WEB APPLICATION FIREWALL USING SAFELINE WAF

Setup, Configuration, and Attack Mitigation

(A Practical Study on Implementing SafeLine Web Application Firewall for Web Attack Protection)

Project Report

N I V E D H C

ncodes404@gmail.com

ABSTRACT

With the increasing use of web applications for handling sensitive data and critical business operations, web-based attacks such as SQL Injection, Cross-Site Scripting (XSS), and Denial-of-Service (DoS) attacks have become a major security concern. Traditional network security mechanisms like firewalls are not sufficient to protect web applications from application-layer attacks. To address this challenge, Web Application Firewalls (WAF) play a crucial role in detecting and blocking malicious HTTP traffic before it reaches the application.

This project focuses on the practical implementation of SafeLine Web Application Firewall (WAF) in a controlled laboratory environment. A deliberately vulnerable web application was deployed using Damn Vulnerable Web Application (DVWA) to simulate real-world web vulnerabilities. Common attacks such as SQL Injection were launched from an attacker machine to test the effectiveness of the WAF.

SafeLine WAF was installed and configured as a reverse proxy in front of the vulnerable web application. The project demonstrates how SafeLine inspects incoming web traffic, detects malicious payloads, and actively blocks attack attempts. Additional security features such as HTTP flood protection and custom deny rules were also explored to enhance application security.

The results show that SafeLine WAF is effective in identifying and mitigating web-based attacks in real time. Through this project, practical knowledge was gained on web application security, attack techniques, and defensive mechanisms. This study highlights the importance of Web Application Firewalls in modern cybersecurity environments and their role in protecting web applications from common and advanced threats

INTRODUCTION

The rapid growth of web applications and online services has significantly increased the exposure of systems to cyber threats. Modern organizations rely heavily on web-based platforms for authentication, data storage, transactions, and communication. As a result, web applications have become a primary target for attackers seeking to exploit vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), and denial-of-service attacks. If left unprotected, these vulnerabilities can lead to data breaches, service disruption, and loss of user trust.

Traditional network security solutions like firewalls and intrusion detection systems mainly focus on filtering traffic based on IP addresses, ports, and protocols. While these mechanisms are essential, they are not sufficient to detect and prevent attacks that operate at the application layer. Many web-based attacks are embedded within legitimate HTTP requests, making them difficult to identify using conventional security tools alone

To overcome these limitations, Web Application Firewalls (WAFs) are deployed to monitor, analyze, and filter HTTP and HTTPS traffic between users and web applications. A WAF inspects incoming requests in real time, detects malicious payloads, and blocks attacks before they reach the application server. This makes WAFs a critical component of modern web security architectures

SafeLine WAF is an advanced web application firewall designed to protect web applications from common and advanced attacks such as SQL Injection, XSS, brute-force attempts, and HTTP flood attacks. It can be deployed as a reverse proxy and provides real-time visibility into web traffic along with detailed security logs.

This project focuses on the practical implementation of SafeLine WAF in a controlled laboratory environment. A vulnerable web application was intentionally deployed to simulate real-world security flaws. Various web attacks were performed to evaluate how effectively SafeLine WAF detects, logs, and blocks malicious activity. The project provides hands-on experience in web application security, attack analysis, and defensive techniques used in real-world cybersecurity environments.

OBJECTIVES

The primary objective of this project is to study, implement, and analyze a Web Application Firewall (WAF) using SafeLine WAF in a controlled laboratory environment. The project is designed to provide practical exposure to web application security by demonstrating how web-based attacks are detected and prevented in real time.

The specific objectives of the project are as follows:

- To understand the fundamentals of web application security and the role of Web Application Firewalls in protecting web applications from application-layer attacks.
- To deploy a vulnerable web application in order to simulate real-world security flaws such as SQL Injection and Cross-Site Scripting (XSS).
- To install and configure SafeLine WAF as a reverse proxy for protecting the web application.
- To analyze how SafeLine WAF inspects HTTP/HTTPS traffic and identifies malicious requests.
- To perform common web-based attacks such as SQL Injection and observe how they are detected and blocked by the WAF.
- To study SafeLine WAF logs and alerts for better understanding of attack patterns and security events.
- To explore additional SafeLine WAF security features such as HTTP flood protection and custom deny rules.
- To gain hands-on experience relevant to real-world web security monitoring and defensive security operations.

SCOPE

The scope of this project is limited to the implementation and analysis of SafeLine Web Application Firewall (WAF) in a controlled laboratory environment. The project focuses on protecting a vulnerable web application from common application-layer attacks by inspecting and filtering HTTP and HTTPS traffic.

This project includes the deployment of a deliberately vulnerable web application using Damn Vulnerable Web Application (DVWA) to simulate real-world web security flaws. The application was protected using SafeLine WAF configured as a reverse proxy. Common web attacks such as SQL Injection were performed to evaluate the effectiveness of the WAF in detecting and blocking malicious requests.

The scope of the project also includes analyzing SafeLine WAF security logs, observing alert generation, and understanding how the WAF enforces security policies. Additional features such as HTTP flood protection and custom IP-based deny rules were explored to enhance the security posture of the application.

The project is restricted to educational and ethical testing in an isolated environment. It does not involve testing on live or production web applications. Advanced enterprise-level integrations such as SIEM integration, large-scale performance benchmarking, or machine learning-based detection mechanisms are outside the scope of this study. However, these aspects are briefly discussed as possible future enhancements.

Overall, the scope of this project is to provide a practical understanding of Web Application Firewall deployment and web attack mitigation using SafeLine WAF without extending into complex enterprise-level implementations.

ARCHITECTURE AND WORKING OF SAFELINE WAF

SafeLine Web Application Firewall follows a layered architecture designed to inspect, analyze, and control web traffic in real time. It operates primarily as a reverse proxy, sitting between the client and the backend web application. This architecture allows SafeLine to intercept all incoming HTTP and HTTPS requests before they reach the application server.

The main components of SafeLine WAF architecture are described below:

1. Traffic Interception Layer

The traffic interception layer is the entry point of SafeLine WAF. All client requests directed to the web application are first received by the WAF. SafeLine listens on standard web ports such as HTTP (80) and HTTPS (443) and acts as an intermediary between users and the backend server.

By operating as a reverse proxy, SafeLine ensures that ensures that the backend web application is not directly exposed to the internet. This layer enables centralized inspection and control of incoming traffic.

