

我們與常人的距離

講者：zeze



目錄

1. 弱點掃描與滲透測試介紹

2. 單一漏洞

- SQLI
- XSS
- 無效的存取控管

3. 多漏洞串聯

- 無效的存取控管 + 不安全的組態設定

4. 實戰練習

- CVE-2013-4338

弱點掃描與滲透測試介紹

弱點掃描 – VULNARIBILITY SCAN(VULNSCAN)

- 簡介：檢查管理的主機、伺服器或網路設備是否存在可能的漏洞，以及確認機器上各個 Port 的狀態與相關的服務
- 特性：自動化、識別已知漏洞、成本低、速度快、測試頻率高

滲透測試 – PENETRATION TEST(PENTEST)

- 簡介：在應用層面或網絡層面進行攻擊，針對具體的功能，尋長正常業務流程中未知的安全缺陷
- 特性：人工+工具、發現且利用漏洞、成本高、速度慢、測試頻率低

不論是弱點掃描，或者滲透測試，最重要的一點就是授權



測試流程介紹

Step 1: 檢測範圍 – 網段、服務、外網、內網……

Step 2: 檢測時程

Step 3: 檢測工具與方式 –

(工具) Nmap、OpenVAS、OWASP ZAP、……

(方式) 埠掃描、DDOS、社交工程、爆破密碼、……

Step 4: 實施檢測

Step 5: 改善作業

Step 6: 進行複檢

測試流程介紹

Step 1: 檢測範圍 – 網段、服務、外網、內網……

Step 2: 檢測時程

Step 3: 檢測工具與方式 –

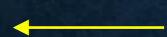
(工具) Nmap、OpenVAS、OWASP ZAP、……

(方式) 埠掃描、DDOS、社交工程、爆破密碼、……

Step 4: 實施檢測

Step 5: 改善作業

Step 6: 進行複檢



這兩個步驟可能會重複很多次



測試的盲點

➤ 測試目的：要把弱點與漏洞填補掉，落實資訊安全

➤ 困難：網站的管理者可能沒有資安的相關知識

可能導致檢測方與受測方反反覆覆的修補與測試，浪費時間，甚至可能最後還是沒完全修好

解釋漏洞


- 成因：介紹漏洞、漏洞的位置、攻擊方法與過程
- 危害：漏洞可能會造成的結果
- 解決方法：提供程式或者設定的修改方法

解釋漏洞

➤ 成因：介紹漏洞、漏洞的位置、攻擊方法與過程

➤ 危害：漏洞可能會造成的結果

➤ 解決方法：提供程式或者設定的修改方法



問題常出現在成因與解法

單一漏洞

介紹

➤ 簡介： 只要透過一個步驟就能完成的攻擊，大多能用工具掃出來的都是這類（不包含掃 CVE）

➤ 溝通難易度： 較低

漏洞較單純，修改也比較容易

SQLI

- 簡介：發生於應用程式與資料庫層的漏洞，在輸入的字串之中夾帶 SQL 指令，這些惡意指令就會被資料庫伺服器誤認為是正常的 SQL 指令而執行，因此遭到破壞或是入侵。
- 常用工具：sqlmap

舉



```
SELECT * FROM users WHERE name = '$USER' and passwd='$PASSWORD' ;
```



\$USER = ' or 1=1--

\$PASSWORD = a

```
SELECT * FROM users WHERE name = " or 1=1--" and passwd='a' ;
```

SQLI 解釋漏洞 – 錯誤示範

➤ 成因：在登入頁面有 SQLI

1. 提供整串網址
2. 簡單介紹 SQLI
3. 條列攻擊的步驟
4. 截圖

➤ 危害：高

➤ 解決方法：不要用字串拼接的方式構造 query

1. 提供具體解決辦法
2. 提供修改建議

SQLI 解釋漏洞 – 正確示範 ✓

- 成因：在 <http://exam.ple/login> 頁面有 SQLI(SQL injection)。SQLI 是發生於應用程式與資料庫層的漏洞，在輸入的字串之中夾帶 SQL 指令，這些惡意指令就會被資料庫伺服器誤認為是正常的 SQL 指令而執行。測試方法為在頁面中的帳號欄位輸入「' or 1=1--」，可以看到登入成功後的頁面。
- 危害：高
- 解決方法：不要用字串拼接的方式構造 query，改使用參數化查詢，參考資料 <http://solution>。不建議使用黑名單方式過濾特殊字元。

exam.ple/login

帳號：' or 1=1--

密碼：*

登入

XSS

- 簡介：XSS 是指攻擊者往 Web 頁面裏插入惡意 html 代碼，當用戶瀏覽該頁之時，嵌入其中的 html 代碼會被執行，從而達到攻擊者的目的。
- 常用工具：XSSStrike, Xspear, xss-scanner, ……

