Lab 7: Authentication

Shivali Mate

I. Introduction

THIS lab contains hands-on exercises that demonstrate common authentication weaknesses in modern web applications. Exercises were performed in two WebGoat environments. The tasks cover password strength, forgot-password flows, multi-level login (TANs), JWT misuse and bypass techniques, authentication bypass, and password reset weaknesses. Emphasis is placed on exploiting vulnerabilities and reading source code to derive remediation recommendations.

II. Methodology

A. Password Strength

Different passwords require substantially different times and costs to crack because of differences in entropy, predictable patterns, and attacker models. Short or dictionary-based passwords have low entropy and are rapidly discovered by dictionary or offline brute-force attacks, while long passphrases or passwords with high randomness dramatically increase the search space and computational cost. The attacker model matters: an online attacker is constrained by rate limits and lockouts whereas an offline attacker with database hashes can try billions of guesses per second.

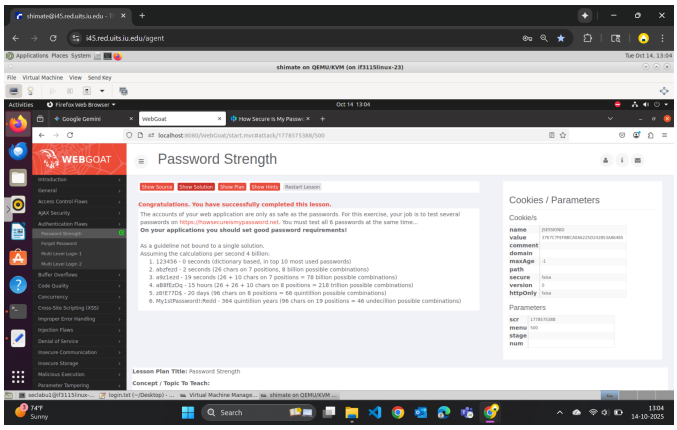

Fig. 1. Password Reset Exercise

In the exercise, six passwords were evaluated concurrently using the online estimator at https://www.security.org/how-secure-is-my-password and recorded the time-to-crack figures as shown in Fig. 1. Each password was input together per the exercise instructions to compare relative strength under the same model.

B. Forgot Password

The Forgot Password exercise was done using username webgoat and the secret-answer mechanic, the known answer red was submitted, and the application was observed returning the password reminder. Common color answers were tried for all the other usernames like admin or Larry until successful admin - green, demonstrating how weak secret questions and no lockout enable account compromise. WebWolf was used to inspect any outgoing password email behaviors where applicable.

C. Multiple Level Login 1

Stage 1: A login as Jane was performed using tarzan and the provided TANs. Stage 2: After a previously used TAN (15648) was obtained via simulated phishing, the TAN submission was intercepted with Burp, the hidden_tan value was modified to 1 in the intercepted request, and it was forwarded to the server to demonstrate bypass attempts when server-side TAN validation is insufficient.

D. Multiple Level Login 2

A login as Joe was performed with the password banana, the TAN submission was intercepted, and the hidden_user value was modified from Joe to Jane before the request was replayed. This showed how trusting client-side hidden fields allows an attacker who controls request contents to impersonate another user if the mapping of TANs to user accounts is not re-validated by the server.

E. Authentication bypass

WebWolf was used to send and retrieve password-reset emails, and email content and workflows were verified. Using a proxy on Burp, the reset flow was observed, parameters were renamed e.g., secquestion11=&secquestion22=, and the request was manipulated to illustrate how malformed parameters or insufficient server-side validation can be abused for bypass. Evidence of the WebWolf email and intercepted reset request is included in Fig 2.

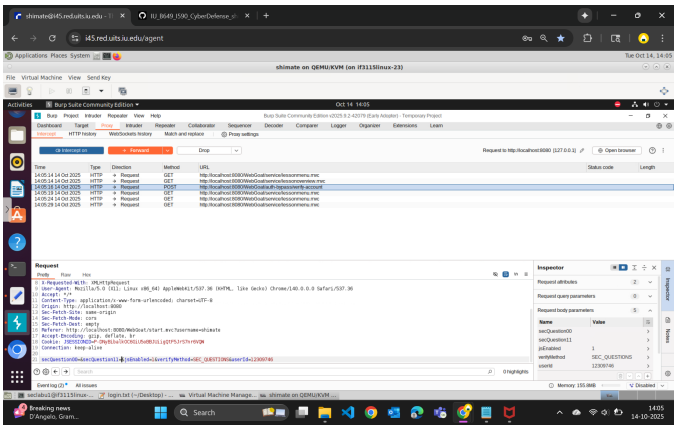

Fig. 2. Authentication Bypass Exercise

F. JWT Token Exercises

The JSON Web Token or JWT token exercises were executed using an intercepting proxy in Burf and WebWolf's decoding utilities.

