Fakemon: a vulnerable web application - Full Laboratory Guide

This guide describes an intentionally vulnerable web application and lab exercises for educational penetration testing and secure development practice which aims to contribute to practical learning for future students of the Exfil Practicum Training Program.

Contents

Disclaimer

This application intentionally contains multiple security vulnerabilities for educational purposes (SQL Injection, Cross-Site Scripting, Broken Access Control). Run the lab only in isolated environments such as local VMs, dedicated lab networks, or properly configured containers that are not accessible from the public Internet.

This application is intentionally vulnerable for educational purposes only. Never attempt these techniques on systems you don't own or have explicit permission to test. The authors and distributors of this lab are not responsible for misuse.

Overview

 Fakemon is a small, intentionally vulnerable Flask web application designed for teaching and practicing common web vulnerabilities and exploitation techniques in a controlled lab. The application uses SQLite for storage and serves static images from a local folder (e.g., {static/images/).

Learning objectives

- Recognize, exploit, and mitigate common web vulnerabilities: SQL Injection (SQLi), Cross-Site Scripting (XSS), Broken Authentication and Access Control, and others.
- Practice safe and reproducible exploitation in a local lab environment.
- Chain multiple vulnerabilities to demonstrate real-world impact.

Lab: Setup & Prereqs

System requirements

• A Linux/macOS/Windows development machine.

• Python 3.10+ and pip.

• (Optional) Docker and Docker Compose for containerized lab.

• Browser (Chrome / Firefox) and an intercepting proxy (Burp Suite Community)

Local Installation

1. Clone or copy the lab project into a working directory from *github repo*.
2. Create virtual environment and install all requirement:

```
python -m venv venv
source venv/bin/activate # or .\venv\Scripts\Activate.ps1 on Windows
pip install -r requirements.txt
```

3. Initialize the database: python init_db.py
4. Start the app: python app.py
5. Open: http://127.0.0.1:5000

Note: To reset DB run: python init_db.py

Docker

List of all vulnerabilities

- SQL Injection (SQLi)
- Deleting users (via admin delete or SQLi)
- Extract admin credentials (SQLi)
- Extract user IDs and roles (SQLi)
- Get Table Names (SQLi)
- Stored Cross-Site Scripting (XSS)
- Reflected Cross-Site Scripting (XSS)
- Brute-forcing login (Auth)
- Verbose Login Errors (Auth)
- Predictable credentials (Auth)
- Weak URLs / token-in-URL bypass / Information Disclosure (Access control)
- Insecure Cookie Attributes (Access control)
- Insecure Direct Object Reference (Access control)
- Vertical privilege escalation (Access control)

1. Lab: Authentication vulnerabilities

1.1 Predictable credentials

Goal: Identify and use default or weak credentials to get access.

1. Go to Login page.
2. Try to Identify valid usernames using verbose error messages.
3. Identify ‘test’ and ‘admin’.
4. Try manually common passwords to get in, or open Burp Suite to brute force the password:
    - Go to Proxy.
    - Open browser, open Fakemon.
    - Try logging in with admin as username.

- In Proxy > HTTP history and search for the POST / LOGIN Request, on username write down one of the found users.
- Right click and send it to Intruder.
- Select the password tested in the request and click the Add button to mark it as payload, it should appear like this `§testpassword§`
- On the Payload panel, choose Simple List and add the passwords you would like to try. Try searching for the top 10 most common passwords. You can also load a txt file.
- Make sure the selected attack option is Sniper Attack.
- Press Start Attack and look for a 302 response; this indicates a successful login.
- Log in using the username and password that you identified and access the user account.

1.2 Brute-forcing login

Goal: Use an automated brute force attack to get into the admin panel.

1. Open Fakemon and try going to the admin page. Add /Admin to the URL, notice you are denied access.
2. Go to the login page and try login in with username = admin. Notice user does exist.
3. On Burp Suite go to Proxy > HTTP history and search for the POST / LOGIN Request.
4. Select the password tested in the request and click the Add button to mark it as payload, it should appear like this `§testpassword§`
5. On the Payload panel, choose Simple List and add the passwords you would like to try.
6. Make sure the selected attack option is Sniper Attack.
7. Press Start Attack and look for a 302 response; this indicates a successful login.
8. Log in using the username and password that you identified and access the admin panel.

Note: You can also try other types of attacks, but it might be slower.

