駭客與鬼客技術演進

江易達 鄭宗寰 林盈達
國立交通大學資訊工程系
E-MAIL: {yida,raijin,ydlin}@cs.nctu.edu.tw
October 20, 2008

摘要

在電腦安全領域中，駭客(Hacker)與鬼客(Cracker)的攻擊是主要的威脅來源。駭客是指破壞系統安全的人，而鬼客則是修改軟體以破解軟體保護的人。為了要建置更安全的系統，我們需要先了解他們的歷史和相關知識。文章中分析 Common Vulnerabilities and Exposures 所統計的前五大漏洞以及幾個 wargame 中常用的攻擊，這些攻擊中跨網站執行碼(Cross-site scripting)比起其他傳統的攻擊造成更大的影響。而鬼客的攻擊都是使用逆向工程(Reverse Engineering)，較沒有一般的方法可以防禦，因此在防禦上只能用加密、程式碼混淆等技術增加破解的難度。

關鍵字：駭客，鬼客，漏洞，逆向工程

1 簡介

近年來隨著電腦網路的快速發展，網路的黑暗面也隨之擴展。從網路銀行、入口網站、軟體發行商甚至個人電腦等，每天都必須面對層出不窮的攻擊。這些攻擊的人常被稱為駭客(Hacker)或鬼客(Cracker)，駭客[1]指的是專門尋找系統漏洞，或是利用漏洞入侵系統的人。而鬼客[2]則是修改軟體以破解授權限制的人。兩者的不同點就在於駭客是入侵、破壞電腦系統，而鬼客則是破壞軟體的保護，大部分情況對系統並不會有影響。

在一般人眼中，他們的確在法律與道德面引起許多爭議。從法律的角度來看，駭客與鬼客的行為大都有相對應的法律條文量刑，不過在這些條文中也是有例外條款，若他們的目的是為了教育、學術研究的話就不算犯罪。

而一般人也可以從搜尋引擎中輕易找到許多教學與工具，因此許多初學者照著這些教學依樣畫葫蘆，也可以攻擊成功。這類人通常被稱為 Script kiddie[3]，他們造成的危害不比熟練的老手差。因此許多人認為，這些資訊越公開，越容易引誘人去執行攻擊，所以此類資訊不能隨意公開。法律上也有許多判決支持這個論點，比如美國國會制訂的「數位千禧年授權法案(Digital Millennium Copyright Act)」中就有規定，如果公布該漏洞後危害的範圍很廣，法院可以下令禁止公開。不過在現在的網路中，沒有什麼知識是可以完全隱藏的，前述的狀況也只是用在該漏洞只針對特定系統。如果漏洞常常出現，就應該公布以提醒人避免此漏洞。而且資訊安全學者可以共同研究並提出共通的防禦方法，同時這些防護方式也不

