

E-government vulnerabilities in 2005

*Vebjørn Moen
André N. Klingsheim
Kent Inge F. Simonsen
Kjell Jørgen Hole*

NoWires Research Group
Department of Informatics
University of Bergen

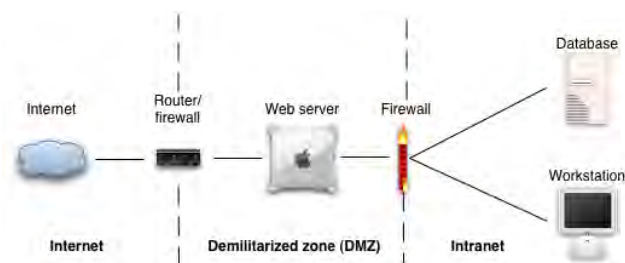
Last updated on August 18, 2011

KJhole.com

Outline

- The problem with Web-based software
- Why e-government?
- Common attacks:
 - [Cross Site Scripting \(XSS\)](#)
 - [SQL injection](#)
- Results of a worldwide investigation
- Collecting Social Security Numbers (SSNs)
- Conclusions

The big problem



- Emphasis on functionality
 - service oriented architectures
- Computer security = network security
- **Software vulnerable to attacks from crackers**

3

E-government definition

- **E-government:** any government function or process that is conducted in a digital form over the internet
- Often in the form of a Web portal that supply individuals and businesses with
 - public information
 - government forms for download
 - contact with government representatives

4

Advantages of e-governments

- Providing services over internet will yield:
 - higher efficiency and quality
 - easier access
 - possibility of individual services
 - increased transparency

Implementation vulnerabilities

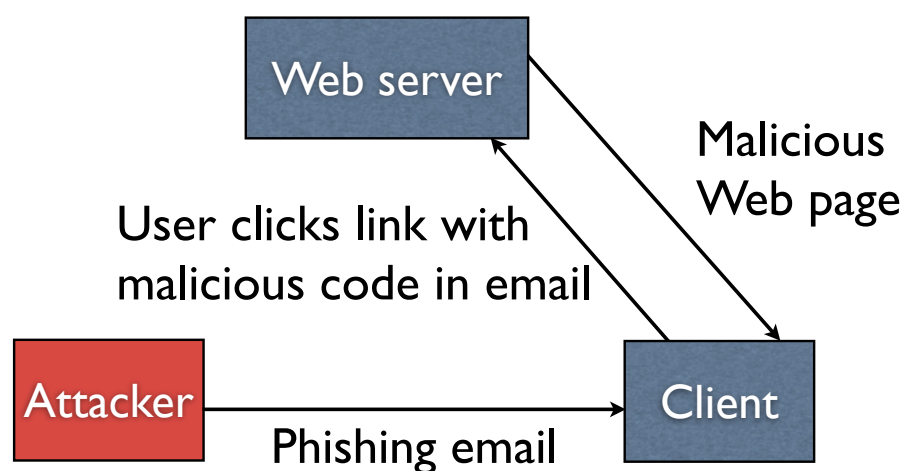
- A worldwide examination of e-governments in 2005 concluded that most implementations had vulnerabilities exploitable by:
 - [Cross Site Scripting \(XSS\)](#)
 - [SQL injection](#)

XSS

- **XSS** is an attack on the clients of a particular Web site
- an attack can lead to total breach of security when client cookies, or other sensitive information, are stolen
- an attack involves *three* parties
 1. the attacker
 2. a client
 3. a Web server

7

XSS attack scenario



8

Explanation

- The Web server reflects **unvalidated** user input on a dynamically generated Web page
- attacker supplies a JavaScript and/or HTML code in the user input, which will be part of dynamically generated Web page
- can use JavaScript to steal cookies
- Cascading Style Sheets and the HTML tag `iframe` can be utilized to control both content and layout of resulting Web page

Explanation ...

- Note that although the vulnerability exists at the Web server, at no time is the server directly harmed
- the server only generates “malicious” Web pages displayed by the client’s browser
- XSS is one of the most common Web application vulnerabilities

Example: attack string

XSS attack string which generates a page with *arbitrary* content on an XSS vulnerable site:

target script at site → <http://www.VulnerableSite.com/search.cgi> ← site under attack

script parameter → `q=<iframe style=height:100%; width=100%; border:none;transparent:none;position:absolute;top:0;left:0;z-index:100; src=http://www.AttackerSite.com/ />` ← insert an HTML page inside another HTML page

HTML page shown →

11

Example ...

