

# Segurança em aplicações Web

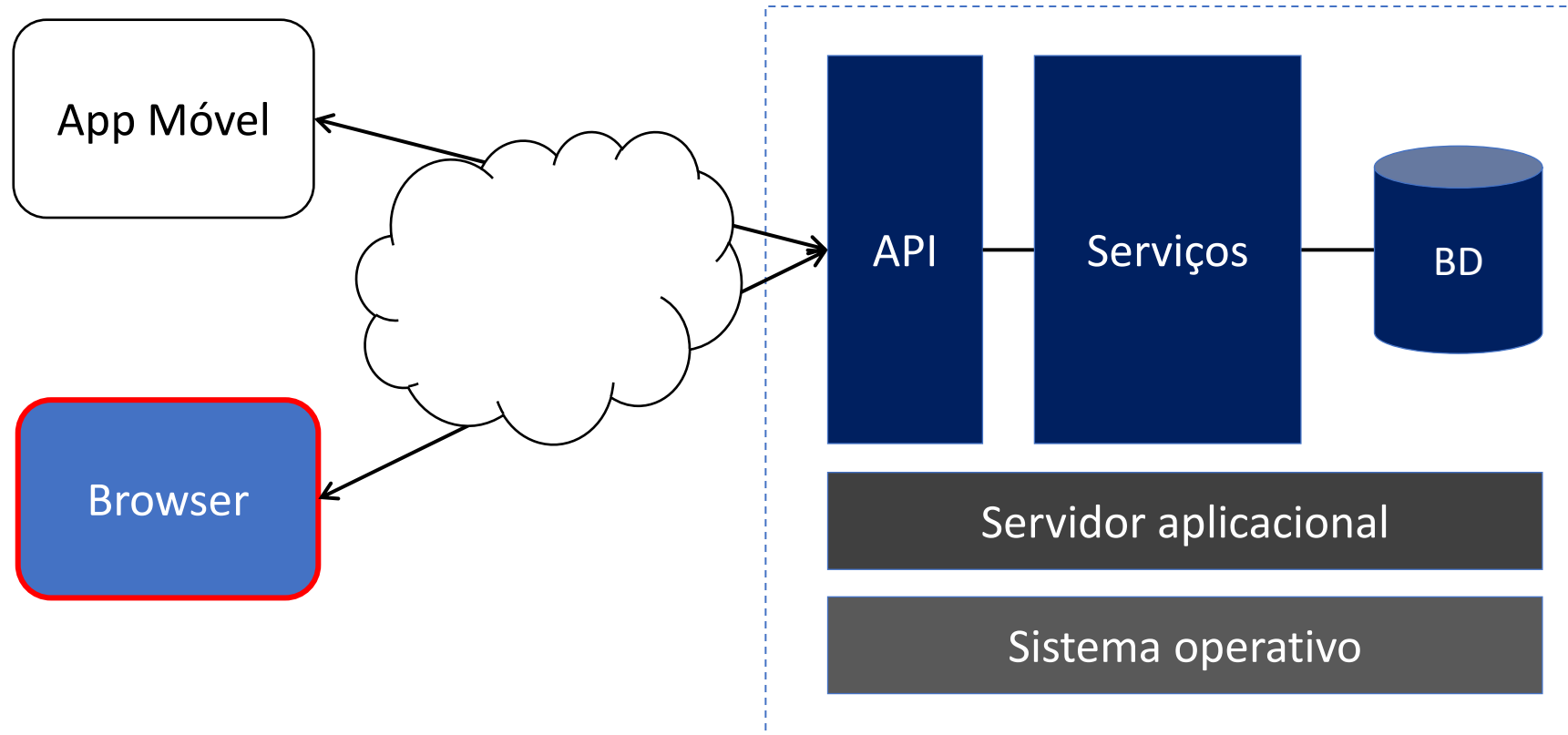
- OWASP Top 10
- Injeção de ataques
- Ferramenta ZAP

# Resumo

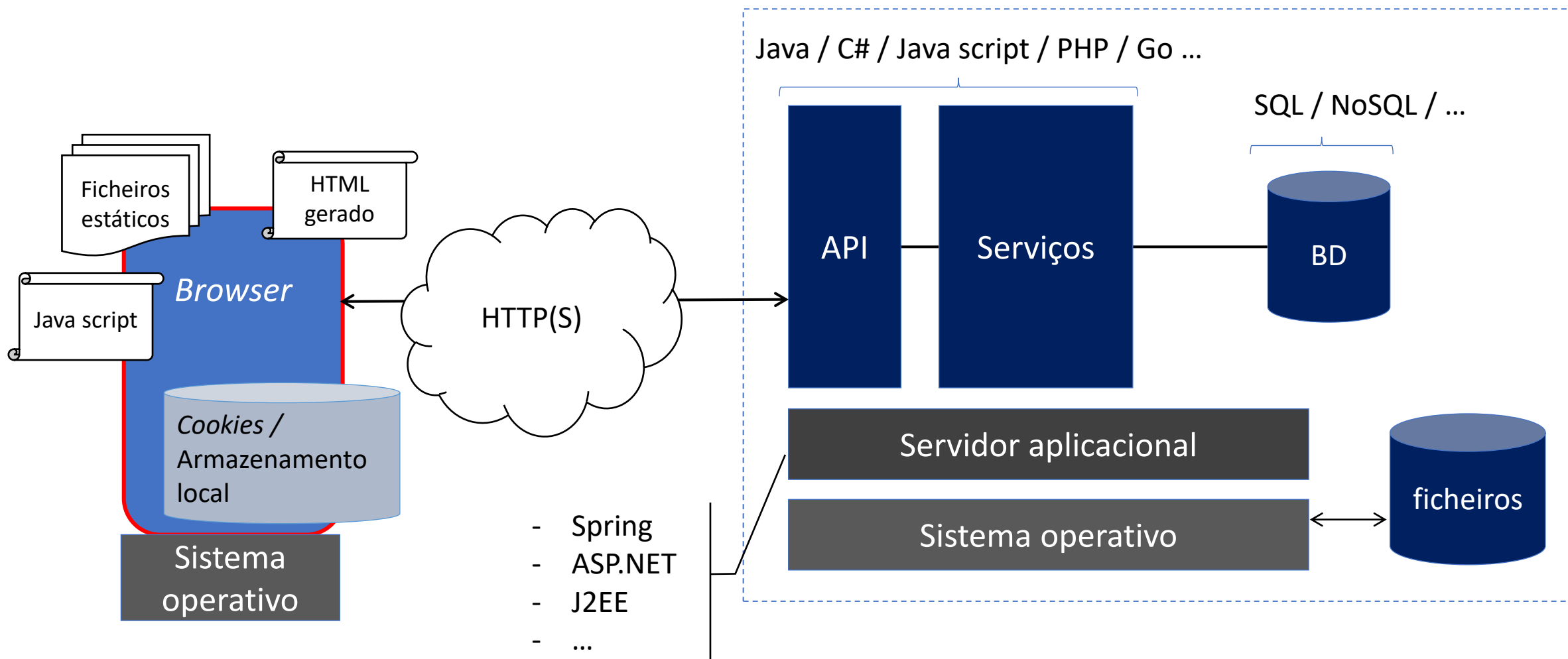
- Ameaças em aplicações web
- OWASP Top 10
- Injeção de ataques
- Fuzzing
- Caso prático com a ferramenta Zed Attack Proxy (ZAP)

# Introdução

- As aplicações web são compostas por várias partes, a correr em diferentes contextos
- O código da aplicação pode estar distribuído, desde o *browser* às base de dados



