以安全性高階程式語言移植網路軟體之研究
On Porting Network Software with a Type-safe Programming Language

黃一軒[*]、李子榮[+]、楊正仁[*]
元智大學資訊工程系
{ihhuang, czyang}@syslab.cse.yzu.edu.tw[*]
s902314@mail.yzu.edu.tw[+]

摘要

隨著網路快速發展，由於軟體中的安全漏洞問題，來自蠕蟲、病毒和駭客的入侵每年都造成很大的經濟損失。為了從根本解決軟體安全的問題，我們探討以安全性高階程式語言來移植既有網路軟體，並研究移植後在軟體安全和效能上的影響。我們以 Cyclone 安全性高階程式語言為例移植漏洞相當多的 WU-FTPD 軟體，並以入侵工具實際測試移植後的軟體安全性。實驗結果顯示，移植後的 WU-FTPD 確實可以防止來自網路的入侵，同時也不會因為安全性檢查而影響程式碼大小和傳輸效能。

關鍵詞：軟體安全，安全性高階程式語言，緩衝區溢位，Cyclone，WU-FTPD

一、 簡介

電腦安全一直是電腦發展史上的一個重要議題，尤其自 1988 年 Robert Morris 散佈電腦蠕蟲以來，隨著網路科技的進度，電腦安全的需求愈形重要。特別是在最近 10 年中，網路攻擊與各式各樣的病毒、木馬與蠕蟲已然嚴重影響人們日常生活與社會經濟。例如：緩衝區溢位 (buffer overflow) 漏洞、允許使用者執行不受信任的程式碼、或是參數攻擊 (argument injection) 等等 [3,4]。藉由網際網路無遠弗屆的連接範圍，網路軟體變成入侵者攻擊電腦系統的方便途徑之一。舉例來說，2003 年 1 月 25 日就發生了一起非常嚴重的電腦入侵事件，代價是數十億美金的商業損失。起因是微軟在以高階程式語言開發的資料庫伺服器程式中不慎留下緩衝區溢位安全性漏洞，結果導致全球超過 90% 採用微軟資料庫系統的電腦，僅在 10 分鐘內就遭到 Slammer 病毒癱瘓 [6]。因此在現代網路如此發達普及的時刻，加強網路軟體的安全性是刻不容緩的任務。

在網路軟體設計的過程中，高階程式語言扮演了相當重要的角色。由於高階程式語言如 C 語言或是 Java 語言具有容易學習和維護的特性，大部分現行的網路軟體都是以高階程式語言設計而成。其中，因為 C 語言具有可操控低階系統指令、記憶體系統、和網路的特性，在網路軟體中，選擇以 C 語言開發的軟體數量相當的多。但是，以 C 語言來開發的軟體常常因為程式設計上的疏忽，而隱藏著安全性漏洞，例如，以 C 語言開發的知名 FTP 伺服器 WU-FTPD [7] 就時常被發現新的安全性漏洞。為了填補這些安全漏洞，常常需要花費不少額外的人力。倘若能夠在軟體開發當時即利用安全性高階程式語言來幫助程式設計師避免錯誤，軟體開發的成本將可以降低。另一方面，若不從軟體開發的程式語言著手，將使軟體的安全性過度依賴程式設計師本身謹慎細心的程度，也無法從根本上達成軟體安全的目的。

因此在本篇論文中，我們探討以安全性高階程式語言來移植既有網路軟體，希望在此研究過程中，一方面驗證程式語言層級的安全性措施的影響，並討論移植後對軟體效

能的影響。我們選定以 Cyclone 程式語言 [5] 做為安全性高階程式語言的研究對象，並以安全性漏洞相當多的 WU-FTPD 伺服器軟體做為研究標的。Cyclone 語言是由 Jim 等人在 2002 年發表的一套安全性程式語言。在設計之初，Cyclone 語言就是為了取代 C 語言而設計。為了使 C 語言程式設計師可以很快速的熟悉 Cyclone 語言，Cyclone 的程式語法刻意設計的與 C 語言一樣。並且為了避免因為緩衝區溢位所造成的傷害，在 Cyclone 中，編譯器會自動地在每一個指標變數後加上邊界檢查。一旦指標變數存取到不合法的記憶體區段，程式就會被終止，以避免執行到惡意的程式碼。Cyclone 同時也利用嚴格的靜態程式碼檢查盡量在編譯階段找出所有可能的安全性漏洞。