2. Protocol Analysis and Request Parsing

Once traffic is intercepted, SafeLine analyzes the structure of HTTP and HTTPS requests. This includes parsing request headers, URLs, parameters, cookies, and request bodies. SafeLine normalizes incoming requests to ensure that obfuscated or encoded payloads are properly decoded before inspection.

This step is essential for identifying malicious patterns that may be hidden within seemingly legitimate requests.

3. Detection and Security Engine

The detection engine is the core component of SafeLine WAF. It applies predefined security rules and detection logic to identify common web-based attacks such as SQL Injection, Cross-Site Scripting (XSS), brute-force attempts, and abnormal request behavior.

SafeLine uses multiple detection techniques, including signature-based rules and behavior-based analysis, to accurately detect malicious traffic. When a request matches a security rule, SafeLine classifies it as malicious and takes the appropriate action

4. Response and Enforcement Module

After detecting a malicious request, SafeLine enforces security policies by blocking the request, returning an error page, or temporarily banning the source IP address. This prevents the attack means from reaching the backend web application.

The enforcement module ensures real-time protection by stopping attacks at the application layer without affecting legitimate users.

5. Logging and Monitoring Module

SafeLine records detailed logs for all detected security events. These logs include information such as source IP address, attack type, timestamp, and action taken. The logging module provides visibility into web attacks and helps administrators analyze attack patterns and security incidents.

These logs are useful for security monitoring, forensic analysis, and improving security rules.

LAB ENVIRONMENT SETUP

The SafeLine WAF project was implemented in a controlled virtual laboratory environment to ensure safe and ethical testing. Virtualization technology was used to simulate real-world attacker and server scenarios without affecting any production systems. The lab setup consists of multiple virtual machines, each assigned a specific role in the experiment.

1. Virtualization Platform

The laboratory environment was created using Oracle VirtualBox. VirtualBox was chosen due to its ease of use, stability, and support for running multiple operating systems simultaneously. All virtual machines were configured using bridged networking to allow proper communication between the attacker system, WAF, and web server.

2. Attacker Machine – Kali Linux

A Kali Linux virtual machine was used as the attacker system. Kali Linux is a penetration testing distribution that includes a wide range of tools for web application testing and attack simulation.This machine was used to perform web-based attacks such as SQL Injection against the vulnerable web application protected by SafeLine WAF.

3. Web Server – Ubuntu Server

An Ubuntu Server virtual machine was used to host the vulnerable web application and SafeLine WAF. Ubuntu Server was selected due to its reliability, security, and strong support for open-source tools.

The following components were installed on the Ubuntu server:

- Apache Web Server
- PHP
- MySQL Database
- Damn Vulnerable Web Application (DVWA)

DVWA was used to intentionally introduce vulnerabilities required for testing the effectiveness of the WAF.

4. Web Application Firewall – SafeLine WAF

SafeLine WAF was installed on the Ubuntu server and configured as a reverse proxy in front of the vulnerable web application. All incoming HTTP and HTTPS traffic was routed through SafeLine before reaching the backend server.

SafeLine was configured to inspect web requests, detect malicious payloads, and block attacks such as SQL Injection in real time.

5. Network Configuration

All virtual machines were connected to the same network to allow seamless communication. IP addresses were identified using network tools, and local DNS resolution was configured to simulate a realistic domain-based web application environment.

This lab setup provided a realistic and secure environment to test web application attacks and evaluate the effectiveness of SafeLine WAF.

VULNERABLE WEB APPLICATION SETUP

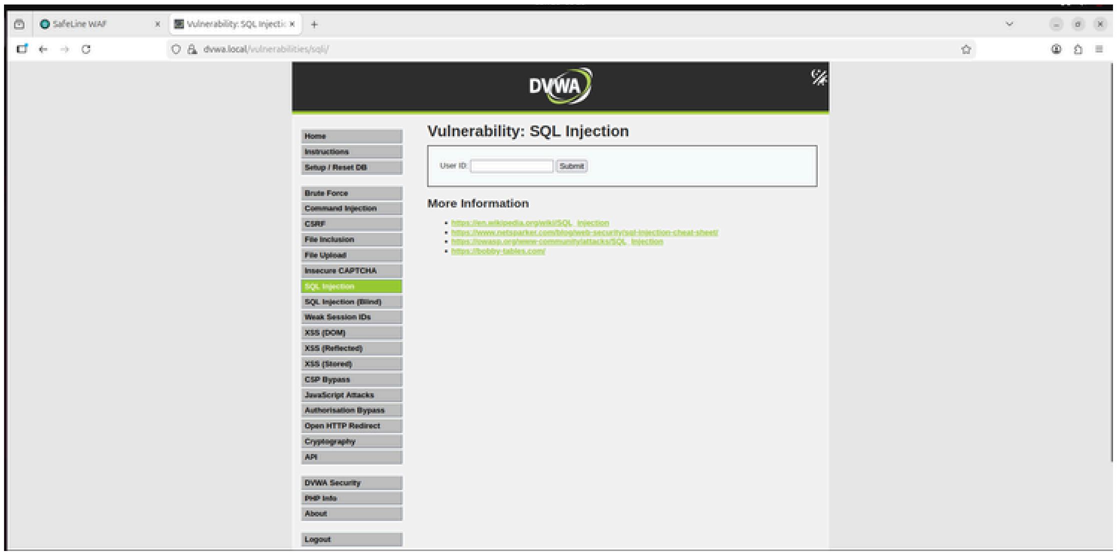
To evaluate the effectiveness of SafeLine Web Application Firewall, a deliberately vulnerable web application was deployed in the laboratory environment. The Damn Vulnerable Web Application (DVWA) was used for this purpose. DVWA is an intentionally insecure web application designed for security testing and learning purposes.

DVWA was installed on the Ubuntu server as part of the LAMP stack, which includes Apache, PHP, and MySQL. The application provides multiple security vulnerability modules that simulate real-world web application flaws in a safe and controlled manner.

1. SQL Injection Vulnerability

The SQL Injection module in DVWA was used to demonstrate how attackers can manipulate backend database queries through malicious input. The login and input fields in this module are intentionally vulnerable to unsanitized user input.

