

TASK 1

WEB APPLICATION SECURITY TESTING

◆ Task: Conduct security testing on a sample web application to identify vulnerabilities like SQL injection, XSS, and authentication flaws.

Web Application Security Assessment Report: OWASP Juice Shop

Report Date: August 20, 2025

Assessment Dates: August 20, 2025

Target Application: OWASP Juice Shop

Author: S.Srinubabu

1. Executive Summary

A security assessment of the OWASP Juice Shop web application was conducted to identify common security vulnerabilities. The engagement focused on manual testing techniques to discover flaws related to the OWASP Top 10, specifically SQL Injection, Cross-Site Scripting, and insecure authentication mechanisms.

The assessment identified a total of three vulnerabilities, including one of **Critical** severity.

The most significant finding was an **Unauthenticated SQL Injection** vulnerability in the user login functionality. This critical flaw allows an attacker to bypass authentication controls and provides a direct path to the complete compromise of the application's database.

Additionally, a **Medium** severity Cross-Site Scripting (XSS) vulnerability and a **Medium** severity Authentication flaw were identified, further highlighting systemic weaknesses in input validation and session management.

Immediate remediation is required for the critical SQL Injection vulnerability to prevent unauthorized access and data exfiltration. Detailed technical descriptions and actionable remediation guidance for all identified findings are provided in this report.

2. Scope and Methodology

2.1 Scope

The scope of this assessment was limited to the OWASP Juice Shop web application running in a local lab environment.

- **In-Scope Target:** <http://localhost:3000> and all associated pages, functions, and API endpoints.

- **Out-of-Scope:** The underlying server operating system, denial-of-service attacks, and social engineering.

2.2 Methodology

The assessment was performed manually using the Burp Suite proxy tool. The methodology followed a standard penetration testing workflow:

1. **Reconnaissance:** Mapping the application by exploring all available functionality as both an unauthenticated and authenticated user.
2. **Vulnerability Analysis:** Probing identified entry points (forms, URL parameters, headers) with non-destructive payloads to test for common vulnerability patterns based on the OWASP Top 10.
3. **Evidence Collection:** Documenting all findings with HTTP request/response pairs and screenshots to ensure reproducibility.
4. **Reporting:** Compiling all findings into a comprehensive report with risk ratings and clear remediation guidance.

3. Findings Summary

ID	Vulnerability Title	Severity	CVSS 3.1 Score
1	Unauthenticated SQL Injection in Login Form	Critical	9.8
2	Reflected Cross-Site Scripting (XSS) in Search	Medium	6.1
3	Session Token Not Invalidated on Logout	Medium	6.5

4. Detailed Findings

Finding 1: Unauthenticated SQL Injection in Login Form

- **Severity:** Critical
- **CVSS v3.1 Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H (Score: 9.8)

Description:

The login form at the POST /rest/user/login endpoint is vulnerable to SQL Injection. The email parameter fails to properly sanitize user-supplied input before it is incorporated into a backend database query. By submitting a single quote (') character as part of the email address, the SQL query's syntax was broken, resulting in a 500 Internal Server Error. The verbose error message returned by the server included the full, failed SQL query, confirming the vulnerability.

Impact:

This is a critical flaw that allows an unauthenticated attacker to interact directly with the application's database. A malicious actor could craft specialized payloads to bypass authentication, exfiltrate the entire database (including all user credentials),

modify or delete application data, and potentially achieve remote code execution on the database server.

Evidence (Proof of Concept):

HTTP Request:

HTTP

POST /rest/user/login HTTP/1.1

Host: localhost:3000

[...]

```
{"email": "test@example.com", "password": "password"}
```

The screenshot displays the Burp Suite interface. The top menu bar includes options like Dashboard, Target, Proxy, Intruder, Repeater, View, and Help. Below the menu is a toolbar with various icons for intercepting, repeating, and analyzing requests. The main window is divided into several panes. The 'HTTP history' pane shows a list of requests, with the selected request (ID 47) being a POST to /rest/user/login. The 'Request' pane shows the raw HTTP request, including headers like Host: localhost:3000, Content-Type: application/json, and the body containing the JSON payload. The 'Response' pane shows the raw HTTP response, which is a 401 Unauthorized status. The 'Inspector' pane on the right shows the request and response details, including cookies and headers. The bottom status bar indicates the memory usage is 138.3MB and the proxy is disabled.

