Security Summary

Server-Side Web Languages

Uta Priss
School of Computing
Napier University, Edinburgh, UK
Outline

PHP-security

Software lifecycle

General Security

Webserver security

PHP security
PHP-security information on the web

Quotes from an on-line forum:
PHP-security information on the web

Quotes from an on-line forum:

“Perl/CGI scripts are very insecure.”
“A PHP programmer does not have to worry about security”.
PHP-security information on the web

Quotes from an on-line forum:

“Perl/CGI scripts are very insecure.”
“A PHP programmer does not have to worry about security”.

“At first my remote connection to Mysql did not work, but then I discovered I only had to stop my firewall and it worked fine.”
PHP-security information on the web

Quotes from an on-line forum:

“Perl/CGI scripts are very insecure.”
“A PHP programmer does not have to worry about security”.

“At first my remote connection to Mysql did not work, but then I discovered I only had to stop my firewall and it worked fine.”

“You can use HTTP_REFERER to make sure that your site can only be accessed from your web form.”
All you need to connect to a database with PHP is something like this:

```php
<?php
$db = pg_pconnect('host=localhost,dbname=a,user=b');
pg_exec($db,'select * from $table');
?>
```
To send an email with PHP back to a user, you’ll need something like this:

```php
<?php
$body = 'Hi, How are you?';
mail($user, 'Subject', $body)
?>
```
Software testing

Traditional approaches for software testing (functional testing, user testing, ...) are useless for security validation.

Security validation:

- no “debugging”, no immediate feedback
- no clear testing protocols
- different types of problems are possible: requires lateral thinking
Security Engineering

see “patterns & practices Security Engineering Index” (msdn.microsoft.com)

▶ Security objectives
▶ Threat modeling
▶ Security design guidelines
▶ Security architecture and design reviews
▶ Security code reviews
▶ Security testing
▶ Security deployment reviews
General security risks

- physical security
- social engineering and human error (e.g. insecure passwords)
- eavesdropping, “man-in-the-middle” attacks
- software flaws (buffer overflows)
- installation of malicious software:
  - Trojan horses, backdoors, viruses, worms
- denial of service (DoS) attacks
General security risks

- physical security
- social engineering and human error (e.g. insecure passwords)
- eavesdropping, “man-in-the-middle” attacks
- software flaws (buffer overflows)
- installation of malicious software:
  - Trojan horses, backdoors, viruses, worms
- denial of service (DoS) attacks

The most common security risk for scripting languages ("user submitted data") is not in this list!
Security Strategies

▶ prevention
Security Strategies

- prevention
  - security guidelines, advisories, common sense
- detection
Security Strategies

- prevention
  security guidelines, advisories, common sense
- detection
  monitor webserver logs, system activity, detection software
- response
Security Strategies

- prevention
  security guidelines, advisories, common sense
- detection
  monitor webserver logs, system activity, detection software
- response
  script-level, webserver, institutional policies
Apache error log:

66.147.118.70-[7/7/06] “GET /phpadmin/main.php HTTP/1.1” 404
66.147.118.70-[7/7/06] “GET /phpmyadmin1/main.php HTTP/1.1” 404
66.147.118.70-[7/7/06] “GET /phpAdmin-2/main.php HTTP/1.1” 404
 Debian Security Advisory - phpmyadmin (DSA 1207-2)

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Debian Security Advisory DSA 1207-2
http://www.debian.org/security/
November 19th, 2006

Package : phpmyadmin
Vulnerability : several
Problem-Type : remote
Debian-specific: no
CVE ID : CVE-2006-1678 CVE-2006-2418 CVE-2006-5116
Debian Bug : 339437 340438 362567 368082 39109

The phpmyadmin update in DSA 1207 introduced a reg
corrects this flaw. For completeness, the original
Webserver security

Web space is often hosted externally and shared with other users.
Webserver security

Web space is often hosted externally and shared with other users.

- disallow server-side includes
- disallow indexes
- only store files in the public_html directory if they really need to be there
- security through obscurity
Webserver security (continued)

Apache’s mod_security

- place Apache in a chroot directory
- POST filtering based on headers, values, IP addresses
- POST payload analysis
- restrict the use of certain HTML tags (e.g. `<script>`)  
- prevent SQL injection ("delete", "insert")
- prevent SHELL commands
- etc

Of course, the server will run slower and use more memory
Other server functions

- Email: protect against spam and phishing
- install email server on different machine from webserver if possible
- don’t allow the www user to send email
- HTACCESS
  useful for group-based restriction to part of site
  not very useful for login/registration of users
- database
  DB security and script security need to be integrated
  prevent SQL injection
PHP security

- Use appropriate functions: 
  htmlspecialchars(); strip_tags(); addslashes(); 
  mysql_real_escape_string(); etc
- apply “hardening” patch to PHP before installing
- PHP safe_mode
  restrict file access, executable directory, disable functions etc