舉



User input : Hello

ex.am/ple.php?s=Hello

Your input:

Hello

User input :

ex.am/ple.php?s=

Your

ex. am 顯示
1

確定

取消



XSS 解釋漏洞 – 錯誤示範

➤ 成因：在搜尋頁面有 XSS

1. 提供整串網址
2. 簡單介紹 SQLI
3. 條列攻擊的步驟
4. 截圖

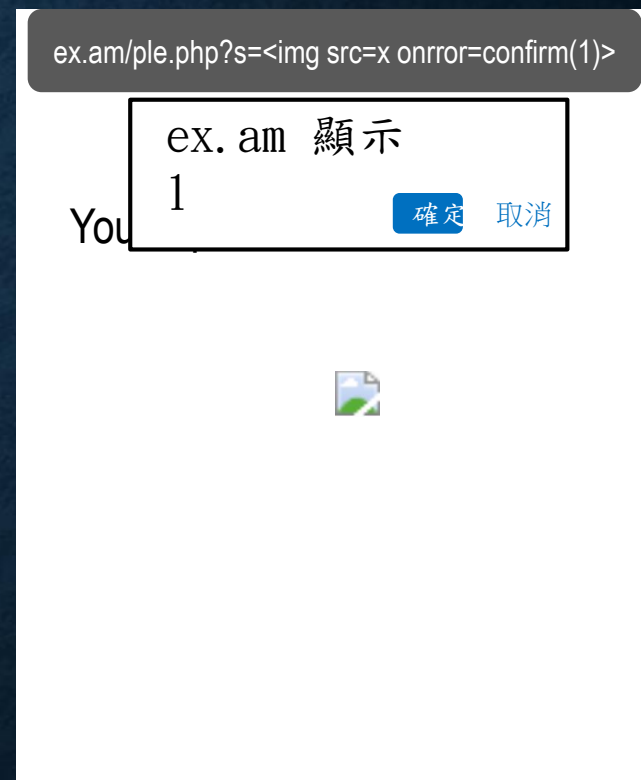
➤ 危害：高

➤ 解決方法：不要讓使用者的輸入被當作可執行的代碼

1. 提供具體解決辦法
2. 提供修改建議

XSS 解釋漏洞 – 正確示範 ✓

- 成因：在 `http://exam.ple/search` 頁面有 XSS(Cross-site scripting)。XSS 是指攻擊者往 Web 頁面裏插入惡意 html 代碼，當用戶瀏覽該頁之時，嵌入其中的 html 代碼會被執行，從而達到攻擊者的目的。測試方法為在頁面的搜尋欄位中輸入「``」，可以看到確認框跳出。
- 危害：高
- 解決方法：不要讓使用者的輸入被當作可執行的代碼，可以將使用者的輸入進行編碼或是移除使用者上傳的 DOM 屬性，參考資料 `http://solution`。不建議使用黑名單方式過濾特殊字元。



SO FAR, 成因的重要性似乎不高(?)

無效的存取控管

- 簡介：攻擊者可透過網址或 HTML 頁面，繞過存取控制；或將自己的權限提升至管理者，進而攻破系統等。



舉



未做好權限控管，一般使用者能繞過登入頁面，可以操作管理者的動作

exam.ple/admin/list

首頁
新聞
介紹
回饋

·
·
·

exam.ple/admin/list/intro

首頁
新聞
介紹
回饋

·
·
·

修改新聞

新增新聞

無效的存取控管 解釋漏洞 – 錯誤示範 ❌

- 成因：在 `http://exam.ple/admin/list` 發現無效存取控管。攻擊者可透過網址或 HTML 頁面，繞過存取控制；或將自己的權限提升至管理者，進而攻破系統等。測試方法為輸入該網址，可以繞過登入介面的身分驗證讓未授權的使用者直接操作管理者功能。
- 危害：高
- 解決方法：在管理者才能操作的頁面做身分驗證，避免攻擊者繞過驗證使用該功能。

