Executive Summary - Red Team WAF Detection & Bypass Lab

Objective:

This Red Team simulation aimed to assess the effectiveness of Web Application Firewall (WAF) protection on a vulnerable web application hosted at http://192.168.1.110. The objective was to identify, fingerprint, and bypass WAF rules using various techniques.

Scope:

- Perform reconnaissance and web fingerprinting using Nmap, WhatWeb, Nikto, and WAFW00F
- Detect WAF signatures and identify rule types (e.g., ModSecurity)
- Actively test WAF filtering mechanisms using obfuscation, fuzzing, header tampering, and encoding
- Document all payloads, detections, and evasion results in a structured format

Techniques Used:

Recon & Fingerprinting:	Nmap, WhatWeb, Nikto, WAFW00F
WAF Fingerprinting:	WAFW00F, Manual Detection
Bypass Techniques:	Burp Suite, FFUF, Payload Obfuscation
Fuzzing & Encoding:	Unicode, Base64, Path Traversal
Header Manipulation:	X-Forwarded-For, Referer, etc.

Summary of Findings:

WAF Detected: ModSecurity (via WAFW00F, response headers)

Filter Evasion Success:

- Base64 + Unicode payload chaining partially bypassed filters
- Header tampering allowed access to restricted areas

- Obfuscated payloads bypassed keyword-based regex detection

WAF Block Behavior:

- 403 Forbidden pages
- Rate limiting observed during fuzzing attempts

Key Takeaways:

- Many basic WAF rules can be bypassed with layered encoding and header manipulation
- Regex-based filters are predictable and can be fuzzed
- Combining multiple evasion techniques increases bypass success
- Manual validation (e.g., Burp Repeater) helps confirm impact

Author:

Name: Aditya Goswami

GitHub: Aditya-Sec

LinkedIn: aditya-kumar-goswami

WAF Bypass Techniques - Technical Findings Report

Target Web Application:

IP: http://192.168.1.110

Environment: DVWA / Custom Lab Setup

Security Layer: ModSecurity WAF detected (via WAFW00F)

Tools & Techniques Overview:

Nmap - Port scanning & service detection - Done

WhatWeb - Web fingerprinting - Done

Nikto - Basic web vuln scanner - Done

WAFW00F - WAF detection - Done

Burp Suite Repeater - Manual bypass validation - Done

Payload Obfuscation - Split payloads, reversed/encoded - Bypassed

Header Tampering - Faked origin/IP via headers - Bypassed

FFUF/WFuzz - Path + parameter fuzzing - Bypassed

Unicode/Base64 Encoding - WAF rule evasion via transformations - Partial

Regex Pattern Detection - Simulated custom rule testing - Done

Observations Per Technique:

1. Payload Chaining - Combined Base64 + Unicode + traversal - Partial bypass

2. Header Tampering - Used X-Forwarded-For: 127.0.0.1 - Bypassed IP filtering

3. Fuzzing - Discovered hidden paths, upload interface - Login bypass via param

4. Manual Payload Testing - Used Burp Suite for raw payload delivery

5. Base64 + Unicode Bypass - Signature filters failed on encoded input

6. Regex-Based Filtering - Filters caught raw, missed obfuscations

Summary of WAF Behavior:

Raw Payloads: Mostly Blocked (403)

Encoded Payloads: Partially Allowed

Header Tampering: Bypassed access controls

Obfuscation: Bypassed simple filters

Key Learning Outcomes:

- WAFs are often misconfigured and rely heavily on pattern-based detection
- Encoding + header manipulation = highly effective bypass strategy
- Tools like Burp Suite, FFUF, and WAFW00F are essential for testing defense depth
- Manual validation is a must after automated testing

Author:

Aditya Goswami - GitHub: Aditya-Sec - LinkedIn: aditya-kumar-goswami