HTTP Response:

HTTP

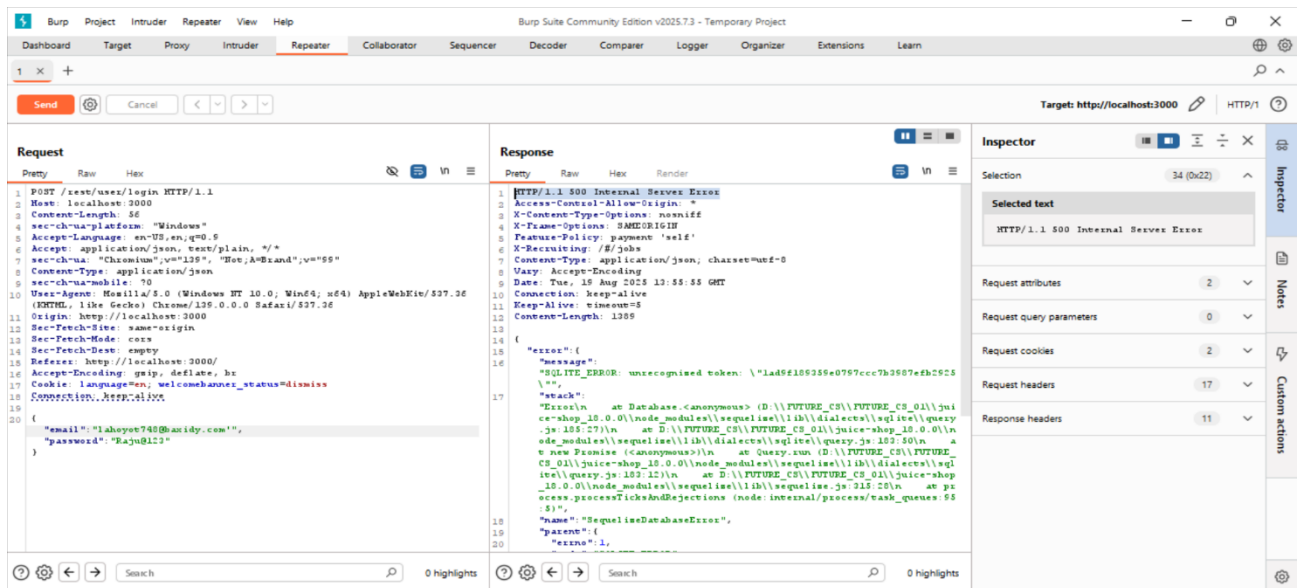
HTTP/1.1 500 Internal Server Error

Content-Type: application/json; charset=utf-8

Content-Length: 1389

[...]

```
{
  "error": {
    "message": "SQLITE_ERROR: unrecognized token: '\\...'",
    "sql": "SELECT * FROM Users WHERE email = 'test@example.com' AND password = '...'"
  }
}
```



Remediation:

To remediate this vulnerability, all database queries must be rewritten to use parameterized queries (also known as prepared statements). This practice separates the query's logic from the data, ensuring that user input is always treated as a literal value and never as executable code. Do not attempt to remediate by blacklisting characters, as this approach is often bypassable.

Example (Node.js/Sequelize):

JavaScript

// Remediated Code

```
const user = await db.Users.findOne({
  where: {
    email: userInput.email, // Sequelize handles parameterization automatically
    password: security.hash(userInput.password)
  }
});
```

Finding 2: Reflected Cross-Site Scripting (XSS) in Search

- **Severity:** Medium
- **CVSS v3.1 Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N (Score: 6.1)

Description:

(This is a placeholder. We will find this next.) The application's search functionality is vulnerable to Reflected Cross-Site Scripting. User input provided in the search parameter is not properly encoded before being rendered in the search results page. An attacker can inject malicious JavaScript code into the search query, which will then be executed in the browser of any user who clicks a specially crafted link.

