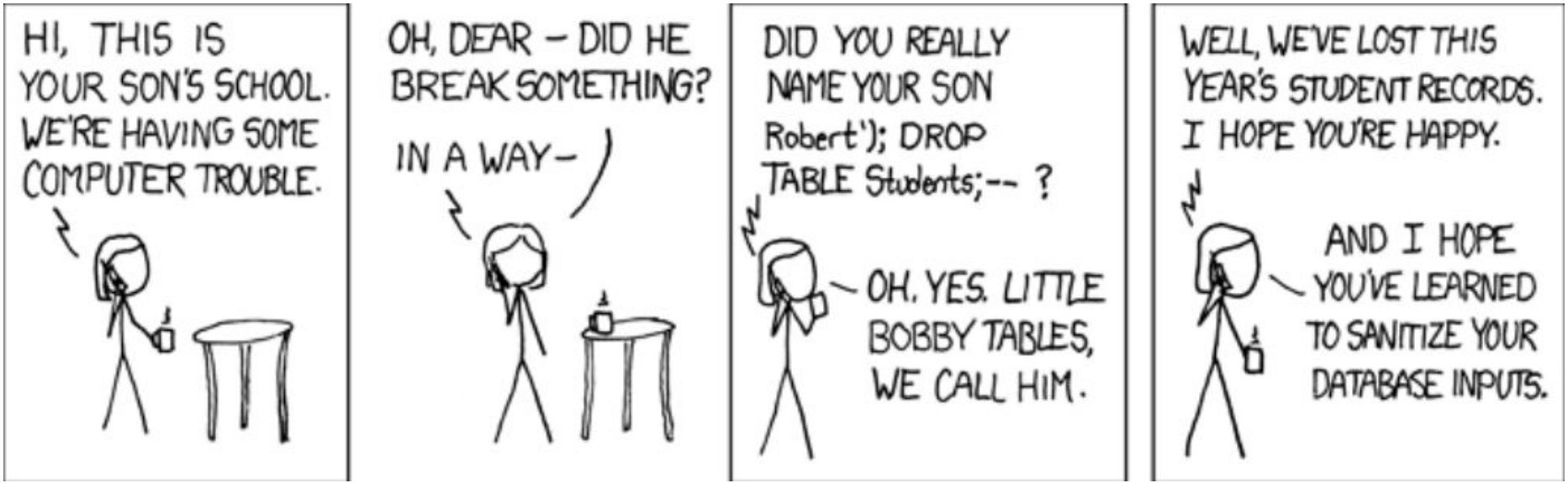
INE5429-07208
Segurança em Computação

Principios de Segurança

Prof. Jean Everson Martina


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

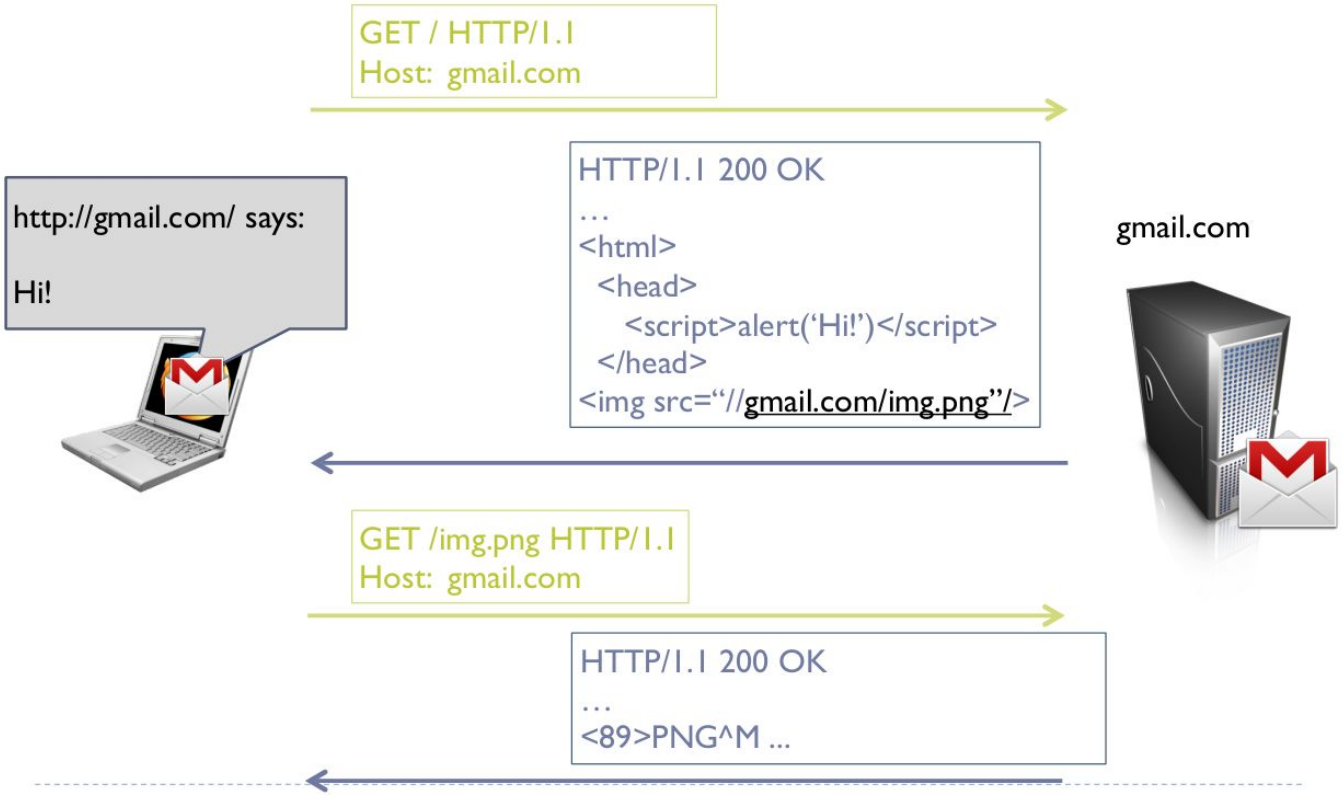
Cookies

- Usados para compensar a falta de manutenção de estado no HTTP
- Enviado pelo Servidor e mencionado pelo cliente para manter sessão
- Sent by server:
  - <html><head>...
  - Set-Cookie:SessionID=9551781512random680541...</head>
  - <body>...
- Sent by client:
  - <head>...
  - Cookie:SessionID=9551781512random680541

