# COMP – 357

# MITIGATION REPORT

Submitted by: Sreehari Jiji

Student id: 10316483

# Exercise 1 — OWASP Juice Shop

## Mitigation Recommendations

### Mitigation for Attack 1: DOM-Based XSS

**-Summary of Attack**

A DOM-based XSS condition was demonstrated through the search feature, where client-side handling of user input allowed injected script to execute in the browser context.

**-Root Cause**

Client-side input was processed and rendered without adequate sanitization and output encoding.

**-Recommended Controls**

- Apply output encoding for any user-supplied content rendered into the DOM.
- Enforce strict input validation on both client and server.
- Implement a strong Content Security Policy (CSP) to reduce script execution risk.
- Avoid unsafe DOM patterns.
- Use security headers consistently across the application.

**-Expected Outcome**

In a production-equivalent system with these controls, injected script content would be treated as text or rejected, preventing browser execution.

**Mitigation for Attack 2: Broken Access Control**

**-Summary of Attack**

An IDOR-style access control weakness was demonstrated by manipulating the client-exposed basket identifier (bid) to change the basket context beyond intended authorization.

**-Root Cause**

Insufficient server-side authorization enforcement on object access tied to authenticated users.

**-Recommended Controls**

- Enforce server-side ownership checks on every request involving user-scoped objects (basket, orders, profile).
- Bind resources to the authenticated identity and validate ownership before read/write operations.
- Use non-guessable identifiers instead of sequential numeric IDs.
- Implement access control tests as part of CI/CD.
- Add logging and alerting for abnormal access patterns.

**-Expected Outcome**

Changing client-visible identifiers would not grant access to other users' resources, and unauthorized access attempts would be denied and logged.

**Mitigation for Attack 3: Broken Authentication**

**-Summary of Attack**

A login bypass was demonstrated using injection-style input in the email field, resulting in unauthorized administrative-level access in the training context.

**-Root Cause**

Authentication logic tolerated crafted input, indicating insufficient validation or unsafe query construction.

**-Recommended Controls**

- Use parameterized queries / prepared statements for all authentication-related database operations.
- Apply strict server-side input validation for identity fields.
- Implement rate limiting and account lockout policies for repeated failed attempts.
- Require multi-factor authentication (MFA) for privileged roles in production.
- Centralize authentication logs and monitor for anomalous login patterns.

**-Expected Outcome**

Injection-style input would fail authentication, and privileged access would require legitimate credentials and additional verification.

# Exercise 2 — Kerberoasting

## Mitigation Recommendations

### Mitigation for Attack: Kerberoasting – Service Account Credential Recovery

#### Summary of Attack

A Kerberoasting demonstration successfully recovered the lab service account password for LAB\svcweb by requesting an SPN-associated service ticket for HTTP/webapp.lab.local and performing controlled offline password cracking. This confirms that SPN-linked service accounts become high-risk targets when weak or long-lived credentials are used.

#### Mitigation Options

Kerberoasting risk can be reduced through layered identity, configuration, and monitoring controls. In production environments, recommended mitigations include:

1. Strong, long, and regularly rotated service account passwords
   Use high-entropy passwords (preferably random, long length) to make offline cracking impractical.
2. Use Group Managed Service Accounts (gMSA) where possible
   gMSAs provide automatic password management and reduce the risk of weak or static service account credentials.
3. Least privilege for service accounts
   Ensure service accounts do not hold unnecessary permissions or membership in privileged groups.
4. Limit service account sprawl and enforce governance
   Maintain an accurate inventory of SPNs and service accounts, remove unused SPNs, and enforce ownership and review cycles.
5. Monitor Kerberos and service ticket activity
   Detect abnormal spikes in TGS requests or unusual patterns from non-administrative users.

6. Separate high-risk services and restrict lateral pathways
   Segment sensitive resources and reduce the blast radius of any single credential compromise.

**Mitigation Demonstrated in This Lab**

While multiple mitigations are applicable, this exercise demonstrated one practical control for validation:
resetting the service account password to a strong, complex value.

The SPN configuration (HTTP/webapp.lab.local) was intentionally left unchanged so the validation specifically measured the impact of credential strength on Kerberoasting feasibility.