Step 4 decoded the captured JWT to inspect its three parts: header, payload and signature. The header revealed the signing algorithm and the payload exposed the user claim, confirming the token's identity information. This decoding step as given in Fig 3 established the baseline needed to test subsequent tampering and verify whether the server enforced signature validation.

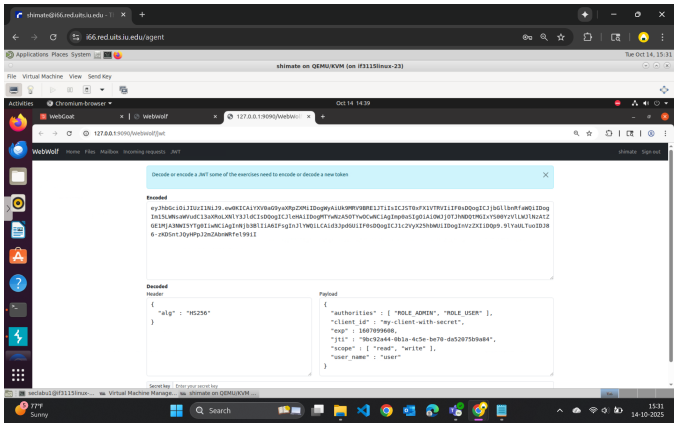

Fig. 3. JWT decoding step

Issued access tokens were captured and decoded to inspect the header, payload, and signature for step 6; the alg, user, and admin fields were noted. The header was changed to "alg":"none" and the payload was edited to set "admin":"true".

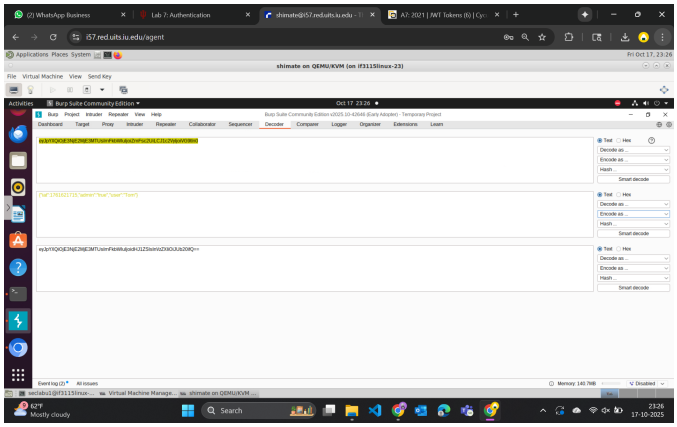

Fig. 4. JWT decoded in Burp - header + payload view

Each modified segment was re-encoded to Base64 and recombined into a three-part token; the signature segment was then removed. The modified token was injected into an intercepted request and replayed to the application via Burp Repeater.

The server-side signature verification was not properly enforced, the modified token was accepted and privileged actions were executed for example, vote reset or deletion operations completed successfully, demonstrating privilege escalation via token tampering.

