SOLENT UNIVERSITY
FACULTY OF BUSINESS LAW AND DIGITAL TECHNOLOGIES

MSc Cyber Security Engineering
Academic Year 2021-2022

Oluwaseun Aransiola

Website vulnerabilities and Web Application Firewall Protection mechanism

Supervisor: Associate Prof. Kalin Penev **September 2022**

This report is submitted in partial fulfillment of the requirements of Solent University for the degree of MSc Cyber Security Engineering

Acknowledgement

I would love to thank my wife, Opeyemi Fikunyi, Aransiola for her unwavering support and understanding during this MSc Dissertation. I would like to appreciate my supervisor, Dr. Kalin for his guidance and for constantly emphasizing the need to read extensively on the subject which has really helped me to improve this work significantly.

Thanks to all Solent University lecturers who have contributed to me over the last year; I'm immensely grateful, it's been a huge privilege to learn from you all. Thank you to all my colleagues for your friendship and for making learning fun.

> "So he said to me, "This is the word of the LORD to Zerubbabel: 'Not by might nor by power, but by my Spirit,' says the LORD Almighty."
>
> Zechariah 4:6

Abstract

Web applications are used to deploy services and products over the internet for easy accessibility. As a result, they have been a constant target for malicious actors to exfiltrate sensitive data leading to service distribution and financial loss. Web Application Firewall (WAF) has been a helpful tool in providing web applications with much-needed security. Nevertheless, attacks targeted against this insulated web application are still successful. This MSc research investigates the factor that could be responsible for the reliability of WAF as a protection tool, illustrate the effectiveness of Application-level authentication to mitigate against WAF bypass, and also analyzes the performance impact of WAF using Apache Bench, Jmeter, and Gatling. The result shows from experiments shows that the configuration of the web server is a primary factor affecting the efficacy of WAF. Furthermore, the analysis reveals that WAF requests connection time increases as the number of concurrent users although the request transfer rate reduces which validates the advantages of the edge server used by most Cloud-based providers. Additionally, this study implements a proposed WAF-VERIFIER to mitigate against WAF bypass as illustrated in this study. It leverages the IP Whitelist approach and WAF footprints to inspect and ensure incoming request is proxied through WAF as intended. The results show web application security was enhanced as the WAF-VERIFIER rejects bypassed requests. Scanning tools were also used to carry out web application vulnerability assessment and the results were mapped with OWASP's Top 10 most critical security risks. The results show that 85% of issues reported in the target system belong to the OWASP Top 10 even though these issues were categorized as *Low* or *Info* severity. Hence, this study recommends web application severity should be treated with the same precedence. WAF should be used as secondary protection layer with the primary protection implemented during the web application development and hardening of the web server, making the application more resilient.

Symbols And Abbreviations

ACL - Access Control List

AWS - Amazon Web Services

CI/CD - Continous Integration Continous Deployment

CWEs - Common Weakness Enumerations

DAST - Dynamic Application Security Testing

LDAP - Lightweight Directory Access Protocol

IDS - Intrusion Detection System

IPS - Intrusion Prevention System

WAF - Web Application Firewall

XSS - Cross-site Scripting

EISP - Enterprise Information Security Policy

ISSPs - Issue-Specific Security Policies

OWASP - Open Web Application Security Project

SysSPs - System Specific Policies

VPS - Virtual Private Server

VPS - Virtual Private Server

List of Figures

List of Tables

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION AND BACKGROUND

This chapter presents overview insights into web applications and challenges that triggered this MSc research. It summarizes existing research in relation to Web Application vulnerabilities and the role of a Web Application Firewall.

1.1 INTRODUCTION

Web applications are one of the most widely used platforms for delivering products and services over the Internet. As a result, it has become a popular and attractive target for security attacks as they are increasingly been used for critical services. Web application security has become more important. They are designed using HTML and CSS and are often rendered through a web server that is connected to a database server usually MySQL or SQLite depending on the programming language such as Java, Python, and PHP. Cross-platform compatibility and ease of development are factors responsible for the popularity and adoption of web applications (Li and Xue 2014).

Web servers are often software such as Apache HTTP Server or Nginx that serves content based on the request it received over the internet. A web server's importance and infrastructure position make it a subject of constant attack. The exploitation of some vulnerabilities causes the majority of security breaches. As a result, ensuring web application security to prevent data breaches is imperative as they are exceedingly regarded as insecure (Bhor 2016). The payload coming from the internet to the webserver is usually the carrier of this threat (Torrano-Gimenez *et al.,* 2011), therefore checking the request header will not suffice. Attackers modified the request payloads to exploit vulnerabilities in a web application that can result in XSS attacks, SQL Injection, and File Inclusion among others. When untrusted data passes via invalidated fields, this is known as Cross-Site Scripting, or XSS (Nunan *et al.*, 2013). This can affect the availability of the web servers or result in loss of data. Attacks against the web server are constantly evolving and dynamic in nature. According to OWASP Top 10 (Marchand-Melsom 2020), 94% of web applications tested have some form of injection vulnerabilities such as SQL, OS, and LDAP injections. A malicious payload is provided to the website and executed by the web application as a result of these attacks and is sometimes used to launch other attacks such as the exfiltration of sensitive information such as cookies (Steffens *et al.,* 2019).

Every day, new web-based attack strategies that pose a threat to web applications are developed. As a result, Web Application Firewall (WAF) studies and improvements are gaining traction to combat and prevent these attacks. The typical objectives of these attacks range from compromising the web server's functionality and availability to getting unauthorized access to the hosted web material. Intrusion Prevention System (IPS) and Intrusion Detection System (IDS) are network equipment that is introduced to the computer network to prevent these attacks (Endraca *et al.,* 2013 p. 451) however, they are not sufficient in protecting web applications.

According to D'Hoinne (2013), a Web Application Firewall is an extraction of IDS and IPS for HTTP applications as it understands HTTP traffic much better, however, the Ponemon survey issued in 2019, highlights the satisfaction with WAFs (Web Application Firewalls) is at 40%, with a performance at only 43% (Ponemon, 2019). Furthermore, the study shows that 65% of WAFs are bypassed during attacks, and as a result, organizations don't think WAFs are effective in securing web applications. With the increase in website attacks in recent times and the increasing adoption of WAF, Gartner's projection is that more than 30% of web applications will be protected by WAF by 2023, which is a 50% increase compared to 2020.

1.2 BACKGROUND

1.2.1 WEB APPLICATIONS AND VULNERABILITIES

The majority of attacks occur due to implementation errors ranging from faulty input validation, improper authorization and authentication procedures, improper session management, and other issues that compromise the intended functionality of the application (Wichers and Williams 2017).

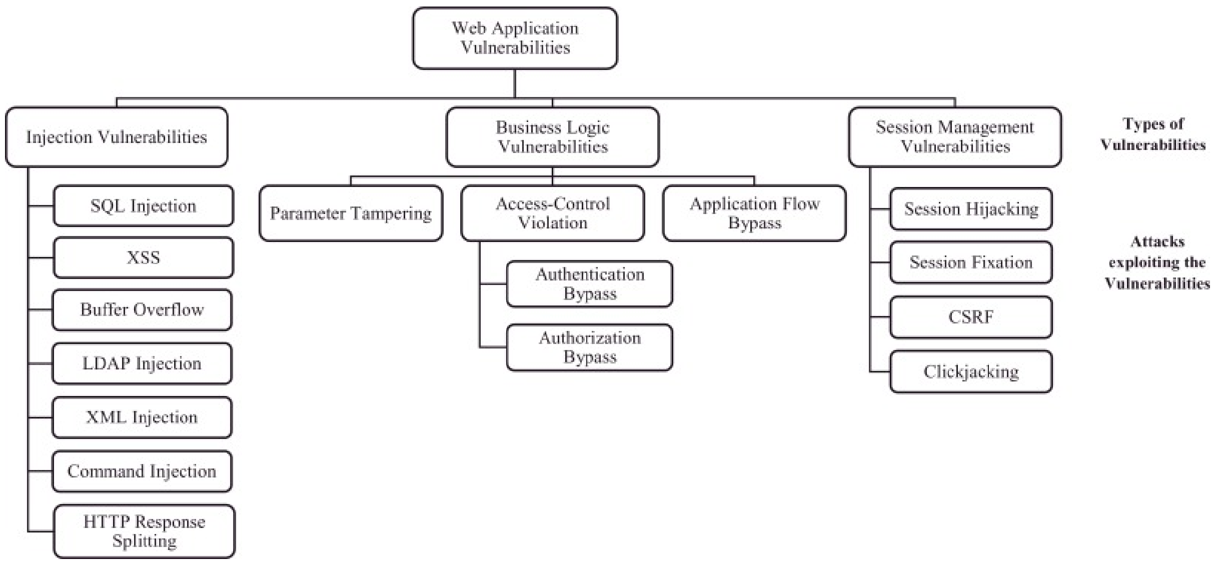


Figure 1: Vulnerabilities and attacks (Deepa and Thilagam 2016)

According to Touseef *et al*., (2019) during the analysis of automated web application, it was discovered that the most occurring risk are SQL, XSS and sensitive data exposure. To secure web applications against SQL Injections and XSS attacks, (Fonseca *et al.,* 2013) outlines code errors that should be avoided in both high-level programming languages such as C#, Java, C++ e.t.c and weakly typed programming language such as PHP. According to W3Techs (2022), PHP is utilized by 77.5% of all websites with a known server-side programming language. Therefore, It is constantly a prey of SQL Injections and XSS attacks.

Assal and Chiasson's (2019) survey on developers, the study affirms that developers are not the weakest security link, and organizational mismanagement are responsible such as the lack of security plans, resources, and security policy such as EISP, ISSPs, and SysSPs as outlined by (Whitman and Mattord 2017. p.54). Bhor (2016) argued that some web application developers are unfamiliar with secure coding standards and prioritize application functionality over security needs hence attacks on the web application can be minimized by utilizing secure coding practices within the web application software's design pattern. Furthermore, they frequently lack the appropriate knowledge of security requirements and web applications: lack of security maturity (Weir *et al*., 2021) hence making them very vulnerable to security exploits.

1.2.2 WEB APPLICATION FIREWALL

Tekerek and Bay (2019) define a Web Application Firewall as software or hardware that monitor, filter, and block HTTP traffic between a client and web server. It can be used as an incorporated firewall in the web server such as Modsecurity or as a separate security layer in the form of a reverse proxy such as Cloudflare (Muzaki et al., 2020). This is the most convenient method of conducting a comprehensive inspection and enforcing policies.

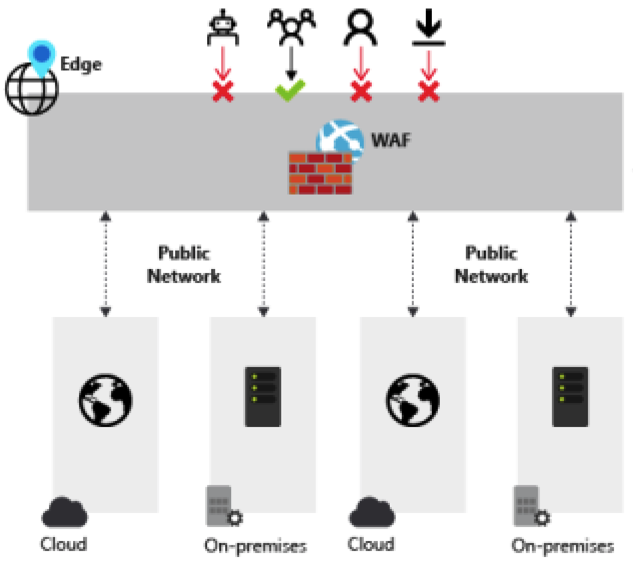


Figure 2 : A Web application firewall sits between the end user and the application server (Djeffal, 2020)

The OSI (Open System Interconnection) was proposed in 1983 and it covers all aspects of network communication which consist of seven layers namely Physical, Data Link, Network, Transport, Session, Presentation, and Application layers (Saxena 2014). The Network layer which provides the network path is where IP operates. UDP and TCP are contained in the Transport layer and it is concerned with the data transportation from the source to the destination system. Traditional Firewall that blocks suspicious packets works at Network and Transport which are Layer 3 and layer 4, however, the Web Application Firewall works at the Application layer which is the highest layer on the model, and understands HTTP traffic much better (Pantoulas, 2022). This is because attackers are targeting HTTP and not just the network and as a result, tools used to protect the Network layer can not sufficiently secure the HTTP/Application layer. WAFs have to cope with extremely expressive languages like SQL and HTML since they work at the application level and as result, it worsens the detection issue (Perera *et al,* 2016).

Furthermore, WAF can cope with application-layer threats in a way that a traditional firewall can't. A typical WAF uses a regular expression that refers to pre-defined rules to prevent malicious or suspicious payloads. The security model can be divided into three categories: the Positive Security Model which is also known as the white list model, the Negative Security Model which is also known as the blacklist model, and the Hybrid security model which is a combination of both Positive and Negative security models. The Negative security model allows all but known harmful payloads by referencing the predetermined rules whereas the positive security model denies all but known good payloads using traffic patterns (Takahashi *et al.*, 2011, Nagendran et al., 2020). Because all that is required is to define blacklist rules based on known attacks, blacklist-based techniques are reasonably fastly adopted, and protect several applications but tend to have more false positives. Whitelist-based approaches, on the other hand, are more likely to block those attacks, however, they require a detailed definition and understanding of the application, hence creating rules is usually time-consuming.

Razzaq et al., (2013) comparative study on various web application firewall solutions, it was observed that it is difficult for organizations to select the best security solution are they are faced with various many open source and commercial solutions. As a result, a proper understanding of the benefit and trade-offs for various WAFs becomes imperative. Mod Security, for example, is unable to understand HTML hidden field attacks while Barracuda, a commercial firewall has a poor reporting and analysis mechanism with load-balancing not so efficient although it has a security effectiveness of 89.7% (Skybakmoen 2019).

Clincy and Shahriar (2018) discussed the importance of the Web Application Firewall and how the positive and negative based detection models affect the effectiveness of the Firewall, Hence, security testing, and good security practices in addition to the implementation of the Web Application Firewall were recommended. Therefore, WAF is a single security component in a security component network of infrastructure and should be treated as such.

Demetrio *et al* (2020) carried out an experiment that evidenced machine learning-based Web Application Firewall faces a real risk of being bypassed. Furthermore, the experiment showed that WAF-A-MoLE, a tool that simulates an adversary payload, bypassed all the considered machine learning-based WAFs. WAF bypass can be intentional to accurately assess the security posture of an application through penetration testing (Bijjou 2019), however, easily bypassed WAFs defeat the whole purpose of this security layer as an adversary can also take advantage of this weakness to exploit the application.

1.3 RESEARCH AIM, OBJECTIVES, AND QUESTION

Research Aim:

The aim of this MSc research project is to investigate the prevalence of web attacks despite the availability Web Application Firewall.

Research Objectives:

- To analyze the performance of Cloud-based WAF
- To discuss the web vulnerabilities issues and the role of WAF
- To illustrate and mitigate WAF Bypass challenges

Research Questions:

What factors affect the reliability of WAF in the protection web application?

How effective is application-level authentication to prevent Web Application Firewall bypass to ensure application reliability?

In summary, web applications will constantly be a target for the adversary because of their crucial role in information and service delivery. Its security is therefore important and the Web Application Firewall has been playing its role in filtering and monitoring HTTP traffic as discussed in the background body of this work. From the literature, some issues were identified such as WAF bypass, False-positves, and configurations among others. Hence the research question is, is WAF a reliable solution for protecting web applications? A pilot study will be carried out in the next phase of the project to look into the protection mechanism of WAF.

CHAPTER 2. PILOT STUDY

This pilot study was carried out to ensure the feasibility of the main research project. The pilot study carried out shows how the existing Web Applications Firewall can be bypassed and how this affects the security of the web application it protects.

2.1 PILOT STUDY INTRODUCTION

Web applications' ease of use and access has made them a common platform for business, social and educational activities. These activities are under constant attack as Checkpoint reports show a 50% increase in corporate network attacks in 2021 compare to 2020 (Checkpoint 2022). As a result, there has been a continual interest in the security of web applications (Singh and Ishtiaque-Al-Mahmood 2022), and the focus has been on proactive measures such as detection and prevention of attacks rather than reactive measures (Moradi *et al.,* 2019). Firewall and IDS scrutinize packets at the network layer but are not effective to monitor packets at the Application layer to protect HTTP web applications which necessitate the usage of Web Application Firewall for protection against SQL Injection and XSS (Shaheed 2022).

The pilot study is to investigate the Web Application Firewall's reliable solution for protecting web applications by testing how attackers bypass this security layer. It uses a range of web application firewall fingerprinting and reconnaissance tools to identify the type of Firewall and how it can be bypassed. Two methods of bypassing were used, Firstly, Locating the origin IP of the server, and secondly, the modification of the request payload. The limitations of the research methods were highlighted. Furthermore, the pilot study will highlight the current state of web application firewall security and give direction to main MSc research.

2.2 PILOT STUDY METHODOLOGY

The following sections highlights the research method used to analyze the current state of web application firewall security and how evading this security affects web application.

2.2.1 Research Method

Research methods are procedures and techniques used in conducting research (Kothari and Gaurav 2019, p.6). Research design, data collection procedure, and data analysis are the three main components of research methods (Atmowardoyo 2018). Scientific research is commonly used to support or refute a theory through investigation, experts have categorized it into two main research approaches: quantitative approach and qualitative approach. Research methodologists have introduced a third research approach that leverages the strength of both quantitative and qualitative which is knowns mixed approach(Morse 2016).

The quantitative method is based on numerical or statistical data and aims to measure something between two or more variables (Castellan 2010), (Apuke 2017). This research approach leads to different types of research methods such as surveys, experiments, quasi-experiments, and correlation studies (Ratelle *et al.,* 2019).

In the survey research method, data are collected by predefined questionnaires and the respondents are representative of the population sampled (Williams 2007). This type of method is common in social sciences (Saunders *et al.,* 2009) and the aim of the survey method is usually explorative, descriptive, or explanatory (Wagner et al., 2020).

The experimental research method investigates the treatment of an intervention in the study group and then assesses the treatment's outcomes (Ross and Morrison, 2013). Leedy and Ormrod (2019) classified this experimental method into three: pre-experimental, truly experimental, and quasi-experimental. In the pre-experimental study, pretest and posttest are used on a subject or a group, and changes are observed after a treatment has been applied to test whether the treatment has the potential to cause change (Thyer 2010). True experiments rely on statistical analysis to support or refute a hypothesis while In Quasi-experiments, independent variables are manipulated, and study participants are not randomly chosen, hence it is typically in the middle of correlational studies and true experiments (Pricee *et al.,* 2015).

The correlational research method is a nonexperimental research quantitative approach that is used to measure the degree of association between variables and allow research to make predictions based on discovered relationships (Seeram 2019).

The qualitative, on the other hand, is a holistic approach that tends to understand “why” or how rather than “how many” (McCusker and Gunaydin 2015). Research methods in this approach include case study, ethnography study, content analysis, grounded theory, phenomenological study, biographical study, participant observation, and narrative inquiry (Atmowardoyo 2018).

The case study research method is widely used in social sciences studies that allow researchers to get an in-depth explanation about a person or group of people by highlighting how and why questions (Saxena and Saxena 2019). In a similar way, ethnography study is more about constructivism and participatory paradigm mostly used to describe research methods of anthropologists (White 2009). The content analysis method is used to analyze content to determine the presence of certain words, themes, or visual data (Stemler 2015). The grounded theory research method focuses on developing theories that are "grounded" in data that has been systematically gathered and examined (Bakker 2019). In the phenomenological method, researchers match their research with a cohesive phenomenological structure in order to understand the substance of participants' lived experiences (Flynn and Korcuska, 2018). The biographical method studies the links between individuals and society (Caetano and Nico 2019), it has a long history in social sciences. The participant observation method is most used alongside the ethnographic method where participants are observed and studied rather been reported by informants (Boccagni and Schrooten 2018). In contrast, In narrative inquiry, stories such as human experiences, history, and culture are sources of data (Clandinin 2016).

In this study, a quantitative research approach was found to be more appropriate using a pre-experimental method in which a pretest was carried out on web applications protected by web application firewalls and a bypass attempt was carried out, a treatment (developed WAF VERIFIER) was then applied and a post-test was further carried to see if the treatment has the potential to cause change. Scholars agree that the choice of method minimizes the threat to internal validity (Cahit 2015).

2.2.2 Data Collection

Information gathering is essential for research and data collection sources can either be primary or secondary (Mazhar *et al.,* 2021). Data collected directly by the researcher are referred to as primary data while secondary data allows the researcher to leverage existing data collected by someone else with the advantage of saving time and resources (Jonhson 2017). In this study, data collection uses primary data obtained from a test bed environment set up to illustrate a web application firewall.

2.2.3 Experimental Design

According to Kothari and Gaurav (2019, p. 39), define experimental design is the framework or structure of an experiment and consists of seven types of designs namely: Before-and-after without control design, After-only with control design, Before-and-after with control, Completely randomized design (C.R Design), Randomized block design (R.B Design), Latin square design (L.S Design) and Factorial design. In this study, a Before-and-After Without Control design was found used which allows the dependent variable before the introduction of treatment, and the dependent variable was measured again after the introduction of treatment as shown in Figure 3. Hence it is found to be more appropriate for this study.

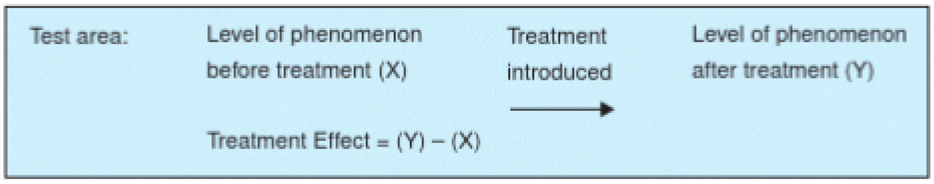

Test area:
Level of phenomenon before treatment (X)
Treatment introduced
Level of phenomenon after treatment (Y)
Treatment Effect = (Y) − (X)

Figure 3: Before-and-After Without Control Design (Kothari and Gaurav 2019, p. 40)

2.2.4 Experimental Setup

The testbed compares different Web Application firewalls on common platforms. The Testbed for this pilot study consists of 2 Deliberately Vulnerable Web Applications (DVWA) that are susceptible to XSS, SQLi, and other forms of attack. It is hosted on VPS on both Digitalocean and AWS Elastic Instance (EC2). The first two web applications were served over an Apache HTTP server.

On each of the VPS servers, a Cloud-based Firewall was configured from different providers. A bypass was attempted on either directly identifying the origin server IP or manipulating the payload.

This led to the research hypothesis that Web Application Firewall effectiveness and protection mechanism can be affected when bypassed either by direct access to the origin server.

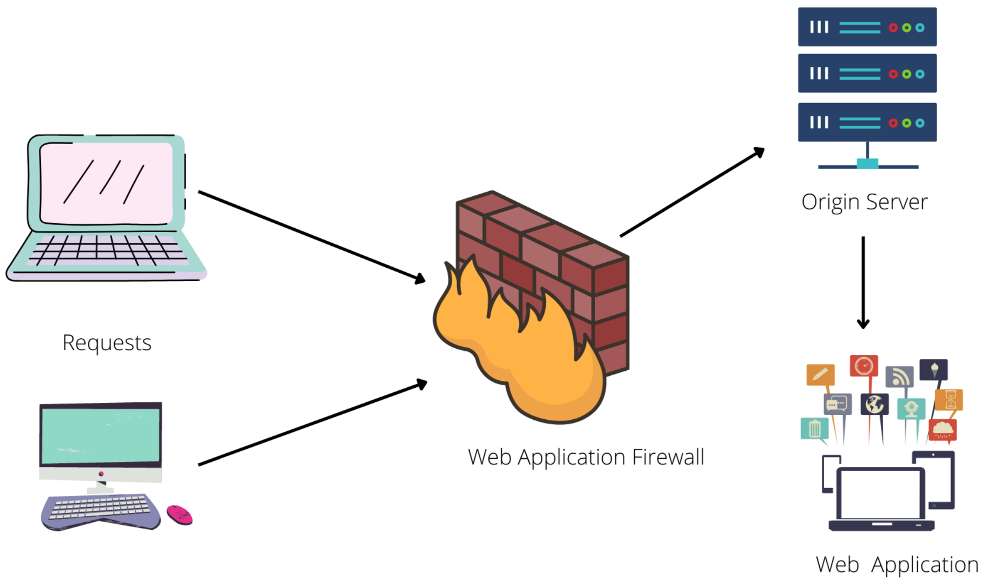
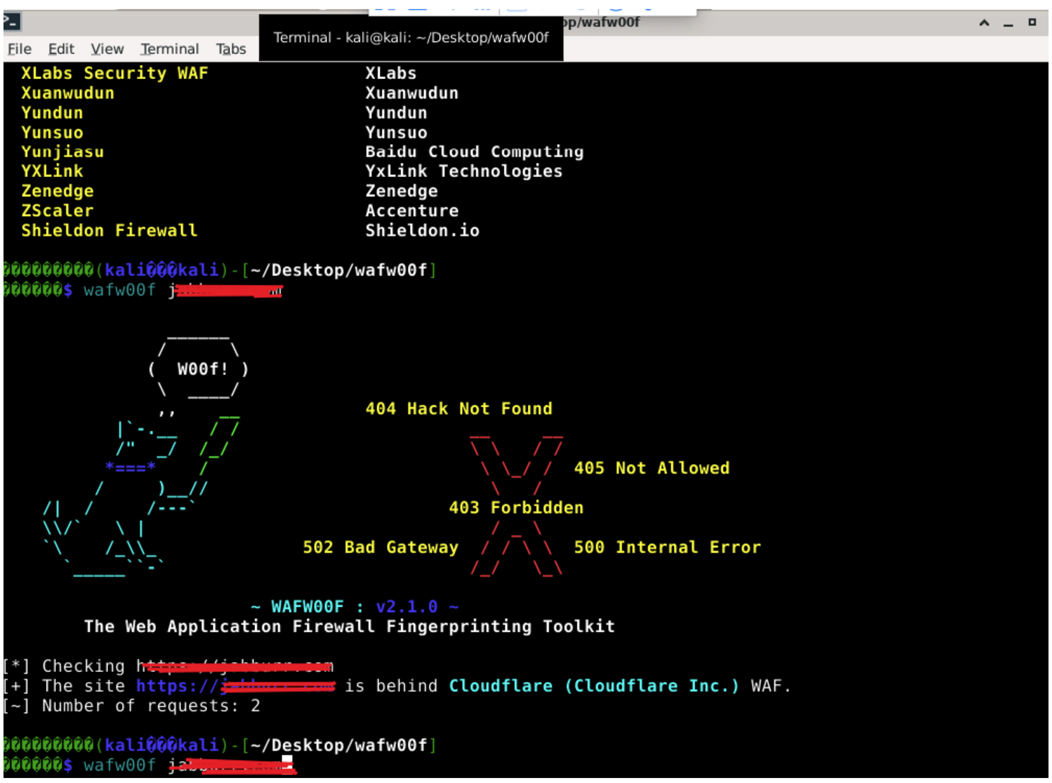


Figure 4: Overview Testbed Setup

2.3 PILOT STUDY RESULTS

Cloudflare is a website security company that offers content-delivery network services, DDoS mitigation, and Internet security for businesses. Businesses leverage Cloudflare Web Application Firewall to block SQL Injection, and XSS through known tactics and payloads.

