

# Pentesting API Starter Pack 101

OWASP Vitoria 28/04/2021

<https://www.youtube.com/watch?v=wugLdoVjoDw>



OWASP®

```
$ cat .agenda
```

- ❖ Disclaimer && whoami && objetivo
- ❖ Contexto
- ❖ Introdução à API's
- ❖ Vetores de Exploração
- ❖ Metodologia
- ❖ Tools
- ❖ Referências

# ./Disclaimer

- ❖ I don't speak on behalf of my employer.
  - ❖ All the ideas and information presented here are from myself
  - ❖ All the external documents and papers referenced here will be linked on “References”, a big thanks for the security professionals and they researches.

```
env | grep -iE “user|home”
```

- ❖ Entusiasta de segurança ofensiva, comunidade hacker e etc;
- ❖ Blog: <https://d34dfr4m3.d3sec.home.blog/>;
- ❖ Github: <https://github.com/d34dfr4m3>;
- ❖ Certificações: LPI1, LPI2, ISO27k, OSCP, ITILv4 e eWPTXv2;
- ❖ Consultor de Segurança Sênior | RedTeamer | Pentester



# echo \$OBJETIVO

- ❖ O objetivo desta talk não é cobrir todos os conceitos, vulnerabilidades, técnicas, táticas ou procedimentos relacionados ao tópico em questão.
- ❖ A expectativa, é que no pouco tempo de duração de agenda, os participantes obtenham os insumos necessários para terem uma melhor experiência com o tópico abordado!

# Contexto

Cases Públicos de explorações de API

Bug Bounty

# Casos Pùblicos de exploração em API's

## Venmo Payments

- ❖ Public Endpoint (No authorization/Authentication Required);
- ❖ Get Request returns latest 20 transactions made on the app by anyone around the world;
- ❖ With Automation, its was possible to scrape 115,000 transactions per day
- ❖ <https://www.wired.com/story/i-scraped-millions-of-venmo-payments-your-data-is-at-risk/>

## Facebook's Breaches

- ❖ In September of 2018, hackers used a vulnerability in Facebook's Developer API to expose millions of users;
- ❖ December of 2018, Facebook photo API exposure exposed private data in a breach affecting up to 6.8 million users and 1,500 apps;
- ❖ Facebook's API per se, over 267 million Facebook IDs, phone numbers, and names were scraped.
- ❖ <https://www.comparitech.com/blog/information-security/267-million-phone-numbers-exposed-online/>
- ❖ <https://developers.facebook.com/blog/post/2018/12/14/notifyin-g-our-developer-ecosystem-about-a-photo-api-bug/>
- ❖ <https://www.nytimes.com/2018/09/28/technology/facebook-hack-data-breach.html>

# Bug Bounty

 zombiehelp54 submitted a report to HackerOne. Jan 8th (4 years ago)

**Description:**

I have found a security vulnerability that allows an attacker to disclose any user's private email. An attacker can disclose any user's private email by creating a sandbox program then adding that user to a report as a participant. Now if the attacker issued a request to fetch the report through the API , the response will contain the invited user private email at the activities object.

**Steps to reproduce:**

1. Go to any report submitted to your program.
2. Add the victim username as a participant to your report.
3. Generate an API token.
4. Fetch the report through the API

```
curl "https://api.hackerone.com/v1/reports/[report_id]" \
-u "api_id:token"
```

The response will contain the invited user email at the `activities` object:

```
"activities": {"data": [{"type": "activity-external-user-invited", "id": "1406712", "attributes": {"message": null, "created_
```

❖ <https://hackerone.com/reports/196655>

# Introdução a API's

Visão Geral

# whatis API

- ❖ Interface de Programação de Aplicações ou Interface de Programação de Aplicação, cuja sigla API provém do Inglês Application Programming Interface, é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por aplicativos que não pretendem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços.
- ❖ Existem diversas APIs, por exemplo: Windows API, remote API's (ex: RPC).
- ❖ O nosso foco será em API's WEB, as duas mais comuns/conhecidas:
  - ❖ Web services (SOAP/XML)
  - ❖ Rest API's (JSON)

# whatis API | grep caracteristicas

- ❖ Possuem padrão de input/output, logo, pode ser acoplado/programado;
- ❖ É independente de linguagem, deve/deveria (rs) funcionar em plataformas de tecnologias (linguagens) diferentes;
- ❖ Tem como objetivo ser seguro (Aceita apenas métodos predefinidos)