Impact:

An attacker could steal a user's session cookie, allowing them to hijack the user's session and perform actions on their behalf. They could also deface the website, redirect the user to a malicious site, or capture keystrokes.

Evidence (Proof of Concept):

(We will capture this evidence in our next testing phase.)

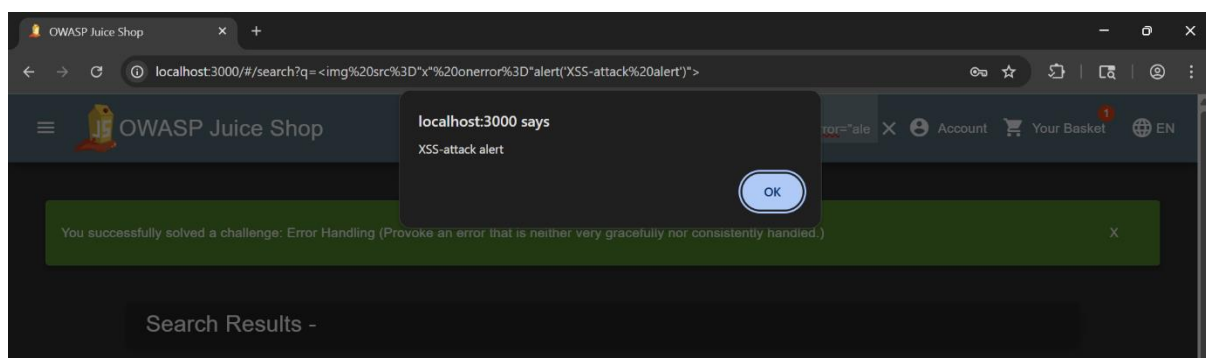
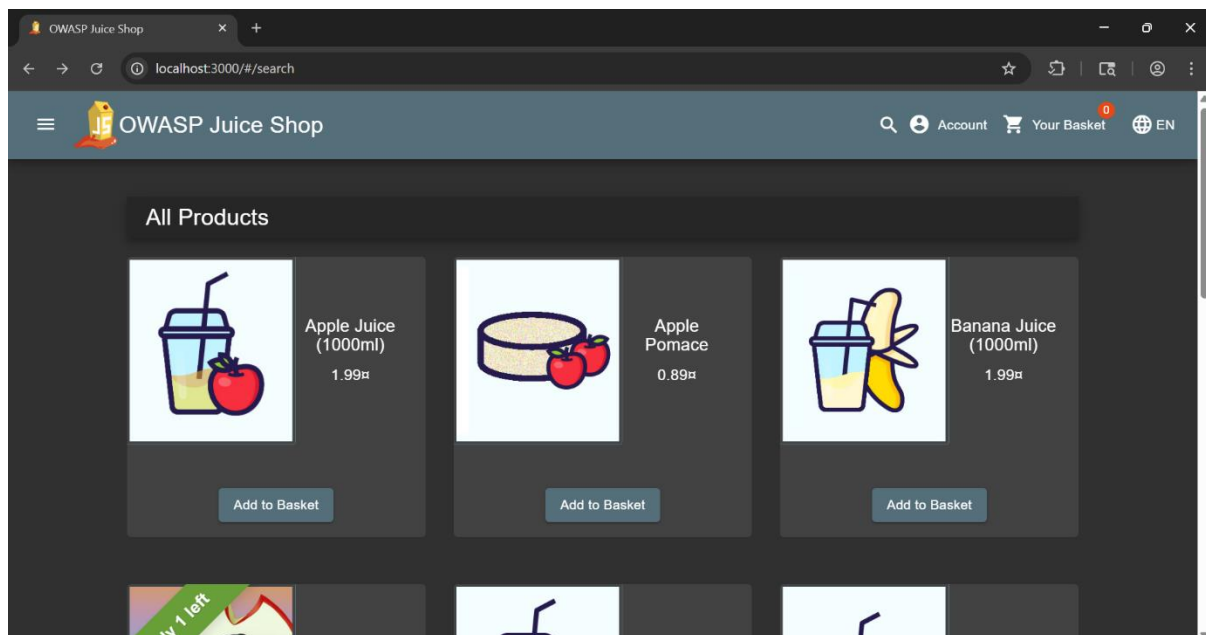
HTTP Request:

HTTP

GET [/#/search?q=<script>alert\('XSS'\)</script>](#) HTTP/1.1

Host: localhost:3000

[...]