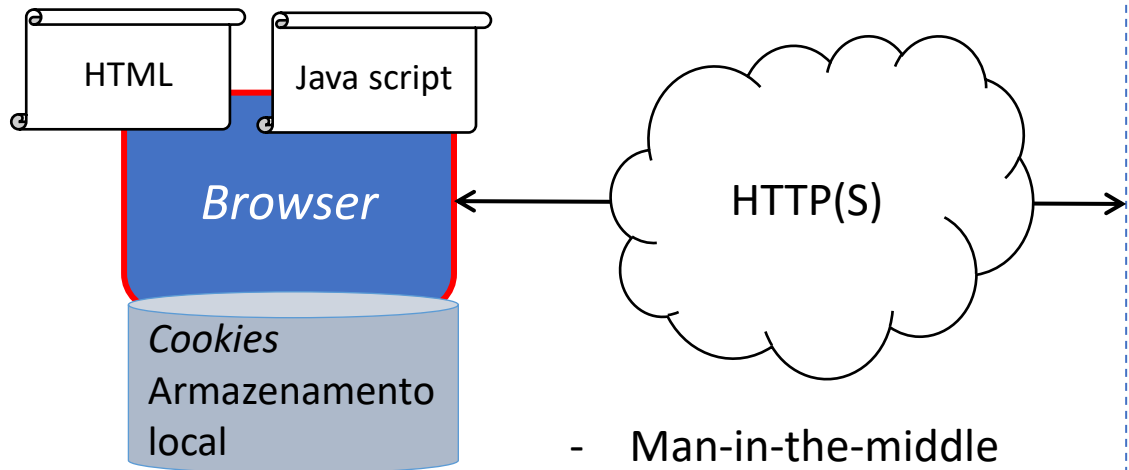
# Tecnologias



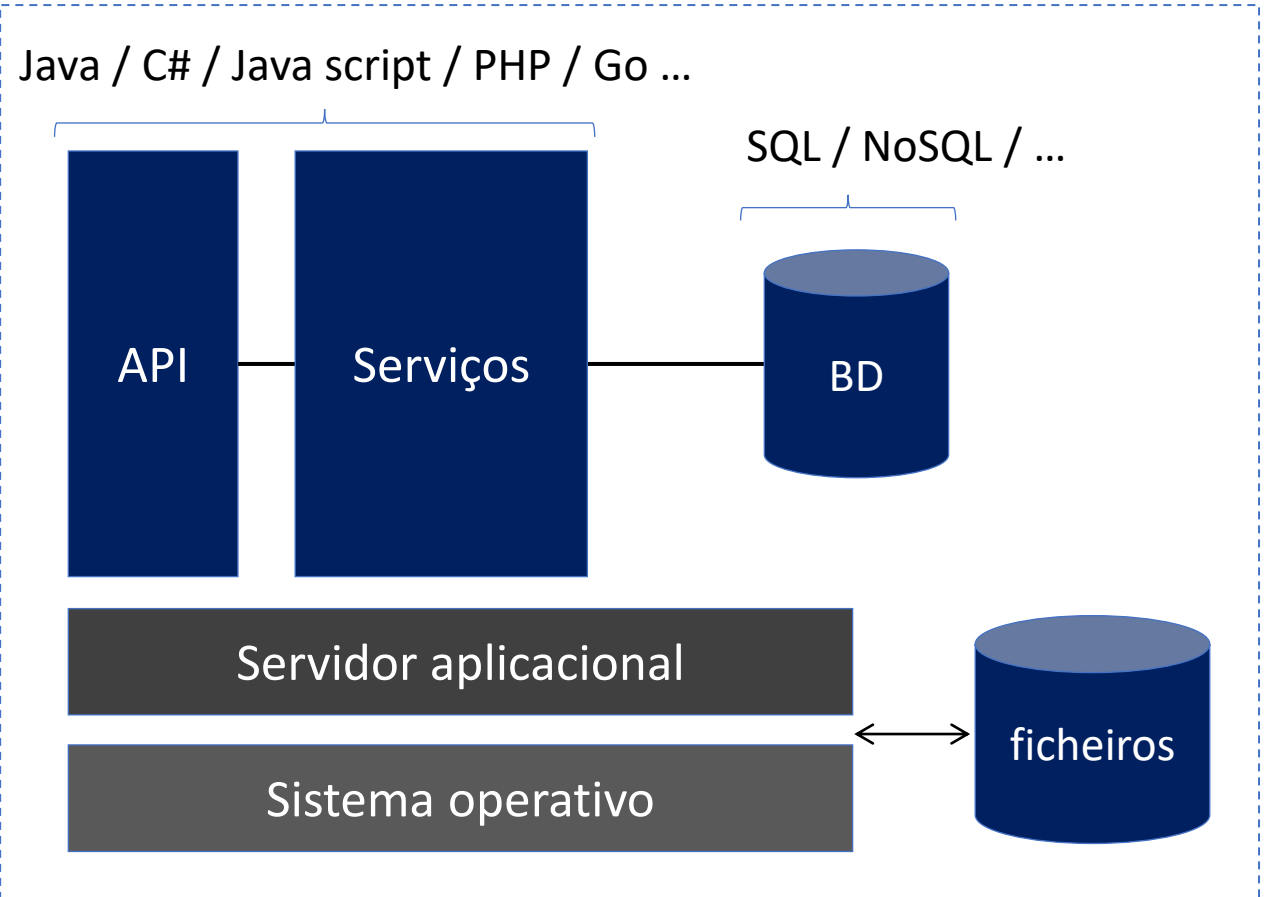
# Ameaças



- Injetar código no *browser* ou na BD
- Forjar autenticação
- Contornar controlo de acessos
- ...



- Man-in-the-middle
- Configurações inseguras
- DDoS
- ...



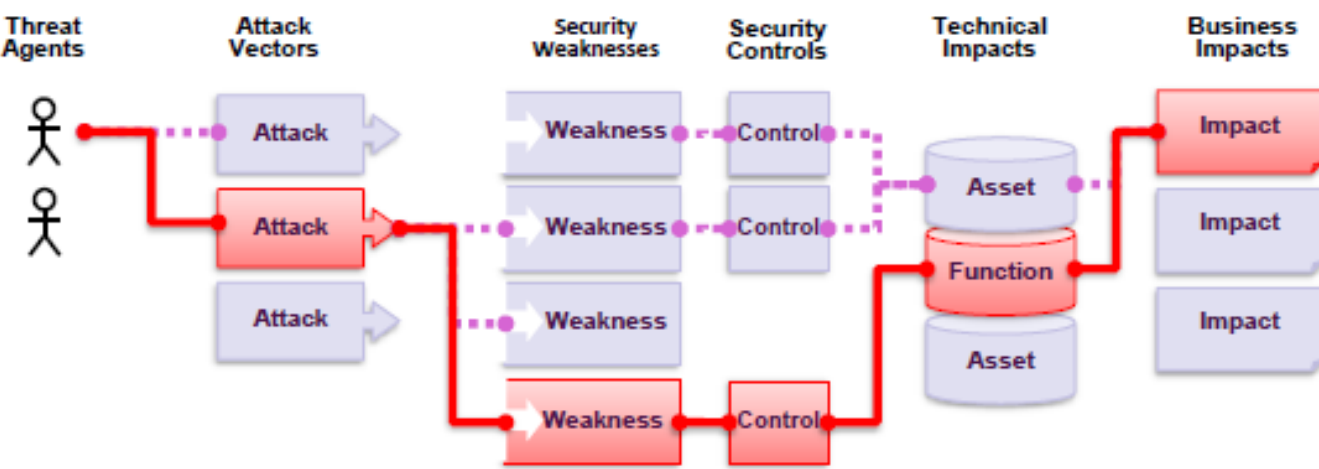
- Vulnerabilidades do SO
- Vulnerabilidades do servidor aplicativo
- ...

