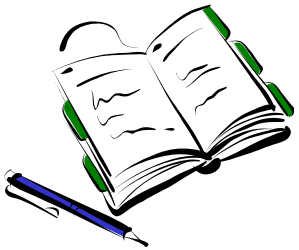
淡江大學

安全的程式碼撰寫說明



日期：99/6/22

講師：叡揚資訊產品顧問 林榮秋

eMail: willy_lin@mail.gss.com.tw

授課大綱

- 為何要重視安全的程式碼撰寫
- 了解已知的程式碼安全弱點
- Web 網站安全的基本觀念
- 如何通過 Fortify 檢測說明
- 常見安全的弱點修復說明
- Q&A

為何要重視安全的程式碼撰寫？

洩漏個資侵隱私 博客來判賠

中時電子報 www.chinatimes.com

更新日期:2008/11/20 05:24

國內知名的「博客來網路書店」網站，去年發生民眾購買金馬影展套票，回覆會員註冊成功的電子郵件，竟然夾帶「外流」先前註冊成功會員的相關個人資料，曾郁智等十七位受害人，為此提起損害賠償訴訟，台北地方法院十九日判決博客來必須賠償每人二千四百元至一萬一千九百元不等。

去年十一月四、五日，曾郁智等十七位住在台北縣市的民眾，為了購買「台北金馬影展套票」，依指示進入博客來網站登錄會員並註冊套票後，卻在博客來回覆註冊成功的電子郵件訊息中，發現夾帶先前已註冊成功的四百七十七位會員資料。

回覆電郵疏失 夾帶會員資料

此案例沒有駭客入侵，是程式開發人員沒有保護個人資料的程式碼安全撰寫觀念

GSS Galaxy Software Services 叡揚資訊 FORTIFY SOFTWARE

99/4/27「個人資料保護法」三讀通過

■ 第二十八條

公務機關違反本法規定，....，依前二項情形，如被害人不易或不能證明其實際損害額時，得請求法院依侵害情節，**以每人每一事件新臺幣五百元以上二萬元以下計算。**對於同一原因事實造成多數當事人權利受侵害之事件，經當事人請求損害賠償者，**其合計最高總額以新臺幣二億元為限**。

■ 第二十九條

非公務機關違反本法規定，致個人資料遭不法蒐集、處理、利用或其他侵害當事人權利者，負損害賠償責任。**但能證明其無故意或過失者，不在此限。**依前項規定請求賠償者，適用前條第二項至第六項規定。

■ 第三十條

損害賠償請求權，**自請求權人知有損害及賠償義務人時起，因二年間不行使而消滅；自損害發生時起，逾五年者 亦同。**

此指當事人不知有損害發生的狀況

「個人資料保護法」第 二 條 本法用詞定義

一、個人資料：指自然人之姓名、出生年月日、國民身分證統一編號、護照號碼、特徵、指紋、婚姻、家庭、教育、職業、病歷、醫療、基因、性生活、健康檢查、犯罪前科、聯絡方式、財務情況、社會活動及其他得以直接或間接方式識別該個人之資料。

二、個人資料檔案：指依系統建立而得以自動化機器或其他非自動化方式檢索、整理之個人資料之集合。

三、蒐集：指以任何方式取得個人資料。

四、處理：指爲建立或利用個人資料檔案所爲資料之記錄、輸入、儲存、編輯、更正、複製、檢索、刪除、輸出、連結或內部傳送。

五、利用：指將蒐集之個人資料爲處理以外之使用。

六、國際傳輸：指將個人資料作跨國（境）之處理或利用。

七、公務機關：指依法行使公權力之中央或地方機關或行政法人。

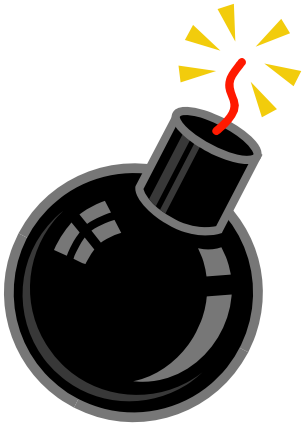
八、非公務機關：指前款以外之自然人、法人或其他團體。

九、當事人：指個人資料之本人。

新版「個資法」對組織與軟體開發人員的衝擊

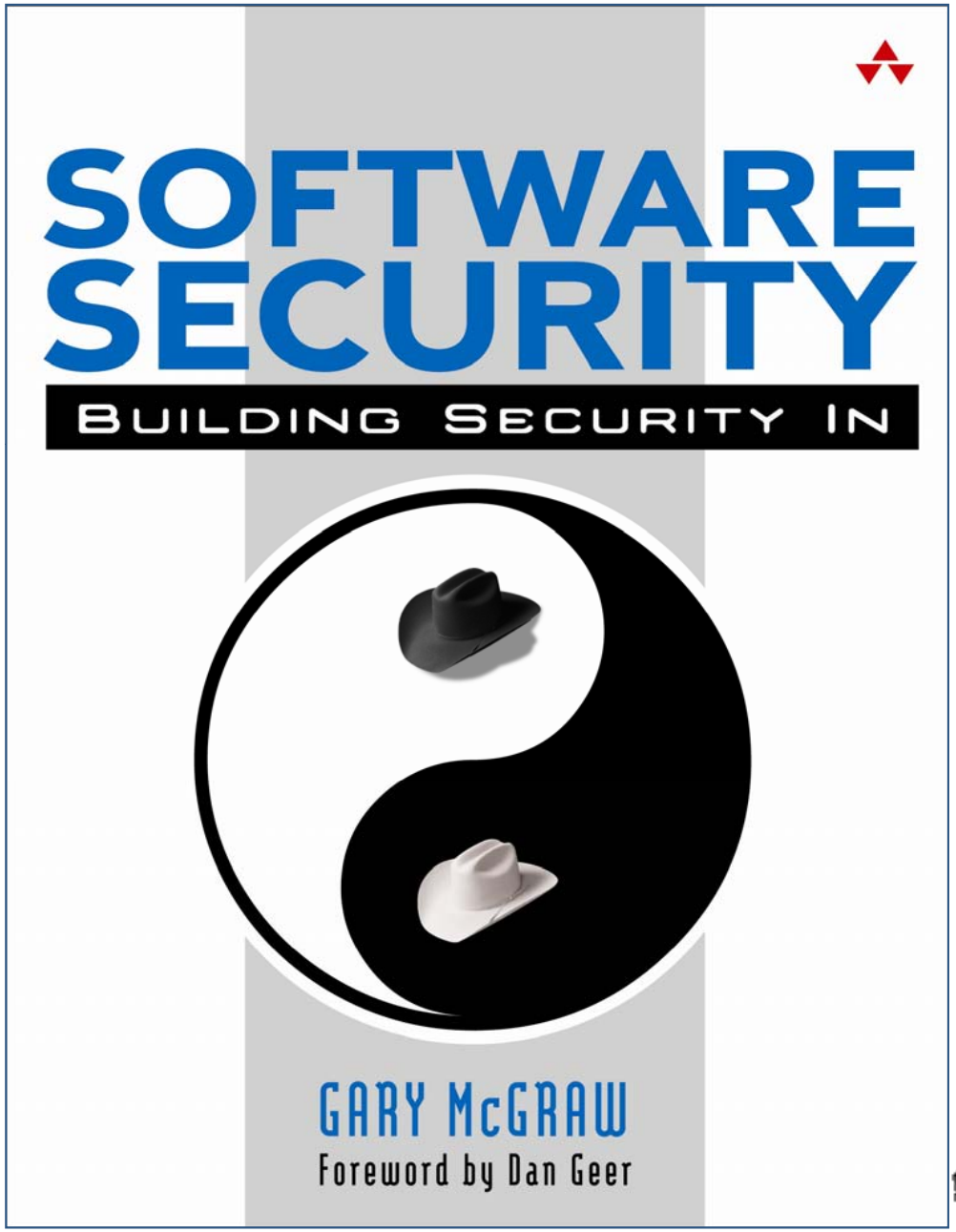
第二十八條

公務機關違反本法規定，....，依前二項情形，如被害人不易或不能證明其實際損害額時，得請求法院依侵害情節，**以每人每一事件新臺幣五百元以上二萬元以下計算。**對於同一原因事實造成多數當事人權利受侵害之事件，經當事人請求損害賠償者，**其合計最高總額以新臺幣二億元為限**。



