網頁應用程式攻擊工具

葉治宏 林盈達 李佳穎
國立交通大學資訊工程學系
Email: chyeh.cs01g@nctu.edu.tw; ydlin@cs.nctu.edu.tw; neko@nctu.edu.tw
November 30, 2012

摘要

當測試網頁應用防火牆 (Web Application Firewall, WAF) 之防禦功能時,經常須對該 WAF 所防護的後端網頁應用程式 (Web Application) 進行攻擊，本研究將針對數種攻擊網頁應用程式之常用工具進行介紹並實作新的攻擊方法。首先，我們針對 Web Application 的各種弱點進行分析與討論，並對現行兩種弱點公告及評分系統進行比較。此外，亦針對幾種常見的網頁應用程式弱點，以實例說明該攻擊如何執行及運作。Metaspliot 是一種擁有許多攻擊模組的平台,本文中，我們說明 Metaspliot 之主要操作原理並與其他相似的攻擊平台進行比較。最後，我們在 Metaspliot 平台上實作兩個新的攻擊模組,並對此攻擊模組之有效性進行測試。

關鍵字: 網頁應用防火牆、弱點、攻擊工具、OWASP Top 10、Metaspliot

1. 弱點介紹

當需要對網頁應用防火牆 (Web Application Firewall, WAF) 進行防禦功能測試時，則需對該 WAF 所防護的後端之網頁應用程式 (Web Application) 進行攻擊，如何收集並開發各種攻擊工具，是一個富有實用價值的議題。首先，我們必須先瞭解現今網頁應用程式具有哪些弱點。本節將介紹 OWASP [1] 與 MITRE [2]，兩個國際組織收集且公告之網頁應用程式的主要弱點。當深入了解各弱點後，將進一步從 OWASP 公告的弱點中選取其中兩項，再以實作攻擊工具的方式來說明這兩項弱點的攻擊行為如何運作。

OWASP

OWASP (The Open Web Application Security Project) 是一個旨在改善軟體安全的非營利組織，為了提高企業對應用程式安全議題的重視程度，OWASP 從 2003 年開始發布 OWASP Top 10，其內容為對於企業具有最嚴重威脅的十種弱點。OWASP Top 10 曾在 2004 年與 2007 年修改過，目前最廣為人知的弱點就是 OWASP 在 2010 年所公布的十種網頁應用程式弱點類型 (OWASP Top 10 for 2010) [3]，如表一。

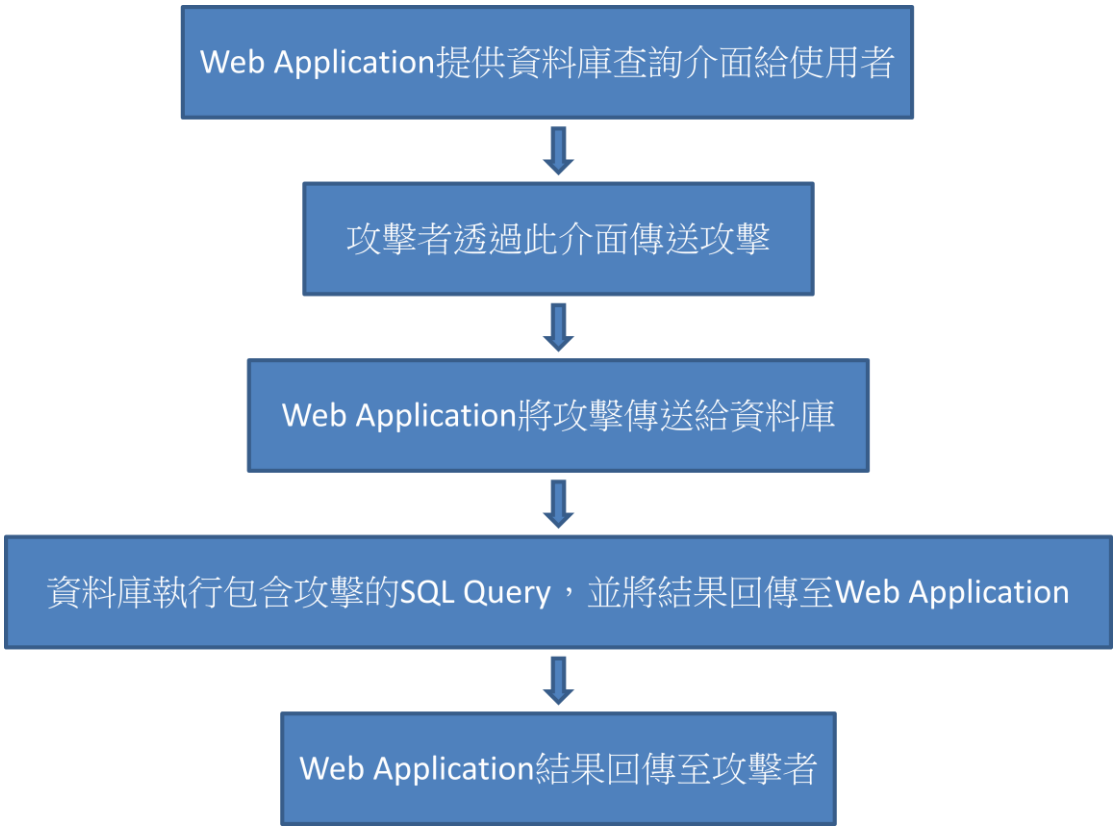
A1: Injection
A2: Cross-Site Scripting (XSS)
A3: Broken Authentication and Session Management
A4: Insecure Direct Object References
A5: Cross-Site Request Forgery (CSRF)
A6: Security Misconfiguration
A7: Insecure Cryptographic Storage
A8: Failure to Restrict URL Access
A9: Insufficient Transport Layer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