斷的會有人找出不足的地方並修正，便成了駭客和資訊安全界的正向循環，進而加強了安全性。

2 歷史

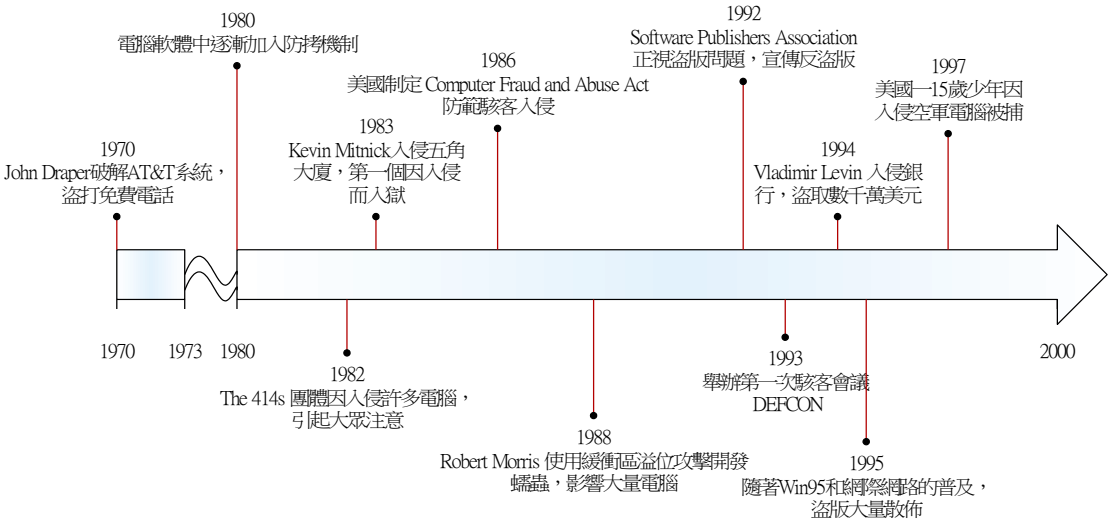


圖 1 – 時間表

早期的幾個重要事件[4-6]如圖 1 所示。可以看到犯罪行為、大規模攻擊等都相當早就有，而且隨著網路擴大，有著傷害變大及攻擊者年齡年輕化的趨勢。軟體破解也是由初期的個人研究，轉變成團體的工作。到了現在，攻擊的目標從伺服器轉向一般大眾，如蠕蟲(worm)、間諜軟體(Spyware)等。防禦範圍不再只有伺服器，一般人也要學著如何防禦攻擊。軟體破解則是因為 P2P 提供比以前 FTP 或 Usenet 更方便的傳輸方法，許多軟體上市當天，甚至上市前就被破解並快速流傳，造成軟體商大量的損失。

為了要防禦他們的攻擊，首先就是要了解他們的攻擊方法。學習時除了閱讀相關知識之外，實際練習這些攻擊也是很重要的過程。但是這並不代表嘗試攻擊網路上的電腦，這麼做只是讓自己陷入犯罪的麻煩中。近年來興起具有挑戰性的遊戲「wargame」，就如同益智遊戲一樣，他提供許多的關卡，每一關都要用不同的攻擊技巧，才能晉級到下一關。在闖關過程中，我們可以學到各類的攻擊技巧，並且瞭解到系統哪裡可能會被攻擊，以及要如何去預防。國內外有許多知名的 wargame 網站，底下列舉其中幾個:

- 台灣冒險家俱樂部 -http://wargame.cna.ccu.edu.tw/
- Hack This Site! -http://www.hackthissite.org/
- Root This Box -http://rootthisbox.org/
- Over The Wire – PullThePlug - http://www.overthewire.org/

嘗試這類挑戰時，須要先連到 wargame 的主機上，他會以網頁或是命令列提供介面來告訴玩家題目。玩家透過這介面嘗試各種不同的攻擊方法，找過關的密碼才能晉級下一關。過程中可以練習到許多的攻擊方法。同時因為這些環境都是別設計過，在攻擊時並不需要擔心會破壞到系統。過關時，有一些基本的規則

要遵守，主要的有下列幾點：

- 不能洩漏過關密碼給其他人
- 不能使用 DoS 或 DDoS 攻擊主機，破壞其他人過關的權利
- 不能使用 wargame 系統再去攻擊其他電腦
- 除了過關需要外，不能修改或刪除系統上的檔案

因為這些網站的目的只是練習和互相競賽，所以這些規則是為了保護系統可以正常運作，讓我們隨時都可以參與遊戲，闖關時不會被其他人惡意的攻擊所干擾。

3 駭客技術解析

雖然目前網路上的攻擊方法相當多，但是攻擊的過程可以歸納成兩個步驟，首先是尋找攻擊目標，之後根據攻擊的漏洞，送出不同的字串。當攻擊成功時，可以取得隱藏的資訊，或是跳過系統認證得到更高的系統權限。