不安全的程式碼，會成為公司或組織的賠償地雷

法律的強制要求 You Must Building Security In



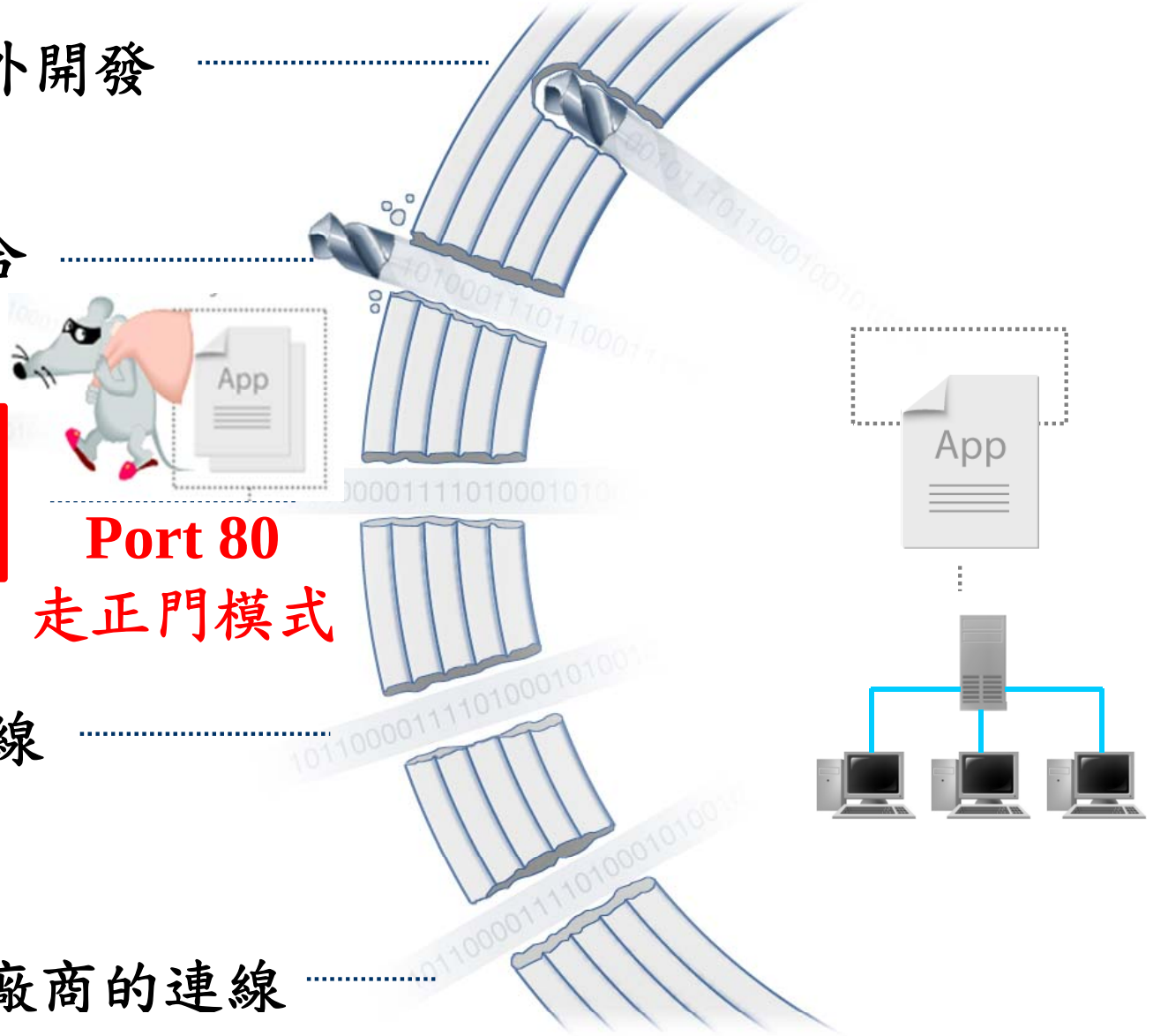
叡揚資訊 FORTIFY SOFTWARE

資訊運用的演變：網際網路應用程式風行全球

委外開發

大型主機系統整合

網際網路應用程式



Port 80

走正門模式

員工VPN的連線

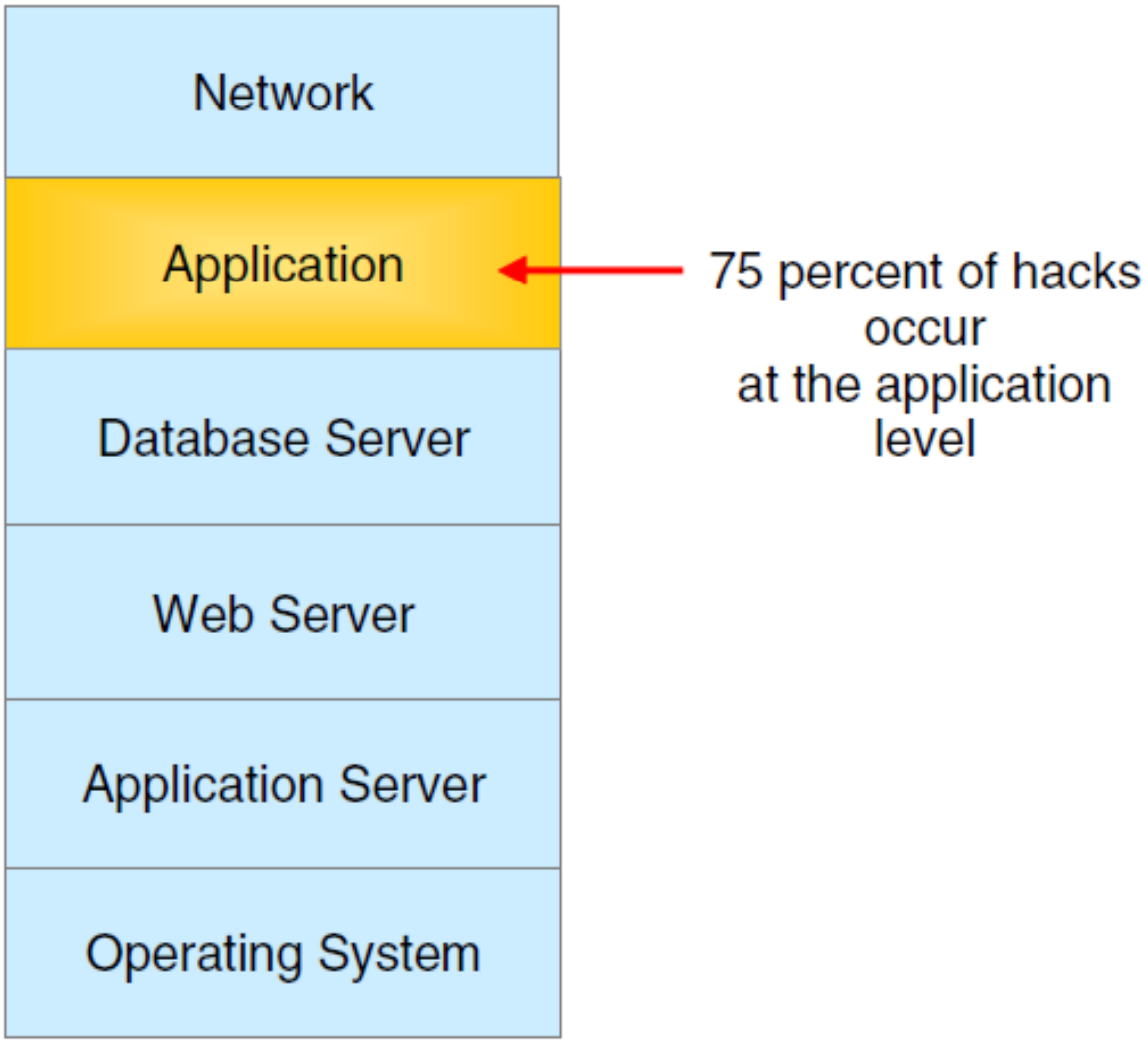
合作夥伴及供應廠商的連線

Now Is the Time for Security at the Application Level

Figure 1. Security 101

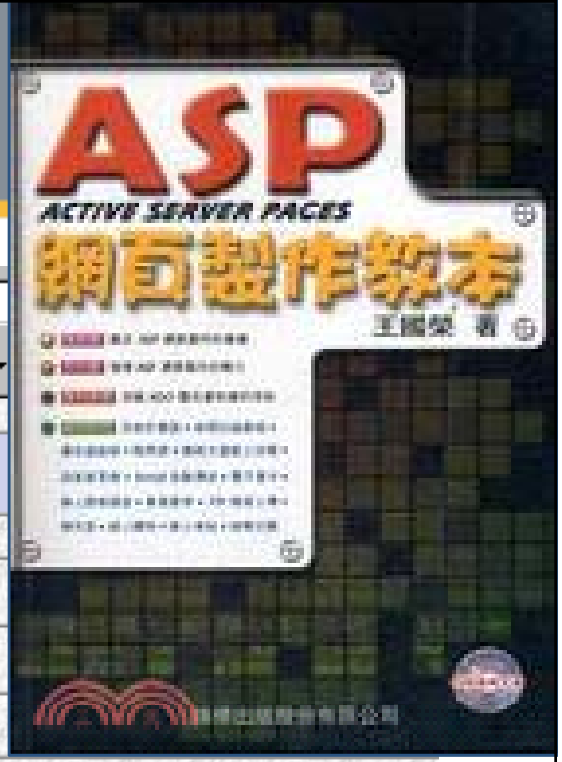
Security is many things to many people …

- Network Layer
- ID Theft
- Physical
- Administrative
- Patches
- Infrastructure
- Denial-of-Service Attacks
- Hacks
- Worms and Viruses
- Terrorism (Cyber or Physical)



Source: Gartner (November 2005)

XSS 惡意圖片連結攻擊



http://localhost/kjasp/ch11/Gform.htm

我的最愛 阿毛的留言表單

阿毛的留言表單

姓名:	Test
Email:	Test@gmail.com
主題:	XSS 攻擊語法輸入
留言:	<script>document.write('<img src = "http://www.playboy.com/girls/pmoy/images/pmoy_buythemag_300x250.jpg">')</script>
心情圖示：	不要心情圖示

送出留言　觀看留言　清掉重寫

XSS 惡意圖片連結攻擊

授課大綱

- 為何要重視安全的程式碼撰寫
- 了解已知的程式碼安全弱點
- Web 網站安全的基本觀念
- 如何通過 Fortify 檢測說明
- 常見安全的弱點修復說明
- Q&A