表一　OWASP Top 10 for 2010

表一所列舉的的十種弱點是目前各企業面臨最嚴重的資安風險。為了使讀者更了解網頁應用程式攻擊如何運作，本研究將以表中第一項 Injection 及第二項 Cross-Site Scripting 兩種攻擊做為範例以進行說明。

Injection

為了方便說明，我們以 Injection 類攻擊中最常見的 SQL Injection 攻擊為例，SQL Injection 的攻擊流程如圖一所示。



圖一　SQL injection 攻擊流程圖

遭受 SQL Injection 攻擊的網頁應用程式通常具有供使用者使用的資料庫查詢介面。最常見的使用者介面為表單 (Form)，使用者可藉由表單輸入所欲查詢的參數，透過網頁應用程式對資料庫進行查詢(Query)。攻擊者可透過在表單中輸入某些帶有特定字元或指令，讓傳至資料庫的查詢語法產生開發過程中未預期的結果。舉例說明，如果攻擊者在表單中輸入參數 Jack 所產生的資料庫查詢語法為：

select * from users WHERE name = 'Jack'，

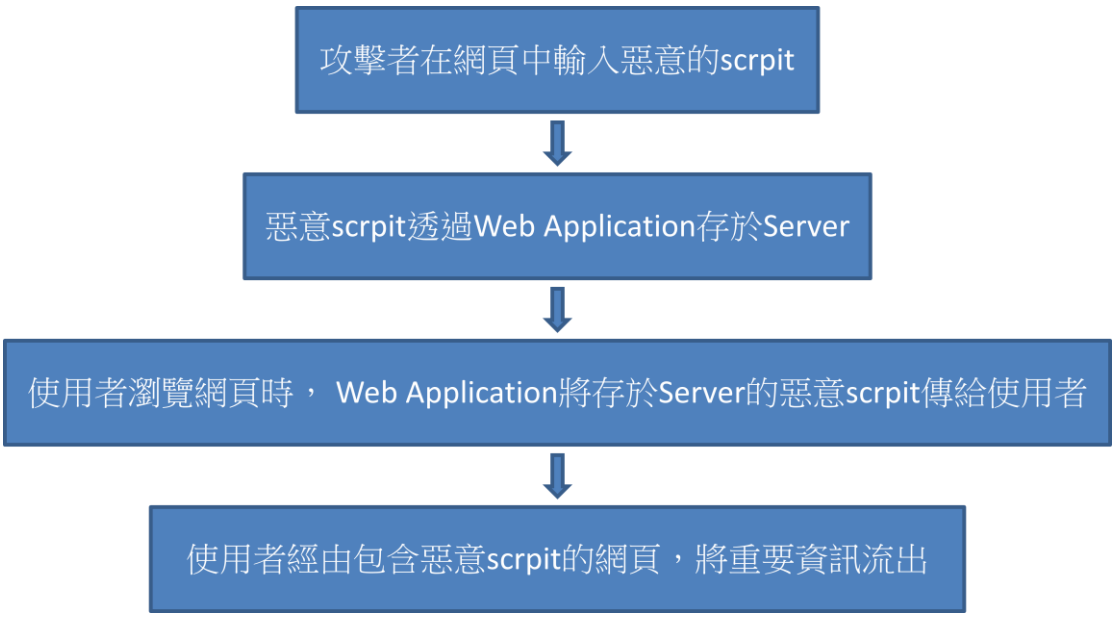
則攻擊者只要將表單中的參數改為'Jack' or '1'='1，對資料庫所產生的查詢語法將變成：