Tracking Cookies

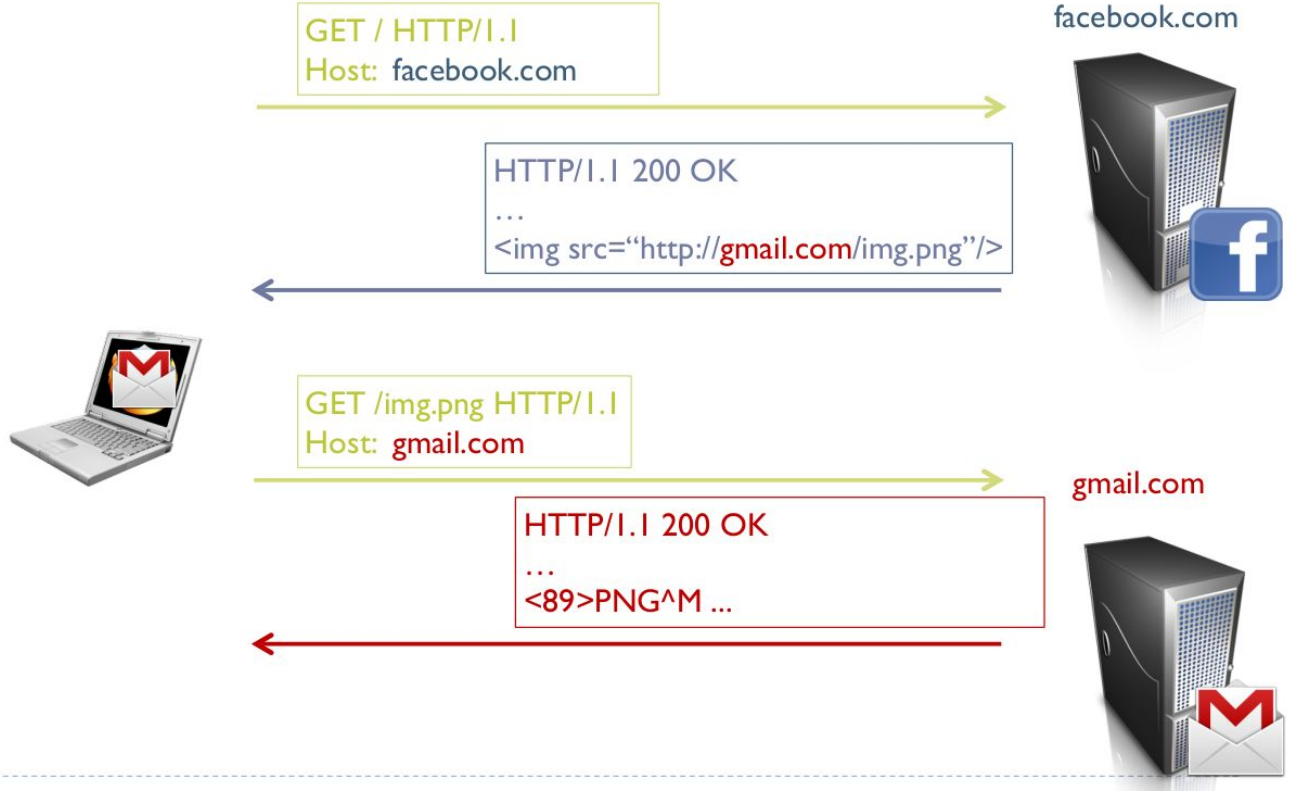
- Um site inclui um componente de outro site e este site envia um cookie
- Por exemplo a UFSC coloca uma imagem externa, ou link social
  - O servidor responde com a uma imagem e um cookie
  - Se outro servidor colocar um link para uma imagem externa no mesmo servidor que a UFSC apontou, o site que hospeda imagem pode ir seguindo onde a pessoa vai e traça o seu perfil
  - Imaginem o que acontece quando voce acessa sites com o google ou com o facebook logado.

