**Test plan for**

**Web application firewall**

*ChangeLog*

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Change Date** | **By** | **Description** |
| version number | Date of Change | Name of person who made changes | Description of the changes made |
| 1.0 | 30/10/2023 | Prachi Sharma | Initial Draft |
|  |  |  |  |

Mr. Abhishek Goyal Mr. Abhishek Goyal

(Project Guide) (Lab Faculty)

Submitted by: Submitted to:

Manya Varshney, 2000290120093, 7B Neha Shukla

Prachi Sharma, 2000290120108, 7B (Project Co-ordinator)

Priyansha Singhal, 2000290120116, 7B

1 Introduction 2

1.1 Scope 2

1.2 Quality Objective 2

1.3 Roles and Responsibilities 2

2 Test Methodology 3

2.1 Overview 3

2.2 Test Levels 3

2.3 Test Completeness 4

3 Test Deliverables 4

4 Resource & Environment Needs 4

4.1 Testing Tools 4

4.2 Test Environment 4

5 Terms/Acronyms 5

**Introduction**

Test strategies for web application firewalls (WAFs) are essential to ensure that they effectively protect web applications from various security threats. Here's a brief introduction to some key strategies:

* **Positive Security Model Testing:** Verify that the WAF allows legitimate traffic while blocking malicious requests. Test this by sending known valid inputs and confirming they are permitted.
* **Negative Security Model Testing:** Test how well the WAF can detect and block malicious traffic. Use a variety of common web application attacks like SQL injection, cross-site scripting (XSS), and remote file inclusion to assess its effectiveness.

**Scope**

When creating a testing report for a web application firewall (WAF), it's essential to consider both functional and non-functional requirements. These requirements help ensure that the WAF is effective, efficient, and meets the desired security standards. Here are examples of functional and non-functional requirements for a WAF testing report:

**Functional Requirements:**

* **Rule Based Blocking:** The WAF should effectively block known and potential security threats such as SQL injection, cross-site scripting (XSS), and other web application attacks.
* **Protection Against Zero-day Vulnerabilities:** The WAF should be able to detect and block attempts to exploit unknown vulnerabilities (zero-day attacks) by heuristic analysis or behavior-based detection.
* **Content Inspection:** The WAF should inspect and filter incoming and outgoing content for malicious code and threats.
* **Session Management:** The WAF should have the ability to manage and protect user sessions from attacks like session fixation and session hijacking.

**Non-Functional Requirements:**

1. **Performance**: The WAF should not significantly degrade the performance of the web application. It must handle expected traffic loads without introducing noticeable latency.
2. **Scalability**: Ensure that the WAF can scale to accommodate increased traffic and resource demands as the web application grows.
3. **Reliability**: The WAF should be highly available and reliable, minimizing downtime or disruption to the web application.
4. **Security Updates**: It should receive regular updates and patches to protect against new threats and vulnerabilities.
5. **Accuracy**: The WAF must accurately distinguish between legitimate and malicious traffic while minimizing false positives and false negatives.

**Quality Objective**

Some objectives are:

* Ensure the Application Under Test conforms to functional and non-functional requirements
* Bugs/issues are identified and fixed before go live

**Roles and Responsibilities**

Developers – Manya Varshney , Prachi Sharma , Priyansha Singhal

Test Manager – Mr. Abhishek Goyal

**Test Methodology**

**Overview**

The decision to adopt a Waterfall methodology for a project is typically based on specific project requirements, constraints, and organizational factors. Here are some common reasons for choosing the Waterfall methodology:

* Well-Defined Requirements: When the project has clearly defined and stable requirements that are unlikely to change significantly throughout the project's lifecycle. Waterfall is suitable when you can gather and document all the requirements up front.
* Low Uncertainty: If there is a high level of confidence in the project scope and objectives, and the technology and processes to be used are well-understood, Waterfall can be a good choice. It is less adaptable to uncertainty and change.
* Regulatory Compliance: In cases where the project needs to adhere to strict regulatory or compliance standards, Waterfall provides a structured and documented approach that can help meet these requirements.
* Large-Scale and Complex Projects: Waterfall can be beneficial for large-scale, complex projects where a comprehensive and detailed project plan is essential for successful execution.