By entering crafted SQL payloads, an attacker can bypass authentication or extract sensitive data from the database. This vulnerability was used to test how SafeLine WAF detects and blocks SQL Injection attempt



- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
- https://owasp.org/www-community/attacks/SQL_Injection
- https://bobby-tables.com/

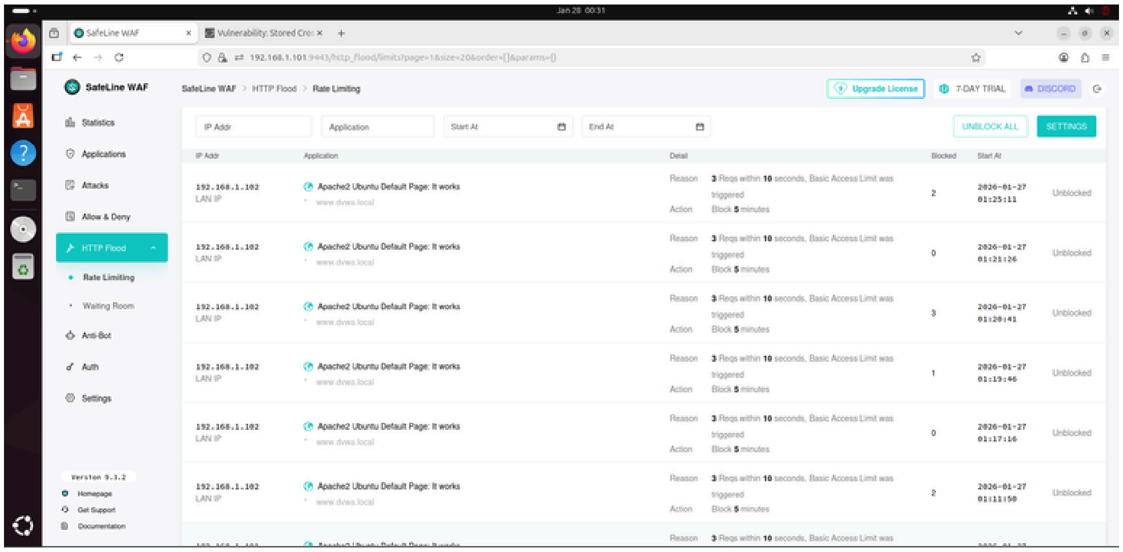
Figure 1: DVWA SQL Injection Page

2. HTTP Flood Defense Configuration (Additional Security Feature)

In addition to protecting against SQL Injection attacks, SafeLine WAF provides protection against HTTP flood and denial-of-service style attacks. HTTP flood attacks involve sending a large number of HTTP requests to overwhelm a web server and degrade its performance or availability.

SafeLine WAF includes built-in HTTP flood defense mechanisms that monitor request rates and detect abnormal traffic behavior. In this project, HTTP flood protection was enabled and configured to limit excessive requests from a single source.

Once the threshold was exceeded, SafeLine automatically blocked or restricted the attacker's IP address, preventing further requests from reaching the backend application. This demonstrated SafeLine's capability to protect web applications from both application-layer attacks and volumetric abuse.



IP Addr	Application	Detail	Blocked	Start At	
192.168.1.102 LAN IP	Apache2 Ubuntu Default Page: It works · www.dvwa.local	Reason 3 Reqs within 10 seconds, Basic Access Limit was triggered Action Block 5 minutes	2	2026-01-27 01:25:11	Unblocked
192.168.1.102 LAN IP	Apache2 Ubuntu Default Page: It works · www.dvwa.local	Reason 3 Reqs within 10 seconds, Basic Access Limit was triggered Action Block 5 minutes	0	2026-01-27 01:21:26	Unblocked
192.168.1.102 LAN IP	Apache2 Ubuntu Default Page: It works · www.dvwa.local	Reason 3 Reqs within 10 seconds, Basic Access Limit was triggered Action Block 5 minutes	3	2026-01-27 01:20:41	Unblocked
192.168.1.102 LAN IP	Apache2 Ubuntu Default Page: It works · www.dvwa.local	Reason 3 Reqs within 10 seconds, Basic Access Limit was triggered Action Block 5 minutes	1	2026-01-27 01:19:46	Unblocked
192.168.1.102 LAN IP	Apache2 Ubuntu Default Page: It works · www.dvwa.local	Reason 3 Reqs within 10 seconds, Basic Access Limit was triggered Action Block 5 minutes	0	2026-01-27 01:17:16	Unblocked
192.168.1.102 LAN IP	Apache2 Ubuntu Default Page: It works · www.dvwa.local	Reason 3 Reqs within 10 seconds, Basic Access Limit was triggered Action Block 5 minutes	2	2026-01-27 01:11:50	Unblocked
		Reason 3 Reqs within 10 seconds, Basic Access Limit was			

Figure 2 : SafeLine WAF HTTP Flood Rate Limiting and Blocking Events

The above figure shows the HTTP flood rate limiting events detected by SafeLine WAF. The attacker IP address 192.168.1.102 exceeded the configured request limit of 3 requests within 10 seconds, which resulted in SafeLine enforcing a temporary block of 5 minutes. These logs confirm that SafeLine WAF successfully detected abnormal traffic behavior and mitigated a potential HTTP flood attack in real time. This demonstrates SafeLine's ability to protect web applications from denial-of-service style attacks in addition to application-layer exploits.

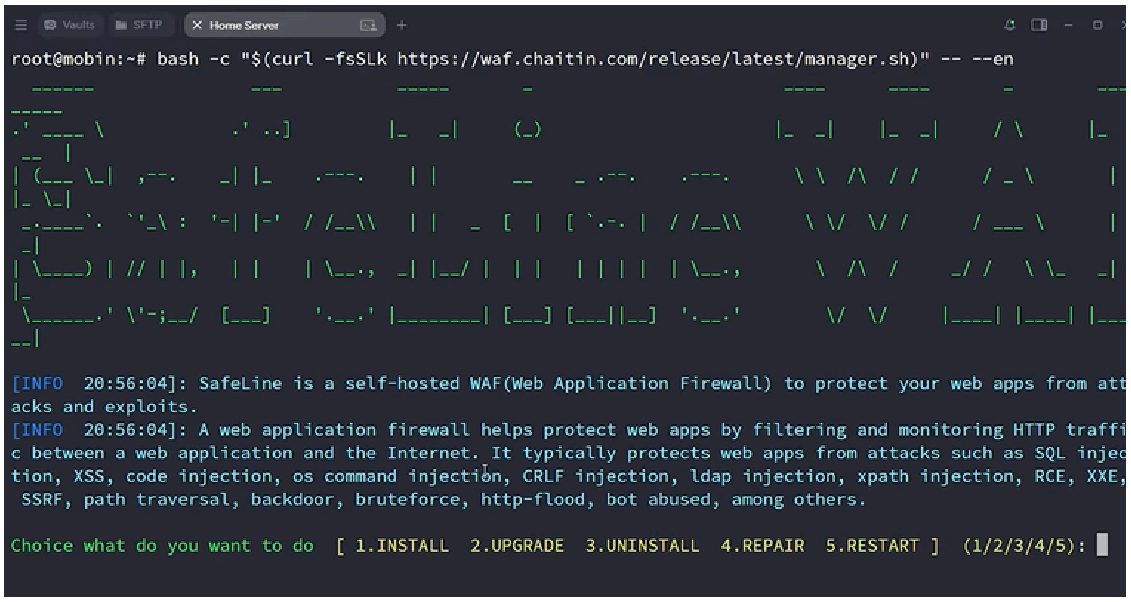
INSTALLATION AND CONFIGURATION OF SAFELINE WAF