# whatis REST

- ❖ Representational State Transfer (REST)
- ❖ Geralmente, o método solicitado é chamado no resource path, exemplo:
  - ❖ GET /api/v2/methodName
- ❖ Dependendo da requisição, os parâmetros podem ser passados de forma diferente.
- ❖ Significado geral de métodos HTTP em REST (Boas práticas, não são requisitos, diferentes implementações são possíveis, ex: post para autenticar):
  - ❖ Get – Obter Resource
  - ❖ Post – Criação de recurso
  - ❖ Put – Update de recurso
  - ❖ Delete – Delete
  - ❖ PATCH - Update parcial de recurso

# whatis REST

- ❖ Um exemplo de requisição REST:
  - ❖ O Path geralmente contém a versão da API
  - ❖ Content-Type application/json header é necessário
  - ❖ Parâmetros são passados em um array JSON
    - ❖ Podem ser passados no formato XML

Request

Pretty Raw \n Actions ▾

```
1 POST /api/v2/notes HTTP/1.1
2 Host: dvws.local
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlc2VyIjo9LjQyNjEwOTk3MjUxNjIwLCJleHAiOjE2NTUyNjUyNjAsImF1ZCI6ImF1ZCIsImVtYWlsIjoiZmF3dXNzb24ubG9naW4uY29tIiwidXNlcm5hbWUiOiJsb2dpbiIsInByZWZhbWUiOiJsb2dpbiJ9
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 54
10 Origin: http://dvws.local
11 Connection: close
12 Referer: http://dvws.local/notes.html
13
14 {
    "name": "note01",
    "body": "notetextfield",
    "type": "note"
}
```

# whatis REST | grep wadl

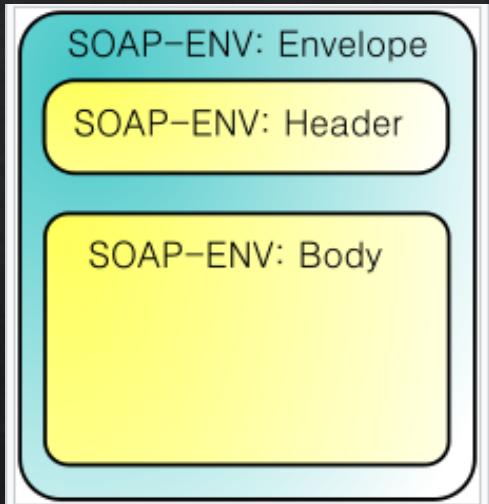
- ❖ REST API possuem uma documentação padrão chamada de WADL.
- ❖ Semelhante ao WSDL (SOAP), serão apresentadas ferramentas para reconstruir os metodos e ganhar tempo.
- ❖ Para facilitar a vida dos desenvolvedores, algumas APIs possuem uma representação human-friendly. Por exemplo o Swagger, possui a descrição de requisição para cada método:
  - ❖ Exemplo de Swagger API: <https://swagger.io/tools/swagger-ui/>
- ❖ Outros recursos podem ser verificados nos links:
  - ❖ <https://swagger.io/>
  - ❖ <https://www.w3.org/TR/wsdl.html>
  - ❖ <https://www.w3.org/Submission/wadl/>
  - ❖ <https://www.w3.org/TR/soap/>

# whatis SOAP

- ◊ SOAP (Simple Object Access Protocol, em português Protocolo Simples de Acesso a Objetos) é um protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída. Ele se baseia na Linguagem de Marcação Extensível (XML) para seu formato de mensagem, e normalmente baseia-se em outros protocolos da camada de aplicação, mais notavelmente em chamada de procedimento remoto (RPC) e Protocolo de transferência de hipertexto (HTTP),

# whatis SOAP

- ◊ SOAP Messages (HTTP Requests) seguem o formato XML e devem conter alguns elementos especiais:
  - ◊ Content type text/xml é requerido/permitido
  - ◊ SOAPAction is sometimes used just for the standard and sometimes needs to hold the called method name.



