

Aplicações Web Vulneráveis

CST em Análise e Desenvolvimento de Sistemas

Luiza Kuze

3 de março de 2025



INSTITUTO FEDERAL
Santa Catarina
Campus São José

Licenciamento



Slides licenciados sob [Creative Commons "Atribuição 4.0 Internacional"](https://creativecommons.org/licenses/by/4.0/)

Agenda

- 1 Introdução
- 2 OWASP Mutillidae II
- 3 OWASP Juice Shop
- 4 Demonstração
- 5 Síntese

Introdução

OWASP

- 1 Comunidade
- 2 Ferramentas, documentação e bibliotecas de segurança

Alguns Projetos Interessantes

- Top 10
- Juice Shop e Mutillidae
- Cheat Sheet Series
- Dependency Check

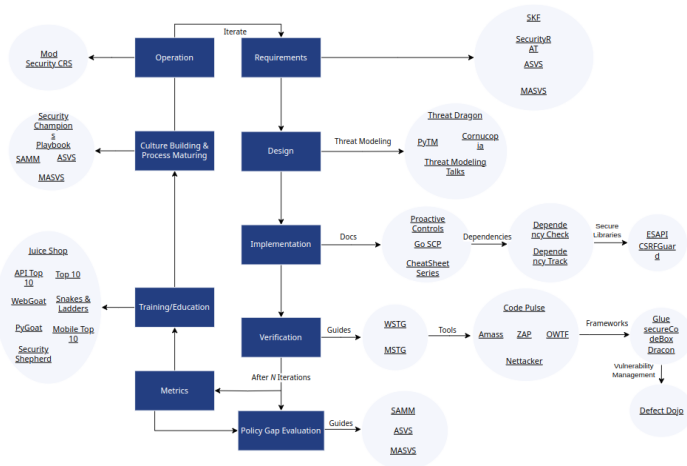


OWASP

Open Web Application
Security Project

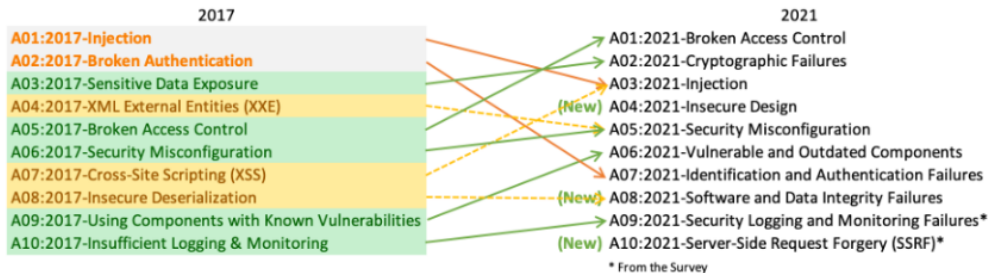
Fonte: owasp.org

Projetos da OWASP



Fonte: owasp.org/www-project-integration-standards/

OWASP Top 10



Fonte: owasp.org/www-project-top-ten/



Vulnerabilidade é uma falha específica que pode ser explorada. Uma **Ameaça** é o potencial para um agente explorar a vulnerabilidades. **Risco** é a potencial perda quando a ameaça acontece.

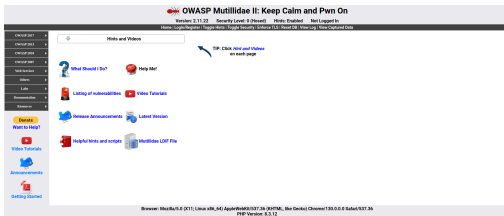
Fonte: informationsecurity.wustl.edu

Motivação para Aplicações Web Vulneráveis

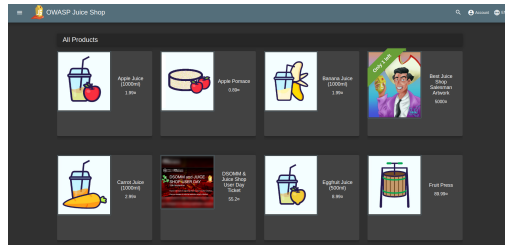
- 1 Treinamento Prático de Segurança
- 2 Testes de Ferramentas de Segurança

Por que duas aplicações vulneráveis?