SafeLine Web Application Firewall was installed and configured on the Ubuntu Server to protect the vulnerable web application. Prior to installation, the system was updated to ensure all required dependencies and packages were up to date. SafeLine WAF was deployed using its official installation script, which simplifies the setup process and ensures proper configuration.

1. SafeLine WAF Installation

SafeLine WAF provides an automated installation method using a shell script. The installation was performed by executing the official SafeLine installation command on the Ubuntu server terminal. This process installed all required components and services for running the WAF.

After successful installation, SafeLine provided a web-based management interface accessible through a secure HTTPS port. The administrator credentials were generated during installation and used to log in to the SafeLine dashboard.



```
root@mobin:~# bash -c "$(curl -fsSLk https://waf.chaitin.com/release/latest/manager.sh)" -- --en
  ______               ___           _____       _                     ____      ____       _        ___
_____
.' ____ \            .' ..]        |_   _|     (_)                   |_  _|    |_  _|     / \      |_
 __  |
| (___ \_|  ,--.    _| |_    .---.    | |       __    _ .--.    .---.      \ \  /\  / /     / _ \      |
|_ \_|
 _.____`.  `'_\ :  '-| |-'  / /__\\   | |   _  [  |  [ `.-. |  / /__\\      \ \/  \/ /     / ___ \     |
 _|
| \____) | // | |,   | |    | \__.,  _| |__/ |  | |   | | | |  | \__.,       \  /\  /     _/ /   \ \_   _|
|_
 \______.' \'-;__/  [___]    '.__.' |________| [___] [___||__]  '.__.'        \/  \/     |____| |____| |___
__|

[INFO  20:56:04]: SafeLine is a self-hosted WAF(Web Application Firewall) to protect your web apps from att
acks and exploits.
[INFO  20:56:04]: A web application firewall helps protect web apps by filtering and monitoring HTTP traffi
c between a web application and the Internet. It typically protects web apps from attacks such as SQL injec
tion, XSS, code injection, os command injection, CRLF injection, ldap injection, xpath injection, RCE, XXE,
 SSRF, path traversal, backdoor, bruteforce, http-flood, bot abused, among others.

Choice what do you want to do  [ 1.INSTALL  2.UPGRADE  3.UNINSTALL  4.REPAIR  5.RESTART ]  (1/2/3/4/5):
```

Figure 3: SafeLine WAF installation command executed in terminal

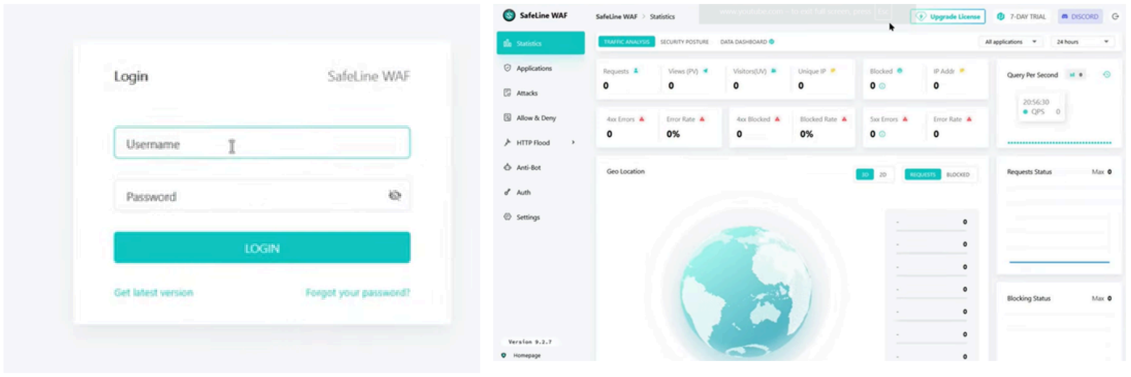


Figure 4: SafeLine WAF login page/dashboard

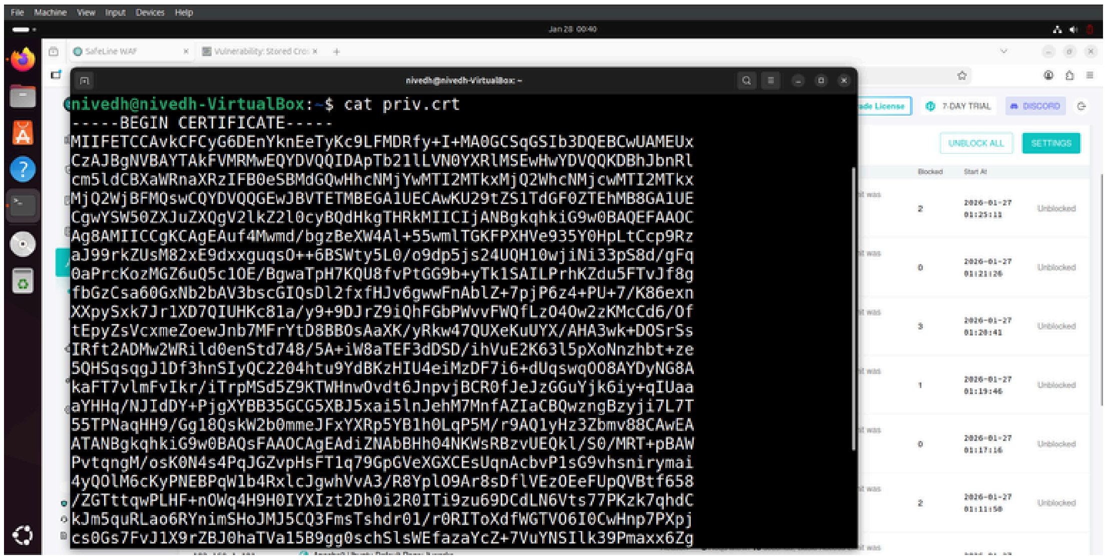
2. Certificate Configuration

To enable secure HTTPS communication, a self-signed SSL certificate was created and imported into SafeLine WAF. This allowed SafeLine to terminate HTTPS connections and inspect encrypted traffic before forwarding it to the backend web application.