# whatis SOAP | grep wsdl

- ◊ API podem conter documentações human and machine-readable. Para API baseadas em SOAP, a documentação é armazenada em arquivos WSDL. Usualmente, esses arquivos ficam armazenados em paths “?wsdl” , exemplo:
  - ◊ <https://api.example.com/api/?wsdl>.
- ◊ Exemplo de serviço SOAP:
  - ◊ <http://www.dneonline.com/calculator.asmx>
- ◊ No endereço <http://www.dneonline.com/calculator.asmx?op=Add>, é possível visualizar um exemplo de requisição SOAP para o serviço de calculadora.
- ◊ A documentação completa pode ser visualizada no endereço abaixo:
  - ◊ <http://www.dneonline.com/calculator.asmx?wsdl>

# Vetores de Ataque

Visão Geral

# OWASP API Security Top 10 - 2019

- ❖ API1:2019 - Broken Object Level Authorization
- ❖ API2:2019 - Broken User Authentication
- ❖ API3:2019 - Excessive Data Exposure
- ❖ API4:2019 - Lack of Resources & Rate Limiting
- ❖ API5:2019 - Broken Function Level Authorization
- ❖ API6:2019 - Mass Assignment
- ❖ API7:2019 - Security Misconfiguration
- ❖ API8:2019 – Injection
- ❖ API9:2019 - Improper Assets Management
- ❖ API10:2019 - Insufficient Logging & Monitoring

# Vetores de Ataque

- ❖ Access Control
- ❖ JWT
- ❖ API Keys
- ❖ Restricted HTTP Methods
- ❖ Input Validation
- ❖ Validação content Types
- ❖ Management endpoints
- ❖ Error Handling
- ❖ Audit Logs
- ❖ Security Headers
- ❖ CORS
- ❖ Sensitive Information in HTTP Requests
- ❖ HTTP Return Code

# Vetores de Ataque

- ❖ **Access Control**
  - ❖ JWT
  - ❖ API Keys
  - ❖ Restricted HTTP Methods
- ❖ **Input Validation**
  - ❖ Validate content Types
  - ❖ Management endpoints
- ❖ Error Handling
- ❖ Audit Logs
- ❖ Security Headers
- ❖ CORS
- ❖ Sensitive Information in HTTP Requests
- ❖ HTTP Return Code

# Access Control

Vetores de Ataque

# Access Control (Authorization/Authentication)

- ❖ Em grandes implementações de API, nem todo método é designado para ser usado por qualquer usuário. Por exemplo, muitas implementações segmentam em perfis (roles) que definem roles read-only e read-write.
- ❖ É mais comum que a API utilize mecanismos de autenticação basic (basic Authentication) do que cookies. Por exemplo, dentro do header da requisição X-Api-Token: aaa.
- ❖ Devido a alguns requerimentos da API, alguns content types específicos ou headers customizados podem ser usados com o método de autenticação sem cookies, nesse sentido, fica menos vulnerável a CSRF.
- ❖ Contudo, geralmente são encontradas falhas de quebra de controle de acesso em API's, é comum encontrar contornos (Bypass) de autorização.
- ❖ Serviços REST não-Publicos devem controlar o acesso para cada endpoint da API. Serviços Web em uma implementação monolítica implementam o controle através de autenticação de usuário, lógica de autorização e gerenciamento de sessão. Essa estrutura tem várias desvantagens em relação a arquiteturas modernas que compõem múltiplos microserviços no estilo RESTful.
  - ❖ Em virtude de diminuir a latência e o acoplamento entre os serviços, a decisão de controle de acesso deve ser realizada pelos endpoints REST
  - ❖ A autenticação de usuários deve ser centralizada em um Provedor de Identidade (IdP), que delega tokens de acesso.

# Formas comuns de validar bugs de Access Control

- ❖ Enumerar potenciais endpoints restritos;
- ❖ Modificar tokens de sessão;
- ❖ Reutilizar tokens de sessão antigos;
- ❖ Contornar (bypass) restrições de acesso (IDOR);
- ❖ Modificar a requisição com parâmetros adicionais como “&admin=True” ou outros;
- ❖ Modificar headers referrer que a aplicação possa estar esperando.

# Preparando o ambiente para testar falhas de controle de acesso ‘complexas’:

- ❖ Preparar as requisições válidas e funcionais para cada endpoint da API;
- ❖ Gerar tokens válidos (ou headers de autorização) para cada usuário da API (Modelo GreyBox);
- ❖ Combinar cada requisição da API com cada token gerado e mapear o comportamento;
- ❖ Testar cada requisição sem um token válido também.

# Tokens de API geralmente são suspeitos por vulnerabilidades de sessão

- ❖ Baixa entropia ou valores preditivos;
- ❖ Invalidação/destruição inadequadas;
- ❖ Possível vazamento de tokens pela infraestrutura da aplicação ou a capacidade de gerar tokens válidos;
- ❖ Tokens específicos que podem conceder acesso a uma interface da API são Json Web Tokens (JWT), também chamados de Bearer Authentication.