根據 Common Vulnerabilities and Exposures(CVE)的漏洞統計[7]，在 2006 年中的分佈如圖 2 所示：

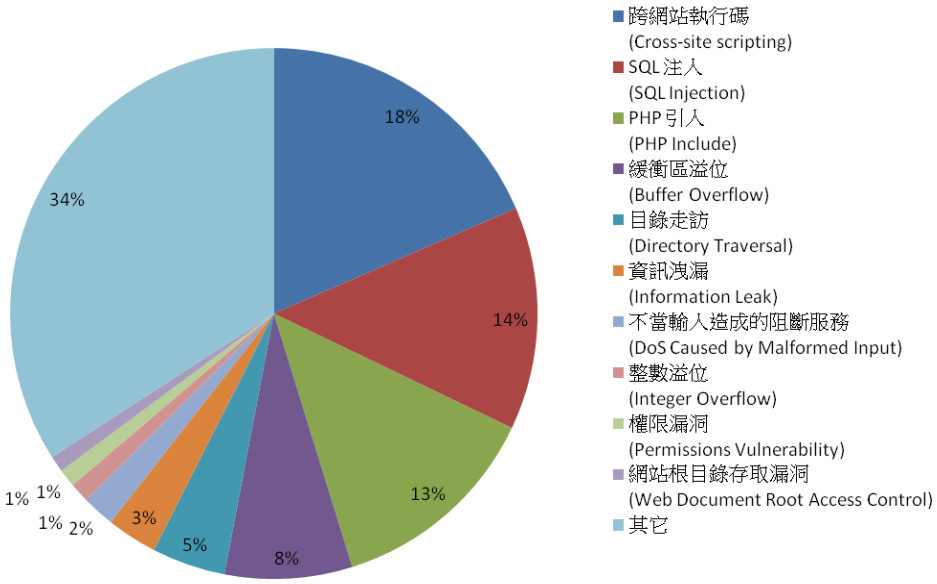


圖 2 – CVE 2006 漏洞統計

從圖中可以發現 2006 年時，漏洞比例前幾名都是針對網頁應用程式的漏洞，如跨網站執行碼、SQL 注入等。這是因為網際網路快速的發展，各類傳統的應用程式都改以網頁的方式運作，加上這類漏洞被使用後，所能獲得的利益比攻擊傳統的應用程式更加巨大，因此駭客們的研究目標轉向各類的網頁應用程式。

接下來將會介紹圖 2 中的前五名漏洞，和其他在 wargame 中常用的技巧。雖然每一種方法各自有不同的原理及攻擊方法，但他們之間的共通性可以整理成表 1。

表 1-攻擊方法比較

	通訊埠掃描	跨網站執行碼	SQL 注入	PHP 引入	緩衝區溢位	目錄走訪	指令注入	共享函式庫置換
主動/被動攻擊	主動	被動	主動	主動	主動	主動	主動	主動
攻擊目標	主機	網站使用者	資料庫	網站	程式	程式	程式	程式
攻擊結果	取得通訊埠資訊	盜取使用者 cookie	控制資料庫	控制程式執行	控制程式執行	資訊洩漏	控制程式執行	控制程式執行
輸入驗證不當	╳	○	○	○	○	○	○	╳
API 使用不當	╳	╳	╳	○	○	╳	╳	╳
不當權限設定	╳	╳	╳	╳	╳	○	╳	○

在這些攻擊分類中，被動攻擊是近來流行的攻擊模式。主動攻擊需要駭客找到特定的目標去攻擊，而跨網站執行碼是被動式的攻擊，駭客只需要透過漏洞在網站中放入攻擊程式碼，之後開啟此網站的使用者都會受影響。所以被動攻擊受害者範圍相當大，常常一個網站被攻擊後就有上萬人受影響。

輸入驗證不當是指程式沒有正確檢查使用者的輸入，因此使用者輸入額外的字元造成系統漏洞。API 使用不當是使用系統 API 時，因為給的參數不當，或使用錯誤的 API 所形成的漏洞。另外不當權限設定是由於設定時沒有限制使用者的權限，才會形成漏洞。

底下介紹這些攻擊的運作方式，並舉出一些例子來分析這些攻擊是如何入侵系統。

(1) 通訊埠掃描(Port Scan)[8, 9]

在確定攻擊目標前，因為電腦相當多，不可能一個個去試有哪些漏洞可以攻擊。所以用掃描的方式，確定哪些電腦和通訊埠有可能攻擊。通訊埠掃描用特殊的網路封包，以不同的回應來偵測對方通訊埠開啟或關閉。依照送出的封包不同，可以分為表 2 幾種方式。