# OWASP

- Open Web Application Security Project (OWASP)
  - Fundação sem fins lucrativos para melhorar a segurança do software
- Comunidade mantém vários projetos abertos
  - Metodologias para identificação de riscos
  - Bibliotecas de código aberto para integração em diferentes linguagens/*frameworks*
  - Projectos para treinar capacidades de analisar segurança em sites e aplicações móveis
  - OWASP Top 10, <https://owasp.org/www-project-top-ten/>
    - Versão atual 2017
    - Versão 2020 em curso

# OWASP – riscos de segurança

- O atacante pode seguir vários caminhos para prejudicar o sistema
- Os diferentes caminhos podem ser mais fáceis ou difíceis. O impacto no negócio também pode variar
- OWASP Risk Rating



- *probabilidade \* impacto*
- *Exploitability*
- *Prevalence*
- *Detectability*

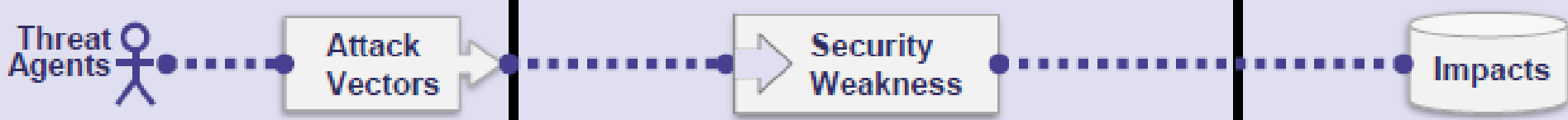
Threat Agents	Exploitability	Weakness Prevalence	Weakness Detectability	Technical Impacts	Business Impacts
Appli- cation Specific	Easy: 3	Widespread: 3	Easy: 3	Severe: 3	Business Specific
	Average: 2	Common: 2	Average: 2	Moderate: 2	
	Difficult: 1	Uncommon: 1	Difficult: 1	Minor: 1	

Exemplo  
para “Injection”

					
App. Specific	Exploitability: 3	Prevalence: 2	Detectability: 3	Technical: 3	Business ?

# A1: Injection

- Exemplos: SQL injection, Serviços de directoria (ex: LDAP)
- Prevenção
  - Usar API segura que evita misturar comandos e dados
  - *Whitelist* (lista de aceites)
  - Object Relational Mapping (ORM)

					
App. Specific	Exploitability: 3	Prevalence: 2	Detectability: 3	Technical: 3	Business ?
Almost any source of data can be an injection vector, environment variables, parameters, external and internal web services, and all types of users. <a href="#">Injection flaws</a> occur when an attacker can send hostile data to an interpreter.		Injection flaws are very prevalent, particularly in legacy code. Injection vulnerabilities are often found in SQL, LDAP, XPath, or NoSQL queries, OS commands, XML parsers, SMTP headers, expression languages, and ORM queries.  Injection flaws are easy to discover when examining code. Scanners and fuzzers can help attackers find injection flaws.		Injection can result in data loss, corruption, or disclosure to unauthorized parties, loss of accountability, or denial of access. Injection can sometimes lead to complete host takeover.  The business impact depends on the needs of the application and data.	



## A2: *Broken Authentication*

- Exemplos: Uso de listas de *password* comuns, sessões que não expiram
- Prevenção
  - Testar novas passwords contra as mais utilizadas
  - Uso de autenticação multi-factor
  - Gerar do lado do servidor identificadores de sessão aleatórios

## A3: *Sensitive Data Exposure*

- Exemplos: uso de configurações fracas de TLS, mau armazenamento de passwords
- Prevenção
  - Verificar automaticamente configurações de servidores
  - Cifrar dados armazenados / usar funções de *hash* fortes
  - Classificar importância dos dados de acordo com a legislação

## A4: XML External Entities (XXE)

- Exemplos: Aplicação aceita e processa diretamente documentos XML
- Prevenção
  - Uso de formatos menos complexos como JSON
  - Desativar o processamento de entidades externas no XML
  - *Whitelist*

## A5: *Broken Access Control*

- Exemplos: Passar controlos de segurança por modificação do URL, Elevação de privilégios por manipulação de cookies / JSON web tokens
- Prevenção
  - Com exceção dos recursos públicos, *Deny by default*
  - Implementação e teste de mecanismos de controlo de acessos para que possam ser reutilizados

## ***A6: Security Misconfiguration***

- Exemplos: Contas por omissão ativas, últimas atualizações não instaladas
- Prevenção
  - Remoção de aplicações não usadas
  - Usar processo automático para identificar configurações apropriadas

## ***A7: Cross-site scripting (XSS)***

- Exemplos: Refletido, Armazenado, DOM-based
- Prevenção:
  - Usar bibliotecas que fazem codificação correta de dados
  - Fazer codificação de output previne XSS armazenado

## ***A8: Insecure Deserialization***

- Exemplos: Serialização direta de estruturas da aplicação (e.g. para cookies)
- Prevenção:
  - Usar mecanismos que permitam verificar integridade
  - Não aceitar objetos serializados de fontes inseguras

## ***A9: Using Components with Known Vulnerabilities***

- Exemplos: Sistemas não atualizados, incluindo SO, app web, base de dados, bibliotecas, ...
- Prevenção
  - Remover dependências com componentes
  - Obter componentes apenas de fontes fidedignas

# A10: *Insufficient Logging & Monitoring*

- Exemplos: Eventos de auditoria não são registados, a aplicação não é capaz de detetar ou alertar sobre ataques em curso
- Prevenção
  - Logins e falhas de acesso são registados com contexto suficiente para identificar contas suspeitas ou maliciosas
  - Registos são guardados tempo suficiente para uma análise forense
  - Estabelecer um plano de resposta a incidentes e plano de recuperação

