An Analytic Approach to Improve Security Features of Web Application using Freeware WAF.

MSc Research Project
MSc In Cybersecurity

Akshay Jadhav
Student ID: X20224630

School of Computing
National College of Ireland

Supervisor: Dr. Arghir Nicolae Moldovan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	AKSHAY SATISH JADHAV		
Student ID:	X20224630		
Programme:	MSc In Cybersecurity	**Year:**	2022-2023
Module:	MSc Research Project		
Supervisor:	Arghir Nicolae Moldovan		
Submission Due Date:	01-02-2023		
Project Title:	An Analytic Approach to improve security features of web application using freeware WAF.		
Word Count:	10166 words		
Page Count:	26 pages		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	AKSHAY SATISH JADHAV
Date:	01-02-2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

An Analytic Approach to improve security features of web application using freeware WAF.

Akshay Jadhav
X20224630

Abstract

In recent years utilization of web applications has increased rapidly in comparison to past decades. Web Applications are being used by various businesses, organizations and on individual level as means of sharing information or for enhancing their business strategies. On the other hand, due to its rapid increase in utilization web applications are more prone to various attacks and threats. So, to overcome from these attacks modern business inculcates various security measures one of which is deploying Web Application Firewall (WAF) to protect the Web application because it is more capable of filtering the packets and blocking vulnerable HTTP packets. This is the focus of the research to implement Web Application Firewall (WAF) and reverse proxy method on web-based application. Second Focus is to compare, analysis and draw conclusion from the different scanner results to predict at what extent Web application firewall (WAF) protect the application from getting attacked. As per the Experiments carried out throughout the research it was seen that Shadow Daemon WAF with its custom rulesets for the web application and reverse proxy method plays an important role in preventing different attacks on web application.
Keywords: Web Application Security, Web Application Firewall (WAF), Shadow Daemon WAF, Reverse Proxy Method, Scanner Tools.

1 Introduction

Cybersecurity is vitally important for providing a high degree of protection in every element of life in this modern period of critical analysis, technological advancement, and completely globalized world of innovation. The hazards and issues related to internet security protocols are also constantly changing. Among the most fundamental and significant ways to prevent cybersecurity is by conducting security testing against vulnerabilities found in Web Applications. As per the survey conducted by SANS and OWSAP about 70% of attacks are being seen on Application layer[1]. Web application overall comprises of web servers and web pages. As per the survey most of the attacks on web application were DDOS attack, SQL injection attack, Cross-site Scripting (XSS) attack, Broken Access control. Amongst this all attacks on web application, the most targeted sectors by intruder were Banking and Finance, Government, Cybersecurity service providers and Education as all work went remotely during COVID. After looking the above-mentioned data in the report, it is very important to improve the security features to overcome the attacks on web application.

[1] https://sansorg.egnyte.com/dl/FrtPvlmWwe

To improve the security features, one solution that can enhance the security features is by implementing Web Application Firewall (WAF). WAF has an ability of packet filtering and blocking dangerous HTTP requests which plays an important role for hacker to attack a web application. Since commercial WAF do not provide protection to Web Application at Application level at that extent and are more expensive to use for the project purpose. So, it is important to use an alternative to it is an open source WAF. There are n-Numbers of open source WAF available in market such as Modsecurity, Shadow Daemon, Ironbee, Naxsi, WebKnight with different features and Security levels associated with it. Shadow Daemon WAF is an effective open-source application security module available for implementing WAF. Shadow Daemon WAF has an easy utilization user interface in which rules can be configured to prevent the web application from getting attacked by threats and attacks. So, keeping the above point in mind we can say that shadow daemon WAF (open source) can be used as an alternative solution to enhance the security features of web applications at low cost.

So, for this project purpose Shadow Daemon open-source web application module for Web Application Firewall (WAF) and Reverse proxy method was used to provide advance level of security to Web application. It has also considered to implement Apache and Nginx server for implementing reverse proxy method as both this server is widely used web server for application deployment as per the survey conducted by Netcraft in 2022[2].I have considered Three Vulnerable Web application and implemented with Reverse proxy and shadow Daemon WAF. After implementing the setup, I have added default and custom rules for the web application in Shadow daemon WAF and used 3 open-source Web application Scanner Tools to scan them at each stage of experimental setup to detect how many attacks are been prevented by proposed solution. It was seen from the research that custom ruleset for WAF plays an important role for preventing Web Application from getting attacked. Following are the contribution and Novelty carried out throughout the project,

Contribution:

- The implementation of the shadow daemon WAF and reverse proxy method provides a comprehensive security solution for web applications, improving the overall security posture of the Web Applications.
- The shadow daemon WAF detects and mitigates web attacks, such as SQL injection, cross-site scripting, and cross-site request forgery, while the reverse proxy method adds an additional layer of protection by hiding the origin server and filtering incoming traffic.
- The implementation is highly configurable and can be adapted to the specific security requirements of each web applications, providing a customized security solution (Customized Rulesets).
- Implementation of various Vulnerability Assessment tools to detect/verify various vulnerabilities in Web Applications at each experimental setup.

Novelty:

- The integration of the shadow daemon WAF (Implemented for first time till now) and reverse proxy method creates a unique solution that offers a comprehensive approach to web application security.

[2] https://news.netcraft.com/archives/category/web-server-survey/

- Unlike traditional security solutions that focus solely on detecting and mitigating attacks, the shadow daemon WAF and reverse proxy method work together to provide proactive security by hiding the origin server and filtering incoming traffic towards Web Applications.

Research Question:

1. How Efficient are the Open Source WAF security solutions available to prevent Web Applications?
   - How Secured are the web application without WAF?
   -How secured are the web application with WAF?
2. How Efficiently Open-Source Scanners can examine vulnerability in Web Applications?

Research Objective:

1. Evaluating Security Features of Web Application without Reverse Proxy and WAF.
2. Evaluating Security Features of Web Application with Reverse Proxy and WAF (default rulesets).
3. Evaluating Security Features of Web Application with Reverse Proxy and WAF (Custom rulesets).
4. Evaluating the Overall results obtained from above setup to draw conclusion how many attacks detected and how many attacks prevented.

The rest of the report is structured as follows, Section 2 of report covers literature review and related work that I have covered throughout my research, section 3 consists of Research Methodology. Section 3 and 4 discuss the design specification and Implementation of the research environment. Section 6 discuss the overall evaluation and case studies that been performed throughout the project. Section 6 and 7 present the discussion about experimental analysis and conclusion respectively.

2 Related Work

In this chapter of research project, it gives the brief description about the topics researched as per project consideration to get an overall execution. The references mentioned would clearly specify the concepts of Firewall, Web Application Firewall, Importance of Reverse proxy method and its implementation work done in past, Deployment of Web Application Firewall to prevent from getting attacked and past analysis done on same, Different types of open-source scanners and its comparison done in the past analysis. All these mentioned concepts are being researched and illustrated in bellow sections.

2.1 Firewall Vs Web Application Firewall

Web applications firewall are used widely by companies to protect their application, they are configured accordingly to requirement. As web application firewall does provide in depth security until and unless they are configured properly. In this research (Clincy & Shahriar, 2018), the author has discussed thoroughly on how configuring WAF can affect the

environment of application and how different WAF's behave. They have mentioned about WAF's which are trained on market to know the input types from users and attacks, later these WAF generate automate rules based on the history of data. The WAF can be configured manually too, but the security professional should have a whole picture of how whitelisting and blacklisting should be configured. As this paper does discuss multiple parameters of configuring WAF but does not have a practical result to confirm these discussed materials.

Several websites have been deployed in recent years which provide multiple services to people using the web application. There has also been increase in attacks and threats to the web application. These types of intrusion can result into data, financial information, and credibility loss the company. In this study (Ghanbari et al., 2016) , the authors have discussed advantages of WAF and comparison of IPS (Intrusion Prevention System) and WAF. They have also mentioned how WAF complies with the best security standards like Payment Card industry Security Standards (PCI DSS) and Open Web Application Security Project (OWASP). Where OWASP has 10 unique approaches to penetration of Web application and methods to prevent such attacks. The paper also discussed how IPS uses methods based on matching traffic and use of signature. The web application structure is way different then applications that uses methods of Traffic pattern (Pattern matching), As the application layer (seventh layer) protection can only be done once the web application structure and features as fully understand. All these features are embedded into Web application Firewall. There are multiple features of security like SQL injection, CSRF protection, Inspection of HTTPS traffic, Web services protection which is not provided by IPS whereas WAF does provide full protection to these services.