表 2 -通訊埠掃描方式

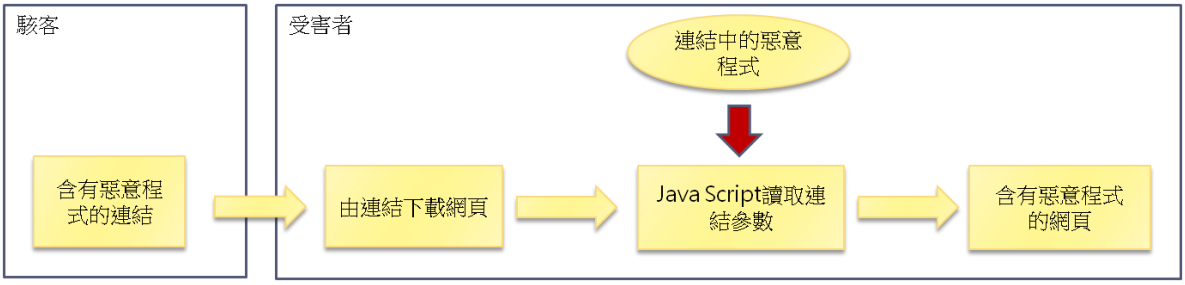
	TCP Connect Scan	UDP Sweep	TCP SYN Scan	SYN/ACK Scan	Fin Scan
管理者權限	不需要	不需要	需要	需要	需要
通訊協定	TCP	UDP/ICMP	TCP	TCP	TCP

攻擊者送出	SYN	UDP 封包	SYN	SYN/ACK	FIN
通訊埠開啟時的回應	SYN/ACK,攻擊者回應 ACK	UDP 封包	SYN/ACK, 攻擊者回應 RST	無回應	無回應
通訊埠關閉時的回應	RST/ACK	ICMP Port Unreachable	RST/ACK	RST	RST

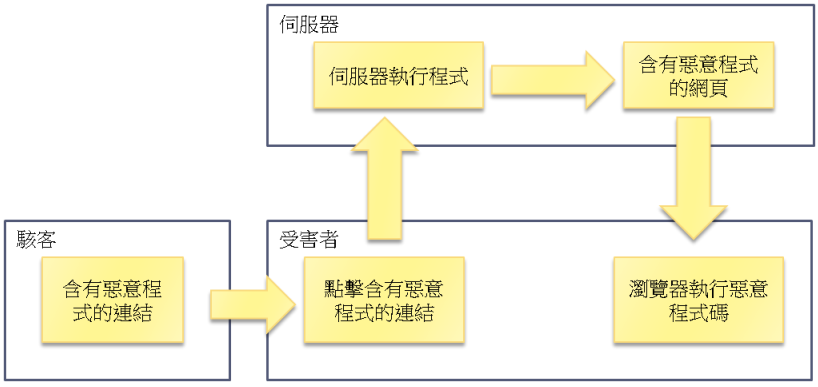
TCP Connect Scan 和 UDP Sweep 這兩種方法都是正常的 socket 連線，所以使用系統的 socket API，不需要管理者權限也可以執行。除此之外都不算是正常的 TCP 連線。所以需用管理者權限，以 Raw Socket API 去執行。但 TCP Connect Scan 最後會成功建立連線，因此伺服器上都會留下連線記錄而被發現。若是使用 raw socket 就不會建立連線而留下記錄了。

(2) 跨網站執行碼(Cross-site scripting)