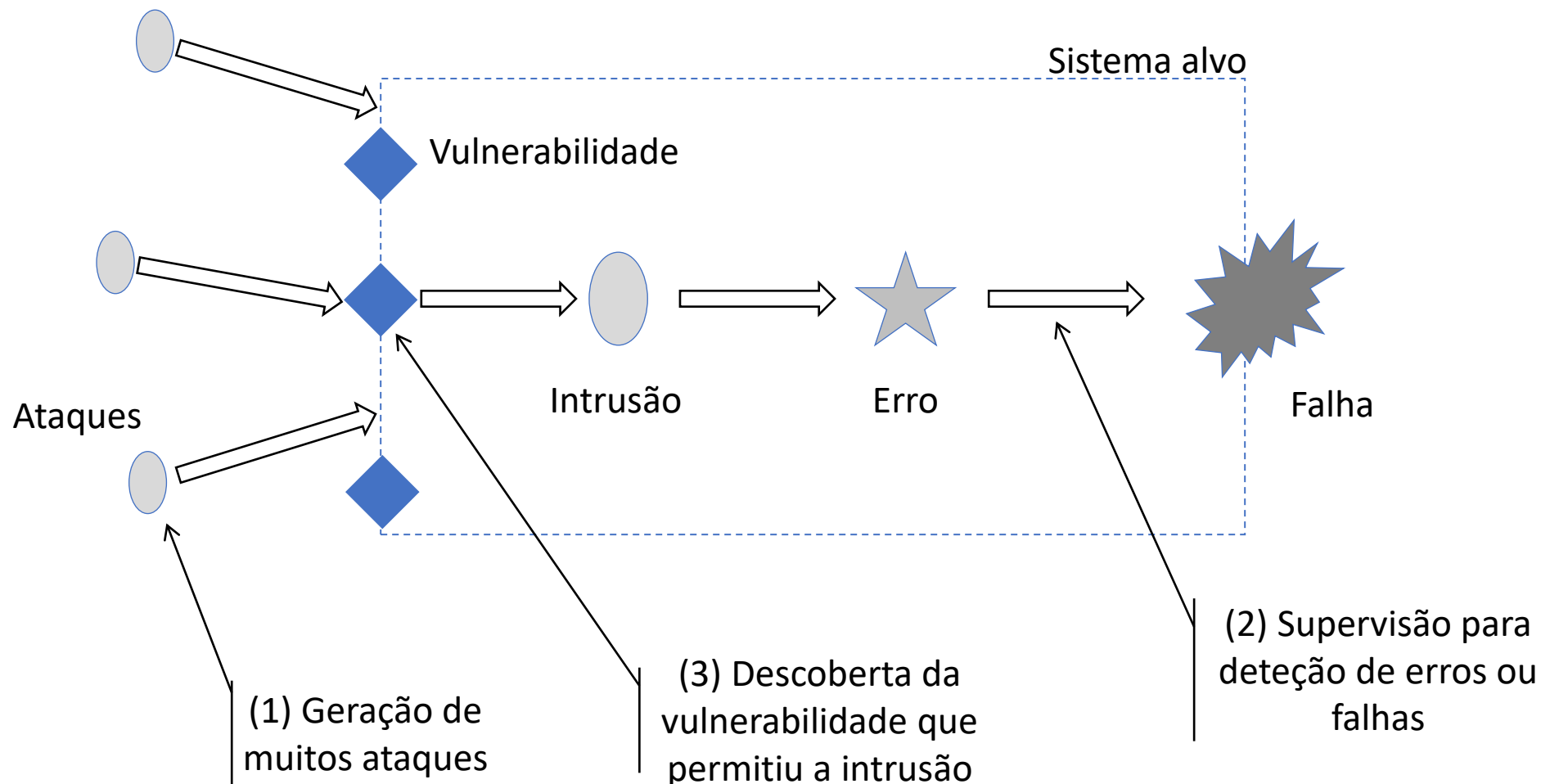
# API Security Top 10 - 2019

- <https://owasp.org/www-project-api-security/>
- Projeto em curso da OWASP que já produziu uma lista Top10
- Esta lista concentra-se em estratégias e soluções para compreender e mitigar as vulnerabilidades exclusivas e riscos de segurança de interfaces de programação de aplicativos (APIs).
- Exemplos
  - **API3:2019 Excessive Data Exposure**  
Em muitos casos as propriedades de um objeto são todas expostas, sem considerar sua sensibilidade individual, deixando para os clientes realizarem a filtragem de dados antes de exibí-los ao utilizador

# Testes de intrusão e procura de vulnerabilidades em aplicações web

Injeção de ataques, varrimento de vulnerabilidades

# Injeção de ataques



Adaptado de <https://ieeexplore.ieee.org/document/1633534>

*"Using Attack Injection to Discover New Vulnerabilities"*



# Fuzzing

«The original work was inspired by being logged on to a modem during a storm with lots of line noise. And the line noise was generating junk characters that seemingly was causing programs to crash. The noise suggested the term "fuzz".»  
*Bart Miller*

- Técnicas para encontrar falhas, injetando dados mal formatados de uma forma automática
- A geração dos dados para *fuzzing* é uma parte essencial do processo

# Fuzzers

- Os *fuzzers* podem ser:
  - Recursivos ou Substitutivos
- Recursivos: Geração das diversas combinações de um determinado alfabeto

```
http://www.example.com/00000000  
...  
http://www.example.com/11000fff  
...  
http://www.example.com/ffffffff
```

- Substitutivos: Substitui a entrada com um conjunto de entradas predefinidas

```
http://www.example.com/>"><script>alert("XSS")</script>&  
http://www.example.com/' ' ; ! -- "<XSS>=&{ ( ) }
```

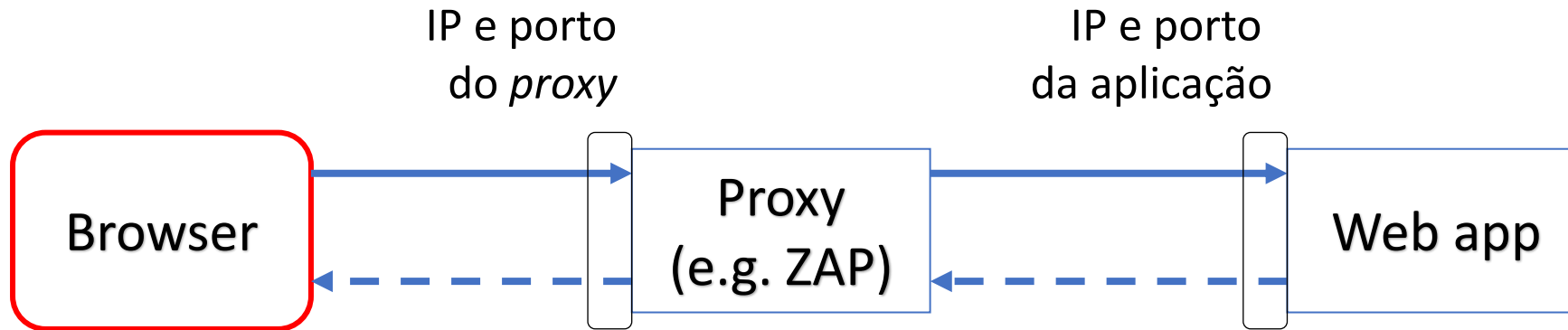
- Exemplo: FuzzDB
  - <https://github.com/fuzzdb-project/fuzzdb>

# Varredores de vulnerabilidades

- Os *fuzzers* e injetores de ataques procuram vulnerabilidades desconhecidas
- Varredores de vulnerabilidades procuram vulnerabilidades conhecidas
  - Percorrem base de dados de vulnerabilidades
  - Injetam ataques
  - Monitorizam o efeito na aplicação tentando detetar se contém a vulnerabilidade
- Requisitos gerais de um varredor de vulnerabilidades web
  - Identificar conjuntos específicos de vulnerabilidades presentes em base de dados públicas
  - Gerar relatório para cada vulnerabilidade
  - Ter uma taxa de falsos positivos aceitável