Fortify 提供程式語言已知安全漏洞說明的網址

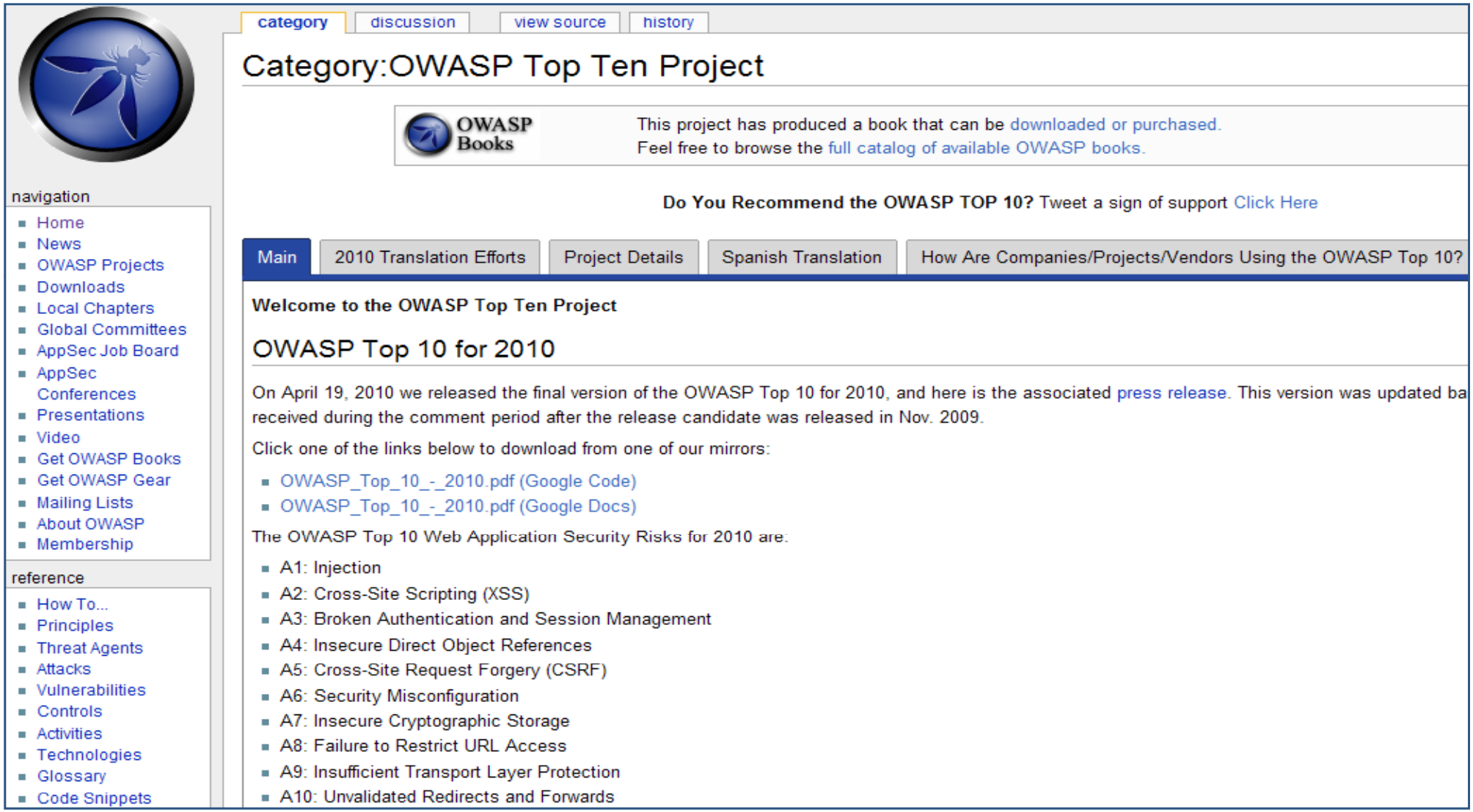
http://www.fortify.com/vulncat/zh_TW/vulncat/index.html

Fortify Taxonomy: Software Security Errors

1. Input Validation and Representation
2. API Abuse
3. Security Features
4. Time and State
5. Errors
6. Code Quality
7. Encapsulation
*. Environment

OWASP Top 10

http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project



category | discussion | view source | history

Category:OWASP Top Ten Project

OWASP Books

This project has produced a book that can be downloaded or purchased.
Feel free to browse the full catalog of available OWASP books.

Do You Recommend the OWASP TOP 10? Tweet a sign of support Click Here

Main | 2010 Translation Efforts | Project Details | Spanish Translation | How Are Companies/Projects/Vendors Using the OWASP Top 10?

navigation

- Home
- News
- OWASP Projects
- Downloads
- Local Chapters
- Global Committees
- AppSec Job Board
- AppSec Conferences
- Presentations
- Video
- Get OWASP Books
- Get OWASP Gear
- Mailing Lists
- About OWASP
- Membership

reference

- How To...
- Principles
- Threat Agents
- Attacks
- Vulnerabilities
- Controls
- Activities
- Technologies
- Glossary
- Code Snippets

Welcome to the OWASP Top Ten Project

OWASP Top 10 for 2010

On April 19, 2010 we released the final version of the OWASP Top 10 for 2010, and here is the associated press release. This version was updated ba
received during the comment period after the release candidate was released in Nov. 2009.

Click one of the links below to download from one of our mirrors:

- OWASP_Top_10_-_2010.pdf (Google Code)
- OWASP_Top_10_-_2010.pdf (Google Docs)

The OWASP Top 10 Web Application Security Risks for 2010 are:

- A1: Injection
- A2: Cross-Site Scripting (XSS)
- A3: Broken Authentication and Session Management
- A4: Insecure Direct Object References
- A5: Cross-Site Request Forgery (CSRF)
- A6: Security Misconfiguration
- A7: Insecure Cryptographic Storage
- A8: Failure to Restrict URL Access
- A9: Insufficient Transport Layer Protection
- A10: Unvalidated Redirects and Forwards

Mapping from 2007 to 2010 Top 10

OWASP Top 10 – 2007 (Previous)		OWASP Top 10 – 2010 (New)
A2 – Injection Flaws	↑	A1 – Injection
A1 – Cross Site Scripting (XSS)	↓	A2 – Cross Site Scripting (XSS)
A7 – Broken Authentication and Session Management	↑	A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object Reference	=	A4 – Insecure Direct Object References
A5 – Cross Site Request Forgery (CSRF)	=	A5 – Cross Site Request Forgery (CSRF)
<was T10 2004 A10 – Insecure Configuration Management>	+	A6 – Security Misconfiguration (NEW)
A8 – Insecure Cryptographic Storage	↑	A7 – Insecure Cryptographic Storage
A10 – Failure to Restrict URL Access	↑	A8 – Failure to Restrict URL Access
A9 – Insecure Communications	=	A9 – Insufficient Transport Layer Protection
<not in T10 2007>	+	A10 – Unvalidated Redirects and Forwards (NEW)
A3 – Malicious File Execution	-	<dropped from T10 2010>
A6 – Information Leakage and Improper Error Handling	-	<dropped from T10 2010>

CWE Top 25

http://cwe.mitre.org/top25/index.html

Common Weakness Enumeration
A Community-Developed Dictionary of Software Weakness Types

Home > CWE/SANS Top 25 2010

CWE List
Full Dictionary View
Development View
Research View
Reports
About
Sources
Process
Documents
Community
Related Activities
Discussion List
Research
CWE/SANS Top 25
CWSS
News
Calendar
Free Newsletter
Compatibility
Program
Requirements
Declarations
Make a Declaration
Contact Us
Search the Site

2010 CWE/SANS Top 25 Most Dangerous Programming Errors

http://cwe.mitre.org/top25/

Document version: 1.03 (pdf)

Date: April 5, 2010

Project Coordinators:
Bob Martin (MITRE)
Mason Brown (SANS)
Alan Paller (SANS)
Dennis Kirby (SANS)

Document Editor:
Steve Christey (MITRE)

Introduction

The 2010 CWE/SANS Top 25 Most Dangerous Programming Errors is a list of the most widespread and critical programming e
serious software vulnerabilities. They are often easy to find, and easy to exploit. They are dangerous because they will frequer
completely take over the software, steal data, or prevent the software from working at all.

