Elastic, "An overview of the Elastic Stack," [Online]. Available: https://www.elastic.co/guide/en/welcome-to-elastic/current/stack-components.html. [Accessed 5 May 2023].

[27] T. Radivilova, L. Kirichenko, D. Ageyev, M. Tawalbeh and V. Bulakh, "Decrypting SSL/TLS traffic for hidden threats detection," *IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT),* 2018.

[28] "Decrypt SSL traffic with the SSLKEYLOGFILE environment variable on Firefox or Google Chrome using Wireshark," F5, [Online]. Available: https://my.f5.com/manage/s/article/K50557518#OnLinux. [Accessed 5 June 2023].

[29] L. Bocheng and Y. Fan, "TTP Traffic Analysis based on Multiple Deep Convolution Network Model Generation Algorithms," *Journal of Physics: Conference Series,* vol. 2037, no. 1, p. 012048, 2021.

[30] I. Sanchez, R. Satta, I. Nai Fovino, G. Baldini, G. Steri, D. Shaw and A. Ciardulli, "Privacy leakages in Smart Home Wireless Technologies," *Proceedings - International Carnahan Conference on Security Technology,* 2014.

[31] D. Guillaume, K. Michael and R. Matthieu, *Stochastic Dual Coordinate Ascent,* 2018.

[32] S.-S. Shai and Z. Tong, "Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization," *Journal of Machine Learning Research ,* vol. 14, pp. 567-599, 2013.

[33] Microsoft, "SdcaLogisticRegressionBinaryTrainer Class," Microsoft, 28 May 2019. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/microsoft.ml.trainers.sdcalogisticregressionbinarytrainer?view=ml-dotnet. [Accessed 4 May 2023].

[34] K. Tran, S. Hosseini, L. Xiao, T. Finley and M. Bilenko, "Scaling Up Stochastic Dual Coordinate Ascent," 2015.

[35] G. Carmen Torrano, V. Alejandro Pérez and M. Gonzalo Álvarez, "HTTP DATASET CSIC 2010," 2010. [Online]. Available: https://www.tic.itefi.csic.es/dataset/. [Accessed 4 May 2023].

[36] S. Lohr, "For Impatient Web Users, an Eye Blink Is Just Too Long to Wait," The New York Times, 29 February 2012. [Online]. Available: https://www.nytimes.com/2012/03/01/technology/impatient-web-users-flee-slow-loading-sites.html. [Accessed 11 June 2023].

[37] "example.com," Wikipedia, 11 May 2023. [Online]. Available: https://en.wikipedia.org/wiki/Example.com. [Accessed 11 June 2023].