select * from users WHERE name = 'Jack' or '1'='1'。

此查詢將會因為 WHERE 條件式的條件'Jack' or '1'='1'恆等於 true，而回傳資料庫中 user 此表內的所有項目。

Cross-Site Scripting

我們以 Cross-Site Scripting 類攻擊中最常見的 Stored Cross-Site Scripting 攻擊為例進行說明，Stored Cross-Site Scripting 的攻擊流程如圖二所示。



圖二 Stored Cross-Site Scripting 攻擊流程圖

Stored Cross-Site Scripting 通常發生於網頁應用程式提供儲存功能供使用者輸入內容，且將內容提供給其他使用者進行瀏覽的情況。攻擊者通常透過網頁應用程式所提供的介面輸入含有攻擊行為的惡意語法，而這些語法會因為網頁應用程式本身的功能儲存於伺服器端。當任何一個使用者透過該網頁應用程式瀏覽攻擊者先前儲存於伺服器端的惡意語法，則該使用者將遭受攻擊。舉例說明，攻擊者在某個具有留言版功能的網頁應用程式留下了具有竊取使用者 cookie

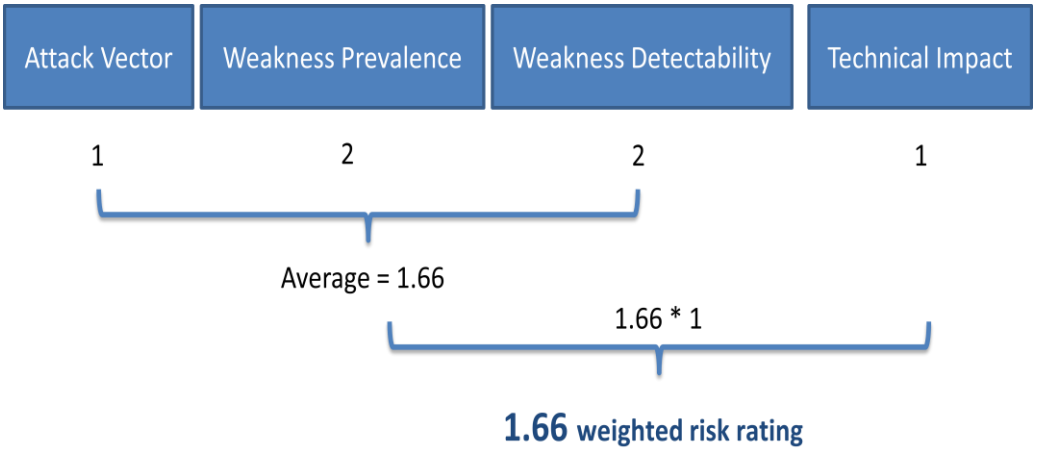
的語法，則當其他使用者瀏覽到攻擊者所發布的留言時，那些含有攻擊的語法將會竊取使用者的 cookie。

OWASP 對於弱點有一套評分方法，OWASP 在 2010 年所公布的最新十種網頁應用程式弱點類型，即以此方法進行評比並歸納出十種最嚴重的弱點。此評分方法主要有四個因子(factor)。第一項為 Attack Vector，表示攻擊者是否很容易針對該弱點進行攻擊。第二項為 Weakness Prevalence， 表示此弱點是否很常出現於網頁應用程式當中。第三項為 Weakness Detectability，表示 弱點是否容易被攻擊者發現。第四項為 Technical Impact，表示網頁應用程式受攻擊後，所受影響是否嚴重。對於每個弱點，OWASP 在每項因子上都依照輕重程度分別給 1~3 分。若在該因子上程度較為嚴重的話給 1 分；若中等則給 2 分；程度輕微的話則給 3 分 (如表二)。