Revisão de HTTP

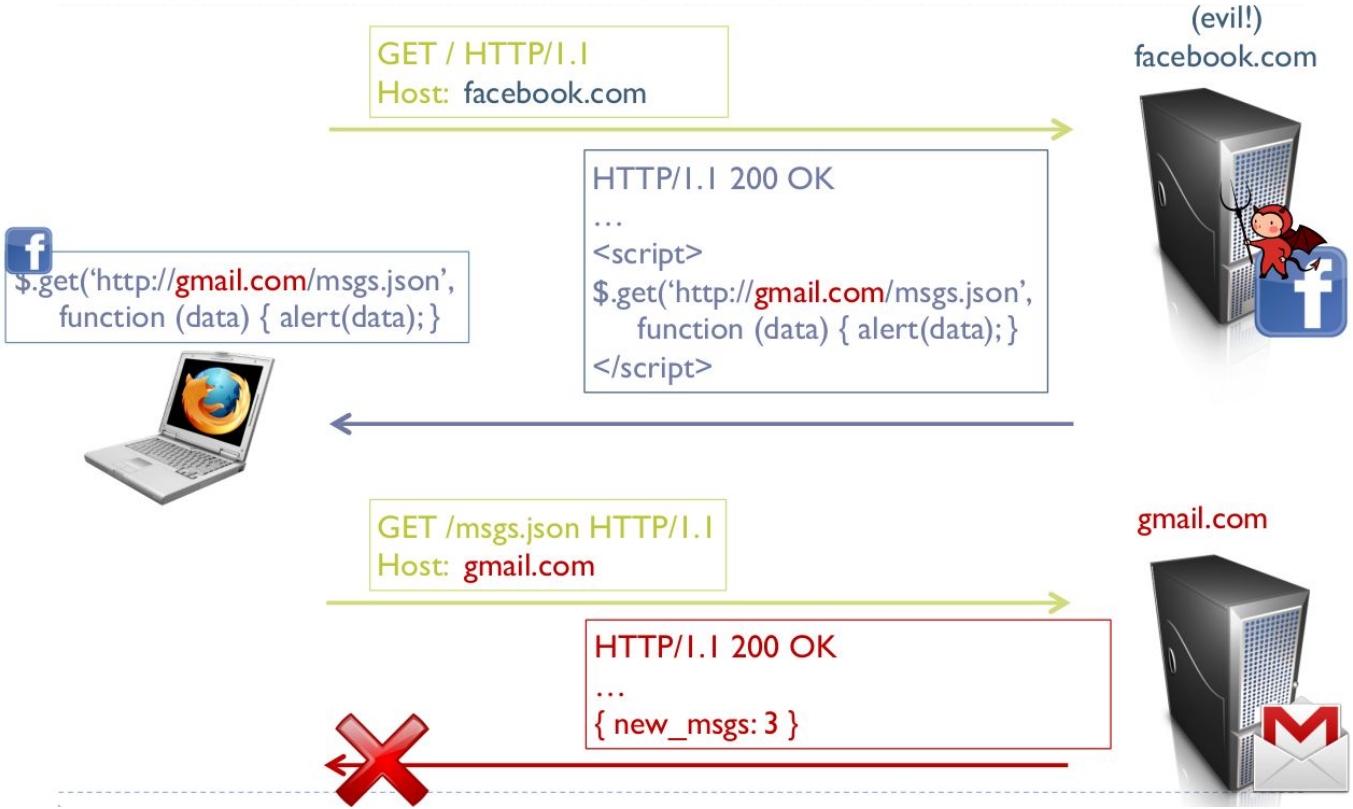


Same-Origin Policy (SOP)

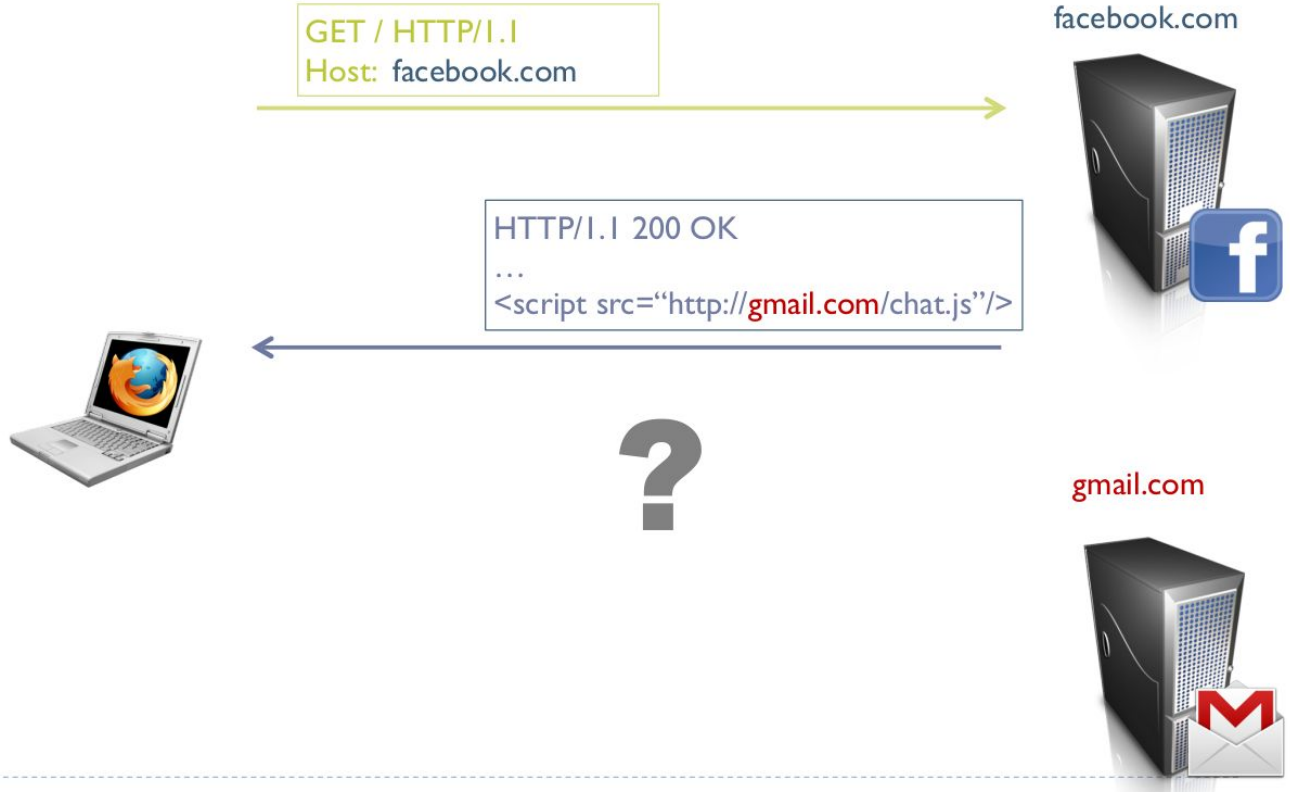
Same-Origin Policy (SOP)



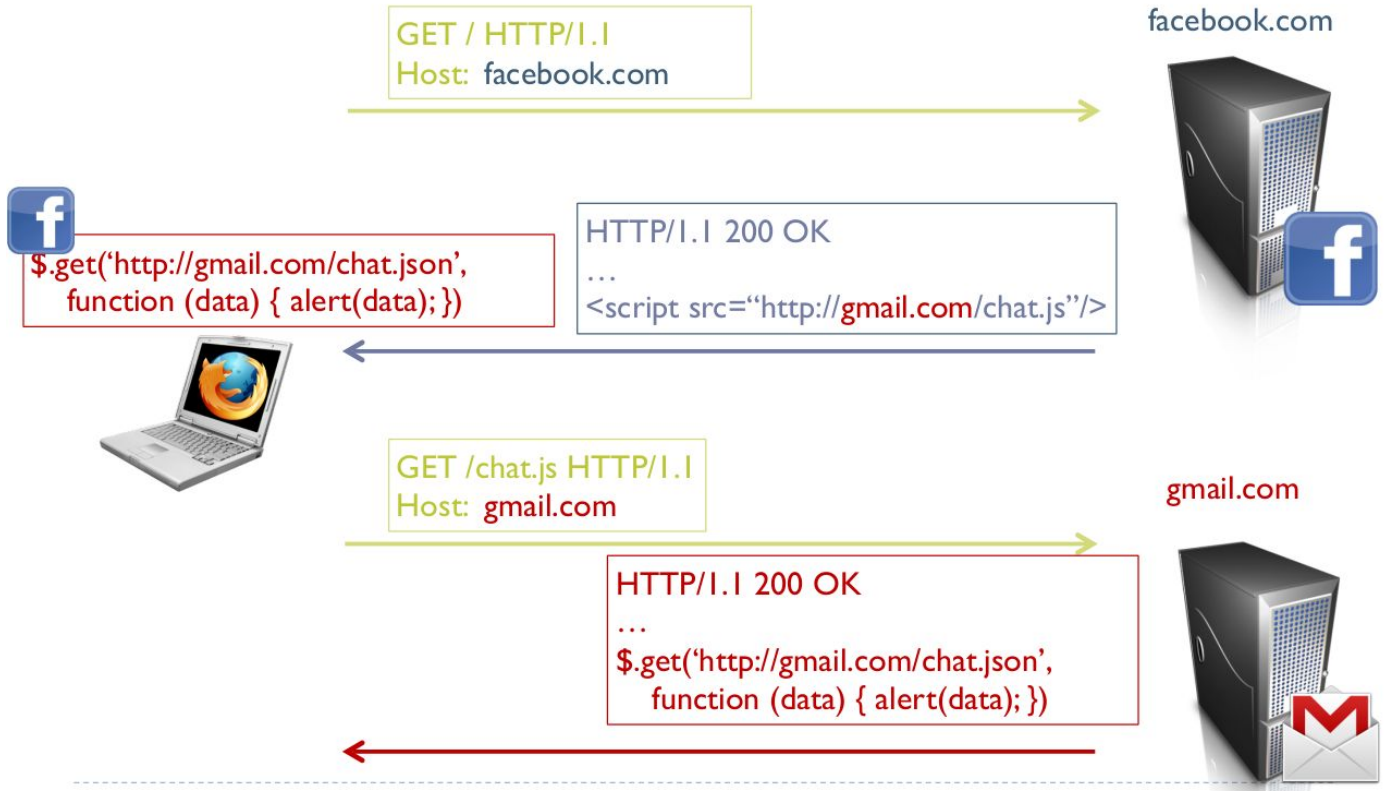
Same-Origin Policy (SOP)



