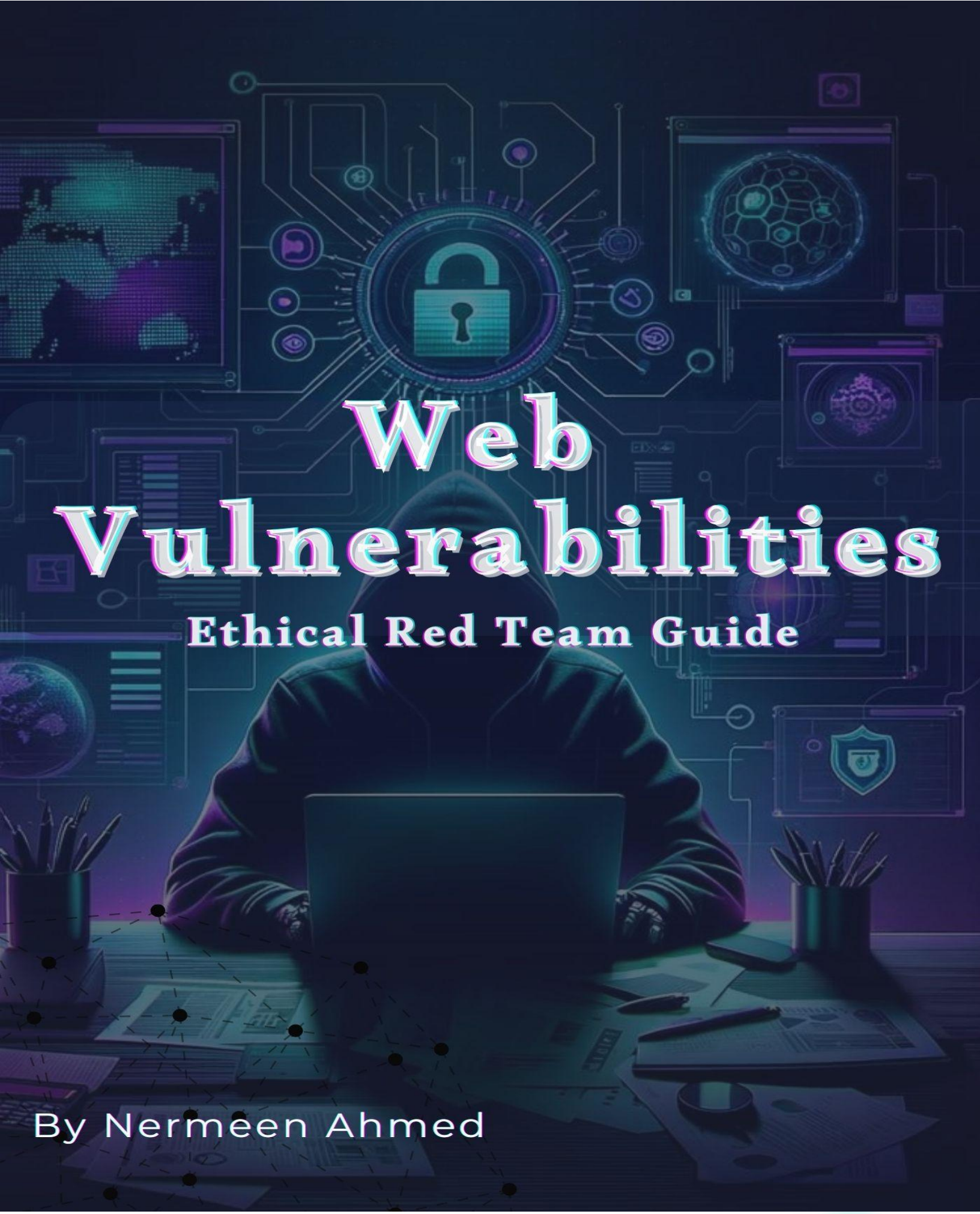
Web
Vulnerabilities
Ethical Red Team Guide
By Nermeen Ahmed

Red Team Awareness Report: Web Vulnerabilities

Author: Nermeen Ahmed Sonbol

Table of Contents

3. Cross-Site Scripting (XSS)

- In-Depth Explanation of XSS
- Types: Stored, Reflected, DOM-Based XSS
- Real-World Consequences
- Comprehensive Mitigation: Encoding, Content Security Policy (CSP)

4. Cross-Site Request Forgery (CSRF)

- How CSRF Exploits Trust
- Impact on User Accounts and Data Integrity
- Defense Mechanisms: Anti-CSRF Tokens, SameSite Cookies

5. Conclusion

6. References / Resources

1. Introduction

The Modern Web Landscape: Web applications are the backbone of modern digital business. However, their complexity and connectivity introduce a wide array of security risks that malicious actors are eager to exploit.