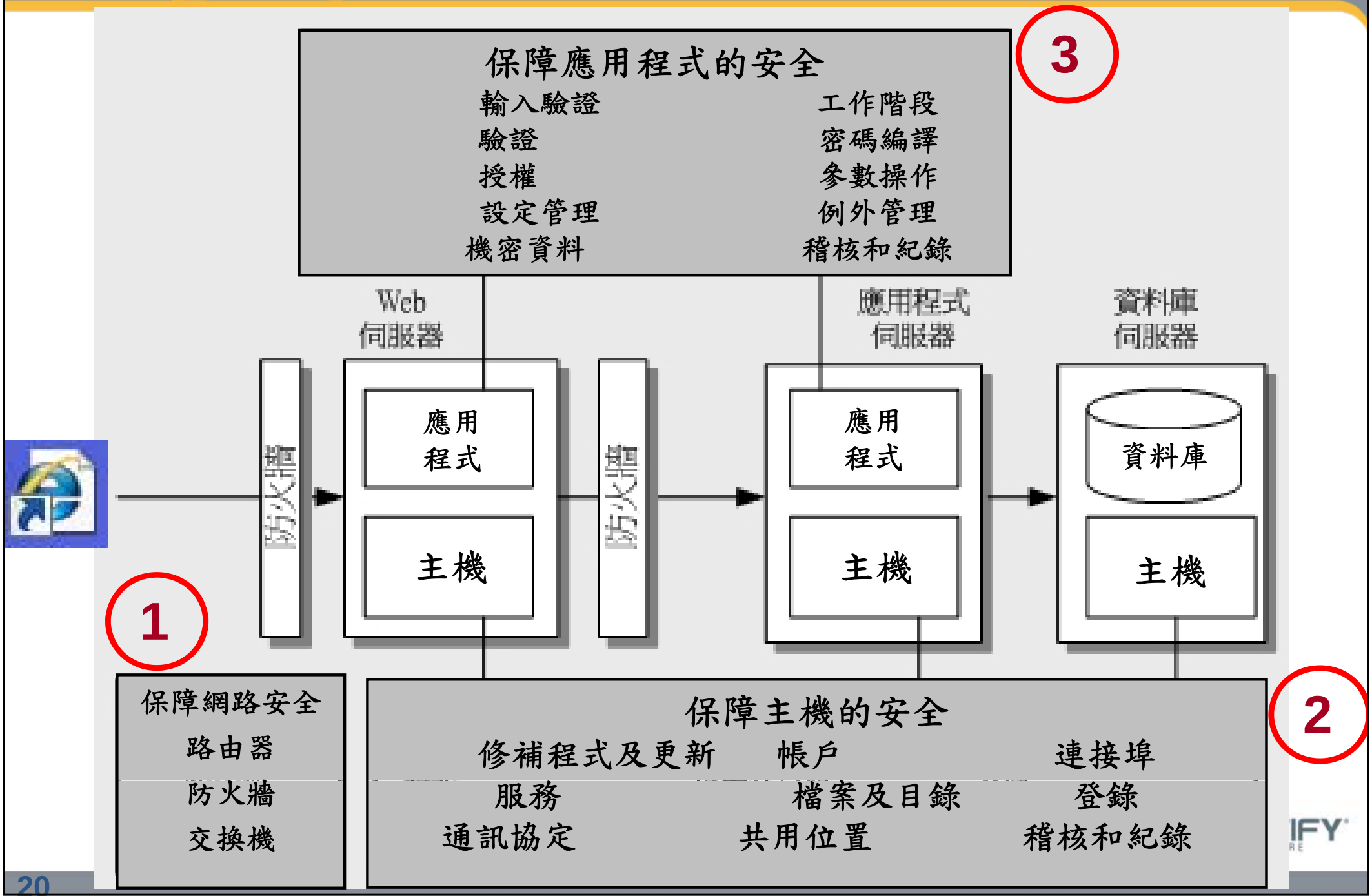
2010 CWE/SANS Top 25 Most Dangerous Programming Errors

Rank	Score	ID	Name
[1]	346	CWE-79	Failure to Preserve Web Page Structure ('Cross-site Scripting')
[2]	330	CWE-89	Improper Sanitization of Special Elements used in an SQL Command ('SQL Injection')
[3]	273	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	261	CWE-352	Cross-Site Request Forgery (CSRF)
[5]	219	CWE-285	Improper Access Control (Authorization)
[6]	202	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[7]	197	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[8]	194	CWE-434	Unrestricted Upload of File with Dangerous Type
[9]	188	CWE-78	Improper Sanitization of Special Elements used in an OS Command ('OS Command Injection')
[10]	188	CWE-311	Missing Encryption of Sensitive Data
[11]	176	CWE-798	Use of Hard-coded Credentials
[12]	158	CWE-805	Buffer Access with Incorrect Length Value
[13]	157	CWE-98	Improper Control of Filename for Include/Require Statement in PHP Program ('PHP File Inclusion')
[14]	156	CWE-129	Improper Validation of Array Index
[15]	155	CWE-754	Improper Check for Unusual or Exceptional Conditions
[16]	154	CWE-209	Information Exposure Through an Error Message
[17]	154	CWE-190	Integer Overflow or Wraparound
[18]	153	CWE-131	Incorrect Calculation of Buffer Size
[19]	147	CWE-306	Missing Authentication for Critical Function
[20]	146	CWE-494	Download of Code Without Integrity Check
[21]	145	CWE-732	Incorrect Permission Assignment for Critical Resource
[22]	145	CWE-770	Allocation of Resources Without Limits or Throttling
[23]	142	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
[24]	141	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
[25]	138	CWE-362	Race Condition

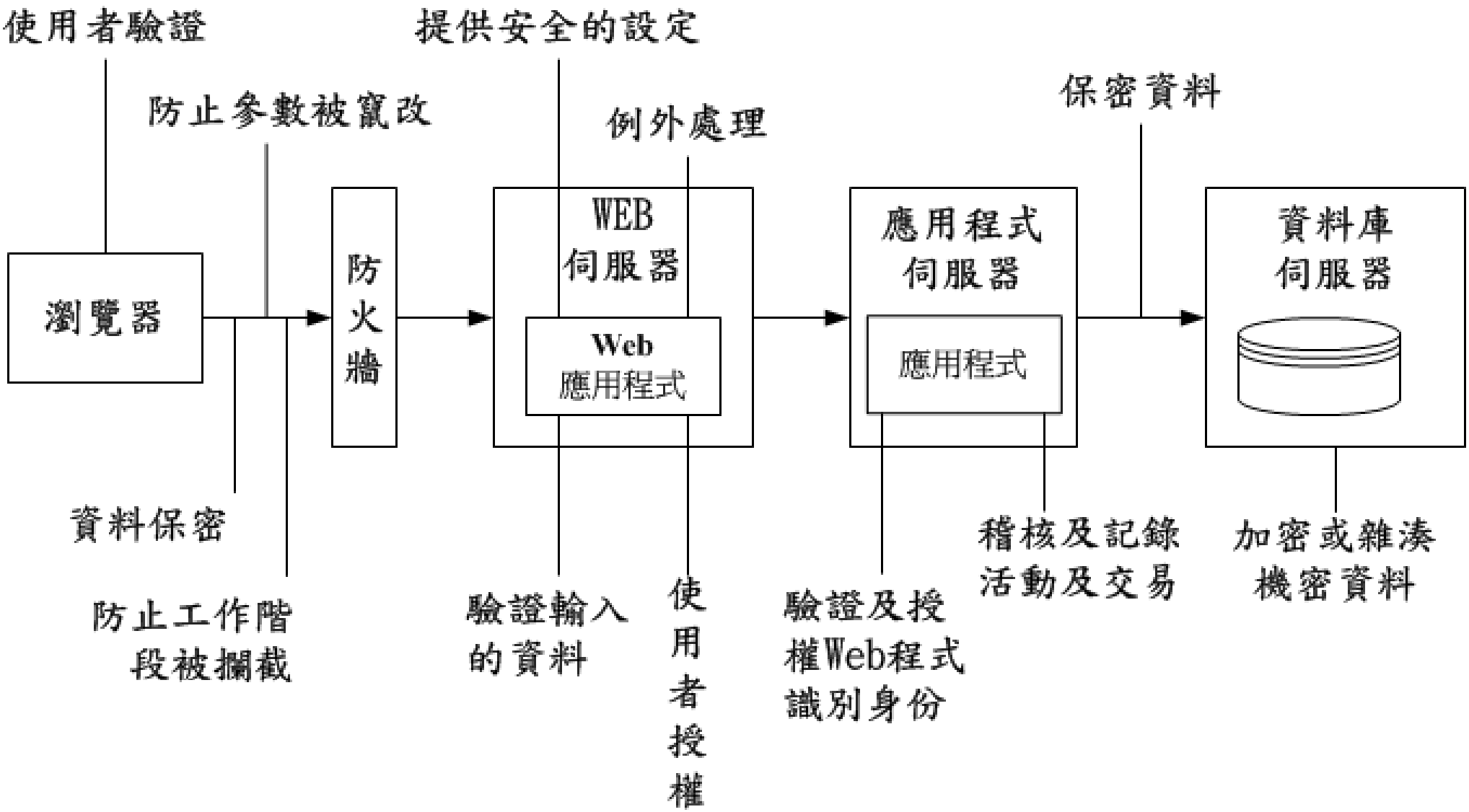
授課大綱

- 為何要重視安全的程式碼撰寫
- 了解已知的程式碼安全弱點
- **Web 網站安全的基本觀念**
- 如何通過 Fortify 檢測說明
- 常見安全的弱點修復說明
- **Q&A**