# Proxies

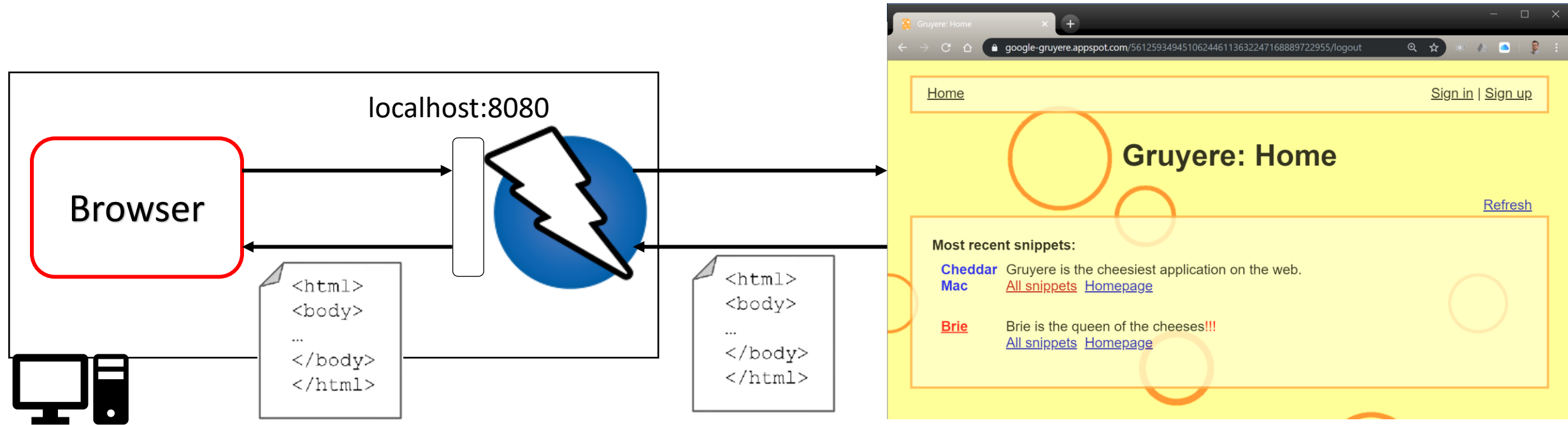
- Interseção de pedidos e respostas
- Ligações HTTPS têm de ser transparentemente intersetadas com *man-in-the-middle* autorizado



# Ferramenta ZAP

# Zed Attack Proxy (ZAP)

- <https://owasp.org/www-project-zap/>



# ZAP – modos de funcionamento

- Passivo – analisa passivamente todos os pedidos que passam por ele ou gerados pelos componentes *crawling*
  - Em termos de teste de penetração, este modo não modifica os dados do site
  - É seguro para os sites em geral, onde não há permissão para atacar
  - Deteta, por exemplo, falta de *headers* críticos ou configuração incorreta de *cookies*
- Ativo – tenta ativamente encontrar vulnerabilidades usando ataques conhecidos sobre alvos selecionados
  - Ataca o site usando técnicas conhecidas para encontrar vulnerabilidades
  - Este modo modifica dados e pode inserir *scripts* maliciosos no site
  - **Só se pode executar este modo para os sites que tenhamos permissão de teste**

# Zed Attack Proxy (ZAP)

The screenshot shows the ZAP interface with four key areas highlighted by green circles and numbered 1 through 4:

- 1** **Lista de sites analisados**: Points to the 'Sites' list on the left sidebar, which contains various URLs like `https://tracking-protection.cdn.mozilla.net` and `https://www.example.org`.
- 2** **Cabeçalhos pedido/resposta**: Points to the 'Header: Text' tab in the central pane, displaying HTTP headers such as `HTTP/1.1 200 OK`, `Cache-Control: max-age=604800`, and `Content-Type: text/html; charset=UTF-8`.
- 3** **Corpo do pedido/resposta**: Points to the 'Body: Text' tab in the central pane, displaying the HTML body content, including `<doctype html>`, `<html>`, `<head>`, `<title>Example Domain</title>`, and `<body>` with CSS styles.
- 4** **Várias informações: alertas, resultado de varrimentos, ...**: Points to the bottom section of the interface, which includes tabs for 'History', 'Search', 'Alerts', 'Output', and 'Active Scan', along with a table showing scan results.

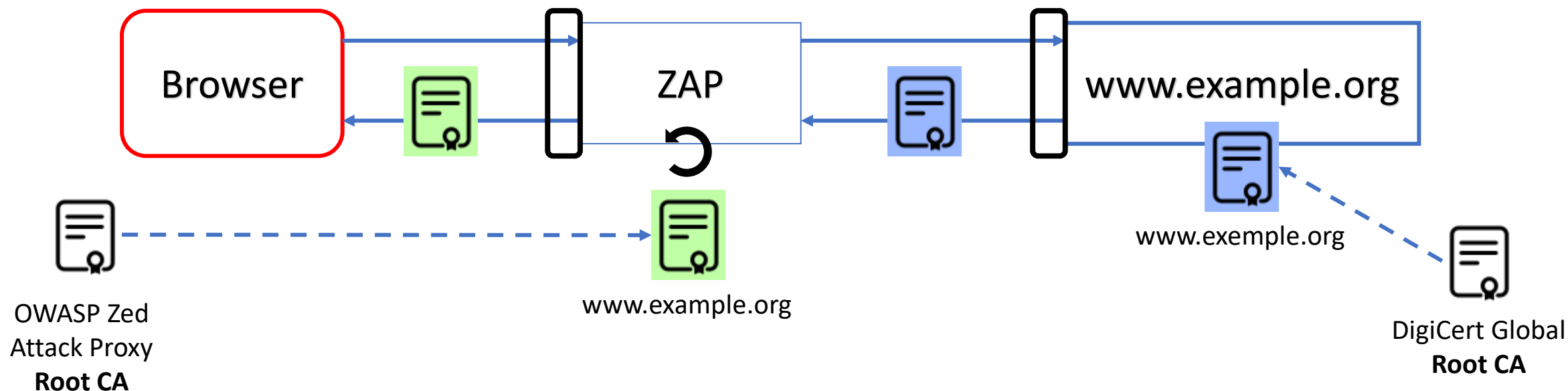
The table at the bottom displays the following data:

Id	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
110	11/5/19, 11:48:18 AM	11/5/19, 11:48:18 AM	GET	<code>https://www.example.org/1035374713256101461</code>	404	Not Found	597 ms	246 bytes	1,256 bytes
112	11/5/19, 11:48:18 AM	11/5/19, 11:48:19 AM	GET	<code>https://www.example.org/</code>	200	OK	138 ms	327 bytes	1,256 bytes
113	11/5/19, 11:48:19 AM	11/5/19, 11:48:19 AM	GET	<code>https://www.example.org/</code>	200	OK	135 ms	322 bytes	1,256 bytes