OWASP Mutillidae II e OWASP Juice Shop



Mutillidae II



Juice Shop

OWASP Mutillidae II

Visão Geral

- PHP/MySQL
- Modo Seguro e Inseguro
- Reset do Ambiente
- Dicas e Vídeos do Youtube
- OWASP Top 10
- Laboratórios



Fonte: owasp.org/www-project-mutillidae-ii/

Como Utilizar?

Docker

```
1 # Clonar o repositório
2 git clone https://github.com/webpwnized/mutillidae-
  docker.git
3 cd mutillidae-docker
4
5 # Composição com o Docker
6 docker compose -f .build/docker-compose.yml up --build
  --detach
7
```



Fonte: docker.com

Repositório: github.com/webpwnized/mutillidae

Vulnerabilidades OWASP Top 10

Exemplo: Cross-Site Scripting (XSS)

OWASP Mutillidae II: Keep Calm and Pwn On

Version: 2.11.23 Security Level: 0 (Hosed) Hints: Enabled Not Logged In

Home | Login/Register | Toggle Hints | Toggle Security | Drop TLS | Reset DB | View Log | View Captured Data

OWASP 2017 A1 - Injection (SQL) A2 - Injection (Other)

OWASP 2013 A2 - Broken Authentication and Session Management

OWASP 2007 A1 - Sensitive Data Exposure A4 - XML External Entities Others A5 - Broken Access Control Labs A6 - Security Misconfiguration A7 - Cross Site Scripting (XSS) A8 - Insecure Deserialization A9 - Using Components with Known Vulnerabilities A10 - Insufficient Logging and Monitoring

Help Me! Video Tutorials

TIP: Click [Hint](#) and [Videos](#) on each page

OWASP 2017 A1 - Injection (SQL) A2 - Injection (Other)

OWASP 2013 A2 - Broken Authentication and Session Management

OWASP 2007 A1 - Sensitive Data Exposure A4 - XML External Entities Others A5 - Broken Access Control Labs A6 - Security Misconfiguration A7 - Cross Site Scripting (XSS) A8 - Insecure Deserialization A9 - Using Components with Known Vulnerabilities A10 - Insufficient Logging and Monitoring

Help Me! Video Tutorials

TIP: Click [Hint](#) and [Videos](#) on each page

OWASP Mutillidae II: Keep Calm and Pwn On

Version: 2.11.23 Security Level: 0 (Hosed) Hints: Enabled Not Logged In

Home | Login/Register | Toggle Hints | Toggle Security | Drop TLS | Reset DB | View Log | View Captured Data

DNS Lookup

tion of this Page

Enter IP or hostname

Hostname/IP

Lookup DNS

Segmento OWASP Top 10 na Mutillidae II

Laboratórios

Exemplo: Laboratório 2 - Pacotes HTTP e Wireshark

OWASP Mutillidae II: Keep Calm and Pwn On

Version: 2.11.23 Security Level: 0 (None) Hints: Enabled Not Logged In

Home | Login/Register | Toggle Hints | Toggle Security | Drop TLS | Reset DB | View Log | View Captured Data

Lab 2: Capturing HTTP Packets with Wireshark

[Back](#) [Help Me!](#)

[Hints and Videos](#)

Use the Firefox browser to connect to Mutillidae. What is the first word in the user-agent string?