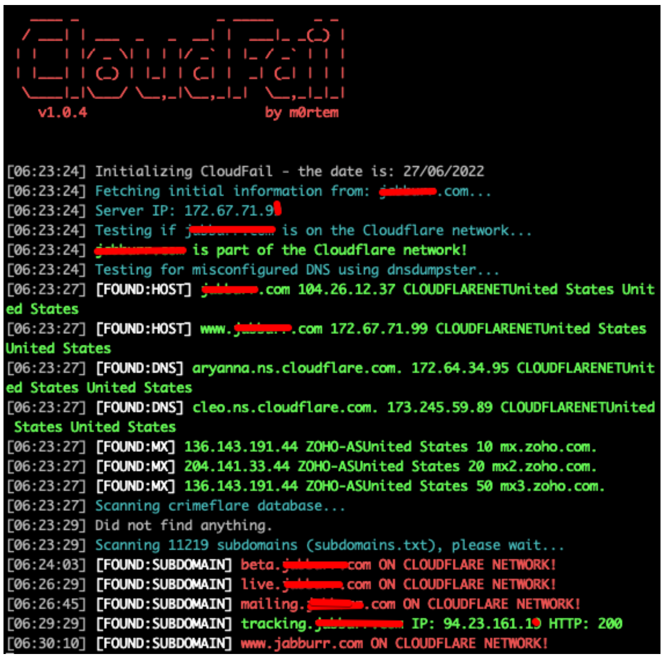
Identifying if the target is protected by a Web Application Firewall is the first step. Wafw00f, A web application fingering tool was used to determine the type of WAF protecting a target such as Cisco ACE XML Gateway, Cloudflare, Incapsula Imperva Inc, IBM Datapower e.t.c (Gauci and Mondal 2020). Our target domain name was tested with Wafw00f and it accurately reveals Cloudflare as shown in Figure 5.


Terminal - kali@kali: ~/Desktop/wafw00f
File Edit View Terminal Tabs

```
  XLabs Security WAF                XLabs
  Xuanwudun                         Xuanwudun
  Yundun                            Yundun
  Yunsuo                            Yunsuo
  Yunjiasu                          Baidu Cloud Computing
  YXLink                            YxLink Technologies
  Zenedge                           Zenedge
  ZScaler                           Accenture
  Shieldon Firewall                 Shieldon.io

?????????(kali???kali)-[~/Desktop/wafw00f]
??????$ wafw00f j[redacted]


                ______
               /      \
              (  W00f! )
               \  ____/
               ,,    __            404 Hack Not Found
           |`-.__   / /                      __     __
           /"  _/  /_/                       \ \   / /
          *===*    /                          \ \_/ /  405 Not Allowed
         /      )__//                          \   /
    /|  /     /---`                     403 Forbidden
    \\/`    \ |                                / _ \
    `\    /_\\_                 502 Bad Gateway / / \ \  500 Internal Error
      `_____``-`                             /_/   \_\

                        ~ WAFW00F : v2.1.0 ~
        The Web Application Firewall Fingerprinting Toolkit

[*] Checking h[redacted]
[+] The site https://[redacted] is behind Cloudflare (Cloudflare Inc.) WAF.
[~] Number of requests: 2

?????????(kali???kali)-[~/Desktop/wafw00f]
??????$ wafw00f ja[redacted]
```

Figure 5: Wafw00f identified target Web Application Firewall

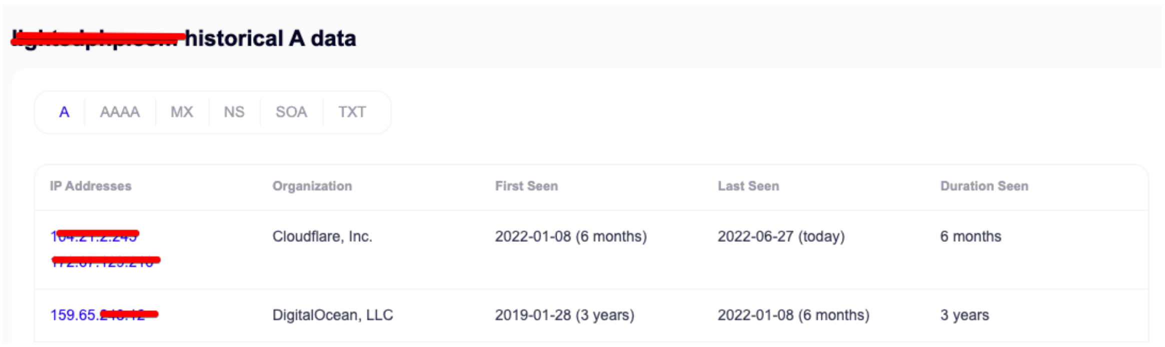
The type of WAF has been identified, Identifying the origin IP can be detected using another reconnaissance tool specifically targeted at Cloudflare WAF such as Cloudfail. Cloudfail was tested on our target and the Origin server IP along with other details such as subdomain was detected as shown in Figure 6.



```
  v1.0.4                          by m0rtem


[06:23:24] Initializing CloudFail - the date is: 27/06/2022
[06:23:24] Fetching initial information from: [redacted].com...
[06:23:24] Server IP: 172.67.71.9[redacted]
[06:23:24] Testing if [redacted] is on the Cloudflare network...
[06:23:24] [redacted] is part of the Cloudflare network!
[06:23:24] Testing for misconfigured DNS using dnsdumpster...
[06:23:27] [FOUND:HOST] [redacted].com 104.26.12.37 CLOUDFLARENETUnited States Unit
ed States
[06:23:27] [FOUND:HOST] www.[redacted].com 172.67.71.99 CLOUDFLARENETUnited States
United States
[06:23:27] [FOUND:DNS] aryanna.ns.cloudflare.com. 172.64.34.95 CLOUDFLARENETUnit
ed States United States
[06:23:27] [FOUND:DNS] cleo.ns.cloudflare.com. 173.245.59.89 CLOUDFLARENETUnited
 States United States
[06:23:27] [FOUND:MX] 136.143.191.44 ZOHO-ASUnited States 10 mx.zoho.com.
[06:23:27] [FOUND:MX] 204.141.33.44 ZOHO-ASUnited States 20 mx2.zoho.com.
[06:23:27] [FOUND:MX] 136.143.191.44 ZOHO-ASUnited States 50 mx3.zoho.com.
[06:23:27] Scanning crimeflare database...
[06:23:29] Did not find anything.
[06:23:29] Scanning 11219 subdomains (subdomains.txt), please wait...
[06:24:03] [FOUND:SUBDOMAIN] beta.[redacted]com ON CLOUDFLARE NETWORK!
[06:26:29] [FOUND:SUBDOMAIN] live.[redacted].com ON CLOUDFLARE NETWORK!
[06:26:45] [FOUND:SUBDOMAIN] mailing.[redacted].com ON CLOUDFLARE NETWORK!
[06:29:29] [FOUND:SUBDOMAIN] tracking.[redacted] IP: 94.23.161.1[redacted] HTTP: 200
[06:30:10] [FOUND:SUBDOMAIN] www.jabburr.com ON CLOUDFLARE NETWORK!
```

Figure 6: Cloudfail identified target Origin IP

The Server IP shown is shown as 172.67.71.9x. Whois command was done on the server IP and it shows the IP belongs to Cloudflare. Furthermore, when a DNS historical data was carried out on SecruityTrails, it shows our target moved to Cloudflare 6 months ago and the origin IP was found as shown in Figure 7



[redacted] historical A data

IP Addresses	Organization	First Seen	Last Seen	Duration Seen
[redacted] [redacted]	Cloudflare, Inc.	2022-01-08 (6 months)	2022-06-27 (today)	6 months
159.65.[redacted]	DigitalOcean, LLC	2019-01-28 (3 years)	2022-01-08 (6 months)	3 years

Figure 7: SecurityTrails identified target

The IP also resolves to the same content and similar domain URL. The advantage of this method renders the WAF protection or DDOS protection futile as requests can easily go to the Origin server without passing through the WAF.

This process was also repeated for other Web Application Firewall and the results were shown in Table 1.0

WAF Provider	Identified Origin IP	Bypassed
Cloudflare	Y	Y
Amazon WAF	Y	N
Modsecurity Proxied Server	Y	Y

Table 1.0 Results of Bypassed WAF

2.3.1 Initial Implementation of WAF-VERIFIER

An initial implementation of WAF-VERIFIER (treatment), application-level used to identify if the request is coming through the WAF, was used during the study and the result is outlined in Table 2.0

WAF Provider	Identified Origin IP	Bypassed	With WAF-VERIFIER (Bypassed)
Cloudflare	Y	Y	N
Amazon WAF	Y	N	N
Modsecurity Proxied Server	Y	Y	Undecided

Table 2.0 Results of Bypassed WAF with WAF Verifier

2.4 PILOT STUDY DISCUSSION

Three Web Applications Firewalls were analyzed. During the experiment, the Origin server behind Cloudflare WAF protection could be identified and bypassed was possible using a combination of tools as shown in this study. Cloudflare among others has more dedicated tools targeted to bypass it. This is largely due to its popularity, ease of use, and other protection services such as DDOS offered to its Clients. There have been several measures suggested by Cloudflare and other security experts to mitigate this bypass, however, there has been less adoption due to the useability and technicality involved which also agrees with Voronkov's (2020) study findings on the Usability of Firewall configurations.

With Amazon WAF protection, the origin server IP can be detected but the request sent to the found IP was declined. This is a result of Amazon security groups which are automatically attached to resources when they are created which is an advantage because little or no configuration is required. The caveat is that it is only restricted to protecting Amazon services such as EC2, Cloudfront, or Elastic load balancer. This probably explains why it's less popular and adopted compared to Cloudflare.

Lastly, Modsecurity proxied server was manually configured on an Nginx server using the default enabled configurations. It applies the Firewall rules to an incoming request and accepts or rejects the request. Accepted requests are forwarded to the Apache server that hosts the web application. The origin IP of this Apache could be detected and the Firewall could be bypassed. ModSecurity Firewall bypassed have been investigated using malicious payload and was found to depend on paranoia level of the Core Rule Set (Singh et al., 2018) but not by directly accessing the origin server IP. Furthermore, generating a public certificate and private certificate between the two servers for communication was found to reject unauthorized requests hence solving the bypassed challenges (Arulkumar et. al., 2012). However, these challenges still exist for clients with little technical skills and who host their web applications with providers where hosting resources are shared. WAF-VERIFIER ensures all requests are routed via the Web Application Firewall as intended and all other requests not going through the firewall were rejected as shown in Table 2.0, therefore preventing bypass. However, with the Modescurity proxy server, the current version of WAF-VERIFIER couldn't identify the request because the proxy server didn't fingerprint the request as in the case of Cloudflare, hence further improvements will be made on WAF VERIFIER in the main project to identify such request.

2.5 PILOT STUDY LIMITATIONS

In this study, the number of analyzed WAFs was limited due to limited access to commercially available Firewalls. Some Firewall providers require an initial use case demonstration before providing access to services which is time-consuming compared to the time required to carry out this study. Secondly, the web application used in DVWA (Dam Vulnerable Web Application) was used for this study, therefore, ensuring the attack will be successful should the firewall be bypassed. In reality, an application should not be this vulnerable placing more responsibility on the developers to place their role in ensuring the security of web applications and not rely on external protection.

2.6 PILOT STUDY CONCLUSION

In this study, It was found that 75% of web applications' firewalls can be bypassed and the bypass success rate largely depends on the web application firewall configuration knowledge and access level of the administrator. The current limitation of the existing WAF was identified and agrees with what was found in literature and other projects such as OSWAP. Attackers are constantly looking for new ways to circumvent the firewalls web developers and infrastructure owners put in place. Despite Cloudflare and other Web Application Firewall providers' documentation, request verifier limitation still exists, and the implications of these limitations are abysmally understood and web owners are frequently unaware of the potential danger with the assumption that the most secure configuration is in place. In the main MSc project, the artifact - WAF-VERIFIER- will be improved to further enforce the security of the Web Application Firewall from more WAF providers.

CHAPTER 3. LITERATURE REVIEW

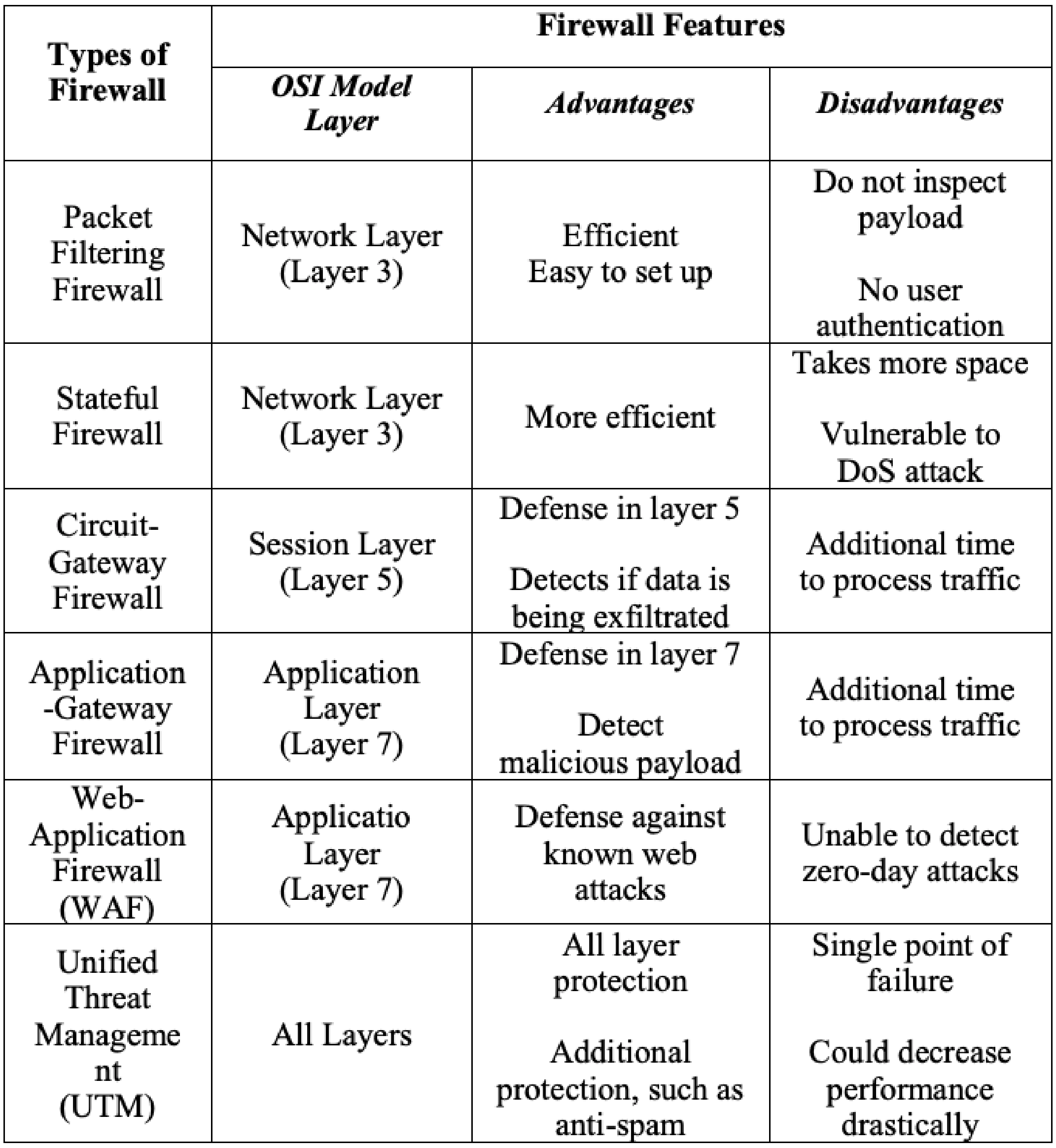
The currently available literature on Web Application Firewall covers various topics. Some researchers are exploring machine learning to secure web applications (Shaheed and Kurdy, 2022; IŞiker and SoĞukpinar, 2021) while others are focused on performance improvement of existing WAF such as changing the threat detection method from signature-based to anomaly-based (Applebaum et. al., 2021). This chapter provides an overview of existing web application threats, OSWAP Top 10 Vulnerability, Web Application Firewall mitigation mechanism, Web Application Firewall Bypass, and configuration concerns.

3.1 WEB APPLICATION FIREWALL

Firewalls are usually the first line of defense in the protection of applications against threats and malicious actors (Lawrence 2011). Originally, these firewalls inspect network traffic at Layer 3 of the Open System Interconnection but have proven insufficient due to the rapid evolution of threats. Hence, different firewalls have been invented to mitigate these threats such as Circuit-gateway Firewall, Packet Filtering Firewall, Stateful-Firewall, Web Application Firewall, and more recently Next-Generation Firewall (NGFW) (Liang and Kim, 2022).

Web Application Firewall, among others, is more specifically designed to mitigate web application threats and it has been progressively used to protect against popular attacks such as XSS and SQL Injection. Despite the fact that NGFW and Web Application Firewall (WAF) both thoroughly scan the packet's Application layer of the OSI Model, They are not interchangeable. NGFW examines every layer of the OSI model and targets the entire network, WAF simply examines the HTTP/HTTPS. WAF hence provides more targeted security for web apps.

According to Liang and Kim (2022), as shown in Figure 8, WAF is unable to detect unknown attacks (zero-day attacks) because a WAF's threat detection mechanism relies primarily on pattern classification. Huang *et. al* (2017) also supported this claim, however, Moradi *et al* (2021) opposed that WAF solution based on the anomaly detection method can detect zero-day attacks. This however explains the increasing adoption of WAF by cooperation as the WAF market $8.05 billion by 2025 (Industryarc, 2021).



Types of Firewall	**Firewall Features**		
	OSI Model Layer	***Advantages***	***Disadvantages***
Packet Filtering Firewall	Network Layer (Layer 3)	Efficient Easy to set up	Do not inspect payload No user authentication
Stateful Firewall	Network Layer (Layer 3)	More efficient	Takes more space Vulnerable to DoS attack
Circuit-Gateway Firewall	Session Layer (Layer 5)	Defense in layer 5 Detects if data is being exfiltrated	Additional time to process traffic
Application -Gateway Firewall	Application Layer (Layer 7)	Defense in layer 7 Detect malicious payload	Additional time to process traffic
Web-Application Firewall (WAF)	Applicatio Layer (Layer 7)	Defense against known web attacks	Unable to detect zero-day attacks
Unified Threat Manageme nt (UTM)	All Layers	All layer protection Additional protection, such as anti-spam	Single point of failure Could decrease performance drastically

Figure 8: Firewall Comparison (Liang and Kim, 2022)

Furthermore, Web Application Firewall is a standard compliance requirement according the Payment Card Industry Data Security Standard. According to PCI DSS (2018, p.11), the requirement 6.6 states that:

"New threats and vulnerabilities must be constantly addressed for web applications open to the Internet, and it must be ensured that these applications are protected against attacks by one of the following methods:

- Install an automated technical solution which detects and prevents attacks on the Web, such as a Web Application Firewall, in front of public web applications to control all traffic continuously”

Therefore making WAF is an essential part of Online business for compliance purposes. However, this security tool is redundant during defense mechanisms due to bypass or misconfigurations arising from shadowing or redundancy (Alicea and Alsmadi, 2021). Mitigating the WAF Bypass and easy configuration becomes imperative.

3.1.1 WAF Detection Mechanism

Payload-based Web Application Firewall analyzes the payload information of the packets. Torrano-Gimenez *et al* (2011) findings on feature selection of payload-based Web Application Firewall using CSIC HTTP data set to improve the effectiveness of Web Application Firewall. The result shows an increase in WAF performance but sacrifices the accuracy and its application is limited to WAF based on the n-gram statistical model.

Rule-based WAF approaches have the advantage of being simple to implement; nevertheless, rule-based systems have a high rate of false positives (legitimate traffic that is accidentally blocked) which leads to denial of service. For instance, Modsecurity, An open-source rule-based WAF, has a false-positive rate of 24% (Betarte *et al.*, 2018). This implies that one of every four legitimate requests is blocked. Tran *et al* (2020) findings on improving Modesecurity WAF with Machine Learning suggested a strategy that combines the machine learning models Decision Tree and Random Forest with ModSecurity, and have greatly reduced the false-positive rate of ModSecurity CRS with their trained model. Usually, security rules are reduced to a small number to prevent false positives, which decreases security coverage.

Anomaly-based detection WAF is becoming more popular because they have more granularity than rule-based WAF and theoretically, it can detect both known and zero-based attacks. According to Betarte *et al.*, (2018), the characteristics of anomaly detection are drawn from the application's expected normal behavior, and it classifies HTTP requests that differ from this behavior as attacks. As a result, specific training is required to differentiate normal traffic from malicious traffic. Moradi *et al.,* (2019) study extracted pertinent characteristics from HTTP request logs using Deep Neural Network (DNN) techniques based on the auto-encoder LSTM (Long short-term memory) model. As research in machine learning-based WAF progresses, a hybrid implementation is been considered for better security.

Tekerek and Bay (2019) investigate employing an Artificial Neural Network (ANN) to implement a hybrid WAF that combines anomaly-based detection and signature-based detection (SBD) or rule-based detection (RBD). Three features were used by the neural network-based component to digitize the incoming traffic: alphanumeric character analysis, letter frequency analysis, and request length analysis. Using the datasets from CSIC 2010, ECML-PKDD 2007, and WAF 2015, the resulting WAF was trained and evaluated. The performance and efficacy of the three-stage signature-based detection and the anomaly-based detection were measured. Performance levels for the suggested WAF are good thanks to its hybrid architecture. Prior to the slower anomaly-based phase, the quicker signature-based phases are run. The anomaly-based detection phase feeds back previous normal and abnormal requests to the signature-based phases to improve detection.

Furthermore, Prabhudesai *et al.*, (2019) study on hybrid WAF explained the protection in three phases. The first stage filters incoming request that matches well-known attack patterns using signature-based OWASP-based rules. Successful requests from this initial stage are moved on to the next stage, which makes use of an AI engine. The request is added to a knowledgebase if malicious traffic is found in the second step; otherwise, it moves on to the final phase which is IP-based. This phase is where the WAF checks popular repositories such as Virustotal to see if the IP has been reported or if it has been flagged as dangerous. When all three stages are successfully completed, the request is forwarded to the application. Cloudflare and other cloud-based WAF providers leverage the use of Hybrid WAF to provide security to their customer's assets.

3.1.2 WAF Security And Web Server Proximity

WAF implementation approaches usually depend on the WAF distance to the application it protects. Network WAF, Host-oriented WAF, and Cloud WAF are types of implementation approaches with each having its pros and cons.

Network WAF, also known as Hardware-based Web Application Firewall is deployed by installation within the Local Area Network (LAN). Due to its physical proximity, lt minimizes latency but it is more expensive and appropriate for large businesses that can afford the cost (Pentasecurity, 2020). As a result, small businesses opt for host-oriented or Cloud-based to optimize the cost-benefit of Web Application Firewall.

Host-oriented WAF solutions are more affordable compared to network WAF and offer more customization options. Because there may be several web apps running on a single web server that hosts web applications. Host-oriented WAFs are more tailored to meet applications' security requirements but their weaknesses are local server resource consumption, implementation complexity, and maintenance costs (Lakhno et al.,2022).

Cloud WAF provides the simplest implementation and reduces operational cost and offers a solution that is constantly updated to protect against new threats without additional work or cost on the customer side (Fernandez et al., 2014). Cloud WAF is a third-party liability and is often provided as Software-as-a-Service (SaaS). One of the most popular clouds web application firewalls is Cloudflare WAF.

Whatever approach has been employed to implement WAF, it is evident that any web application firewall (WAF) must also constantly evolve and gather the information necessary to defend against application layer threats. The distance between Cloud WAF and the server is usually exploited to bypass the firewall by direct IP access. The details of this WAF bypass are presented in the next section.

3.1.4 Configuration and Bypass Concerns

Firstly, a popular WAF bypass method is warping the request payload. This usually occurs by encoding the payload request in hexadecimal or decimal making it difficult for the WAF to understand leading to bypass (Qiu 2013). This encoding is usually understood by the browser leading to an XSS attack or SQLi attack.

Direct Blocked URL

http://www.testwebsite.com/pages.php?id=.90 union select
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23

Encoded Bypassed URL

http://www.testwebsite.com/pages.php?page_id=%2e%30%
75%4e%49%4f%6e%28%73%65%6c%65%63%74%20%31%2
c%32%2c%33%2c%34%2c%35%2c%36%2c%37%2c%38%2c%
39%2c%31%30%2c%31%31%2c%31%32%29

Furthermore, algorithms such as DQN have also been used to deform XSS payload to bypass rule-based WAF which leads to an effect known as the anti-anti-virus effect (Li *et al,* 2019). This method of the bypass is mainly associated with rule-based or signature-based WAF which explains why many organizations are leveraging Cloud-based WAF that uses a combination of rule-based and Machine Learning approaches to detect vulnerabilities.

Lastly, origin exposure is another WAF bypass method in which the adversary directly targets the origin server using the IP address (Jin *et al.,* 2018). Origin servers with static IPs are susceptible to this type of exposure. Origin IPs can be directly accessed when a Cloud-based WAF provider is down as a result of service maintenance (Vissers *et al.*, 2015) or by searching DNS repositories such as SecurityTrails for the history of origin domain names and their IPs. Suggestions have been made to mitigate this bypass by changing origin IP addresses on demand (Jia *et al.,* 2014) and port forwarding at the edge (Vissers *et al.,* 2015) or by whitelisting incoming traffic to the origin servers.

3.2 WEB APPLICATION THREATS

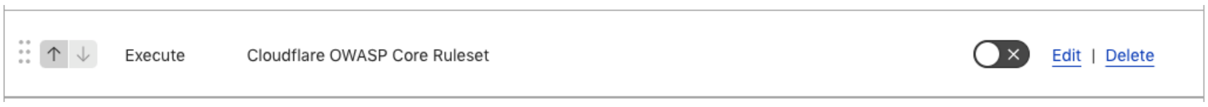
One of the most prevalent types of cyberattacks is to target web apps. According to IBM Security X-Force 2022 reports, 11% of all incidents are server access attacks and it is the second-most common attack. Vulnerability exploitation such as XSS, and SQLi increases by 33% from 2020 indicating the importance of this attack vector (IBM 2022 p.14). The Open Web Application Security Project (OWASP) community provided a data-driven awareness document to minimize web application security risks highlighting the top ten most critical web application vulnerabilities. The most recent version that obtained acceptance at the start of this project is OWASP top 10 2021 (OWASP Foundation, 2021). The 2021 rank is listed in Table 3

A01:2021 - Broken Access Control
A02:2021 - Cryptographic Failures
A03:2021 - Injection
A04:2021 - Insecure Design
A05:2021 - Security Misconfiguration
A06:2021 - Vulnerable and Outdated Components
A07:2021 - Identification and Authentication Failures
A08:2021 - Software and Data Integrity Failures
A09:2021 - Security and Monitoring Failures
A10:2021 - Server-Side Request Forgery (SSRF)

Table 3: OWASP Top 10 2021 (OWASP Foundation, 2021)

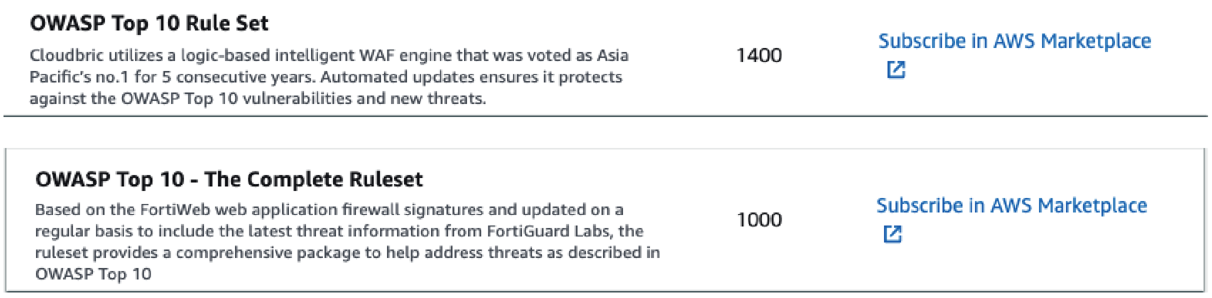
OWASP 2017 vulnerabiltiies still exists, however, OWASP Top 10 2021 has been expanded and therefore contain three new vulnerabilities; Insecure Design, Software, and Data Integrity Failures and Server-Side Request Forgery (SSRF). This is a result of web application development changes in the last couple of years with new architectures layouts such as

Microservices, RESTFUL APIs, and Single Page Applications (Sϕhoel *et al.,* 2018). Most Web Application Firewall providers have built-in security rule implementation based on OSWAP documents and are constantly updated based on the latest version available from the official repository. Figure 9 and 10 shows OWASP integration in Cloudflare and Amazon Firewall respectively.