2. Lab: SQL Injection

2.1 Confirm SQL Injection

Goal: confirm `/search?q=` is injectable.

1. Go to the search page.
2. Try searching for something, notice the `/search?q=picu` structure
3. Modify the q parameter, giving it the value `' OR '1'='1` Use the search bar or intercept the request in Burp Suite.

4. Send the request. Notice that you get all cards back, thus confirming is injectable.

Note: Check out the SQLi cheat sheet [SQL injection cheat sheet | Web Security Academy]

2.2 Extract user IDs and roles

Goal: confirm `/search?q=` is injectable.

1. Go to the search page.
2. Try searching for something, notice the `/search?q=picu` structure
3. Modify the q parameter, giving it the value `' UNION SELECT id, username, role, credits, NULL, NULL FROM users --` Use the search bar or intercept the request in Burp Suite.
4. Send the request. You will get placeholders for cards with the names corresponding to the users and the type to the role they have.

2.3 Extract admin credentials and Delete users

Goal: Steal the admin credentials to access the admin panel and delete accounts.

1. Go to the search page.
2. Try searching for something, notice the `/search?q=picu` structure
3. Modify the q parameter, giving it the value `' UNION SELECT username, password, role, credits, NULL, NULL FROM users WHERE username='admin' --` Use the search bar or intercept the request in Burp Suite.
4. Send the request. You will get placeholders for a card corresponding to the admins password.
5. Use this credential to log in.
6. Access the admin panel and delete any account or card you want.

2.4 Get Table Names

Goal: Get the names for all the tables from the DB

1. Go to the search page.
2. Try searching for something, notice the `/search?q=picu` structure
3. Modify the q parameter, giving it the value `' UNION SELECT 1, name, type, 1, NULL, NULL FROM sqlite_master WHERE type='table' LIMIT 1 OFFSET 0 --` Use the search bar or intercept the request in Burp Suite.

4. Send the request, change the OFFSET to 1, 2, 3, etc. to get different tables names in the response. Try this until you get cards instead.

Note: Theres a more direct way of doing this, try to come up with it!

3. Lab: XSS

3.1 Stored XSS

Goal: Demonstrate a persistent XSS by storing a script in a comment that executes when other users view comments.

1. Navigate to any card page (e.g., /cards/1).
2. Scroll to comments section.
3. In comment form, enter payload in text field: `<script>alert('This is XSS')</script>`
4. Submit comment.
5. Reload page and observe alert executes.
6. Try opening from another browser or a different account and check if it still shows up.

3.2 Reflected XSS

Goal: Demonstrate a reflected XSS by supplying a payload in a URL that is reflected back and executed.

1. Navigate to `/search` endpoint.
2. In search box, enter payload: `<script>alert('This is reflected XSS')</script>`
3. Submit search.
4. Observe alert executes immediately.

4. Lab: Access Control

4.1 Insecure Direct Object Reference

Goal: Check for IDOR, look for the exclusive legendary card.

1. Navigate to home page and observe cards listed.
2. Click on different cards and notice the endpoint are `/cards/<card_id>`
3. Directly access `/cards/1` in browser. Notice that you can directly access cards this way.
4. Try accessing other card IDs until you find the Legendary hidden Card.

4.2 Token-in-URL bypass Using Leaked Secrets in Documentation

Goal: Access the admin panel using a query admin token, find the token in the Fakemon code on Github.

1. Navigate to /admin endpoint (should show "Access denied").
    2. Add query parameter: `/admin?admin_token=token`
    3. Access admin panel successfully.
    4. To access admin actions go to lab 4.3.

4.3 Vertical Privilege Scalation via Cookie Tampering

Goal: Modify cookies to gain unauthorized privileges.

1. Make an account and navigate to /admin endpoint (should show "Access denied").
    2. Open your browsers DevTools.
    3. Go to Application>Storage>Cookies
    4. Notice there is an is_admin cookie set to 0. Change it to 1.
    5. Reload the page.
    6. Access admin panel successfully and its actions.

Note: Try doing this in Burp Suite too.

4.4 Weak URLS

Goal: Steal all users information using only exposed endpoints.

1. Check for IDOR.
2. After confirming it try the exposed endpoint `/_dump/users`
3. Select pretty print, you can now see all users' sensible info.

References & further reading

Exfil Webpage

Fakemon Github repository

OWASP Top Ten

Web Security Academy (PortSwigger)

Got any questions? Contact me: nancy.burgos@exfilsecurity.com