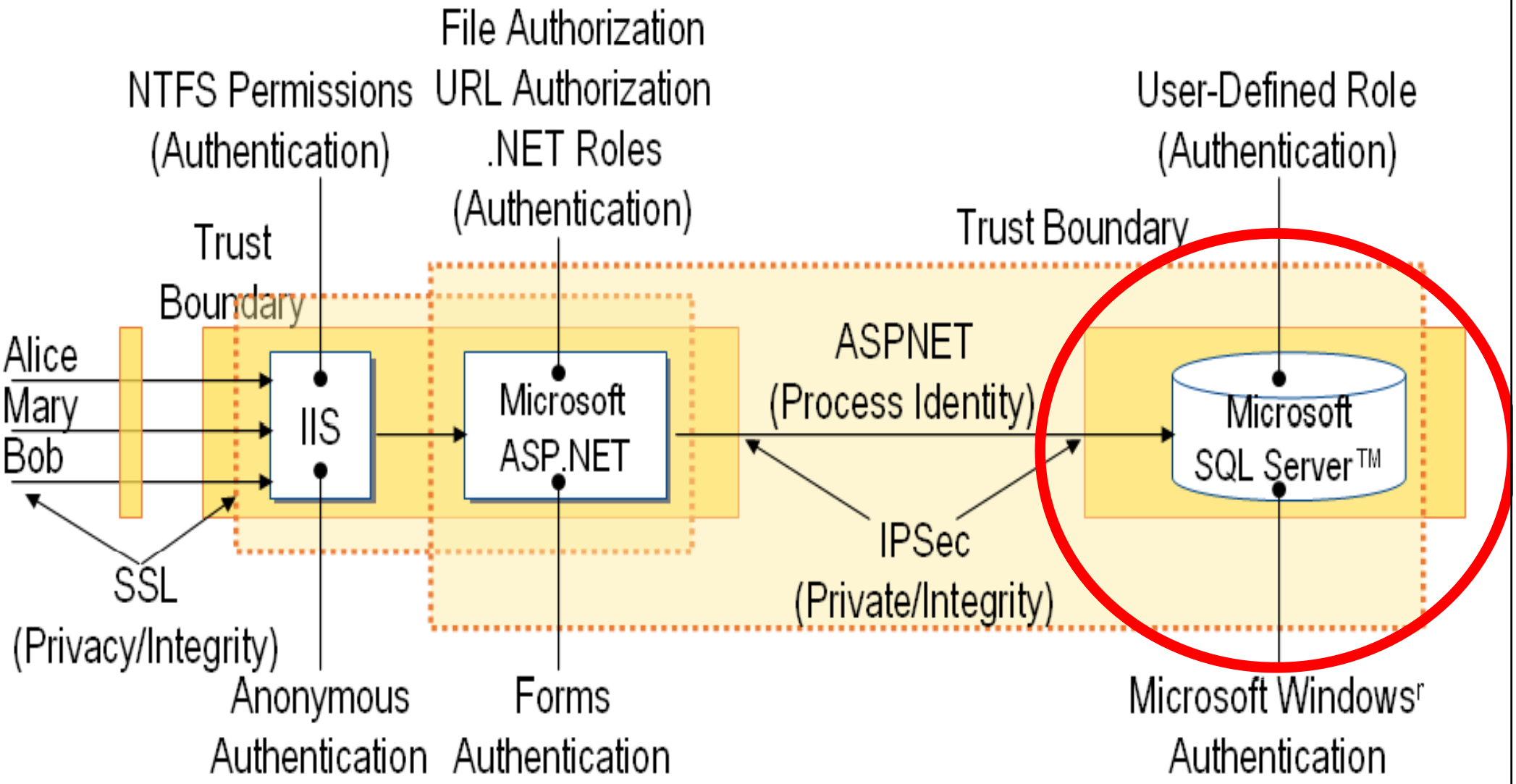
網站安全最基本的三個區域防禦



在設計 Web 安全網站必須面對的問題



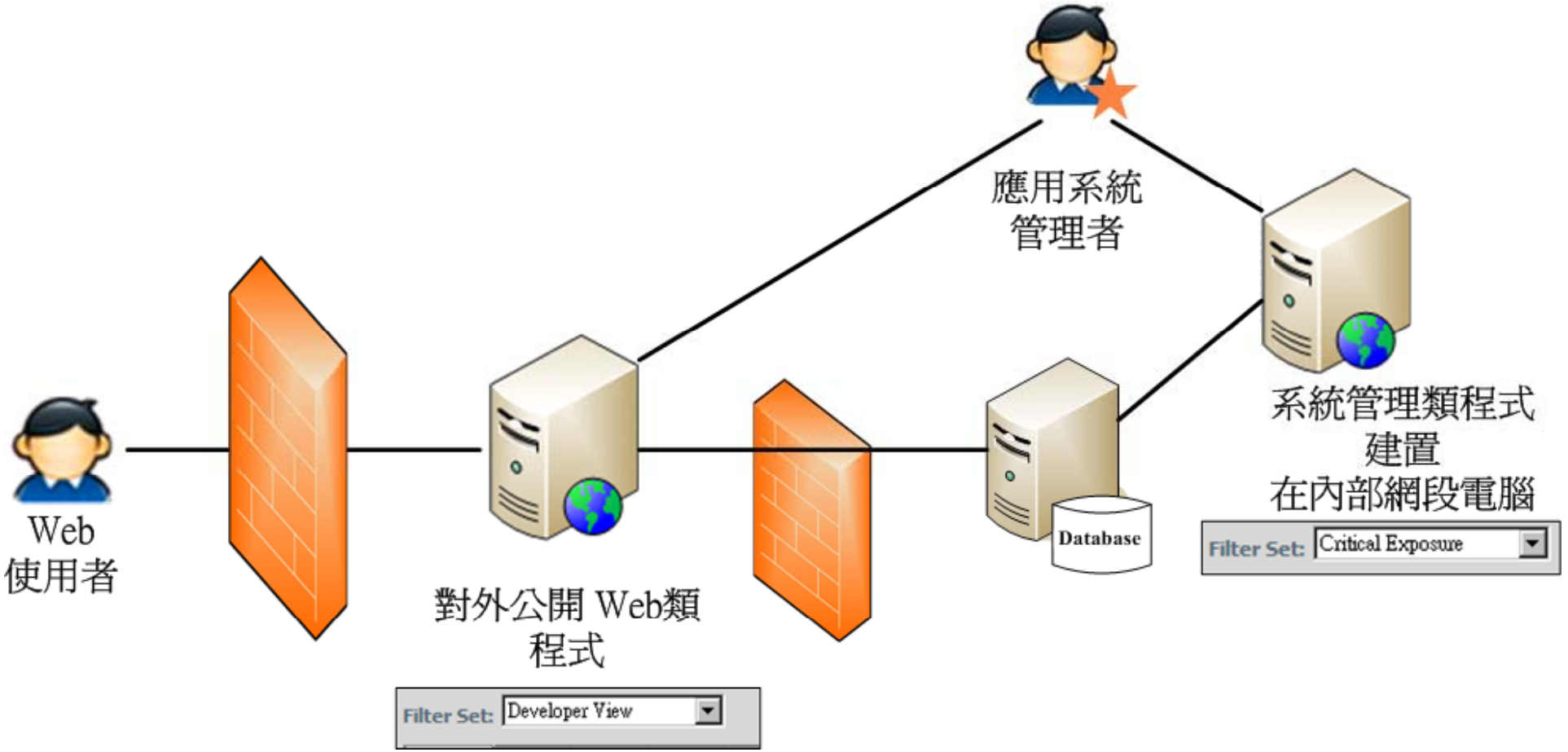
Web 應用程式權限控管 Plan - Key Item



Key Item：禁止讀取 MS SQL System Table

系統資料表	所屬的資料庫	功能用途
syslogins	**master**	能夠連線至 SQL Server 的登入帳號即是記錄於系統資料表 sysxlogins 中，亦即 syslogins 中的每一筆資料記錄即代表一個能夠連結至 SQL Server 的登入帳號。如果您需要去存取 syslogins 中的資訊，請透過系統檢視表 sys.server_principals 來完成。
sysmessages	**master**	系統資料表 sysmessages 中的每一筆資料記錄即代表 SQL Server 能夠傳回的系統錯誤或警告。如果您需要去存取 syslogins 中的資訊，應透過系統檢視表 sys.messages 來完成。
sysdatabases	**master**	系統資料表 sysdatabases 中的每一筆資料記錄即代表 SQL Server 中的每一個資料庫。如果您需要去存取 syslogins 中的資訊，應透過系統檢視表 sys.databases 來完成。
sysusers	**所有的資料庫**	系統資料表 sysusers 中的每一筆資料記錄，即代表所屬資料庫中的每一個 Windows 使用者、Windows 群組、SQL Server 使用者或 SQL Server 角色。如果您需要去存取 syslogins 中的資訊，請透過系統檢視表 sys.database_principals 來完成。
sysobjects	**所有的資料庫**	系統資料表 sysobjects 中的每一筆資料記錄，即代表所屬資料庫中的每一個物件。如果您需要去存取 syslogins 中的資訊，應透過系統檢視表 sys.objects 來完成。

Web 應用程式佈署架構的安全考量



授課大綱

- 為何要重視安全的程式碼撰寫
- 了解已知的程式碼安全弱點
- Web 網站安全的基本觀念
- 如何通過 Fortify 檢測說明
- 常見安全的弱點修復說明
- Q&A

如何通過 Fortify 檢測說明

1. 使用已知的安全程式撰寫函式
2. 針對自訂檢核函式，撰寫通過的客製化規則

使用已知的安全程式撰寫函式

[V] 安全的寫法 (撰寫使用參數化 SQL 語法)

```
string userName = ctx.getAuthenticatedUserName();
conn = new SqlConnection(_ConnectionString);
conn.Open();

SqlCommand query = new SqlCommand(
"SELECT * FROM items WHERE itemname=@ItemName
AND owner=@OwnerName", conn);

query.Parameters.AddWithValue("@ItemName", ItemName.Text);
query.Parameters.AddWithValue("@OwnerName", userName);

SqlDataReader objReader = objCommand.ExecuteReader();
```

針對自訂檢核函式，撰寫通過的客製化規則

Module1.vb

Module1 (宣告)