Execute Cloudflare OWASP Core Ruleset Edit | Delete

Figure 9 OWASP in Cloudflare WAF Managed rules



OWASP Top 10 Rule Set Cloudbric utilizes a logic-based intelligent WAF engine that was voted as Asia Pacific's no.1 for 5 consecutive years. Automated updates ensures it protects against the OWASP Top 10 vulnerabilities and new threats.	1400	Subscribe in AWS Marketplace
OWASP Top 10 - The Complete Ruleset Based on the FortiWeb web application firewall signatures and updated on a regular basis to include the latest threat information from FortiGuard Labs, the ruleset provides a comprehensive package to help address threats as described in OWASP Top 10	1000	Subscribe in AWS Marketplace

Figure 10 OWASP Top 10 in Amazon WAF Managed rules

The continuing subsections consist of individual sections of the OWASP Top 10 explained in depth.

3.2.1 Broken Access Control

It is ranked first and appears to be most crucial due to the adverse implication such as escalation of privilege which threatens the confidentiality, integrity, and availability of data which are the three important properties of data according to the CIA triad in Figure 11.

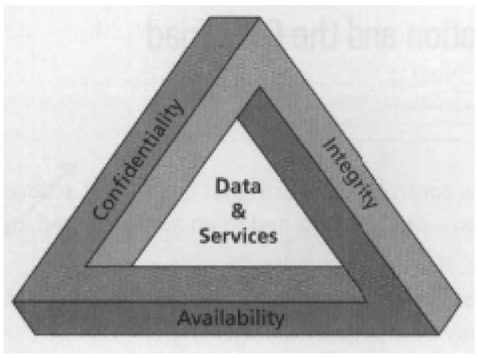
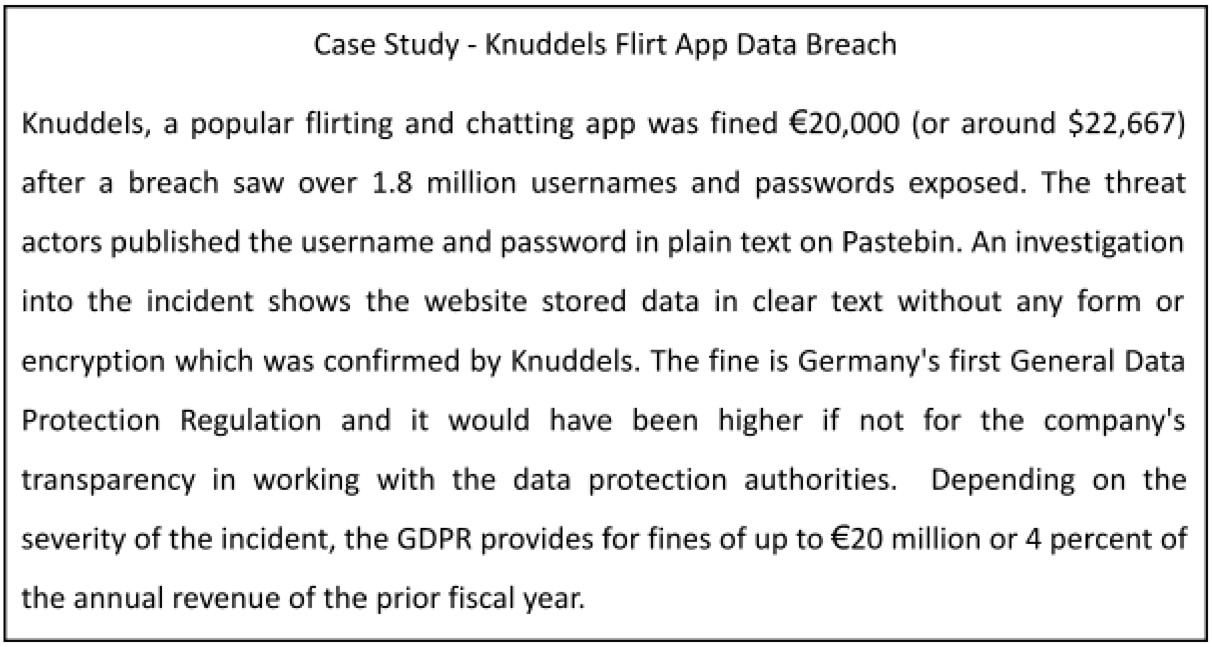


Figure 11 The C.I.A triad (Whitman and Mattord, 2017)

BAC vulnerability is a negation of properties associated with individuals who access data such as Identification, Authentication, Authorization, and Accountability (Kumar *et al.,* 2018). Hassan et al (2018) examinations on the quantitative assessment of Broken Access Control (BAC), session misconfiguration, and inadequate validation of input are the major factors contributing to BAC vulnerability. BAC Notable Common Weakness Enumerations (CWEs) include CWE-22, CWE-276, CWE-497, and CWE-601 among others.

3.2.2 Cryptographic Failures

Formerly known as sensitive data exposure in the OWASP 2017 occurs due to weak or nonexistent encryption algorithm that leads to exposes sensitive application data such as credit cards, personal health records, passwords, and invalidated server certificates, among others (Disawal and Suman, 2021). Popular Notable Common Weakness CWEs is CWE-259: Use of Hard-coded Password (OWASP Foundation, 2021). This vulnerability is not only a security issue, by extension, it can also be a data privacy issue as in the case of Knuddels Flirt App Data breach (Figure 12) whose breach leads to a General Data Protection Regulation (GDPR) of $22,000 (Seals, 2018).



Case Study - Knuddels Flirt App Data Breach

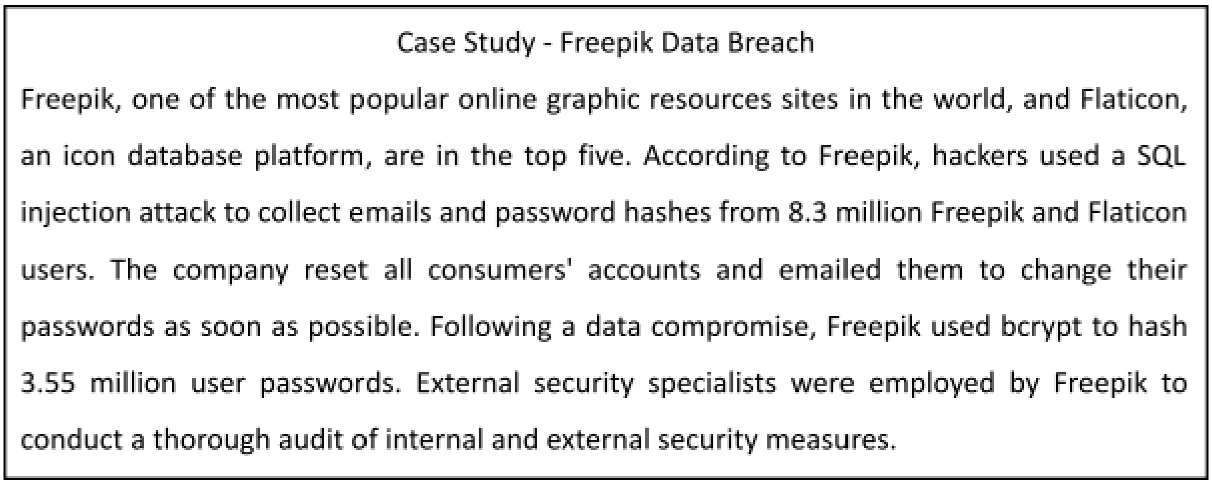
Knuddels, a popular flirting and chatting app was fined €20,000 (or around $22,667) after a breach saw over 1.8 million usernames and passwords exposed. The threat actors published the username and password in plain text on Pastebin. An investigation into the incident shows the website stored data in clear text without any form or encryption which was confirmed by Knuddels. The fine is Germany's first General Data Protection Regulation and it would have been higher if not for the company's transparency in working with the data protection authorities. Depending on the severity of the incident, the GDPR provides for fines of up to €20 million or 4 percent of the annual revenue of the prior fiscal year.

Figure 12 Knuddels Flirt App Data Breach (Seals, 2018)

Web Application Firewall protection mechanism in combating Cryptographic failures includes enforcement of encryption directive for data in transit such as HTTP Strict Transport Security (HSTS) (OWASP Foundation, 2021). The Cryptographic Failures protection mechanism is a shared responsibility on web developer end to protect data at rest by using the appropriate encryption method,salting password while Web Application Firewall along with the web server protects data in transit such as enforecent of TLS.

3.2.3 Injection

Injection attack occurrs when application blind trusts user input, as a result, an attacker sends malicious data to the application to process it therefore leading to system execution flow (Djeki *et al.,* 2022). Popular Notable Common Weakness Enumeratons are CWE-79 and CWE-89 which are Cross-site Scripting and SQL Injection respectively. Freepik, was a recent victim of SQL Injection vulnerability leading to data breach of 8.3 Million records (Figure 13) (Gatlan 2020).



Case Study - Freepik Data Breach

Freepik, one of the most popular online graphic resources sites in the world, and Flaticon, an icon database platform, are in the top five. According to Freepik, hackers used a SQL injection attack to collect emails and password hashes from 8.3 million Freepik and Flaticon users. The company reset all consumers' accounts and emailed them to change their passwords as soon as possible. Following a data compromise, Freepik used bcrypt to hash 3.55 million user passwords. External security specialists were employed by Freepik to conduct a thorough audit of internal and external security measures.

Figure 13: Freepik Data Breach (Gatlan, 2020)

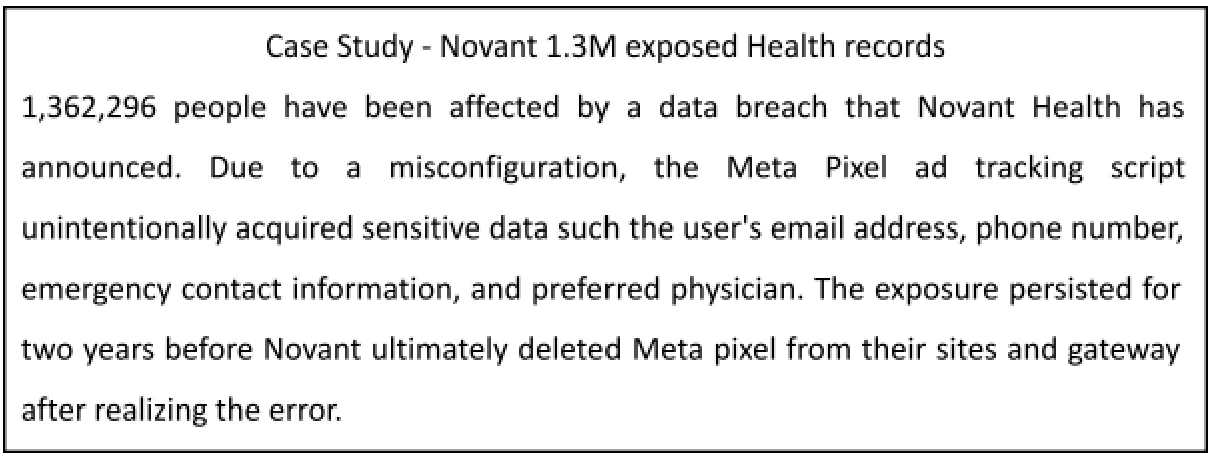
Alsobhi and Alshareef, (2020) suggest Input sanitization, and separation of data from commands to mitigate against injection vulnerability. Additionally, Web Application Firewalls have always been an effective tool in combating this form of attack and are often recommended (PCI DSS, 2018, p.11).

3.2.4 Insecure Design

This new category involves architectural flaws in the application development such as showing error messages containing code line number to users, question and answer based credential recovery workflow (OWASP, 2021). Furthermore, OWASP differentiates insecure design from insecure implementation stating that perfect implementation can not fix insecure design. Additionally, lack of risk assessment such as application level of exposure, before application development is a major contributing factor to this vulnerability. Web Application Firewall mitigate against this vulnerability by implementing Bot Defense, Limit resource consumption by service, threat modeling (F5 Operation Guide, 2022).

3.2.5 Security Misconfiguration

Security misconfiguration occurs when security standards, protocols, and controls are not implemented in a system as required such as insecure default configurations (Fredj *et al.*, 2020). An attacker could exploit security misconfiguration and the severity depends on the location of the misconfiguration. A misconfiguration by Novant Health (Figure 14), An Health care provider led to an exposure of 1.3M patients health records (Toulas, 2022).



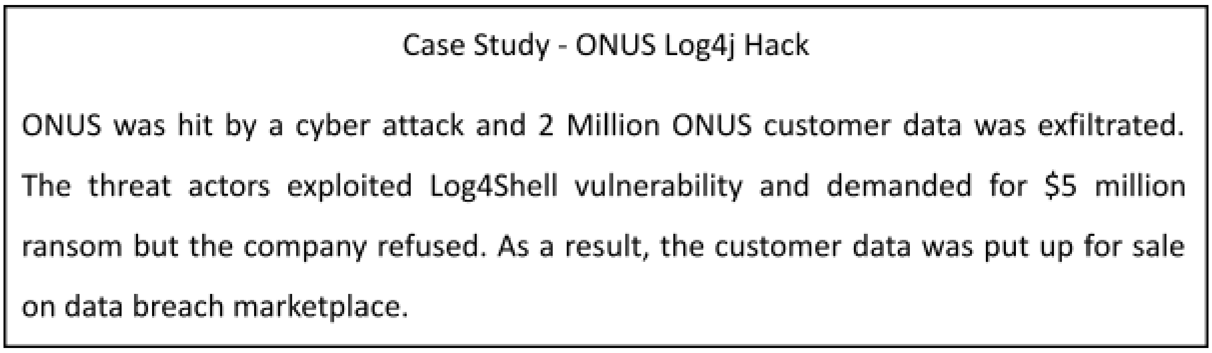
Case Study - Novant 1.3M exposed Health records

1,362,296 people have been affected by a data breach that Novant Health has announced. Due to a misconfiguration, the Meta Pixel ad tracking script unintentionally acquired sensitive data such the user's email address, phone number, emergency contact information, and preferred physician. The exposure persisted for two years before Novant ultimately deleted Meta pixel from their sites and gateway after realizing the error.

Figure 14: Novant 1.3M exposed Health records (Toulas, 2022)

3.2.6 Vulnerable and Outdated Components

Web Applications don’t run in isolation, it depends on various components such as runtime environments, libraries, APIs, and Databases. If any part of this component is outdated or not patched, it can make the application vulnerable. An example is the remote code execution, Log4j , CVE-2021-44228 – Vulnerability in Apache Log4j Library which ranks among the top 2 most exploited vulnerability (IBM 2022 p.23). ONUS, A Vietnamese crypto trading platform suffered a cyber as a result of running a payment software with Log4j vulnerability (Figure 15). Sensitive data has been exfiltrated before ONUS could patch their system with attackers demanding a $5 Million ransom (Sharma 2021).


Case Study - ONUS Log4j Hack

ONUS was hit by a cyber attack and 2 Million ONUS customer data was exfiltrated. The threat actors exploited Log4Shell vulnerability and demanded for $5 million ransom but the company refused. As a result, the customer data was put up for sale on data breach marketplace.

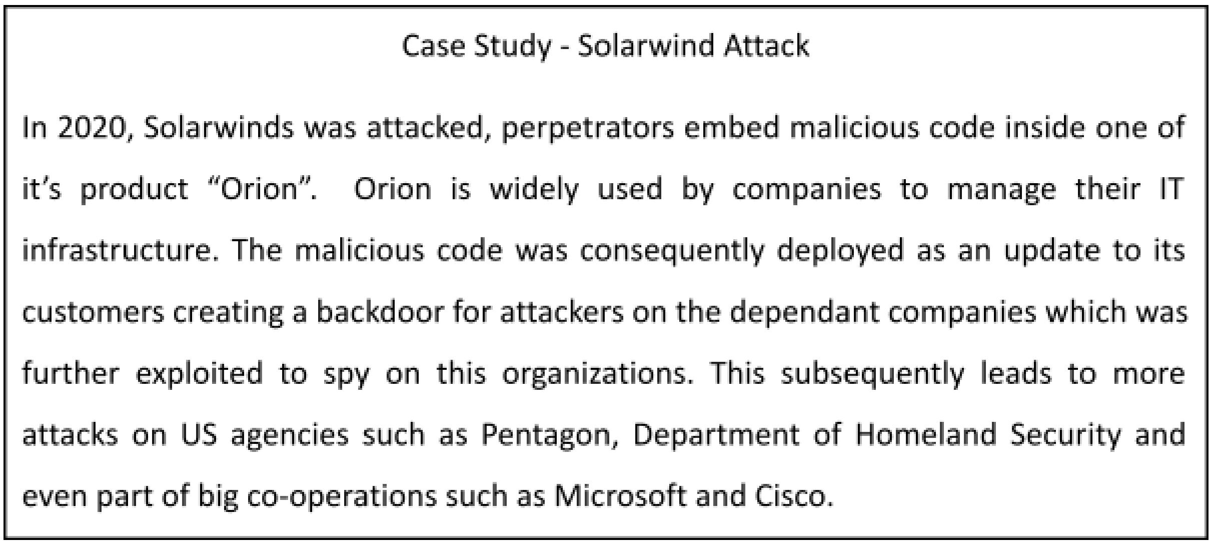
Figure 15: ONUS Log4j Hack (Sharma 2021)

3.2.7 Identification and authentication failures

Formerly Broken Authentication occurs as a result of failures in the implementation or level of protection provided by an application's identity, authentication, or session management capabilities (Fredj et. al., 2020). Multifactor Aut2hentication, and Single sign-on (SSO) are usually the deployed defense strategy (Djeki *et al.,* 2022). Web Application Firewall protection mechanism against this form of attack by implementing cookie encryption,CSRF protection and rate limiting to prevent an automated attack such as dictionary attack and brute-force attack.

3.2.8 Software and Data Integrity Failures

According to Rahman and Tomar (2020), Lack of integrity verification of application dependencies from thirdparty providers such open source libraries are major sources of risks. These libraries and services are usually used to perform CI/CD Pipelines or process application data saving developer time but can also been a source of vulnerability such as the CWE-829 - Inclusion of Functionality from Untrusted Control Sphere (OWASP 2021). Figure 16 gives case study of Solarwinds attack that makes 33,000 customer IT systems vulnerable (Jibilian and Canales, 2021).



Case Study - Solarwind Attack

In 2020, Solarwinds was attacked, perpetrators embed malicious code inside one of it's product "Orion". Orion is widely used by companies to manage their IT infrastructure. The malicious code was consequently deployed as an update to its customers creating a backdoor for attackers on the dependant companies which was further exploited to spy on this organizations. This subsequently leads to more attacks on US agencies such as Pentagon, Department of Homeland Security and even part of big co-operations such as Microsoft and Cisco.

Figure 16 : Solarwind attack (Jibilian and Canales, 2021)

3.2.9 Security Logging and Monitoring Failures

It is impossible for an application to be 100% secure, as a result, monitoring applications for failed login attempts, input validation failures are essential for administrators to identify incidents before they finally result into a breach (He *et al.,* 2021). Building a reliable application necessitate good logging and monitoring practices. Web Applications Firewalls play an essential role in Logging blocked events for further actions if required by the administrators. Furthermore, OWASP recommends adoption of NIST incident response and recovery plan (OWASP, 2021).

3.2.10 Server-Side Request Forgery (SSRF)

This is a flaw web-based applications that allows attackers to send specially crafted requests to servers to compromise services accessed from restricted internal resources or external networks (OWASP 2021). Protection mechanism usuall include the use of VPN,WAF, or other network access control list (ACL) (Arora *et al.*, 2021). Due to the fact that modern hackers have the payload lists, tools, and expertise to get around regular expressions and denial lists, OWASP advises against employing them to reduce SSRF.

A report by Bach-Nutman (2020) showed how the Top 10 OWASP vulnerability impacts businesses and web application users and therefore suggested a reduction of attack surface area by monitoring and addressing known vulnerabilities. In the later chapters, this project suggests the deployment WAF-VERIFIER serves as a monitoring tool for web developers to mitigate and control this bypass.

3.3 LITERATURE REVIEW LIMITATIONS

The literature review gave an expository of what have been published on the subject. Databases including (ScienceDirect, IEEE, Googlescholar) was used using the chosen literature subject keywords and the results of the search were carefully examined and reported in accordance with the PRISMA criteria (Moher et al. 2009). Systematic literature review has many advantages, it also has its disadvantages such as attrition bias. Futhermore., this literature discussed web application vulnerablities based on OSWAP top 10 and how Web Application Firewall protects against these vulnerabilities. However, web applications consist of more vulnerabilities than those highlighted in OWASP Top 10.

3.4 LITERATURE SUMMARY

In summary, existing web application firewalls protection mechanism and web vulnerabilities has been examined. WAF is not only a security tool, but also PCI DSS requirement for financial facing web application. This literature review discuss the threats of web applications vulnerabilities according OWASP top 10 and with recent case study and the implication of these threats. As a result, Web Application adminstrators are deploying managed WAF to mitigate against threats. However, Configuration and bypass concerns are major factors affecting the effectives of WAF and not surprisingly, this challenge is on the top five (5), Security configuration, in the OWASP Top 10. The application of WAF have been used enhance the security posture of web application as it is usually recommended as a defensive method with Cloud-based WAF increasingly adopted to curtail the ever growing threat landscape plaguing public-facing web applications.

CHAPTER 4. METHODOLOGY

In continuity with Chapter 2, this chapter contains the method used to achieve the purpose of this study. Thus it outlines the method used to analyze the performance of WAF, evaluate the potency of Cloud-based WAF and illustrate and mitigate WAF bypass challenges. This method. The research employed a quantitative approach which best supports the research aim objectives. The pre-experimentation method was used in the Pilot study as illustrated in 2.2.1, however, the experimental method is used to emphasize on using performance testing tools, bypassing WAFs, and mitigating the WAF bypass using the developed artefact.

4.1 EXPERIMENTAL SETUP

The test bed environment was carried out on a cloud VPS. The hardware specification is detailed on Table 4.

No	**Device**	**Specification**
1	Cloud Provider	DigitalOcean
2	CPU	1 vCPU
3	RAM	1 GB
4	Storage	25 GB NVme SSDs

Table 4 : VPS/Hardware specification

The software specification used in this study is detailed in Table 5.

No	**Device**	**Specification**
1	OS	Ubuntu 22.04 LTS (GNU/Linux 5.15.0-41-generic x86_64)
2	Apache	Apache/2.4.52
3	PHP	PHP/8.1.2
4	Database	MYSQLi/8.1.2

Table 5 : Software specification

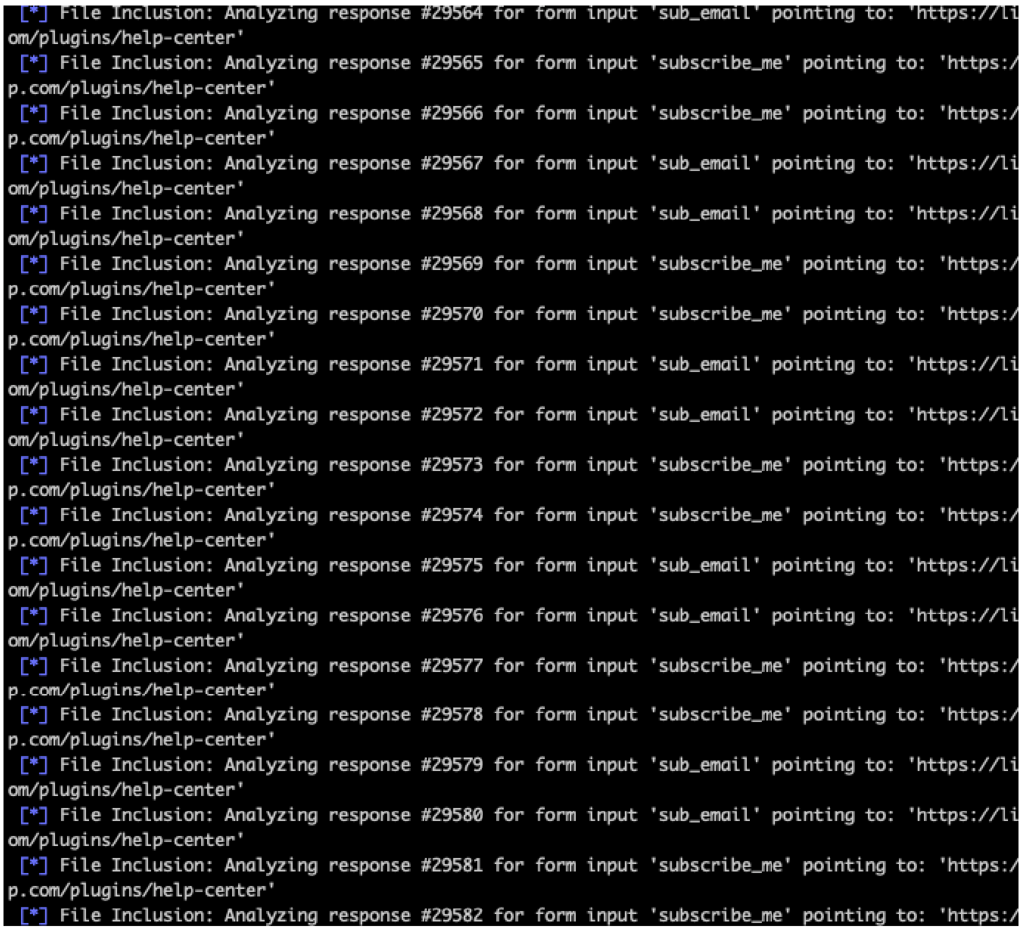
In the pilot study, Modsecurity was deployed and proxied to provide protection to web application. Trustwave, the company Modsecurity (the most popular open source WAF), is announcing its end of life (Trustwave 2021). In order to overcome this limitation and ensure the result of these findings is helpful beyond the EOL of Modsecurity, Cloudflare WAF will be explicitly used in this study. A web application basic Create, Read, Update and Delete (CRUD) was deployed on the web server.

