

Attack and Defend Web Applications | Oluwaseun Quadri

Case Study: PrestaShop Web Application

1. Introduction

Web applications are common targets for cyberattacks due to vulnerabilities arising from insecure coding practices, misconfigurations, and lack of defensive controls. This assignment focuses on attacking and defending a real-world web application (PrestaShop) using industry-standard security tools and frameworks.

The objectives of this exercise are to:

- Identify web application vulnerabilities based on the OWASP Top 10
- Perform security testing using OWASP ZAP
- Implement a Web Application Firewall (WAF) using ModSecurity
- Validate the effectiveness of defensive controls

2. Environment Setup

2.1 Target Application

- **Application:** PrestaShop (E-commerce platform)
- **Server OS:** Ubuntu Linux
- **Web Server:** Apache 2.4
- **Target IP Address:** 192.168.56.102

2.2 Attacker Machine

- **Operating System:** Kali Linux
- **Security Tools Used:**
 - OWASP ZAP
 - Browser (Firefox with proxy enabled)

3. Attack Phase – Web Application Security Testing

3.1 Tool Used: OWASP ZAP

OWASP ZAP (Zed Attack Proxy) is an open-source web application security scanner used to identify vulnerabilities during development and testing.

3.2 Proxy Configuration

The Kali Linux browser was configured to route traffic through OWASP ZAP:

- HTTP Proxy: 127.0.0.1

- Port: 8080

This allowed OWASP ZAP to intercept and analyze all HTTP requests sent to the PrestaShop application.

3.3 Automated Active Scan

An Active Scan was initiated against the target application:

- Target: http://192.168.56.102
- Scan Type: Active Scan

OWASP ZAP performed automated testing for common vulnerabilities including injection attacks, cross-site scripting (XSS), authentication issues, and security misconfigurations.

Vulnerabilities Identified (Before WAF):

- Cross-Site Scripting (XSS)
- Missing Anti-CSRF Tokens
- Cookies without HttpOnly flag
- Information Disclosure via headers

These findings align with multiple categories in the **OWASP Top 10**.

4. Defense Phase – Web Application Firewall Implementation

4.1 Tool Used: ModSecurity

ModSecurity is an open-source Web Application Firewall (WAF) module for Apache that provides real-time protection against common web attacks.

4.2 Installation and Configuration

ModSecurity was installed and enabled on the Apache web server. The recommended configuration file was copied and activated: /etc/modsecurity/modsecurity.conf

Key configuration change: SecRuleEngine On

This enabled full blocking mode instead of detection-only mode.

4.3 OWASP Core Rule Set (CRS)

The OWASP Core Rule Set was enabled to provide standardized protection against known attack patterns such as SQL injection and XSS.

5. Security Validation (Re-Attack Phase)

After enabling ModSecurity, the OWASP ZAP Active Scan was executed again against the same target.

Observed Results:

- Fewer alerts detected by OWASP ZAP

- Multiple requests blocked by the WAF
- HTTP 403 Forbidden responses returned
- ModSecurity audit logs recorded blocked attacks

This confirms that the WAF successfully mitigated several attack vectors previously identified.

6. PrestaShop Security Checklist (Summary)

The following security best practices were reviewed and partially implemented:

- Secure authentication mechanisms
- Input validation and sanitization
- Web Application Firewall deployment
- Reduced information leakage via headers
- Monitoring through audit logs

7. Mapping to OWASP Top 10

OWASP Category	Observation
A03: Injection	Injection attempts blocked by WAF
A05: Security Misconfiguration	Improved via ModSecurity
A07: Identification & Authentication Failures	Login testing performed

8. Conclusion

This assignment demonstrated the complete lifecycle of web application security testing: attack, defense, and validation. OWASP ZAP was effective in identifying vulnerabilities within the PrestaShop application, while ModSecurity significantly reduced the attack surface by blocking malicious requests.

The exercise highlights the importance of layered security controls, regular vulnerability scanning, and proactive defense mechanisms in protecting modern web applications.