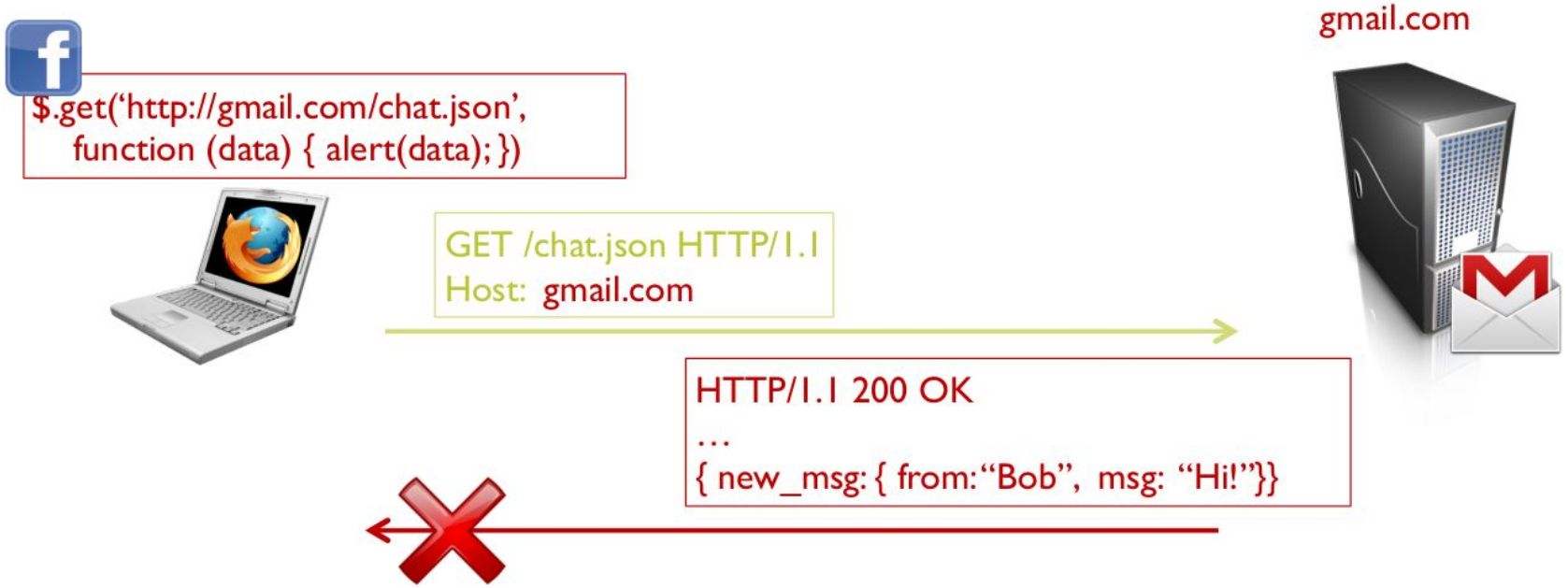
Same-Origin Policy (SOP)



Same-Origin Policy (SOP)