Top of Form

**Test Levels**

Testing a Web Application Firewall (WAF) typically involves multiple test levels to ensure comprehensive coverage of its security features and effectiveness. These test levels can be organized as follows:

1. Unit Testing:
   * Rule Validation: Verify that individual security rules within the WAF are correctly configured and accurately detect or block specific types of attacks.
   * Logging and Alerting: Test that the WAF generates appropriate logs and alerts for specific rule violations.
2. Integration Testing:
   * Rule Interaction: Assess how different security rules interact when multiple rules are applied to the same request or response. Ensure they do not conflict or produce unintended outcomes.
   * Communication with Other Security Components: Test the WAF's ability to integrate with other security components in your infrastructure, such as intrusion detection systems (IDS) or load balancers.
3. System Testing:
   * Rule Coverage: Validate that the WAF provides comprehensive coverage for known vulnerabilities and attacks, including SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and other common web application threats.
   * Custom Rule Testing: Ensure that any custom rules configured to protect application-specific vulnerabilities are working as intended.

**Test Completeness**

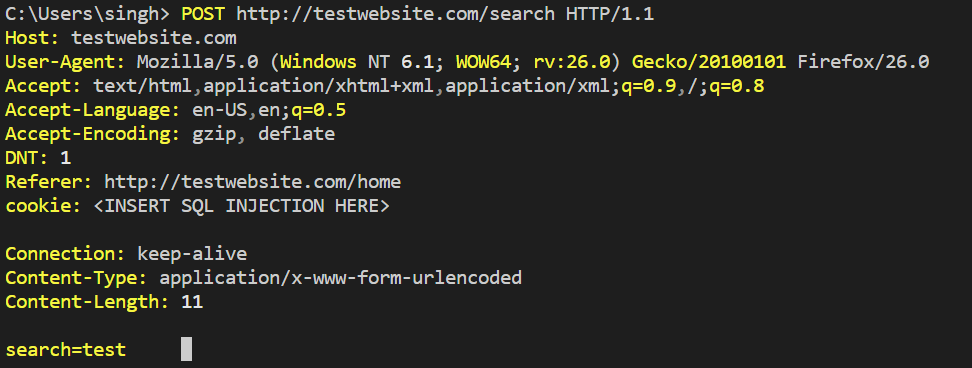
Few criteria to check Test Completeness are:

* 100% test coverage
* All open bugs are fixed or will be fixed in next release

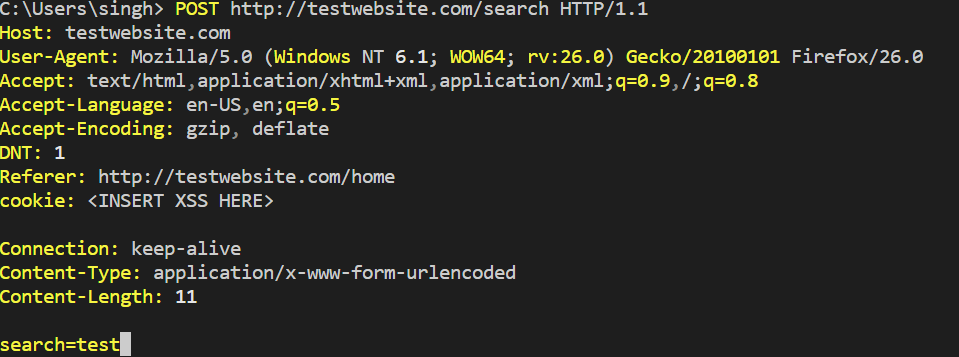
**Test Deliverables**

Testing a Web Application Firewall (WAF) involves creating various test cases to ensure it effectively detects and mitigates web application threats. Here are some example test cases for a WAF:

1. **SQL Injection Test Cases**: a. Send a request with a SQL injection payload in a URL parameter. b. Inject SQL code into form fields and verify that the WAF blocks or alerts on the malicious request. c. Use various SQL injection techniques (e.g., UNION-based, Blind SQLi) to test the WAF's detection capabilities.