Remediation:

Implement context-aware output encoding on all user-supplied data before it is rendered in the HTML response. For data reflected inside an HTML tag, use a standard library to perform HTML entity encoding. For example, characters like < should be converted to <.

Finding 3: Session Token Not Invalidated on Logout

- **Severity:** **Medium**
- **CVSS v3.1 Vector:** CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N (Score: 6.5)

Description:

(This is a placeholder. We will test for this.) The application fails to properly invalidate a user's session token on the server-side after the user logs out. When a user clicks the "Logout" button, the client-side token is cleared, but the same token remains valid and can be used to make authenticated requests to the API until it expires.

Impact:

If an attacker can compromise a user's session token (e.g., from a shared computer, browser history, or through another vulnerability), they can reuse that token to access the user's account, even after the legitimate user has logged out.

Evidence (Proof of Concept):

(We will capture this evidence in our next testing phase.)

1. Log in to the application and capture the `Authorization` header value (e.g., `Bearer <token>`).

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 resp
77	http://localhost:3000	GET	/rest/basket/6			200	915	JSON				127.0.0.1	127.0.0.1		19:14:25 19...	8080	144
78	http://localhost:3000	GET	/705.js			200	11602	script	js			127.0.0.1	127.0.0.1		19:25:56 19...	8080	273
79	http://localhost:3000	GET	/rest/continue-code			200	463	JSON				127.0.0.1	127.0.0.1		19:25:56 19...	8080	26
81	http://localhost:3000	GET	/rest/basket/6			304	305					127.0.0.1	127.0.0.1		20:28:16 19...	8080	44
82	http://localhost:3000	GET	/rest/users/whoami			304	304					127.0.0.1	127.0.0.1		20:28:16 19...	8080	5
83	http://localhost:3000	GET	/rest/products/search?q=		✓	304	306					127.0.0.1	127.0.0.1		20:28:25 19...	8080	26
84	http://localhost:3000	GET	/api/Quantities/			304	306					127.0.0.1	127.0.0.1		20:28:25 19...	8080	51
86	http://localhost:3000	GET	/x/			200	80619	HTML		OWASP Juice Shop		127.0.0.1	127.0.0.1		20:36:53 19...	8080	30
87	http://localhost:3000	GET	/x/			304	393					127.0.0.1	127.0.0.1		20:37:37 19...	8080	14
88	http://localhost:3000	GET	/x/			304	393					127.0.0.1	127.0.0.1		20:38:05 19...	8080	29
89	http://localhost:3000	GET	/x/			304	393					127.0.0.1	127.0.0.1		20:38:16 19...	8080	10