`exam.ple/admin/list`

首頁
新聞
介紹
回饋

·
·
·

問題出在哪？

真實事件改編

複測時……

exam.ple/admin/list

Forbidden

exam.ple/admin/list/intro

首頁
新聞
介紹
回饋

修改新聞

·
·
·

新增新聞

無效的存取控管 解釋漏洞 – 正確示範 ✓

➤ 成因：在 `http://exam.ple/admin/list` 發現無效存取控管。攻擊者可透過網址或 HTML 頁面，繞過存取控制；或將自己的權限提升至管理者，進而攻破系統等。測試方法為輸入該網址，可以繞過登入介面的身分驗證讓未授權的使用者直接操作管理者功能。此頁面下的 `/home`, `/news`, `/intro`, `/feedback`, …… 底下的所有頁面都可以用此方法繞過。

➤ 危害：高

➤ 解決方法：在管理者才能操作的頁面做身分驗證，避免攻擊者繞過驗證使用該功能。

exam.ple/admin/list

首頁
新聞
介紹
回饋
.
.

exam.ple/admin/list/intro

首頁
新聞
介紹
回饋
.
.

修改新聞

新增新聞

多漏洞串聯

介紹

➤ 簡介：透過多個低危害的單一漏洞配合，造成較大危害的漏洞出現。

➤ 溝通難易度：較高

這類型的漏洞通常攻擊步驟繁雜，解釋起來也比較困難，而且常常會漏掉一些東西沒改。

無效的存取控管 + 不安全的組態設定

- 無效的存取控管：攻擊者可透過網址或 HTML 頁面，繞過存取控制；或將自己的權限提升至管理者，進而攻破系統等。
- 不安全的組態設定：應用程式、伺服器，及平台的設定，若缺少適當的設定，可能導致敏感性資訊外洩，進而容易遭受入侵或攻擊。

舉



管理者的圖片上傳頁面沒做身分驗證

圖片上傳沒有擋好檔案類型

exam.ple/admin/imgupload

選擇檔案

上傳

exam.ple/admin/imgupload

hack.php

上傳

上傳成功

解釋漏洞 – 錯誤示範 ❌

➤ 成因：在 `http://exam.ple/admin/imgupload` 發現無效存取控管與不安全的組態設定，可以讓未授權的使用者拿到 shell。無效存取控管為攻擊者可透過網址或 HTML 頁面，繞過存取控制；或將自己的權限提升至管理者，進而攻破系統等。不安全的組態設定為應用程式、伺服器，及平台的設定，若缺少適當的設定，可能導致敏感性資訊外洩，進而容易遭受入侵或攻擊。測試方法為輸入該網址，可以繞過登入介面的身分驗證直接使用管理者的檔案上傳功能。並且可以在上傳的 php 類型的檔案中填入 Reverse shell，上傳後打開此檔案頁面即可拿到 shell。

➤ 危害：高

➤ 解決方法：在管理者才能操作的頁面做身分驗證，避免攻擊者繞過驗證使用該檔案上傳功能。在檔案上傳功能頁面設立白名單，擋掉不安全的檔案類型，不建議使用黑名單的方式。

exam.ple/admin/imgupload

選擇檔案 上傳

exam.ple/admin/imgupload

hack.php 上傳

上傳成功

切成兩份寫

沒有漏洞是切成兩個解決不了的，有的話就三個！

解釋漏洞 – 正確示範 – 1 ✓

➤ 成因：在 `http://exam.ple/admin/imgupload` 發現無效存取控管。無效存取控管為攻擊者可透過網址或 HTML 頁面，繞過存取控制；或將自己的權限提升至管理者，進而攻破系統等。測試方法為輸入該網址，可以繞過登入介面的身分驗證直接使用管理者的檔案上傳功能。

➤ 危害：中

➤ 解決方法：在管理者才能操作的頁面做身分驗證，避免攻擊者繞過驗證使用該檔案上傳功能。

`exam.ple/admin/imgupload`

選擇檔案

上傳

解釋漏洞 – 正確示範 – 2 ✓

- 成因：在 `http://exam.ple/admin/imgupload` 發現不安全的組態設定。不安全的組態設定為應用程式、伺服器，及平台的設定，若缺少適當的設定，可能導致敏感性資訊外洩，進而容易遭受入侵或攻擊。測試方法為可以在上傳的 php 類型的檔案中填入 Reverse shell，上傳後打開此檔案頁面即可拿到 shell。
- 危害：高
- 解決方法：在檔案上傳功能頁面設立白名單，擋掉不安全的檔案類型，不建議使用黑名單的方式。



實戰練習

CVE-2013-4338

- 簡介：發生在 wordpress 3.6.1 前的不安全反序列化漏洞
- 官方描述：wp-includes/functions.php in WordPress before 3.6.1 does not properly determine whether data has been serialized, which allows remote attackers to execute arbitrary code by triggering erroneous PHP unserialize operations.