☐ Firefox
☐ Gecko
☐ Linux
☐ Mozilla
☐ Trident

[Submit](#)

Choose the best answer or view Hints and Videos

Lab 2

Lab Hint

Use Wireshark to observe the browser user-agent string

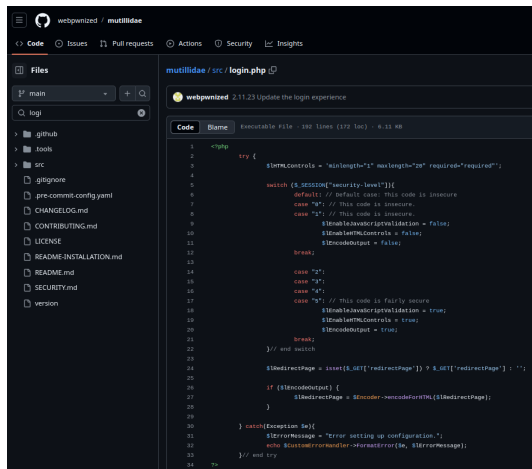
Videos

- [How to Install Wireshark in Windows 10](#)
- [Introduction to Wireshark](#)
- [Introduction to Packet Analysis - Capturing Network Traffic with TCPDump \(Part 1\)](#)
- [Introduction to Packet Analysis - Capturing Network Traffic with TCPDump \(Part 2\)](#)
- [Introduction to Packet Analysis - Packet Analysis with Wireshark \(Part 1\)](#)
- [Introduction to Packet Analysis - Packet Analysis with Wireshark \(Part 2\)](#)
- [Mutillidae: Lab 2 Walkthrough](#)

Segmento de Laboratórios na Mutillidae II

Mitigar Vulnerabilidades

Explorar o código fonte



The screenshot shows the GitHub interface for the 'webpwnized / mutillidae' repository. The 'Files' sidebar on the left shows the directory structure: main, logs, .github, .tools, and src. The 'src' directory is expanded, showing files like .gitignore, pre-commit-config.yaml, CHANGELOG.md, CONTRIBUTING.md, LICENSE, README-INSTALLATION.md, README.md, SECURITY.md, and version. The main content area displays the 'login.php' file, which is 192 lines long and 6.11 KB in size. The code is written in PHP and includes a try-catch block for handling exceptions. The code is as follows:

```
1 <?php
2     try {
3         $htmlControls = 'minlength="1" maxlength="20" required="required";
4
5         switch ($$_SESSION["security-level"]){
6             default: // Default case: This code is insecure
7                 case "0": // This code is insecure.
8                 case "1": // This code is insecure.
9                     $enableJavaScriptValidation = false;
10                    $enableHTMLControls = false;
11                    $enableOutput = false;
12                    break;
13
14                    case "2":
15                    case "3":
16                    case "4":
17                    case "5": // This code is fairly secure
18                        $enableJavaScriptValidation = true;
19                        $enableHTMLControls = true;
20                        $enableOutput = true;
21                    break;
22                } // end switch
23
24                $redirectPage = isset($_GET['redirectPage']) ? $_GET['redirectPage'] : '';
25
26                if ($enableOutput) {
27                    $redirectPage = $encoder->encodeForHTML($redirectPage);
28                }
29
30            } catch (Exception $e){
31                $errorMessage = "Error setting up configuration.";
32                echo $CustomErrorHandler->formatError($e, $errorMessage);
33            } // end try
34 }
```

Repositório da Mutillidae no GitHub

OWASP Juice Shop

Visão Geral

- Javascript/SQLite
- Cenário Realista de E-commerce
- Desafios



Fonte: owasp.org/www-project-juice-shop/

Como Utilizar?

Docker

```
1 docker run --rm -p 127.0.0.1:3000:3000 bkimminich/juice  
2   -shop
```