# Live PoC

## API1:2019 Broken Object Level Authorization

- ❖ APIs tend to expose endpoints that handle object identifiers, creating a wide attack surface Level Access Control issue. Object level authorization checks should be considered in every function that accesses a data source using an input from the user.

```
git clone https://github.com/snoopysecurity/dvws-node.git  
docker-Compose up -d
```

(Caso o serviço Web não inicie): docker-compose start web

- Users:
  - test:test
  - admin:letmein
- Steps:
  - Login as admin and create a secret note
  - Login as test and find out how reach the admins note



# Input Validation

Vetores de Exploração

# Input Validation

- ❖ Do not trust input parameters/objects.
- ❖ Validate input: length/range/format and type.
- ❖ Achieve an implicit input validation by using strong types like numbers, booleans, dates, times or fixed data ranges in API parameters.
- ❖ Constrain string inputs with regexps.
- ❖ Reject unexpected/illegal content.
- ❖ Make use of validation/sanitation libraries or frameworks in your specific language.
- ❖ Define an appropriate request size limit and reject requests exceeding the limit with HTTP response status 413 Request Entity Too Large.
- ❖ Consider logging input validation failures. Assume that someone who is performing hundreds of failed input validations per second is up to no good.
- ❖ Have a look at input validation cheat sheet for comprehensive explanation.
- ❖ Use a secure parser for parsing the incoming messages. If you are using XML, make sure to use a parser that is not vulnerable to XXE and similar attacks.

# Pontos de falhas comuns para testar a API:

- ❖ No Header da Requisição
- ❖ Nos parâmetros da URL
- ❖ Parâmetros da Requisição
- ❖ Upload de Arquivos (put/delete requests)
- ❖ Testar diferentes métodos de requisição (Get,Post,Head,Put e etc)

# Live PoC

## API8:2019 Injection

- ❖ Injection flaws, such as SQL, NoSQL, Command Injection, etc., occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's malicious data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

```
git clone https://github.com/snoopysecurity/dvws-node.git  
docker-Compose up -d
```

(Caso o serviço Web não inicie): docker-compose start web

- Users:
  - test:test
  - root:mysecretpassword
- Serviço de geração de Senha



# Metodologias

The wayy you go

# Create your own methodology

- ❖ Recon
- ❖ Vulnerabilidades Técnicas (RCE, SQLi, XXE, XSS e etc)
- ❖ Vulnerabilidades lógicas (IDOR, Priv Esc, info Leak e etc)

# Abordagens comuns

## BlackBox

- ❖ What is the API name and version? Is it a custom implementation or, for example, an open-source product?
- ❖ Is there any online documentation available? Are there any interesting methods?
- ❖ Does the documentation exist on the target server (?wsdl, ?wadl, or similar)?
- ❖ Does the API require authentication, or is publicly available?
- ❖ If there is both local and public documentation for an API, do they match? Maybe some methods were hidden from local users (typically ones that allow insecure operations).
- ❖ Fuzzing/Tampering
- ❖ OWASP CheatSheet

## GreyBox

- ❖ Pre-Engagement Interactions
  - ❖ Contas de diferentes privilégios (user, admin)
- ❖ Documentação da API

# Tools

letting the weapon be an extension of oneself

letting the tool be an extension of oneself

You are already a weapon

# Recon

- ❖ Your purpose is to gather as many API endpoints as possible and to be able to speak to them. You should also be able to get the WSDL/WADL file for further testing. Reconstructing API calls from a raw WSDL/WADL file would be time-consuming, so a proper tool might help you to do it faster. For API testing and parsing WSDL/WADL files into a ready-to-use method set, you might want to use Postman, the free edition of SOAPUI, or the Burp Pro extension called WSDLER.

# TOOLS

- ❖ Enumeração de methods através de wadl/wsdl (burp)
  - ❖ Burp
  - ❖ Postman
- ❖ Enumeração de endpoints (gobuster, dirsearch, ffuz, wfuz)
  - ❖ Wordlists: <https://github.com/danielmiessler/SecLists>

# Referências

- ❖ <https://www.bugcrowd.com/resources/webinars/api-security-testing-for-hackers/>
- ❖ <https://www.bugcrowd.com/resources/webinars/bad-api-hapi-hackers/>
- ❖ <https://infosecwriteups.com/bug-bounty-broken-api-authorization-d30c940ccb42>
- ❖ <https://hackerone.com/reports/196655>
- ❖ <https://owasp.org/www-project-api-security/>
- ❖ [https://cheatsheetseries.owasp.org/cheatsheets/REST\\_Security\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html)

Thank you security community!