透過移植 WU-FTPD 伺服器的過程，我們討論了以安全性高階程式語言移植網路軟體所需要處理的步驟。由於 C 語言已被用以開發許多網路軟體，根據我們所提出的移植步驟，相信許多網路軟體都能因而受惠，去除許多潛在性的安全問題。除此之外，我們所移植的 WU-FTPD，不僅僅可以做為 Cyclone 語言的移植範例，也可以直接安裝為現行 Unix 作業系統中的 FTP 伺服器，而且傳輸效能並未受到影響。

本論文的第二節將回顧 WU-FTPD 上的發展及其安全性漏洞。第三節敘述以 Cyclone 開發 WU-FTPD 應用程式時的詳細步驟。第四節分析 C 版本與 Cyclone 版本 WU-FTPD 的效能差異。最後總結本文並探討未來研究方向。

二、 WU-FTPD 的安全問題

WU-FTPD 是由 Myers 和 O'Connor 在 Washington University 共同發展的一套 FTP 伺服器 [7]。因為利用 WU-FTPD 架設 FTP 伺服器的步驟簡單明瞭，經過多年的發展，WU-FTPD 已經成為 Unix 作業系統中使用最廣泛的 FTP 伺服器之一。然而這套系統卻也常被發現安全上的漏洞，成為入侵者最佳的攻擊目標之一。因此，確保 WU-FTPD 具有高安全性是很重要的。如果 WU-FTPD 包含可供入侵者利用的安全性漏洞，會影響到數以萬計的使用者，我們也因而選定以 WU-FTPD 做為此次研究的標的。

最近一次在 WU-FTPD 伺服器上被發現的漏洞發佈於 2003 年 6 月，在 WU-FTPD 的最新版本 2.6.2 中，其中的 realpath.c 檔案中有一個函式 fb_realpath 存在一個差一錯誤 (off-by-one error)，如圖 1 所示。這個差一錯誤存在於第 302 行程式碼的地方。在程式碼第 296 行時，當路徑指向根目錄時，旗標 rootd 的值為 1，反之則為 0。在程式碼第 306 行的地方，如果路徑不是指向根目錄，則程式會幫解析出來的路徑加上「/」符號。值得注意的地方在於，一旦 resolved 變數加上「/」符號後超過了 MAXPATHLEN 的大小，程式就會出現緩衝區溢位錯誤，並可能給予入侵者一個獲得 superuser 權限的機會。在原始含有漏洞的程式碼中可以觀察到，在第 302 行程式碼的地方，當初的程式設計師確實有考慮到緩衝區溢位的可能性，但是卻因為疏忽，誤將 if(strlen(resolved) + strlen(wbuf) + !rootd + 1 > MAXPATHLEN)輸入成 if (strlen(resolved) + strlen(wbuf) + rootd + 1 > MAXPATHLEN)，並造成差一錯誤。

在上述的差一錯誤中，很明顯的是由於當初撰寫 realpath.c 檔案的程式設計師誤將 rootd 旗標的值寫反所造成。在傳統以 C 語言開發的應用程式中，像這類因為程式設計師疏忽所造成的安全性問題，恐怕被發現的比例只是冰山一角。因此，若不從程式語言著手改善軟體安全性，我們將過度依賴程式設計師的軟體開發程序上的安全性，而無法確保可以達成軟體安全的目標。

```
…       …
102     char *p, *q, wbuf[MAXPATHLEN];
…       …
292     /*
293       * Join the two strings together, ensuring that the right thing
294       * happens if the last component is empty, or the dirname is root.
295       */
296     if (resolved[0] == '/' && resolved[1] == '\0')
297     rootd = 1;
298     else
299     rootd = 0;
300
301     if (*wbuf) {
302     if (strlen(resolved) + strlen(wbuf) + rootd + 1 > MAXPATHLEN) {
303           errno = ENAMETOOLONG;
304           goto err1;
305     }
306     if (rootd == 0)
307           (void) strcat(resolved, "/");
308     (void) strcat(resolved, wbuf);
309     }
…       …
```