Fonte: docker.com

Repositório: github.com/juice-shop/juice-shop

Score Board

The screenshot shows the OWASP Juice Shop Score Board. At the top, the header includes the OWASP Juice Shop logo and navigation links for Account, Your Basket, and EN. Below the header, the Score Board section displays a progress bar for 43% completion and a Coding Score of 0%. A row of six star icons represents the difficulty levels, with the first five stars (1-5) being green and the sixth (6) being blue with a '6/11' indicator. Below the stars are buttons for 'Show all', 'Show solved', 'Show tutorials only', and 'Show unavailable'. A row of category tabs is visible, including Broken Access Control, Broken Anti Automation, Broken Authentication, Cryptographic Issues, Improper Input Validation, Injection, Insecure Deserialization, Miscellaneous, Security Misconfiguration, and Security through Obscurity. The main table lists challenges with columns for Name, Difficulty, Description, Category, Tags, and Status. The challenges are: Access Log (5 stars, Sensitive Data Exposure, solved), Admin Registration (4 stars, Improper Input Validation, solved), Allowed Bypass (5 stars, Unvalidated Redirects, solved), Bjorn's Favorite Pet (4 stars, Broken Authentication, solved), Blockchain Hype (6 stars, Security through Obscurity, unsolved), CAPTCHA Bypass (4 stars, Broken Anti Automation, solved), CSRF (4 stars, Broken Access Control, unsolved), and Change Bender's Password (6 stars, Broken Authentication, unsolved).

Name	Difficulty	Description	Category	Tags	Status
Access Log	★★★★★	Gain access to any access log file of the server.	Sensitive Data Exposure		<input checked="" type="checkbox"/> solved
Admin Registration	★★★★	Register as a user with administrator privileges.	Improper Input Validation		<input checked="" type="checkbox"/> solved
Allowed Bypass	★★★★★	Enforce a redirect to a page you are not supposed to redirect to	Unvalidated Redirects	Prerequisite	<input checked="" type="checkbox"/> solved
Bjorn's Favorite Pet	★★★★	Reset the password of Bjorn's OWASP account via the Forgot Password mechanism with the original answer to his security question.	Broken Authentication	OSINT	<input checked="" type="checkbox"/> solved
Blockchain Hype	★★★★★★	Learn about the Token Sale before its official announcement.	Security through Obscurity	Code Analysis Corruption	<input type="checkbox"/> unsolved
CAPTCHA Bypass	★★★★	Submit 10 or more customer feedbacks within 20 seconds.	Broken Anti Automation	Brute Force	<input checked="" type="checkbox"/> solved
CSRF	★★★★	Change the name of a user by performing Cross-Site Request Forgery from another origin.	Broken Access Control		<input type="checkbox"/> unsolved
Change Bender's Password	★★★★★★	Change Bender's password into slumCHastic without using SQL Injection or Forgot Password.	Broken Authentication		<input type="checkbox"/> unsolved

Acesso em: localhost:3000/#/score-board

Solution Report

OWASP Juice Shop Solution Report

GitHub

Intro

Admin Registration

Admin Section

Bjoern's Favorite Pet

Bonus Payload

Bully Chatbot

CAPTCHA Bypass

Confidential Document

CSRF

Database Schema

Deluxe Fraud

Deprecated Interface

DOM XSS

Error Handling

Exposed Metrics

Five-Star Feedback

Forged Feedback

Forged Review

GDPR Data Erasure

Login Admin

Login Amy

Login Bender

Login Jim

Five-Star Feedback

Five-Star Feedback

Difficulty: ★★

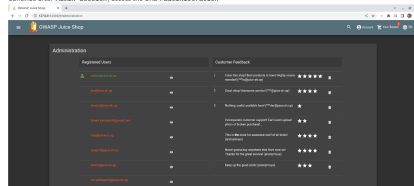
Description: Get rid of all 5-star customer feedback.

Category: Broken Access Control

Tags:

Solution:

Continue after Admin Session, access the URL /administration



Como utilizar: owasp-juice-shop.vercel.app

Mitigar vulnerabilidades

Explorar o código fonte

juice-shop / data / static / codefixes / loginAdminChallenge.info.yml

bkimminich Add info file for "Login Admin" coding challenge

Code Blame 13 lines (13 loc) · 1.29 KB

```
1 fixes:
2   - id: 1
3     explanation: "Trying to prevent any injection attacks with a custom-built blocklist mechanism is doomed"
4   - id: 2
5     explanation: "This fix unfortunately goes only half the way to using the binding mechanism of Sequelize"
6   - id: 3
7     explanation: "This fix uses the binding mechanism of Sequelize to create the equivalent of a Prepared"
8   - id: 4
9     explanation: "Using the built-in binding (or replacement) mechanism of Sequelize is equivalent to creating"
10 hints:
11   - "Try to identify any variables in the code that might contain arbitrary user input."
12   - "Follow the user input through the function call and try to spot places where it might be abused for malicious"
13   - "Can you spot a place where a SQL query is being cobbled together in an unsafe way?"
```