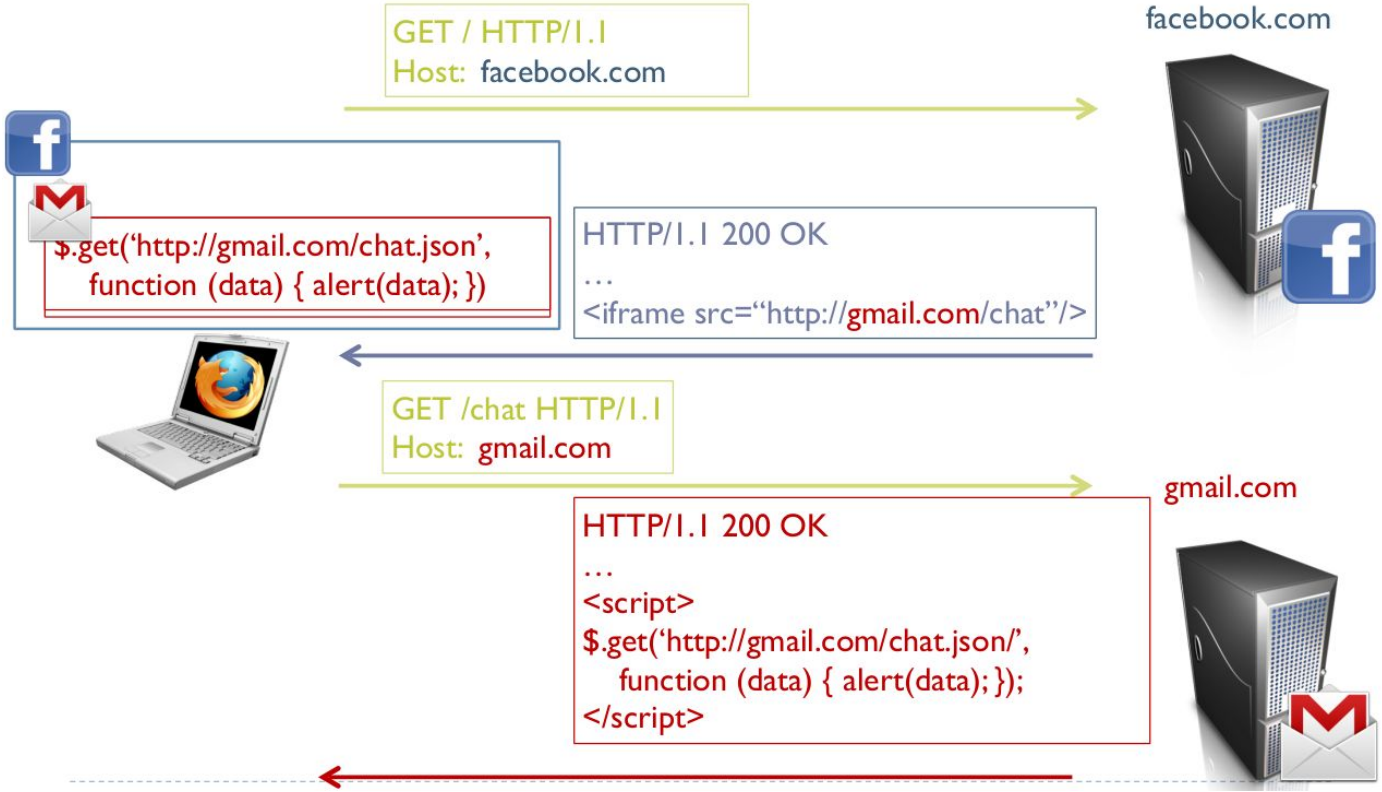


Same-Origin Policy (SOP)

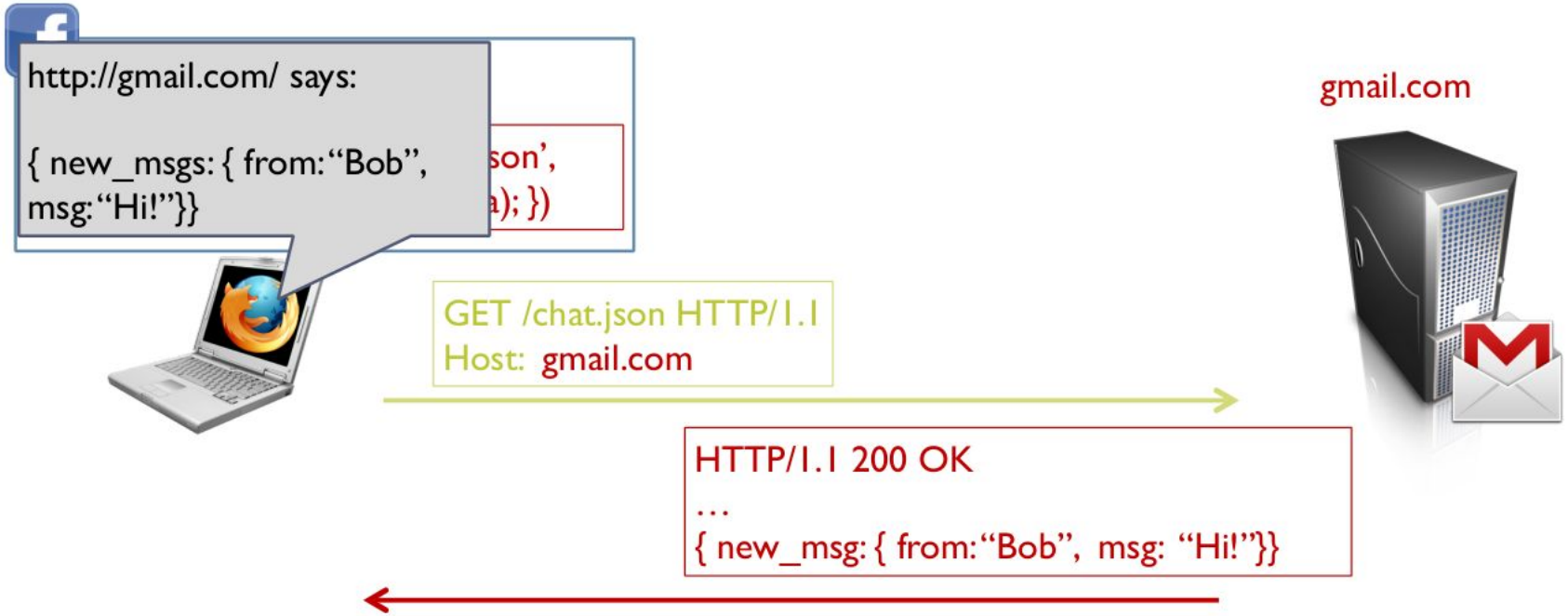


Same-Origin Policy (SOP)

Same-Origin Policy (SOP)

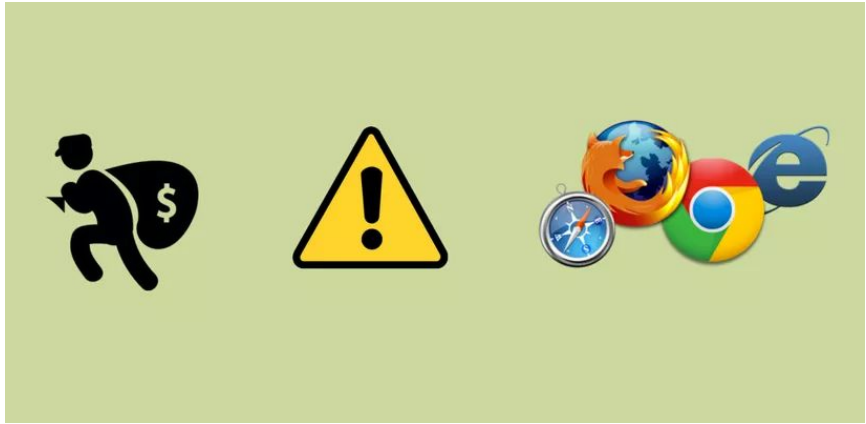


Same-Origin Policy (SOP)



Cross Site Request Forgery

- Como fazer uma transferência de banco de uma pessoa:
  - Sabemos que a URL é:
    - http://bank.com/transfer.do?acct=MARIA&amount=100000
  - Podemos tentar:
    - Porn!
  - Ou isso:
    - <img src="http://bank.com/transfer.do?acct=MARIA&amount=100000" width="0" height="0"border="0">