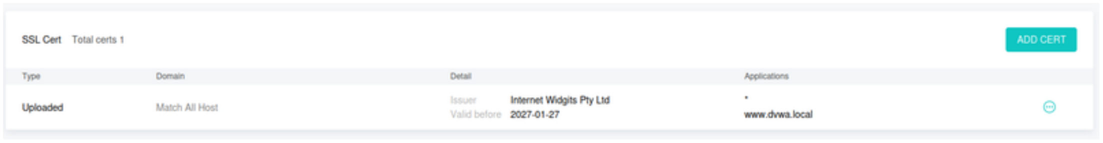
The certificate and private key were configured through the SafeLine management interface and assigned to the protected application.

On Ubuntu, create a self-signed certificate to use for HTTPS:

```
sudo mkdir /etc/ssl/dvwa

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \

 -keyout /etc/ssl/dvwa/dvwa.key \

 -out /etc/ssl/dvwa/dvwa.crt \
```



```
nivedh@nivedh-VirtualBox:~$ cat priv.crt
-----BEGIN CERTIFICATE-----
MIIFETCCAvkCFCyG6DEnYknEeTyKc9LFMDRfy+I+MA0GCSqGSIb3DQEBCwUAMEUx
CzAJBgNVBAYTAkFVMRMwEQYDVQQIDApTb21lLVN0YXRlMSEwHwYDVQQKDBhJbnRl
cm5ldCBXaWRnaXRzIFB0eSBMdGQwHhcNMjYwMTI2MTkxMjQ2WhcNMjcwMTI2MTkx
MjQ2WjBFMQswCQYDVQQGEwJBVTETMBEGA1UECAwKU29tZS1TdGF0ZTEhMB8GA1UE
CgwYSW50ZXJuZXQgV2lkZ2l0cyBQdHkgTHRkMIICIjANBgkqhkiG9w0BAQEFAAOC
Ag8AMIICCgKCAgEAuf4Mwmd/bgzBeXW4Al+55wmlTGKFPXHVe935Y0HpLtCcp9Rz
aJ99rkZUsM82xE9dxxguqsO++6BSWty5L0/o9dp5js24UQH10wjiNi33pS8d/gFq
0aPrcKozMGZ6uQ5c1OE/BgwaTpH7KQU8fvPtGG9b+yTk1SAILPrhKZdu5FTvJf8g
fbGzCsa60GxNb2bAV3bscGIQsDl2fxfHJv6gwwFnAblZ+7pjP6z4+PU+7/K86exn
XXpySxk7Jr1XD7QIUHKc81a/y9+9DJrZ9iQhFGbPWvvFWQfLzO40w2zKMcCd6/Of
tEpyZsVcxmeZoewJnb7MFrYtD8BBOsAaXK/yRkw47QUXeKuUYX/AHA3wk+DOSrSs
IRft2ADMw2WRild0enStd748/5A+iW8aTEF3dDSD/ihVuE2K63l5pXoNnzhbt+ze
5QHSqsqgJ1Df3hnSIyQC2204htu9YdBKzHIU4eiMzDF7i6+dUqswqOO8AYDyNG8A
kaFT7vlmFvIkr/iTrpMSd5Z9KTWHnwOvdt6JnpvjBCR0fJeJzGGuYjk6iy+qIUaa
aYHHq/NJIdDY+PjgXYBB35GCG5XBJ5xai5lnJehM7MnfAZIaCBQwzngBzyji7L7T
55TPNaqHH9/Gg18QskW2b0mmeJFxYXRp5YB1h0LqP5M/r9AQ1yHz3Zbmv88CAwEA
ATANBgkqhkiG9w0BAQsFAAOCAgEAdiZNAbBHh04NKWsRBzvUEQkl/S0/MRT+pBAW
PvtqngM/osK0N4s4PqJGZvpHsFT1q79GpGVeXGXCEsUqnAcbvP1sG9vhsnirymai
4yQOlM6cKyPNEBPqW1b4RxlcJgwhVvA3/R8YplO9Ar8sDflVEzOEeFUpQVBtf658
/ZGTttqwPLHF+nOWq4H9H0IYXIzt2Dh0i2R0ITi9zu69DCdLN6Vts77PKzk7qhdC
kJm5quRLao6RYnimSHoJMJ5CQ3FmsTshdr01/r0RIToXdfWGTVO6I0CwHnp7PXpj
cs0Gs7FvJ1X9rZBJ0haTVa15B9gg0schSlsWEfazaYcZ+7VuYNSIlk39Pmaxx6Zg
```