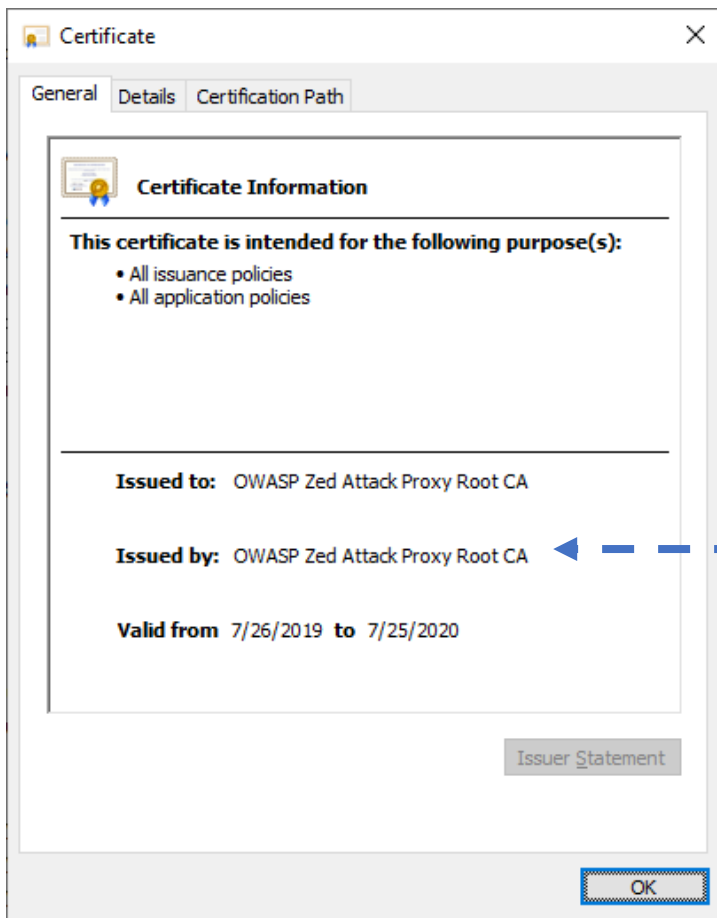


# ZAP – Setup

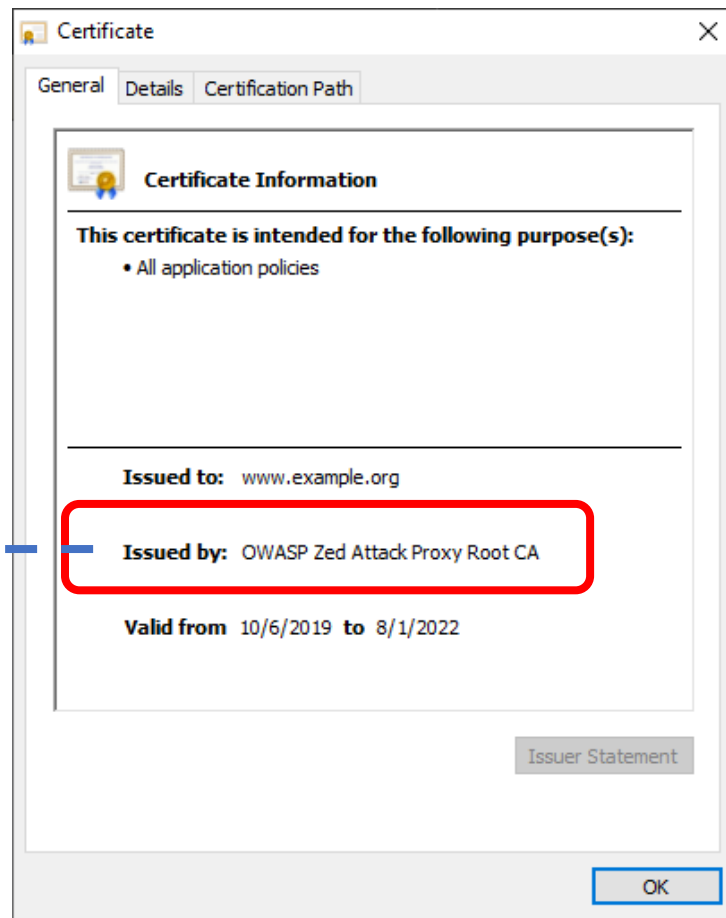
- Configurar proxy no *browser* e raiz de confiança
- Nos *sites* com HTTPS o ZAP **gera** novo certificado com o nome do original
  - O novo certificado é assinado com o “OWASP Zed Attack Proxy Root CA”



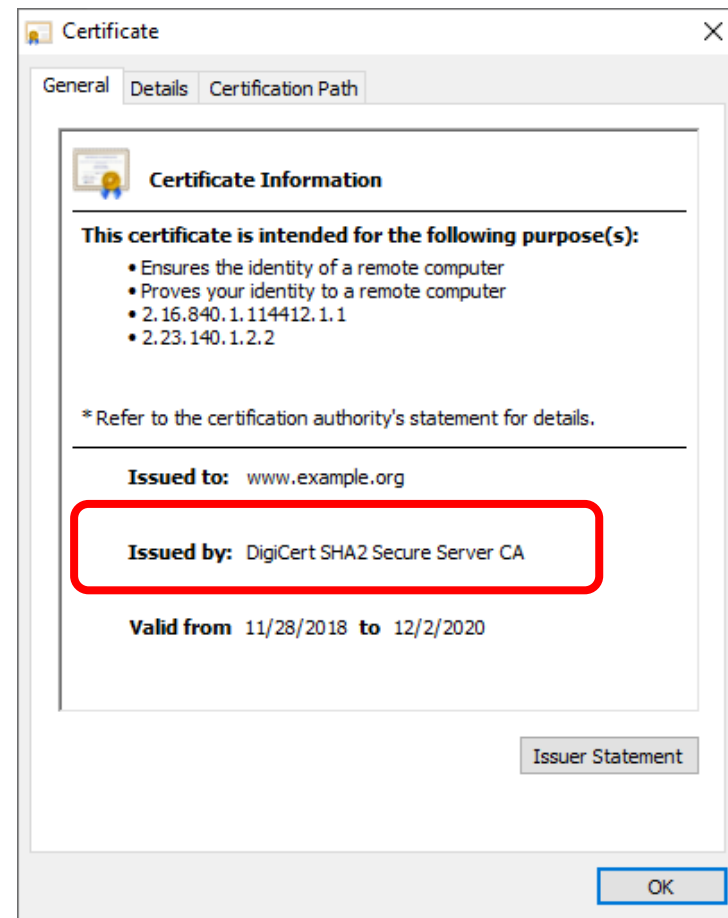
# Certificado original e modificado



Raiz de confiança



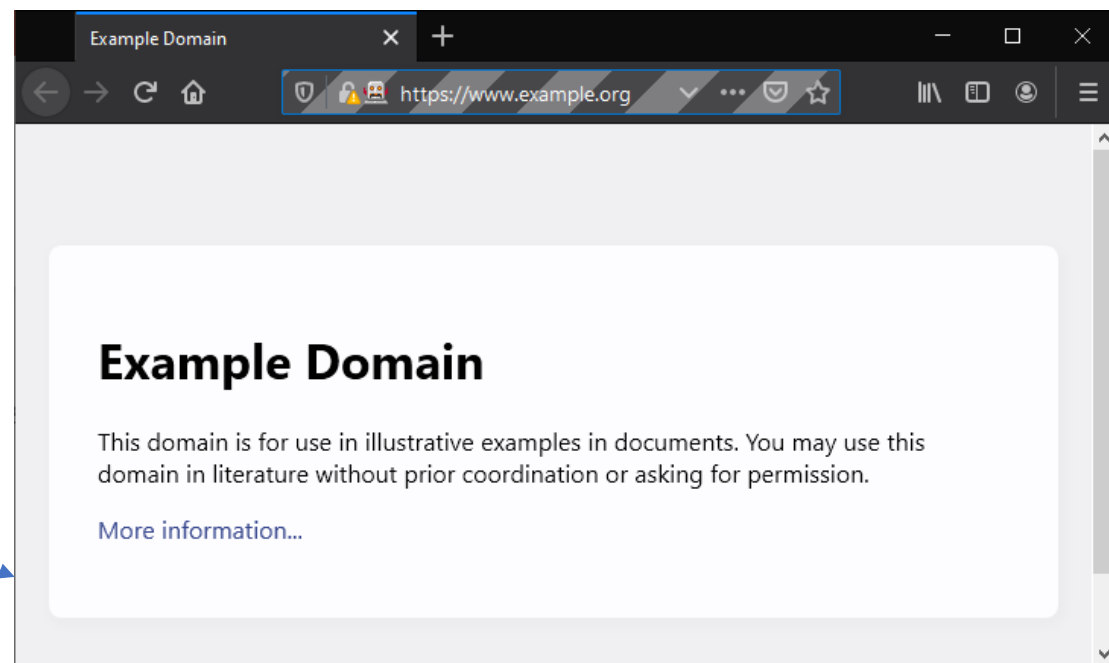
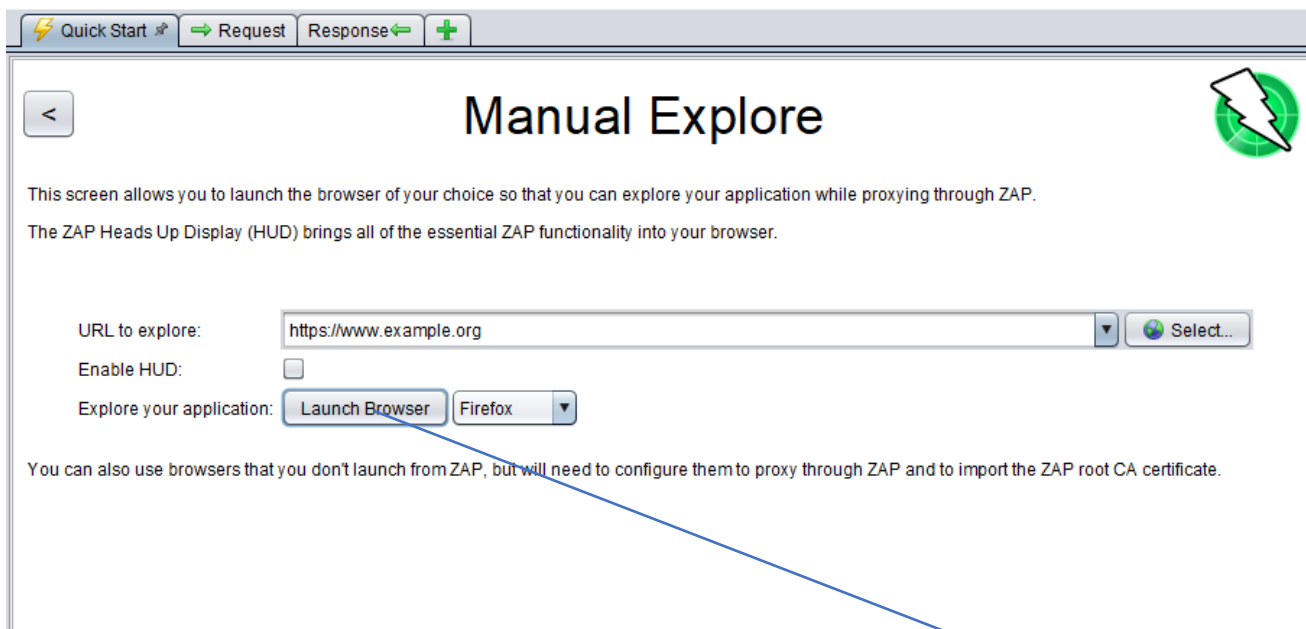
gerado pelo ZAP



