




# Vulnerabilidades na Web e em Bases de Dados


Parte II: Vulnerabilidades

Segurança de Software  
2019  
Nuno Santos



## Aula passada

- ▶ Estudámos mais duas classes de vulnerabilidades:



- ▶ Devido a corridas
- ▶ Devido a validação de entradas

▶ 2 SS - Nuno Santos 2019



## E a Web?

- ▶ “Hundreds of thousands of Facebook users were hit (...) by a trick that spreads a clickjacking worm once the victim has been fooled into “liking” a page. Once that is done the action installs a Trojan and recommends the page to the victim’s friends.”
- ▶ Ellen Messmer, ‘Likejacking’ exploit fools Facebook users and friends, NetworkWorld.com, June 1<sup>st</sup> 2010



## Na web existem muitas vulnerabilidades

- ▶ A web sofre imensamente das 3 fontes de problemas:
  - ▶ complexidade, extensibilidade, conectividade
- ▶ Não uma tecnologia, mas um “monte” de tecnologias
  - ▶ HTTP, HTTPS, HTML, XML, PHP, Java, ASP.NET, PHP, cookies, SQL, web services, Flash, Silverlight, frameworks,...
- ▶ Muitas vulnerabilidades reportadas e abusadas
  - ▶ OWASP Project Top 10 Vulnerabilities
    - ▶ The Open Web Application Security Project (OWASP)
    - ▶ <https://www.owasp.org>





## Plano para esta aula

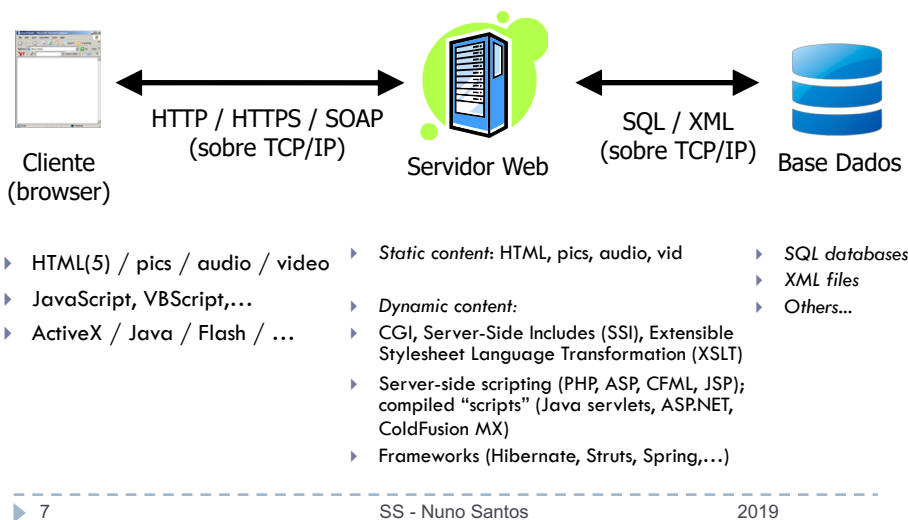
- ▶ Arquitectura das aplicações Web
- ▶ Principais vulnerabilidades em aplicações Web



## Arquitectura das aplicações Web



## Aplicação Web: Um sistema distribuído



## Protocolo HTTP / HTTPS

- Dois tipos de mensagens HTTP: **request, response**

- HTTP request message:**

- ASCII (human-readable format)

request line (GET, POST, ... commands)

linhas de cabeçalho

```
GET /index.html HTTP/1.1\r\nHost: www-net.cs.umass.edu\r\nUser-Agent: Firefox/3.6.10\r\nAccept-Language: en-us,en;q=0.5\r\nConnection: keep-alive\r\n\r\n
```

carriage return character  
line-feed character

carriage return, line feed no início indica o final das linhas de cabeçalho

Corpo opcional vem aqui (sem corpo em pedidos GET)



## Interface de input (entradas) numa aplicação web

- ▶ Uma página geralmente inclui um *form* (formulário) para entradas pelo utilizador; como é que é enviado para o servidor?