juice-shop / data / static / codefixes / loginAdminChallenge_1.ts

J12934 Migrate remaining commonjs require in models to esm

Code Blame 41 lines (39 loc) · 1.69 KB

```
1 import {BasketModel} from "../../models/basket";
2
3 module.exports = function login () {
4   function afterLogin (user: { data: User, bid: number }, res: Response, next: NextFunction) {
5     BasketModel.findOrCreate({ where: { UserId: user.data.id } })
6       .then(async ([basket]: [BasketModel, boolean]) => {
7         const token = security.authorize(user)
8         user.bid = basket.id // keep track of original basket
9         security.authenticatedUsers.put(token, user)
10         res.json({ authentication: { token, bid: basket.id, email: user.data.email } })
11       }).catch((error: Error) => {
12         next(error)
13       })
14   }
15
16   return (req: Request, res: Response, next: NextFunction) => {
17     if (req.body.email.match(/.*[-:]*/) || req.body.password.match(/.*[-:]*/)) {
18       res.status(451).send(res.__('SQL Injection detected.'))
19     }
20     models.sequelize.query(`SELECT * FROM Users WHERE email = '${req.body.email || ''}' AND password = '${`
```

Repositório: github.com/juice-shop/juice-shop

Demonstração

Demonstração

Roteiro

- 1 Executar a aplicação Juice Shop utilizando o Docker
- 2 Cumprir dois desafios: **Handling Error** e **Login Admin**
- 3 Mostrar o impacto associados a essas falhas

Desafio: Handling Error

Tratamento de erro inadequado

Descrição

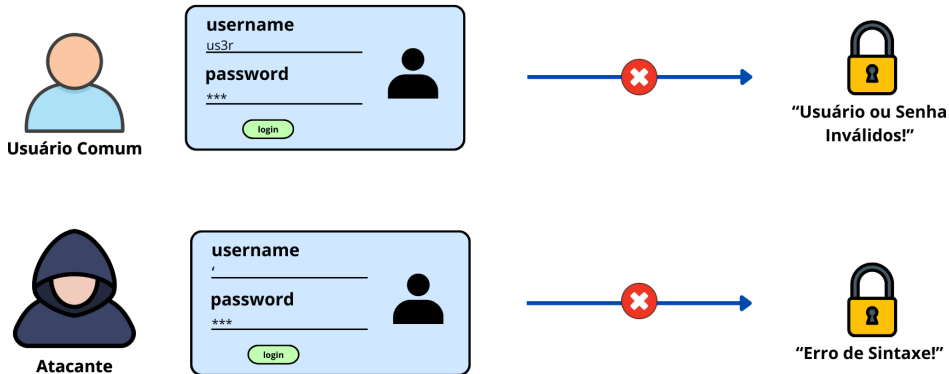
"Provocar um erro que não é tratado de forma elegante nem consistente"

Qual o problema disso?

Desafio **Login Admin**

Desafio: Handling Error

Cenário: Autenticação mal sucedida



Desafio: Handling Error

Cenário: Autenticação mal sucedida



```
SELECT * FROM users  
WHERE username "us3r"  
AND password "123";
```



```
SELECT * FROM users  
WHERE username "us3r"  
AND password "123";
```






Desafio: Handling Error

Cenário Bom e Ruim

Invalid email or password.

E-mail
user

Senha
... 

 Entrar  Log in com o google

☐ Lembrar-me

Esqueceu sua senha? Ainda não é um usuário?

[object Object]

E-mail
,

Senha
... 