As WAF offers security services which are flexible and can be tailored according to application yet are easy to deploy. This research (Pałka & Zachara, 2011), discusses the implementation and deployment of WAF with configuring the parameters by matching them with the traffic pattern generated from user usage. The paper also discusses the common attacks like Script injections, parameter tampering, forceful browsing, Cross-site scripts and many others. It also mentions the difficulty of configuring the WAF, as every aspect of security needs to be covered. As learning WAF by own is not that difficult of a task, but the person is required to be careful of the rules set.

2.2 Web Application Security and Vulnerabilities Associated with It.

Attackers intend to exploit vulnerabilities in web application for accessing the sensitive data. This research (A Survey on Web Application Vulnerabilities and Countermeasures, n.d.), is a survey for security aspects on web application including hacking tools, methods to improve web application and critical vulnerabilities. They have majorly discussed on types of vulnerabilities and fundamentals. Also, the mitigation solutions for the critical vulnerabilities of Web Application. They have highly discussed on how to protect the following web application vulnerabilities: - Injections, Cross Site Scripting, Broken Authentication and Session Management, Insecure Direct Object References, and many others.

The loopholes for the vulnerability are created by web developers who are not well versed with security. In this study (An Analysis and Classification of Vulnerabilities in Web-Based Application Development, n.d.), Web application vulnerabilities during the deployment phase are analyzed and classified to identify the countermeasure, confidentiality, access

complexity, weakness, and severity level of security. The authors have mentioned about the Web development life cycle phases. They have marked the vulnerabilities that occurred in the phases with the parameters for low to critical. The major vulnerabilities had occurred during the implementation phase due to improper techniques for implementation and mitigation. This does gives idea that security just does not start when the application is deployed but it starts when the application is starting to develop for secured web application.

2.3 Past Analysis Done on Web Application and WAF to prevent from attacks.

As multiple business-related tasks are supported by the web applications for multiple organizations. In this paper (Muzaki et al., 2020), the authors have demonstrated the implementation of Reverse proxy method for web application and use web application firewall – ModSecurity. Modsecurity is an open source WAF and can detect and preventing attacks. The reverse proxy stands as a first filter for attacks as the request from client does go to reverse proxy first then it goes to web application. The author has tested infrastructure setup for Cross site scripting, Unauthorized Vulnerability web scanning and SQL injection, Modsecurity was successfully in protecting these attacks.

2.4 Implementation of Reverse Proxy Method

In this study (Arnaldy & Hati, 2020), to optimize the web server and maintain security Reverse proxy and WAF has been used. They have used Nginx as reverse proxy and ModSecurity as WAF for the infrastructure. The Waf was tested for SQL injection, Local file Inclusion, Remote File Inclusion and Cross site scripting. The Reverse proxy has successfully applied for optimization and tested for services such as transfer time, connection time and request time. Also, the author has configured the attacks on Web application to be notified from sys logs, for this TOKEN bot was used. For optimization the author could also have used a high availability proxy or Load balancer.

In research (Valeur et al., 2006), the security divided into sensitive and non-sensitive parts with approach of web application with reverse proxy for anomaly detection system. In the Reverse proxy settings with anomaly detection system, it calculates score. If the score is below threshold the request is not suspected to be suspicious and forwarded to web server. The design is quite effective for routing limited requests to servers that have sensitive information. The system was analyzed in real time. For analyzing the approach, they had a PHP analyzer for detecting the metric involving the paths to databases with critical data in comparison to standard web application available in market.

2.5 Implementation of WAF to Improve Security Features of Web Application

Worldwide cyber-attacks and data breaches have increased in numbers. In this study (Kiruba et al., 2022), they have implemented Waf majorly focusing on the Application layer. They have also used a prebuild Application Programming Interface to analyze the incoming and outgoing request. With addition to this they have used an application known as Detectify which provides deep layer security to users. This methodology has increased the security standards for transactions and accessing information.

As major attacks take place on Web application. It has become a necessary step to protect web application not only through network firewall but also thought other means. In this research (Razzaq et al., 2013), different WAF solutions are compared on necessary features. The authors have discussed about Mod Security, Imperva's Secure Sphere, Barracuda network application gateway, Breach Security's Web Defined, F5-Big IP, Web Sniper, I-Sentry, Secure IIS, Web Defend, Anchiva, Profense, Citrix, WebApp secure and Server Defender AI are some WAF and other systems which are compared and explained. As with critical analysis all the Web application firewalls limitations were highlighted in the paper. All Waf have different advantages which can be used for protecting the Web application accordingly.

This research (Lewandowski et al., 2020) , have used Spider trap 5 months activity for honeypot on accessible by two domains and their IPs. In this paper the author has shortly discussed about shadow daemon and how shadow daemon ability to learn the attacks to set rules, as all the requests are logged for honeypot (can be a web application) working behind. It also has ability to separate the request using the whitelist and blacklist rules. Some rules can be manually set in shadow daemon, also pre-defined rules can be used to protect the application too.

2.6 Importance of Automatic Testing (Scanners) over manual testing.

As there are multiple ways to protect a web application but how can someone confirm the application are protected and checking each vulnerability one by one is a tedious task to do. Companies had come up with scanner for vulnerabilities of web application. In this study (Makino & Klyuev, 2015), they have used 2 scanners – Skipfish and OWASP ZAP for detecting the vulnerabilities of DVWA and WAVSEP. The major vulnerabilities focused in here are Cross Site Scripting, SQL injection and File inclusion. The vulnerabilities were evaluated on basis of high, medium, low and informational. The paper concluded that OWASP ZAP is far better than Skipfish according to the experiments done. Both were not able to detect much of RFI vulnerabilities in application.

In this research (Fong & Okun, 2007), they have mentioned and defined Web application and its tools, and vulnerabilities of security related to web application. They have also discussed about the functions of web application scanners. The paper has also defined web application and web application scanner with how they function. Some of the commercial web application scanner like Appscan, webking, webinspect and NTOspider have been stated with functional requirement for web application scanner. In largely discussed the issues which are there while testing with a scanner.

In this study (Amankwah et al., 2020) , they have compared 3 commercial and 5 open source WAF – IBM AppScan, Skipfish, Iron WASP, Acunetix, OWASP ZAP, Vega, HP Webinspect and Arachini for their capabilities of detecting vulnerability. They were tested on 2 vulnerable applications DVWA and WebGoat. The capabilities of Web application scanners were evaluated on basis of OBE (OWASP web benchmark), WASSEC, Youden index, recall and precision. The paper discusses about the vulnerability type like Denial of Service, Buffer Overflow, Authentication Flaws, Code Execution, Cross Site Request Forgery, SQL injection and others. The paper concludes that the commercial WAF are effective at detecting the vulnerabilities but also Some open-source scanners are equally efficient.

In this research (Mburano & Si, 2019), there has been comparison of OWASP benchmark results with the existing results of WAVSEP. The paper has chosen OWASP benchmark since it's an open-source program and regularly updated and has number of contributors. OWASP ZAP and ARACHINI were compared for results of Command Injection, LDAP injection, SQL injection, Cross Site Scripting and Path Traversal. The paper concluded that, both the scanners performed different in many sectors therefore neither of them can be considered as an all-rounder but can be used according to the requirement of our security purpose. Also, OWASP ZAP does provide better results than Arachni in basis of XSS, CMDI and SQLI.