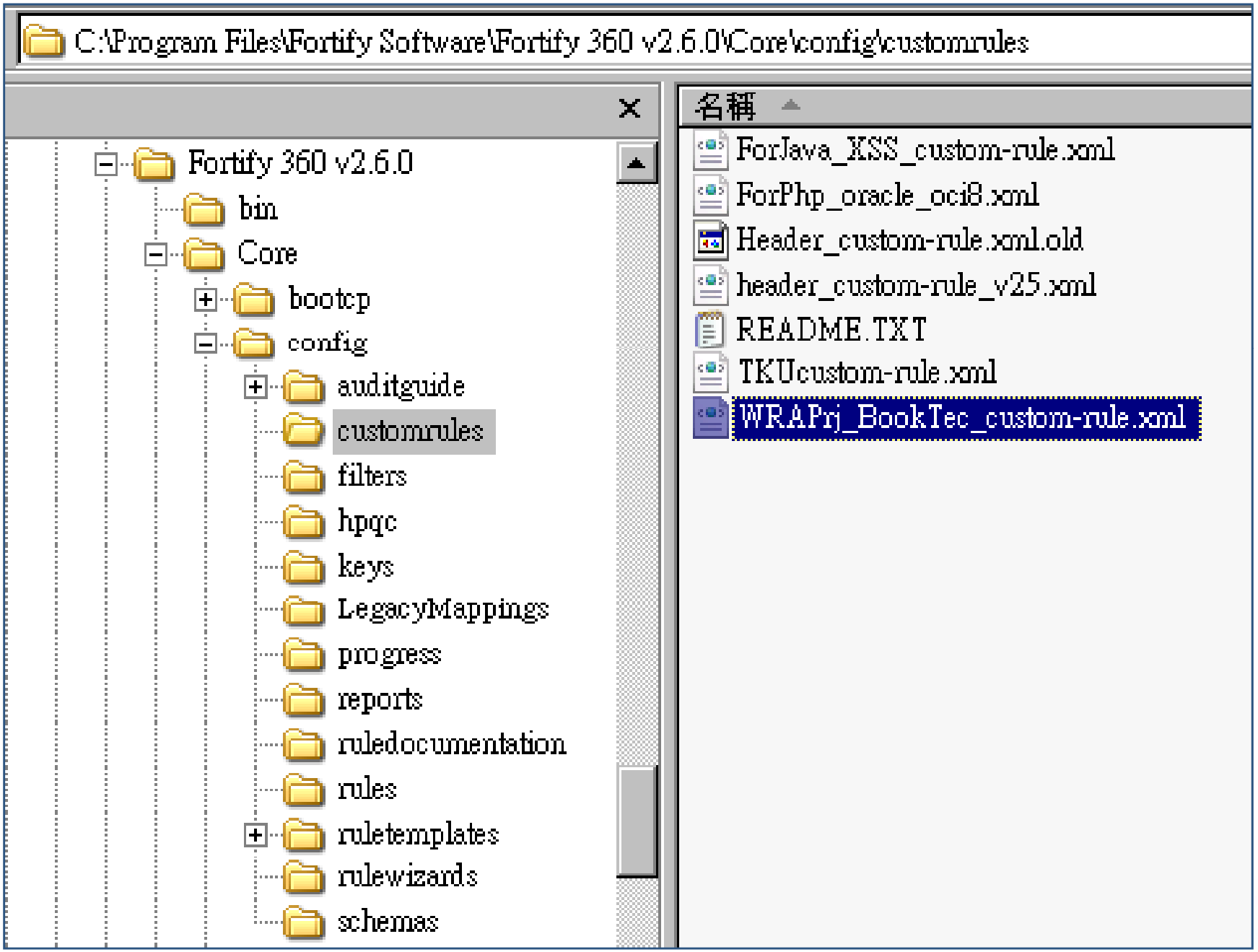
```
19  Function secure_PathManipulation(ByVal strPathFile As String) As String
20
21    '請使用  VB.Net System.Text.RegularExpressions.Regex 物件
22    '或其他安全函式()
23    '在此處撰寫您的程式路徑安全的白名單字元檢查邏輯程式.
24
25        Dim whitelist As String = "[^0-9a-zA-Z\/_\.-]"
26        Dim pattern As Regex = New Regex(whitelist)
27
28        If pattern.IsMatch(strPathFile) Then
29          If strPathFile.IndexOf("\\192.168.100.28") > -1 Or strPathFile.IndexOf("\tempPDF\") > -1 Or strPathFile.IndexOf("\bPDF\") > -1 Then
30            Return strPathFile
31          Else
32            Return ""
33          End If
34        Else
35          Return ""
36            End If
37
38        Return strPathFile
39  End Function
40
```

針對自訂檢核函式，撰寫通過的客製化規則

WRAPrj_BookTec_custom-rule.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <RulePack xmlns="xmlns://www.fortifysoftware.com/schema/rules">
3    <RulePackID>ABF6BB50-7DEE-4F36-A14C-A01F24CBE65E</RulePackID>
4    <SKU>WRAPrj_BookTec_CustomRule</SKU>
5    <Name>WRAPrj_BookTec_CustomRule</Name>
6    <Version>1.0</Version>
7    <Description><![CDATA[]]></Description>
8    <Rules version="3.8">
9      <RuleDefinitions>
10       <DataflowCleanseRule formatVersion="3.8" language="dotnet">
11         <RuleID>A6063DB8-3844-47F8-B651-AC48F75E5C3D</RuleID>
12         <TaintFlags>+VALIDATED_PATH_MANIPULATION</TaintFlags>
13         <FunctionIdentifier>
14           <NamespaceName>
15             <Pattern>.*</Pattern>
16           </NamespaceName>
17           <ClassName>
18             <Pattern>Module1</Pattern>
19           </ClassName>
20           <FunctionName>
21             <Pattern>secure_PathManipulation</Pattern>
22           </FunctionName>
23           <ApplyTo implements="true" overrides="true" extends="true"/>
24         </FunctionIdentifier>
25         <OutArguments>return</OutArguments>
26       </DataflowCleanseRule>
27     </RuleDefinitions>
28   </Rules>
29 </RulePack>
30
```

通過的客製化規則放到指定的目錄



授課大綱

- 為何要重視安全的程式碼撰寫
- 了解已知的程式碼安全弱點
- Web 網站安全的基本觀念
- 如何通過 Fortify 檢測說明
- 常見安全的弱點修復說明
- Q&A

修復弱點之前要先了解專有名詞

Fortify 專有名詞說明

Source / Sink

安全弱點修復說明

Source
程式碼區段

Sink
程式碼區段

CCProcessing.cs, line 30 (SQL Injection)

Fortify Priority:	Low	Folder	Info
Kingdom:	Input Validation and Representation		

Abstract: 在CCProcessing.cs行中的第30行，方法GetSavedCC()呼叫使用未經驗證的輸入所建立的SQL查詢。此呼叫可讓攻擊者修改敘述的意義或是執行任意的SQL指令。

Sink: CCProcessing.cs:30

```
28  using(SqlConnection myConnection = new SqlConnection(ConfigurationSettings.AppSettings["ConnectionString"]))
29  {
30   using(SqlCommand myCommand = new SqlCommand(cmd, myConnection))
31   {
32    myConnection.Open();
```

CCProcessing.cs, line 54 (SQL Injection)

Fortify Priority:	High	Folder	Hot
Kingdom:	Input Validation and Representation		

Abstract: 在CCProcessing.cs行中的第54行，方法StoreCCNum()呼叫使用未經驗證的輸入所建立的SQL查詢。此呼叫可讓攻擊者修改敘述的意義或是執行任意的SQL指令。

Source: CheckOut.aspx.cs:153 System.Web.UI.WebControls.TextBox.get_Text()

```
151
152   if (EnterCCNum.Checked && RememberCCNum.Checked)
153    Components.CCProcessing.StoreCCNum(customerId, CCNum.Text);
154
155         // Place the order
```

Sink: CCProcessing.cs:54 System.Data.SqlClient.SqlCommand()

```
52  using(SqlConnection myConnection = new SqlConnection(ConfigurationSettings.AppSettings["ConnectionString"]))
53  {
54   SqlCommand myCommand = new SqlCommand(cmd, myConnection);
55
56   myConnection.Open();
```

專有名詞說明

- **Source**

  有安全風險的來源變數

  (網頁輸入, 資料庫讀出, 傳遞參數, 外部 XML 檔案, …)

- **Sink**

  安全漏洞事件可能會發生的程式碼列

  (Fortify 監控的各種語言有安全風險的函式或物件方法)

- **Tainted Input**

  被有安全風險的來源變數，傳染的變數或屬性

.Net 程式碼片段範例

```
private void cmdLogin_Click(object sender, System.EventArgs e)
        {
            Label lblMsg = null;
            try
            {
                string txtUser = Request.Form["txtUsername"];
                string txtPassword = Request.Form["txtPassword"];
                string strCnx = "server=localhost;database=northwind;uid=sa;pwd=;";
                SqlConnection cnx = new SqlConnection(strCnx);
                cnx.Open();

                string strQry = "SELECT Count(*) FROM Users WHERE UserName='" +
                txtUser + "' AND Password='" + txtPassword + "'";
                int intRecs;

                SqlCommand cmd = new SqlCommand(strQry, cnx);
                intRecs = (int)cmd.ExecuteScalar();

                if (intRecs > 0)
                {
                    FormsAuthentication.RedirectFromLoginPage(txtUser, false);
                    cnx.Close();
                }
                else
                {
                    lblMsg.Text = "Login attempt failed.";
                }
            }
            catch (Exception err)
            {
                Response.Write(err.StackTrace.ToString());
            .....
```

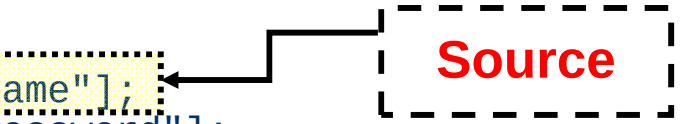
Source、Sink 說明

```
private void cmdLogin_Click(object sender, System.EventArgs e)
        {
            Label lblMsg = null;
            try
            {
                string txtUser = Request.Form["txtUsername"];
                string txtPassword = Request.Form["txtPassword"];
                string strCnx = "server=localhost;database=northwind;uid=sa;pwd=;";
                SqlConnection cnx = new SqlConnection(strCnx);
                cnx.Open();

                string strQry = "SELECT Count(*) FROM Users WHERE UserName='" +
                txtUser + "' AND Password='" + txtPassword + "'";
                int intRecs;