original

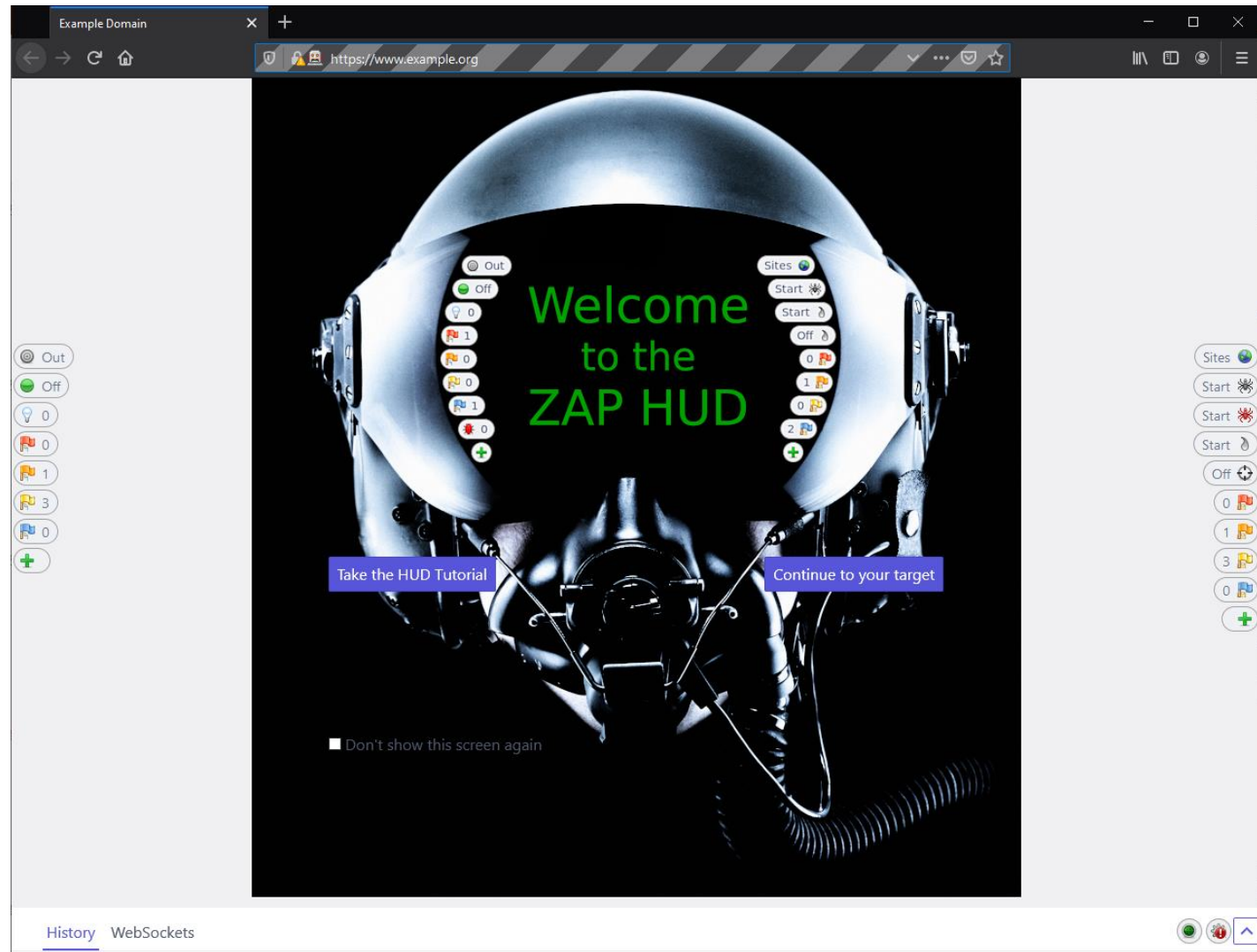
# Setup automático

- “Quick Start” possibilita o arranque do *browser* com configurações predefinidas