3 Research Methodology

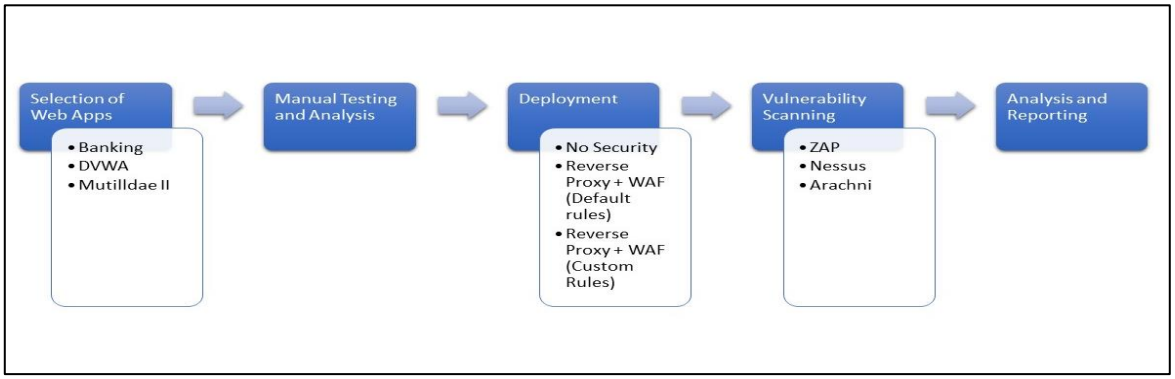

Figure 1:Research Methodology

In this chapter of research, important aspects of the project are being discussed in detailed steps regarding the methodology that have followed throughout project plan. The project plan compromises of 5 stages and which had made me evaluate my results based on the same. The Methodology has been illustrated in Figure 1.

3.1 Selection of Web Apps

For this project various vulnerable web application available on internet such as DVWA, Mutillidae II, etc and got to know all the ground information from the same. After researching a lot, it was seen that many individuals all over the world have already done penetration testing on the above applications and reports are already available for the same. So, taking all the parameters into account one random PHP web application was selected from sourcecodester.com which was Global online Banking system which was a vulnerable application. So, I decided to consider the same application and move further with my project plan and Deployed three Web Applications for the research. Table 1 illustrates the three Web Applications with there Versions used for project and its Language.

Table 1: Web Application Selection

Web application	Version	Language
Global Online Banking System[3]	N/A	PHP/MYSQL
Damn Vulnerable Web Application (DVWA)[4]	2.0.1 (latest)	PHP/MYSQL
QWASP Mutillidae II[5]	2.10.8 (latest)	PHP/MYSQL

[3] https://www.sourcecodester.com/php/14868/banking-system-using-php-free-source-code.html
[4] https://sourceforge.net/projects/dvwa.mirror/
[5] https://owasp.org/www-project-mutillidae-ii/

3.2 Manual Testing & Analysis

After selecting the appropriate Web Applications for my research project then it was decided to manually test the Web Applications. Since due to less time span for completing the evaluation of project I decide to manually test only Banking Application and analyse it in depth to find appropriate results. So, for manual penetration testing on Banking Application using open-source web application security testing tool called as Burp suite has been used. Basically, Manual Testing is done on the Banking application to understand two cases. First to cross verify the scanner results by performing the attacks on same to check whether the same attacks are being detected. Second is to check the PATH and CALLER parameters to create new rule set to protect the application from different attacks that are been detected.

3.3 Deployment

After successfully completing the manual test in previous step, then it was decided to deploy the Web Applications as per the plan to evaluate whole project successfully. Then Web Applications were deployed under three experimental setups such as Web Application with No Security, Web Application with Reverse Proxy +WAF (Default rules) and Web Application with Reverse Proxy +WAF (Custom rules). Each of them has been explained in brief below.

3.3.1 Web Application with No security

In this step of deployment, after successfully selecting the Web Application for research the next step was deployment of Web Applications over web. So, to do that I have deployed selected Web Applications over web using XAMPP.

XAMPP for Apache and MYSQL[6]

XAMPP is basically a well know cross platform web server which helps developer to develop and deploy their application on web server. The control panel of XAMPP server which allows the web application to host on Apache port and by adding its database file with help of phpMyAdmin panel on web. This port numbers can be changed depending on which port you want to deploy the web application by just changing httpd.config file for Apache.

In this manner for project purpose 3 selected Web Applications over web was deployed using XAMPP and No security features were added to protect the web Applications.

3.3.2 Web Application with Reverse Proxy + WAF (default rules)

In this step of deployment, after successfully deploying the Web Applications over web using XAMPP, the main aim of this research was to provide two levels of security to the Web Applications from getting attacked. First is by implementing Reverse Proxy and second by deploying Shadow Daemon WAF.

Reverse Proxy Setup

For project purpose web applications was deployed on Apache server on port 5000 and another server called as NGINX server on port 3000 and also set the parameters in both server configuration files which makes Nginx server to act as a proxy server for Apache. This is the first level of security to application as whatever the traffic coming on Apache server will first go to the proxy server (NGINX) and then it will transfer the request to Apache server.

Shadow Daemon WAF installation with default rules

[6] https://www.ionos.com/digitalguide/server/tools/xampp-tutorial-create-your-own-local-test-server/

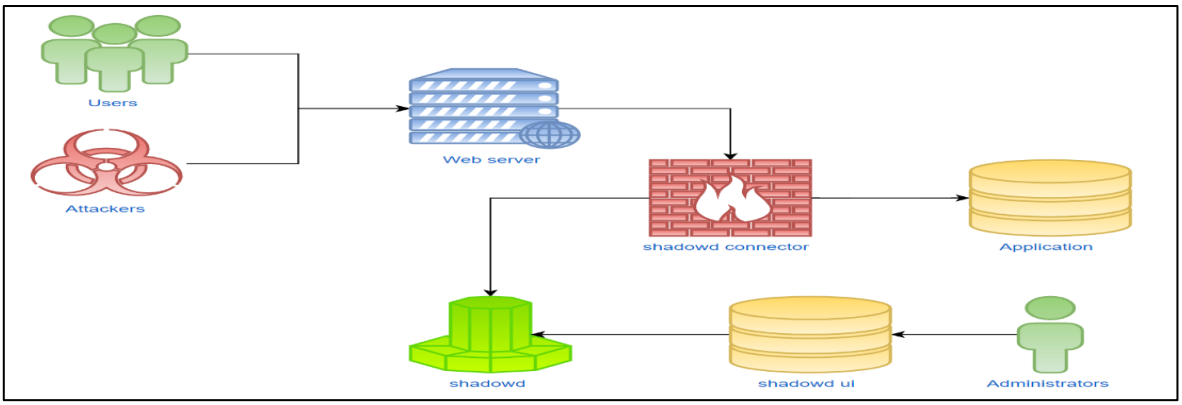

Figure 2 : Shadow Daemon Working Setup[7]

After successfully implementing reverse proxy, now it's time to provide second level of security protection to web application by deploying shadow Daemon WAF. So, referring the official document of shadow daemon WAF[8] open-source shadow daemon WAF was deployed by cloning its repository from GitHub and setting up user credentials and created profile for my web application. To connect web applications and shadow daemon WAF PHP connector has been installed which records all the request coming in and out to the web application. The default rules for protecting shadow_ui and common headers are available on GitHub which was imported into Shadow Daemon WAF [9].The Figure 2 illustrates the setup and working of Shadow Daemon WAF.

3.3.3 Web Application with Reverse Proxy + WAF (Custom rules)

In this step we will be using the same setup which we have deployed above for Web Application with Reverse Proxy +WAF for our research with default rulesets. Now here comes the main part to provide second level of security to Web Application. For this experimental setup the default rule set were removed and then applied with new rule set depending upon the result obtained from manual testing and scanner results. Then started creating the new rule set by referring the PATH and CALLER value obtained from Manual Testing and same detected on the shadow daemon WAF.

3.4 Vulnerability Scanning

After successfully deploying the Web Applications under various Experimental setup, now it was important to select Various Web Application Vulnerability Scanner tools. While conducting the research my main objective was to select the scanner tools which are well known in cybersecurity industries. For project various SAST and DAST open-source tools were researched, and three open-source tools were selected depending on past analysis done on OWASP Benchmark to scan Vulnerabilities in my web applications. All the selected tools are the proxy tools which have scanning tools built right into them and it can also be used as a web proxies. It provides the precise control over the request response interaction of the selected target. These tools prove to be more efficient for the penetration tester to manually test the application as it easily scans the post login queries. The selected tools for the research are being illustrated in Table 2 with different parameters.