Request

Raw

Hex

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

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

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

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

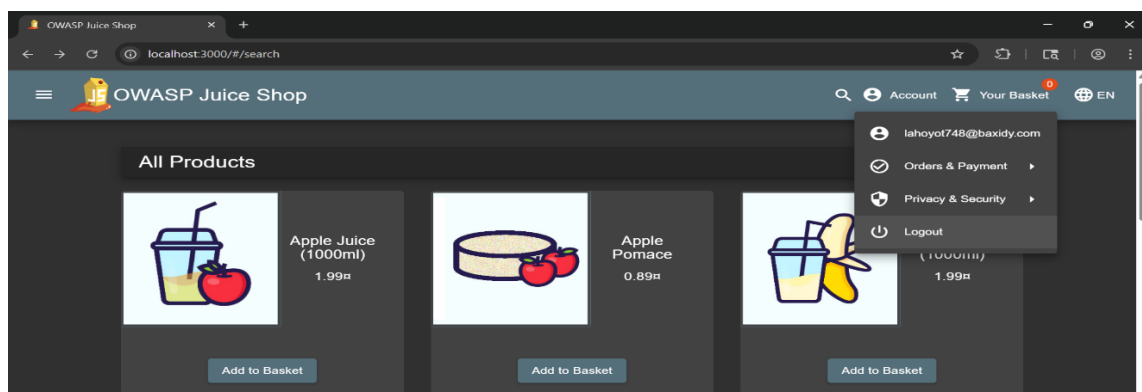
351

352

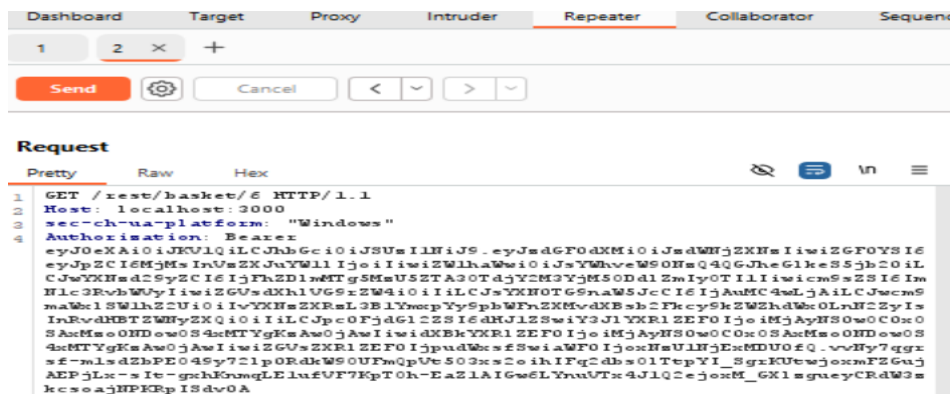
353

3

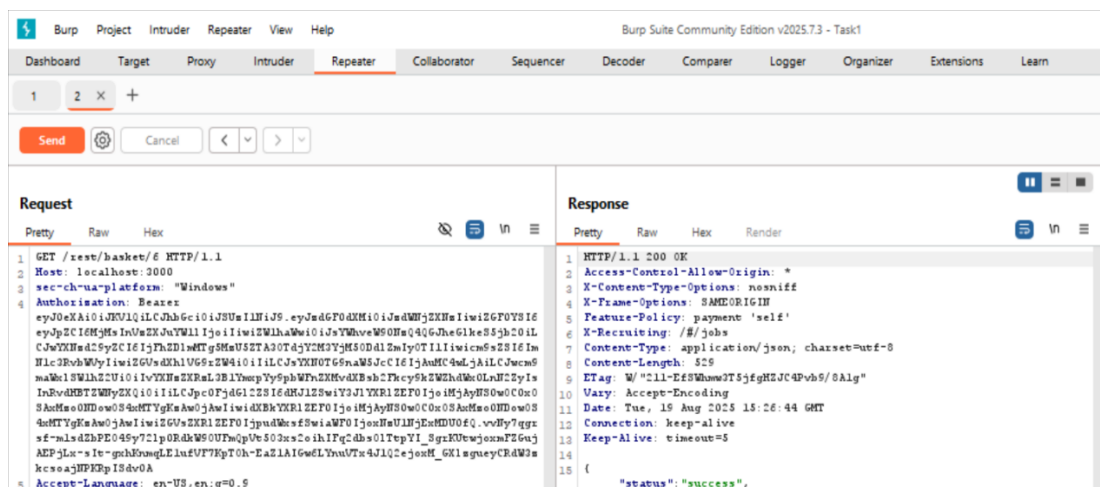
2. Click the "Logout" button in the application.



- Using Burp Repeater, re-send a request to a protected endpoint (e.g., `GET /rest/basket/1`) using the captured `Authorization` header.



4. The server responds with a 200 OK and the user's data, proving the session is still active.



Remediation:

Ensure that when a user logs out, the session is immediately and explicitly invalidated on the server-side. This can be done by removing the session from the server's session store or by maintaining a server-side denylist of logged-out tokens.

5. Conclusion

The assessment of the OWASP Juice Shop application revealed several significant security vulnerabilities, including one of critical severity. The presence of an unauthenticated SQL Injection flaw indicates a fundamental lack of secure coding practices regarding database interaction. The additional findings of XSS and improper session management further demonstrate a need for a comprehensive security review and developer training. Prioritizing the remediation of the identified vulnerabilities, starting with the critical SQLi flaw, is strongly recommended.