In other to experiment and get the desired data needed, a wide variety of tools are used. The following subsections illustrated the crucial tools used in this study to get primary data and for analysis:

4.2 SOFTWARE AND TOOLS

4.2.1 Arachni

Arachni is one of the tools recommended by OWASP for website application a robust, modular, high-performance Ruby framework created with the intention of assisting penetration testers and administrators in assessing the security of contemporary web applications. It is able to transverse and learn by following the web application behavior and also perform meta-analysis by avoiding false positives. It is unrestricted and publicly available on GitHub anyone can examine and contribute to the source code. It is cross-platform and works with all major operating systems, including Linux, Mac OS X, and Microsoft Windows (Arachni 2022). Portable packages are used for distribution, enabling quick deployment. It is flexible enough to handle a variety of use cases, from a straightforward command-line scanner utility to a worldwide high-performance grid of scanners, from a Ruby library allowing for automated audits with adjustable concurrencies and the ability to detect server health automatically.



```
 [*] File Inclusion: Analyzing response #29564 for form input 'sub_email' pointing to: 'https://li
om/plugins/help-center'
 [*] File Inclusion: Analyzing response #29565 for form input 'subscribe_me' pointing to: 'https:/
p.com/plugins/help-center'
 [*] File Inclusion: Analyzing response #29566 for form input 'subscribe_me' pointing to: 'https:/
p.com/plugins/help-center'
 [*] File Inclusion: Analyzing response #29567 for form input 'sub_email' pointing to: 'https://li
om/plugins/help-center'
 [*] File Inclusion: Analyzing response #29568 for form input 'sub_email' pointing to: 'https://li
om/plugins/help-center'
 [*] File Inclusion: Analyzing response #29569 for form input 'subscribe_me' pointing to: 'https:/
p.com/plugins/help-center'
 [*] File Inclusion: Analyzing response #29570 for form input 'subscribe_me' pointing to: 'https:/
p.com/plugins/help-center'
 [*] File Inclusion: Analyzing response #29571 for form input 'sub_email' pointing to: 'https://li
om/plugins/help-center'
 [*] File Inclusion: Analyzing response #29572 for form input 'sub_email' pointing to: 'https://li
om/plugins/help-center'
 [*] File Inclusion: Analyzing response #29573 for form input 'subscribe_me' pointing to: 'https:/
p.com/plugins/help-center'
 [*] File Inclusion: Analyzing response #29574 for form input 'subscribe_me' pointing to: 'https:/
p.com/plugins/help-center'
 [*] File Inclusion: Analyzing response #29575 for form input 'sub_email' pointing to: 'https://li
om/plugins/help-center'
 [*] File Inclusion: Analyzing response #29576 for form input 'sub_email' pointing to: 'https://li
om/plugins/help-center'
 [*] File Inclusion: Analyzing response #29577 for form input 'subscribe_me' pointing to: 'https:/
p.com/plugins/help-center'
 [*] File Inclusion: Analyzing response #29578 for form input 'subscribe_me' pointing to: 'https:/
p.com/plugins/help-center'
 [*] File Inclusion: Analyzing response #29579 for form input 'sub_email' pointing to: 'https://li
om/plugins/help-center'
 [*] File Inclusion: Analyzing response #29580 for form input 'sub_email' pointing to: 'https://li
om/plugins/help-center'
 [*] File Inclusion: Analyzing response #29581 for form input 'subscribe_me' pointing to: 'https:/
p.com/plugins/help-center'
 [*] File Inclusion: Analyzing response #29582 for form input 'subscribe_me' pointing to: 'https:/
```

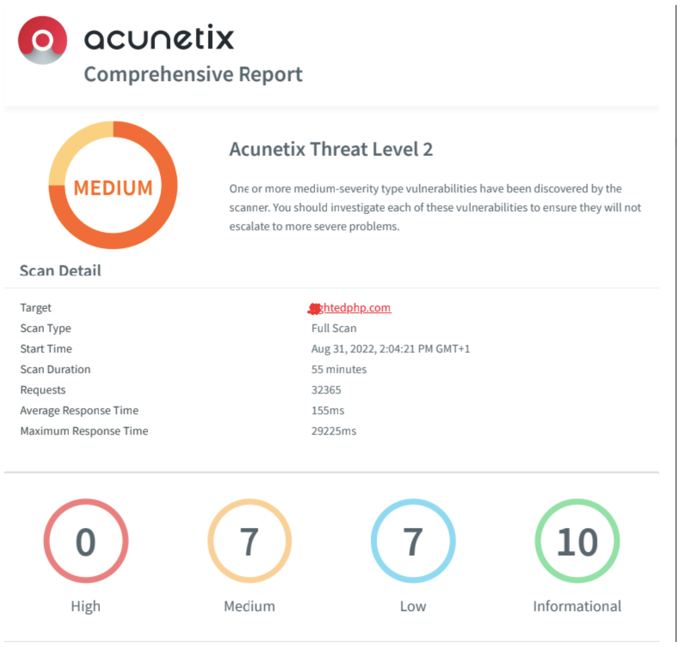
Figure 17: Arachni command on target

Arachni version 1.6.1.3 was used in this study to enumerate the web application vulnerabilities and investigate the role of WAF to mitigate these vulnerabilities. Figure 17 shows the target enumeration and the result is reported and analyzed and compared with a similar tool called Acunetix.

4.2.2 Acunetix

With cutting-edge technologies, Acunetix is a leader in automated web application security testing and DAST. The term Dynamic Application Security Testing (DAST) Tool is widely used to describe this group of technologies that look for security flaws in applications such as Command Injection, SQL Injection, Insecure design, and Cross-Site Request Forgery. Acunextix is a multithreaded, incredibly quick crawler and scanner that can continuously crawl millions of pages. It contains an easy-to-use Login Sequence Recorder that enables the automatic scanning of intricate password-protected sections and has the highest identification of WordPress vulnerabilities which is a PHP-based web application.

Consequently, the target machine in this study is also PHP based application making Acunetix a more suitable vulnerability scanner. It contains a vulnerability management feature that is built in and can produce a wide range of technical and compliance reports as shown in Figure 18.



acunetix
Comprehensive Report

MEDIUM

Acunetix Threat Level 2

One or more medium-severity type vulnerabilities have been discovered by the scanner. You should investigate each of these vulnerabilities to ensure they will not escalate to more severe problems.

Scan Detail

Target	[redacted]htedphp.com
Scan Type	Full Scan
Start Time	Aug 31, 2022, 2:04:21 PM GMT+1
Scan Duration	55 minutes
Requests	32365
Average Response Time	155ms
Maximum Response Time	29225ms

High	Medium	Low	Informational
0	7	7	10

Figure 18: Acunetix report on target machine

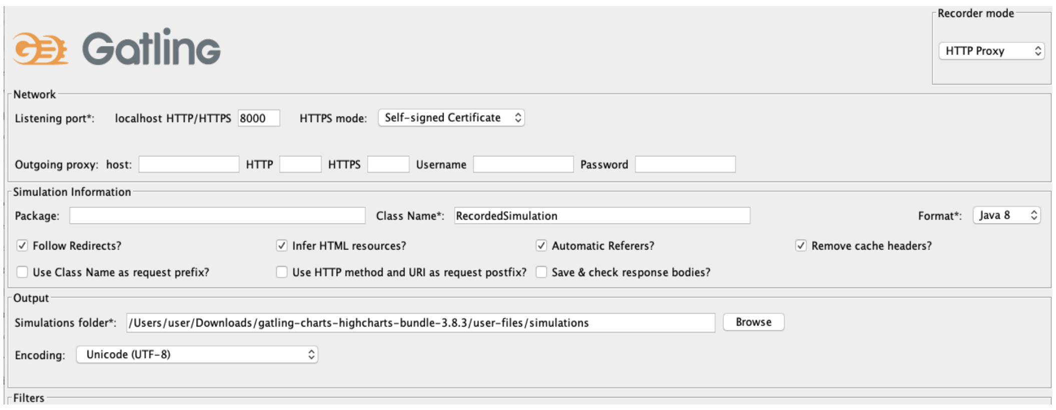
The result of Acunetix was extensively discussed in the later chapter and compared with Arachni results and the role of Cloudflare on these results.

4.2.3 Jmeter

The Apache JMeter program, which is 100% pure Java and open source software, is used to load test functional functionality and track performance. It can be used to simulate a huge pressure on a server, network, or object in order to test its strength or examine overall performance under various load conditions. Based on the results of JMeter performance testing, system developers must continue their investigation and identify the primary causes of system performance flaws because there are differences between the testing environment and the production environment and the network environment is complex (Wang and Wu, 2019). As a result, the Gatling tool was further used to test the performance of the target machine, and a comparative analysis was carried out.

4.2.4 Gatling

It is a free open-source performance testing tool for web applications with support for HTTP, WebSocket, Server-Sent Events, and JMS. Gatling offers a simple GUI that is solely available for the test recorder as shown in Figure 19. Gatling does, however, offer a method for developing test cases in a manner that is legible and writeable using a domain-specific language. This method is used in this study. Furthermore, it has a powerful validation system. It was used to generate using an asynchronous non-blocking approach and provides a clean report and visualization which is its edge over other performance testing tools (Shrivastava and Prapulla, 2020).


Recorder mode: HTTP Proxy

Network
Listening port*: localhost HTTP/HTTPS 8000 HTTPS mode: Self-signed Certificate
Outgoing proxy: host: HTTP HTTPS Username Password

Simulation Information
Package: Class Name*: RecordedSimulation Format*: Java 8
✓ Follow Redirects? ✓ Infer HTML resources? ✓ Automatic Referers? ✓ Remove cache headers?
☐ Use Class Name as request prefix? ☐ Use HTTP method and URI as request postfix? ☐ Save & check response bodies?

Output
Simulations folder*: /Users/user/Downloads/gatling-charts-highcharts-bundle-3.8.3/user-files/simulations
Encoding: Unicode (UTF-8)

Filters

Figure 19: Gatling GUI

4.2.5 Apache Bench Tool

Apache Bench (ab) is a utility for evaluating the performance of the Apache HTTP server. Its purpose is to provide an impression of the functionality of Apache server installation. It shows how many requests per second the web application can handle. It was used as the third performance enumeration tool in this study and the results were recorded and analyzed.

4.3 DATA COLLECTION METHODS

The primary data was collected during the enumeration of the vulnerability of the target web application and performance analysis using Gotling and Jmeter. Prior to the selection of the tools, a search was carried out. Firstly across the wide internet to simulate how web administrators would for such tool primary using Google search. The second search was on academic databases such as Google scholar and ScienceDirect to discover tools used by other security researchers. After finding every tool in various databases, they were all filtered using a set of inclusion criteria as outlined below.

- A tool should perform Dynamic application security testing (DAST)
- The tool should be recommended by OWASP
- The tool should have active development and be updated regularly.
- The tool should have more than a developer behind it to ensure its wide coverage and functionality.

Applicable methods were extensively discussed in Chapter 2.2.1. Due to the practical nature of web vulnerabilities and WAF, the Experimental method is found to be more appropriate for this study, hence it was used. A quantitative approach was used and the results were compared with those found in other literature to ensure reliability for adoption as a reference for future research and a guide for web administrators and web developers to mitigate risks associated with web application vulnerabilities. Furthermore, the Web Application Firewall logs, Cloudflare, while the analysis was ongoing the details of the logs were monitored.

4.4 DESIGN OF ARTEFACT

Prior to the implementation of the artefact, WAF-VERIFIER, as shown in Figure 20. It was designed and made use of both automatic and manual mechanisms (which can be set in the UI interface) depending on the web administrator configuration; it implements a fine-grained application rule-set that defines the allowed IPs within an application. Once this set is defined, it is virtually impossible for an attacker to assess the web application directly.

Components of WAF VERIFIER

- Extraction of data from requests e.g IP
- Extraction of metadata from the request such as server name and other fingerprints made by the WAF.
- Request Logs
- Request Validation

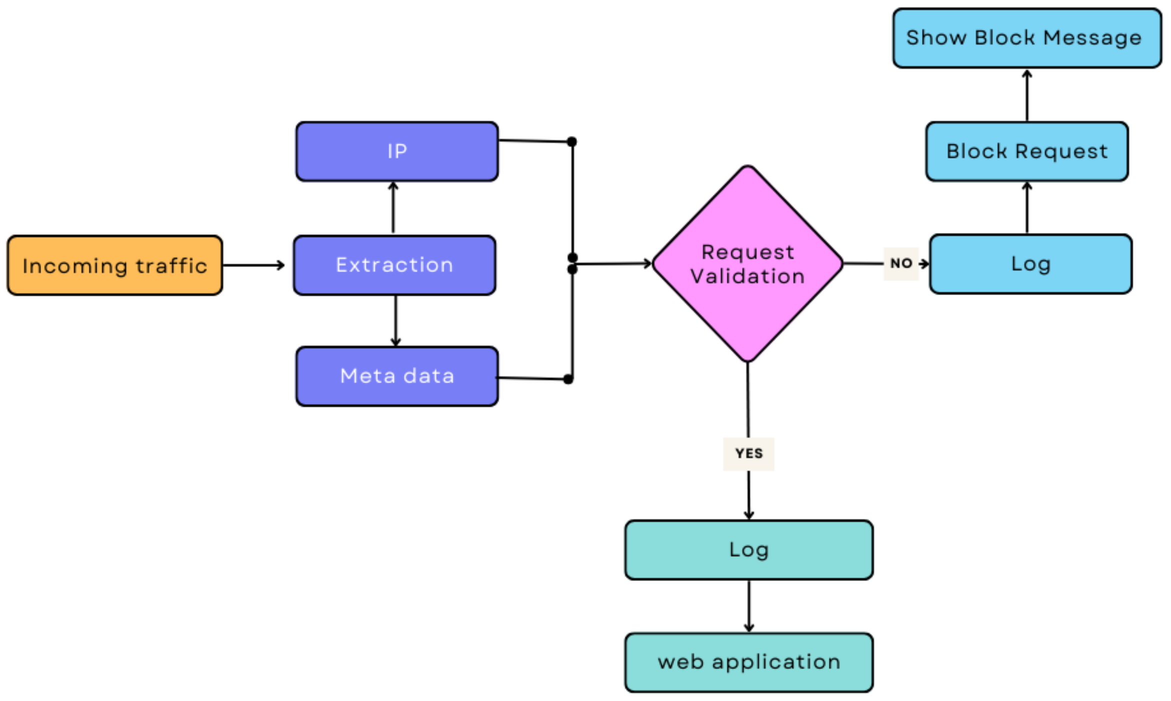


Figure 20: WAF VERIFIER Design Flow Diagram

After the design and writing of the code, WAF VERIFIER is then deployed on the target web application to protect the website and evaluate its capabilities in practice.

4.5 EXPERIMENTS

4.5.1 Experiment 1

To enumerate the role of the Web Application Firewall on the Target System, it becomes imperative to analyze the vulnerability nature of the system. A vulnerability assessment of the target system was carried out using the Arachni tool. The steps followed as shown in Figure 21. Cloudflare WAF was also monitored to see the Blocking rate and the results were reported.

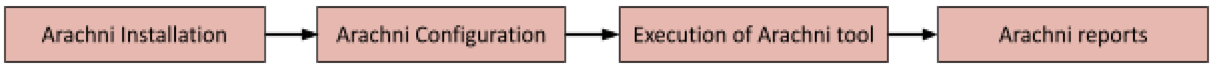


Figure 21: Experiment 1 procedure

4.5.2 Experiment 2

As established by various literature as outlined in the literature review, different vulnerability tools have their strength and weakness. Similar to Experiment 1, a second vulnerability scanning was carried out using Acunectix following outlined steps as shown in Figure 22 below. Manual testing was also carried out to check the veracity of the tool.

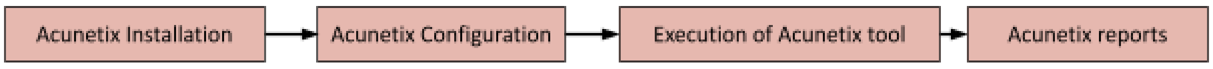


Figure 22: Experiment 2 procedure

4.5.3 Experiment 3

In order to achieve one of the objectives of this study which is to illustrate WAF bypass and mitigation. In the pilot study, the bypass was carried out on the range of configured WAF as shown in Figure 2. This experiment is to illustrate the extent of protection provided by the developed Artefact in this project. The artefact is designed in Figure 21 and attempts to validate the request before it gets to the web application as shown in Figure 22. The result was documented and analyzed in the later chapter of this project.

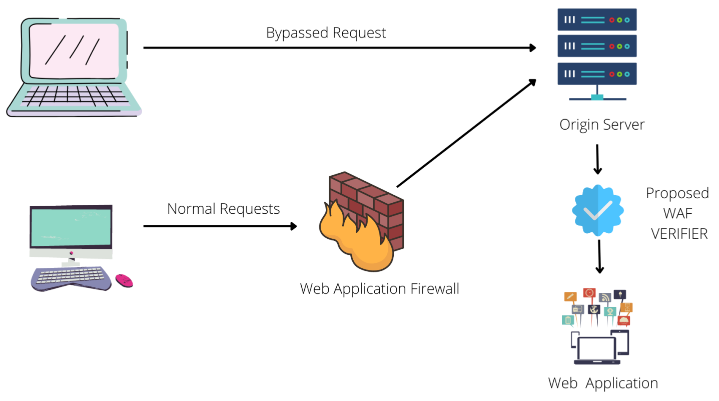


Figure 23: WAF Bypass with WAF-VERIFIER implementation topology

4.5.3 Experiment 4

Performance analysis of WAF was carried out using Jmeter and Gatling tools. Simulated requests are sent concurrently to the server. The WAF was configured such that OWASP ModSecurity Core Rule Set was set sensitivity High. This sensitivity was toggled between Low, Medium, and High. Furthermore, concurrent users were set to 100,200, to 1000 concurrent users, and the server response was monitored. Bypassed requests were also made in the target system and using the same number of concurrent users and the requests were analyzed.

4.6 CHOSEN METHODS LIMITATION

Experimental research method was selected for this study. Experimental makes it possible to determine the cause and effect relationship, hence it was used to determine the cause and effect relation between WAF and web vulnerability. Hence making it a more appropriate method for this study. Due to simulations associated with the experimental research method, some researchers suggest it creates an artificial approach (Kothari and Gaurav 2019, p.33) which makes the findings less appropriate in the real world (Sørensen *et al.,* 2010), however, this study's experimental environment and its design imitate an ideal web application protected a real Web Application Firewall, as closely as possible, making the findings in this research more reliable and applicable.

4.7 ETHICS

This MSc research project received ethical approval from Solent University as shown in Appendix (I). This research does not require public participation as a result, participants' rights or privacy were not examined.

4.8 PROJECT MANAGEMENT

The task that must be completed as part of this project have been identified, together with a timetable for each subtask, in order for the project to be executed efficiently. Time management and organization are crucial because this project will take 17,000 words to complete. The Gantt diagram (Appendix II) shows how the time was shared across each task with an estimated timeline.

CHAPTER 5. RESULTS, ANALYSIS, AND DISCUSSION OF FINDINGS

The outcomes of the experiments are used to derive results in this study. These results were plotted into graphs to provide a proper visual grasp of the results.

5.1 EXPERIMENT 1

Following the first vulnerability scan of the target system using Acunectix, Table 6 shows the issues and severity in the target web application.

Issues	**Severity**	**OSWAP Top 10**
Application Error Messages	Medium	✔ (A09)
Directory Listings		✔ (A05)
PHP allow_url_fopen enabled		✔ (A05)
PHP open_basedir is not set		✔ (A05)
PHPinfo page		×
Vulnerable Javascript Libraries		✔ (A08)
Clickjacking: X-Frame-Options header missing	Low	✔ (A04)
Cookies with missing, inconsistent or contradictory properties		✔ (A04)
Cookies without Secure flag set		✔ (A04)
HTTP Strict Transport Security (HSTS) not implemented		✔ (A05)
Insecure inline Frame (iframe)		✔ (A04)
Login page password guessing attack		✔ (A04)
Content Security Policy (CSP) not implemented	Informational	✔ (A05)
Insecure Referrer Policy		✔ (A04)
Javascript Source map detected		×
Outdated Javascript Libraries		✔ (A06)
Password type input with auto-complete enabled		✔ (A04)
Possible server path disclosure Unix		✔ (A05)
Reverse proxy Detected		×
Subresource Integrity (SRI) not implemented		✔ (A08)

Table 6: Vulnerability report on the target web application

After the scanning was completed, the results contain a list of issues and their severity which is the first 2 columns in Table 6. In a quest to enumerate the impact of this vulnerability and the risk implication on the web application, the issues are then mapped to the OWASP Top 10. It was found that Acunectix, similar to other vulnerability scanning tools categorizes web application vulnerability according to the severity which is *High*, *Medium*, *Low,* and *Informational* as shown in Table 7. As the category name implies, it means vulnerability categorized as *Informational* is less critical when compared to the *High* severity category. It was observed in this Table 6 that the majority of the *Low* and *Informational* vulnerabilities actually belong to the OWASP Top 10 list which implies they pose the most security risk to the web application as suggested by the vulnerability scanning tool in Figure 6 and Table 7 respectively.

Severity	**Vulnabilities**	**Instances**
High	0	0
Medium	7	7
Low	7	7
Informational	8	10
Total	22	24

Table 7: Target web application vulnerability based on severity

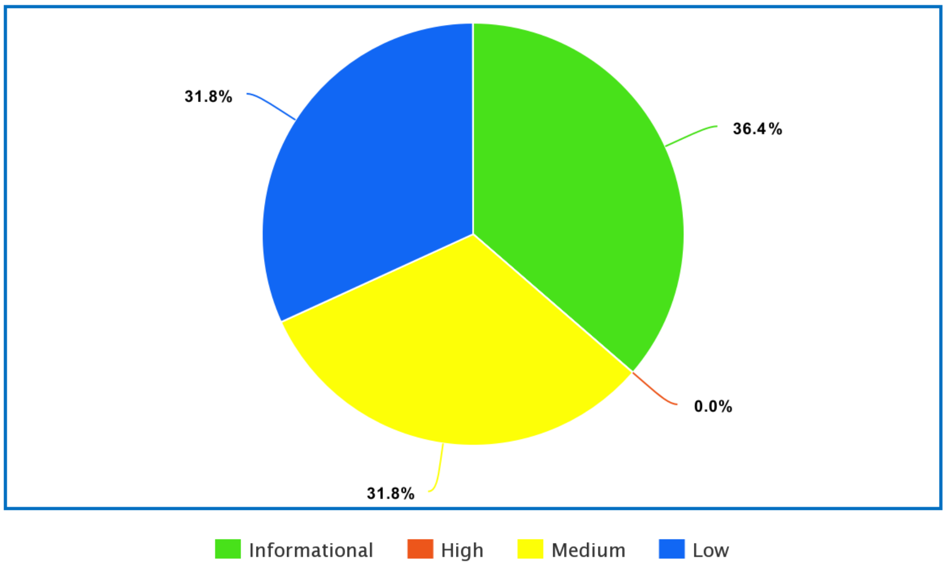


Figure 24. Acunetix Pie chart of the percentage of vulnerabilities on the target

> Web vulnerability scanners have been found to be useful in revealing problems in web applications but earlier findings have also shown that there are differences in the outcomes provided by various scanners (Alsaleh et al., 2017) Although little has been said about the severity categorization or interpreting using OWASP as carried out in this study.

Furthermore, Figure 24 illustrate the scanning result on a pie chart which shows *Informational* severity has the highest number of instance in the target web application with 36.4%, and Medium and Low have the same 31.8% and 0% for *High* severity. While these statistics literally suggest the target web application is relatively secure, especially with no *High* severity issue, Figure 25 shows only 15% of the reported issues can not be mapped to OWASP Top 10, and a staggering 85% is known as a most critical vulnerability security risk in web application and contain 5 out of the Top 10.

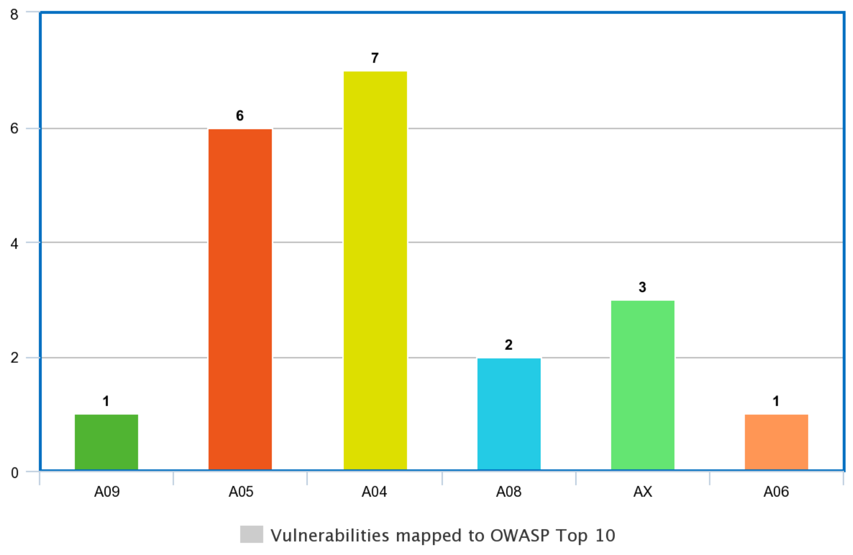


Figure. 25. Vulnerabilities and OWASP Top 10

Key

A04:2021 - Insecure Design

A05:2021 - Security Misconfiguration

A06:2021 - Vulnerable and Outdated Components

A08:2021 - Software and Data Integrity Failures

A09:2021 - Security Logging and Monitoring Failures

AX - Not Mapped OWASP Top 10

5.2 EXPERIMENT 2

The second vulnerability scanning was carried out on the target web application and the issues found with the severity was data is shown in Table 8 Similar to the Acunectix, the scanning issues were mapped to the OWASP Top 10 to determine the criticality of the issues and risk they posed as recorded in the 4th column. Additionally, the frequency column is the number of times each issue occurred in the web application. The Audited elements include Links, Forms, Cookies, XMLs, JSONs, UI inputs, and UI forms. The full URL enumerated is 65 and nearly 58% of those have issues with only 27 counted as safe.

Issues	**Severity**	**Frequency**	**OWASP Top 10**
Cross-Site Request Forgery (Trusted)	High	11	✔ (A10)
Common directory (Trusted)	Medium	4	✔ (A05)
Missing 'Strict-Transport-Security' header (Trusted)		1	✔ (A05)
Password field with auto-complete (Trusted)	Low	1	✔ (A04)
Missing 'X-Frame-Options' header (Trusted)		1	×
Interesting response (Trusted)	Informational	25	×
Email Address Disclosure		1	×
HTTPOnly cookie (Trusted)		1	✔ (A04)
Insecure cookie (Trusted)		1	✔ (A04)

Table 8 : Target scan result using Arachni tool

Arachni tool shows the target web application has 11 instances of CSRF issue which is a critical issue but this wasn't detected by Acunectix as it was not reported. This justifies the use of more than one tool for the vulnerability scanning and risk assessment of web applications.

This agrees with Kritikos *et al* (2019 p. 18) survey on vulnerability assessment tools whose comparative analysis shows different scanning tools are best for each criterion. As a result, for more vulnerability coverage, more than one tool should be deployed for vulnerability scanning of web applications.

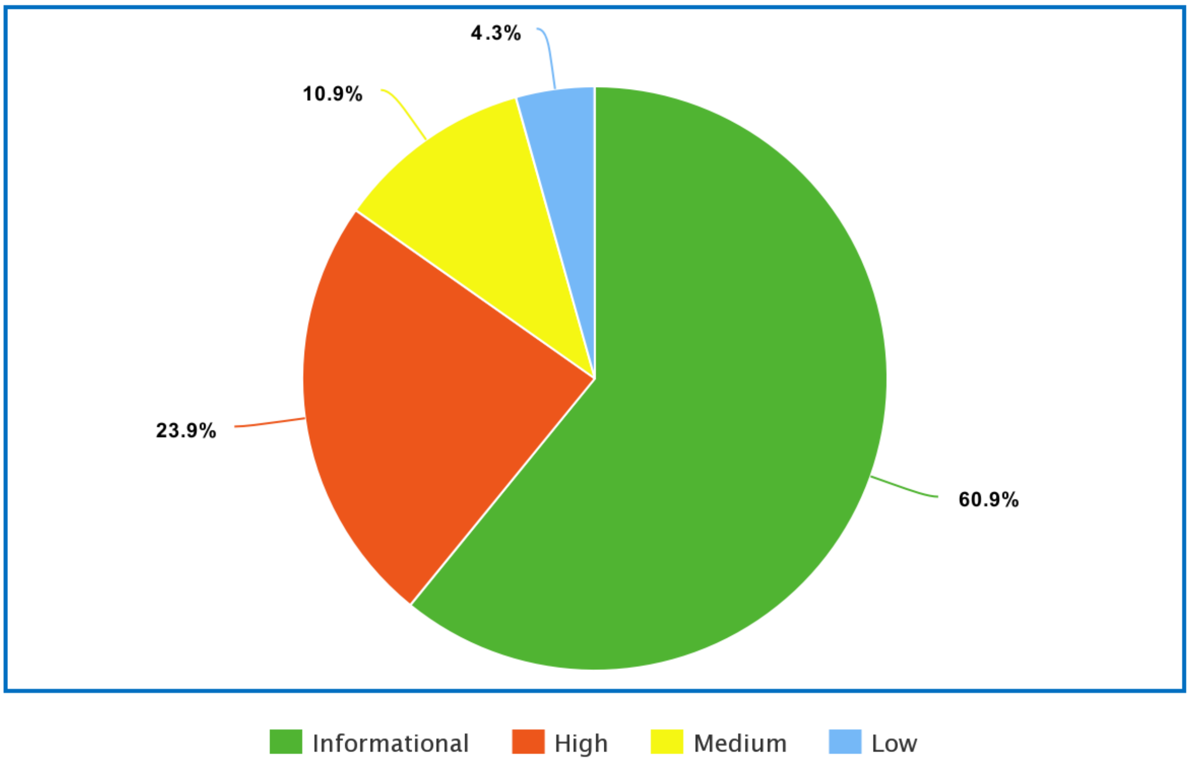


Figure 26 Target scan result from Arachni tool

A web vulnerability fundamental operation includes: Determining which pages are components of the web application using web crawlers; Extract the Data Entry Points (DEP) of the pages you viewed; Creating a malicious HTTP request to the target application that contains harmful patterns in each DEP parameter to simulate an attack; Check for vulnerabilities in the HTTP response (Aarya et al., 2018 p.125).