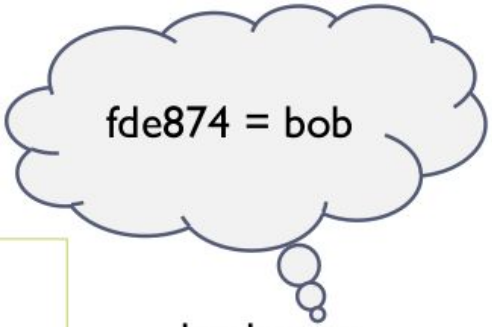


Cross-site Request Forgery (CSRF)

Cross-site Request Forgery (CSRF)

Cross-site Request Forgery (CSRF)

Click me!!!
http://bank.com/transfer?to=badguy&amt=100



GET /transfer?to=badguy&amt=100 HTTP/1.1
Host: bank.com
Cookie: login=fde874

HTTP/1.1 200 OK
….
Transfer complete: -$100.00

Cross-site Request Forgery (CSRF) - Defesas

- Precisamos autenticar que cada ação do usuário origina do nosso site
- Uma maneira: cada ação recebe um token associado a ela
  - Em uma nova ação (página), verifique se o token está presente e correto
  - O atacante não consegue encontrar um token para outro usuário e, portanto, não pode fazer ações em nome do usuário

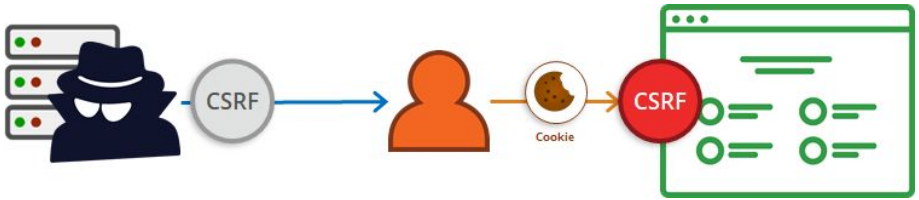
Cross-site Request Forgery (CSRF) - Defesas

<a …>Pay $25 to Joe:
http://bank.com/transfer?to=joe&amt=25&token=8d64

Cross-site Request Forgery (CSRF) - Defesas

- Formulário gerado dinamicamente:

```
<form action="/transfer.do" method="POST">

<input type="text" name="recipient"/>

<input type="number" name="amount"/>

<input type="text" name="TXID" value="8d64"/>

<input type="submit" value="Transfer money"/>

</form>
```



- Cookie: fde874
- POST bank.com/transfer.do?recipient=joe&amount=25&TXID=8d64

Injeção de Código

- Código mal escrito e não sanitizado:
  - <?php system("/bin/ls " .$_GET['USER_INPUT']);?>
- Possíveis entradas
  - ;comando_malicioso
  - | comando_malicioso
  - `comando_malicioso`
- Isso executa:
  - system("/bin/ls ; comando_malicioso");

Injeção de Código

foo:

```
<?php

echo system(“ls ” . $_GET[“path”]);
```

Injeção de Código

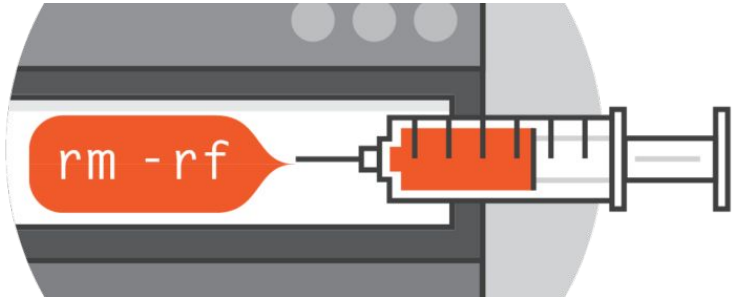
```
<?php

echo system(“ls ” . $_GET[“path”]);
```

GET /?path=$(rm –rf /) HTTP/1.1


```
<?php

echo system(“ls $(rm -rf /)”);
```

Injeção de Código - Defesas

- Cusado por confusão entre dados e código
  - O servidor pensou que o usuário iria prover dados e ele mandou código
- São as mais comuns e perigosas classes de vulnerabilidades:
  - Shell Injection - Acabamos de ver!
  - SQL Injection
  - Cross-Site Scripting (XSS)



Injeção SQL

- SQL é uma linguagem para perguntar (query) um banco de dados:
  - Quantas pessoas vivem em Florianopolis?
    - SELECT COUNT(*) FROM users WHERE location = Florianopolis
  - Existe um usuários com o nome bob e a senha abc123?
    - SELECT * FROM users WHERE username=bob and password=abc123
  - Destrua tudo que você sabe sobre usuários!
    - DROP TABLE users

Injeção SQL

- Considere um SQL onde o usuário pode escolher a cidade ($city):
  - SELECT * FROM users WHERE location=“$city“
- O que um atacante pode fazer?
  - $city ← Florianopolis; DELETE FROM users WHERE 1=1
- A consulta SQL vira:
  - SELECT * FROM users WHERE location=Florianopolis; DELETE FROM users WHERE 1=1

Injeção SQL

Injeção SQL - Defesas

- Tenha certeza que os dados são interpretados como dados!
- Básico: Trate caracteres de controle:

```
function sanitize($str)
{
   str_replace(
      array('\\', "\0", "\n", "\r", "'", '"', “;"),
      array('\\\\', '\\0', '\\n', '\\r', "\\'", '\\"', ‘\\;'),
      $str);
}
```