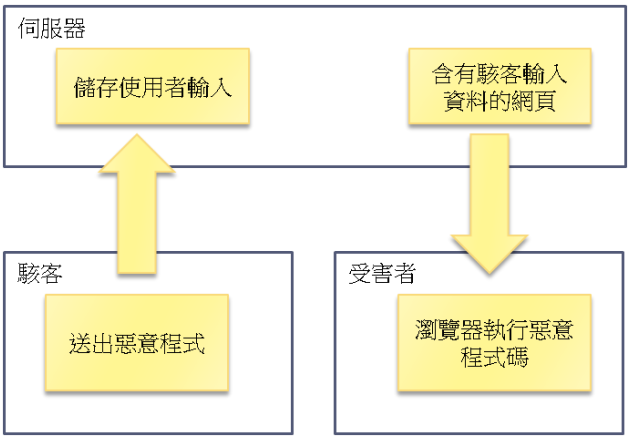
現在的網頁應用程式如討論區、部落格，大都將使用者輸入的資料顯示在網頁中。但問題在於若輸入可執行的 JavaScript 程式，瀏覽器讀取到這段程式碼時就可能會去執行他，因此讓駭客有執行惡意程式的機會。通常駭客插入的程式會讀取使用者 cookie，將 cookie 傳送給駭客後，駭客就可以偽裝成該使用者，取得該使用者的資料與權限。依照程式碼插入頁面的方式，跨網站執行碼可以分類成三種[10-12]—DOM(Document Object Model)、非持久(Non-Persistent)及持久(Persistent)。這三類的攻擊流程分別在圖 3 的(A)，(B)及(C)中。



(A)DOM 式攻擊



(B)非持久攻擊



(C)持久攻擊

圖 3 - 跨網站執行碼執行流程

DOM 式的攻擊是在 JavaScript 中讀取使用者輸入資料時，因程式沒有做好輸入檢查而給駭客有插入執行碼的機會。例如說以下的程式，他會讀取網址列參數並寫進網頁中。

```
var pos=document.URL.indexOf("name=")+5;
document.write(document.URL.substring(pos,document.URL.length));
```

駭客攻擊的方式是產生一個連結如下，引誘使用者去點他，就可以讓使用者執行 alert(document.cookie)這段程式。

```
Index.htm?name=<script>alert(document.cookie)</script>
```

執行後會在客戶端的瀏覽器跳出一對話框，顯示此網站儲存的 cookie。駭客可以用 XMLHttpRequest 物件將 cookie 送至駭客的電腦中，這樣就可以蒐集到使用者的 cookie 而偽裝成該使用者，得到他的權限。

非持久攻擊則是伺服器執行網頁的程式碼如 PHP、ASP 時，將使用者輸入的資訊內嵌在網頁後顯示給使用者，這也會讓駭客有嵌入執行碼的機會。底下的 PHP 程式單純將使用者輸入顯示出來。因為這個頁面只在使用者點下連結時存在，所以被稱為非持久攻擊。

```
<?echo “hello, $_GET[‘name’]” ?>
```

而使用者點了一個連結如下，就會執行了內嵌在 name 變數的程式碼。

```
hello.php?name=<script>alert(document.cookie)</script>
```

最後一種持久攻擊是最常見的攻擊法。一些使用者互動的網站如部落格，討論區都會讓使用者看到其他人輸入的文章。同樣的若沒有仔細檢查，讓駭客輸入了惡意程式碼，每個讀取該網頁的人都會受到影響。不像前幾種要使用者點連結，這種攻擊只要讀取頁面就會執行，所以影響的數量可能高達上萬人，這也是最近此類漏洞特別被重視的原因。

(3) SQL 注入(SQL Injection)

現在的網頁應用程式後端常以資料庫來管理資料。因此需要 SQL 語言來操作資料庫。若操作時會以使用者輸入來當作 SQL 查詢的一部分，且沒有適當的過濾，就有可能形成漏洞，執行任意的 SQL 指令。底下的例子，網站以使用者的帳

號及密碼查詢資料庫。若回傳資料數不為零，代表使用者通過認證。

```
SELECT * from users where id=$id and pass=$pass
```

但若這裡輸入的 id 為 myid，pass 為 1 or 1=1，就會讓 SQL 語法變成如下

```
SELECT * from users where id=myid and pass=1 OR 1=1
```

資料庫在運算時，就會當作如圖 4 的剖析樹處理

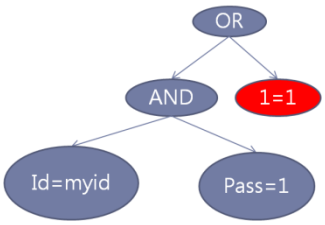


圖 4 - SQL 剖析樹

因為最上層有一個 1=1 為真，會讓所有資料都滿足條件，因此讓使用者不論密碼正不正確都通過認證。形成很大的安全漏洞。

(4) PHP 引入(PHP Include)[13, 14]