	Attack Vector	**Weakness Prevalence**	**Technical Impact**	**Weakness Detectability**
1 分	Easy	Widespread	Easy	Severe
2 分	Average	Common	Average	Moderate
3 分	Difficult	Uncommon	Difficult	Minor

表二 OWASP 對於弱點的評分標準

OWASP 將 Attack Vector、Weakness Prevalence 與 Weakness Detectability 三項因子的分數加總取平均後，再與 Technical Impact 的分數相乘，最後得到的結果就是 OWASP 針對該弱點所評估出的風險等級 (Risk Rating)。當風險等級的數字越小，則表示該風險越嚴重。舉例說明，若 Injection 攻擊的 Attack Vector 評分為 1、Weakness Prevalence 為 2、Weakness Detectability 為 3。此三項因子的平均值為 1.66。若 Injection 的 Technical Impact 評分為 1，將 1.66 與 1 相乘的結果為 1.66，即為 Injection 評估後的風險等級分數 (如圖三)。



圖三 OWASP 評分方式

MITRE

MITRE 是美國聯邦政府所贊助的一家公司。除了 OWASP 在 2010 年公告的十種弱點外，MITRE 也先後制定了兩種弱點公告系統。這兩種公告弱點系統分別為 CVE (Common Vulnerabilities and Exposures) [4] 與 CWE (Common Weakness Enumeration) [5]。MITRE 先於 1999 年發表了 CVE 系統，CVE 列舉出已知的安全漏洞或者已經暴露出來的弱點，並給定一個共同使用的名稱。共同的安全漏洞名稱，可以幫助使用者在各自獨立的各種弱點資料庫與漏洞評估工具中共用資訊。CVE 就像是安全議題中的"關鍵字"，如果在一個弱點報告中指明一個弱點含有 CVE 名稱，便能快速地在其它與 CVE 相容的資料庫中找到相應的解決方法。然而 CVE 只粗略地分類，無法因應軟體開發者的需求。因此 MITRE 在 2005 年開始發展新的弱點與攻擊的分類概念，目的在於定義軟體缺陷 (Software Weakness)。軟體的缺陷 (Weakness) 與弱點 (Vulnerability) 是不同的，缺陷會導致許多弱點，而弱點將會直接讓攻擊者用於攻擊。MITRE 在 CVE 的基礎上，蒐集許多會產生弱點的真實程式碼，並以缺陷的角度制定了新的分類方法，針對軟體弱點的 CWE 便因此發展出來。

上述兩個系統各自有其針對弱點的評分系統。CVE 的評分系統為 CVSS (Common Vulnerability Scoring System)，而 CWE 的評分系統則是 CWSS (Common Weakness Scoring System)。這兩套評分系統的計算方法與前述 OWASP 評分方法的計算方法類似，然 CVSS 與 CWSS 各自設有許多評分的因子。對於每種弱點或是缺陷，其計算法式也都先給定每項因子分數，再將每項因子的分數乘以權重後加總得到該弱點或缺陷的評分。因此，這兩套評分系統的基本概念非常相似，僅有部分的評分因子不同而已。

表三把 CWSS 使用的 18 項評分因子分為三大類，與風險高低相關的評分因子會被歸類於 Base Finding Group，與攻擊難易程度相關的評分因子歸類於 Attack Surface Group，而與關某特定環境下相關的評分因子則歸類於 Environmental Group，評分之計算公式如圖四所示。由圖四得知，先將三類評分因子各自的分數計算出來後，再將此三個分數相乘，最後便能得到評比的總分。

CVSS 與 CWSS 評分項目的比較表詳見於表四中，表四將兩種系統相似的評分因子做了對照。舉例說明，CWSS 的評分因子 Deployment Scope，在 CVSS 中也有一些意義相似的評分因子，分別為 Access Complexity (AC) 及 Target Distribution (TD)。

CWSS score = **BaseFindingSubscore** * **AttackSurfaceSubscore** * **EnvironmentSubscore**
BaseSubscore = [(10 * TI + 5*(AP + AL) + 5*FC) * f(TI) * IC] * 4.0
AttackSurfaceSubscore = [20*(RP + RL + AV) + 20*SC + 10*IN + 5*(AS + AI)] / 100.0
EnvironmentSubscore =[(10 * BI + 3*(DI + EX) + 3*P + RE) * f(BI) * EC] / 20.0