圖 1：realpath.c 原始碼中的 fb_realpath 函式片段

相對於傳統的 C 語言，我們採用的 Cyclone 安全性高階程式語言可以解決軟體安全過度依賴程式設計師的問題。在軟體開發的過程中，程式設計師只需要依照寫作習慣撰寫程式，而 Cyclone 會自動在每一個存取記憶體的變數後加上邊界檢查程式碼。在軟體越界存取到非法的記憶體區塊前，軟體會先執行到這些邊界檢查的程式碼並觸發軟體中斷，使得入侵者無法利用安全性漏洞取得 superuser 的權限。此外，Cyclone 也運用靜態檢查的技術在編譯程式時進行程式語法如資料型態、迴圈、條件句的檢查等等。不過將軟體移植到 Cyclone 語言後，由於 Cyclone 在軟體上附加了額外的邊界檢查程式碼，軟體效能會稍稍降低。然而相較於因為不安全的軟體而造成的損失，犧牲效能以換取更安全的軟體是值得的。總結來說，結合動態的邊界檢查與靜態的語法檢查，Cyclone 得以負起把關軟體安全的重責大任。

三、 移植 WU-FTPD

從 C 語言移植網路軟體到 Cyclone 語言的過程分為兩個階段，第一個階段是修改函式庫標頭檔 (header file)，第二個階段是修改軟體的原始碼。由於 Cyclone 語言尚未將所有的 C 標準函式庫標頭檔改寫並且安裝在 Cyclone 語言的預設函式庫目錄內，我們必須將需要被使用但卻不在 Cyclone 函式庫內的 C 標準函式庫標頭檔複製到 Cyclone 函式庫中，並且修改標頭檔，使得標頭檔得以符合 Cyclone 語法。此外，由於 Cyclone 語言對於程式語法有相當嚴格的限制，例如變數型態的正確宣告以及初始值、條件判斷式的例外條件等，像這些語法上的細節部分都必須在原始碼中明確地定義清楚，使我們也必須花費大量的時間修改軟體的原始碼。詳細的移植過程則分述如下。

表 1：手動修改標頭檔時常見之錯誤訊息與對應之處理

訊息	對應處理
Filename.cyc: xxxx.h No such file or directory	在 Cyclone 預設的 include 資料夾中建立找不到的標頭檔
Filename.cyc: undeclared identifier xxx	
Undifined identifier	到 C 語言標準函式庫尋找相對應的巨集變數、資料結構和巨集函式，如果使用到以 C 語言撰寫的外部函式，程式碼在宣告外部函式時必須加上 extern "C"

3.1. 標頭檔的增補

在發展 Cyclone 編譯器的時候，Jim 等人除了提供 Cyclone 語言的專用函式庫外也使用了舊有的 C 語言標準函式庫。但是他們目前只移植了較常用的函式，至於與作業系統溝通的系統函式 (system call) 則有很大的一部份尚未移植完成。因此，在目前移植軟體的過程中，我們必須自行將各個尚未完成移植的標頭檔移植到 Cyclone 語言環境之內。根據 Cyclone 操作手冊中的介紹 [1]，移植標頭檔的方法有兩種，第一種是依靠程式設計師手動修改，第二種是運用附在 Cyclone 編譯器軟體的 buildlib 應用程式協助程式設計師移植標頭檔。我們將手動修改標頭檔時會遭遇的錯誤訊息和相對應的解決方法表列於表 1。