PHP 的 include()函式可以將其他檔案引入動態網頁的程式碼中執行，這方法在共用函式上很方便。但因為他也支援讀取遠端伺服器的程式碼，所以若他的路徑可以由使用者輸入決定，就有機會改變路徑，讀取駭客的程式碼在伺服器執行。如底下的例子：

```
include($lib."/lang.inc");
```

若駭客在他自己的機器 ex.com 上放一個 lang.inc 檔，內容為：

```
system("cat /etc/passwd");
```

之後再將$lib 變數設定成 http://ex.com，就可以在對方伺服器上執行 cat /etc/passwd，取得使用者密碼檔。

(5) 緩衝區溢位(Buffer Overflow)[15, 16]

一般程式中都會先規劃一塊記憶體空間來存使用者輸入，這空間大小是有限的。當程式讀取輸入字串時沒有使用適當的 API 作範圍限制，就會超出分配好的記憶體範圍，而修改到不屬與該程式可用的記憶體內容。駭客可以利用這點，改變作業系統在記憶體中該程式的記錄，控制程式的執行。

最常見的攻擊方式為針對堆疊(stack)攻擊，程式中呼叫函式時，系統會在堆疊中儲存回傳位址及區域變數等資料，回傳位址是記錄函式執行完畢後要跳到哪裡繼續執行。駭客可以先將惡意程式碼放在緩衝區或是環境變數中，之後輸入過長的字串，修改記錄回傳位址的空間，就可以控制函式執行完畢後跳到惡意程式碼的位址執行駭客的程式。

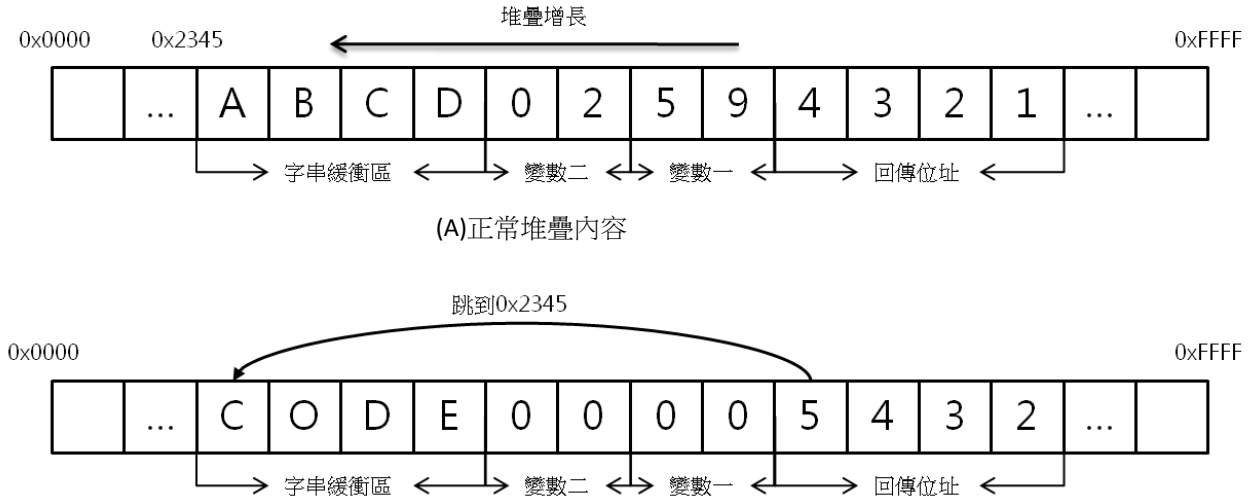


圖 5 - 緩衝區溢位比較

如圖 5 的(A)，在 X86 系統中堆疊是往低記憶體位址增長，而寫入資料時是反過來，從低記憶體位址開始寫。所以當讀取文字到緩衝區時，若沒有限制大小，駭客可以推算出要寫入多少才會到回傳位置，然後再寫入要改變的值，就可以改變回傳位址。圖 5(B)中因為沒有檢查輸入字串長度，輸入'C' 'O' 'D' 'E' 0x0 0x0 0x0 0x0 0x5 0x4 0x3 0x2，就可以改變回傳位址的值為 0x2345。