9. Appendix

Appendix A: WAF Rule File

- File Path: /etc/modsecurity/modsecurity.conf
- Key Directive: SecRuleEngine On

Appendix B: Tools Used

- OWASP ZAP
- ModSecurity
- Apache Web Server
- Kali Linux

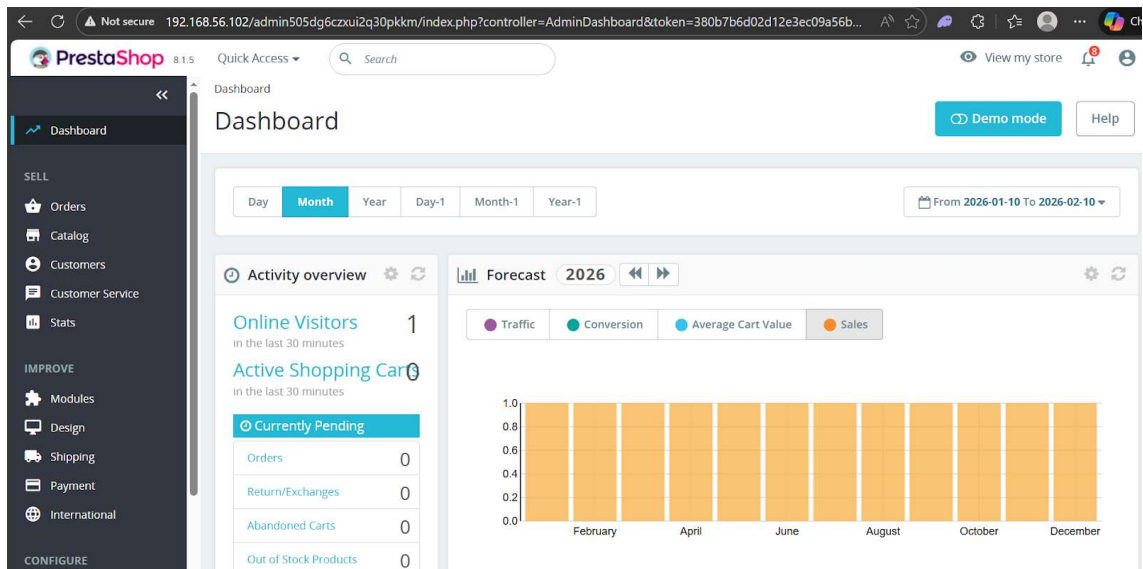


Fig.1 image indicating pretashop admin Dashboard

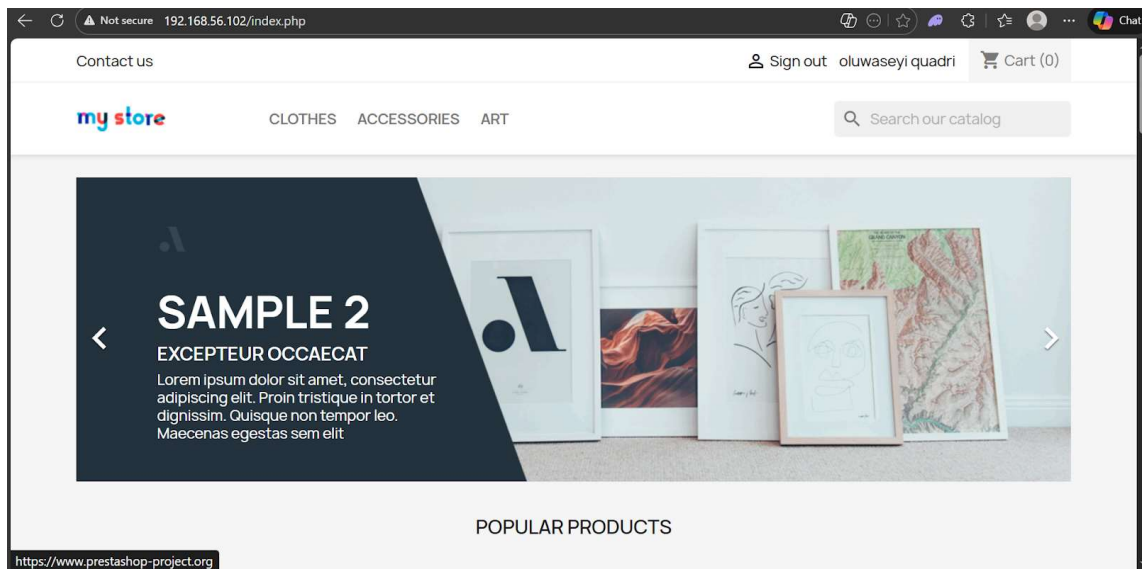


Fig.2 Image indicating the dashboard of a regular user in the prestashop

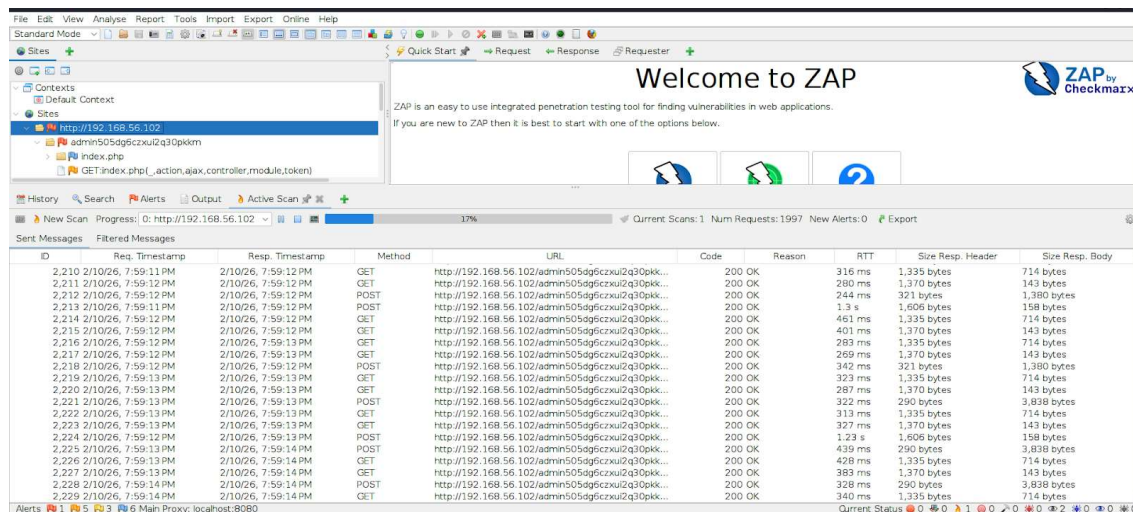


Fig.3 image indicating Zap running an active scan before Modsecurity enabling

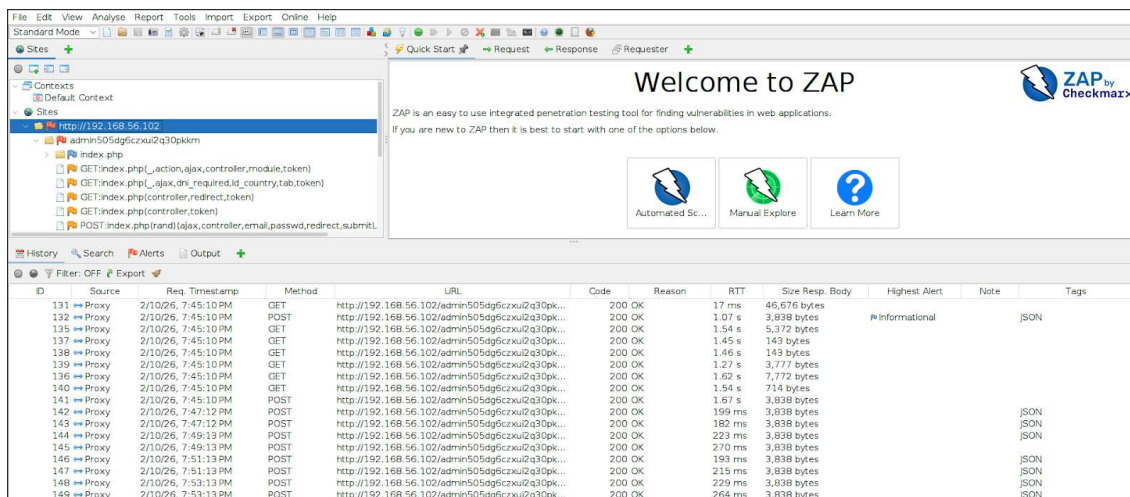


Fig.4 image indicating a successful scan on Zap

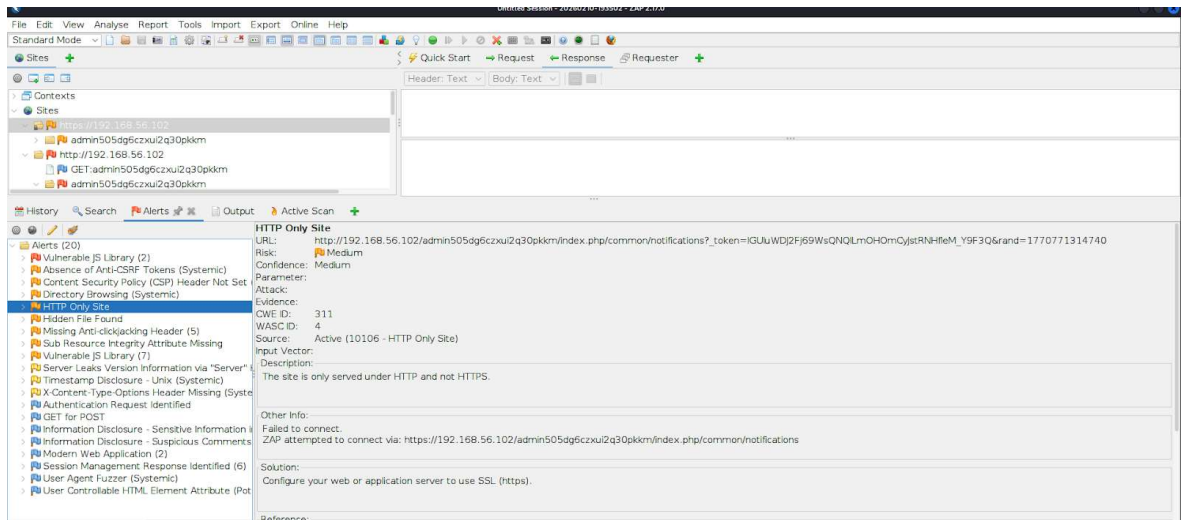


Fig.5 image indicating alerts after scanning on Zap

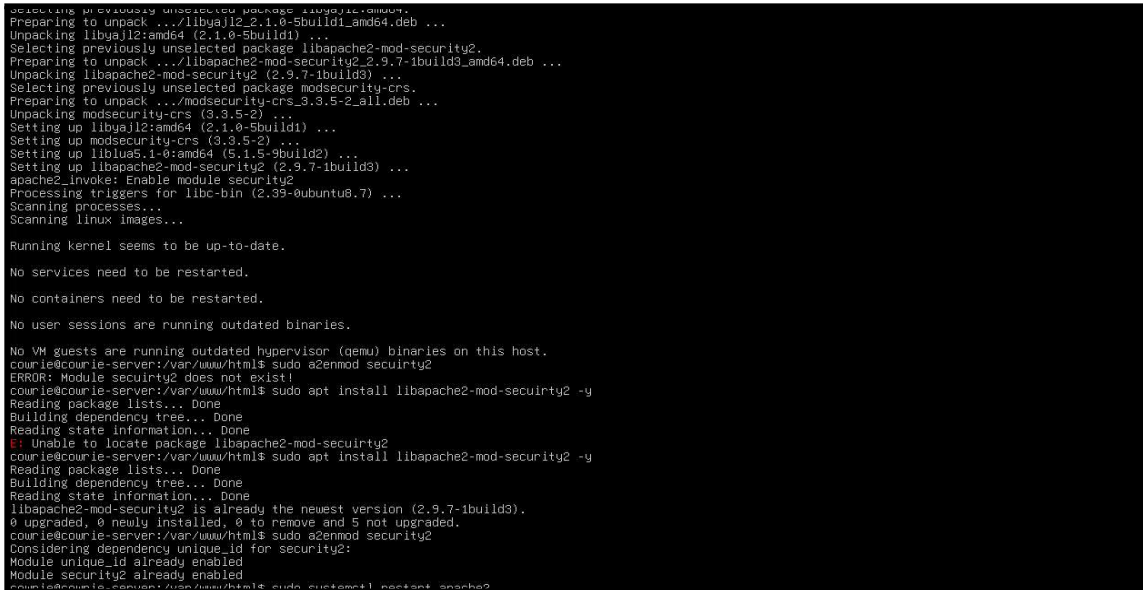


Fig.6 Image showing installation of Modsecurity

```
GNU nano 7.2 /modsecurity.conf *
# Rule engine initialization
# Enable ModSecurity, attaching it to every transaction. Use detection
# only to start with, because that minimises the chances of post-installation
# disruption.
SecRuleEngine On

# -- Request body handling
# Allow ModSecurity to access request bodies. If you don't, ModSecurity
# won't be able to see any POST parameters, which opens a large security
# hole for attackers to exploit.
SecRequestBodyAccess On

# Enable XML request body parser.
# Initiate XML Processor in case of xml content-type
#
SecRule REQUEST_HEADERS:Content-Type "(?:application(?:/soap|/)|text/xml)" \