- Note that the vulnerable script `search.cgi` is contained in the vulnerable site
- The script searches for the content defined by the variable `q` and displays the search result
- The `iframe` overwrites the search result and displays the content from the attack site

12

Modified attack string

XSS attack string can be URL-encoded to better hide the content:

```
http://www.VulnerableSite.com/search.cgi?  
q=%3C%69%66%72%61%6D%65%20%73%74%79%6C%65  
%3D%68%65%69%67%68%74%3A%31%30%30%25%3B%77  
%69%64%74%68%3A%31%30%30%25%3B%62%6F%72%64  
%65%72%3A%6E%6F%6E%65%3B%74%72%61%6E%73%70  
%61%72%65%6E%74%3A%6E%6F%6E%65%3B%70%6F%73  
%69%74%69%6F%6E%3A%61%62%73%6F%6C%75%74%65  
%3B%74%6F%70%3A%30%3B%6C%65%66%74%3A%30%3B  
%7A%2D%69%6E%64%65%78%3A%31%30%30%3B%20%73  
%72%63%3D%22%68%74%74%70%3A%2F%2F%65%76%69  
%6C%73%69%74%65%2E%63%6F%6D%2F%22%20%2F%3E
```

Successful XSS attack

- An attack is successful if a victim visits an URL containing the XSS attack
 1. usually, an attack is initiated by spoofing an e-mail from a reliable sender
 2. the victim clicks on a link provided in the e-mail
 3. the client's Web browser displays "malicious" Web page

What went wrong?

- The problem is that the Web server echoes back a parameter regardless of its value
- Of course, there is no good reason for a valid parameter to contain HTML tags or a script

What is SQL?

- Structured Query Language (SQL) is a standard computer language for accessing and manipulating databases
- SQL works with database programs like MS Access, MS SQL Server, Oracle, Sybase, etc.
- There are many different versions of SQL. All must support the same major keywords in a similar manner

SQL injection

- Many client-server applications utilize databases
- An application is vulnerable to SQL injection if **unvalidated** user input can generate SQL queries
- SQL injection has more severe consequences than XSS because it is possible to compromise the integrity of a database

Example of SQL injection

- Typical SQL query used to generate dynamic Web page: **everything** **table in database**

```
SELECT * FROM articles  
WHERE id='<user input>';
```
- An attacker can select the user input:

```
' ; DROP ('articles');
```

Modified SQL query

- Attacker can use the SQL commands
`SELECT * FROM articles WHERE id='';`
`DROP ('articles');`
- The commands will select some data, delete the table “articles” in the database, and then generate an SQL error due to the single quotation mark

Changed threat model

- Internet was created for information sharing
- E-governments will improve and extend the information sharing
- The Web is *completely* different from radio, TV, and postal services because there is an **up-channel**
- The up-channel has changed the threat model
 - allows crackers to attack Web sites from all over the world as many times as they like

Malicious data mining

- **Malicious data mining:** search for data to be used in malicious attacks on private, business or government systems
- Increased risk when e-government databases are made accessible through Web applications

Investigation

- **E-government Web applications:** all Web pages under
 - a gov.<country code> sub domain, and/or
 - Web pages from government, ministries, and parliament
- Probed e-government Web applications for XSS and SQL injection vulnerabilities

Investigation ...

- A Web application is said to be vulnerable to XSS if it is possible to insert script or HTML code on a dynamic Web page
- The application is vulnerable to SQL injection if it is possible to change the structure of an SQL query run by a dynamic Web page
- For simplicity, a country is said to be vulnerable if there exists one vulnerable e-government application

Methods

- There exist programs to check Web applications for common vulnerabilities
- During the investigation all e-governments were checked manually to control the pages and input vectors used

Results in 2005

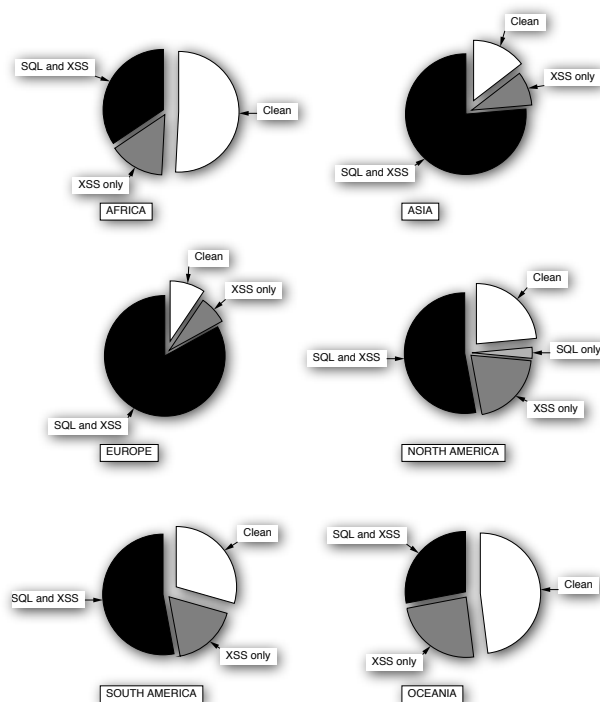
- Located e-government sites for 212 countries
- 173 countries were vulnerable to either XSS or SQL injection
- **81.6% of the countries with Web portals were vulnerable to simple attacks**
- Useful definitions:
 - Oceania: Australia and a huge number of widely scattered islands across the Pacific Ocean
 - G8: Britain, Canada, France, Germany, Italy, Japan, Russia, and the United States