[7] https://shadowd.zecure.org/documentation/architecture/
[8] https://shadowd.zecure.org/overview/introduction/
[9] https://github.com/zecure/shadowd_rules

Table 2:Evaluated Scanner Details

Tool Name	Version	License	Tool type	Price
OWSAP Zed Attack Proxy[10]	2.12.0	Apache License v2.0	Proxy	Free
Nessus Essentials[11]	10.4.1	Proprietary Tenable, Inc	Proxy	Free
Arachni[12]	1.6.1.3	Arachni Public Source License v1.0	Scanner	Free

The first selected tool is OWASP ZAP for which **zap.sh** file was downloaded for Linux installer and then provided the suitable permission and installed ZAP in system and then by setting up the proxy scanned the web application by giving the parameters to zap and started automatic scan to find Vulnerabilities in application.
The second selected tool is Nessus in which the same Nessus for Linux installer file was downloaded and then after created an account with the authentication code match and then scanned the web application by giving an appropriate parameter of the application.
The third selected tool is Arachni which is an open-source scanner using Arachni. For which the Arachni Linux installer file were downloaded and scaned the web application by giving specific parameters of the web application to perform scan.
When this scanner tools were scanned on Web Applications at each experimental setup it generates a scan report which displays the vulnerability present into web applications according to their severity. These generated reports are available in HTML and XML format which can be downloaded and saved for further research work. Table 3 illustrates the severity output table of the selected scanners for research.

Table 3:Severity Table for Scanners

Severity	OWSAP ZAP	Nessus Essentials	Arachni
Critical	×	✓	×
High	✓	✓	✓
Medium	✓	✓	✓
Low	✓	✓	✓
Informational	✓	✓	✓

3.5 Analysis & Reporting

This is basically the last step of my methodology, in which the overall obtained results of the selected scanners were analysed when tested under different experimental setup. This help to draw the conclusion of the evaluation to restates the objective of the research as mentioned below,

i) Drawing conclusion by referring scanning results from different scanners at each experimental setups and manual testing, how many attacks are been prevented without using WAF and after using reverse proxy and WAF.
ii) Drawing conclusion by referring scanning results to illustrate the importance of custom rulesets in preventing attacks.
iii)Drawing conclusion how efficient open source WAF detects vulnerability in web applications.

Considering the above factors into account, it helps in reporting the overall results and conclude our work precisely in this step.

[10] https://www.zaproxy.org/
[11] https://www.tenable.com/products/nessus/nessus-essentials
[12] https://www.arachni-scanner.com/

4 Design Specification

The Following Figure 3 illustrates the complex design specification of research work. In this chapter it has been explained the important components of research project and their specification required for implementation work.

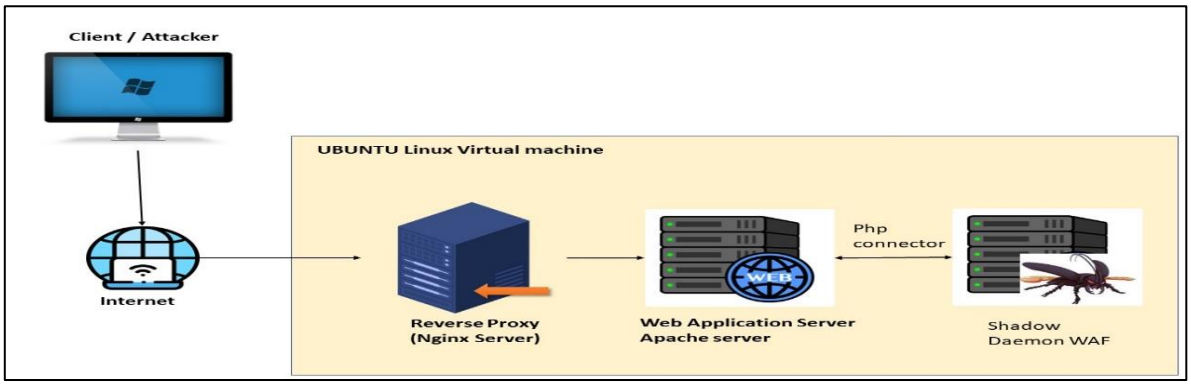

Figure 3:Network Architecture

4.1 VMware Workstation Pro and Ubuntu Linux

To start with the research, work the important factor need to launch the project is deploying Linux based operating system. For this project, one open-source software called VMware workstation pro 16.2.4 version was used, and which is configured in the available windows operating system. It was used because it allows the use of multiple operation system on single host machine which is most reacquired for my research project. Considering all the parameters for the project UBUNTU Linux machine has been choose for the project. It is on open source and consist of various built-in functions which make it feasible for my project. It has been configured as a virtual machine in VMware Workstation Pro with the following configuration as mentioned in Table 4. The Linux machine is then powered on and all the necessary software's and updates are being made which all are reacquired for the research project.

Table 4:Ubuntu Configuration

Ubuntu Linux	Description
Version	V22.04.1
Processors	4
RAM	4GB
Hard disk (SCSI)	40GB

4.2 XAMPP server for Apache and MYSQL

To deploy the web application on the Apache server and to add web application database XAMPP server has been used. XAMPP 8.1.12 Version has been used. As default Apache server get install at localhost (127.0.0.1) at port number 80, but for the project requirements I have shifted working of Apache server from port 80 to port 5000 by making changes in **httpd.config** file for Apache and then added Web applications database using PhpMyAdmin and got my web application running on port 5000.

4.3 NGINX server

To provide first level of security to Web Application deployed on Apache server I have installed NGINX server[13] in ubuntu machine through command line. Since the installed

[13] https://docs.nginx.com/nginx/admin-guide/installing-nginx/installing-nginx-open-source/

NGINX server get default deployed on port 80, but as per the project requirements it was made changes in the NGINX configuration file called **sites-available>default** and changed its running port from port 80 to port 3000. NGINX server v 1.18.0 has been installed for this project.

4.4 Reverse Proxy:

This is the method to provide First level of security to Web Application by establishing Reverse proxy method between Apache and Nginx server[14] each running on different ports. Since the main motive of reverse proxy is to transfer all the traffic coming to Apache server should first hit Nginx server and then it will transfer to Apache server to prevent various attacks from intruder. For Example, if N number of request are been hitting on web by an intruder to gain access to Apache server here reverse proxy plays an important role where NGINX server will act as a proxy server for Apache and it will validate all the request coming through it and then pass only those request which are validated and reject all the invalidated requests and give client 403 forbidden error and also take the result from the validated request from Apache server and send back to the appropriate client. So, in this case client will never know the real IP address of the Apache server and which will on other way prevent the attacks. Prevention of attack takes place in this method as NGINX will act as a proxy server for Apache server just by hiding its original IP address. This reverse proxy has been configured by making changes in the nginx configuration file and had passed the proxy_pass parameters for Apache server.

4.5 Shadow Daemon WAF implementation with PHP connector:

To provide second level of security to the web application by preventing it from getting attacked shadow daemon WAF has been implemented. Shadow Daemon WAF is basically an open source WAF and has been deployed to protect the application running on Apache port. For this GitHub repository of shadow daemon WAF has been cloned and deployed WAF on port 8080 and created profile for my web application. The installed shadow daemon has version 2 which was released under GNU General Public License. To connect my web application to shadow daemon WAF PHP connector has been downloaded for WAF which has PHP version 2.2.0. The main specification is that you need to make changes to Apache server to connect shadow user interface to web application deployed on Apache server which will track all the request coming in and out through web application which is been done by me in this setup.

4.6 Scanners for Web Application Vulnerability Assessments:

To overcome your manual work and to have strong proof about the detected attacks different scanners has been used. For this project Three open-source scanners named as OWASP ZAP, ARACHNI and NESSUS were used and had run this scanner all over at different stages of the evaluations to get down the results.