Reference: <https://www.slideshare.net/phdays/php-wordpress>


```

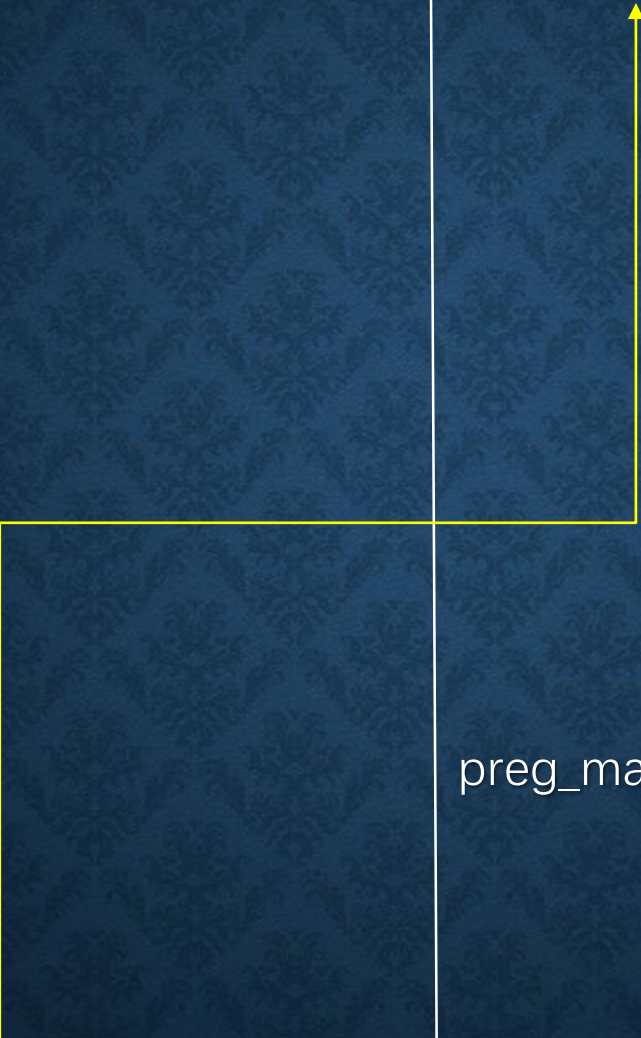
function is_serialized( $data ) {
    if ( ! is_string( $data ) )
        return false;
    $data = trim( $data );
    if ( 'N;' == $data )
        return true;
    $length = strlen( $data );
    if ( $length < 4 )
        return false;
    if ( ':' != $data[1] )
        return false;
    $lastc = $data[$length-1];

```

```

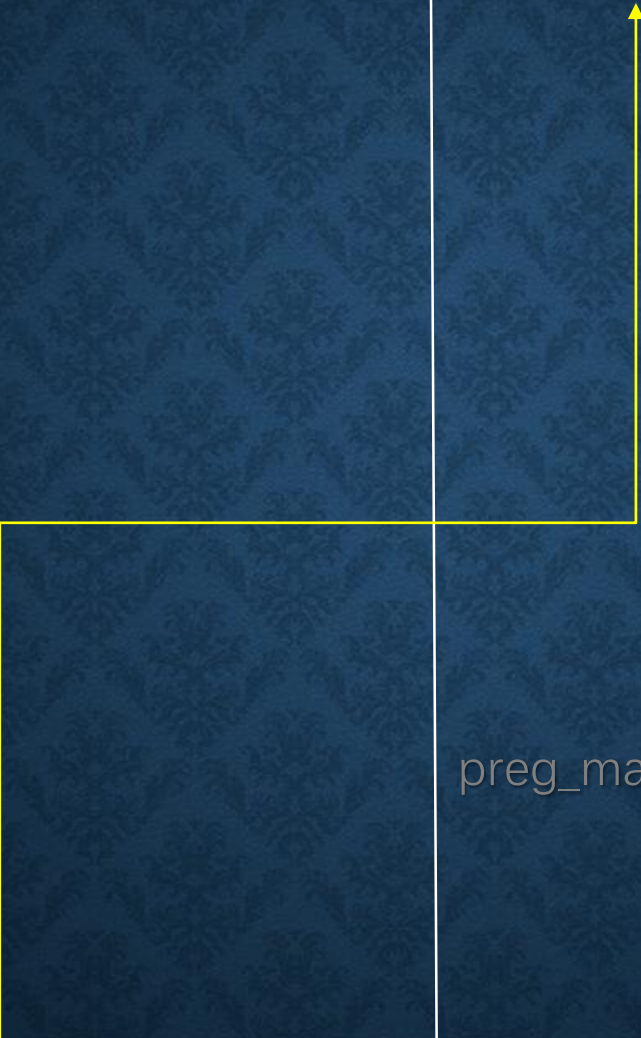
    if ( ';' != $lastc && '}' != $lastc )
        return false;
    $token = $data[0];      switch ( $token ) {
        case 's' :
            if ( '"' != $data[$length-2] )
                return false;
        case 'a' :
        case 'O' :
            return (bool)
preg_match( "/^{$token}: [0-9]+:/s", $data );
        case 'b' :
        case 'i' :
        case 'd' :
            return (bool)
preg_match( "/^{$token}: [0-9.E-]+;\\$/", $data );
    }
    return false;}