25

Percentage of vulnerabilities in e-governments for each continent. Number of countries enclosed in parenthesis

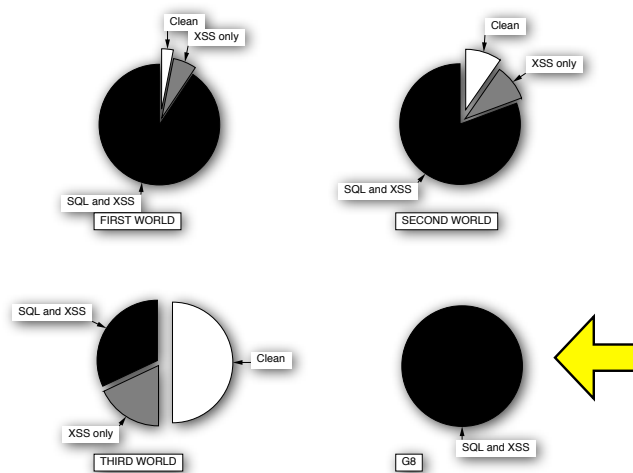
Continent	Only XSS	Only SQL	XSS and SQL	XSS or SQL	None
Africa (61)	14.75 (9)	0.00 (0)	34.43 (21)	49.18 (30)	50.82 (31)
Asia (55)	9.09 (5)	0.00 (0)	76.36 (42)	85.45 (47)	14.55 (8)
Europe (53)	7.55 (4)	0.00 (0)	83.02 (44)	90.57 (48)	9.43 (5)
North America (34)	20.59 (7)	2.94 (1)	52.94 (18)	76.47 (26)	23.53 (8)
Oceania (25)	24.00 (6)	0.00 (0)	28.00 (7)	52.00 (13)	48.00 (12)
South America (17)	17.65 (3)	0.00 (0)	52.94 (9)	70.59 (12)	29.41 (5)

26

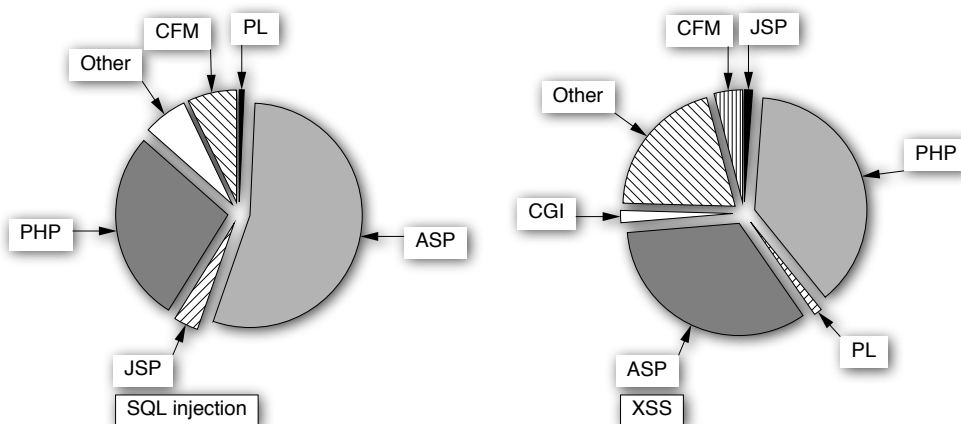


Percentage of vulnerabilities in e-governments for different country categories. Number of countries enclosed in parenthesis

Country category	Only XSS	Only SQL	XSS and SQL	XSS or SQL	None
1st World (32)	6.25 (2)	0.00 (0)	90.63 (29)	96.88 (31)	3.12 (1)
2nd World (31)	9.68 (3)	0.00 (0)	80.64 (25)	90.32 (28)	9.68 (3)
3rd World (50)	18.00 (9)	0.00 (0)	32.00 (16)	50.00 (25)	50.00 (25)
G8 (8)	0.00 (0)	0.00 (0)	100.00 (8)	100.00 (8)	0.00 (0)