Purpose and Scope: This report provides a detailed, educational overview of common web application vulnerabilities from a Red Team perspective. The focus is on understanding these weaknesses to better defend against them, not to exploit them maliciously. All examples are theoretical and designed for safe, ethical testing in controlled labs.

Ethical Framework: A core tenet of Red Teaming is operating within strict ethical and legal boundaries. Awareness of these vulnerabilities must be coupled with a commitment to using this knowledge solely for improving security posture.

For educational and defensive purposes only. This document is intended for awareness and must not be used to attack live systems. Perform tests only in authorized, controlled environments.

2. SQL Injection (SQLi)

Detailed Definition

SQL Injection is a code injection technique where an attacker can insert or "inject" malicious SQL statements into an entry field (e.g., a login form) for execution by the backend database. This occurs when user input is not properly validated, sanitized, or parameterized.

Potential Impact

- Data Breach: Extraction of sensitive information (usernames, passwords, personal data).
- Data Manipulation: Altering, adding, or deleting database records.
- Authentication Bypass: Gaining unauthorized access to administrative functions.
- Full System Compromise: In some cases, leading to Remote Code Execution on the database server.

Safe, Theoretical Example

Imagine a login form in which user input is inserted directly into a database query. For awareness, this pattern is dangerous because attacker-controlled input can change the logic of the query and lead to unauthorized access.

Do NOT publish or test vulnerable concatenation examples. Instead, demonstrate the correct, safe approach using parameterized queries (pseudocode):

(parameterized queries)

```
sql = "SELECT * FROM users WHERE username = ? AND password = ?" db.execute(sql, [username, password])
```

Responsible note: This example is for educational purposes only. Do not attempt exploitative testing on production systems without explicit written authorization. Follow responsible disclosure procedures if you discover a vulnerability.

Mitigation Strategies

• Prepared Statements (Parameterized Queries): The most effective defense. It separates SQL code from data.

• Stored Procedures: When implemented safely.

• Input Validation: Whitelist allowed characters and patterns.

• Principle of Least Privilege: Database accounts used by the application should have minimal necessary permissions.

3. Cross-Site Scripting (XSS)

In-Depth Explanation

XSS vulnerabilities allow attackers to inject malicious client-side scripts (usually JavaScript) into web pages viewed by other users. The script executes in the victim's browser, hijacking their session, defacing websites, or redirecting them to malicious sites.

Types

• **Stored XSS:** The malicious script is permanently stored on the target server (e.g., in a comment forum) and delivered to every user who visits the page.

• **Reflected XSS:** The malicious script is reflected off the web server, such as in an error message or search result, often requiring the user to click a crafted link.

• **DOM-based XSS**: The vulnerability exists in the client-side code (JavaScript) rather than the server-side code, manipulating the Document Object Model (DOM).

Comprehensive Mitigation

• Output Encoding: Convert potentially dangerous characters into their HTML entities before rendering user-supplied data (e.g., < becomes <).

• Content Security Policy (CSP): A HTTP header that tells the browser which sources of content are trusted, effectively blocking inline scripts and scripts from unauthorized domains.

• Input Validation: Sanitize input on both the client and server-side.

4. Cross-Site Request Forgery (CSRF)

How CSRF Exploits Trust

CSRF tricks an authenticated user into unknowingly submitting a malicious request to a website where they are currently logged in. It exploits the site's trust in the user's browser.

Impact

An attacker could force a user to change their email address, transfer funds, or change their password without their consent.

Defense Mechanisms

- **Anti-CSRF Tokens:** A unique, unpredictable token is associated with the user's session and included in every state-changing form or request. The server validates this token before processing the request.
- **SameSite Cookie Attribute:** Setting the SameSite attribute on cookies to Strict or Lax prevents the browser from sending cookies in cross-site requests, mitigating CSRF attacks.
- **Re-authentication:** Requiring the user to re-enter their password for sensitive actions.

5. Conclusion

A deep understanding of web application vulnerabilities is fundamental for any Red Team professional. This knowledge, when applied within a strict ethical framework, empowers organizations to identify and remediate risks proactively. The role of the Red Team is not just to break systems, but to build a more resilient and secure organization through continuous testing, education, and collaboration.

6. References / Resources

1. **OWASP Foundation:** https://owasp.org/ (Especially the OWASP Top 10, Testing Guide, and Cheat Sheets)
2. **PortSwigger Web Security Academy:** https://portswigger.net/web-security (Free, excellent learning resource)
3. **NIST Cybersecurity Framework & Guidelines**
4. **SANS Institute Reading Room** (Whitepapers on various security topics)
5. **Hack The Box / TryHackMe** (Platforms for practical, ethical skill development in controlled environments)