Furthermore, it was observed that each of these tools found the same vulnerability issues such as Missing 'Strict-Transport-Security' header, Insecure cookie, HTTPOnly cookie, Password field with auto-complete although Password field with the auto-complete issue is placed in *Low* severity in Arachni tool while Acunectix categorized it under *Informational*. Additionally, 60.9% of issues found are *Informational* with 89.28% of those issues named

Interesting response (Trusted). Acunectix on the other provides more specific issues than Arachni which is more generic. As a result, 58% of Arachni results could not be placed on OWASP Top 10 as shown in Figure 27. Arachni was able to mapped *Server-Side Request Forgery* issue under the *High* severity category as expected, although other *Low* and *Informational* severity are still found in the OWASP Top 10 web application most critical security risks as depicted in Figure 27

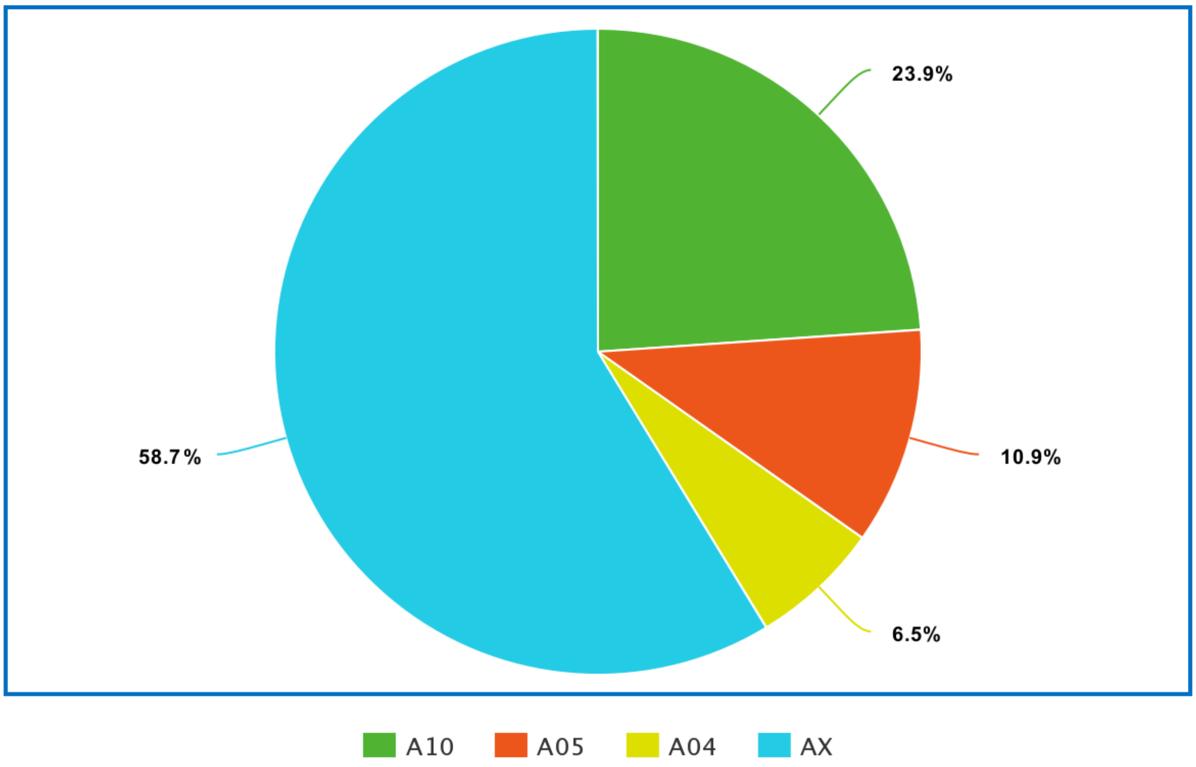


Figure 27: Target scan result from Acunectix tool

Key

A04:2021 - Insecure Design

A05:2021 - Security Misconfiguration

A10:2021 - Server-Side Request Forgery

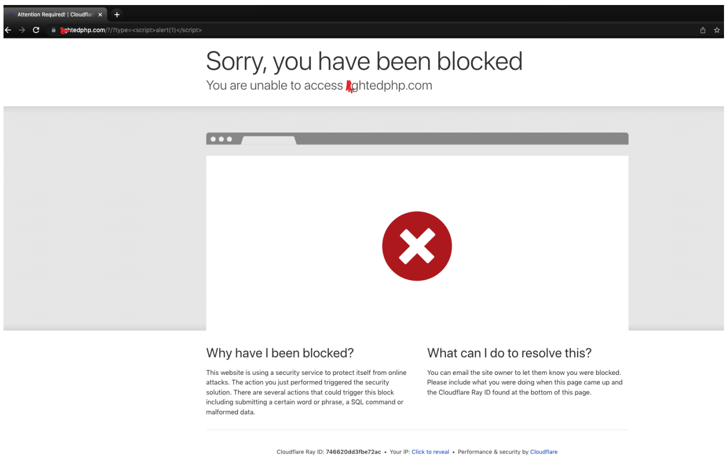
AX - Not Mapped OWASP Top 10

5.3 WAF PERFORMANCE

To illustrate the WAF performance, a range of different attack simulation was carried out on the target web application including Remote File Inclusion (RFI), Cross-Site Scripting (XSS), Local File Inclusion, SQL Injection.

5.3.1 Attack Simulation

The first attack was carried out by inputting an example of an XSS Scripting tag in the query as a URL parameter by making the GET REQUEST METHOD from the web browser. This attack was simulated through both Direct IP access and Domain name as illustrated in Figure 28, Figure 29, and Figure 30. The example of XSS used is <script> alert(1) </script> which is usually used in exfiltrating data from target web application (Djeki *et al.,* 2022).



Attention Required! | Cloudflar

[redacted]ghtedphp.com/?/?type=<script>alert(1)</script>

Sorry, you have been blocked

You are unable to access [redacted]ghtedphp.com

Why have I been blocked?

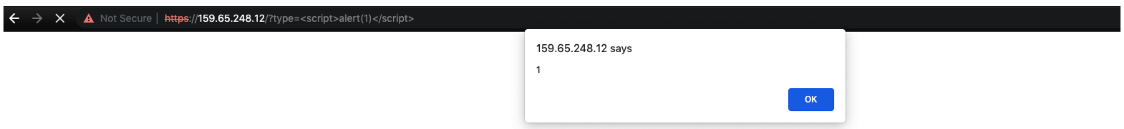
This website is using a security service to protect itself from online attacks. The action you just performed triggered the security solution. There are several actions that could trigger this block including submitting a certain word or phrase, a SQL command or malformed data.

What can I do to resolve this?

You can email the site owner to let them know you were blocked. Please include what you were doing when this page came up and the Cloudflare Ray ID found at the bottom of this page.

Cloudflare Ray ID: 746620dd3fbe72ac • Your IP: Click to reveal • Performance & security by Cloudflare

Figure 28: Cloudflare WAF Blocks XSS

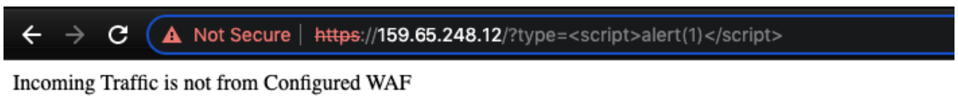


Not Secure | https://159.65.248.12/?type=<script>alert(1)</script>

159.65.248.12 says

1

Figure 29: Bypassed WAF Successful XSS attack on target Web Application



Not Secure | https://159.65.248.12/?type=<script>alert(1)</script>

Incoming Traffic is not from Configured WAF

Figure 30: WAF-VERIFIER Block request on target Web Application

Similar to the XSS attack in Figures 28, 29, and 30, the Local File Inclusion Attack was also carried out. Example Local execution was appended as a GET Parameter in the URL address is (?type=../../../etc/passwd) with an attempt to make the web application process it. The result shows the WAF blocks the request and also logs it for the web administrator.

```
Ray ID 746ebde1dcafe648
Method GET HTTP Version HTTP/2
Host lightedphp.com
Path /etc/passwd
Query “”
User agent Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36
IP address 90.197.4X.X
ASN AS5607 BSKYB-BROADBAND-AS
Country United Kingdom
Service Managed rules Rule ID 100005
Rule message DotNetNuke - File Inclusion - CVE:CVE-2018-9126, CVE:CVE-2011-1892
Rule group Cloudflare Specials
Action taken Block
```

The logs indicate the WAF is using the signature-based approach to block this particular attack which agrees with literature that recommends WAF (Alsobhi and Alshareef, 2020) to mitigate against Injection attack which ranks Top 5 on the OWASP.

Figure 31 shows the analytics from the WAF during the attack simulation. The analytics does not take into account the bypassed request as expected which means the WAF didn't pass through the WAF.

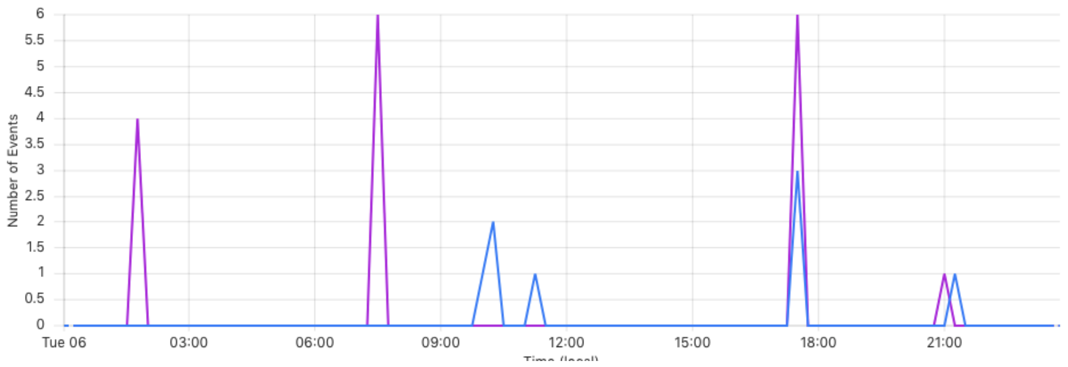


Figure 31 WAF overview during attack simulation

5.3.2 WAF-Verifier Implementation

WAF-VERIFIER coding concept follows the design layout in Figure 20. Its implementation on Figure 23 shows it can offer a layer of security and authentication at the application layer just before the request gets into the application. By performing the authentication on the application layer, it is possible to offer this protection transparently, i.e., protect the web application so it can continue to operate without any notion of bypassed concerns. The authentication mechanism extracts the IP and meta-data from the request. Below is an example of the response header:

```
cache-control: max-age=14400
cf-cache-status: EXPIRED
cf-ray: 7466ff519d7d7521-LHR
content-encoding: br
content-type: text/html
date: Tue, 06 Sep 2022 11:52:14 GMT
etag: W/"718c-57edad08014d6-gzip"
last-modified: Mon, 07 Jan 2019 09:25:49 GMT
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
report-to:
{"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v3?s=MpNM8alY5QxTg2pkvFx
UGPNaKWusO1dWcY4PCwAhZ8rNcZGQYMH63g2I1SUbCprJmLYXYNCUXtpwMPVpB1hzMUSb
ulieHFyYl%2FcuBuBXSHjBFp%2F9Zb7Z3eb8ScaNSpsk"}],"group":"cf-nel","max_age":604800}
server: cloudflare
vary: Accept-Encoding
```

Cloud-based Web Application Firewall usually leave a fingerprint on the server through which request are proxied through as shown the above log. The *server* key is usually *apache* or *Nginx* but because the request was proxied through a Cloud-based WAF, it replaces the original server name. This among other metadata is used by the WAF-VERIFIER to verify where the source of the request. The server name is usually saved in the $_SERVER Global variable in the PHP-based web application.

Studies have been carried out to illustrate how malicious actors can manipulate the Request Header (Jaafar et al., 2020), as a result, WAF-VERIFIER does not only depend on Header information attributes to authenticate the source of the request. White-listing is an effective defense-in-depth method that mitigates various attack vectors (Jain and Gupta, 2016), hence WAF-VERIFIER employs a white-list-based approach to the IP.

Unlike UDP traffic that could be subjected to an IP spoofing attack, HTTP is a TCP protocol which means it uses a *handshake* to establish a connection. The metadata extracted is compared with the configured WAF used by the web administrator (the default WAF is Cloudflare) as this is WAF used in this study. Following the experiment as shown in Figures 28, 29, and 30, the tool gives excellent IP-based protection to web applications. The WAF-VERIFIER tool effect was verified

5.4 EXPERIMENT 3

The performance impact of WAF was measured by deploying three major tools following the selection criteria mentioned in the research methodology. Apache Bench tool was first used to illustrate concurrent requests on the server with and without WAF and the results are recorded in Table 9. This is based on 1000 requests for each 100 - 1000 concurrent requests as using the following command

```
#ab -c 100 -n 1000 https://www.xyghtedphp.com/
```

Concurrency	With WAF (ms)	Without WAF (ms)
100	18	9
200	34	101
300	105	102
400	152	128
500	229	103
600	254	126
700	598	107
800	445	144
900	581	153
1000	904	139
Average Connection Times	332	111.2

Table 9 : Request Connection Time

Concurrency	With WAF (Kbytes/sec)	Without WAF (Kbytes/sec)
100	2,092.82	11,307.63
200	2,015.21	11,620.37
300	1,644.50	11,516.07
400	1,105.01	11,592.13
500	1,088.24	11,199.31
600	985.37	10,862.43
700	779.27	10,943.01
800	679.96	11,503.38
900	550.36	11,267.55
1000	489.62	10,981.59

Table 10 : Request Transfer Rate

In Figure 32, the request connection rate increases as the number of concurrent users on the web application are increased.

Additionally, its shows the network latency is higher with WAF connections than without WAF which agrees with Arnaldy and Hati's (2020) performance analysis on a reverse proxy which is similar to the WAF used in this study.

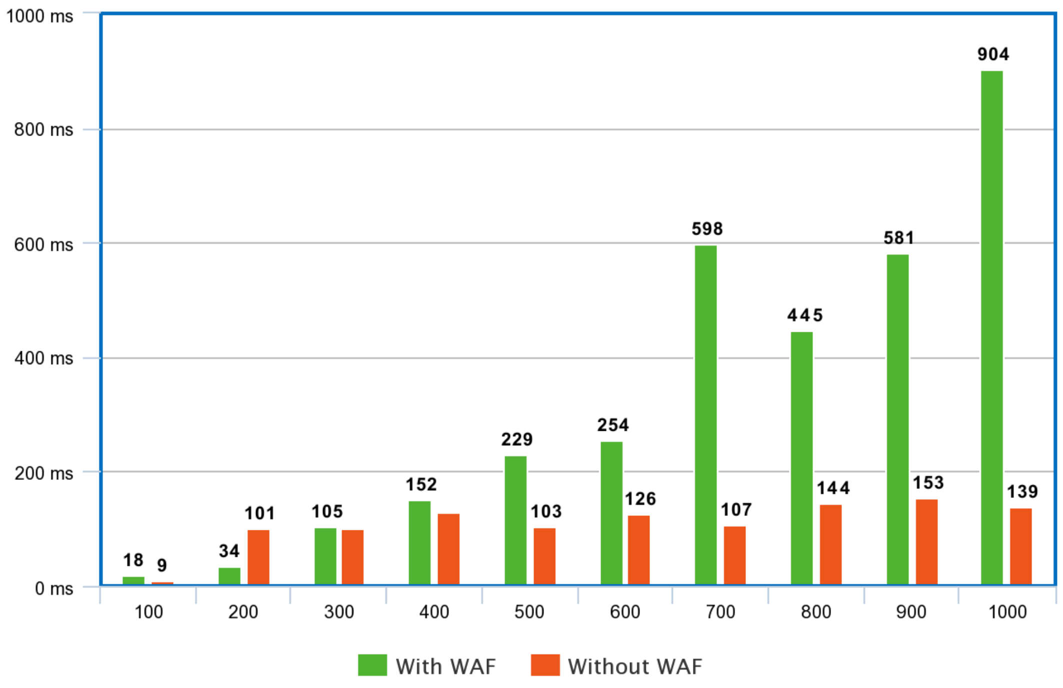


Figure 32: Request Connection Time

The average connection time of a web application with WAF is 3 times more compared to a connection without WAF. This is likely due to the protection mechanism used by the Web Application Firewall that filters both outgoing requests to the server and incoming responses from the server. In Figure 33, the request rate for concurrent requests without WAF is much more than for those with WAF.

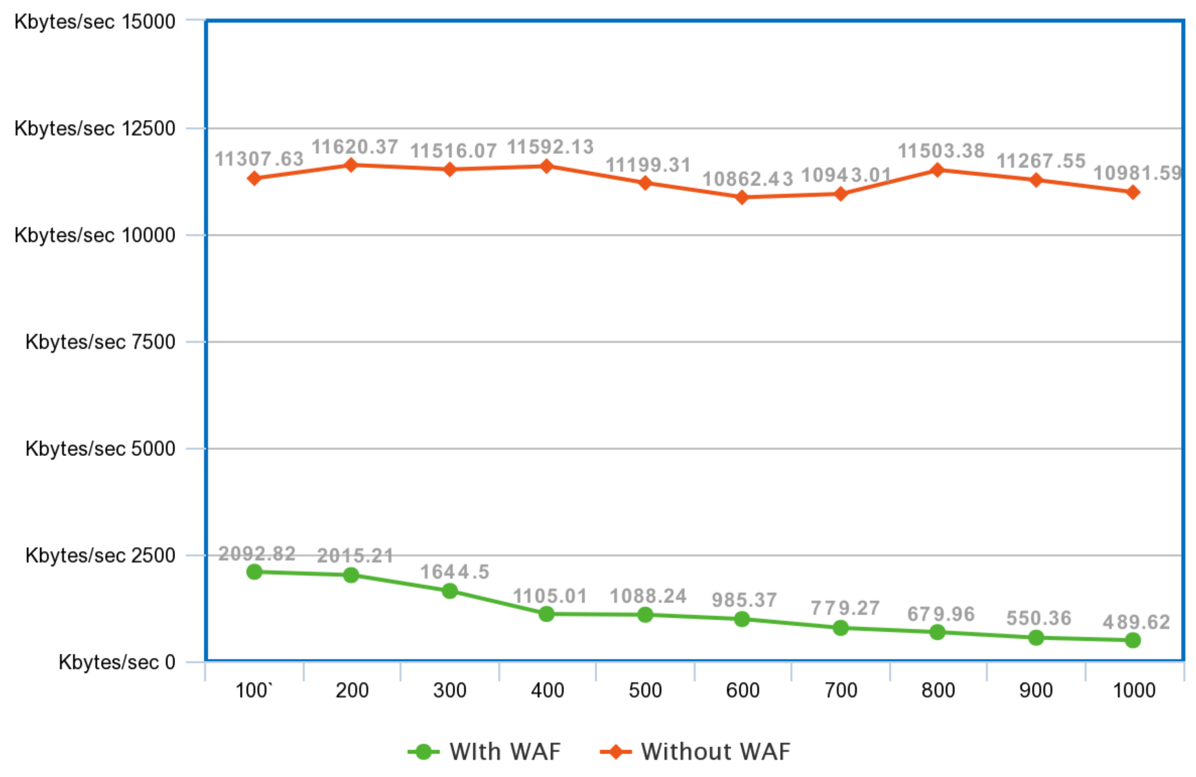


Figure 33: Request Transfer Rate

Cloud-based WAF used in this study is likely responsible for the huge request transfer rate as WAF used also double as a Content Delivery Network (CDN) which means it the request are cached on the Cloudflare Edge server. Additionally, as the request concurrency increases, the request rate with WAF also decreases even though the connection time increases. The study shows WAF request rate is inversely related to connection time while the request without WAF has a high request transfer rate but also low connection time.

Jmeter was also used as a second tool to analyze the performance of WAF. Similar to the transfer rate measured in with Apache Bench, the throughput rate was also analyzed based on the number of concurrent users. The throughput results were similar to ApaheBench, although when the concurrent user hit 700 upwards, the error rate increased from 0% to 19.4% and further to 40%.

This occurs as a result of a Browser integrity check or DDOS attacks usually carried out by the Jmeter tool (Grabovsky *et al.,* 2018).

Hence the Web Application Firewall blocks the request from going to the origin server. Gatling performance analysis was carried out on HTTP requests of both WAF and non-WAF requests as illustrated in Figures 34 and 35 respectively. The request was downloaded from the browser network tab to a HAR file. The HAR was converted to a Scalar file for the Gatling simulation.

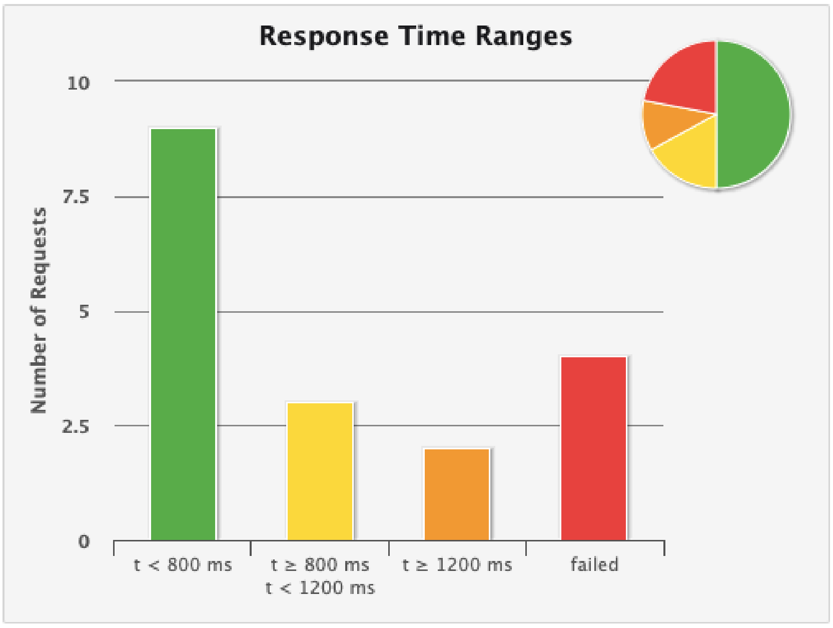


Figure 34 Gatling request response time rate without WAF

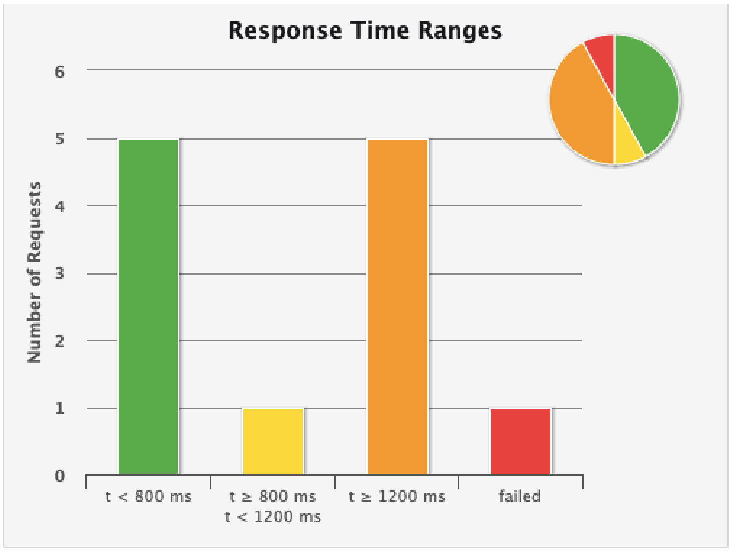


Figure 35: Gatling request response time rate with WAF

Figure 35, indicates proxied WAF request response time is longer when compared to non-proxied requests with many of the requests taking longer than 1200ms. This agrees with the analysis using the Apache Bench tool although the request transfer performance improves with WAF proxied requests.

> The results of this study using 3 performance testing tools agree with Thomas-Reynolds and Butakov's (2020) performance analysis of WAF, although there was based on Modsecurity WAF using the Siege testing tool.

Recall the research question for this study:

RQ1: “How effective is application-level authentication to prevent Web Application Firewall bypass to ensure application reliability?”

The experimental results show that the proposed intervention (WAF-VERIFIER) led to improvements in the security layer of the web application. Researchers have proposed server-level authentication by using certificates on both the origin server and WAF (SHOBIRI et al., 2021), although the method can be subjected to a man-in-the-middle attack if the origin server does not authenticate the server’s certificate. However, the intervention proposed in this study has the most impact when there is more than one web application running on the server or where developers have limited access or knowledge about how to configure certificates on both servers. Most of the Bypass techniques proposed by Nagendran *et al* (2020) and were blocked by WAF as illustrated in Experiment 3, however, WAF-VERIFIER protects the web application by ensuring the requests that have not been filtered by the WAF didn’t get to the web application. The source code of the proposed intervention is outlined in Appendix III and it is submitted as an Artefact for this MSc research project.

RQ2: What factors affect the reliability of WAF in the protection web application?

The results of this MSc research, Configuration issue are a major factor affecting the reliability of Web Application Firewall as it accounts for 35% of the critical vulnerability. This means for every 10 vulnerabilities found, approx 4 are likely to be configuration issues as shown in Figure 25. Surprisingly, this configuration is not the Firewall configuration but the web server configuration as enumerated by the web vulnerabilities assessment. The web vulnerabilities assessment shows the security of web application does not only depends on the web application but also on the web server configuration. Some of the vulnerabilities detected by the scanning tools on the target web applications include “Directory Listing”, “PHP allow_url_fopen enabled”, and “PHP open_basedir not set” which are critical security issues according to OWASP. This study's findings agree with Clincy and Shahriar (2018) on WAF security models and configuration which suggested that using the default configuration on a web server can lead to vulnerabilities and render WAF protection futile. Their work is orthogonal to this Msc research since it doesn’t consider any specific Web Application Firewall. In this study, Cloudflare Web Application was used and the results accentuate the importance of server hardening.

Based on the empirical analysis in this Chapter, the aim of this study was accomplished along with the objectives and research question.

CHAPTER 6. CONCLUSION

This paper advocates the general framework for improving the security of existing cloud-based WAFs. In this research, A systematic review of the literature was carried out using the PRISMA method, a Quantitative approach using the experimental research method was implemented to analyze the security mechanism of existing WAF in a testbed environment, and WAF-Verifier, server-based software was developed to mitigate WAF bypass. Although researchers have suggested dynamic IP, as a mitigation approach to preventing this bypass (Ghaznavi et al., 2021), the limitation that is many VPS provider uses static IP, and besides, and web attack can happen before IP changes. As a result, the WAF-VERIFIER solution was proposed in this study. WAF VERIFIER is an application-level authentication approach to enforcing web application firewall performance and ensuring the security gate are not bypassed.