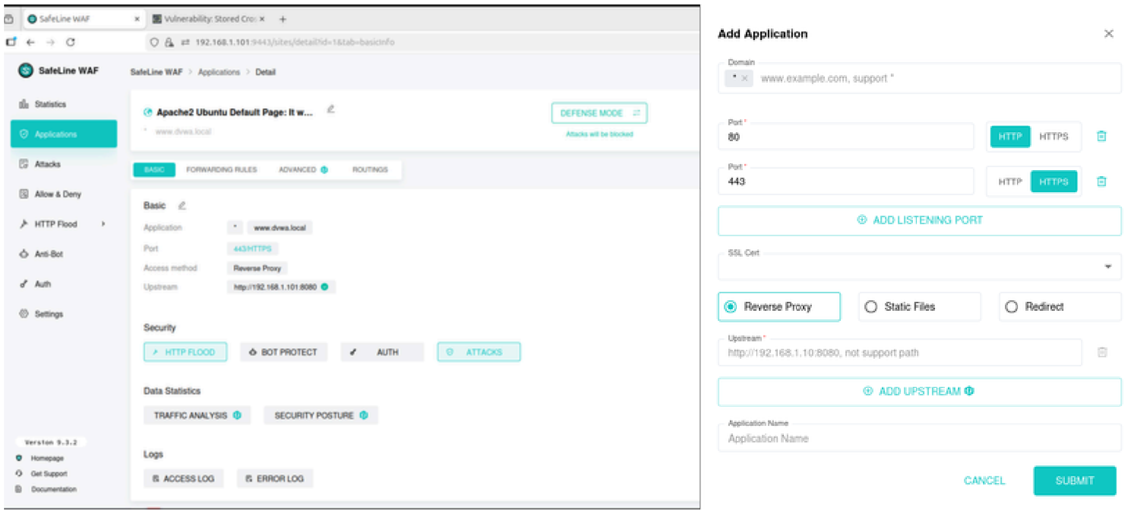
SSL Cert Total certs 1

Type	Domain	Detail	Applications
Uploaded	Match All Host	Issuer Internet Widgits Pty Ltd Valid before 2027-01-27	* www.dvwa.local

Figure 5: SSL certificate configuration in SafeLine

3. Application Onboarding

The vulnerable web application was onboarded into SafeLine WAF by defining the application domain and backend server details. SafeLine was configured to forward traffic to the backend web server running on a custom port while listening on standard HTTPS port 443.
This ensured that all incoming requests passed through SafeLine WAF before reaching the web application.



192.168.1.101:9443/sites/detail?id=1&tab=basicinfo

SafeLine WAF > Applications > Detail

Apache2 Ubuntu Default Page: It w...
www.dvwa.local

DEFENSE MODE — Attacks will be blocked

Basic

Application	* www.dvwa.local
Port	443/HTTPS
Access method	Reverse Proxy
Upstream	http://192.168.1.101:8080

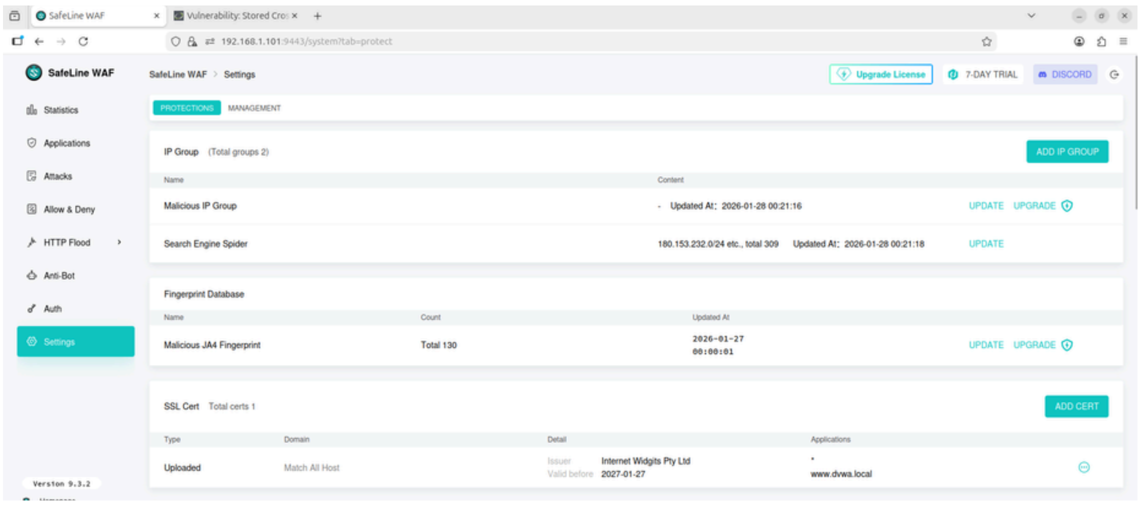
Add Application

Domain	* www.example.com, support *	
Port *	80	[✓ HTTP] HTTPS
Port *	443	HTTP [✓ HTTPS]
SSL Cert		
	[✓ Reverse Proxy] Static Files Redirect	
Upstream *	http://192.168.1.10:8080, not support path	
Application Name	Application Name	

Figure 6:Application onboarding / reverse proxy configuration in SafeLine

4. Security Policy Configuration

SafeLine's default security policies were enabled to protect against common web attacks. These policies included SQL Injection detection, request filtering, and rate limiting mechanisms. Blocking mode was enabled to ensure malicious requests were actively blocked rather than only logged
This configuration allowed SafeLine to detect and mitigate attacks in real time.



192.168.1.101:9443/system?tab=protect

SafeLine WAF > Settings

IP Group (Total groups 2)

Name	Content	
Malicious IP Group	- Updated At: 2026-01-28 00:21:16	UPDATE UPGRADE
Search Engine Spider	180.153.232.0/24 etc., total 309 Updated At: 2026-01-28 00:21:18	UPDATE

Fingerprint Database

Name	Count	Updated At	
Malicious JA4 Fingerprint	Total 130	2026-01-27 00:00:01	UPDATE UPGRADE

SSL Cert Total certs 1

Type	Domain	Detail	Applications
Uploaded	Match All Host	Issuer Internet Widgits Pty Ltd Valid before 2027-01-27	* www.dvwa.local

