Assignment 6.1: Vulnerability Analysis

Joshua A Wortz

University of San Diego

Course Number: CSOL-500

Dr. Michelle Moore

02/16/2025

Cyber attackers exploiting common security vulnerabilities are increasingly targeting web applications. SQL Injection (SQLi), Cross-Site Scripting (XSS), and Server-Side Request Forgery (SSRF) are among the most prevalent and dangerous. Effective mitigation strategies can easily safeguard these vulnerabilities, as they all have significant real-world implications.

SQL Injection (SQLi)

SQL Injection is a code injection technique that allows attackers to manipulate SQL queries and gain unauthorized access to databases. By injecting malicious SQL code into application input fields, attackers can bypass authentication mechanisms, access confidential information, and modify or delete data.

The exploitation mechanism attackers utilize is to insert malicious SQL statements into input fields, potentially allowing them to retrieve, modify, or delete sensitive data. For example, it is common to validate that a username exists in an SQL database. Attackers can take advantage of this standard check by sending specific SQL strings in the username dialog that will always result in a true statement, allowing for the bypassing of authentication safeguards.

The Heartland Payment Systems breach (Reidenbach & Wang, 2021) was one prominent example of an SQLi attack that exposed millions of credit card records, emphasizing the financial and reputational risks of inadequate database security. The attackers utilized the external web application meant for Heartland's customers to track purchases made using payment cards in their retail stores. The attackers gained access to the database using an SQLi attack that began in 2007, but the breach remained undiscovered until 2009.

SQLi can compromise sensitive information, leading to financial loss, legal consequences, and reputational damage. Attackers can exploit SQLi vulnerabilities to manipulate databases, extract confidential customer data, and alter business-critical records. This type of

breach often results in regulatory penalties, class-action lawsuits, and a loss of consumer trust, which can negatively impact a company's financial stability and brand credibility. Furthermore, affected organizations may face operational disruptions and increased security costs to remediate the vulnerability and prevent future attacks.

One common way to secure against SQLi is to use secure coding practices, such as preparing statements to prevent SQLi (Abikoye et al., 2020). Coding safety checks for user inputs into web applications should be used to avoid escaping characters and other SQLi techniques from passing queries from user input directly into the SQL database. Another standard tool is the implementation of Web application firewalls. WAFs can filter and block malicious SQL inputs (Zukran & Siraj, 2021) and intercept queries before they reach the web server.

Cross-Site Scripting (XSS)

XSS vulnerabilities are similar to SQLi in that they are also injection-type attacks. However, the targets of XSS attacks are typically the end users' browsers. There are multiple techniques; however, the common theme is that a user receives a website with malicious Javascript code. This code can often leak private information like session cookies, allowing attackers to spy on or pretend to be others without knowing the username or password.

Attackers exploit input validation flaws to execute scripts in users' browsers, potentially leading to session hijacking or data theft. By injecting malicious JavaScript code, attackers can gain unauthorized access to user sessions, steal credentials, or redirect victims to phishing websites.

The British Airways data breach (Aljaidi, 2023) involved a URL redirect attack facilitated by XSS, exposing customer payment data. Attackers manipulated web page elements

to intercept sensitive payment information, leading to significant financial and reputational consequences for the airline.

XSS attacks compromise user privacy, facilitate phishing, and damage brand trust. Attackers can utilize XSS to distribute malware, deface websites, or escalate privileges within an application. Organizations suffering from XSS attacks often experience decreased consumer confidence, financial liabilities, and the need for extensive remediation efforts to patch security vulnerabilities and reassure customers.

Validating user inputs and encoding output prevents script execution (Hsiao & Wang, 2023). Secure input handling techniques ensure the application only processes expected data formats and prevents unintended code from being executed alongside trusted code. Content Security Policy (CSP) restricts the execution of unauthorized scripts, mitigating XSS risks. This security measure prevents malicious scripts from loading and executing within a browser environment.

Server-Side Request Forgery (SSRF)

SSRF vulnerabilities allow attackers to manipulate server-side requests, potentially accessing internal services. These attacks exploit the trust between a web application and backend resources, allowing attackers to retrieve internal data, perform unauthorized actions, or even exploit cloud services.

Attackers trick the server into making unintended requests, which can lead to data exfiltration or system compromise.SSRF vulnerabilities allow attackers to manipulate server-side requests, potentially accessing internal services.

The Capital One data breach (Khan et al., 2022; Novaes Neto et al., 2020) exploited an SSRF flaw in Amazon Web Services (AWS), exposing millions of sensitive records. Attackers

leveraged misconfigured access controls to gain unauthorized access to confidential customer information stored in cloud databases.

SSRF can lead to unauthorized data access, cloud service abuse, and further internal network exploitation. It can also be used as an entry point for larger-scale attacks, enabling attackers to pivot deeper into an organization's infrastructure. SSRF vulnerabilities can be the starting point for data breaches, can cause operational downtime, and can lead to significant compliance violations if left unmitigated.

Proper access control policies limit server access to trusted sources (Khan et al., 2022). Organizations should enforce strict allow-lists and ensure that only necessary services can make external requests. As highlighted in the Capital One case, enforcing security mechanisms in cloud environments reduces SSRF risks (Novaes Neto et al., 2020). Organizations should implement security measures such as network segmentation and request filtering to limit exposure to SSRF threats.

Conclusion

Addressing application vulnerabilities requires a combination of secure coding practices, automated detection tools, and adherence to cybersecurity best practices. Failing to mitigate these risks can lead to significant financial, operational, and reputational damages. Real-world case studies and academic insights highlight the importance of proactive security measures in protecting web applications from evolving threats. Organizations must continually update their security strategies and ensure compliance with established industry standards to minimize the risk of exploitation.

References

Abikoye, O. C., Abubakar, A., Dokoro, A. H., Akande, O. N., & Kayode, A. A. (2020). A novel technique to prevent SQL injection and cross-site scripting attacks using Knuth-Morris-Pratt string match algorithm. *EURASIP Journal on Information Security*, *2020*, 1-14.

Aljaidi, M. (2023, December). A Comprehensive Technical Analysis of URL Redirect Attacks: A Case Study of British Airways Data Breach. In *2023 24th International Arab Conference on Information Technology (ACIT)* (pp. 1-5). IEEE.

Hsiao, W. C., & Wang, C. H. (2023, June). Detection of SQL Injection and Cross-Site Scripting Based on Multi-Model CNN Combined with Bidirectional GRU and Multi-Head Self-Attention. In *2023 5th International Conference on Computer Communication and the Internet (ICCCI)* (pp. 142–150). IEEE.

Khan, S., Kabanov, I., Hua, Y., & Madnick, S. (2022). A Systematic Analysis of the Capital One Data Breach: Critical Lessons Learned. *ACM Transactions on Privacy and Security*, *26*(1), 1–29. https://doi.org/10.1145/3546068

Novaes Neto, N., Madnick, S., de Paula, M. G., & Malara Borges, N. (2020). A Case Study of the Capital One Data Breach (Revised). *SSRN Electronic Journal. doi*, *10*.

Reidenbach, M., & Wang, P. (2021). Heartland Payment Systems: Cybersecurity Impact on Audits and Financial Statement Contingencies. *Issues in Accounting Education*, *36*(2), 93–109. https://doi.org/10.2308/ISSUES-18-010

Zukran, B., & Siraj, M. M. (2021, October). Performance comparison on sql injection and xss detection using open source vulnerability scanners. In *2021 International Conference on Data Science and Its Applications (ICoDSA)* (pp. 61-65). IEEE.