In Summary contributions of this study include

- Deployment of WAF-VERIFIER servers as bypass mitigation on Webserver
- Web vulnerability scanning tool severity categorization (*High*, *Low*, *Medium*, or *Info*) could be misleading and confusing to web developers and web administrators as discovered in this study. Some Issues are categorized as *Low* severity or *Informational* and sometimes belong to OSWAP Top 10 High critical security as illustrated in the earlier chapter, as a result, it is recommended all issues should be treated with the same precedence following risk management standards such as ISO 27001 as outlined in section A.12.6.1
- Effectiveness of WAF relies not on how sophisticated the WAF protection is but also on the configuration and hardening of the origin server.
- There are more attack vectors against web applications than what WAF could handle as shown in the findings of this study. Hence it is recommended that the security of web applications should start from the inside-out such as sanitization of inputs, and using updated libraries (Fredj et al., 2020) which implies the security of web applications starts from the web application developer
- Component Interaction vulnerability assessment; while most vulnerability scanning tools focus on each web application component separately, It was seen that this is

insufficient due unexpected behavior could be discovered in component interactions with one another. As a result, the application topology must be considered in order to have a comprehensive perspective of the entire application and enumerate possible sources of risks, its structure, and how its components interact with one another such as the interaction of the web application with the WAF, as well as other dependencies such hosting and communication

6.1 FUTURE RECOMMENDATION FOR FURTHER RESEARCH

The findings of this MSc research project should, like with all study, be construed with caution. There are a number of limitations in this study, therefore additional research is recommended.

- A limited number of Web Application Firewalls was considered because most Web Application Firewalls are enterprise solutions making them difficult to be analyzed in this study. As a result, the results of this study could be taken further so more Web Applications Firewalls are analyzed.
- Although this study implements WAF-VERIFIER which is an application-level authentication approach. A study of other bypass mitigation approaches should is recommended.
- Further study into the relationship between the web application scanning tools severity report and OWASP should continue to provide web administrators and developers with clear-cut vulnerability assessment on the web application.

6.2 FUTURE RECOMMENDATION FOR INDUSTRY

Web applications will usually be a target for malicious actors because they are easily accessible making them more vulnerable to cyberattacks. As a result, it is imperative for web application administrators and developers to build a resilient web application such that tools like Web Application Firewall will serve as a secondary protection tool rather than a primary. Furthermore, vulnerability threats and issues detected by scanning tools should be treated with the same precedence as most issues can equally put data and services at risk.

Additionally, risks do not only exist within the application as shown in this study, they can also come within service integration and communication as shown during WAF bypass. At times, organizations force web developers to trade security for functionality to satisfy customers; this study recommends against this as it might have legal and financial consequences as shown throughout the case study in the literature review section of this study.

REFERENCES

Aarya, P.S., Rajan, A., Sachin, K.P.S., Gopi, R. and Sreenu, G., 2018, June. Web Scanning: Existing Techniques and Future. In 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 123-128). IEEE.

Alicea, M. and Alsmadi, I., 2021. Misconfiguration in Firewalls and Network Access Controls: Literature Review. Future Internet, 13(11), p.283.

Alsaleh, M., Alomar, N., Alshreef, M., Alarifi, A. and Al-Salman, A., 2017. Performance-based comparative assessment of open source web vulnerability scanners. Security and Communication Networks, 2017.

Alsobhi, H. and Alshareef, R., 2020, September. Sql injection countermeasures methods. In 2020 International Conference on Computing and Information Technology (ICCIT-1441) (pp. 1-4). IEEE.

Applebaum, S., Gaber, T. and Ahmed, A., 2021. Signature-based and machine-learning-based web application firewalls: A short survey. Procedia Computer Science, 189, pp.359-367.

Apuke, O.D., 2017. Quantitative research methods: A synopsis approach. Kuwait Chapter of Arabian Journal of Business and Management Review, 33(5471), pp.1-8.

Arachni. 2022. GitHub - Arachni/arachni: Web Application Security Scanner Framework. [online] Available at: <https://github.com/Arachni/arachni> [Accessed 1 September 2022].

Arnaldy, D. and Hati, T.S., 2020, September. Performance Analysis of Reverse Proxy and Web Application Firewall with Telegram Bot as Attack Notification On Web Server. In 2020 3rd International Conference on Computer and Informatics Engineering (IC2IE) (pp. 455-459). IEEE.

Arora, N., Singh, P., Sahu, S., Keshari, V.K. and Vinoth Kumar, M., 2021. Preventing SSRF (Server-Side Request Forgery) and CSRF (Cross-Site Request Forgery) Using Extended Visual Cryptography and QR Code. In Proceedings of Second International Conference on Smart Energy and Communication (pp. 215-227). Springer, Singapore.

Arulkumar, C.V., Jeyakumar, K., Malarmathi, M. and Shanmugapriya, T., 2012. Secure communication in unstructured P2P networks based on reputation management and self certification. International Journal of Computer Applications, 15, pp.1-3.

Assal, H. and Chiasson, S., 2019, May. 'Think secure from the beginning' A Survey with Software Developers. In Proceedings of the 2019 CHI conference on human factors in computing systems (pp. 1-13).

Atmowardoyo, H., 2018. Research methods in TEFL studies: Descriptive research, case study, error analysis, and R & D. Journal of Language Teaching and Research, 9(1), pp.197-204.

Bach-Nutman, M., 2020. Understanding the top 10 owasp vulnerabilities. arXiv preprint arXiv:2012.09960.

Bakker, J.I., 2019. Grounded theory methodology and grounded theory method: Introduction to the special issue. Sociological Focus, 52(2), pp.91-106.

Betarte, G., Giménez, E., Martínez, R. and Pardo, A., 2018, December. Improving web application firewalls through anomaly detection. In 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 779-784). IEEE.

Bhor, R.V. and Khanuja, H.K., 2016, August. Analysis of web application security mechanism and attack detection using vulnerability injection technique. In 2016 International Conference on Computing Communication Control and automation (ICCUBEA) (pp. 1-6). IEEE.

Bijjou, K., 2019, November. Web Application Firewall Bypassing: An Approach for Penetra. In In Depth Security Vol. III: Proceedings of the DeepSec Conferences (Vol. 3, p. 29). BoD–Books on Demand.

Boccagni, P. and Schrooten, M., 2018. Participant observation in migration studies: An overview and some emerging issues. Qualitative research in European migration studies, pp.209-225.

Caetano, A. and Nico, M., 2019. Forever young: Creative responses to challenging issues in biographical research. Contemporary Social Science, 14(3-4), pp.361-378.

Cahit, K.A.Y.A., 2015. Internal validity: A must in research designs. Educational Research and Reviews, 10(2), pp.111-118.

Castellan, C.M., 2010. Quantitative and qualitative research: A view for clarity. International journal of education, 2(2), p.1.

Check Point. 2022. Check Point 2022 Cyber Security Report. [online] Available at: <https://resources.checkpoint.com/cyber-security-resources/check-point-softwares-2022-security-report> [Accessed 1 July 2022].

Clandinin, D.J., 2016. Engaging in narrative inquiry. Routledge.

Clincy, V. and Shahriar, H., 2018, July. Web application firewall: Network security models and configuration. In 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC) (Vol. 1, pp. 835-836). IEEE.

D'Hoinne, J., Hils, A. and Neiva, C., 2014. Magic quadrant for web application firewalls. Technical report, Gartner, Stamford, CT.

Deepa, G. and Thilagam, P.S., 2016. Securing web applications from injection and logic vulnerabilities: Approaches and challenges. Information and Software Technology, 74, pp.160-180.

Desmet, L., Piessens, F., Joosen, W. and Verbaeten, P., 2006, November. Bridging the gap between web application firewalls and web applications. In Proceedings of the fourth ACM workshop on Formal methods in security (pp. 67-77).

Demetrio, L., Valenza, A., Costa, G. and Lagorio, G., 2020, March. Waf-a-mole: evading web application firewalls through adversarial machine learning. In Proceedings of the 35th Annual ACM Symposium on Applied Computing (pp. 1745-1752).

Disawal, S. and Suman, U., 2021, March. An Analysis and Classification of Vulnerabilities in Web-Based Application Development. In 2021 8th International Conference on Computing for Sustainable Global Development (INDIACom) (pp. 782-785). IEEE.

Djeffal, S., 2020. Adaptive defense against adverserial artificial intelligence at the edge of the cloud using evolutionary algorithms (Doctoral dissertation, Massachusetts Institute of Technology).

Djeki, E., Degila, J., Bondiombouy, C. and Alhassan, M.H., 2022, April. Preventive Measures for Digital Learning Spaces' Security Issues. In 2022 IEEE Technology and Engineering Management Conference (TEMSCON EUROPE) (pp. 48-55). IEEE.

Endraca, A., King, B., Nodalo, G., Maria, M.S. and Sabas, I., 2013. Web Application Firewall (W AF). International Journal of e-Education, e-Business, e-Management and e-Learning, 3(6), p.451.
F5 Operation Guide, 2022. K39707080: Insecure design (A4) | Secure against the OWASP Top 10 for 2021. [online] F5. Available at: <https://support.f5.com/csp/article/K39707080> [Accessed 22 August 2022].

Fernandez, E.B., Yoshioka, N. and Washizaki, H., 2014, March. Patterns for cloud firewalls. In Proceedings of the Asian Conference on Pattern Languages of Programs (AsianPLoP).

Flynn, S.V. and Korcuska, J.S., 2018. Credible phenomenological research: A mixed-methods study. Counselor Education and Supervision, 57(1), pp.34-50.

Fredj, O.B., Cheikhrouhou, O., Krichen, M., Hamam, H. and Derhab, A., 2020, November. An OWASP top ten driven survey on web application protection methods. In International Conference on Risks and Security of Internet and Systems (pp. 235-252). Springer, Cham.

Fonseca, J., Seixas, N., Vieira, M. and Madeira, H., 2013. Analysis of field data on web security vulnerabilities. IEEE transactions on dependable and secure computing, 11(2), pp.89-100.

Gatlan, S., 2020. Freepik data breach: Hackers stole 8.3M records via SQL injection. [online] BleepingComputer. Available at: <https://www.bleepingcomputer.com/news/security/freepik-data-breach-hackers-stole-83m-records-via-sql-injection/> [Accessed 19 August 2022].

Gauci, S. and Mondal, P., 2020. GitHub - EnableSecurity/wafw00f: WAFW00F allows one to identify and fingerprint Web Application Firewall (WAF) products protecting a website.. [online] GitHub. Available at: <https://github.com/EnableSecurity/wafw00f> [Accessed 24 June 2022].

Ghaznavi, M., Jalalpour, E., Salahuddin, M.A., Boutaba, R., Migault, D. and Preda, S., 2021. Content delivery network security: a survey. IEEE Communications Surveys & Tutorials, 23(4), pp.2166-2190.

Grabovsky, S., Cika, P., Zeman, V., Clupek, V., Svehlak, M. and Klimes, J., 2018, November. Denial of service attack generator in Apache JMeter. In 2018 10th International Congress on

Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT) (pp. 1-4). IEEE.

Hacker, A.J. and CISSP, I., 2008. Importance of web application firewall technology for protecting web-based resources. ICSA Labs an Independent Verizon Business, p.7.

Hassan, M., Ali, M., Bhuiyan, T., Sharif, M. and Biswas, S., 2018. Quantitative assessment on broken access control vulnerability in web applications. In International Conference on Cyber Security and Computer Science 2018.

He, S., He, P., Chen, Z., Yang, T., Su, Y. and Lyu, M.R., 2021. A survey on automated log analysis for reliability engineering. ACM Computing Surveys (CSUR), 54(6), pp.1-37.

H. -C. Huang, Z. -K. Zhang, H. -W. Cheng and S. W. Shieh, 2017. Web Application Security: Threats, Countermeasures, and Pitfalls, in Computer, vol. 50, no. 6, pp. 81-85, doi: 10.1109/MC.2017.183.

IBM, 2022. IBM Security X-Force Threat Intelligence Index 2022 Full Report. p.14

Industryarc . 2021. Web Application Firewall Market Share, Size and Industry Growth Analysis 2021 - 2026. [online] Available at: <https://www.industryarc.com/Research/Web-Application-Firewall-Market-Research-502436> [Accessed 10 August 2022].

IŞiker, B. and SoĞukpinar, İ., 2021, December. Machine Learning Based Web Application Firewall. In 2021 2nd International Informatics and Software Engineering Conference (IISEC) (pp. 1-6). IEEE.

Jaafar, A.G., Ismail, S.A., Abdullah, M.S., Kama, N., Azmi, A. and Yusop, O.M., 2020. Recent Analysis of Forged Request Headers Constituted by HTTP DDoS. Sensors, 20(14), p.3820.

Jain, A.K. and Gupta, B.B., 2016. A novel approach to protect against phishing attacks at client side using auto-updated white-list. EURASIP Journal on Information Security, 2016(1), pp.1-11.

Jia, Q., Wang, H., Fleck, D., Li, F., Stavrou, A. and Powell, W., 2014, June. Catch me if you can: A cloud-enabled DDoS defense. In 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (pp. 264-275). IEEE.

Jibilian, I. and Canales, K., 2021. The US is readying sanctions against Russia over the SolarWinds cyber attack. Here's a simple explanation of how the massive hack happened and why it's such a big deal. [online] Business Insider. Available at: <https://www.businessinsider.com/solarwinds-hack-explained-government-agencies-cyber-security-2020-12> [Accessed 25 August 2022].

Jin, L., Hao, S., Wang, H. and Cotton, C., 2018, June. Your remnant tells secret: Residual resolution in ddos protection services. In 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (pp. 362-373). IEEE.

Johnston, M.P., 2017. Secondary data analysis: A method of which the time has come. Qualitative and quantitative methods in libraries, 3(3), pp.619-626.

Kothari, C.R. and Gaurav, G., 2019. Research Methodology: Methods and techniques third edition. New Age International, New Delhi.

Kritikos, K., Magoutis, K., Papoutsakis, M. and Ioannidis, S., 2019. A survey on vulnerability assessment tools and databases for cloud-based web applications. Array, 3, p.100011.

Kumar, P.R., Raj, P.H. and Jelciana, P., 2018. Exploring data security issues and solutions in cloud computing. Procedia Computer Science, 125, pp.691-697.

LAKHNO, V., BLOZVA, A., KASATKIN, D., CHUBAIEVSKYI, V., SHESTAK, Y., TYSHCHENKO, D. and BRZHANOV, R., 2022. EXPERIMENTAL STUDIES OF THE FEATURES OF USING WAF TO PROTECT INTERNAL SERVICES IN THE ZERO TRUST STRUCTURE. Journal of Theoretical and Applied Information Technology, 100(3).

Lawrence, C., 2011. Next Generation Firewalls for Dummies.

Leedy, P.D. and Ormrod, J.E., 2019. Practical research: Planning and design. Pearson. One Lake Street, Upper Saddle River, New Jersey 07458.

Mazhar, S.A., Anjum, R., Anwar, A.I. and Khan, A.A., 2021. Methods of data collection: A fundamental tool of research. Journal of Integrated Community Health (ISSN 2319-9113), 10(1), pp.6-10.

Li, L. and Wei, L., 2019, October. Automatic XSS detection and automatic anti-anti-virus payload generation. In 2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC) (pp. 71-76). IEEE.

Liang, J. and Kim, Y., 2022, January. Evolution of Firewalls: Toward Securer Network Using Next Generation Firewall. In 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 0752-0759). IEEE.

Li, X. and Xue, Y., 2014. A survey on server-side approaches to securing web applications. ACM Computing Surveys (CSUR), 46(4), pp.1-29.

Marchand-Melsom, A. and Nguyen Mai, D.B., 2020, June. Automatic repair of OWASP Top 10 security vulnerabilities: A survey. In Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (pp. 23-30).

McCusker, K. and Gunaydin, S., 2015. Research using qualitative, quantitative or mixed methods and choice based on the research. Perfusion, 30(7), pp.537-542.

MOHER, D. et al., 2009. Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement. PLoS medicine, 6(7), e1000097

Moradi Vartouni, A., Mehralian, S., Teshnehlab, M. and Sedighian Kashi, S., 2019. Auto-Encoder LSTM Methods for Anomaly-Based Web Application Firewall. International Journal of Information & Communication Technology Research, 11(3), pp.49-56.

Moradi Vartouni, A., Shokri, M. and Teshnehlab, M., 2021. Auto-threshold deep SVDD for anomaly-based web application firewall.

Morse, J.M., 2016. Mixed method design: Principles and procedures. Routledge.

Muzaki, R.A., Briliyant, O.C., Hasditama, M.A. and Ritchi, H., 2020, October. Improving Security of Web-Based Application Using ModSecurity and Reverse Proxy in Web Application Firewall. In 2020 International Workshop on Big Data and Information Security (IWBIS) (pp. 85-90). IEEE.

Nagendran, K., Balaji, S., Raj, B.A., Chanthrika, P. and Amirthaa, R.G., 2020, March. Web application firewall evasion techniques. In 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS) (pp. 194-199). IEEE.

Nunan, A.E., Souto, E., Dos Santos, E.M. and Feitosa, E., 2012, July. Automatic classification of cross-site scripting in web pages using document-based and URL-based features. In 2012 IEEE symposium on computers and communications (ISCC) (pp. 000702-000707). IEEE.

Owasp.org. 2021. OWASP Top Ten | OWASP Foundation. [online] Available at: <https://owasp.org/www-project-top-ten/> [Accessed 14 August 2022].

Pantoulas, E., 2022. Description, analysis and implementation of a Web Application Firewall (WAF). Creation of attack scenarios and threats prevention (Master's thesis, Πανεπιστήμιο Πειραιώς).

PCI Data Security Standards Data, 2018, *The Prioritized Approach to Pursue PCI DSS Compliance*. v3.2.1 6.6, 1.2.

Pentasecurity., 2020. 3 Types of Web Application Firewalls: How to Choose? | Penta Security Systems Inc.. [online] Penta Security Systems Inc. Available at: <https://www.pentasecurity.com/blog/3-types-web-application-firewalls/> [Accessed 12 August 2022].

Perera, A.C., Kesavan, K., Bannakkotuwa, S.V., Liyanapathirana, C. and Rupasinghe, L., 2016, December. E-commerce (WEB) Application security: Defense against Reconnaissance. In 2016 IEEE International Conference on Computer and Information Technology (CIT) (pp. 732-742). IEEE.

Ponemon, L., 2019. The State of Web Application Firewalls. [online] ponemonsullivanreport. Available at: <https://ponemonsullivanreport.com/2019/07/the-state-of-web-application-firewalls/> [Accessed 29 April 2022].

Prabhudesai, P., Bhalerao, A.A. and Prabhudesai, R., 2019. Web Application Firewall: Artificial Intelligence Arc. International Research Journal of Engineering and Technology (IRJET).

Price, P.C., Jhangiani, R.S. and Chiang, I.C.A., 2015. Quasi-experimental research. Research methods in psychology.

Qiu Y., 2013, Analysis and defense of XSS cross-site scripting attacks[M]. Posts & Telecom Press.

Rahman, R.U. and Tomar, D.S., 2020. Taxonomy of Login Attacks in Web Applications and Their Security Techniques Using Behavioral Biometrics. In Modern Theories and Practices for Cyber Ethics and Security Compliance (pp. 122-139). IGI Global.

Ratelle, J.T., Sawatsky, A.P. and Beckman, T.J., 2019. Quantitative research methods in medical education. Anesthesiology, 131(1), pp.23-35.

Razzaq, A., Hur, A., Shahbaz, S., Masood, M. and Ahmad, H.F., 2013, March. Critical analysis on web application firewall solutions. In 2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS) (pp. 1-6). IEEE.

Ross, S.M. and Morrison, G.R., 2013. Experimental research methods. In Handbook of research on educational communications and technology (pp. 1007-1029). Routledge.

Sɸhoel, H., Jaatun, M.G. and Boyd, C., 2018, June. OWASP Top 10-Do Startups Care?. In 2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security) (pp. 1-8). IEEE.

Sørensen, F., Mattsson, J. and Sundbo, J., 2010. Experimental methods in innovation research. Research policy, 39(3), pp.313-322.

Saunders, M., Lewis, P. and Thornhill, A., 2009. Research methods for business students. Pearson education.

Saxena, A. and Saxena, A., 2019. Case study: A research method. Voice of Intellectual Man-An International Journal, 9(2), pp.55-68.

Saxena, P., 2014. OSI reference model–a seven-layered architecture of OSI model. International Journal of Research, 1(10), pp.1145-1156.

Seals, T., 2018. Knuddels Flirt App Slapped with Hefty Fine After Data Breach. [online] Threatpost.com. Available at: <https://threatpost.com/knuddels-flirt-app-slapped-with-hefty-fine-after-data-breach/139384/> [Accessed 19 August 2022].

Seeram, E., 2019. An overview of correlational research. Radiologic technology, 91(2), pp.176-179.

Shaheed, A. and Kurdy, M.H.D., 2022. Web Application Firewall Using Machine Learning and Features Engineering. Security and Communication Networks, 2022.

SHOBIRI, B., MANNAN, M. and YOUSSEF, A., 2021. CDNs' Dark Side: Security Problems in CDN-to-Origin Connections.

Sharma, A., 2021. Fintech firm hit by Log4j hack refuses to pay $5 million ransom. [online] BleepingComputer. Available at: <https://www.bleepingcomputer.com/news/security/fintech-firm-hit-by-log4j-hack-refuses-to-pay-5-million-ransom/> [Accessed 22 August 2022].

Shrivastava, S. and Prapulla, S.B., 2020. Comprehensive Review of Load Testing Tools.

Singh, C. and Ishtiaque-Al-Mahmood, M.A.T., 2022. Novel Approach to Secure Websites with Machine Learning Classifiers.

Singh, J.J., Samuel, H. and Zavarsky, P., 2018, April. Impact of paranoia levels on the effectiveness of the modsecurity web application firewall. In 2018 1st International Conference on Data Intelligence and Security (ICDIS) (pp. 141-144). IEEE.

Skybakmoen, T., 2019. Next generation firewall comparative analysis.

Steffens, M., Rossow, C., Johns, M. and Stock, B., 2019. Don't Trust The Locals: Investigating the Prevalence of Persistent Client-Side Cross-Site Scripting in the Wild.

Stemler, S.E., 2015. Content analysis. Emerging trends in the social and behavioral sciences: An Interdisciplinary, Searchable, and Linkable Resource, pp.1-14.

Takahashi, H., Ahmad, H.F. and Mori, K., 2011, March. Application for Autonomous Decentralized Multi Layer Cache System to Web Application Firewall. In 2011 Tenth International Symposium on Autonomous Decentralized Systems (pp. 113-120). IEEE.

Tekerek, A.D.E.M. and Bay, O.F., 2019. Design and implementation of an artificial intelligence-based web application firewall model. Neural Network World, 29(4), pp.189-206.

Thomas-Reynolds, D. and Butakov, S., 2020. Factors Affecting the Performance of Web Application Firewall.

Thyer, B.A., 2010. Pre-experimental and quasi-experimental research designs. The handbook of social work research methods, pp.183-204.

Torrano-Gimenez, C., Nguyen, H.T., Alvarez, G., Petrović, S. and Franke, K., 2011, May. Applying feature selection to payload-based web application firewalls. In 2011 Third International Workshop on Security and Communication Networks (IWSCN) (pp. 75-81). IEEE.

Toulas, B., 2022. Misconfigured Meta Pixel exposed healthcare data of 1.3M patients. [online] BleepingComputer. Available at: <https://www.bleepingcomputer.com/news/security/misconfigured-meta-pixel-exposed-healthcare-data-of-13m-patients/> [Accessed 24 August 2022].

Touseef, P., Alam, K.A., Jamil, A., Tauseef, H., Ajmal, S., Asif, R., Rehman, B. and Mustafa, S., 2019, July. Analysis of automated web application security vulnerabilities testing. In Proceedings of the 3rd International Conference on Future Networks and Distributed Systems (pp. 1-8).

Tran, N.T., Nguyen, V.H., Nguyen-Le, T. and Nguyen-An, K., 2020, November. Improving ModSecurity WAF with machine learning methods. In International Conference on Future Data and Security Engineering (pp. 93-107). Springer, Singapore.

Trustwave. 2021. End of Sale and Trustwave Support for ModSecurity Web Application Firewall. [online] Available at: <https://www.trustwave.com/en-us/resources/security-resources/software-updates/end-of-sale-and-trustwave-support-for-modsecurity-web-application-firewall/> [Accessed 28 August 2022].

Vissers, T., Van Goethem, T., Joosen, W. and Nikiforakis, N., 2015, October. Maneuvering around clouds: Bypassing cloud-based security providers. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (pp. 1530-1541).

Voronkov, A., 2020. Usability of Firewall Configuration: Making the Life of System Administrators Easier (Doctoral dissertation, Karlstads universitet).

W3techs.com. 2022. Usage Statistics and Market Share of PHP for Websites, June 2022. [online] Available at: <https://w3techs.com/technologies/details/pl-php> [Accessed 5 June 2022].

Wagner, S., Mendez, D., Felderer, M., Graziotin, D. and Kalinowski, M., 2020. Challenges in survey research. In Contemporary Empirical Methods in Software Engineering (pp. 93-125). Springer, Cham.

Wang, J. and Wu, J., 2019, June. Research on performance automation testing technology based on JMeter. In 2019 International Conference on Robots & Intelligent System (ICRIS) (pp. 55-58). IEEE.

Weir, C., Becker, I. and Blair, L., 2021, May. A passion for security: intervening to help software developers. In 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 21-30). IEEE.

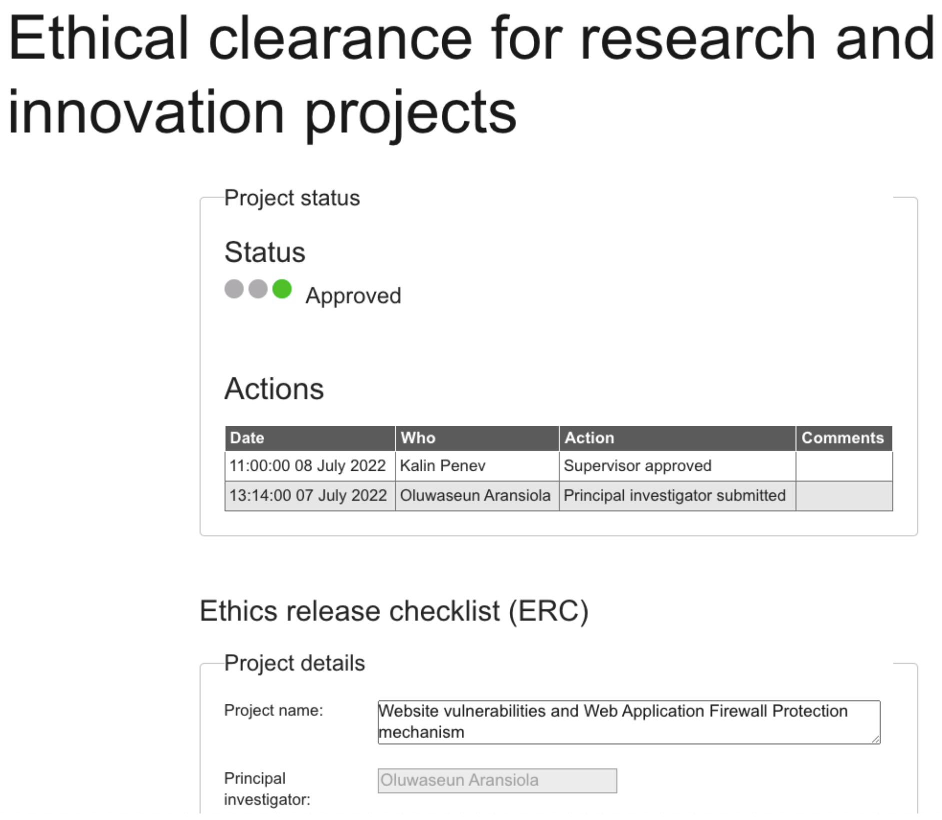
White, J., Drew, S. and Hay, T., 2009. Ethnography versus case study-positioning research and researchers. Qualitative Research Journal, 9(1), pp.18-27.

Whitman, M.E. and Mattord, H.J., 2017. Management of information security. Cengage Learning.

Wichers, D. and Williams, J., 2017. Owasp top-10 2017. OWASP Foundation, 3, p.4.