Figure 7:SafeLine Security Policy Settings

ATTACK DEMONSTRATION – SQL INJECTION

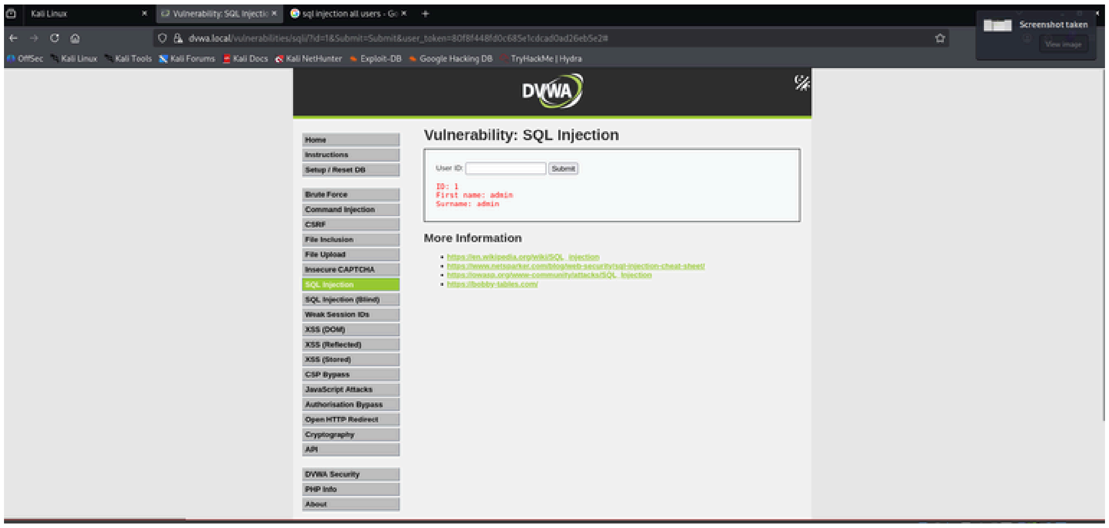
To evaluate the effectiveness of SafeLine Web Application Firewall, a SQL Injection attack was performed against the vulnerable web application. SQL Injection is a common web application attack technique that allows an attacker to manipulate backend database queries through malicious input.

In this project, the SQL Injection vulnerability provided by DVWA was used to simulate a real-world attack scenario.

1. SQL Injection Without WAF Protection

Initially, the vulnerable web application was accessed without enforcing SafeLine WAF blocking rules. Malicious SQL payloads were entered into the input field to manipulate the backend database query.

By using a crafted SQL injection string, the attacker was able to bypass normal application logic and retrieve unintended results. This confirmed that the application was vulnerable to SQL Injection attacks.



dvwa.local/vulnerabilities/sqli/?id=1&Submit=Submit&user_token=80f8f448fd0c685e1cdcad0ad26eb5e2#

Vulnerability: SQL Injection

```
ID: 1
First name: admin
Surname: admin
```

More Information

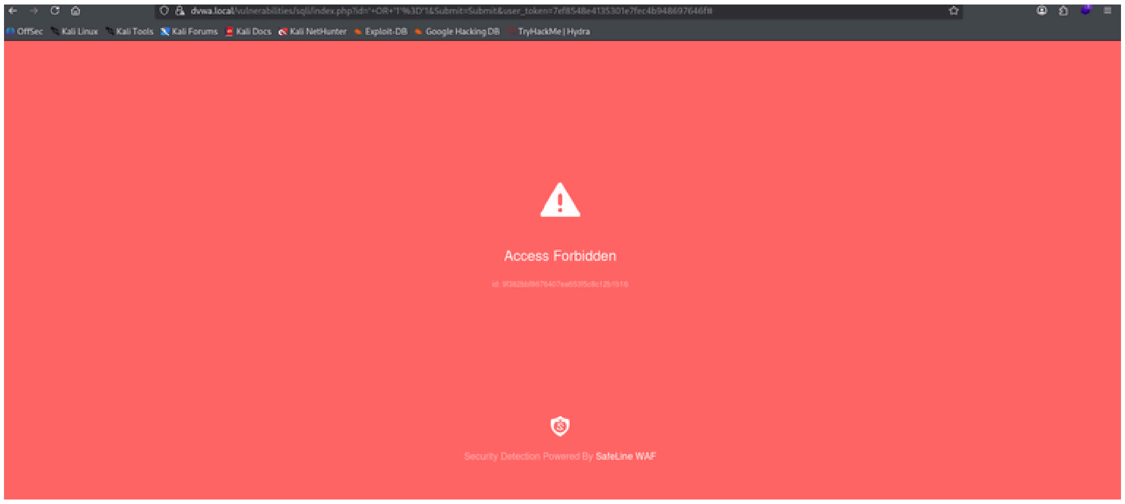
- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
- https://owasp.org/www-community/attacks/SQL_Injection
- https://bobby-tables.com/

Figure 8: Successful SQL Injection Attempt Without WAF

2. SQL Injection With SafeLine WAF Enabled

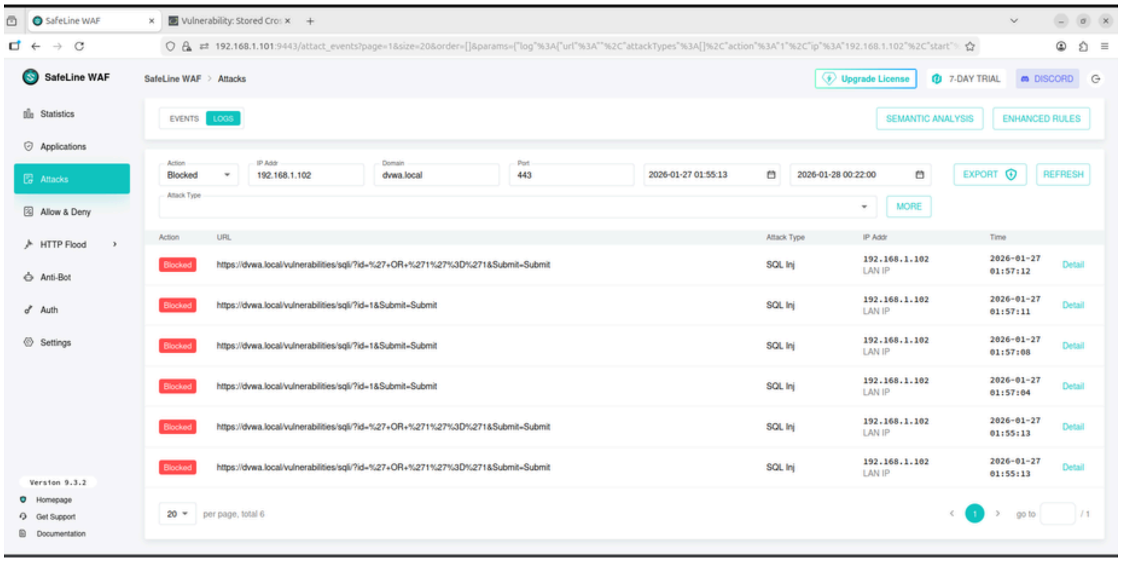
After confirming the vulnerability, SafeLine WAF protection was enabled in blocking mode. The same SQL Injection payload was attempted again from the attacker system.