(6) 目錄走訪(Directory traversal)

此類漏洞以常發生在網頁伺服器。伺服器上的根目錄對應到主機上某個目錄，而使用者只能讀取此目錄下的檔案。但沒有限制好的話，會讓使用者有機會輸入"../"之類的路徑而跳到上一層目錄，可以看到任意檔案甚至執行程式。如微軟的 MS00-057 IIS 漏洞[17]就是此類一個很著名的例子。

(7) 指令注入(Command Injection)

他的攻擊方法和前面的 SQL 注入很像。只是他插入的對象是程式呼叫命令列執行的指令。以下的程式使用者輸入檔名存在$file 變數中，然後程式呼叫 cat 指令顯示此檔案：

```
system( "cat $file" );
```

但是命令列中有一些特殊字元如;或&&等等，可以在一行中包含兩個指令，讓 shell 去輪流執行。因此可以輸入";rm –rf /"，$file 變數內容為";rm –rf /"，這樣傳送給 shell 的指令會是：

```
cat ;rm -rf /
```

而 shell 就會看成 cat 和 rm –rf /兩個指令。這樣就可以讓駭客刪除系統上所有檔案。

(8) 共享函式庫置換(Shared Library Exchange)

共享函式庫的概念在各作業系統中幾乎都有，作業系統在執行程式時，從外部的檔案讀取函式，提供給程式呼叫。此類的方法問題在於若外部檔案有同名的函式，系統會優先使用他。因此可以寫程式編譯成共享函式庫，然後讓原來程式

執行時載入，進而以該程式的權限執行任意函式。

例如說 GNU C 函式庫執行時會根據 LD_PRELOAD 環境變數載入共享函式庫。若原本的程式有用 printf 印出字串，就可以另外寫一個 printf 函式：

```
void printf(…){System(“bash”);}
```

編譯成共享函式庫後載入，取代原來要呼叫的函式。

4 鬼客技術解析

「逆向工程(Reverse Engineering)[18]」是面對一個未知道裝置時，瞭解它底層原理所使用的技巧。用在程式上，就是從已編譯過機器碼中，去分析或修改他的執行流程。鬼客常利用這種技巧，修改軟體來破解序號、光碟認證等授權限制。

在破解時，最繁雜的部分是程式的分析。因為從機器碼幾乎不可能還原成他原來的程式碼，只能將他反組譯後從組合語言來分析。所以有一些方便的工具如 OllyDbg[19]、IDA Pro[20]之類的軟體，提供反組譯，字串搜尋，中斷點等功能，加快分析的速度。底下的例子是呼叫一個檢查註冊的函式，若不通過程式就結束：

```
if(reg_check()!=REGISTERED)
    return;
```

將他編譯成執行檔以後，我們可以用工具來轉換成如下的組合語言：

```
call reg_check
cmp eax,REGISTERED
je ok
retr
ok:
```

這段程式中，reg_check 函式的回傳值存在 eax 暫存器中，若 eax 的值和 REGISTERED 相同則會跳到 ok 後繼續執行，若不同則執行 retr 結束程式。我們可以刪除掉 retr 這個指令，讓他在比對不相同時，也會繼續執行 ok 後的程式而不會結束程式，這樣就可以跳過他的註冊檢查了。

由此可見簡單的程式反組譯後很簡單就可以修改，因此要防禦鬼客的破解，通常是使用程式碼混淆(code obfuscation)、加密等方法加大分析上的難度。

5 結語

雖然上面介紹了這麼多的攻擊點，但這也只是冰山一角而已。由於網際網路的發達，駭客和鬼客已經轉變成透過網路，隱密的團體行動。同時因為網路快速傳播，新的漏洞一被發現後，就會很快的被大量利用而造成損失。攻擊的對象也從網路伺服器延伸到一般的使用者。在這些漏洞中，跨網站執行碼是最近影響最大的漏洞，許多網站和使用者都因為此漏洞而受害。而鬼客的破解雖然原理只有一種，但也很難預防，只能用更複雜的加密，程式碼混淆等方法加大破解的難度。面對如此多變的攻擊，要做到完全預防是相當困難的一件事。所以我們只能夠了解他們如何產生，並且避免寫出有問題的程式。在寫程式時，有下面幾點需要注