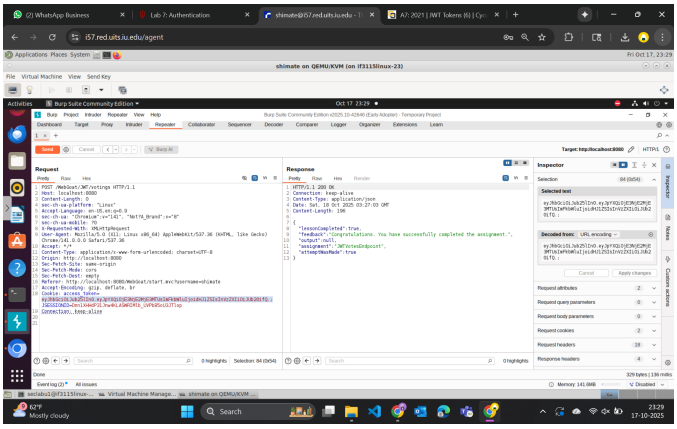

Fig. 5. Server response in Burp Repeater after changing header/payload

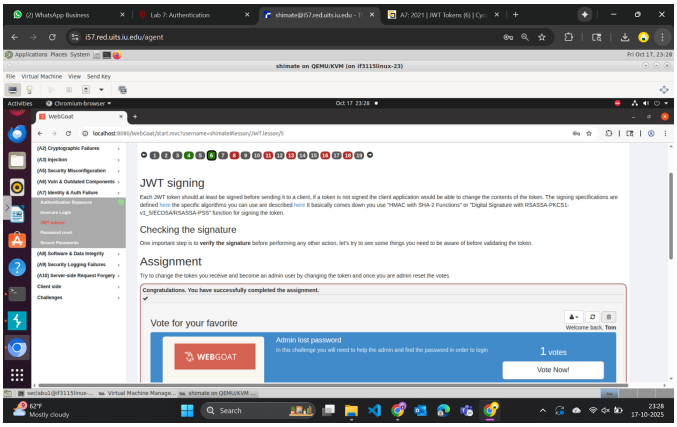

Fig. 6. Evidence of deleted/reset votes - server UI/response

The Questions in Step 8 are answered as follows:
1. What is the result of the first code snippet?
A. Solution 1: It throws an exception in line 12.
Explanation: parseClaimsJws(...) enforces signature verification; if the token is unsigned or tampered, signature validation fails and a JwtException is raised.

2. What is the result of the second code snippet?
A. Solution 2: Invoked the method removeAllUsers at line 7
Explanation: parse(...) (the non-JWS parser) can allow token parsing without enforcing signature verification in this context, so a manipulated token with "admin":"true" in the claims is accepted and the admin branch executes.

G. Password Reset

The WebGoat reset tasks were completed using WebWolf to receive reset emails, testing the reset page with known usernames and guessed security question answers, and enumerating answers for available accounts e.g., tom, admin, larry. Vulnerable accounts were tracked and which answered; admin - green, larry - yellow.

The next exercise examined how common password-recovery questions are often weak. Questions such as 'What is your favorite animal?' or 'What is your nickname?' can usually be answered from a target's social media, while prompts like 'Who was your childhood hero?' often have predictable answers e.g., 'Superman', making them poor choices for secure recovery.

III. Tools and Learning Experience

The exercises were executed against WebGoat with WebWolf for email inspection; Burp Suite served as the intercepting proxies for capture, modification, and replay of requests. These tools enabled direct experimentation with password-strength estimation, forgot-password flows, TAN-based multi-level login, JWT decoding/manipulation, and reset-email workflows, reinforcing practical lessons: JWTs are header.payload.signature and must have signatures verified; alg:none or removed signatures allow privilege escalation if not rejected; security-question answers must be treated like passwords; and client-side state like hidden fields, TANs must never be trusted.

IV. Findings and Discussion

The lab exposed concrete failure modes: weak secret answers and no lockout returned credentials e.g., webgoat + red, TAN flows could be abused by tampering with intercepted hidden fields or reused TANs, and JWTs altered header-alg:none, payload admin-true, signature removed granted admin actions where signature checks were absent. Source inspection showed parseClaimsJws(...) enforces signature validation and rejects tampered tokens, while looser parsers e.g., parse(...) can accept manipulated tokens in some implementations.

Although WebGoat/WebWolf are effective teaching tools, they omit several production defenses rate limiting, MFA, HSM-backed keys, WAF/SIEM. To improve both pedagogy and defensive value, the lab should include server-side mitigations and demonstrations: strict JWT validation rejecting alg:none, server-tracked single-use TANs with replay protection, hashed/salted storage for security-question answers, and exercises showing rate limiting, anomaly detection, and verification of fixes.

V. Conclusion

This lab reinforced that authentication remains a critical source of vulnerabilities: weak recovery questions, mutable client-side state, and incorrect token validation can all lead to account compromise or privilege escalation. Effective remediation requires a layered approach: strong password policies, secure reset flows that do not expose answers, server-side enforcement of one-time tokens, correct JWT validation with vetted libraries, rate limiting, and monitoring and alerting to detect suspicious activity.

References

[1] OWASP. WebGoat Project: Documentation and Exercises. Available at: https://github.com/WebGoat/WebGoat
[2] OWASP. WebWolf: Companion for Capturing Emails. Available at: https://github.com/WebGoat/WebWolf
[3] IETF. JSON Web Token (JWT) Specification RFC 7519. Available at: https://datatracker.ietf.org/doc/html/rfc7519
[4] Security.org. Password Strength Estimator. Available at: https://www.security.org/how-secure-is-my-password/
[5] GoodSecurityQuestions. Guidance on Selecting Security Questions. Available at: http://goodsecurityquestions.com/