圖四 CWSS 的評分公式 (http://cwe.mitre.org/cwss/index.html)

Metric Group	Factors
Base Finding Group	* Technical Impact (TI) * Acquired Privilege (AP) * Acquired Privilege Layer (AL) * Internal Control Effectiveness (IC) * Finding Confidence (FC)
Attack Surface Group	* Required Privilege (RP) * Required Privilege Layer (RL) * Access Vector (AV) * Authentication Strength (AS) * Authentication Instances (AI) * Level of Interaction (IN) * Deployment Scope (SC)
Environmental Group	* Business Impact (BI) * Likelihood of Discovery (DI) * Likelihood of Exploit (EX) * External Control Effectiveness (EC) * Remediation Effort (RE) * Prevalence (P)

表三 CWSS 的各項評分因子 (http://cwe.mitre.org/cwss/index.html)

CVSS	CWSS
Confidentiality Impact (C), Integrity Impact (I), Availability Impact (A), Security Requirements (CR, IR, AR), Collateral Damage Potential (CDP)	Technical Impact
Access Complexity (AC), Target Distribution (TD)	Deployment Scope
Access Vector (AV)	Access Vector
Access Complexity (AC)	Required Privilege Level
N/A	Authentication Strength
Authentication (Au)	Authentication Instances
N/A	Likelihood of Discovery
N/A	Likelihood of Exploit
Access Complexity (AC)	Interaction Requirements
Access Complexity (AC), Remediation Level (RL)	Internal Control Effectiveness (IC)
Access Complexity (AC)	External Control Effectiveness

	(EC)
Report Confidence (RC)	Finding Confidence
N/A	Remediation Effort (RE)
Exploitability (E)	N/A
Target Distribution (TD)	N/A

表四 CVSS 與 CWSS 各項評分因子之比較
(http://cwe.mitre.org/cwss/index.html)

2. 常見攻擊工具簡介

本研究已於前面章節簡介網頁應用程式這種攻擊的運作方式，當我們了解其運作原理後，本章節將繼續介紹可實現攻擊行為的各種工具。

Metasploit

Metasploit [6] 是一個相當知名的開放原碼 (Open Source) 滲透測試工具，它已內建 902 個攻擊模組，其中有 213 個是針對網路的攻擊模組。Metasploit 也提供了良好擴充性，使用者可以在該平台上，自行開發新的攻擊模組。

一般而言，若想要使用 Metasploit 來進行攻擊，大致有四個主要步驟(使用步驟參考 Metasploit Framework Users Guide Release 4.3 [7])。

步驟一：先選定一個攻擊模組。Metasploit 的使用介面為命令行 (Command Line) 模式，選定模組的指令為 use。將想要使用的攻擊模組所存放的檔案路徑做為 use 指令的參數，便能選定該攻擊模組。舉例說明，若想使用的攻擊模組為 HTTP Blind SQL Injection Scanner，所需下達的指令為：msf > use auxiliary/scanner/http/blind_sql_query

步驟二：必須事先知道此模組所需設定的各種參數。使用 show options 指令就能顯示此模組的所有參數。

步驟三，設定適當的參數。使用 set 指令便能給定參數值，set 的第一個參數為使用者想要設定的參數名稱 (參數名稱可用第二步的方法得知)，set 的第二個參數為使用者想要設定的參數值。舉例說明，若想要設定遠程主機 (Remote Host) 其 IP 位址為 192.168.20.129，則所需下達的指令即為：msf > set RHOST 192.168.20.129

