Report: Nginx Web Application Firewall (WAF) Installation on Kali Linux

Ziad Mahmoud Ahmed Abdelgwad

---

1. Introduction

This document details the successful installation and configuration of a robust Web Application Firewall (WAF) on a Kali Linux system. The WAF was implemented by integrating the ModSecurity 3.0 engine with the Nginx web server. The primary objective of this report is to demonstrate the WAF's functionality through a series of test cases, verifying its ability to detect and block common web-based attacks.

---

2. Prerequisites and Environment

Operating System: Kali Linux

Web Server: Nginx (version 1.26.3)

WAF Engine: ModSecurity 3.0 with the OWASP Core Rule Set (CRS)

The installation process involved compiling the ModSecurity library and its Nginx connector from source, ensuring compatibility and modularity with the Nginx server.

---

3. Installation and Configuration Overview

The installation was completed in several key phases:

Dependency Installation: All necessary build tools and libraries were installed to support the compilation of ModSecurity and its Nginx connector.

ModSecurity Core Compilation: The **libmodsecurity** library was compiled and installed from its official GitHub source.

Nginx Connector Module Compilation: A dynamic module (**ngx_http_modsecurity_module.so**) was compiled against the existing Nginx version, ensuring it could be loaded without reinstalling the entire web server.

Nginx Integration: The compiled module was loaded in the Nginx main configuration file using the **load_module** directive.

WAF Configuration: The ModSecurity engine was configured to operate in blocking mode (**SecRuleEngine On**), and the OWASP Core Rule Set (CRS) was installed and included in the ModSecurity configuration file (**modsecurity.conf**).

Server Block Activation: ModSecurity was activated for the default Nginx virtual host by adding the **modsecurity on;** and **modsecurity_rules_file** directives within the **server** block.

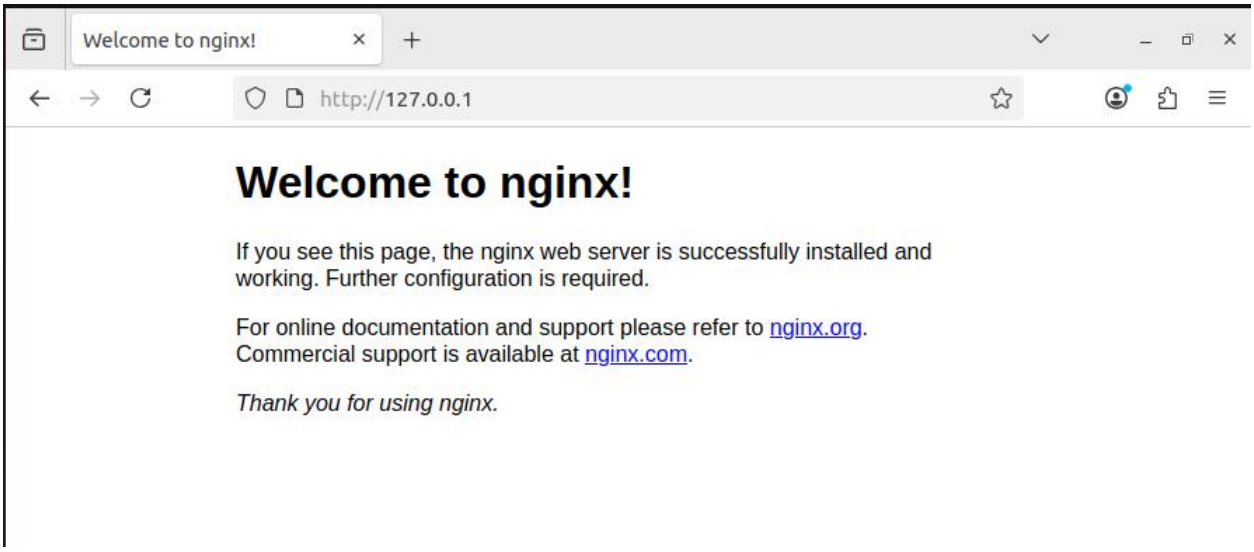
---

4. WAF Test Cases and Verification

To confirm the WAF is functioning correctly, a series of attack simulations were conducted. The verification for each test case consists of two parts: observing a **403 Forbidden** response in the web browser and verifying a corresponding "Access denied" alert in the Nginx error log.

4.1. Baseline Test: Normal Access

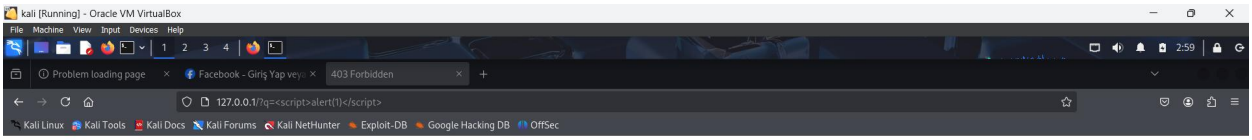
Before any attack simulations, a baseline test was performed to ensure the web server was operating normally.