5 Implementation

As discussed in the previous section regarding the Design Specifications of the parameters and software's used to run the project. Now lets us discuss how it has been used them in the project and how they are implemented. The complex Implementation process is illustrated in Figure 4.

[14] https://www.atatus.com/blog/how-to-set-up-a-reverse-proxy-in-nginx-and-apache/

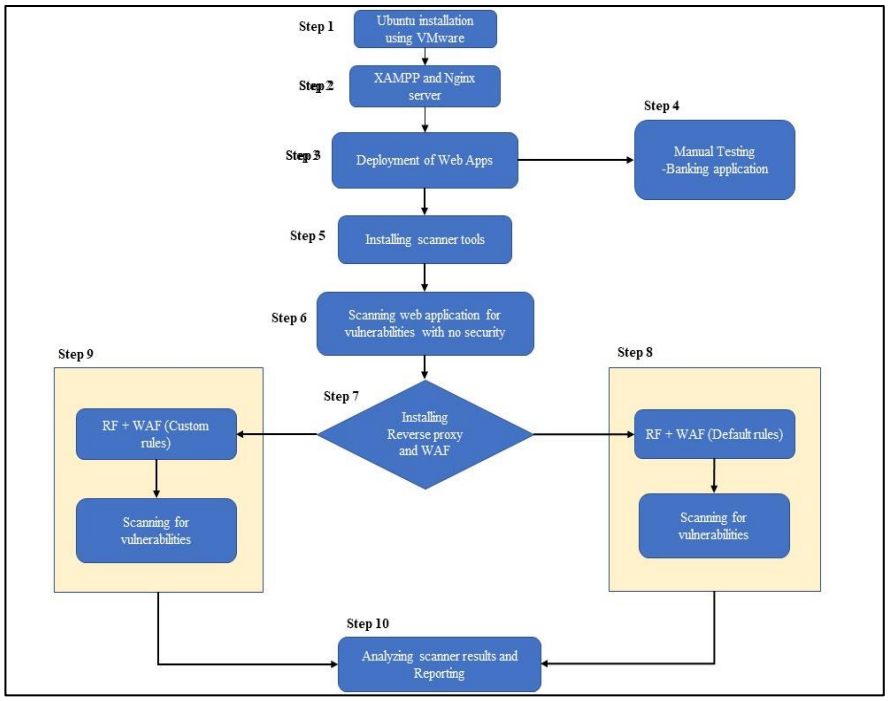

Figure 4: Implementation Flow Diagram

5.1 Environment Setup:

Step 1: Ubuntu installation using VMware.
Starting with setup of my project my primary operating system is Windows OS which has a configuration about 8GB RAM and 1TB Hard Disk. But since there are multiple software's and configurations needed to proceed with my project Linux system would provide me flexibility and more functionality. In this way the implementation process starts with the installation and configuration of UBUNTU virtual machine through VMware Workstation Pro. For this setup **ubuntu-22.04.1-desktop-amd64 .iso** image file has been downloaded and setup the parameters for the UBUNTU machine and get it configured and installed. Now my virtual machine is ready to run my project with basic configuration of 4GB RAM ,40GB Hard Disk and 4 Processors.
Step 2: XAMPP and Nginx installation
Second step in my setup installation is to install XAMPP to run my Apache Server and MYSQL server to add database of my Web Applications and Installation of Nginx server. For this purpose, **XAMPP-linux-installer-8.1.12.0** file has been downloaded and installed in the system though Terminal by using commands: First giving permission to installation file by using **sudo chmod 777 XAMPP-linux-installer-8.1.12.0** and then running the installation file to install the XAMPP setup. This will default run the Apache server on **localhost:80/dashboard** and MYSQL server to add web application database on **localhost/phpMyAdmin**.
After installation of XAMPP next is to install Nginx server on ubuntu machine. For this installation following command **sudo apt install nginx** has been used, which will install nginx sever on port 80 by default.
Step 3: Deployment of Web Apps
Third step is to host my Web applications on Apache server. For this Banking application code file has been downloaded from [15] and had moved the banking application folder on

[15] https://www.sourcecodester.com/php/14868/banking-system-using-php-free-source-code.html

/opt/lamp/htdocs folder and run my application on **http://localhost/banking** and then redirect to **http://localhost/phpmyadmin/** and then added banking_db.sql file to run the application. After this implementation I got my web application running on **http://localhost/banking/**. Same procedure has been followed to implement DVWA and Mutillidae II Web Applications. DVWA[16] and Mutillidae [17]has been cloned using GitHub Repository and installation has been done successfully.

Step 4: Manual Testing

Next step in my installation setup is to manual test the application to find out which are the Vulnerabilities present in the Web Applications. Due to the less time span I have done Manual testing on Banking Application by passing the following parameters and intersecting it with proxy tool called as Burp suit. Table 5 defines all the parameters that I have passed to detect the vulnerability in application:

Table 5:Parameter Passed to detect known Vulnerabilities

Vulnerability	Parameters passed/Description	Location
SQL Injection	admin' or 1=1#	http://localhost/banking/admin/login.php
XSS attack	Payload- <script>alert(document.cookie)</script>	https://localhost/banking/admin/?page=accounts/manage_account&id=7
Lack of password Brute Force	Unknow credentials and intercept with burp suit to get parameters	http://localhost/banking
HTML injection attack	Payload- < a href=https://www.ncirl.ie/>Click Here	http://localhost/banking/client/?page=user
Session fixation attack	http://localhost/banking/admin/login.php and intercept with burpsuit which generates session cookie PHPSESSID	http://localhost/banking/admin/login.php
Missing Rate Limiters	http://localhost/banking/admin/?page=accounts/manage_account go to this location and intercept with burpsuit and add payload position	http://localhost/banking/admin/?page=accounts/manage_account
Insecure session Termination	Visit to this mentioned location and update the admin account and intercept with burp suit	http://localhost/banking/admin/login.php
Account Tampering	Go to the mentioned location and intercept request with burp suit and the tamper the values of user account.	http://localhost/banking/client/?page=user
Weak password policy	Go to the mentioned location and intercept with burp suit and then change the password for admin user and then again intercept it will be successful and then try to login u will get incorrect credentials error	http://localhost/banking/admin/?page=user
Directory Listing	This location mentioned are unblocked need to be blocked on application level by adding parameters in .htaccess files for Apache and Nginx server	http://localhost/banking/classes http://localhost/banking/uploads
Clickjacking	Security Headers in .htaccess file for Apache server.	http://localhost/banking/admin

[16] https://github.com/digininja/DVWA

[17] https://github.com/webpwnized/mutillidae

Vulnerability	Parameters passed/Description	Location
Password transmitted in plain text	Login to account at given location and intercept with burp suit u can see password in clear text	http://localhost/banking/admin/login.php
Missing HTTP response header	Secure HTTP response headers are not added in .htaccess files of Apache server	http://localhost/banking/admin/login.php http://localhost/banking

Step 5: Installing Scanner Tools

Fifth step in setup installation is to download Three selected scanners such as ZAP[18], Nessus Essentials[19] and Arachni[20]. For this purpose, installation files have been downloaded for all the scanner tools for Linux system from its official websites. After that I have configured them to setup the proxy for Firefox where my web application is running and with my scanner. After that I have scanned each scanner with Web application and generated the results for the same.

Step 6: Scanning Web Apps with No security

In this step of implementation, selected Web Apps have been already deployed on Apache without implementing any security features. In this step each scanner tool were used for scanning under the different experimental setup of web application with No security and generated the scanning results of each scanner tool.

Step 7: Installing Reverse Proxy and WAF

Sixth step in my setup installation is to setup reverse proxy between Apache and Nginx server running on different ports. As default both Apache and Nginx server were running on same port i.e port 80. So, to setup the reverse proxy I have done changes in the configuration files for both servers which are as follows:

Apache server: I have made changes in Apache configuration file called **httpd.conf** and set the listening port of Apache on port 5000.

