Incident Response Playbook: Cross-Site Scripting (XSS)

Team AnubisX

Version 1.0
September 18, 2025

Document Control

Attribute	Value
Version	1.0
Status	Final
Owner	AnubisX Security Team
Review Cycle	Every Quarter

Contents

1 Introduction

1.1 Purpose

The purpose of this playbook is to provide a structured incident response plan for handling Cross-Site Scripting (XSS) incidents. The objective is to detect, analyze, contain, and eradicate XSS vulnerabilities and attacks, minimizing impact on users and applications.

1.2 Scope

This playbook applies to all web applications, APIs, and services hosted by the organization. It covers reflected, stored, and DOM-based XSS, user impact assessment, and remediation activities.

2 Overview of a Cross-Site Scripting (XSS) Attack

XSS is a web application vulnerability that allows attackers to inject malicious scripts into web pages viewed by users. These scripts can hijack sessions, steal cookies, redirect users, or execute malicious actions in the context of the victim's browser. Common exploitation vectors include input fields, URLs, and vulnerable client-side code.

3 Incident Response Phases

This playbook follows the NIST Incident Response lifecycle framework.

3.1 Phase 1: Preparation

Goal: To ensure the team is prepared to detect and respond to XSS incidents.

- **Roles and Responsibilities:** Define roles (Incident Commander, Web Security Lead, Application Owner, SOC Analysts, DevSecOps).
- **Tools & Resources:** Ensure availability of WAF, SIEM, DAST/SAST, CSP monitoring, and browser developer tools.
- **Controls:** Implement input validation, output encoding, Content Security Policy (CSP), and secure coding practices.
- **Training:** Conduct web security awareness for developers and QA teams.

3.2 Phase 2: Identification Analysis

Goal: To confirm XSS attack occurrence and determine its scope and severity.

1. **Initial Triage:** Collect WAF alerts, application logs, payload samples, and user reports. Open an official incident ticket and activate secure communication channels.
2. **Initial Analysis and IOC Evaluation:** Analyze logs and alerts to identify Indicators of Compromise (IOCs). Common IOCs include:
    - Logs showing suspicious script tags or encoded payloads in HTTP requests.
    - WAF alerts detecting XSS payload patterns.
    - User reports of unauthorized pop-ups, redirects, or credential prompts.
    - Observed cookie/session exfiltration attempts via JavaScript.

3. **Severity Level Assessment:** Classify the incident to ensure appropriate allocation of resources. Severity is determined based on the type of XSS vulnerability (e.g., reflected vs. stored), the scope of user impact, and the potential damage, such as session hijacking.

Level	Description	Example	MTTD	MTTR
Low	Single reflected XSS attempt blocked by WAF, no user impact.	Attempted payload in query string is blocked by WAF.	6-12 hours	24-48 hours
Medium	Successful reflected/DOM XSS affecting limited users.	A malicious link sent via phishing exploits reflected XSS to affect a small group of users.	12-24 hours	2-4 days
High	Stored XSS exploited in production, multiple users impacted.	A persistent script in a public comment section captures session tokens from multiple users.	24-48 hours	4-7 days
Critical	Widespread XSS across core applications causing account/session hijacking.	An injected script steals authentication tokens from thousands of users across the platform.	48 hours	7-14 days

Table 1: Incident Severity Matrix

4. **Alert Validation (TP vs. FP):** Correlate suspicious activity with other data points.
   - **If True Positive (TP):** The activity is confirmed as malicious script execution. **Action:** Immediately proceed to the **Containment** phase, escalate to the Incident Commander, and activate the playbook.
   - **If False Positive (FP):** The activity is confirmed benign (e.g., developer testing). **Action:** Document findings, close the alert, and recommend tuning detection rules.
5. **Incident Declaration:** If confirmed, formally declare an XSS incident and escalate to the Incident Commander, Application Owner, Legal, and Communications.

3.3 Phase 3: Containment

Goal: To limit the attack's scope and prevent further damage.

- **Short-Term Containment (Immediate Actions):**
  - Block malicious input patterns via WAF rules.
  - Temporarily disable affected functionality (e.g., comment fields, user profile forms).
  - Force logout and rotate sessions/tokens for all users if hijacking is suspected.
- **Evidence Preservation:** Preserve logs, payload samples, and database snapshots for forensic analysis.

3.4 Phase 4: Eradication

Goal: To remove the root cause and all malicious artifacts.

- **Root Cause Analysis:** Identify the vulnerable code path.

- **Vulnerability Patching:** Patch the code by implementing proper input validation and context-aware output encoding.
- **Payload Removal:** Remove any injected malicious payloads from databases (for stored XSS).
- **Security Hardening:** Apply or tighten Content Security Policy (CSP) to block inline scripts and untrusted sources.

3.5 Phase 5: Recovery

Goal: To safely restore systems and business operations.

- **System Restoration:** Deploy the patched code to production.
- **Enhanced Monitoring:** Increase monitoring of WAF and application logs for any signs of recurrence.
- **Validation:** Perform regression testing and security scans to confirm the fix.
- **Business Continuity:** Re-enable disabled features gradually while under close observation.

3.6 Phase 6: Post-Incident Activities (Lessons Learned)

Goal: To strengthen resilience and prevent recurrence.

- **Post-Incident Meeting:** Conduct a blameless post-mortem meeting within two weeks of incident closure.
- **Final Incident Report:** Create a detailed report covering the root cause, impact, response actions, and lessons learned.
- **Action Plan:** Update secure coding guidelines, enhance developer training, and improve automated scanning (DAST/SAST) and WAF rules.

4 MITRE ATT&CK Framework Mapping

XSS ATT&CK Mapping

- **Tactic: Initial Access**
  - *T1190 – Exploit Public-Facing Application.*
- **Tactic: Execution**
  - *T1059.007 – Command and Scripting Interpreter: JavaScript.*
- **Tactic: Persistence**
  - *T1505.001 – Server Software Component: Web Shell* (if script facilitates upload).
- **Tactic: Credential Access**
  - *T1539 – Steal Web Session Cookie.*
- **Tactic: Defense Evasion**
  - *T1027 – Obfuscated Files or Information.*
- **Tactic: Exfiltration**
  - *T1041 – Exfiltration Over C2 Channel.*
- **Tactic: Impact**
  - *T1496 – Resource Hijacking.*
  - *T1565.003 – Stored Data Manipulation.*