```



```
function is_serialized( $data ) {  
    if ( ! is_string( $data ) )  
        return false;  
  
    $data = trim( $data );  
    if ( 'N;' == $data )  
        return true;  
  
    $length = strlen( $data );  
    if ( $length < 4 )  
        return false;  
  
    if ( ':' !== $data[1] )  
        return false;  
  
    $lastc = $data[$length-1];
```

```
    if ( ';' !== $lastc && '}' !== $lastc )  
        return false;  
  
    $token = $data[0];    switch ( $token ) {  
        case 's' :  
            if ( '"' !== $data[$length-2] )  
                return false;  
  
            case 'a' :  
            case 'O' :  
                return (bool)  
preg_match( "/^{$token}: [0-9]+: /s", $data );  
  
            case 'b' :  
            case 'i' :  
            case 'd' :  
                return (bool)  
preg_match( "/^{$token}: [0-9.E-]+; \$/", $data );  
    }  
    return false;}
```



MYSQL VS UTF-8

前提：

- UTF-8 支援 U+0000 到 U+10FFFF
- MYSQL 支援 UTF-8

但是 MYSQL 的 UTF-8 只支援 U+0000 到 U+FFFF，所以就空出了 U+10000 到 U+10FFFF。
如果輸入 U+10000 到 U+10FFFF 到 MYSQL，就會連同該字元後面的東西一起被消失。

利用方法

如果輸入「s:3:"dadadadada";」，後面再接一個 U+10000 到 U+10FFFF 的編碼……

1. 將資料存入 MYSQL 時被 `is_serialized()` 判定成 FALSE
2. 從 MYSQL 取出來時要被 `is_serialized()` 判定成 TRUE

如此一來就可以讓任意序列化後的字串被反序列化

解釋漏洞

➤ 成因：在 wp-includes/functions.php 發現不安全的反序列化漏洞。當攻擊者提供竄改後的惡意物件進行反序列化，可能導致應用程式或 API 出現不安全的風險。測試方法為輸入一個反序列化後的字串接著 U+10000 到 U+10FFFF 的編碼，使 is_serialized() 回傳 FALSE；並透過 MYSQL 只能支援 U+0000 到 U+FFFF 的特性，使 is_serialized 回傳 TRUE。攻擊者可以藉此繞過 is_serialized() 執行任意指令。

➤ 危害：高

➤ 解決方法：判斷是否為序列化字串的方法不能只用最後一個字元就判斷不是序列化字串，而是該根據狀況與需求去做調整。

exam.ple/

s:3:"asd";❤ 提交

asd

```
function is_serialized( $data ) {  
function is_serialized( $data, $strict = true ) {
```

```
    $lastc = $data[$length-1];  
    if ( ';' !== $lastc && '}' !== $lastc )  
        return false;  
    if ( $strict ) {  
        $lastc = $data[ $length - 1 ];  
        if ( ';' !== $lastc && '}' !== $lastc )  
            return false;  
    } else {  
        // ensures ; or } exists but is not in the first X chars  
        if ( strpos( $data, ';' ) < 3 && strpos( $data, '}' ) < 4 )  
            return false;  
    }  
}
```

```
    if ( '"' !== $data[$length-2] )  
        if ( $strict ) {  
            if ( '"' !== $data[ $length - 2 ] )  
                return false;  
        } elseif ( false === strpos( $data, '"' ) ) {  
            return false;  
        }  
}
```

```
return (bool) preg_match( "/^{$token}:[0-9.E-]+;\$/", $data );  
$send = $strict ? '$' : '';  
return (bool) preg_match( "/^{$token}:[0-9.E-]+;$send/", $data );
```


THANKS FOR LISTENING[✦]