**Validation Method**

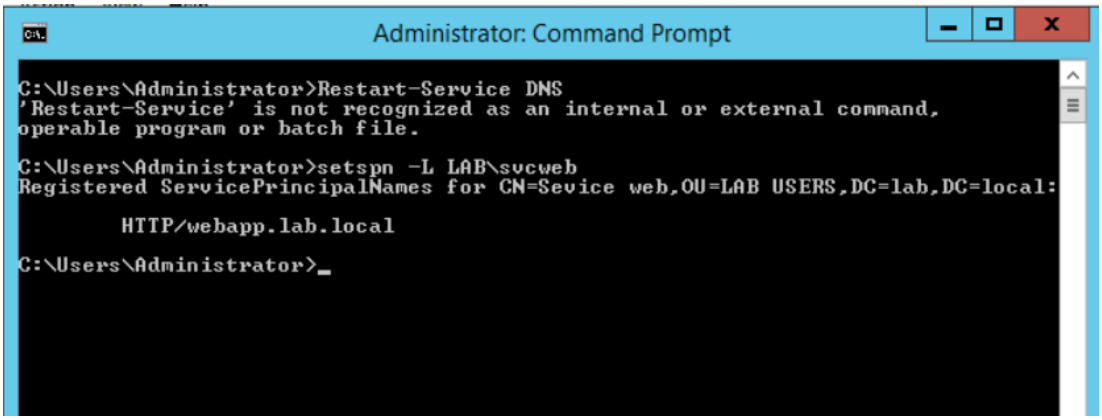
After the new service account password was applied, a fresh service ticket for the SPN-associated account was requested and extracted. The updated ticket hash was tested using the same offline cracking approach used in the baseline demonstration to confirm whether the attack remained practical.

**Validation Result**

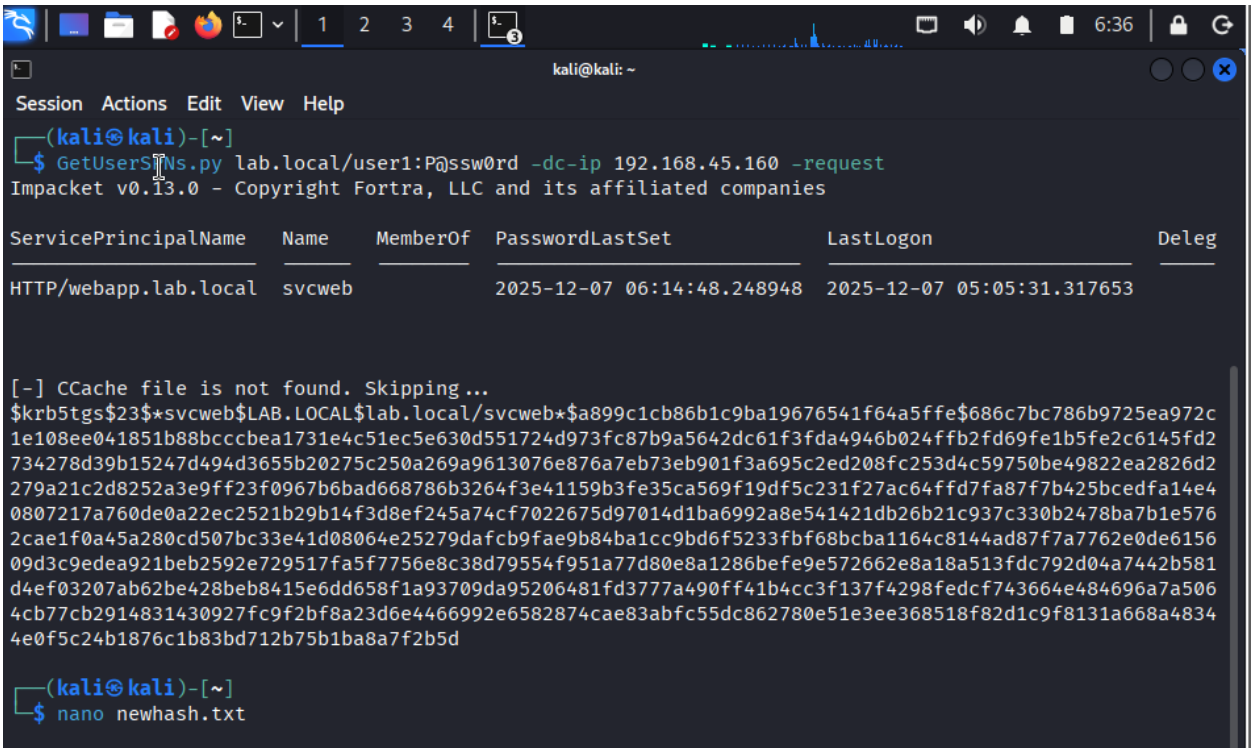
After the new password was applied, John the Ripper did not recover the updated service account password within the lab test scope. This demonstrates that improving service account password strength significantly reduces Kerberoasting effectiveness in real-world equivalents.

**Evidence**

- SPN verification after new password — output confirming HTTP/webapp.lab.local remains registered to LAB\svcweb.




- Updated service ticket hash after new password — extraction output showing a new TGS hash for the hardened account.



- John the Ripper output after new password — confirms the updated password was not recovered during the validation attempt.