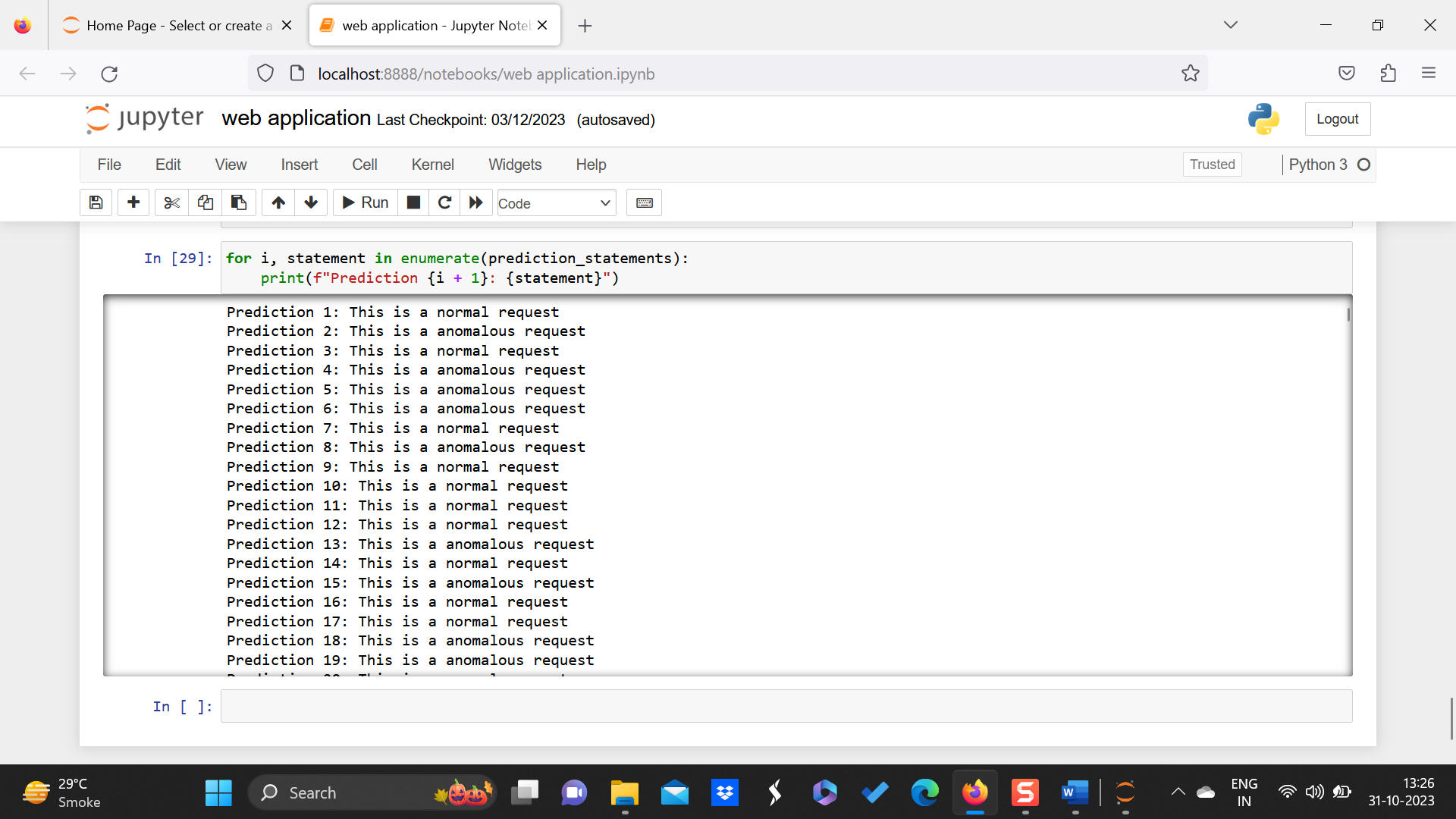


1. **Cross-Site Scripting (XSS) Test Cases**: a. Inject XSS payloads in input fields and assess whether the WAF detects and blocks them. b. Test for various types of XSS attacks, including stored, reflected, and DOM-based XSS.



1. **Cross-Site Request Forgery (CSRF) Test Cases**: a. Craft a CSRF attack and verify that the WAF prevents the forged request from being executed. b. Test with different HTTP methods (e.g., POST, GET) to ensure protection against CSRF attacks.





**Resource & Environment Needs**

**Test Environment**

It mentions the minimum **software** requirements that will be used to test the Application.

Following **software’s** are required in addition to client-specific software.

1. Windows 8 and above
2. Web server

**Terms/Acronyms**

Make a mention of any terms or acronyms used in the project

| **TERM/ACRONYM** | **DEFINITION** |
| --- | --- |
| API | Application Program Interface |
| AUT | Application Under Test |