HackTrek: Vulnerable Web-App Guide

This guide addresses specific vulnerabilities in a controlled environment for educational purposes.

Challenges and Solutions

1. NoSQL Injection

Goal: Exploit NoSQL injection vulnerabilities to retrieve or manipulate database records.

Solution:

1. Inject the following payloads to bypass filters or retrieve specific records:

    - Retrieve All Users:

```
.*
```

    - Retrieve Admin Users:

```
^admin.*
```

2. Observe the server's response to confirm the data leak.

What to Do If Errors Occur:

- **Error Observation:** Look for syntax errors or empty responses.
- **Solution:** Validate the endpoint functionality by providing expected inputs and retry the payloads. Suggest using input sanitization and query parameter validation to prevent injections.

---

2. Login Admin

Goal: Crack the admin login credentials.

Solution:

1. Capture the login request using Burp Suite.
2. Configure Intruder for brute-forcing:

    - Set the payload position on the `password` field.
    - Use a password list containing common passwords (e.g., `admin`, `password123`).

3. Run the attack and monitor responses for successful authentication.

What to Do If Errors Occur:

- **Error Observation:** If the server blocks attempts or throttles requests, reduce attack speed or add delays.
- **Solution:** Suggest implementing rate limiting, account lockout mechanisms, and password strength policies.

---

3. Login TestUser

Goal: Crack the regular user login credentials.

Solution:

1. Use Burp Suite's Intruder for brute-forcing the `password` field with a dictionary of weak passwords.
2. Capture successful responses to identify valid credentials.
3. Another way is to check the NoSQL Injection challenge result to find Test User details.

What to Do If Errors Occur:

- **Error Observation:** Authentication errors or blocked requests.
- **Solution:** Recommend two-factor authentication and robust password requirements to mitigate risks.

---

4 & 5. Privacy Policy and Confidential Document

Goal: Access restricted documents or pages.

Solution:

1. Access Privacy Policy:
    - Navigate to `/privacy-policy`.
    - Acknowledge the privacy policy.
2. Download Confidential Document:
    - Access `/confidential-document` and attempt to modify headers or cookies to simulate authorized access.
    - Download the document for validation.

What to Do If Errors Occur:

- **Error Observation:** If access is denied, verify headers, tokens, or session cookies.
- **Solution:** Recommend encrypting documents and validating access tokens for all requests.

---

6. Bonus Payload (XSS)

Goal: Inject a payload into the URL to exploit reflected XSS.

Solution:

1. Inject sample payloads into URL parameters:
    - Payload for Audio:

```
?payload=<iframe width="100%" height="60" scrolling="no" frameborder="no"
allow="autoplay" src="https://www.soundhelix.com/examples/mp3/SoundHelix-Song-
1.mp3"></iframe>
```

- Payload for Video:

```
?payload=<iframe width="560" height="315"
src="https://www.youtube.com/embed/dQw4w9WgXcQ" frameborder="0" allow="autoplay;
encrypted-media" allowfullscreen></iframe>
```

2. Observe browser behavior to confirm payload execution.

What to Do If Errors Occur:

- **Error Observation:** Payloads may not render due to input filtering or improper encoding.
- **Solution:** Suggest proper input encoding and content security policies (CSP) to prevent XSS attacks.

---

7. File Upload

Goal: Test for unrestricted file upload vulnerabilities.

Solution:

1. Use Burp Suite to intercept file upload requests.
2. Modify the file extension to a potentially malicious type (e.g., `.php`, `.html`) and attempt the upload.
3. Verify if the file can be executed on the server.

What to Do If Errors Occur:

- **Error Observation:** Rejected uploads or improper validation.
- **Solution:** Suggest using file type whitelisting, MIME-type validation, and restricting execution of uploaded files.

---

8. File Upload with Size Limit

Goal: Bypass file size restrictions.

Solution:

1. Intercept the upload request using Burp Suite.
2. Modify the `Content-Length` header to bypass size limits.
3. Alternatively, split the file into smaller chunks if chunked uploads are supported.

What to Do If Errors Occur:

- **Error Observation:** Server may return `413 Request Entity Too Large` or truncation errors.
- **Solution:** Enforce strict size validation on both client and server sides.

---

9. Reflected XSS

Goal: Exploit reflected XSS vulnerabilities in user input fields or URL parameters.

Solution:

1. Inject JavaScript payloads into query parameters:

   ```
   <script>alert('Reflected XSS');</script>
   ```

2. Use Burp Suite to test payload variations dynamically.
3. Observe the browser for script execution.

What to Do If Errors Occur:

- **Error Observation:** If scripts don't execute, check for encoding or sanitization.
- **Solution:** Implement comprehensive input validation and output encoding to prevent reflected XSS.

---

10. Unvalidated Redirect

Goal: Exploit open redirect vulnerabilities.

Solution:

1. Identify endpoints that accept URL parameters (e.g., `/redirect?target=`).
2. Inject malicious URLs:

   ```
   /redirect?target=https://oldcrypto.com/address1
   ```

3. Confirm if redirection occurs to the injected URL.

What to Do If Errors Occur:

- **Error Observation:** If redirection fails, verify URL encoding and parameter usage.
- **Solution:** Restrict URL parameters to a whitelist of trusted domains.

---

11. Weak Password Validation

Goal: Test and identify weak password mechanisms.

Solution: Suggest using strong password policies, password managers, and multi-factor authentication to mitigate vulnerabilities.

---

12. Bully Chatbot

Goal: Exploit the chatbot's functionality to retrieve coupon codes through automated spamming.

Solution:

1. Identify chatbot input fields or endpoints that accept messages.

2. Use automation tools (e.g., Selenium or Python scripts) to repeatedly send messages with the text "coupon"
3. Observe responses to identify and extract valid coupon codes.

What to Do If Errors Occur:

- **Error Observation:** The chatbot may block repeated messages or implement anti-spam measures.
- **Solution:** Recommend adding rate limiting, CAPTCHA challenges, and monitoring tools to prevent abuse.

This guide provides practical steps for testing vulnerabilities while adhering to ethical and educational practices.