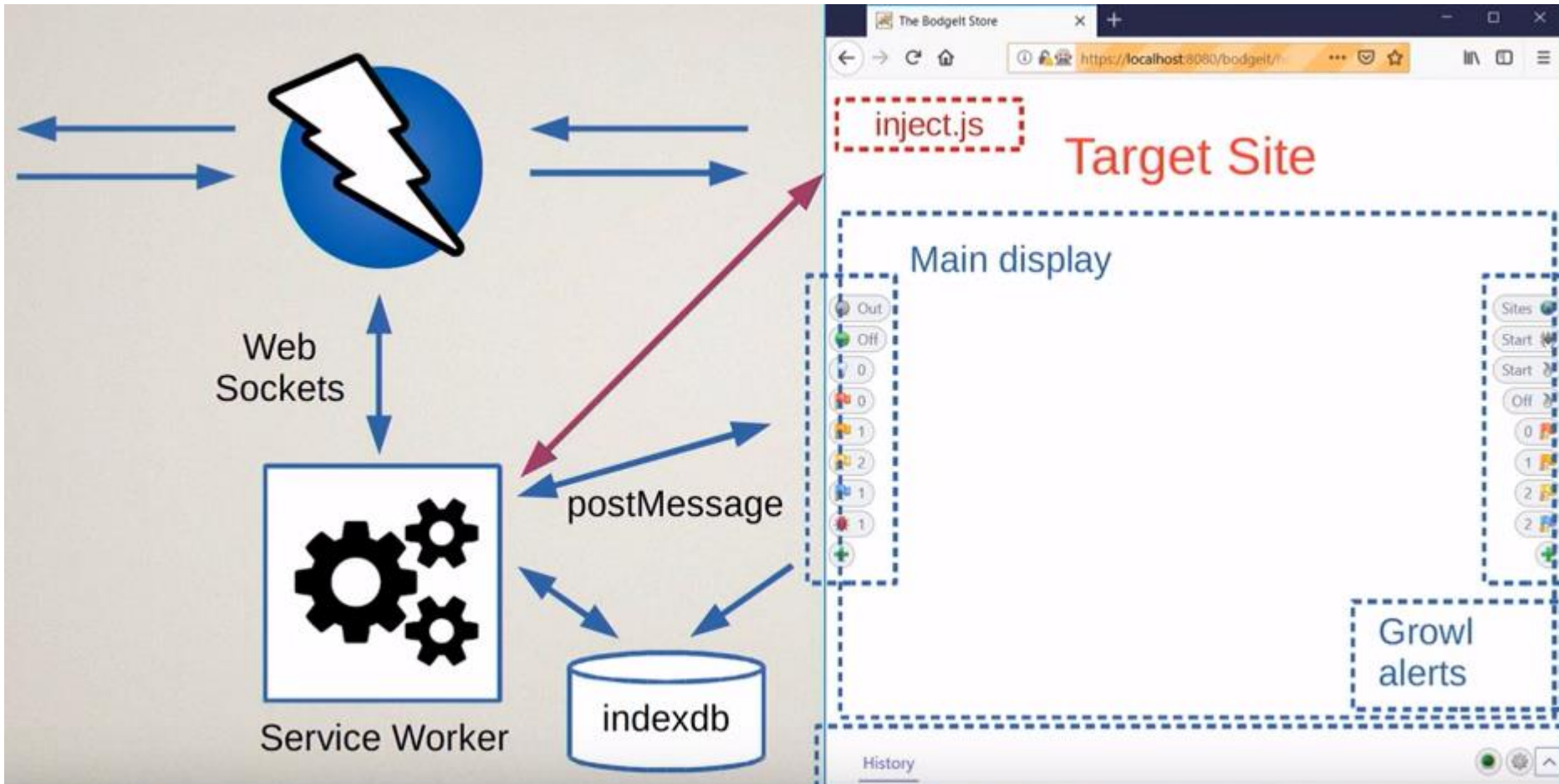


# ZAP com *Heads Up Display* (HUD)

- Interface minimalista
- Conjunto base de opções, que pode ser configurado
  - Alertas de página
  - Alertas de site
  - *Crawler*
  - Varrimento de vulnerabilidades
  - Histórico de interações
  - ...



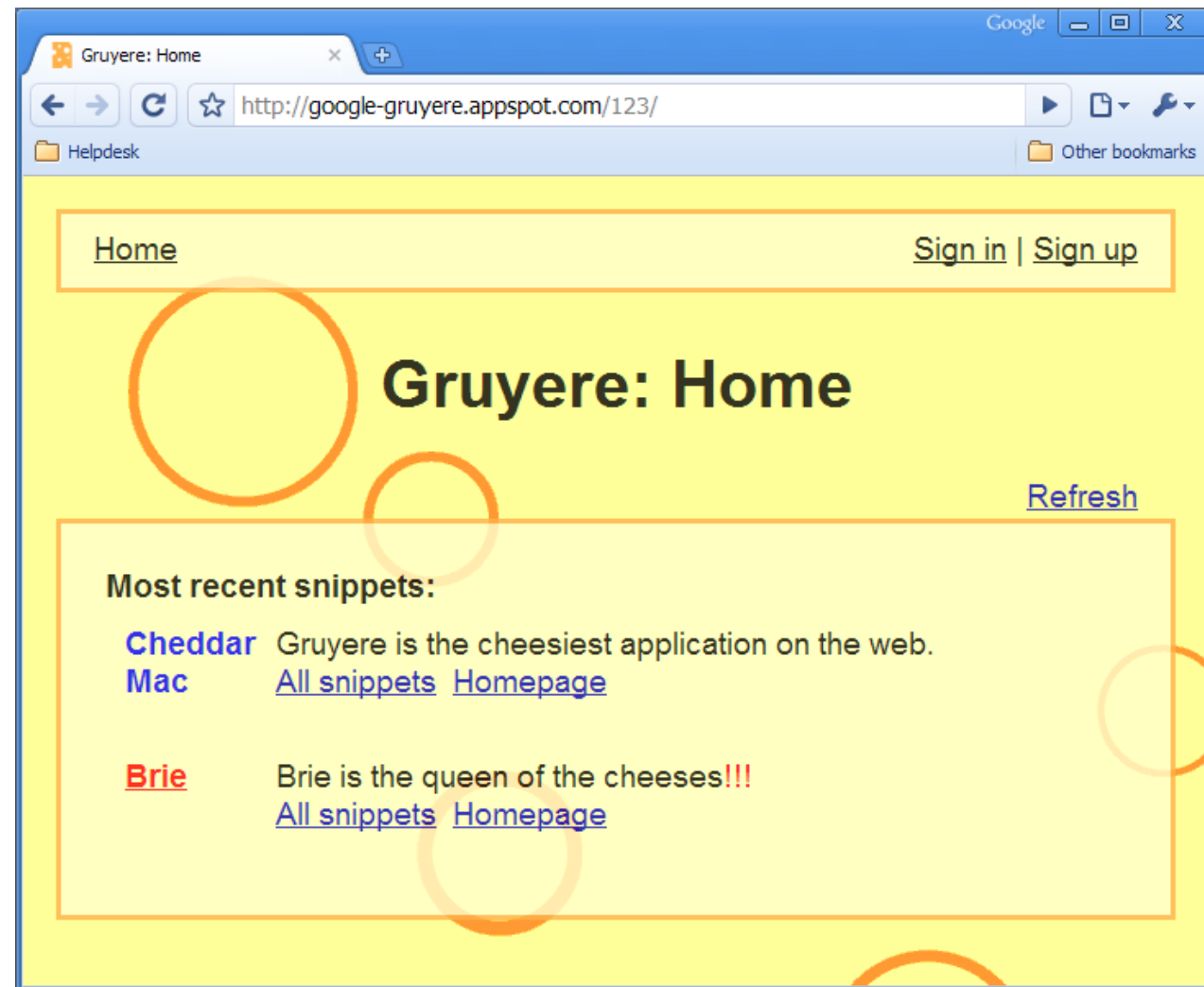
# Detalhes do HUD



[https://www.youtube.com/watch?v=1hbKGDgx\\_p0](https://www.youtube.com/watch?v=1hbKGDgx_p0)

# Google Gruyere

- Web Application Exploits and Defenses
- <https://google-gruyere.appspot.com/>
- Aplicação escrita em python, propositadamente com várias falhas de segurança



# Ataques

- Modificar pedidos em trânsito
  - Contas com mais de 16 letras no username
- XSS (armazenado)
  - <https://google-gruyere.appspot.com/...../newsnippet.gtl>
    - Experimentar inserir o seguinte texto nos *snippets*
    - `<a onmouseover="alert(1)" href="#">read this!</a>`
- Obter password de administrador (id *administrator*) com *fuzzing*
  - Vetor de fuzzing à medida com Top500 passwords
  - <https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/500-worst-passwords.txt>
  - <https://raw.githubusercontent.com/danielmiessler/SecLists/master/Username/top-1000-usernames.txt>

# Fuzzing para login de “administrator”