                SqlCommand cmd = new SqlCommand(strQry, cnx);
                intRecs = (int)cmd.ExecuteScalar();

                if (intRecs > 0)
                {
                    FormsAuthentication.RedirectFromLoginPage(txtUser, false);
                    cnx.Close();
                }
                else
                {
                    lblMsg.Text = "Login attempt failed.";
                }
            }
            catch (Exception err)
            {
                Response.Write(err.StackTrace.ToString());
            .....
```



Source、Sink 說明

```
private void cmdLogin_Click(object sender, System.EventArgs e)
        {
            Label lblMsg = null;
            try
            {
                string txtUser = Request.Form["txtUsername"];
                string txtPassword = Request.Form["txtPassword"];
                string strCnx = "server=localhost;database=northwind;uid=sa;pwd=;";
                SqlConnection cnx = new SqlConnection(strCnx);
                cnx.Open();

                string strQry = "SELECT Count(*) FROM Users WHERE UserName='" +
                txtUser + "' AND Password='" + txtPassword + "'";
                int intRecs;

                SqlCommand cmd = new SqlCommand(strQry, cnx);
                intRecs = (int)cmd.ExecuteScalar();

                if (intRecs > 0)
                {
                    FormsAuthentication.RedirectFromLoginPage(txtUser, false);
                    cnx.Close();
                }
                else
                {
                    lblMsg.Text = "Login attempt failed.";
                }
            }
            catch (Exception err)
            {
                Response.Write(err.StackTrace.ToString());
            .....
```

Source

Tainted input assigned

txtUser
沒有函式安檢
直接指定
可能有安全風險
Tainted input

Source、Sink 說明

```
private void cmdLogin_Click(object sender, System.EventArgs e)
        {
            Label lblMsg = null;
            try
            {
                string txtUser = Request.Form["txtUsername"];
                string txtPassword = Request.Form["txtPassword"];
                string strCnx = "server=localhost;database=northwind;uid=sa;pwd=;";
                SqlConnection cnx = new SqlConnection(strCnx);
                cnx.Open();

                string strQry = "SELECT Count(*) FROM Users WHERE UserName='" +
                txtUser + "' AND Password='" + txtPassword + "'";
                int intRecs;

                SqlCommand cmd = new SqlCommand(strQry, cnx);
                intRecs = (int)cmd.ExecuteScalar();

                if (intRecs > 0)
                {
                    FormsAuthentication.RedirectFromLoginPage(txtUser, false);
                    cnx.Close();
                }
                else
                {
                    lblMsg.Text = "Login attempt failed.";
                }
            }
            catch (Exception err)
            {
                Response.Write(err.StackTrace.ToString());
            .....
```

Source

Tainted input assigned

Tainted input txtUser used in assignment

Source、Sink 說明

```
private void cmdLogin_Click(object sender, System.EventArgs e)
        {
            Label lblMsg = null;
            try
            {
                string txtUser = Request.Form["txtUsername"];
                string txtPassword = Request.Form["txtPassword"];
                string strCnx = "server=localhost;database=northwind;uid=sa;pwd=;";
                SqlConnection cnx = new SqlConnection(strCnx);
                cnx.Open();

                string strQry = "SELECT Count(*) FROM Users WHERE UserName='" +
                txtUser + "' AND Password='" + txtPassword + "'";