意：

1. 對使用者輸入詳細檢查，過濾各類特殊字元及限制輸入字串長度
2. 使用白名單式的權限控管，先預設全部關閉，再開放有需要的項目
3. 設計程式時，不要只考慮預定的輸入，有可能會有預料外的輸入
4. 重要的驗證程式碼以加密，程式碼混淆或是限制反組譯器執行來避免破解

6 參考資料

[1] c. Wikipedia. "Hacker (computer security)," 25 September 2008 07:53 UTC; http://en.wikipedia.org/w/index.php?title=Hacker_(computer_security)&oldid=240738842.

[2] c. Wikipedia. "Software cracking," 25 September 2008 07:57 UTC; http://en.wikipedia.org/w/index.php?title=Software_cracking&oldid=237892668.

[3] L. Spitzner, "Know Your Enemy," *Parts I, II, III." Available online: www. linuxnewbie. org/nhf/intel/security/enemy. html.(For Parts II and III, replace" enemy" with" enemy2" and" enemy3," respectively.)*, 2001.

[4] c. Wikipedia. "Timeline of computer security hacker history," 7 August 2008 14:57 UTC; http://en.wikipedia.org/w/index.php?title=Timeline_of_computer_security_hacker_history&oldid=230393939.

[5] C. Iozzio. "The Cyber Crime Hall of Fame," http://www.pcmag.com/article2/0,2704,2329604,00.asp.

[6] c. Wikipedia. "Warez," 26 September 2008 02:09 UTC; http://en.wikipedia.org/w/index.php?title=Warez&oldid=240218901.

[7] S. Christey, and R. A. Martin, *Vulnerability Type Distributions in CVE*, 2007.

[8] M. de Vivo, E. Carrasco, G. Isern *et al.*, "A review of port scanning techniques," *ACM SIGCOMM Computer Communication Review,* vol. 29, no. 2, pp. 41, 1999.

[9] O. Arkin. "Network Scanning Techniques," http://www.arcert.gov.ar/webs/textos/netscan.pdf.

[10] D. Endler, "The Evolution of Cross-Site Scripting Attacks," *Whitepaper, iDefense Inc.(May 2002) http://www.cgisecurity.com/lib/XSS.pdf*.

[11] C. GovCertUK, "Cross Site Scripting Techniques and mitigation," 2007.

[12] "[Cross-site Scripting] Threat Classification - Web Application Security Consortium," http://www.webappsec.org/projects/threat/classes/cross-site_scripting.shtml.

[13] S. Clowes, "" A Study In Scarlet–Exploiting Common Vulnerabilities in PHP Applications," *SecureReality,*

http://www.securereality.com.au/archives/studyinscarlet.txt.

[14] "CWE-98: Insufficient Control of Filename for Include/Require Statement in PHP Program (aka 'PHP File Inclusion')," http://cwe.mitre.org/data/definitions/98.html.

[15] C. Cowan, P. Wagle, C. Pu *et al.*, "Buffer overflows: attacks and defenses for the vulnerability of the decade," *Foundations of Intrusion Tolerant Systems, 2003 [Organically Assured and Survivable Information Systems]*, pp. 227-237, 2003.

[16] E. Levy, "Smashing the stack for fun and profit," *Phrack magazine,* vol. 7, pp. 49, 1996.

[17] "Microsoft Security Bulletin (MS00-057)," http://www.microsoft.com/technet/security/bulletin/ms00-057.mspx.

[18] E. J. Chikofsky, and Jh, II, "Reverse engineering and design recovery: a taxonomy," *Software, IEEE,* vol. 7, no. 1, pp. 13-17, 1990.

[19] "OllyDbg v1.10," http://www.ollydbg.de/.

[20] "IDA Pro Disassembler - multi-processor, windows hosted disassembler and debugger," http://www.hex-rays.com/idapro/.