  "id:'200000',phase:1,t:none,t:lowercase,pass,nolog,ctl:requestBodyProcessor=XML"

# Enable JSON request body parser.
# Initiate JSON Processor in case of JSON content-type; change accordingly
# If your application does not use 'application/json'
#
SecRule REQUEST_HEADERS:Content-Type "application/json" \
  "id:'200001',phase:1,t:none,t:lowercase,pass,nolog,ctl:requestBodyProcessor=JSON"

# Sample rule to enable JSON request body parser for more subtypes.
# Uncomment or adapt this rule if you want to engage the JSON
# Processor for "+json" subtypes
#
SecRule REQUEST_HEADERS:Content-Type "application/(a-z0-9-)+json" \
  "id:'200006',phase:1,t:none,t:lowercase,pass,nolog,ctl:requestBodyProcessor=JSON"

# Maximum request body size we will accept for buffering. If you support
# file uploads then the value given on the first line has to be as large
# as the largest file you are willing to accept. The second value refers
# to the size of data, with files excluded. You want to keep that value as
# low as practical.
#
SecRequestBodyLimit 13107200
SecRequestBodyFilesLimit 131072
```

Fig.7 image showing Modsecurity configuration Of WAP

ID	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
20,623	2/10/26, 9:45:27 PM	2/10/26, 9:45:27 PM	GET	http://192.168.56.102/js/admin.js?v=%27case+wh...	403	Forbidden	217 ms	216 bytes	279 bytes
