

FEBRUARY 20, 2026

TECHNICAL INTERNSHIP PROJECT REPORT

DEVELOPMENT OF A MODULAR
PAYLOAD GENERATION FRAMEWORK

Team Lead: Muhammad Ahsan Ayaz
Prepared by: Team Beta
Offensive Security Interns

TABLE OF CONTENTS

• Executive Summary	Page 3
• Project Objective	Page 3
• Technical Architecture	Page 4
◦ 3.1 Core Components	
◦ 3.2 Technology Stack	
• Module Breakdown	Page 5
◦ 4.1 XSS Payload Engine	
◦ 4.2 SQL Injection Simulation	
◦ 4.3 Command Injection Patterns	
• Advanced Features & Obfuscation	Page 6
• Ethical Compliance & OWASP Standards	Page 6
• Conclusion & Deliverables	Page 7

03

EXECUTIVE SUMMARY

This report details the design and development of a Python-based CLI tool developed during the internship tenure at IT SOLERA. The Payload Generation Framework is an educational utility designed to demonstrate exploitation patterns and defensive bypass logic. Aligned with OWASP ethical standards, the tool focuses on simulating payload structures for XSS, SQLi, and Command Injection to help security professionals understand and harden Web Application Firewalls (WAFs).

PROJECT OBJECTIVE

The primary goal was to bridge the gap between theoretical vulnerability research and practical defensive implementation.

Modular Design: Create an extensible framework where new attack patterns can be added as modules.

Defensive Analysis: Showcase why and how certain payloads bypass standard regex filters.

Safety & Ethics: Ensure the tool remains a template generator with no active "attack" or "request-sending" capabilities.

04

TECHNICAL ARCHITECTURE

The tool is built using a modular object-oriented approach in Python.

3.1 Core Components

- CLI Interface: Built using argparse, allowing users to specify modules, database types, and encoding formats via flags.
- Generation Engine: A logic-based engine that constructs strings based on the "Context" (e.g., HTML vs. JavaScript).
- Transformation Layer: A post-processing module that applies obfuscation (Base64, URL Encoding, Hex, and Whitespace abuse).

3.2 Technology Stack

- Primary Language: Python 3.10+
- Libraries: argparse, json, base64, re
- Documentation: Markdown / GitHub Wiki

MODULE BREAKDOWN

4.1 XSS Payload Module

Demonstrates Cross-Site Scripting concepts through context-aware templates:

- **Reflected/Stored/DOM:** Templates designed for specific injection points.
- **Bypass Logic:** Implements case manipulation (e.g., ``) and tag switching to evade simple string-match filters.
`</p>`

4.2 SQL Injection Module (Simulation)

Generates database-specific strings for **MySQL**, **PostgreSQL**, and **MSSQL**:

- **Techniques:** Includes Union-based and Error-based string generation.
- **Evasion:** Demonstrates comment-based bypasses (e.g., `SELECT/*internal*/password`).

4.3 Command Injection Module

Focuses on OS-level command separation:

- **OS Logic:** Differentiates between Windows (`&, ||`) and Linux (`;, &&`) separators.
- **Pattern Analysis:** Explains the failure of blacklisted character filters.

06

ADVANCED FEATURES & OBFUSCATION

The framework includes a secondary layer of "Evasion Logic" to simulate real-world attacker behavior:

- **Encoding:** Automatic conversion to URL, Base64, and Hex formats.
- **Whitespace Abuse:** Utilizing tabs and newlines to break signature-based detection.
- **Export Formats:** Supports .json and .txt exports for integration into professional scanners like **Burp Suite** or **OWASP ZAP**.

ETHICAL COMPLIANCE & OWASP STANDARDS

As per guidelines:

1. No Live Traffic: The tool does not send HTTP requests by default.
2. Educational Labeling: Every generated payload is accompanied by a disclaimer.
3. OWASP Alignment: Developed in strict accordance with the OWASP Code of Ethics and the Web Security Testing Guide (WSTG).

CONCLUSION & DELIVERABLES

The project successfully meets the internship requirements by providing a functional, documented, and ethically sound security tool.

- Repository: [GitHub Link](#)
- Documentation: Comprehensive README.md and ethics guide included.
- Impact: This framework serves as a foundational tool for internal security training and WAF configuration testing.