 Entrar  Log in com o google

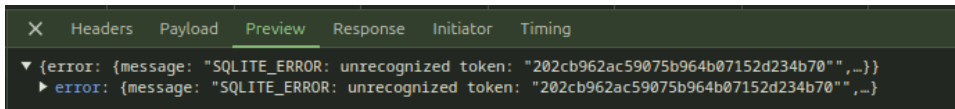
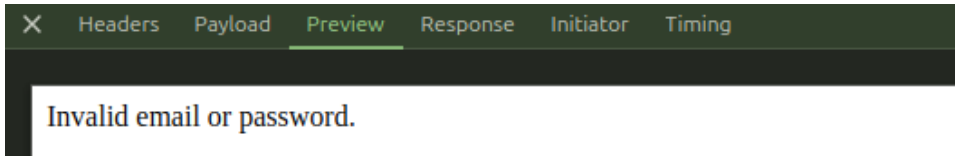
☐ Lembrar-me

Esqueceu sua senha? Ainda não é um usuário?

Interface de Login na Juice Shop

Desafio: Handling Error

Cenário Bom e Ruim



Acessar o DevTools (tecla F12) no navegador antes de realizar o login

Desafio: Login Admin

Proposta

Descrição

"Efetuar login com a conta de usuário do administrador"

Qual o problema disso?

Modificar configurações do sistema, visualizar endereços e entre outros.

Desafio: Login Admin

Cenário: Autenticação bem sucedida



Usuário Comum

```
SELECT * FROM users  
WHERE username "user"  
AND password "123";
```



Atacante

```
SELECT * FROM users  
WHERE username " ' OR 1=1 -- "  
AND password "123";
```



Desafio: Login Admin

Cenário: Autenticação bem sucedida



The image shows a dark-themed login interface. At the top, the word "Login" is displayed in white. Below it are two input fields. The first field is labeled "E-mail *" and contains the text "' OR 1=1--". The second field is labeled "Senha *" and contains three dots, with a toggle icon (an eye) on the right side. Below the password field is a link that says "Esqueceu sua senha?". At the bottom of the form is a blue "Log in" button with a right-pointing arrow icon. Below the button is a checked checkbox followed by the text "Lembrar-me".

Aplicando SQL Injection na Juice Shop

Desafio: Login Admin

Mitigar a vulnerabilidade

"Onde está o erro?"

- 1 **Repositório:** github.com/juice-shop/juice-shop
 - **Diretório com vulnerabilidade:** *"juice-shop/routes/login.ts"*
 - **Diretório com solução:** *"juice-shop/data/static/codefixes"*
- 2 **Recomendações OWASP:** cheatsheetseries.owasp.org

Desafio: Login Admin

Mitigar a vulnerabilidade

```
ts login.ts 8, M x
routes > ts login.ts > login > <function>
20 module.exports = function login () {
32 }
33
34 return (req: Request, res: Response, next: NextFunction) => {
35   verifyPreLoginChallenges(req) // vuln-code-snippet hide-line
36   models.sequelize.query('SELECT * FROM Users WHERE email = '${req.body.email} AND password = '${security.hash(req.body.password)}'
37   AND deletedAt IS NULL', { model: UserModel, plain: true }) // vuln-code-snippet vuln-line loginAdminChallenge loginBenderChallenge loginJimC
```

Código da Juice Shop com vulnerabilidade

```
ts loginAdminChallenge_4_correct.ts 9+ x
data > static > codefixes > ts loginAdminChallenge_4_correct.ts > login > <function>
3 module.exports = function login () {
15
16   return (req: Request, res: Response, next: NextFunction) => {
17     models.sequelize.query('SELECT * FROM Users WHERE email = $1 AND password = $2 AND deletedAt IS NULL',
18     { bind: [ req.body.email, security.hash(req.body.password) ], model: models.User, plain: true })
```

Código da Juice Shop sem vulnerabilidade

Desafio: Login Admin

Mitigar a vulnerabilidade



Série de folhas de dicas OWASP



Procurar



OWASP/CheatSheetSeries

☆ 28 mil 🗣 3,9 mil

Série de folhas de dicas
OWASP

Ruby on Rails

Segurança SAML

[Prevenção de injeção de SQL](#)