步驟四，只要完成前述步驟並下達 exploit 指令便能啟動此攻擊模組。

w3af

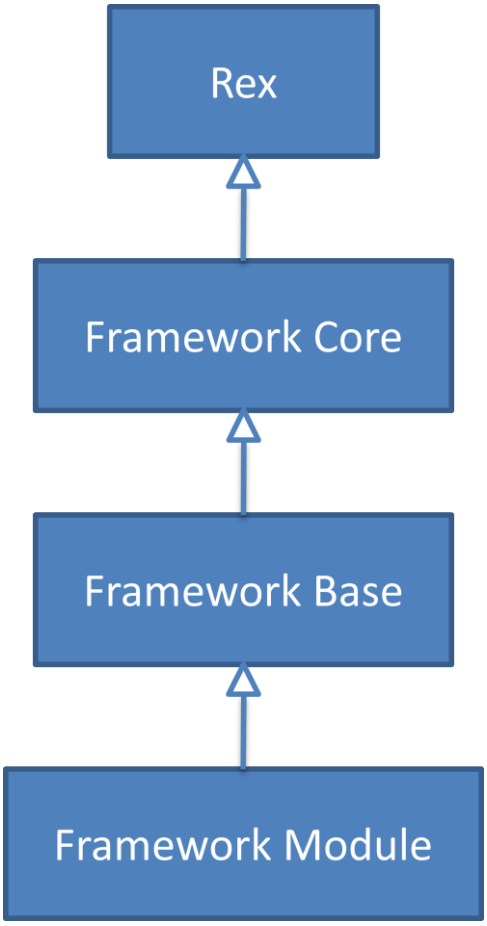
w3af 的全名為 Web Application Attack and Audit Framework [8]，是針對網頁應用程式攻擊的工具。w3af 也是一個開放原碼的平台，但目前它並未開放讓

使用者自行增加攻擊模組。此工具與 Windows 7 並不相容，因此對一般使用者而言並不方便。但值得注意的是，此工具尚在發展階段，根據該計畫在 2009 年所訂下的目標，此工具未來也會類似 Metasploit，提供使用者自行擴充攻擊模組。因此，未來 w3af 若發展成熟，它亦有可能取代許多現有的攻擊工具。

3.開發攻擊模組之框架

雖然 w3af 是針對網頁應用程式攻擊的工具，但目前它尚未開放讓使用者自行增加攻擊模組。而 Metasploit 雖然並非針對網頁應用程式進行攻擊的工具，但它卻能讓使用者可以在該平台上自行開發新的攻擊模組。因此，本研究選定 Metasploit 為開發平台，並嘗試在該平台上自行開發針對網頁應用程式攻擊的模組。我們將利用 Metasploit Framework 開發一個可對目標網頁伺服器進行目錄掃描的攻擊模組。

Metasploit Framework 是以 Ruby 語言寫成的許多模組與應用程式介面 (Application Programming Interface)，其架構圖如圖五(參考自：Metasploit 3.0 Developer's Guide [9])。



圖五 Metasploit Framework 架構圖

Metasploit 架構中最基本的部分為 Rex 函式庫，Rex 即為 Ruby Extension 之縮寫。Rex 包含了 Socket、Protocol 等較底層部分的實作，並將那些部分封裝起來。Framework Core 為架構中的第二層，此層則繼承了許多 Rex 中的物件，主要負責實做攻擊模組、Session 等可能會用到的程式介面。Framework Base 為架構中的第三層，它繼承了 Framework Core 的物件，而其設計是為了因應不同方面對 Framework Core 物件之多種需求，因而有不同的設計。架構的最上層為第四層的 Framework Module，此層架構則實做了所有的攻擊模組。

本研究於 OWASP Top 10 2010 所列舉的弱點中，選擇第一項：Injection 與第八項：Failure to Restrict URL Access 作為標的，並在 Metasploit 上針對這兩項弱點實作攻擊模組。

模組一：Injection

本模組是針對 Injection 中常見的 SQL Injection 弱點進行攻擊，在執行過程中，能造成 SQL Injection 的字串傳送給網頁應用程式伺服器進行 SQL 查詢。本模組造成 SQL Injection 的惡意指令將以字串的方式存於 Hash 資料結構當中，如圖六所示。

```
inivalstr = [
    [ 'numeric',
    " AND #{rnum}=#{rnum} "
    ],
    [ 'single quotes',
    "' AND '#{rnum}'='#{rnum}"
    ],
    [ 'double quotes',
    "\" AND \"#{rnum}\"=\"#{rnum}"
    ],
    [ 'OR single quotes uncommented',
    "' OR '#{rnum}'='#{rnum}"
    ],
    [ 'OR single quotes closed and commented',
    "' OR '#{rnum}'='#{rnum}'--"
    ],
    [ 'hex encoded OR single quotes uncommented',
    "'%20OR%20'#{rnum}'%3D'#{rnum}"
    ],
    [ 'hex encoded OR single quotes closed and commented',
    "'%20OR%20'#{rnum}'%3D'#{rnum}'--"
    ]
]
```