SafeLine WAF detected the malicious SQL injection pattern in the HTTP request and immediately blocked the request. The attack was prevented from reaching the backend application, and a block response was returned to the attacker



```
dvwa.local/vulnerabilities/sqli/index.php?id='+OR+'1'%3D'1&Submit=Submit&user_token=7ef8548e4135301e7fec4b948697646f#
```

Figure 9: SQL Injection Blocked by SafeLine WAF



```
192.168.1.101:9443/attact_events?page=1&size=20&order=[]&params={"log"%3A{"url"%3A""%2C"attackTypes"%3A[]%2C"action"%3A"1"%2C"ip"%3A"192.168.1.102"%2C"start"
```

Action: Blocked | IP Addr: 192.168.1.102 | Domain: dvwa.local | Port: 443 | 2026-01-27 01:55:13 | 2026-01-28 00:22:00 | Attack Type:

Action	URL	Attack Type	IP Addr	Time	
Blocked	https://dvwa.local/vulnerabilities/sqli/?id=%27+OR+%271%27%3D%271&Submit=Submit	SQL Inj	192.168.1.102 LAN IP	2026-01-27 01:57:12	Detail
Blocked	https://dvwa.local/vulnerabilities/sqli/?id=1&Submit=Submit	SQL Inj	192.168.1.102 LAN IP	2026-01-27 01:57:11	Detail
Blocked	https://dvwa.local/vulnerabilities/sqli/?id=1&Submit=Submit	SQL Inj	192.168.1.102 LAN IP	2026-01-27 01:57:08	Detail
Blocked	https://dvwa.local/vulnerabilities/sqli/?id=1&Submit=Submit	SQL Inj	192.168.1.102 LAN IP	2026-01-27 01:57:04	Detail
Blocked	https://dvwa.local/vulnerabilities/sqli/?id=%27+OR+%271%27%3D%271&Submit=Submit	SQL Inj	192.168.1.102 LAN IP	2026-01-27 01:55:13	Detail
Blocked	https://dvwa.local/vulnerabilities/sqli/?id=%27+OR+%271%27%3D%271&Submit=Submit	SQL Inj	192.168.1.102 LAN IP	2026-01-27 01:55:13	Detail

20 per page, total 6 · 1 · go to / 1

Figure 10: SafeLine WAF Log Showing SQL Injection Detection

TESTING AND RESULTS

After completing the installation and configuration of SafeLine Web Application Firewall, multiple tests were conducted to evaluate its effectiveness in protecting the vulnerable web application. The testing phase focused on verifying whether SafeLine could accurately detect and block malicious web traffic while allowing legitimate requests to pass through.

1. SQL Injection Testing Results

During SQL Injection testing, malicious input payloads were submitted through the vulnerable DVWA input fields. When SafeLine WAF protection was disabled, the SQL Injection attack executed successfully, confirming the presence of a vulnerability in the application.

Once SafeLine WAF was enabled in blocking mode, the same SQL Injection payloads were detected and blocked immediately. SafeLine generated security alerts and prevented the malicious requests from reaching the backend database. This confirmed that SafeLine WAF effectively mitigates SQL Injection attacks in real time.

2. HTTP Flood Defense Testing Results

To test HTTP flood protection, repeated HTTP requests were generated from a single source within a short time interval. SafeLine WAF monitored the request rate and detected abnormal traffic behavior once the configured threshold was exceeded.

SafeLine automatically enforced rate limiting and temporarily blocked the source IP address. The block events were logged in the SafeLine dashboard, confirming that the HTTP flood defense mechanism was functioning correctly.

3. Observations

- Legitimate web traffic was allowed without interruption.
- Malicious SQL Injection attempts were blocked instantly.
- Excessive request rates triggered automatic IP blocking.
- Detailed logs and alerts were generated for all detected attacks.

ADVANTAGES AND LIMITATIONS OF SAFELINE WAF

SafeLine Web Application Firewall offers effective protection against a wide range of web-based attacks. Like any security solution, it has both strengths and limitations. Understanding these aspects is important for effective deployment and management.

Advantages of SafeLine WAF

One of the major advantages of SafeLine WAF is its ability to protect web applications from application-layer attacks such as SQL Injection and HTTP flood attacks in real time. By operating as a reverse proxy, SafeLine ensures that malicious traffic is filtered before reaching the backend server.

SafeLine WAF provides an easy-to-use web-based management interface that simplifies configuration, monitoring, and analysis. Security policies such as rate limiting and request filtering can be enabled without complex manual rule creation

Another advantage is its detailed logging and alerting capability. SafeLine generates comprehensive logs that help administrators understand attack patterns and take appropriate action. The built-in HTTP flood defense adds an extra layer of protection against denial-of-service style attacks.

SafeLine WAF is also flexible and suitable for both learning environments and real-world deployments, making it a valuable tool for web application security.

Limitations of SafeLine WAF

One limitation of SafeLine WAF is that advanced enterprise features may require a licensed version, which can limit access to certain capabilities in free or trial editions. Proper tuning is required to avoid false positives that may block legitimate traffic.

SafeLine WAF focuses primarily on web application-layer protection and does not replace other security mechanisms such as network firewalls or intrusion detection systems. It must be used as part of a layered security approach.

Additionally, SafeLine WAF requires proper configuration and monitoring to remain effective. Misconfiguration may reduce its ability to detect attacks or impact application performance.

CONCLUSION

This project successfully demonstrated the implementation and practical evaluation of SafeLine Web Application Firewall (WAF) in a controlled laboratory environment. The primary objective was to understand how a Web Application Firewall protects vulnerable web applications from application-layer attacks, and this objective was achieved through hands-on experimentation.

A deliberately vulnerable web application was deployed and tested against common web attacks such as SQL Injection. SafeLine WAF was installed and configured as a reverse proxy to inspect incoming web traffic. The results showed that SafeLine effectively detected and blocked SQL Injection attempts in real time, preventing malicious requests from reaching the backend application.

In addition to SQL Injection protection, SafeLine's HTTP flood defense mechanism was tested and proven effective in identifying and mitigating excessive request-based attacks. The WAF successfully enforced rate limiting and temporarily blocked attacker IP addresses, demonstrating its capability to protect web applications from denial-of-service style threats.

Through this project, valuable practical knowledge was gained in web application security, attack analysis, and defensive security mechanisms. The implementation highlights the importance of Web Application Firewalls as a critical component of modern cybersecurity architectures. Overall, this project confirms that SafeLine WAF is an effective solution for protecting web applications and provides a strong foundation for further exploration in web security and defensive operations.