Gestão de Segredos

Arquitetura de Nuvem Segura

Design de produto seguro

Protegendo folhas de estilo em
cascata

Prevenção de falsificação de
solicitação do lado do servidor

Gerenciamento de Sessão

Segurança da cadeia de
suprimentos de software

Symfony

Cadeia de caracteres de cifra
TLS

Folha de dicas para prevenção de injeção de SQL

Introdução

Este cheat sheet ajudará você a evitar falhas de injeção de SQL em seus aplicativos. Ele definirá o que é injeção de SQL, explicará onde essas falhas ocorrem e fornecerá quatro opções para se defender contra ataques de injeção de SQL. Ataques [de injeção de SQL](#) são comuns porque:

1. Vulnerabilidades de injeção de SQL são muito comuns e
2. O banco de dados do aplicativo é um alvo frequente de invasores porque normalmente contém dados interessantes/críticos.

O que é um ataque de injeção de SQL?

Índice

[Exemplo de declaração
preparada Java segura](#)

[Exemplo de declaração
preparada C# .NET segura](#)

[Exemplo de instrução
preparada \(parâmetros
nomeados\) da linguagem de
consulta do Hibernate \(HQL\)](#)

[Outros exemplos de
declarações preparadas
seguras](#)

[Opção de defesa 2:
Procedimentos armazenados](#)

[Abordagem segura para
procedimentos
armazenados](#)

[Quando os procedimentos
armazenados podem
aumentar o risco](#)

Dicas para desenvolvimento: cheatsheetseries.owasp.org

SQL Injection: Como eu pensaria nisso?



Para saber mais:

github.com/payloadbox/sql-injection-payload-list

Síntese

Síntese

- 1 Conhecimento básico em projetos da OWASP (exemplo: Top 10)
- 2 Conhecer ambientes seguros para aprender vulnerabilidades (exemplo: Mutillidae e Juice Shop)
- 3 Simular ataques na prática (exemplo: SQL Injection)
- 4 Uso de Ferramentas de Segurança (exemplo: Cheat Sheet Series)

Referências

Projetos da OWASP:

OWASP Juice Shop

OWASP Mutillidae II

OWASP Top 10

OWASP Risk Rating Methodology

OWASP Dependency Check

OWASP Cheat Sheets

Ferramentas Extra de Estudo

jwt.io

hashes.com

md5hashing.net

Ícones:

Flavicon

Canva

Extra

Como descobrir a senha?

No desafio **Login Admin**, a senha do usuário não foi utilizada.
Ainda assim, o atacante poderia querer descobri-la.

Análise da Autenticação

Name	X Headers Payload Preview Response Initiator Timing Cookies
<input checked="" type="checkbox"/> styles.css	▼ Request Payload view source
whoami	▼ {email: "' OR 1=1--", password: "123"}
whoami	email: "' OR 1=1--"
login	password: "123"

Name	X Headers Payload Preview Response Initiator Timing Cookies
<input checked="" type="checkbox"/> styles.css	▼ {authentication: {,...}}
whoami	▼ authentication: {,...}
whoami	bid: 1
whoami	token: "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXRzIiw"
login	umail: "admin@juice-sh.op"

Login do atacante na Juice Shop pelo DevTools (Google)

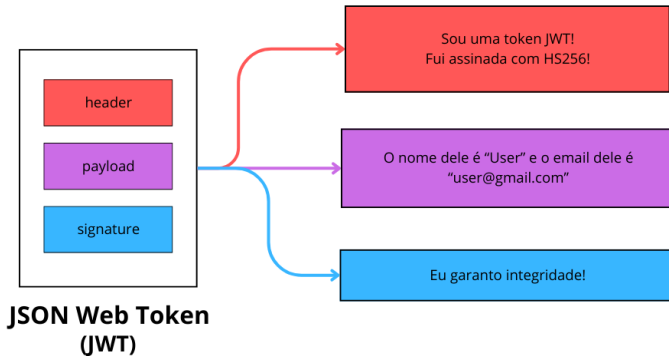
Como saber a senha?