Nginx server: I have made changes in Nginx configuration file called **sites-availabe >default** and set the listening port to 3000 and restart the XAMPP server and got my both servers running on different ports. So, to setup Nginx as proxy server for Apache I have made changes again in sites-available >default file of Nginx and passed the proxy parameters as

```
location /banking {
  Proxy_pass http://localhost:5000/banking.
}
location /DVWA {
  Proxy_pass http://127.0.0.1:5000/DVWA.
}
location /mutillidae {
  Proxy_pass https://127.0.0.1:5000/mutillidae.
}
```

And then restarting the XAMPP server and restarting the nginx services my reverse proxy was successfully setup and in running condition and getting my banking application running on both ports.

After successfully installing Reverse Proxy next shadow daemon WAF need to be deployed to detect all the request coming in and out through web application. For this purpose, I have installed shadow daemon user interface by cloning the shadowd repository from git clone [21]and

[18] https://www.zaproxy.org/download/
[19] https://www.tenable.com/downloads/nessus?loginAttempted=true
[20] https://www.arachni-scanner.com/download/
[21] https://github.com/zecure/shadowdctl.git

then creating up the user and hosting up the shadow_ui interface on http://127.0.0.1:8080 and creating profile for Web Applications.

Step 8: Installing Reverse Proxy +WAF (default rules)

After successfully deploying web applications, setting up reverse proxy and shadow daemon WAF next step is to import the default rules for shadow daemon. The default rules for shadow Daemon WAF are being imported from GitHub repository [22]and then scanned each scanner with web applications to generate the report for the same setup.

Step 9: Installing Reverse Proxy +WAF (Custom rules)

After successfully deploying web application, setting up reverse proxy and shadow daemon WAF next step is to import the custom rules for shadow daemon to prevent the Banking application from getting attacked. After checking the above results of scanner and manual testing I have got PATH and CALLER value to create rules for the Banking application. I have used various prebuilt shadow daemon regress expression to block the attacks. Table 6 illustrates Blacklist rules for SQL injection, Table 8 illustrates Blacklist rules for XSS attacks and Table 7 illustrates Blacklist rules for HTML injection.

Table 6: Blacklist Rules for SQL Injection

SQLI	PATH	CALLER	Regular Expression used to create WAF Rules
User Login Page	POST|username|include_parameters|*	/opt/lampp/htdocs/banking/index.php	admin'or 1=1# 'OR1-- - \bfind_in_set \b.*?\(.+?,.+?\) \bmysql.*?\..*?user\, \bunion\b.+?\bselect\b \bdelete\b.+?\bfrom\b, --.+? \[\$(ne|eq|lte?|gte?|n?in|mod|all|size|exists|type|slice|or)\] \/*.*?*\/ \bbenchmark\b.*?\(.+?,.+?\)
Admin Login Page	POST|username|include_parameters|*	/opt/lampp/htdocs/banking/admin/login.php	

Table 7: Blacklist Rules for HTML Injection

HTML INJECTION	PATH	CALLER	Regular Expression used to create Blacklist rules
User account Page	POST|account_number|include_parameters|*	opt/lampp/htdocs/banking/client/?page=user	Click Here [\n\r]\s*\b(?:to|b?cc)\b\s*:.*?\@ >.*?<\s*\/?[\w\s]+> ['"]\s*\+\s*['"] \bfirefoxurl\s*: &#?(\w+); <!-.+?--> , ['"].*?>

Table 8: Blacklist Rules for XSS

XSS Attack	PATH	CALLER	Regular Expression used to create Blacklist rules

[22] https://github.com/zecure/shadowd_rules

Admin Manage Account Page	POST|account_number|include_parameters|*	/opt/lampp/htdocs/banking/classes/Master.php	<script>alert(document.cookie)</script> >.*?<\s*\/?[\w\s]+> \bfunction\b[^(]*\([^)]*\) ["']\s*\+\s*["'] +=\s*\(\s*["'] =\s*\w+\s*\+\s*["'] \bdocument\b.*?\. \bfirefoxurl\s*: &#?(\w+); \bon\w+\s*= <(html|body|meta|link|i?frame|script|map) <(form|button|input|keygen|textarea|select|option) <a\b.+?\bhref\b <base\b.+?\bhref\b.+?> <!-.+?--> \$\(.+?\),\)\s*\[,["'].*?>

Directory Listing Attack:

In this attack we directly need to blacklist the entire path from getting access,

http://localhost/banking/classes

PATH= GET|classes

CALLER= /opt/lamp/htdocs/banking/classes

http://localhost/banking/upload

PATH= GET|upload

CALLER= /opt/lamp/htdocs/banking/uploads

Clickjacking Attack:

Need to add following Headers to .htaccess file for Apache server in Banking application

Header set X-Frame-Options “DENY”

Header set X-Frame-Options “SAMEORIGIN”

Header set Content-Security-Policy “frame-ancestors ‘none’;”

Missing Security HTTP Response Header:

Need to add following Headers to .htaccess file for Apache server in Banking application

Header set X-Frame-Options “DENY”

Header set X-Frame-Options “SAMEORIGIN”

Header set Content-Security-Policy “frame-ancestors ‘none’; default-src ‘self’”

Header set X-Content-Type-Options “”

Header set X-XSS-Protection “0”

Header set Strict-Transport-Security “max-age=63072000; includeSubdomains;”

After adding the above rules and making changes in .htaccess file I have scanned my Banking application with each selected scanners and then have analyse the report generated from it.

Step 10: Analysing and Reporting

In this step I have Analysed the scanner reports generated at each experimental setup and reported the vulnerabilities detected and prevented as per my knowledge which is been illustrated in Evaluation Chapter.

6 Evaluation

In this chapter I have presented all findings on the above experimental setup that have performed throughout research work which was been proposed. I have scanned my web application at different stages to acquire the results of scanners which has been illustrated below. This scanner results are going to show the No of vulnerabilities found in the web application at different stages with count of CRITICAL, HIGH, MEDIUM, LOW, INFORMATIONAL issues found at each stage. I have also used MS EXCEL to visualize the obtained results from the scanners and to represent them graphically. At final stage of experiment, I will be comparing the results of how many attacks are been detected at initial stage and how many are prevented at final stage of experiment.

6.1 Banking Application

6.1.1 Manual Testing Results

As it has been discussed in the previous section of Methodology, manual testing was done on Banking application to check which are the vulnerability present after checking out the results of the scanners and following are the list of 12 vulnerability illustrated in Table 7 that I have detected manual in Banking web application. CVSS score and Severity has been assigned to vulnerability by using common vulnerability scoring system calculator V3.1[23] by passing the parameters in Base Score Metrics.

Table 7:Manual Testing Results

No.	Vulnerability Names	CVSS	Severity	OWASP 2021
1	SQL Injection Attack	9.3	CRITICAL	Injection
2	Cross-site scripting XSS Attack	8.4	HIGH	Injection
3	Brute Force (Password) Attack	8.1	HIGH	Identification &Authentication Failure
4	HTML injection Attack	7.9	HIGH	Injection
5	Session Fixation Attack	7.7	HIGH	Identification &Authentication Failure
6	Missing Rate Limiters	6.5	MEDIUM	Security Misconfiguration
7	Insecure Session Termination	6.2	MEDIUM	Identification &Authentication Failure
8	Weak Password Policy	4.4	MEDIUM	Broken Access Control
9	Directory Listing	4	MEDIUM	Security Misconfiguration
10	Clickjacking Attack	3.6	LOW	Security Misconfiguration
11	Password Transmitted in Plain Text	2.9	LOW	Cryptographic Failures
12	Missing Secure HTTP Header	2.9	LOW	Security Misconfiguration

6.1.2 Experiment 2: Analyzing Scanner Results

In this step of analysis, the scanner results of selected three tools have been analysed and compared the results obtained at different stages for Banking Application. Following are the count illustrated in Table 8 which shows that ZAP count for No security is 30, RP+WAF (default rules) is 28 and RP+WAF (custom rules) is 12 which is in decreasing order which states that the vulnerabilities are prevented after applying custom rules for shadow Daemon WAF against various Web Applications. Same count of calculations goes for Nessus and Arachni Vulnerability Scanners as illustrated in Table 8.

