Lab 5.pdf

Sonal Chandrakant Hasbi

I. Introduction

Injection vulnerabilities (SQL, command, XPath, log spoofing) allow an attacker to influence a program's execution by sending crafted input. This lab explores exploitation techniques and mitigation strategies using WebGoat (v2023.8 and v7.1). The goals are: (1) demonstrate how fields can be abused to run unauthorized queries/commands, (2) illustrate parameterized queries and other mitigations, and (3) explain limitations of defenses.

II. Tools & Environment

- **WebGoat 2023.8 / 7.1**: Deliberately vulnerable web application for learning injection attacks.
- **Browser / Developer Tools**: To inspect requests and responses.
- **VM** : Isolates testing environment.

III. Methodology and Results

We followed WebGoat's exercises and recorded payloads and behavior. Screenshots appear after each subsection.

A. SQL Injection (Mitigation) — WebGoat 2023.8 (steps 1–12)

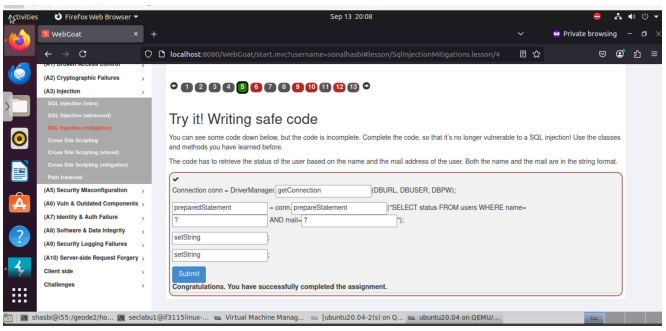

Fig. 1: SQL Injection mitigation (Task 5)

Query Task 9 : SELECT * FROM user_data WHERE last_name = 'a';selectfromuser_system_data;–'

Query Task 10 : a';//seselectlect////frfromom/ × /user_system_data;

Query Task 12 : 104.130.219.202

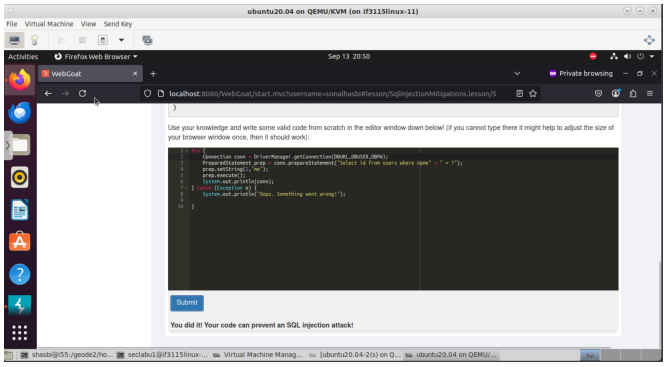

Fig. 2: SQL Injection mitigation (Task 6)

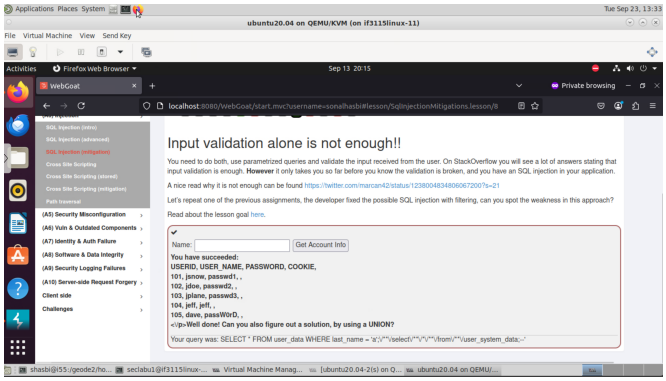

Fig. 3: SQL Injection mitigation (Task 9)

B. Command Injection — WebGoat 7.1

Command injection occurs when user input is passed to a shell command. I clicked on inspect and changed the value to get the output. (Fig. 6)

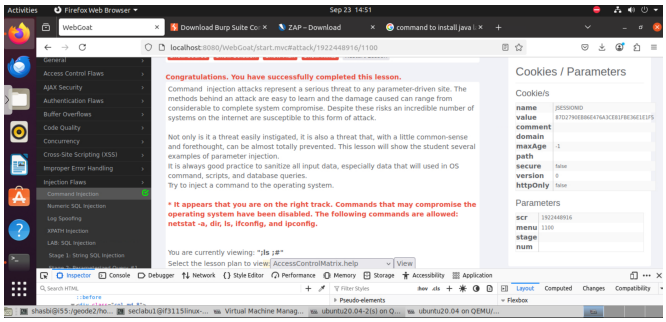

Fig. 6: Command Injection

C. Log Spoofing — WebGoat 7.1

Log spoofing happens when untrusted input is written to logs and later viewed or re-parsed, allowing attackers to inject fake entries or scripts. We tested injecting HTML/script into

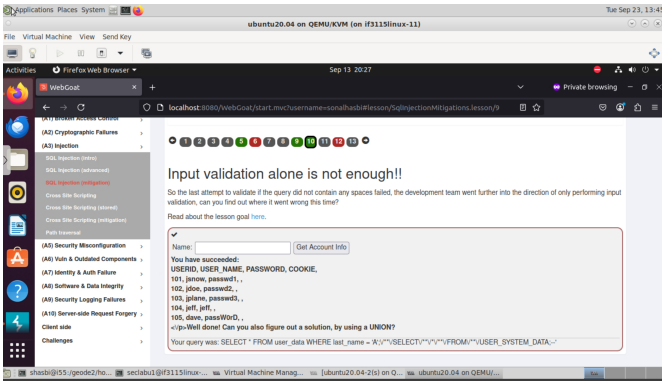

Fig. 4: SQL Injection mitigation (Task 10)