A diagram of a web form. It consists of a rectangular text input field. Below the input field are two buttons: 'Google Search' and 'I'm Feeling Lucky'. A blue arrow points from the text 'como é que é enviado para o servidor?' in the list above to the input field.

### Usando o método POST:

- ▶ O input é carregado para o servidor no corpo do pedido HTTP

### No próprio URL:

- ▶ Usa o método GET
- ▶ O input é enviado no campo URL da linha de pedido:  
`www.somesite.com/animalsearch.php?animal=monkey&food=banana`



## Manutenção do estado numa aplicação web

- ▶ O modelo base é *stateless* (sem estado)...
  - ▶ ...mas as aplicações Web não triviais precisam manter estado
- ▶ Exemplos de informação de estado:
  - ▶ Usuário está logado? Qual a conta do usuário?...
- ▶ State tracking:
  - ▶ **Cookie:** o servidor envia um ID ao cliente que tem que incluir em cada pedido (servidor guarda estado para cada ID)
  - ▶ Na prática isto não é tão simples e tem gerado muitas vulnerabilidades

## Principais vulnerabilidades em aplicações web

11

SS - Nuno Santos

2019



## Vulnerabilidades no top 10

| OWASP Top 10 – 2007 (Previous)                         | OWASP Top 10 – 2010 (New)                         | OWASP Top 10 – 2013 (New)                         |
|--|---|---|
| A2 – Injection Flaws                                   | A1 – Injection                                    | A1 – Injection                                    |
| A1 – Cross Site Scripting (XSS)                        | A2 – Cross Site Scripting (XSS)                   | A2 – Broken Authentication and Session Management |
| A7 – Broken Authentication and Session Management      | A3 – Broken Authentication and Session Management | A3 – Cross-Site Scripting (XSS)                   |
| A4 – Insecure Direct Object Reference                  | A4 – Insecure Direct Object References            | A4 – Insecure Direct Object References            |
| A5 – Cross Site Request Forgery (CSRF)                 | A5 – Cross Site Request Forgery (CSRF)            | A5 – Security Misconfiguration                    |
| <was T10 2004 A10 – Insecure Configuration Management> | A6 – Security Misconfiguration (NEW)              | A6 – Sensitive Data Exposure                      |
| A10 – Failure to Restrict URL Access                   | A7 – Failure to Restrict URL Access               | A7 – Missing Function Level Access Control        |
| <not in T10 2007>                                      | A8 – Unvalidated Redirects and Forwards (NEW)     | A8 – Cross-Site Request Forgery (CSRF)            |
| A8 – Insecure Cryptographic Storage                    | A9 – Insecure Cryptographic Storage               | A9 – Using Known Vulnerable Components            |
| A9 – Insecure Communications                           | A10 – Insufficient Transport Layer Protection     | A10 – Unvalidated Redirects and Forwards          |
| A3 – Malicious File Execution                          | <dropped from T10 2010>                           |   |
| A6 – Information Leakage and Improper Error Handling   | <dropped from T10 2010>                           |   |

- ▶ Top 10 2013 semelhante a 2010's
- ▶ Outras classificações: Web Application Security Consortium (WASC) mas têm menos granularidade

▶ 12

SS - Nuno Santos

2019



## A1 – Falhas por injeção

- ▶ Vários tipos
  - ▶ SQL Injection (a mais frequente)
  - ▶ Outras: XML, LDAP, XPath, HTML, OS command injection, etc
- ▶ Ideia principal: o servidor web aceita input (entradas) que é mal interpretado por um interpretador
  - ▶ Exemplos de interpretadores: DBMS, XML, LDAP,...