圖六 用來存進行 SQL Injection 的字串或字元的 Hash

在第一節所提及的字串形式，即屬於此 Hash 中第四項 OR single quotes uncommented 類型，至於其他種可能造成 SQL Injection 的字串也列舉於此 Hash 當中，在此便不一一詳細說明。

本模組利用 send_request_cgi()將 Hash 中所提及的字串形式一一傳送給網頁應用程式伺服器 (如圖七)。

```
trueres = send_request_cgi({
    'uri'       =>  datastore['PATH'],
    'vars_get'  =>  testgvars,
    'method'    =>  http_method,
    'ctype'     => 'application/x-www-form-urlencoded',
    'cookie'    => datastore['COOKIE'],
    'data'      => datastore['DATA']
}, 20)
```

圖七 此模組中負責傳送 http request 的 CGI 函式

本模組對目標網頁應用程式伺服器傳送許多 SQL Injection 的字串，當有任一組字串達到攻擊的效果，即可證明此網頁應用程式具有 Injection 類型的弱點。

模組二：Failure to Restrict URL Access

程式設計人員往往於網頁原始碼中遺留路徑資訊，將造成常見的 Failure to Restrict URL Acces 弱點。本模組針對網頁原始碼進行分析，搜尋網頁原始碼中遺留路徑資訊。藉由獲得的路徑資訊，便能對這些路徑嘗試進行瀏覽。與前面介紹的模組類似，本模組可藉由 send_request_cgi()對網頁應用程式傳送 http request(如圖八)。

```
res = send_request_cgi({
    'uri'       =>  tpath,
    'method'    => 'GET',
    'ctype'     => 'text/plain'
    }, 20)
```

圖八 此模組中負責傳送 http request 的 CGI 函式

如獲得正常的 http response，我們便能對回傳的網頁應用程式的原始碼進行分析。利用條件判斷式，搜尋原始碼中是否有遺留子目錄的資訊 (如圖九) 。圖九中，我們藉由分析網頁原始碼標籤中的內容，判斷是否存在子目錄的資訊。另一方面，我們也能藉由此方法搜尋是否有遺留父目錄的資訊 (如圖十)。圖十中，我們利用類似前述的方法，搜尋可能存有父目錄資訊內容的標籤。

```
if res.to_s.include? "<title>Index of /" and res.to_s.include? "<h1>Index of /"
```

圖九 搜尋子目路資訊地條件判斷

```
if res.to_s.include? "[To Parent Directory]</A>" and res.to_s.include? "#{tpath}</H1><hr>"
```

圖十 搜尋父目路資訊地條件判斷

4. 實驗結果

在 Metasploit 平台上開發完模組後若要執行並且進行測試，則必須將它移植進入 Metasploit Framework，再藉由 Metasploit 所提供之 Console 進行測試。若測試模組的作業系統為 Windows，則必須將所開發的新模組存於以下路徑：

C:\Users\$user\.msf4\modules\exploits

若為 Linux 作業系統，則必須將所開發的新模組存於以下路徑：

/home/$user/.msf4/modules/exploits/

我們以 OWASP Broken Web Applications Project [10] 提供的虛擬機作為測試攻擊模組之網頁應用程式伺服器。OWASP Broken Web Applications Project 為 OWASP 的相關計畫之一，在此計畫中，OWASP 提供了一個內建數個具有許多安全漏洞網頁應用程式伺服器的虛擬機器，此虛擬機器為.vmx 檔案，必須使用 VMware 開啟。以 VMware 開啟此虛擬機之後，這些網頁應用程式就會架設於該虛擬機器的 Private IP 位址。我們以 192.168.1.223 作為此網頁應用程式的位址。下面將說明本研究開發的兩個模組，分別對於此虛擬機進行測試的結果。