[23] https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator

Table 8:Scanner Results for Banking Application

Scanner	Stages	Critical	High	Medium	Low	Info	Total
ZAP	No Security	0	0	10	10	10	30
	RP+WAF (default rules)	0	0	9	11	8	28
	RP+WAF (custom rules)	0	0	4	5	3	12
Nessus	No Security	1	0	7	0	18	26
	RP+WAF (default rules)	1	0	3	6	30	40
	RP+WAF (custom rules)	0	0	1	2	10	13
Arachni	No Security	0	0	6	8	6	20
	RP+WAF (default rules)	0	0	6	9	4	19
	RP+WAF (custom rules)	0	0	2	2	3	7

The Following Table 9 illustrates the Attacks detected for Web Application with No security Vs attacks detected for Web Application with Reverse Proxy and WAF (Default rules) Vs attacks detected for Web Application with Reverse Proxy and WAF (Custom rules) for Each Scanner.

Table 9:Vulnerabilty Assessment on Banking Application

Vulnerabilities	Banking Application Conditions								
	OWSAP ZAP			NESSUS ESSENTIAL			ARACHNI		
	No Security	RP+ WAF (default rules)	RP+ WAF (custom rules)	No Security	RP+ WAF (default rules)	RP+ WAF (custom rules)	No Security	RP+WAF (default rules)	RP+ WAF (custom rules)
SQL Injection Attack	✓	✓	×	✓	✓	×	✓	✓	×
Cross-site scripting XSS Attack	✓	✓	×	✓	✓	✓	✓	✓	×
Brute Force (Password) Attack	✓	✓	✓	✓	✓	✓	✓	✓	×
HTML injection Attack	✓	✓	×	✓	✓	×	✓	✓	×
Session Fixation Attack	✓	✓	✓	✓	✓	✓	✓	✓	✓
Missing Rate Limiters	✓	✓	✓	✓	✓	✓	✓	✓	✓
Insecure Session Termination	✓	✓	✓	✓	✓	✓	✓	✓	✓
Weak Password Policy	✓	✓	✓	✓	✓	✓	✓	✓	✓
Directory Listing	✓	×	×	✓	×	×	✓	×	×
Clickjacking Attack	✓	✓	×	✓	✓	✓	✓	✓	✓
Password Transmitted in Plain Text	✓	✓	✓	✓	✓	✓	✓	✓	✓
Missing Secure HTTP Header	✓	×	×	✓	×	×	✓	×	×

As per the count of the above scanner results it is seen that when we have scanned the scanner tools to the experimental setup for Banking application with No security contains approximately 30-40 numbers of Vulnerabilities in the Banking application as mentioned above. The same when I have scanned the scanner tool to the experimental setup for web application with reverse proxy and shadow Daemon with default rules for Banking application it was seen that there is slight change in vulnerability count as it protects the threats related to Headers and Directories, no other major attack has been prevented. The same when I have scanned the scanner tool to the experimental setup for web application with reverse proxy and shadow Daemon custom rules for Banking application attacks such as SQL injection, cross-site scripting, HTML injection attack, Directory Listing, clickjacking, and various HTTP

headers are being prevented using different rulesets and header files implication on WAF.This states that custom rules for Web applications plays an important role for preventing attacks through WAF.

6.2 DVWA Web Application

6.2.1 Experiment 1: Analyzing Scanner Results

In this step of analysis, the scanner results of selected Three tools have been analysed and compared the results obtained at different stages for DVWA Application. Following are the count illustrated in Table 10 which shows that ZAP count for No security is 30, RP+WAF (default rules) is 26 which is in decreasing order which states that the vulnerabilities are prevented after applying 1st level of security after implementing Reverse proxy which goes same for Nessus and Arachni Vulnerability scanners.

Table 10:Scanner Results for DVWA Application

Scanner	Stages	Critical	High	Medium	Low	Informational	Total
ZAP	No Security	0	1	8	11	10	30
	RP+WAF (default rules)	0	0	8	10	8	26
Nessus	No Security	1	1	9	1	20	32
	RP+WAF (default rules)	0	0	3	3	24	30
Arachni	No Security	0	0	10	5	2	17
	RP+WAF (default rules)	0	0	10	5	3	18

The Following Table 11 illustrates the Attacks detected for DVWA Application with No security Vs attacks detected for DVWA Application with Reverse Proxy and WAF (Default rules) for each scanner.

Table 11:Vulnerabilty Assessment on DVWA Application

Vulnerabilities	DVWA Application Conditions					
	OWSAP ZAP		NESSUS ESSENTIAL		ARACHNI	
	No Security	RP+WAF (default rules)	No Security	RP+WAF (default rules)	No Security	RP+WAF (default rules)
SQL Injection	✓	✓	✓	✓	✓	✓
Persistent cross site scripting	✓	✓	✓	✓	✓	✓
Command execution	×	✓	✓	✓	✓	✓
Local file injection	✓	✓	✓	✓	✓	✓
Session Fixation	✓	✓	✓	✓	✓	✓
Password transmitted in plain text	✓	✓	✓	✓	✓	✓
clickjacking	✓	✓	✓	✓	✓	✓
Missing secure HTTP response header	✓	×	✓	×	✓	×
Cross-site request Forgery	✓	✓	✓	✓	✓	✓
Directory listing	✓	×	×	×	✓	×

Since Manual testing was not possible to conduct on DVWA application, creating custom rules for these applications was not possible which give PATH and CALLER value for the same. So, when I scanned the scanner tools to the experimental setup for DVWA application with No security detects almost 30 numbers of Vulnerabilities are present in the web application. Comparing with the results of deploying DVWA Application under reverse proxy and default rule set of Shadow Daemon WAF it has been seen that major attacks on DVWA Application is not prevented. Default rules of WAF only prevent its user interface and some HTTP header

issues has been resolved. So, it is important to add custom rules to WAF to prevent DVWA Application from getting attacked. It is also seen that from Arachni count that instead of decreasing the vulnerability count after implementation of security it has added informational threats which don't have major impact on web application.

6.3 Mutillidae II Web Application

6.3.1 Experiment 1: Analyzing Scanner results

In this step of analysis, the scanner results of selected tools have been analysed and compared the results obtained at different stages for Mutillidae II Application. Following are the count illustrated in Table 12 which shows that ZAP count for No security is 32, RP+WAF (default rules) is 28 which is in decreasing order which states that the vulnerabilities are prevented after applying 1st level of security after implementing Reverse proxy which goes same for Nessus and Arachni Vulnerability scanners.

Table 12:Scanner Results for Mutillidae II Application

Scanner	Stages	Critical	High	Medium	Low	Informational	Total
ZAP	No Security	0	1	9	11	11	32
	RP+WAF (default rules)	0	1	10	10	7	28
Nessus	No Security	0	0	4	0	27	31
	RP+WAF (default rules)	0	0	4	3	22	29
Arachni	No Security	0	17	11	9	3	40
	RP+WAF (default rules)	0	0	7	21	6	34

The Following Table 13 illustrates the Attacks detected for Mutillidae Application with No security Vs attacks detected for Mutillidae Application with Reverse Proxy and WAF (Default rules) for each scanner.