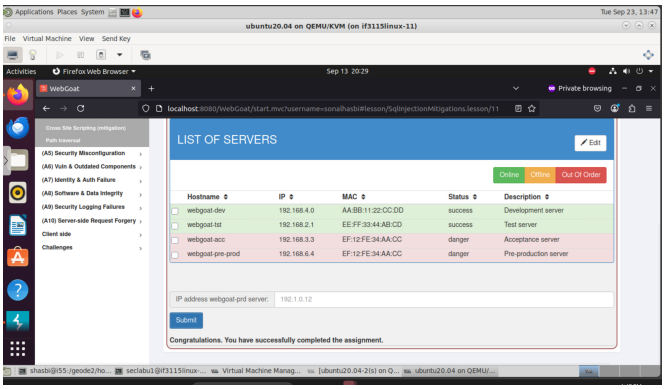

Fig. 5: SQL Injection mitigation (Task 12)

log fields (user-agent, username) to demonstrate log pollution.(Fig. 7) **Query Log Spoofing : Smith%0d%0aLogin Successeded for username: admin**

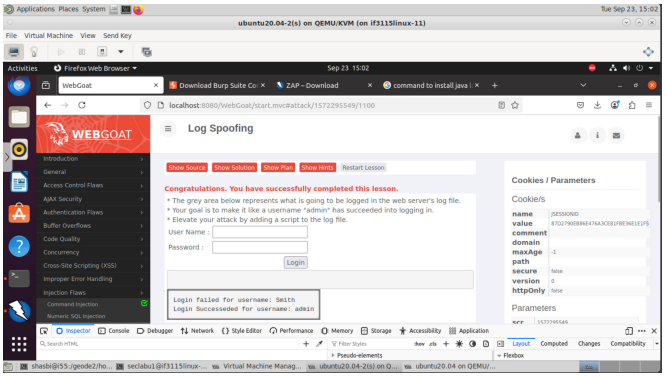

Fig. 7: Log Spoofing

D. XPath Injection — WebGoat 7.1

XPath injection manipulates XPath queries for XML data analogous to SQL injection on relational DBs.(Fig. 8) Payloads to bypass authentication included:

```
' or '1'='1
' or 1=1 or 'a'='a
' or true() or '
```

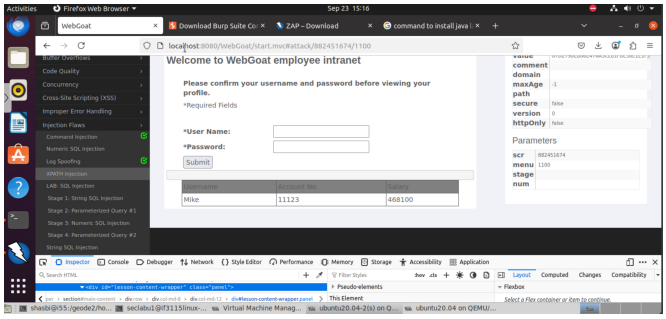

Fig. 8: Xpath Injection

When an XPath expression uses untrusted concatenation, the attacker can create conditions that evaluate to true and retrieve nodes.

IV. Knowledge Gained and Real-World Usefulness

- **HTTP & request/response lifecycle:** Understanding how inputs travel from client to server is crucial for locating injection points.
- **Cookies and sessions:** Attackers may use session-related flaws in combination with injection flaws.
- **Secure development practices:** Parameterized queries, input validation, output encoding, least privilege.
- **Tool proficiency:** Using WebGoat and intercepting proxies helps discover and verify vulnerabilities in a controlled environment.

These skills are directly useful in industry penetration testing, secure SDLC reviews, and patch validation.

V. Limitations of Tools & Suggestions for Improvement

- **WebGoat realism:** intentionally vulnerable apps are excellent for learning but differ from complex real-world systems. Adding exercises that mimic modern frameworks/ORM pitfalls would increase realism.
- **Automation coverage:** integrate more automated test harnesses or auto-generation of payloads for fuzzing.
- **Reporting enhancements:** richer hints on how to fix the exact code sample (not just the vulnerability).

VI. Conclusion

The lab reinforced that injection vulnerabilities are often a result of mixing code and data (string concatenation). Parameterized queries and restricting the contexts in which user input is used are effective mitigations, but developers must avoid constructing SQL identifiers or executing untrusted strings in other interpreters (shells, XPath engines). Defense-in-depth (input validation, output encoding, principle of least privilege, WAFs) provides layered protection.

References

[1] OWASP WebGoat: https://owasp.org/www-project-webgoat/. (Used WebGoat v2023.8 and v7.1 exercises.)
[2] W3Schools: XPath Syntax. https://www.w3schools.com/xml/xpath_syntax.asp.

[3] W3Schools: XPath Reference. https://www.w3schools.com/xml/xml_xpath.asp.
[4] W3C: XML Path Language (XPath) 3.1. https://www.w3.org/TR/2017/REC-xpath-31-20170321/#id-logical-expressions.
[5] OWASP Top Ten — Injection: https://owasp.org/www-community/attacks/SQL_Injection.