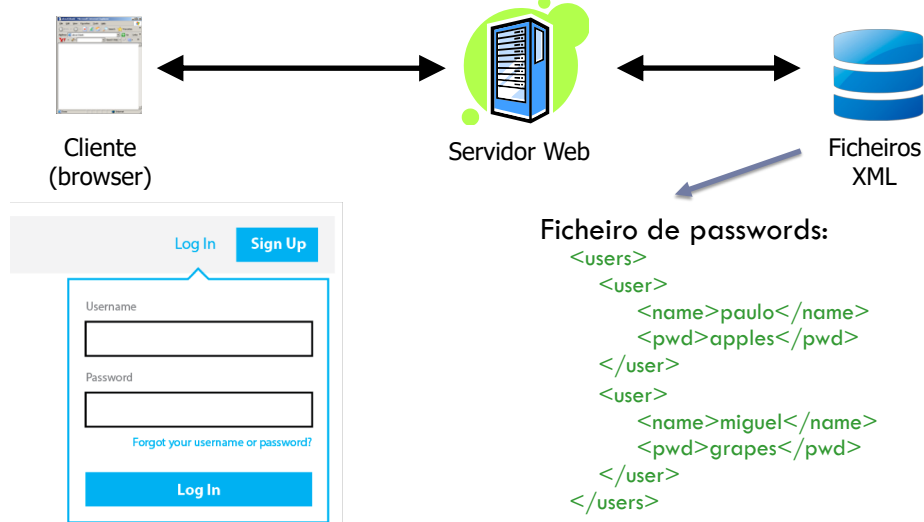
▶ 13

SS - Nuno Santos

2019



## Exemplo de injeção de XML



▶ 14

SS - Nuno Santos

2019



## Um ataque de injeção de XML...

- ▶ O que acontece se um utilizador malicioso mudar a password para a password seguinte?

oranges</pwd></user><user><name>pirate</name><pwd>potatoes

- ▶ Ficheiro antes

```
<users>
  <user>
    <name>paulo</name>
    <pwd>apples</pwd>
  </user>
  <user>
    <name>miguel</name>
    <pwd>grapes</pwd>
  </user>
</users>
```



```
<users>
  <user>
    <name>paulo</name>
    <pwd>apples</pwd>
  </user>
  <user>
    <name>miguel</name>
    <pwd>grapes</pwd>
  </user>
  <user>
    <name>alice</name>
    <pwd> oranges </pwd>
  </user>
  <user>
    <name>pirate</name>
    <pwd>potatoes</pwd>
  </user>
</users>
```

▶ 15

SS - Nuno Santos

2019



## SQL injection

- ▶ Causas destas vulnerabilidades

- ▶ Input do usuário injectado no meio de comandos SQL +
- ▶ SQL usa vários meta-caracteres / metadados

- ▶ Exemplo (PHP/MySQL):

```
$username = $_HTTP_POST_VARS['username'];
$password = $_HTTP_POST_VARS['password'];
$query = "SELECT * FROM logintable WHERE user = '" .
    $username . "' AND pass = '" . $password . "'";
$result = mysql_query($query);
if(!$result) die_bad_login();
```

metadata

username: root  
password: root' OR pass <> 'root  
Query: SELECT \* FROM logintable  
WHERE user = 'root' AND pass =  
'root' OR pass <> 'root'





## A2 – Cross Site Scripting (XSS)

- ▶ Vulnerabilidade muito disseminada e perigosa
- ▶ Permite a um atacante executar um *script* (um programa) no browser da vítima
  - ▶ Linguagem de scripting é geralmente o **JavaScript (JS)**
  - ▶ Mas é possível com outras linguagens
- ▶ Cross-site scripting (XSS)
  - ▶ O utilizador confia no site vulnerável
  - ▶ A ideia é convencê-lo a confiar em dados maliciosos enviados pelo servidor
  - ▶ Bom para obter cookies ou as credenciais (username / password)