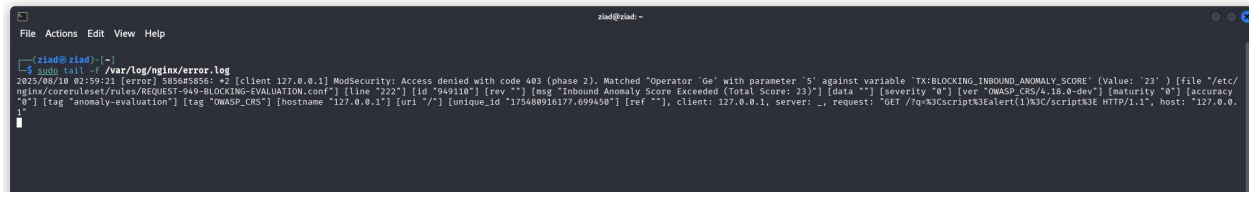


4.2. Test Case 1: Cross-Site Scripting (XSS) Attack

A request containing a simple XSS payload was sent to the server to test for client-side script injection.

Payload: **http://127.0.0.1/?q=<script>alert(1)</script>**



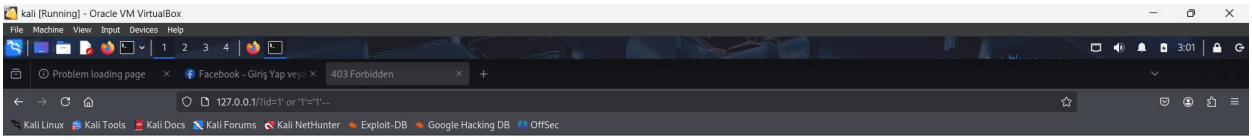


Verification: The **403 Forbidden** error in the browser and the log entry referencing **REQUEST-941-APPLICATION-ATTACK-XSS.conf** confirm that the WAF successfully identified and blocked the attack.

4.3. Test Case 2: SQL Injection (SQLi) Attack

A request containing a classic SQL injection payload was sent to the server to test for database manipulation attempts.

Payload: **http://127.0.0.1/?id=1' or '1'='1**

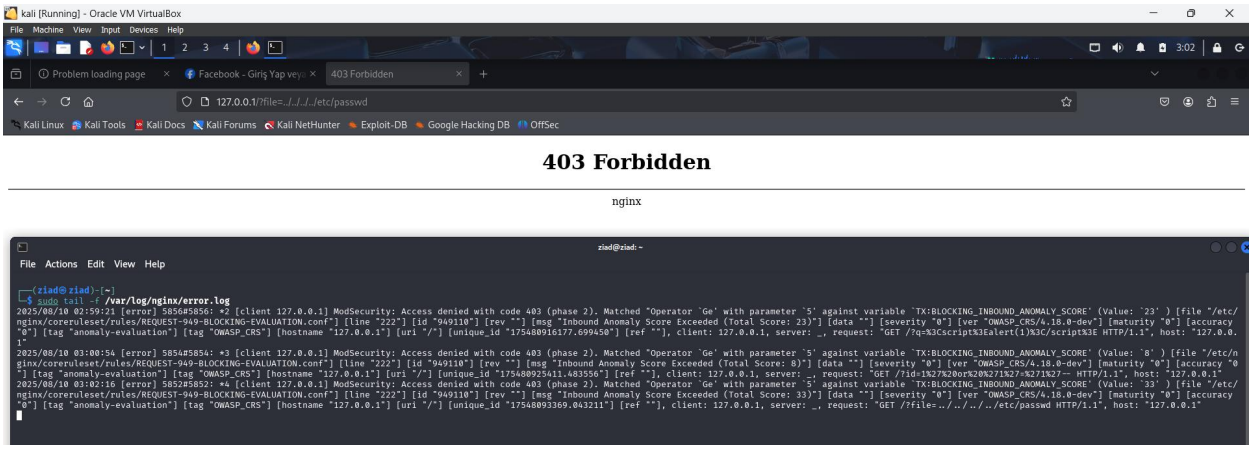


Verification: The **403 Forbidden** response and the log entry referencing **REQUEST-942-APPLICATION-ATTACK-SQLI.conf** verify that the WAF is effectively protecting against SQL injection attacks.

4.4. Test Case 3: Local File Inclusion (LFI) Attack

A request containing a directory traversal payload was sent to the server to test for attempts to read local system files.

Payload: **http://127.0.0.1/?file=../../../../etc/passwd**



Verification: The **403 Forbidden** response and the log entry referencing **REQUEST-930-APPLICATION-ATTACK-LFI.conf** confirm the WAF's capability to defend against local file inclusion and path traversal attacks.

---

5. Conclusion

The installation of ModSecurity with Nginx was successful. The comprehensive testing, documented with screenshots, provides conclusive evidence that the Web Application Firewall is correctly configured and actively protects the server from a variety of common and critical attack vectors, including Cross-Site Scripting, SQL Injection, and Local File Inclusion. This robust security layer significantly enhances the defense of the web server against malicious activity.