APPENDICES

APPENDIX I - ETHICS APPROVAL



Ethical clearance for research and innovation projects

Project status

Status

●●● Approved

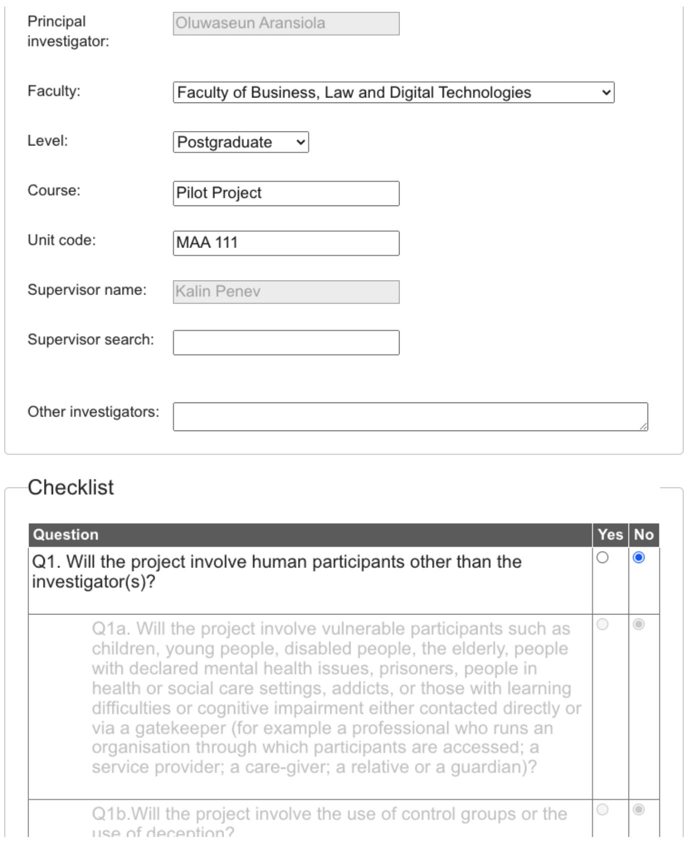
Actions

Date	Who	Action	Comments
11:00:00 08 July 2022	Kalin Penev	Supervisor approved	
13:14:00 07 July 2022	Oluwaseun Aransiola	Principal investigator submitted	

Ethics release checklist (ERC)

Project details

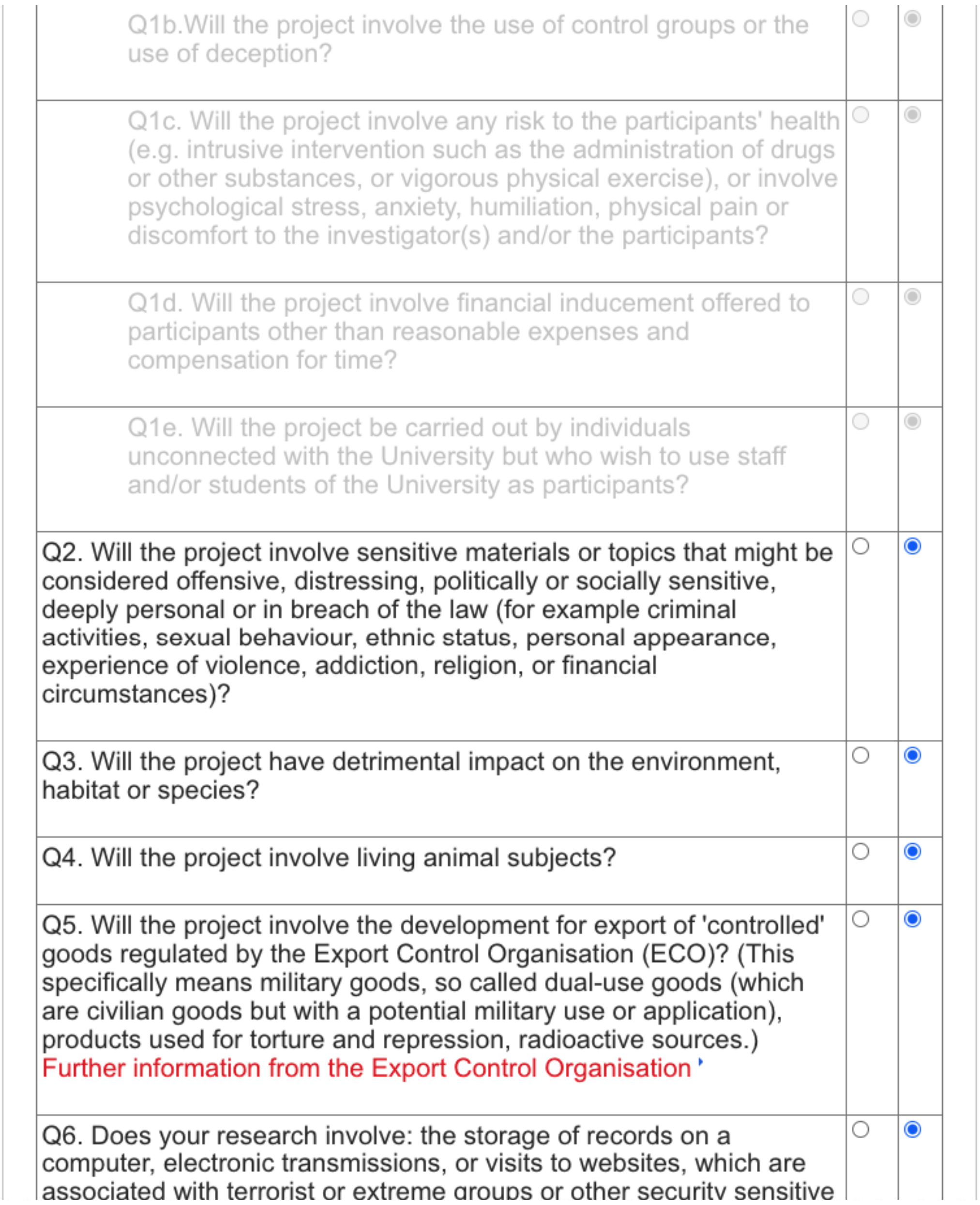
Project name:	Website vulnerabilities and Web Application Firewall Protection mechanism
Principal investigator:	Oluwaseun Aransiola



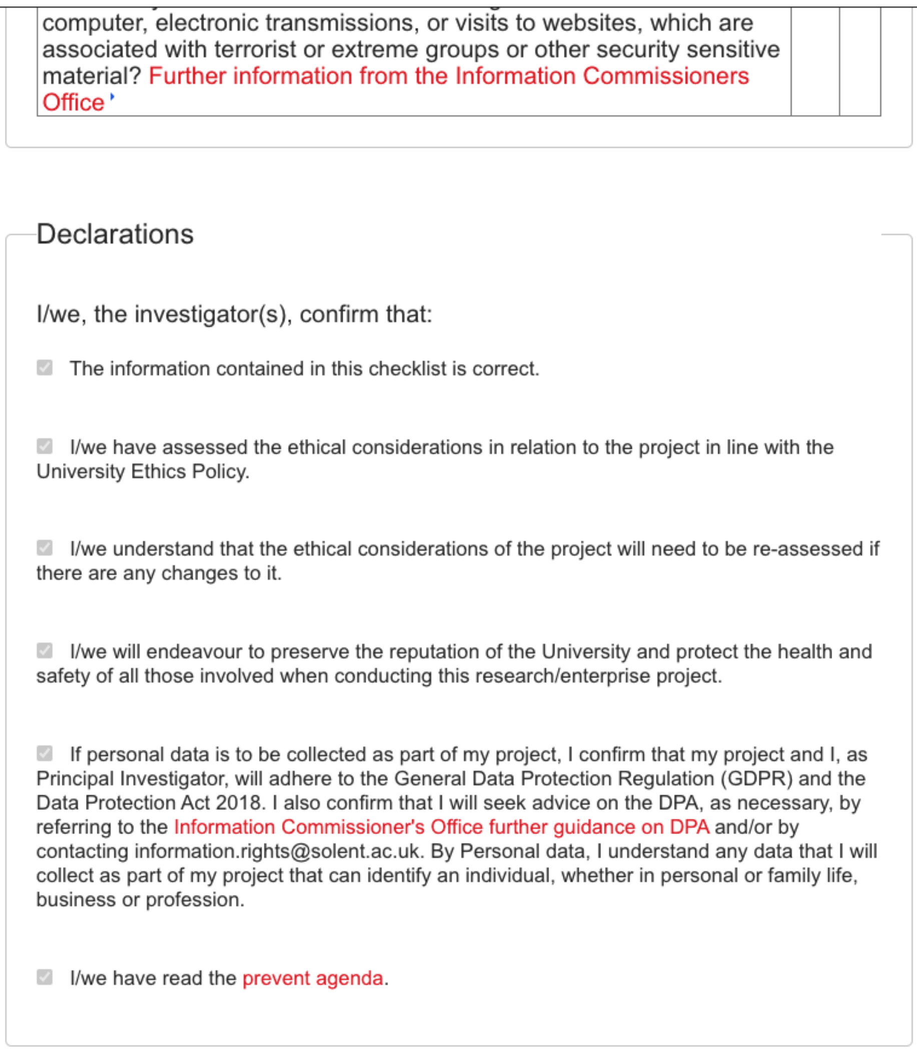
Principal investigator:	Oluwaseun Aransiola
Faculty:	Faculty of Business, Law and Digital Technologies
Level:	Postgraduate
Course:	Pilot Project
Unit code:	MAA 111
Supervisor name:	Kalin Penev
Supervisor search:	
Other investigators:	

Checklist

Question	Yes	No
Q1. Will the project involve human participants other than the investigator(s)?	○	●
Q1a. Will the project involve vulnerable participants such as children, young people, disabled people, the elderly, people with declared mental health issues, prisoners, people in health or social care settings, addicts, or those with learning difficulties or cognitive impairment either contacted directly or via a gatekeeper (for example a professional who runs an organisation through which participants are accessed; a service provider; a care-giver; a relative or a guardian)?	○	●
Q1b.Will the project involve the use of control groups or the use of deception?	○	●



Q1b.Will the project involve the use of control groups or the use of deception?	○	◉
Q1c. Will the project involve any risk to the participants' health (e.g. intrusive intervention such as the administration of drugs or other substances, or vigorous physical exercise), or involve psychological stress, anxiety, humiliation, physical pain or discomfort to the investigator(s) and/or the participants?	○	◉
Q1d. Will the project involve financial inducement offered to participants other than reasonable expenses and compensation for time?	○	◉
Q1e. Will the project be carried out by individuals unconnected with the University but who wish to use staff and/or students of the University as participants?	○	◉
Q2. Will the project involve sensitive materials or topics that might be considered offensive, distressing, politically or socially sensitive, deeply personal or in breach of the law (for example criminal activities, sexual behaviour, ethnic status, personal appearance, experience of violence, addiction, religion, or financial circumstances)?	○	◉
Q3. Will the project have detrimental impact on the environment, habitat or species?	○	◉
Q4. Will the project involve living animal subjects?	○	◉
Q5. Will the project involve the development for export of 'controlled' goods regulated by the Export Control Organisation (ECO)? (This specifically means military goods, so called dual-use goods (which are civilian goods but with a potential military use or application), products used for torture and repression, radioactive sources.) Further information from the Export Control Organisation	○	◉
Q6. Does your research involve: the storage of records on a computer, electronic transmissions, or visits to websites, which are associated with terrorist or extreme groups or other security sensitive	○	◉



computer, electronic transmissions, or visits to websites, which are associated with terrorist or extreme groups or other security sensitive material? Further information from the Information Commissioners Office		

Declarations

I/we, the investigator(s), confirm that:

✓ The information contained in this checklist is correct.

✓ I/we have assessed the ethical considerations in relation to the project in line with the University Ethics Policy.

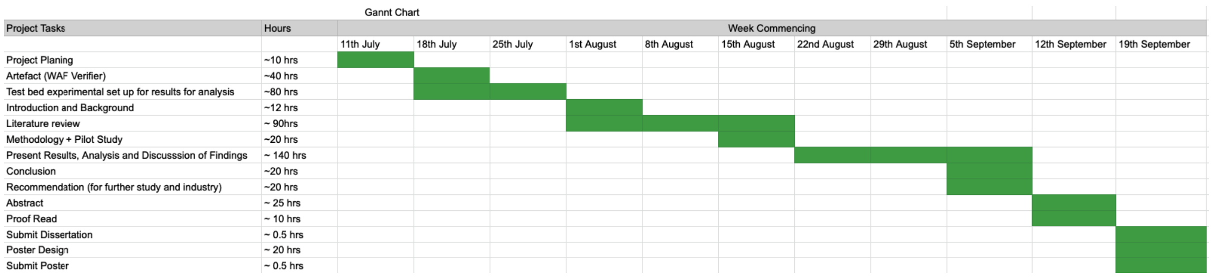
✓ I/we understand that the ethical considerations of the project will need to be re-assessed if there are any changes to it.

✓ I/we will endeavour to preserve the reputation of the University and protect the health and safety of all those involved when conducting this research/enterprise project.

✓ If personal data is to be collected as part of my project, I confirm that my project and I, as Principal Investigator, will adhere to the General Data Protection Regulation (GDPR) and the Data Protection Act 2018. I also confirm that I will seek advice on the DPA, as necessary, by referring to the Information Commissioner's Office further guidance on DPA and/or by contacting information.rights@solent.ac.uk. By Personal data, I understand any data that I will collect as part of my project that can identify an individual, whether in personal or family life, business or profession.

✓ I/we have read the prevent agenda.

APPENDIX II - GANTT CHART

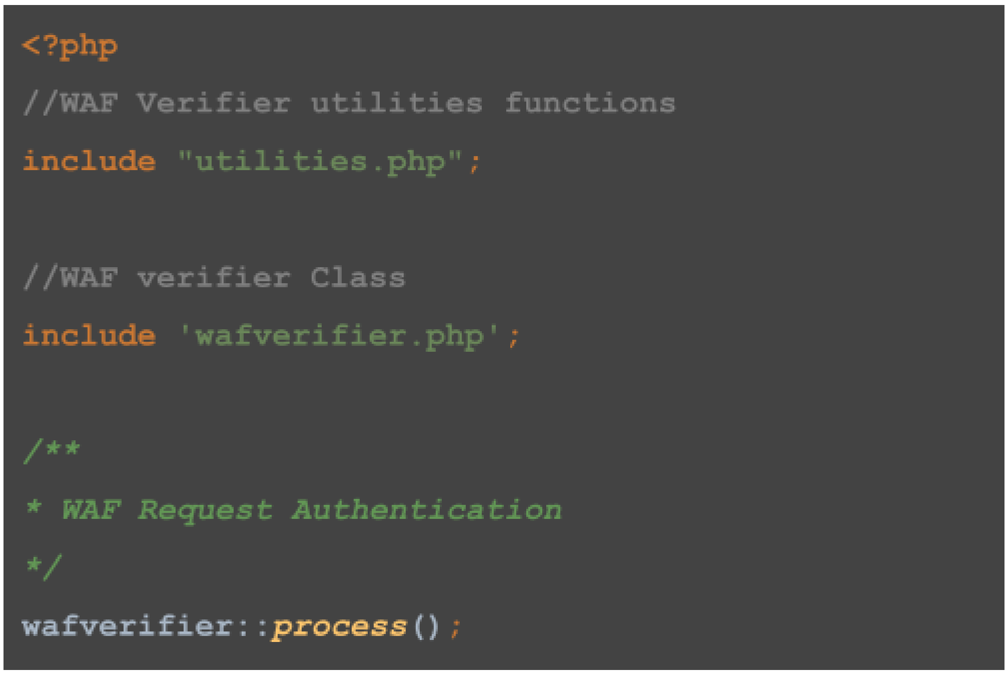


Gannt Chart

Project Tasks	Hours	Week Commencing										
		11th July	18th July	25th July	1st August	8th August	15th August	22nd August	29th August	5th September	12th September	19th September
Project Planing	~10 hrs	🟩										
Artefact (WAF Verifier)	~40 hrs		🟩									
Test bed experimental set up for results for analysis	~80 hrs		🟩	🟩								
Introduction and Background	~12 hrs				🟩							
Literature review	~ 90hrs				🟩	🟩	🟩					
Methodology + Pilot Study	~20 hrs						🟩					
Present Results, Analysis and Discusssion of Findings	~ 140 hrs							🟩	🟩	🟩		
Conclusion	~20 hrs									🟩		
Recommendation (for further study and industry)	~20 hrs									🟩		
Abstract	~ 25 hrs										🟩	
Proof Read	~ 10 hrs										🟩	
Submit Dissertation	~ 0.5 hrs											🟩
Poster Design	~ 20 hrs											🟩
Submit Poster	~ 0.5 hrs											🟩

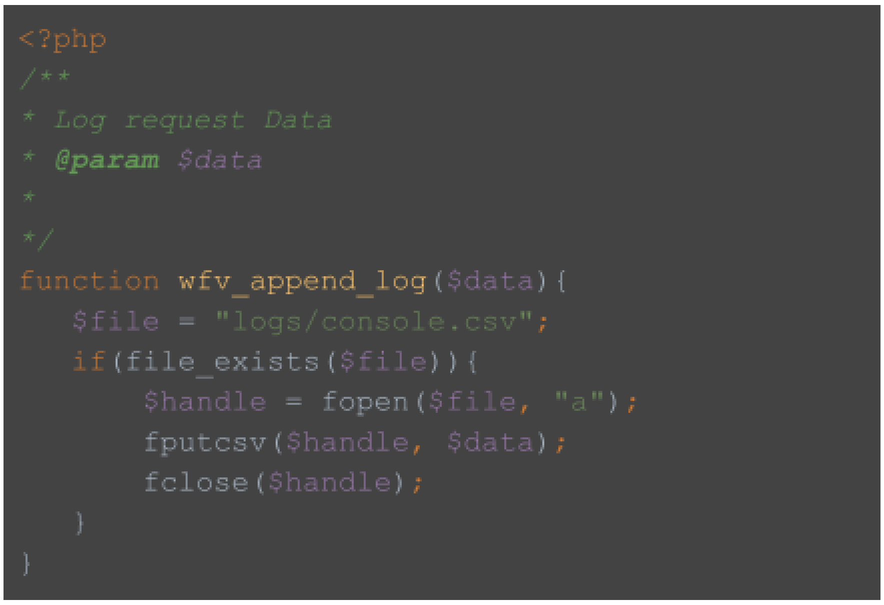
APPENDIX III - WAF-VERIFIER

FILE PATH - /autoload.php

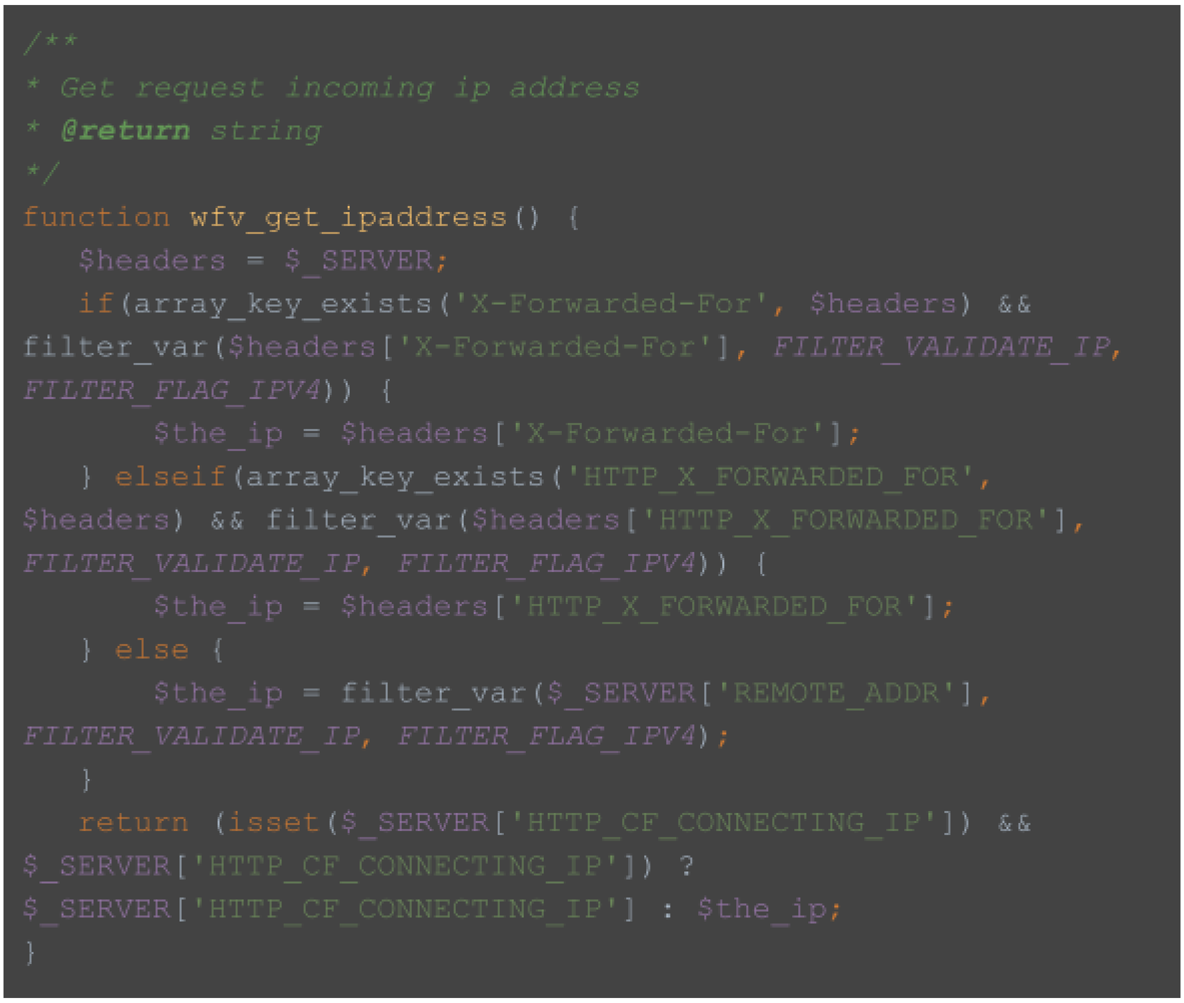


```php
<?php
//WAF Verifier utilities functions
include "utilities.php";

//WAF verifier Class
include 'wafverifier.php';

/**
 * WAF Request Authentication
 */
wafverifier::process();
```

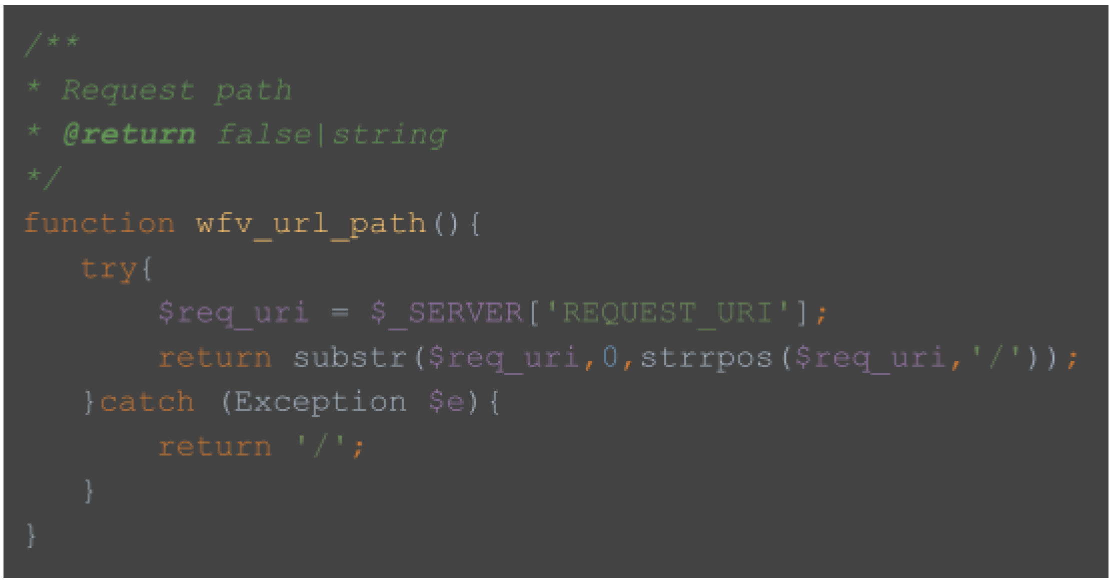
FILE PATH - /utilities.php



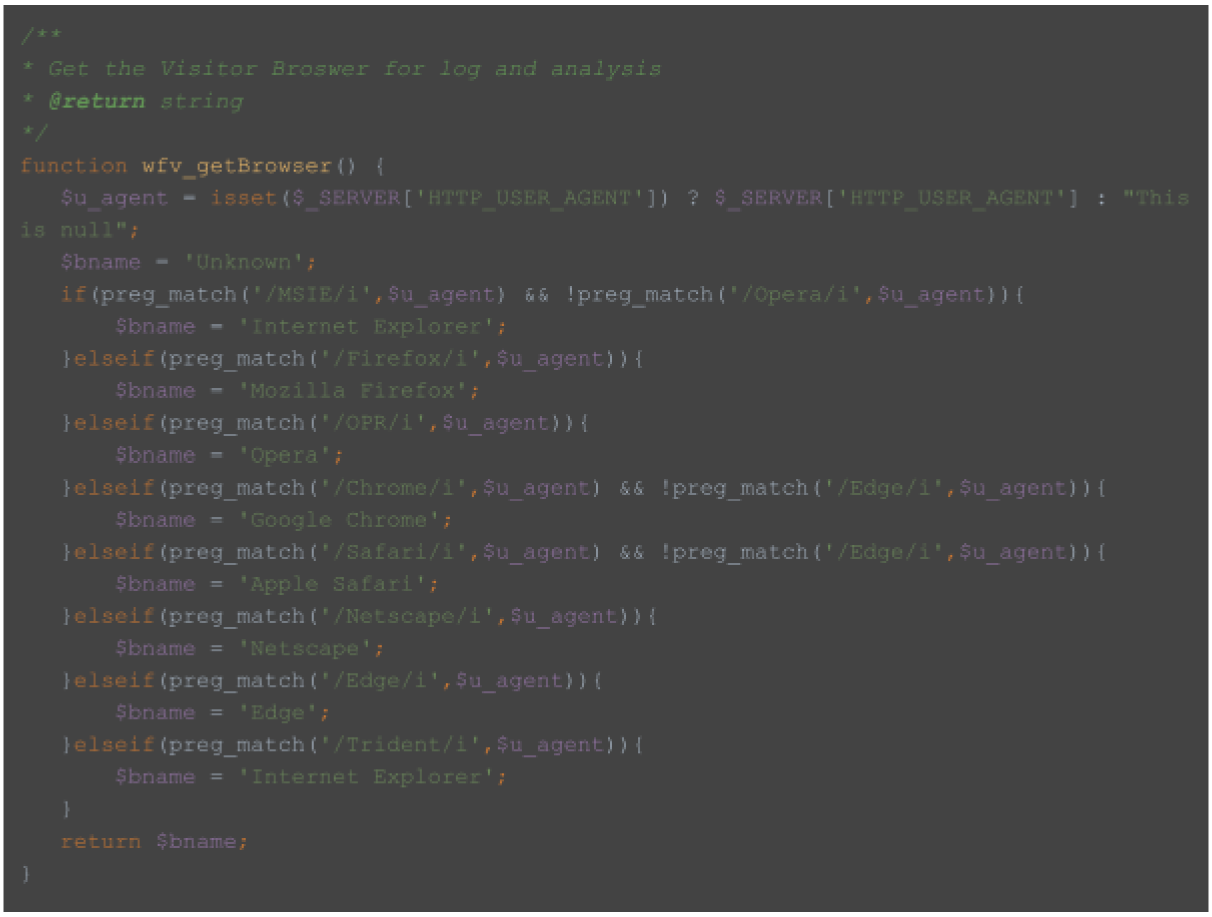
```php
<?php
/**
* Log request Data
* @param $data
*
*/
function wfv_append_log($data){
   $file = "logs/console.csv";
   if(file_exists($file)){
       $handle = fopen($file, "a");
       fputcsv($handle, $data);
       fclose($handle);
   }
}
```



```php
/**
* Get request incoming ip address
* @return string
*/
function wfv_get_ipaddress() {
   $headers = $_SERVER;
   if(array_key_exists('X-Forwarded-For', $headers) &&
filter_var($headers['X-Forwarded-For'], FILTER_VALIDATE_IP,
FILTER_FLAG_IPV4)) {
       $the_ip = $headers['X-Forwarded-For'];
   } elseif(array_key_exists('HTTP_X_FORWARDED_FOR',
$headers) && filter_var($headers['HTTP_X_FORWARDED_FOR'],
FILTER_VALIDATE_IP, FILTER_FLAG_IPV4)) {
       $the_ip = $headers['HTTP_X_FORWARDED_FOR'];
   } else {
       $the_ip = filter_var($_SERVER['REMOTE_ADDR'],
FILTER_VALIDATE_IP, FILTER_FLAG_IPV4);
   }
   return (isset($_SERVER['HTTP_CF_CONNECTING_IP']) &&
$_SERVER['HTTP_CF_CONNECTING_IP']) ?
$_SERVER['HTTP_CF_CONNECTING_IP'] : $the_ip;
}
```



```php
/**
* Request path
* @return false|string
*/
function wfv_url_path(){
    try{
        $req_uri = $_SERVER['REQUEST_URI'];
        return substr($req_uri,0,strrpos($req_uri,'/'));
    }catch (Exception $e){
        return '/';
    }
}
```