20,624	2/10/26, 9:45:27 PM	2/10/26, 9:45:27 PM	GET	http://192.168.56.102/js/admin.js?v=%22case+wh...	403	Forbidden	144 ms	216 bytes	279 bytes
20,625	2/10/26, 9:45:27 PM	2/10/26, 9:45:27 PM	GET	http://192.168.56.102/js/admin.js?v=8.1.5+%2F+c...	200	OK	287 ms	316 bytes	49,697 bytes
20,626	2/10/26, 9:45:25 PM	2/10/26, 9:45:27 PM	GET	http://192.168.56.102/index.php?action=getAllWishl...	404	Not Found	2.18 s	346 bytes	45,349 bytes
20,627	2/10/26, 9:45:27 PM	2/10/26, 9:45:28 PM	GET	http://192.168.56.102/js/query	301	Moved Permanently	246 ms	268 bytes	320 bytes
20,628	2/10/26, 9:45:27 PM	2/10/26, 9:45:28 PM	GET	http://192.168.56.102/index.php?action=getAllWishl...	403	Forbidden	594 ms	216 bytes	279 bytes
20,629	2/10/26, 9:45:28 PM	2/10/26, 9:45:28 PM	GET	http://192.168.56.102/index.php?action=getAllWishl...	403	Forbidden	141 ms	216 bytes	279 bytes
20,630	2/10/26, 9:45:28 PM	2/10/26, 9:45:29 PM	GET	http://192.168.56.102/index.php?action=getAllWishl...	403	Forbidden	477 ms	216 bytes	279 bytes
20,631	2/10/26, 9:45:29 PM	2/10/26, 9:45:29 PM	GET	http://192.168.56.102/js/query/plugins	301	Moved Permanently	350 ms	276 bytes	328 bytes
20,632	2/10/26, 9:45:29 PM	2/10/26, 9:45:29 PM	GET	http://192.168.56.102/index.php?action=getAllWishl...	403	Forbidden	545 ms	216 bytes	279 bytes