Table 13:Vulnerabilty Assessment on Mutillidae Application

Vulnerabilities	Mutillidae Application Conditions					
	OWSAP ZAP		NESSUS ESSENTIAL		ARACHNI	
	No Security	RP+WAF (default rules)	No Security	RP+WAF (default rules)	No Security	RP+WAF (default rules)
Persistent(stored)cross-site scripting	✓	✓	✓	✓	✓	✓
cross-site scripting (XSS)in script context	✓	✓	×	✓	✓	✓
Local file Injection	✓	✓	✓	✓	✓	✓
Directory listing	✓	✓	✓	✓	✓	✓
SQL injection	✓	✓	✓	✓	✓	✓
HTML injection	✓	✓	✓	✓	✓	✓
Path Transversal	✓	×	×	✓	✓	✓
Missing Security Headers	✓	×	✓	×	✓	×
Missing HTTP response headers	✓	×	✓	×	✓	×
Private IP Transversal	×	×	×	✓	✓	✓
Remote File Injection	✓	✓	✓	✓	✓	✓
HTTP cookies leading to XSS attack	✓	✓	✓	✓	✓	✓

Since Manual testing was not possible to conduct on Mutillidae application, creating custom rules for these applications was not possible. So, when I scanned the scanner tools to the experimental setup for mutillidae application with No security detects almost 40 numbers of Vulnerabilities are present in the web application. Comparing with the results of deploying Mutillidae Application under reverse proxy and default rule set of Shadow Daemon WAF it has been seen that major attacks on Application is not prevented. Default rules of WAF only prevent its user interface and some HTTP header issues has been resolved. So, it is important to add custom rules to WAF to prevent DVWA Application from getting attacked.

6.4 Discussion

In this chapter we will discuss the overall results of the N experiments performed above to clearly define the objective of our research. As for the research project three Web Applications such as Banking, DVWA and Mutillidae II as demonstrated above for studies were considered and analysed. When I deployed the First Web Application (Banking) and tested them manually and by using Automatic Scanners without adding any security features it was seen that there are n-Number of Vulnerabilities present in web application. When same application was tested by adding security features to them by implementing them under Reverse proxy and Shadow Daemon WAF with default and custom rules for the web application it was seen that, critical threats marked as per the OWAS top 10 were protected when custom rules were implemented for Banking web application. In second setup when DVWA and Mutillidae II Web Application were deployed without deploying it under security features and after implementing it under reverse proxy and WAF default rules it was seen that no major attack was prevented as per the scanner results. This proves our First objective of research question that security features such as reverse proxy and custom rules for WAF plays an important role in preventing Web application from getting attacked.

When we scanned each Web Applications under different experimental setup such as without Reverse proxy and WAF, with reverse proxy and WAF default rules and with reverse proxy and WAF custom rules and analyse the scanner results it was seen that it crawls deep into application while time of scanning the web application to find vulnerabilities and gives all the detailed information about the vulnerabilities which is not been detected by manual penetration tester. Which proves second objective of research question that Automatic scanner tools gives more accurate results and overcomes the manual testing and perhaps make it work more faster than manual testing. But it is also observed that OWAS ZAP and Arachni gives detailed deep crawled results of Web Application in comparison to Nessus Essential Web Vulnerability Scanner. All these factors need to be taken into consideration and further work need to be done on the above project setup to get an enhanced and clear results for the same.

7 Conclusion and Future Work

The main objective of my research was to implement Reverse proxy and Shadow Daemon WAF to prevent the Web application from getting attacked. After performing various testing and analysing the results of different scanners it is been seen that implementation Reverse Proxy and Shadow Daemon WAF with custom rules for web-based application can enhance the security features of web application. The attacks such as SQL injection, Cross -site scripting, clickjacking, HTTP headers, Directory listing, all this threats to the web application are being prevented by reverse proxy and Shadow Daemon Web application firewall (WAF). This in turns also states the importance to deploying WAF custom rulesets for any web application is more important to protect the web-based application over default WAF rules used to prevent shadow user interface. It also states the importance of open-source scanners for vulnerabilities detection over manual testing.

After analysing the output obtained from the research, it can be stated that there is an opportunity to improve the research work done. This can add Future Work as, to detect the web attacks efficiently development can be done to generate attack logs as well as to set an alert alarm system which will prove beneficial for the program administrator to know if any attempt is made to attack the web application. Improvement can be done on scanner tools and can also implement new open-source scanners to compare it with commercial tools for its efficient working and attack detection functionalities. Ruleset for Shadow Daemon WAF can be simplified for smooth user experience. Implication of new custom rulesets to prevent DVWA and Mutillidae II Web Application form getting attacked from different threats.

8 Acknowledgment

I would Specially thank my supervisor Dr. Arghir Nicolae Moldovan who have guided me thought my research project and pushed me ahead to gain tremendous knowledge in field of cybersecurity and helped me out to evaluate my research project successfully. Lastly, I express my gratitude towards my whole family and friends for their great support throughout my whole process.

References

A survey on web application vulnerabilities and countermeasures | IEEE Conference Publication | IEEE Xplore. (n.d.). Retrieved 19 December 2022, from https://ieeexplore.ieee.org/document/6316697

Amankwah, R., Chen, J., Kudjo, P. K., & Towey, D. (2020). An empirical comparison of commercial and open-source web vulnerability scanners. *Software - Practice and Experience*, *50*(9), 1842–1857. https://doi.org/10.1002/SPE.2870

An Analysis and Classification of Vulnerabilities in Web-Based Application Development | IEEE Conference Publication | IEEE Xplore. (n.d.). Retrieved 19 December 2022, from https://ieeexplore.ieee.org/document/9441467

Arnaldy, D., & Hati, T. S. (2020). Performance Analysis of Reverse Proxy and Web Application Firewall with Telegram Bot as Attack Notification on Web Server. *2020 3rd International Conference on Computer and Informatics Engineering, IC2IE 2020*, 455–459. https://doi.org/10.1109/IC2IE50715.2020.9274592

Clincy, V., & Shahriar, H. (2018). Web Application Firewall: Network Security Models and Configuration. *Proceedings - International Computer Software and Applications Conference*, *1*, 835–836. https://doi.org/10.1109/COMPSAC.2018.00144

Fong, E., & Okun, V. (2007). Web application scanners: Definitions and functions. *Proceedings of the Annual Hawaii International Conference on System Sciences*. https://doi.org/10.1109/HICSS.2007.611

Ghanbari, Z., Rahmani, Y., Ghaffarian, H., & Ahmadzadegan, M. H. (2016). Comparative approach to web application firewalls. *Conference Proceedings of 2015 2nd International Conference on Knowledge-Based Engineering and Innovation, KBEI 2015*, 808–812. https://doi.org/10.1109/KBEI.2015.7436148

Kiruba, B., Saravanan, V., Vasanth, T., & Yogeshwar, B. K. (2022). OWASP Attack Prevention. *3rd International Conference on Electronics and Sustainable Communication Systems, ICESC 2022 - Proceedings*, 1671–1675. https://doi.org/10.1109/ICESC54411.2022.9885691

Lewandowski, P., Janiszewski, M., & Felkner, A. (2020). SpiderTrap - An Innovative Approach to Analyze Activity of Internet Bots on a Website. *IEEE Access*, *8*, 141292–141309. https://doi.org/10.1109/ACCESS.2020.3012969

Makino, Y., & Klyuev, V. (2015). Evaluation of web vulnerability scanners. *Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2015*, *1*, 399–402. https://doi.org/10.1109/IDAACS.2015.7340766

Mburano, B., & Si, W. (2019). Evaluation of web vulnerability scanners based on OWASP benchmark. *26th International Conference on Systems Engineering, ICSEng 2018 - Proceedings*. https://doi.org/10.1109/ICSENG.2018.8638176

Muzaki, R. A., Briliyant, O. C., Hasditama, M. A., & Ritchi, H. (2020). Improving Security of Web-Based Application Using ModSecurity and Reverse Proxy in Web Application Firewall. *2020 International Workshop on Big Data and Information Security, IWBIS 2020*, 85–90. https://doi.org/10.1109/IWBIS50925.2020.9255601

Pałka, D., & Zachara, M. (2011). Learning web application firewall - Benefits and caveats. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6908 LNCS*, 295–308. https://doi.org/10.1007/978-3-642-23300-5_23

Razzaq, A., Hur, A., Shahbaz, S., Masood, M., & Ahmad, H. F. (2013). *Critical analysis on web application firewall solutions*. 1–6. https://doi.org/10.1109/ISADS.2013.6513431

Valeur, F., Vigna, G., Kruegel, C., & Kirda, E. (2006). An anomaly-driven reverse proxy for web applications. *Proceedings of the ACM Symposium on Applied Computing*, *1*, 361–368. https://doi.org/10.1145/1141277.1141361