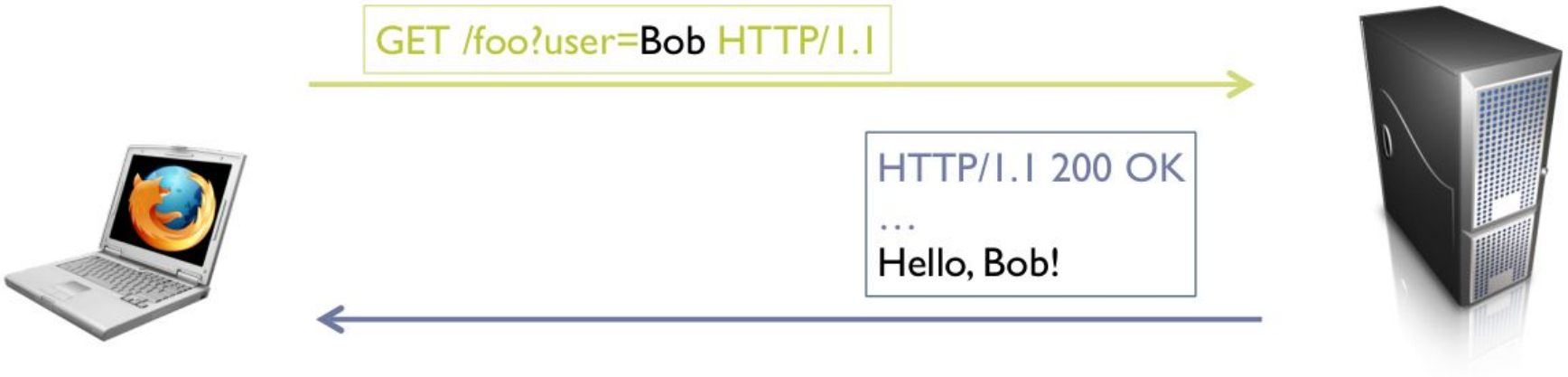
- Declare dados como dados:

```
$pstmt = $db->prepare(
    "SELECT * FROM users WHERE location=?");
$pstmt->execute(array($city));  // Data
```

Cross-Site Scripting (XSS)

foo:

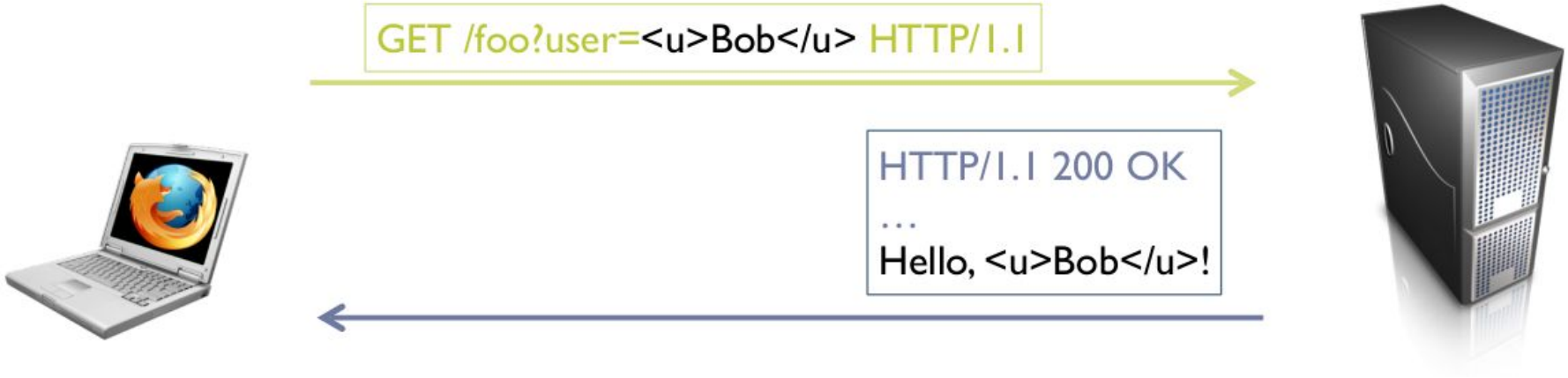
```
<?php

echo “Hello, ” . $_GET[“user”] . “!”;
```



Cross-Site Scripting (XSS)

foo:

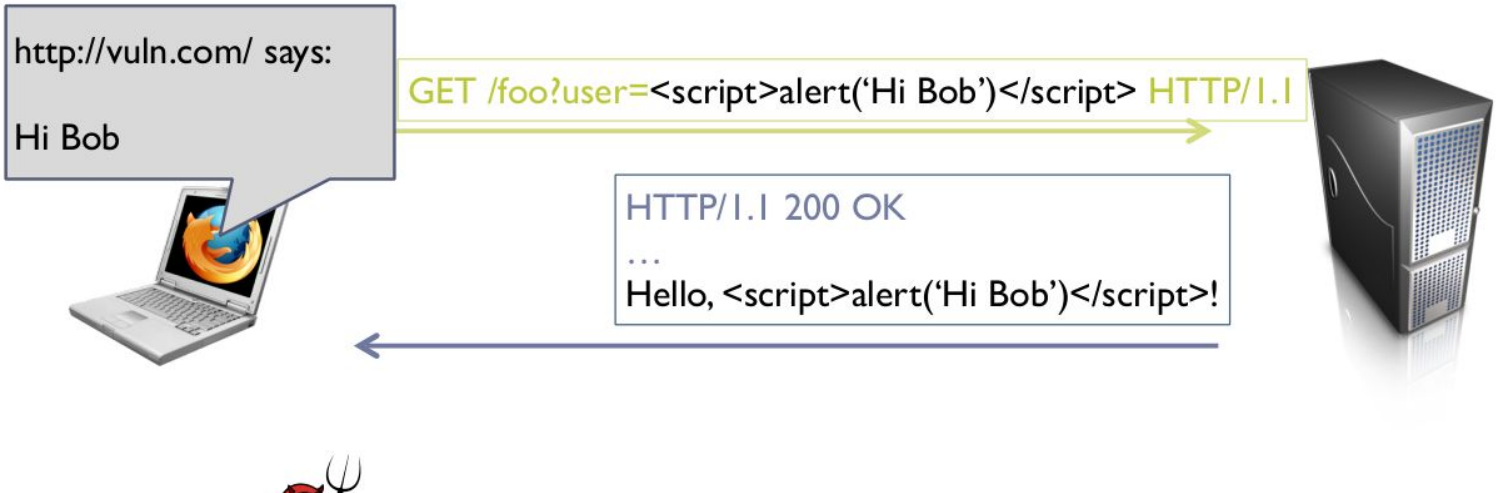
```
<?php

echo “Hello, ” . $_GET[“user”] . “!”;
```



Cross-Site Scripting (XSS)

foo:

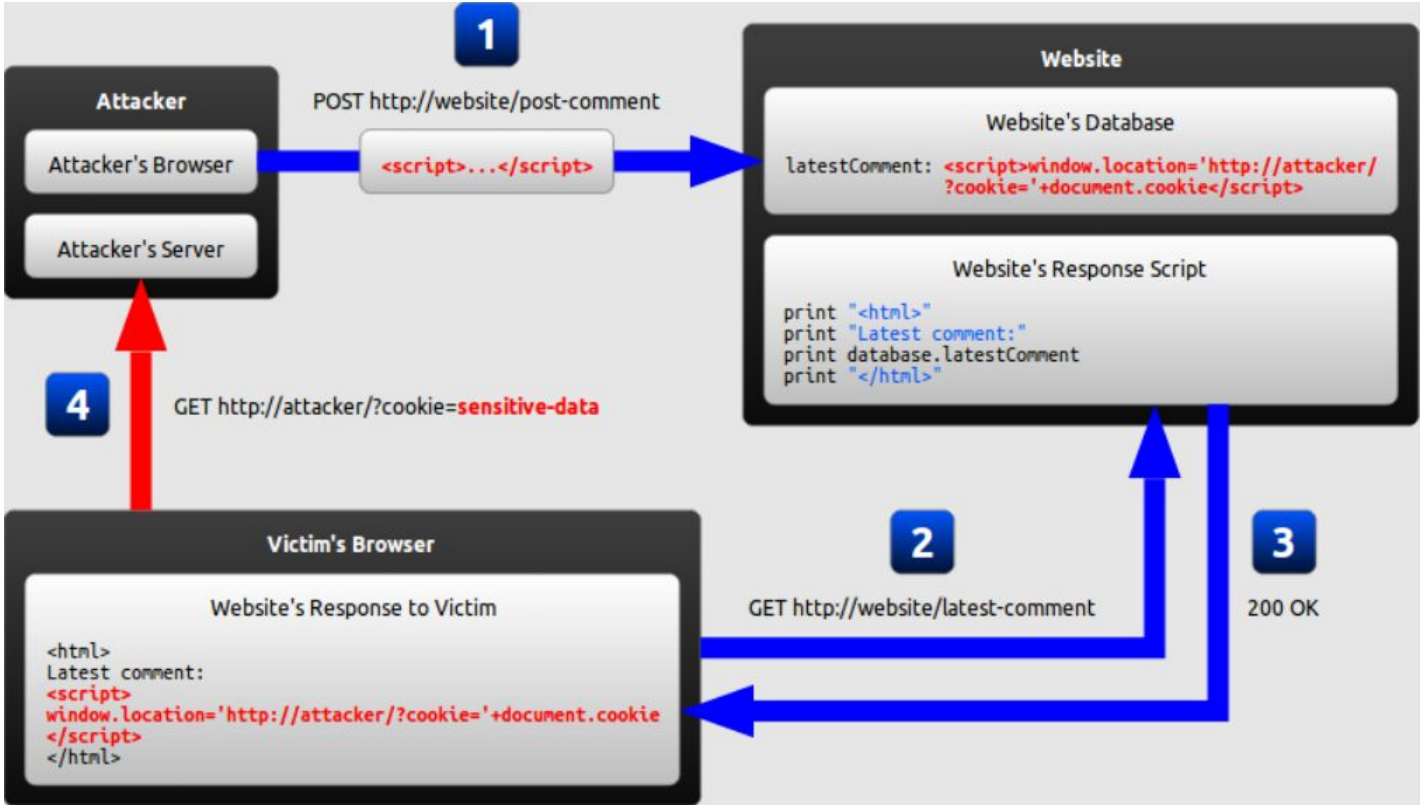
```
<?php

echo “Hello, ” . $_GET[“user”] . “!”;
```



Click me!!!

http://vuln.com/foo?user=<script>alert(‘XSS’)</script>

Exemplo Cross-Site Scripting (XSS)

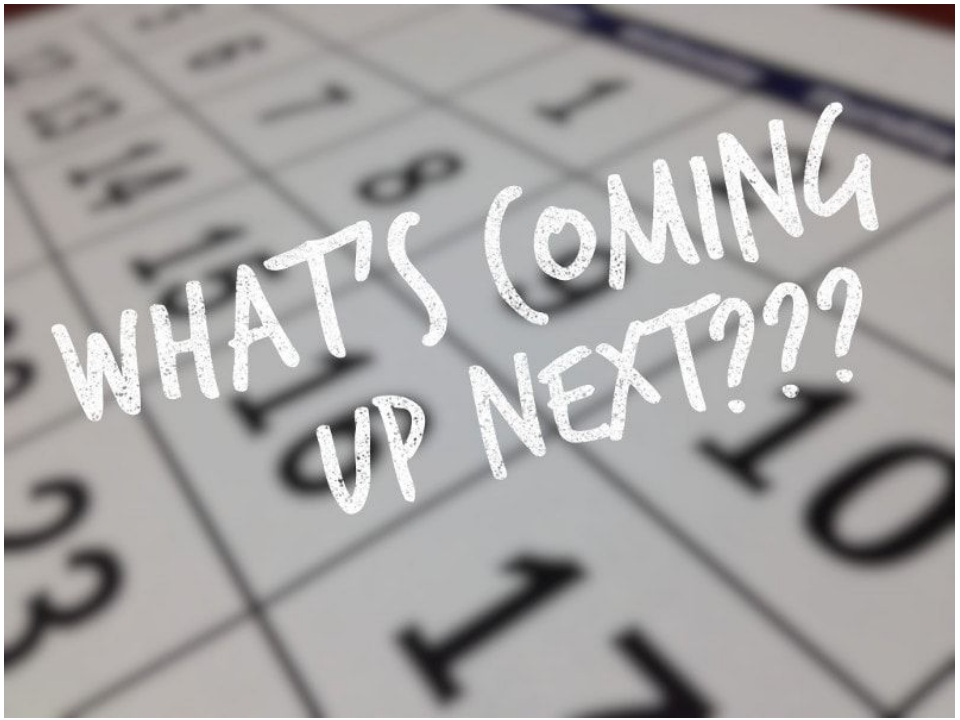


Cross-Site Scripting (XSS) - Defesas

- Garanta que dados são dado e não código!
  - Trate ou rejeite caracteres especiais.
  - Quais?Depende do contexto dos dados ($data)
    - Em documentos HTML:
      - $data
    - Dentro de um TAG:
      - <a href="http://site.com/$data">
    - Dentro de javascript:
      - var x = "$data";
  - Não pode deixar passar nenhum!
- Framework pode ajudar: Você declara os dados controlados pelo usuário e o framework faz o resto
- É algo bem Dificil de fazer do jeito certo:
  - https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

Próximas Aulas

- Prática:
  - Trabalho Individual IV
    - Envolve todo este conteúdo que vimos na aula de hoje e o que veremos nas próximas.
- Teórica:
  - OWASP TOP 10



Perguntas?

jean.martina@ufsc.br