20,633	2/10/26, 9:45:29 PM	2/10/26, 9:45:29 PM	GET	http://192.168.56.102/js/query/plugins/alerts	301	Moved Permanently	467 ms	283 bytes	335 bytes
20,634	2/10/26, 9:45:30 PM	2/10/26, 9:45:30 PM	GET	http://192.168.56.102/js/query/plugins/chosen	301	Moved Permanently	486 ms	283 bytes	335 bytes
20,635	2/10/26, 9:45:31 PM	2/10/26, 9:45:31 PM	GET	http://192.168.56.102/js/query/plugins/fancybox	301	Moved Permanently	212 ms	285 bytes	337 bytes
20,636	2/10/26, 9:45:31 PM	2/10/26, 9:45:31 PM	GET	http://192.168.56.102/js/query/plugins/growl	301	Moved Permanently	208 ms	282 bytes	334 bytes
20,637	2/10/26, 9:45:27 PM	2/10/26, 9:45:32 PM	GET	http://192.168.56.102/index.php?fc=module&modul...	404	Not Found	4.86 s	346 bytes	47,221 bytes
20,638	2/10/26, 9:45:32 PM	2/10/26, 9:45:32 PM	GET	http://192.168.56.102/index.php?fc=module&modul...	403	Forbidden	316 ms	216 bytes	279 bytes
20,639	2/10/26, 9:45:32 PM	2/10/26, 9:45:32 PM	GET	http://192.168.56.102/index.php?fc=module&modul...	403	Forbidden	275 ms	216 bytes	279 bytes
20,640	2/10/26, 9:45:32 PM	2/10/26, 9:45:32 PM	GET	http://192.168.56.102/js/query/plugins/timepicker	301	Moved Permanently	66 ms	287 bytes	339 bytes
20,641	2/10/26, 9:45:33 PM	2/10/26, 9:45:33 PM	GET	http://192.168.56.102/js/query/ui	301	Moved Permanently	19 ms	271 bytes	323 bytes
20,642	2/10/26, 9:45:32 PM	2/10/26, 9:45:33 PM	GET	http://192.168.56.102/index.php?fc=module&modul...	403	Forbidden	415 ms	216 bytes	279 bytes