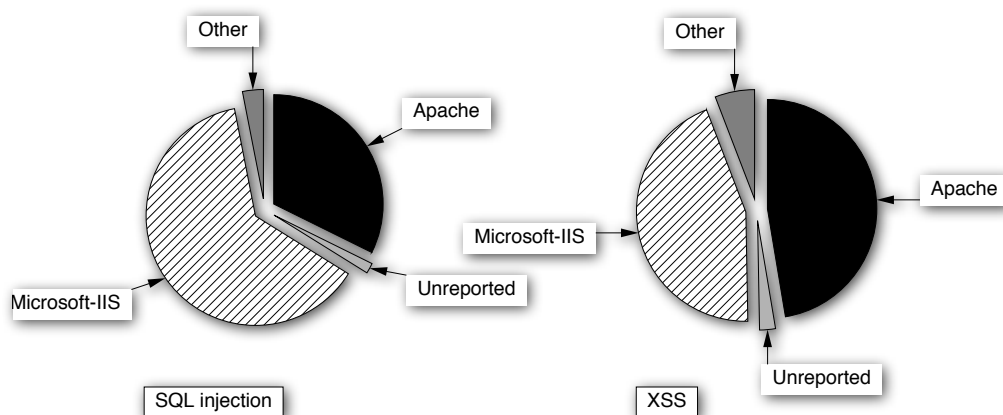


Utilized Web technologies



ASP.Net—programming framework to build Web applications

Web server software



31

“Invulnerable” sites

- The **complexity** of a Web site can be estimated by its number of pages and the number of different technologies used to generate the pages
- The sites with no discovered vulnerabilities seemed to be less complex than the vulnerable sites
- In other words, the results indicate that there is a trade-off between security and complexity
 - a secure Web application generates few Web pages with limited functionality (?)

32

Defenses

- Defending against XSS and SQL injection attacks requires *input validation*
- Input can be validated on two different levels:
 - Web application
 - Web server
- Defending against these attacks is a continuous process
- [Sverre H. Huseby, *Innocent Code: a Security Wake-up Call for Web Programmers*. Wiley, 2004.](#)

33

SSNs

- SSNs are unique “labels” identifying people
- Norwegian SSNs are not secret, but access to SSNs linked to personal information is restricted
- Norwegian SSNs have a well defined structure that makes it possible to generate all valid SSNs for any given day/month/year for both men and women

34

More on SSNs

- Unfortunately, SSNs are used for authentication purposes in many cases
- To gain more insight into the problem of SSN misuse, we investigated how SSNs were used in Norway
- In 2005, there existed government Web sites that allowed us to filter valid SSN numbers belonging to real persons

Collecting SSNs

- A short Python script was written to filter real SSNs using the Web site of a Norwegian pension fund
- for each member SSN, the site returned the member's name, home address, and name and address of the member's workplace
- the script obtained the SSNs of all members of the Norwegian cabinet
- the script also made it possible to build a large database containing the names and SSNs of all the pension fund members (we didn't build the database)

Strange attitude

- Both the pension fund and the Norwegian Data Inspectorate (Datatilsynet) were informed about our findings
- The pension fund only made minor changes to their site
- It seemed to us that the pension fund didn't get the seriousness of the situation

Using SSNs to do mischief

- Many Web portals “authenticate” users based on knowledge of a valid SSN and perhaps a name
- In 2005 it was possible—using only a name and an SSN—to
 - open a new bank account
 - apply for loans
 - order a new tax card or health card

Using SSNs ...

- A simple script using different SSNs could log on to order/apply for a new tax card/health card for all Norwegians
- During 2003 and 2004 it was possible to use lists of valid SSNs to brute-force accounts in online banks

Social engineering

- **Social engineering:** the practice of obtaining confidential information by manipulation of legitimate users
- A social engineer may use the telephone or internet to trick people into revealing sensitive information or getting them to do something that is against typical policies
- The combination of Web-based attacks, SSN filtering, and social engineering is a powerful technique that can cause a lot of problems

Conclusions

- The vast majority of dynamic e-government Web pages were vulnerable to XSS and SQL injection in 2005
- Attacks had been known for years, but still e-government was vulnerable
- E-government increased the risk of malicious data mining
- Some Web sites made it possible to filter out useful information, e.g. SSNs belonging to real persons

Source

- V. Moen, A. N. Klingsheim, K. I. F. Simonsen, and K. J. Hole, “Vulnerabilities in E-Governments,” in *Proc. 2nd International Conference on Global E-Security (ICGeS-06)*, London, England, April 20–22, 2006
- www.nowires.org/Papers-PDF/ICGeS_egov.pdf