Injection 測試

在 OWASP Broken Web Applications Project 中有許多網頁應用程式，我們選定其中一個網頁應用程式－peruggia 作為測試目標。peruggia 的位址為 http://192.168.48.129/peruggia/ ，此網頁應用程式具有會員登入身分驗證功能，我們將對此功能的頁面進行模組一的測試。測試的結果如下：

```
msf  auxiliary(myModule_1) > exploit

[*] - Testing 'numeric' Parameter username:
[*] Possible 'numeric' SQL Injection Found username
[*] - Testing 'single quotes' Parameter username:
[*] Possible 'single quotes' SQL Injection Found username
[*] - Testing 'double quotes' Parameter username:
[*] Possible 'double quotes' SQL Injection Found username
[*] - Testing 'OR single quotes uncommented' Parameter username:
[*] Possible 'OR single quotes uncommented' Injection Found username
[*] - Testing 'OR single quotes closed and commented' Parameter username:
[*] Possible 'OR single quotes closed and commented' Injection Found username
[*] - Testing 'hex encoded OR single quotes uncommented' Parameter username:
[*] - Testing 'hex encoded OR single quotes closed and commented' Parameter username:
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

根據此模組的設計，此攻擊模組會對該目標伺服器進行七種字串的測試。在測試封包發出之前，此模組會印出 Testing 字樣，而後面接上此次嘗試攻擊的字串種類。接著此模組會印出 Parameter 字樣，而後面接著印出該網頁應用程式存取這些字串的變數名稱。若某些特定字串在目標網頁應用程式上能達到 SQL Injection 的效果，變會印出 Possible 字樣，後面分別接上字串的種類、SQL Injection Found 與存取測試字串的變數名稱。我們以 numeric 類型的字串舉例說明。當此模組在目標網頁應用程式的表單，對於 username 此變數的輸入介面進

行 numeric 類字串的測試時，便會印出:

- Testing 'numeric' Parameter username:

若此類字串攻擊成功，便會印出:

Possible 'numeric' SQL Injection Found username

如果字串測試失敗，如 hex encoded OR single quotes uncommented 與 hex encoded OR single quotes closed and commented 兩類的字串，則不會印出任何東西。

由上述的結果我們可以得知，此模組能成功地對目標網頁應用程式進行數個 SQL Injection 攻擊。此結果亦表示 OWASP Broken Web Applications Project 中的 peruggia 網頁應用程式具有 OWASP Top 10 中所公布的 Injection 弱點。

Failure to Restrict URL Access 測試

此模組針對 OWASP Broken Web Applications Project 所提供的虛擬機之網頁應用程式進行 Failure to Restrict URL Access 弱點攻擊。此攻擊模組將會對該網頁應用程式的原始碼進行分析。若能在原始碼中分析出任何子目錄或者父目錄，便會印出 Found Directory Listing 字樣，後面接著該網頁應用程式的網址。若是無法找出任何子目錄或父目錄的資訊，則不會印出此字樣。其執行結果如下：

```
msf  auxiliary(myModule_2) > exploit

[*] Found Directory Listing http://192.168.1.223:80/
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

上述結果表示此模組能對目標 (IP 位址為 192.168.19.130 的虛擬機器) 進行目錄掃描，且表示位於 http://192.168.1.223:80/ 的網頁伺服器具有 OWASP Top 10 中所公布的 Failure to Restrict URL Access 弱點。

5. 結論

本研究首先闡述常見的網頁應用程式弱點並介紹知名的網路弱點公告系統 OWASP、CVE 及 CWE，並以 OWASP Top 10 2010 所公告的 Injection 與 Cross-Site Scripting 的弱點為例，說明相關攻擊如何運作。此外，亦針對目前常見的攻擊工具 Metasploit，簡介其使用方法及增加攻擊模組的方法。最後，我們也在 Metasploit 平台成功開發兩個攻擊模組並進行攻擊測試。未來如有新的網頁應用程式弱點與攻擊方式被揭露，在了解該攻擊的運作原理後，便能在 Metasploit 平台上開發此種新攻擊的模組。透過新增 Metasploit 攻擊模組，未來可滿足網頁應用程式防火牆對於新弱點防護的測試需求。

參考文獻

[1] OWASP
Available: https://www.owasp.org/
[2] MITRE
Available: http://www.mitre.org/
[3] OWASP Top 10 – 2010:
Available: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
[4] CVE
Available: http://cve.mitre.org/
[5] CWE
Available: http://cwe.mitre.org/
[6] Metasploit
Available: http://www.metasploit.com/
[7] Metasploit Framework Users Guide Release 4.3
Available: http://www.metasploit.com/
[8] w3af
Available: http://w3af.sourceforge.net/
[9] Metasploit 3.0 Developer's Guide
Available: http://www.metasploit.com/
[10] OWASP Broken Web Applications Project
Available:
https://www.owasp.org/index.php/OWASP_Broken_Web_Applications_Project