            }
            catch (Exception err)
            {
                Response.Write(err.StackTrace.ToString());
            .....
```

Source

Tainted input assigned

安全事件發生的必要條件

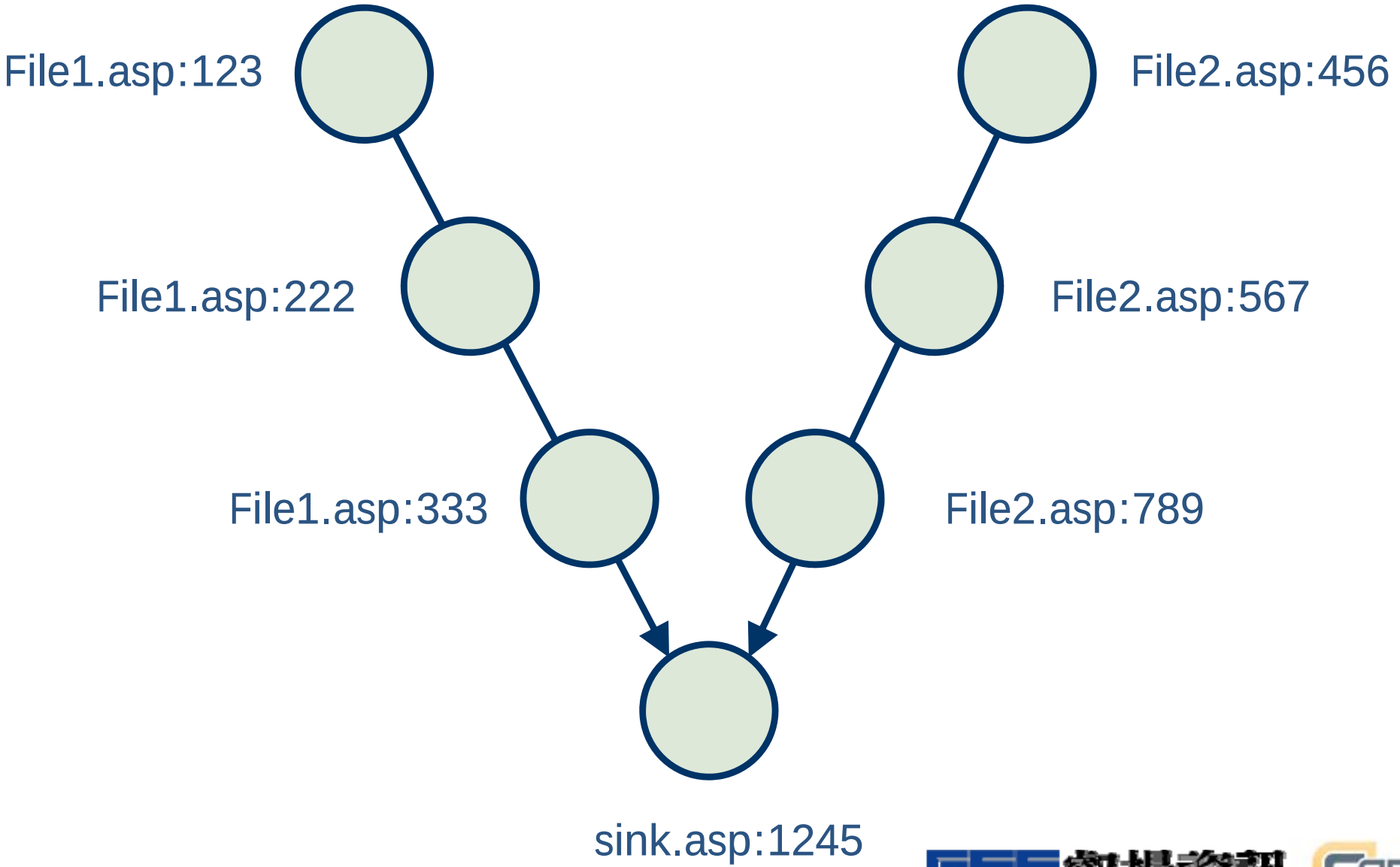
1. 有 Source And 有 Sink ＞安全事件才會有可能發生
2. 有 Source And 沒有 Sink ＞不會發生但未來仍有潛在風險
3. 有 Sink And 沒有 Source ＞不會發生但未來仍有潛在風險

(1)

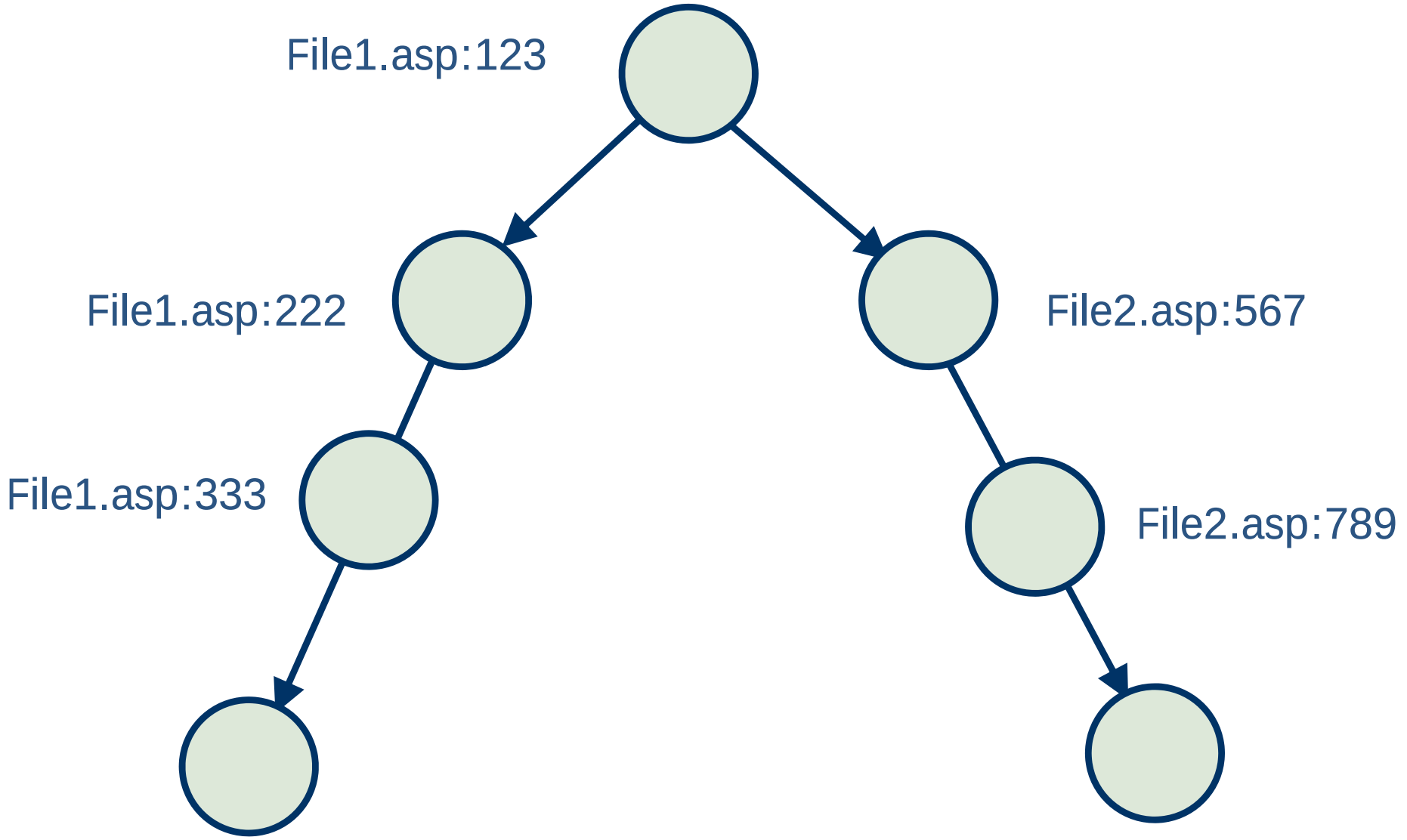
(2)

(3)

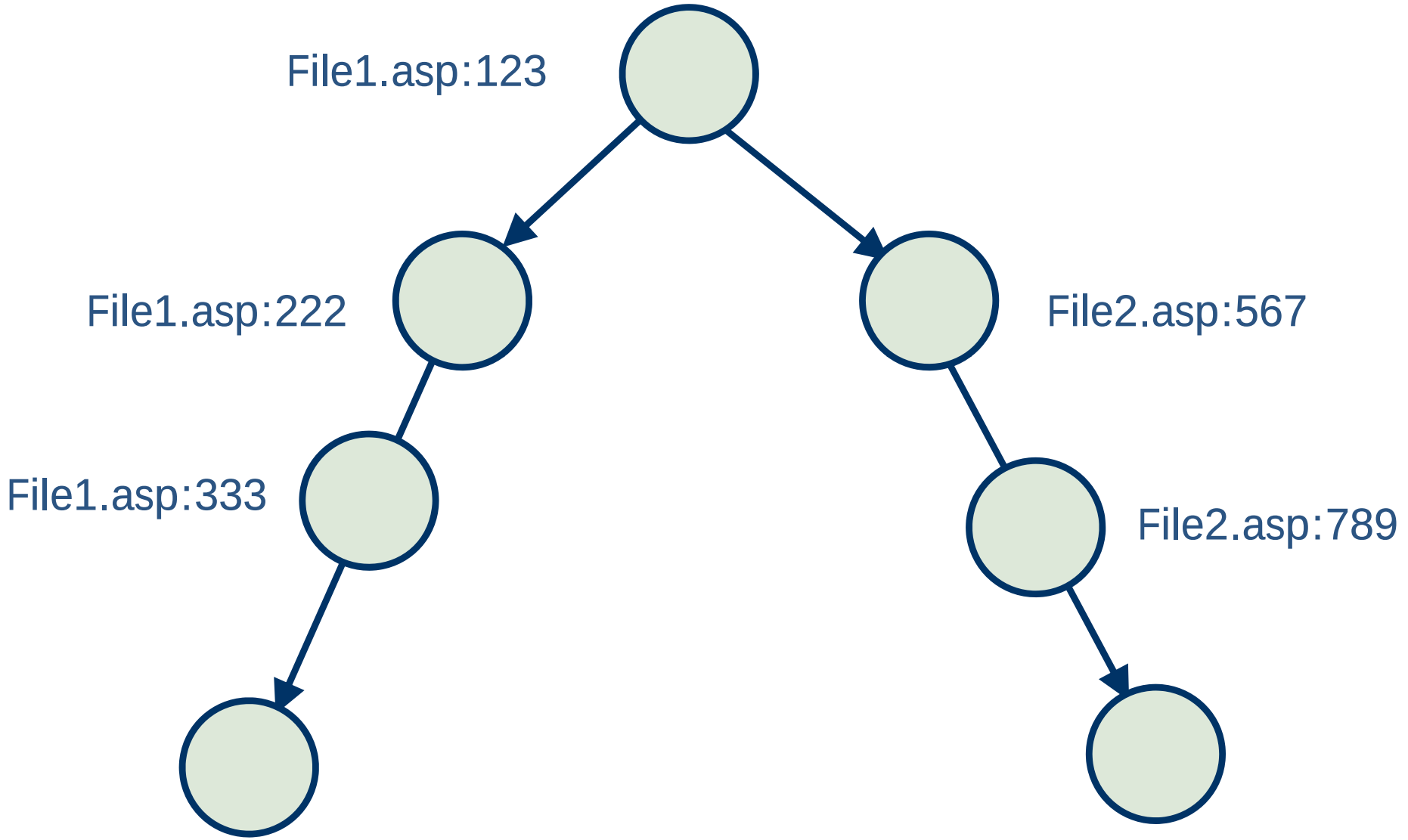
問題收斂模式： Many Sources to One Sink



問題收斂模式： One Source to Many Sinks

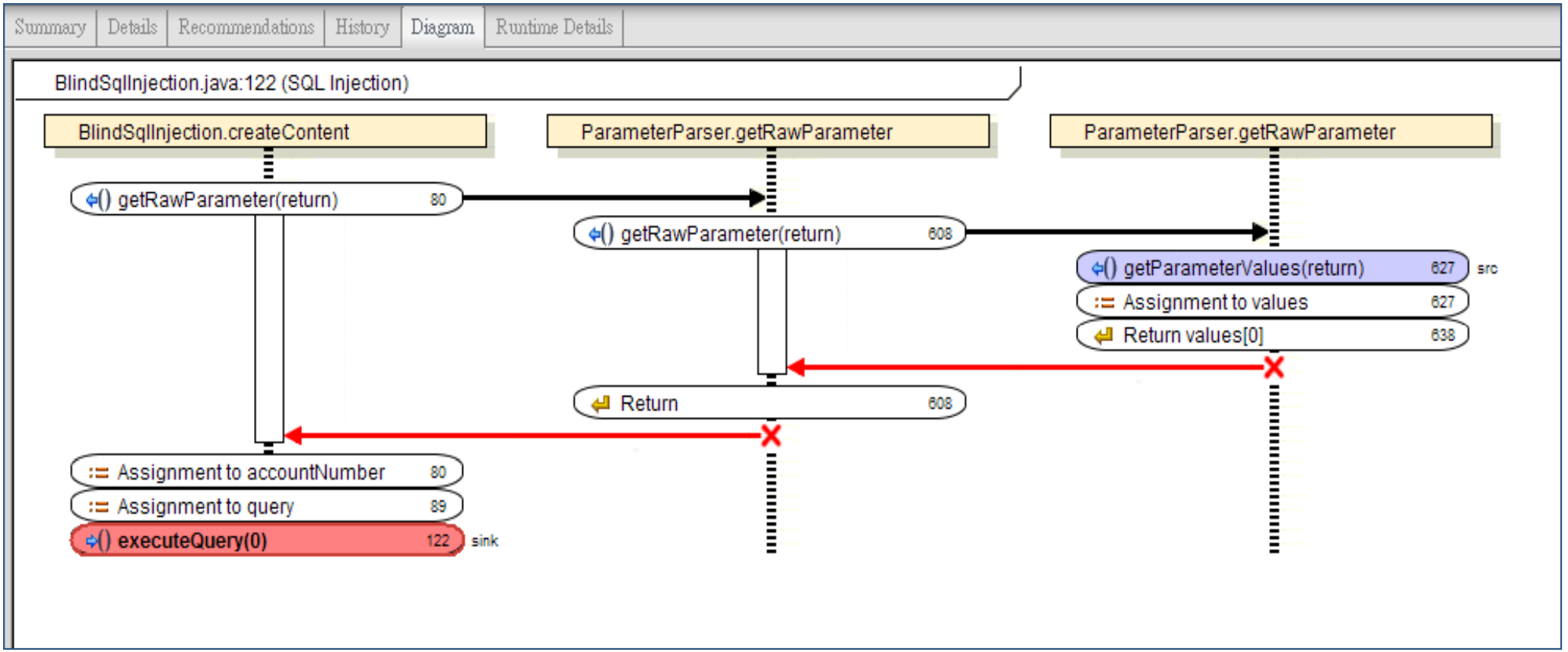


問題收斂模式： One Source to Many Sinks



叡揚資訊

選定程式碼修復的下手點 Source / Sink



休息一下 15 分鐘 …

常見安全的弱點修復說明

1. SQL Injection
2. Cross Site Scripting (XSS)
3. HTTP Response Splitting
4. Command Injection
5. Path Manipulation
6. Cross Site Request Forgery (CSRF)
7. Password Management
8. Race Conditions
9. Error Handling
10. Misconfiguration