Fig.8 image scanning after enabling the ModSecurity on zap

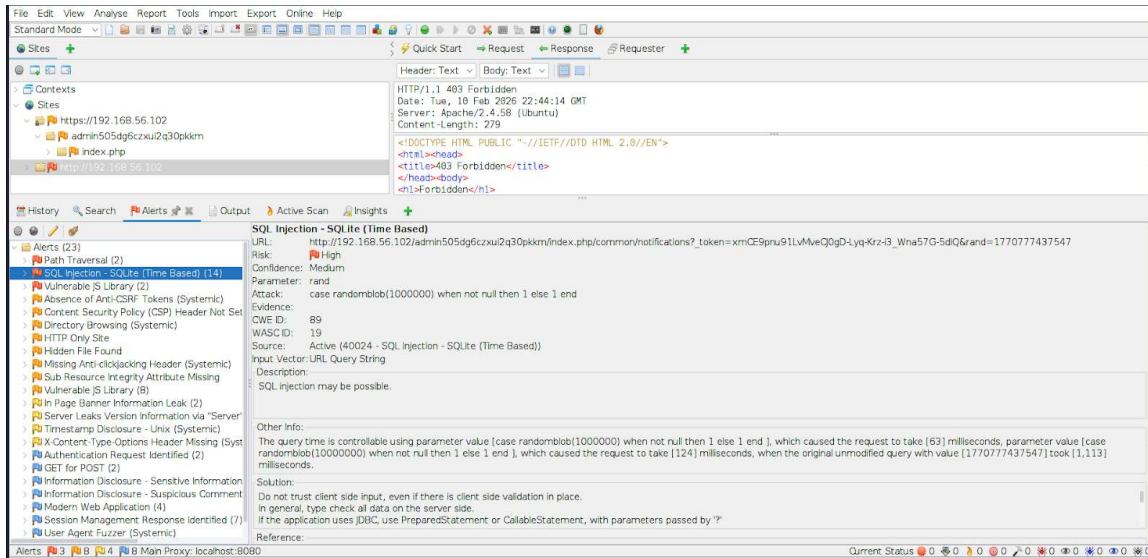


Fig. 9 image showing scan result after enabling ModSecurity