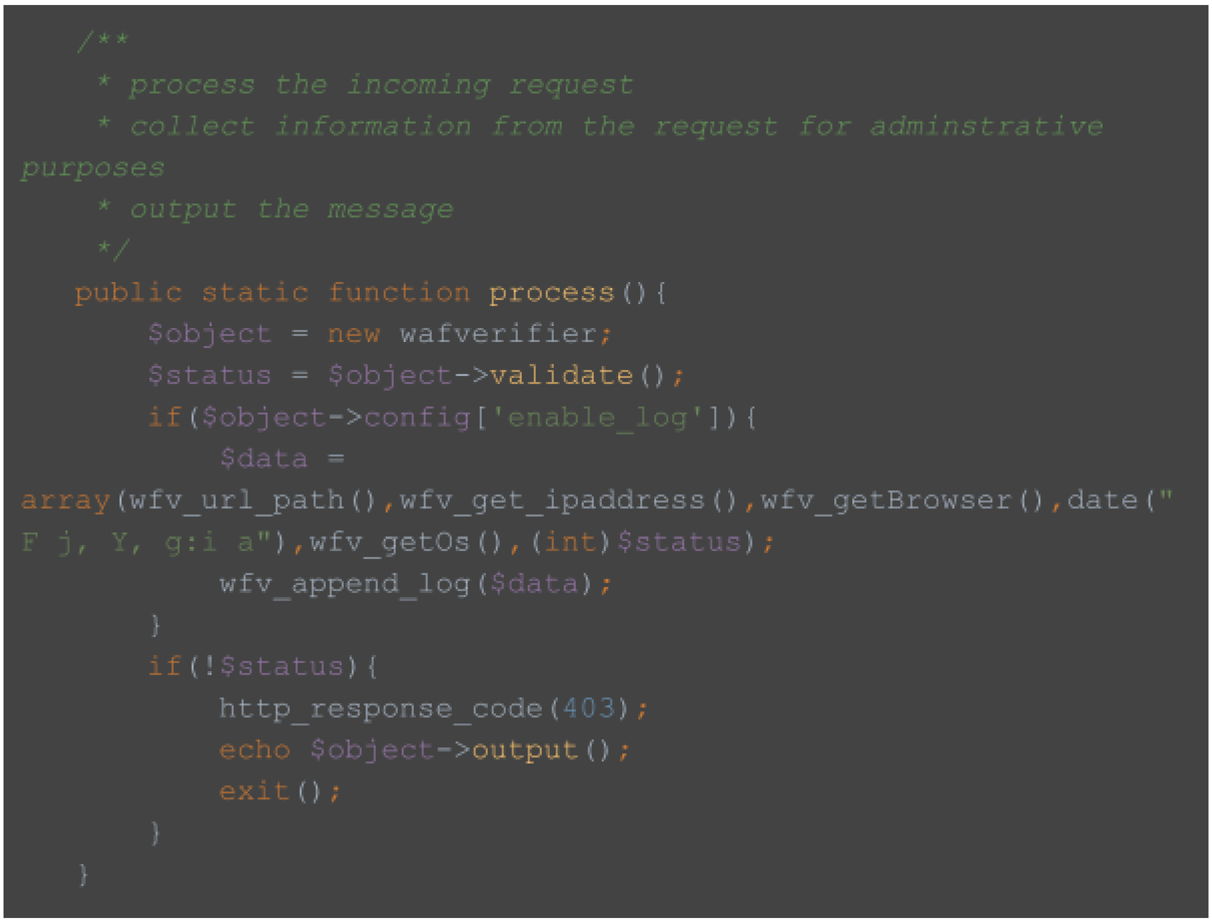


```php
/**
* Get the Visitor Broswer for log and analysis
* @return string
*/
function wfv_getBrowser() {
   $u_agent = isset($_SERVER['HTTP_USER_AGENT']) ? $_SERVER['HTTP_USER_AGENT'] : "This
is null";
   $bname = 'Unknown';
   if(preg_match('/MSIE/i',$u_agent) && !preg_match('/Opera/i',$u_agent)){
       $bname = 'Internet Explorer';
   }elseif(preg_match('/Firefox/i',$u_agent)){
       $bname = 'Mozilla Firefox';
   }elseif(preg_match('/OPR/i',$u_agent)){
       $bname = 'Opera';
   }elseif(preg_match('/Chrome/i',$u_agent) && !preg_match('/Edge/i',$u_agent)){
       $bname = 'Google Chrome';
   }elseif(preg_match('/Safari/i',$u_agent) && !preg_match('/Edge/i',$u_agent)){
       $bname = 'Apple Safari';
   }elseif(preg_match('/Netscape/i',$u_agent)){
       $bname = 'Netscape';
   }elseif(preg_match('/Edge/i',$u_agent)){
       $bname = 'Edge';
   }elseif(preg_match('/Trident/i',$u_agent)){
       $bname = 'Internet Explorer';
   }
   return $bname;
}
```

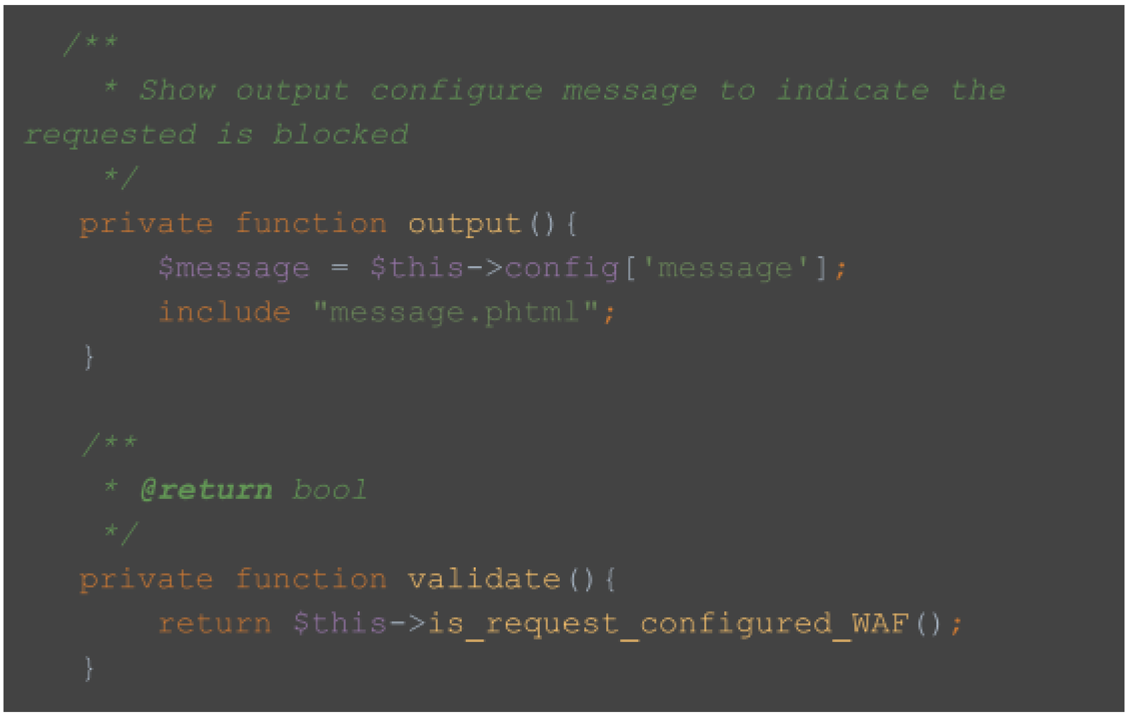
```php
/**
* Get the Operating system of the visitor for Log and
Analysis
* @return string
*/
function  wfv_getOs(){
   $hua = isset($_SERVER['HTTP_USER_AGENT']) ?
$_SERVER['HTTP_USER_AGENT'] : "";
   $os = "Others";
   if($hua == "") return $os;
   if(preg_match('/android/i', $hua)) {
       $os = 'Android';
   } elseif (preg_match('/linux/i', $hua)) {
       $os = 'Linux';
   } elseif (preg_match('/iphone/i', $hua)) {
       $os = 'iPhone';
   } elseif (preg_match('/macintosh|mac os x/i', $hua)) {
       $os = 'Mac';
   } elseif (preg_match('/windows|win32/i', $hua)) {
       $os = 'Windows';
   }
   return $os;
}
```

FILE PATH - /utilities.php

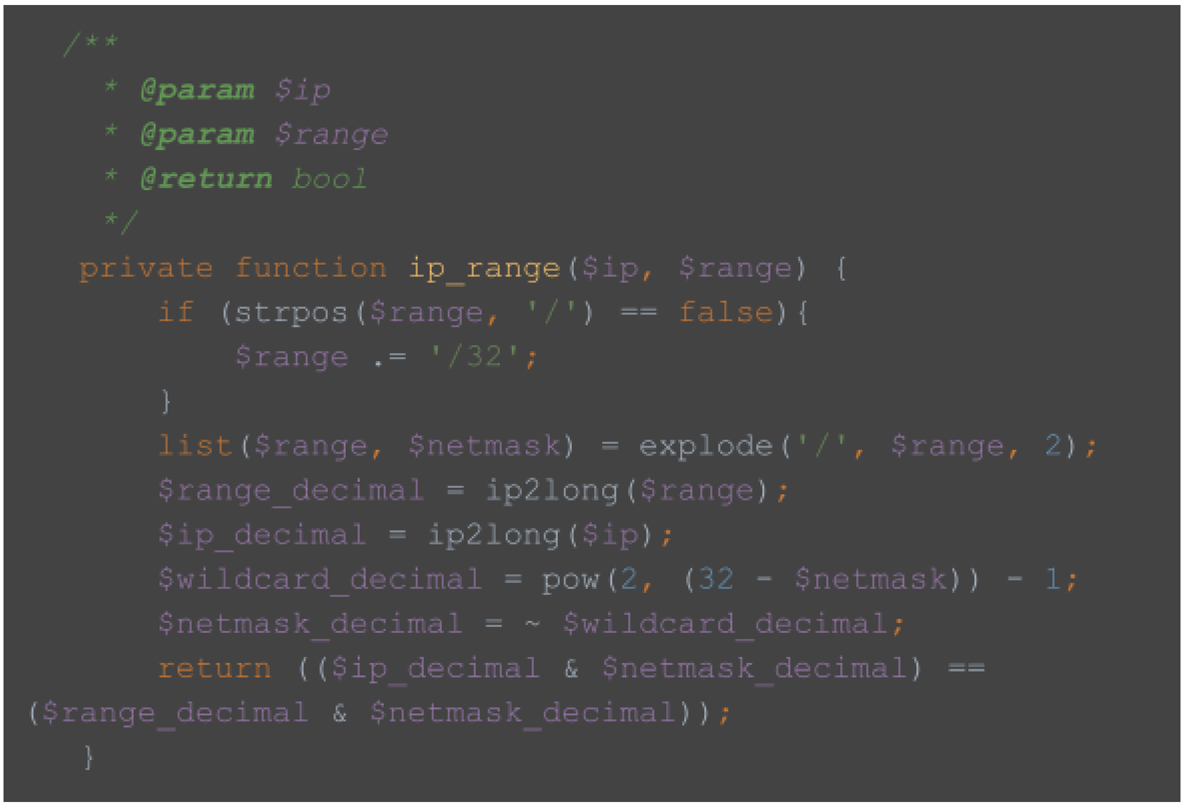
```php
<?php

class WafVerifier
{
   private $config;

   /**
    * Load the Configuration File
    */
   public function __construct()
   {
       $this->config = include 'config.php';
   }
```



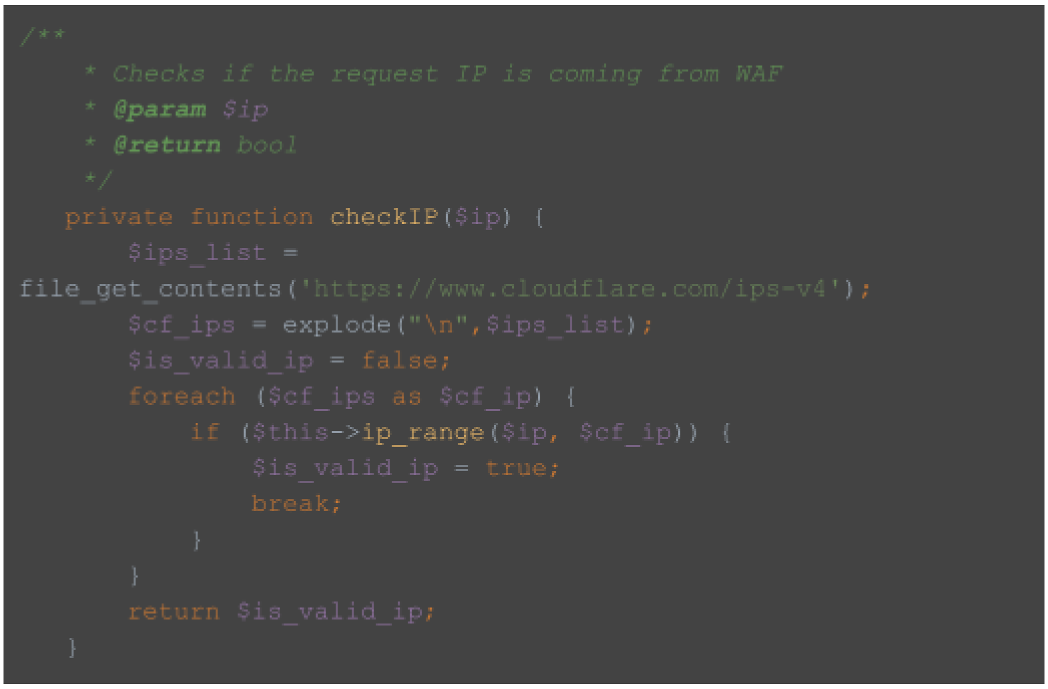
```
    /**
     * process the incoming request
     * collect information from the request for adminstrative
purposes
     * output the message
     */
    public static function process(){
        $object = new wafverifier;
        $status = $object->validate();
        if($object->config['enable_log']){
            $data =
array(wfv_url_path(),wfv_get_ipaddress(),wfv_getBrowser(),date("
F j, Y, g:i a"),wfv_getOs(),(int)$status);
            wfv_append_log($data);
        }
        if(!$status){
            http_response_code(403);
            echo $object->output();
            exit();
        }
    }
```



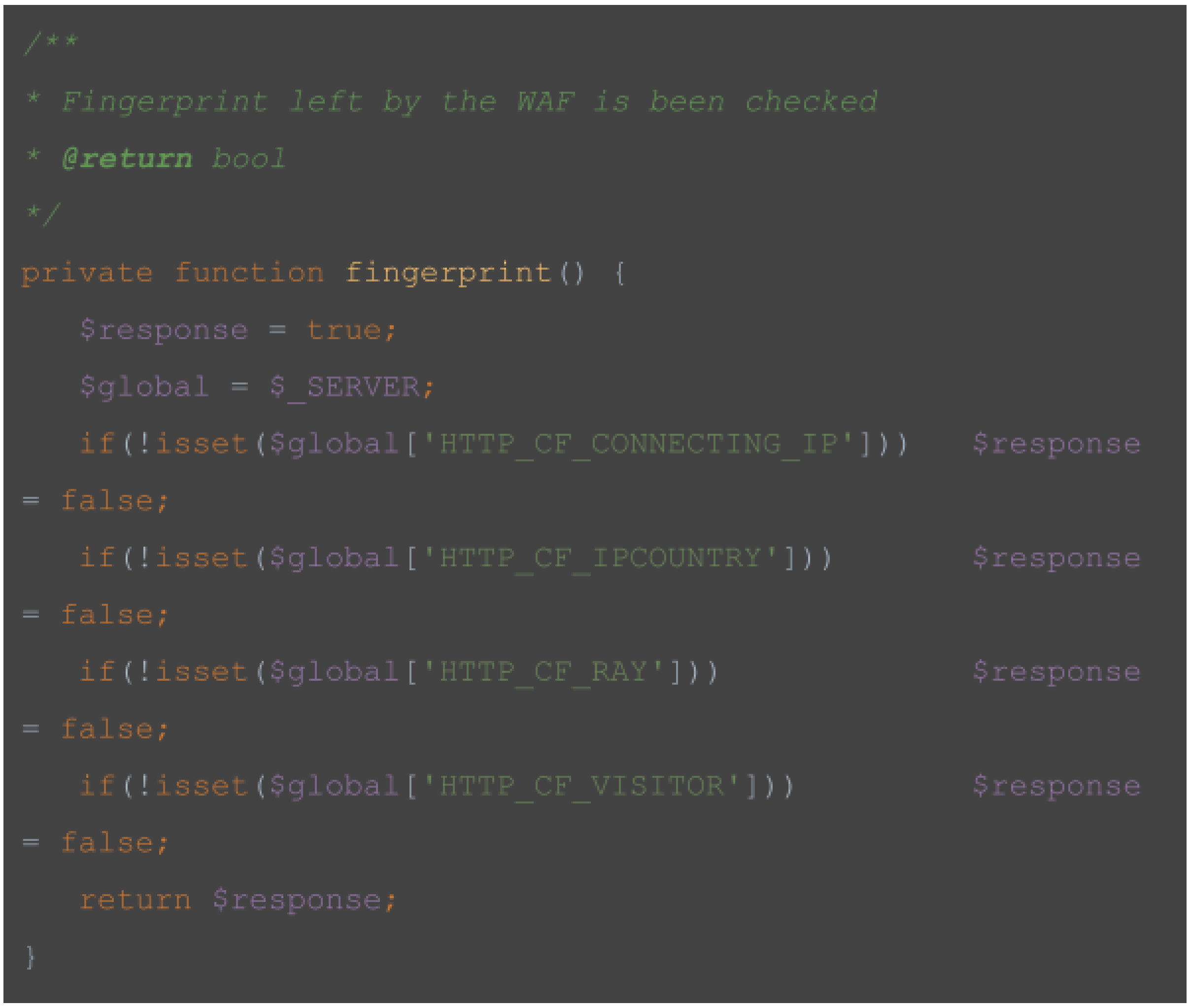
```
  /**
    * Show output configure message to indicate the
requested is blocked
    */
   private function output(){
       $message = $this->config['message'];
       include "message.phtml";
   }

   /**
    * @return bool
    */
   private function validate(){
       return $this->is_request_configured_WAF();
   }
```



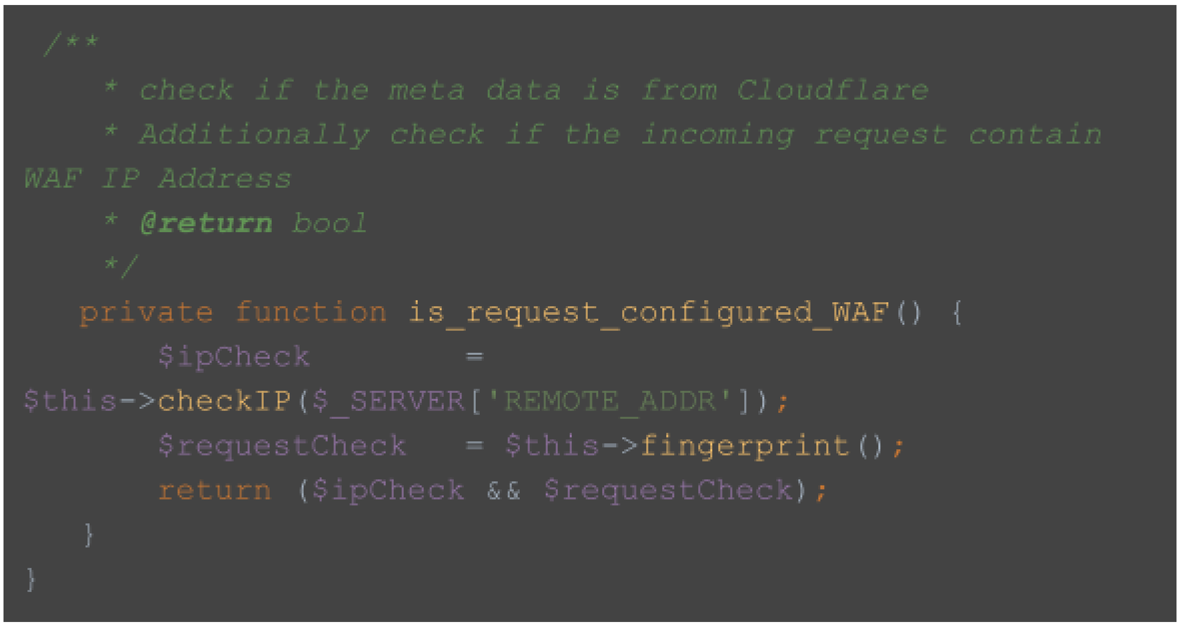
```
 /**
   * @param $ip
   * @param $range
   * @return bool
   */
  private function ip_range($ip, $range) {
      if (strpos($range, '/') == false){
          $range .= '/32';
      }
      list($range, $netmask) = explode('/', $range, 2);
      $range_decimal = ip2long($range);
      $ip_decimal = ip2long($ip);
      $wildcard_decimal = pow(2, (32 - $netmask)) - 1;
      $netmask_decimal = ~ $wildcard_decimal;
      return (($ip_decimal & $netmask_decimal) ==
($range_decimal & $netmask_decimal));
  }
```



```
/**
    * Checks if the request IP is coming from WAF
    * @param $ip
    * @return bool
    */
   private function checkIP($ip) {
       $ips_list =
file_get_contents('https://www.cloudflare.com/ips-v4');
       $cf_ips = explode("\n",$ips_list);
       $is_valid_ip = false;
       foreach ($cf_ips as $cf_ip) {
           if ($this->ip_range($ip, $cf_ip)) {
               $is_valid_ip = true;
               break;
           }
       }
       return $is_valid_ip;
   }
```

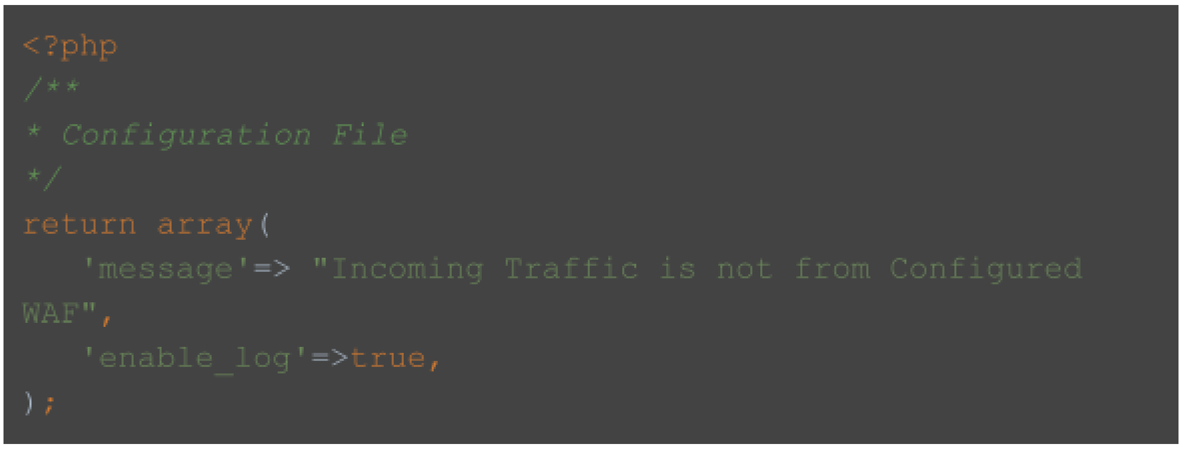


```
/**
* Fingerprint left by the WAF is been checked
* @return bool
*/
private function fingerprint() {
   $response = true;
   $global = $_SERVER;
   if(!isset($global['HTTP_CF_CONNECTING_IP']))   $response
= false;
   if(!isset($global['HTTP_CF_IPCOUNTRY']))       $response
= false;
   if(!isset($global['HTTP_CF_RAY']))             $response
= false;
   if(!isset($global['HTTP_CF_VISITOR']))         $response
= false;
   return $response;
}
```



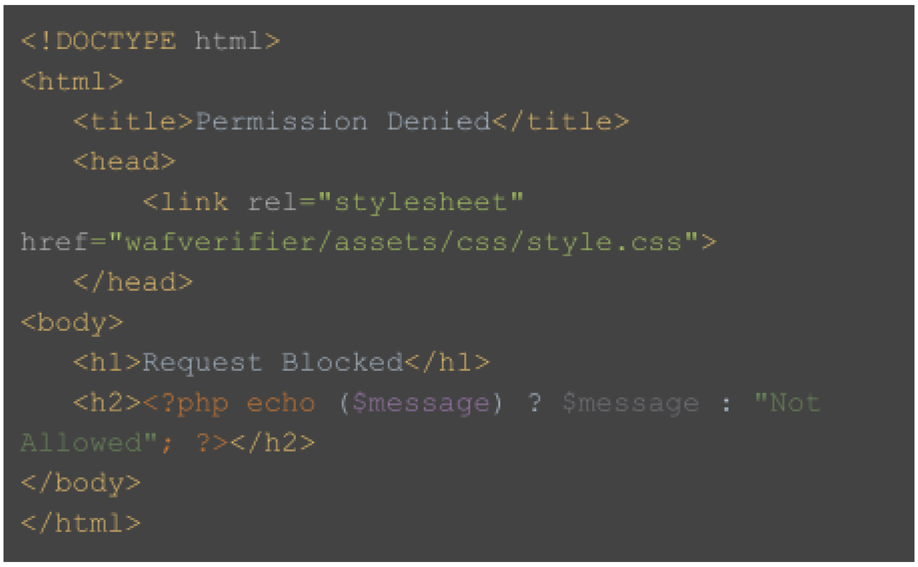
```
 /**
    * check if the meta data is from Cloudflare
    * Additionally check if the incoming request contain
WAF IP Address
    * @return bool
    */
   private function is_request_configured_WAF() {
       $ipCheck        =
$this->checkIP($_SERVER['REMOTE_ADDR']);
       $requestCheck   = $this->fingerprint();
       return ($ipCheck && $requestCheck);
   }
}
```

FILE PATH - /config.php



```php
<?php
/**
* Configuration File
*/
return array(
   'message'=> "Incoming Traffic is not from Configured
WAF",
   'enable_log'=>true,
);
```

FILE PATH - /message.phtml



```html
<!DOCTYPE html>
<html>
   <title>Permission Denied</title>
   <head>
       <link rel="stylesheet"
href="wafverifier/assets/css/style.css">
   </head>
<body>
   <h1>Request Blocked</h1>
   <h2><?php echo ($message) ? $message : "Not
Allowed"; ?></h2>
</body>
</html>
```