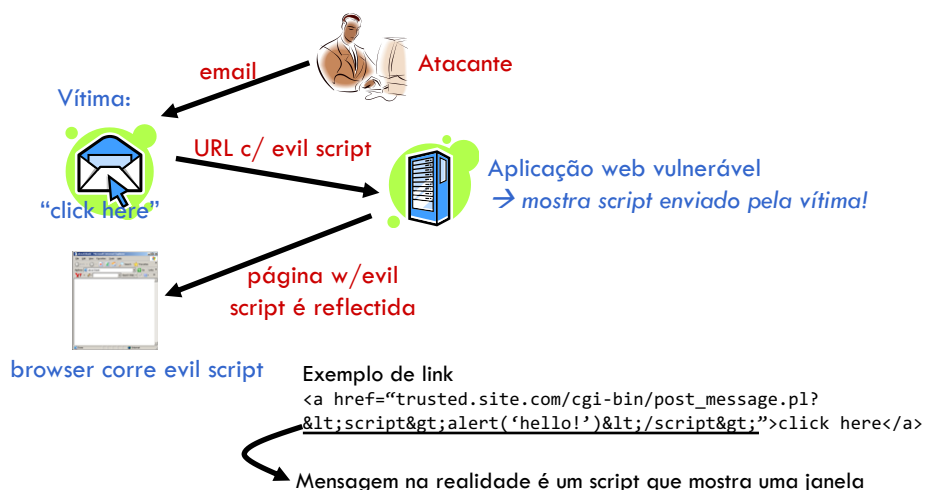
▶ 17

SS - Nuno Santos

2019



## Reflected XSS



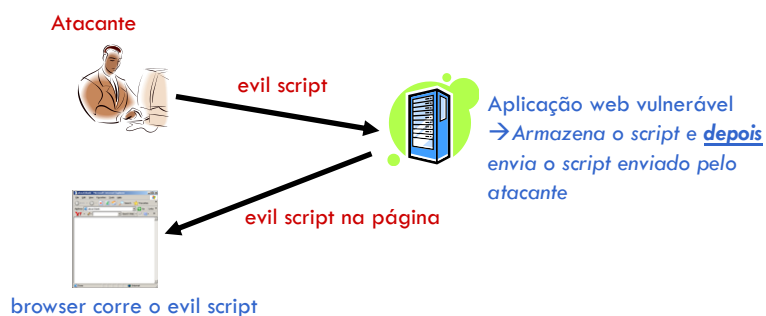
▶ 18

SS - Nuno Santos

2019



## Stored XSS



- ▶ Scripts podem ser semelhantes ao anterior

▶ 19

SS - Nuno Santos

2019



## Outras vulnerabilidades

- ▶ A3 – Quebra de autenticação e de sessões
  - ▶ Ex: session hijacking – atacante descobre um ID de sessão aberta e envia commands para essa sessão
- ▶ A4 – Referência directa de objectos
  - ▶ O site expõe a referência para uma implementação interna de um objecto sem controlo de acesso apropriado, ex., fotografias acessíveis directamente por URL, etc.
- ▶ A5 – Cross Site Request Forgery (CSRF)
  - ▶ Força o utilizador a executar acções não desejadas num site no qual a pessoa se encontra autenticada

▶ 20

SS - Nuno Santos

2019



## Conclusões

- ▶ As aplicações Web reúnem todos os ingredientes para sofrerem vulnerabilidades
- ▶ Essas vulnerabilidades são muito frequentes, muito abusadas actualmente por atacantes, e com consequências de grande gravidade
- ▶ Desses ataques, são particularmente comuns e graves, os ataques por injeção de código e XSS

▶ 21

SS - Nuno Santos

2019



## Referências e próxima aula

- ▶ **Bibliografia**
  - ▶ [Correia17] Capítulos 8 e 9
- ▶ **Próxima aula**
  - ▶ Auditoria e teste de software

▶ 22

SS - Nuno Santos

2019