Algoritmo

- 1 **Análise do token JWT:** Verificar o resumo criptográfico da senha
- 2 **Algoritmo de hash:** Verificar qual algoritmo de hash utilizado para guardar a senha
- 3 **Senha do usuário:** Encontrar a senha do usuário a partir do resumo criptográfico

1) Análise do token JWT

Estrutura



1) Análise do token JWT

Encoded

PASTE A TOKEN HERE

```
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXRzIiwiaWF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFPbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pb2IiImRlbHV4ZVRva2VuIjoiaiwibGFzdExvZ2luSXAiOiJ1bmRlZmluZWQiLCJwcm9maWx1SW1hZ2UiOiJhc3NldHMvcH VibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pb2I5bWVmc2IiLCJ0b3RwU2VjcmV0IjoiaiwiaXNB Y3RpdmUiOiNydWUsImNyZWZ0ZWRBdCI6IjIwMjQ tMTAtMzEgMTY6NDA6MzUuMTY5ICswMDowMCIsIn VwZGF0ZWRBdCI6IjIwMjQ tMTAtMzEgMTY6NDE6N TMuOTA0ICswMDowMCIsImRlbGV0ZWRBdCI6bnVs bH0sIm1hdCI6MTc2MDM5ODU0E4M30.r-25-BU6Vb_j1s4DssTYDYFjos8yjkSyTNRWM5THq4n zwP3-JHpA5CSenZxe55DM77Wf7IFb-FRy5GS-LGPZNpSXRf1FzsFRVv559zeZ7LarYzZYOHY_1d2 wxthU74-rKHB_yw0RbEP1MCWDXYy3Gq_3_Uez-9SvJnmExldb-g"
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

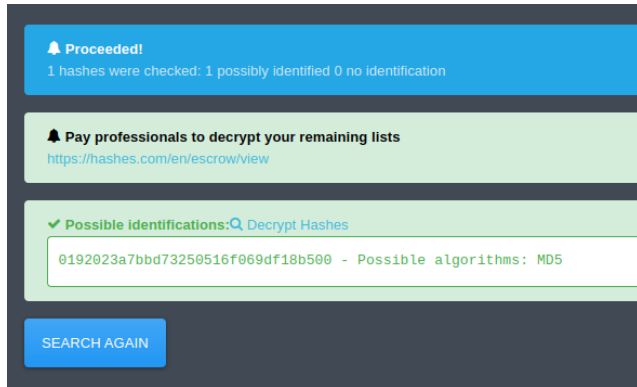
```
{  "typ": "JWT",  "alg": "RS256"}
```

PAYLOAD: DATA

```
{  "status": "success",  "data": {    "id": 1,    "username": "",    "email": "admin@juice-sh.op",    "password": "0192023a7bbd73250516f069df18b500",    "role": "admin",    "deluxeToken": "",    "lastLoginIp": "undefined",    "profileImage": "assets/public/images/uploads/defaultAdmin.png",    "totpSecret": "",    "isActive": true,    "createdAt": "2024-10-31 16:40:35.169 +00:00",    "updatedAt": "2024-10-31 16:41:53.904 +00:00",    "deletedAt": null  },  "iat": 1730398183}
```

Ferramenta: jwt.io

2) Algoritmo de hash



The screenshot displays the hashes.com interface with the following elements:


- Proceeded!** (blue notification bar):
 - 1 hashes were checked: 1 possibly identified 0 no identification
- Pay professionals to decrypt your remaining lists** (green bar):
 - <https://hashes.com/en/escrow/view>
- Possible identifications:** (green bar) with a magnifying glass icon and a link to [Decrypt Hashes](#).
- Hash and Algorithm:** A white box containing the text: `0192023a7bbd73250516f069df18b500` - Possible algorithms: MD5
- SEARCH AGAIN** (blue button)

Ferramenta: hashes.com

3) Senha do usuário


Md5 hash
calculated hash digest

0192023a7bbd73250516f069df18b500

 Copy Hash

Md5 value
Reversed hash value

admin123

 Copy Value

Ferramenta: md5hashing.net