在手動移植大型應用程式到 Cyclone 平台時，最常看到的錯誤訊息就是找不到標頭檔定義的錯誤訊息。如果出現找不到檔案或目錄，我們需要將尚未完成移植的檔案建立在 Cyclone 預設的函式庫路徑下。同樣地，如果 Cyclone 編譯器找不到巨集變數、資料結構或是巨集函式的定義，我們需要到原始 C 語言標準函式庫中將相對應的定義複製到 Cyclone 預設的函式庫路徑中。比較特別的是，如果我們使用到的 C 外部函式是可以信賴的，不需要再以 Cyclone 語法改寫，我們可以在宣告時加上 extern "C"宣告字。

除了手動修改標頭檔資訊外，我們也可以利用 buildlib 工具來減少繁瑣的修改工作。在我們的移植過程中，我們也是採用這個方式來移植應用程式。

buildlib 能夠掃瞄 C 語言標準函式庫內的所有標頭檔，並且自動建置 Cyclone 版本的標準函式庫，大大的減少一再重複的繁瑣工作。不過，使用 buildlib 前，必須為它撰寫特別的控制命令，因此運用 buildlib 工具的移植方法仍然無法完全克服移植時的繁瑣工作。

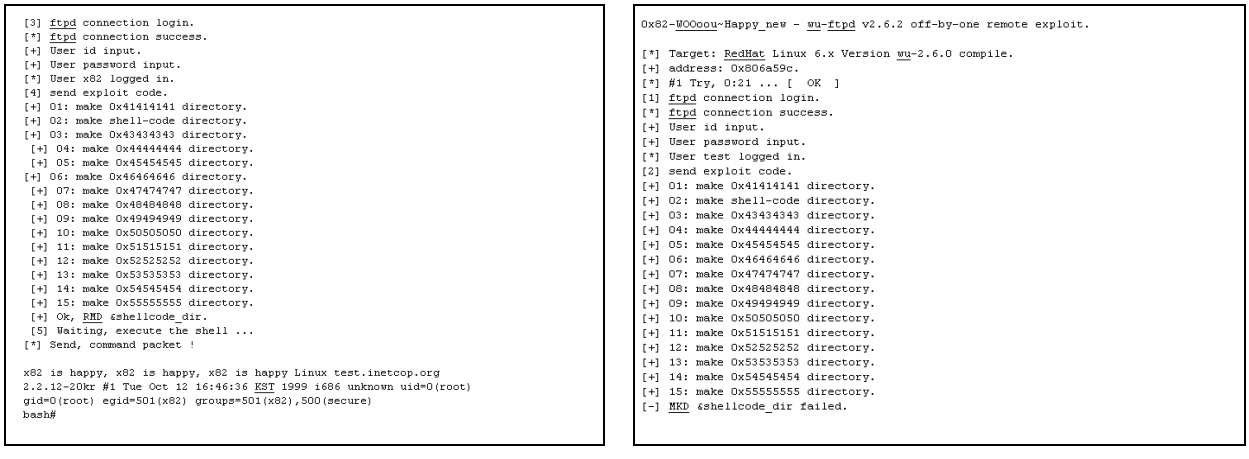
3.2. 修改 WU-FTPD 原始碼

除了上述修改標頭檔的工作，在移植大型應用程式時必須花費最多時間在修改程式主體原始碼上。我們將移植過程中最常遭遇的八種修改方式條列如下。

1. 因為 Cyclone 為每一個指標變數加上邊界限制，同時為了區分陣列變數是否以 null 值做為結尾，我們必須將陣列變數加上@zeroterm 符號。
2. Cyclone 語言添增了 fat pointer 指標變數，為了避免傳統 C 語法中的 pointer 變數因為位址計算錯誤導致存取到非法記憶體區域，在 Cyclone 的語法中所有的指標變數運算都只能作用在 fat pointer 變數上。
3. 所有 Cyclone 的變數型態宣告是必須很明確的，換句話說，像 const 或是 non-const 等宣告都必須在程式碼中標明。
4. 如果存取到未初始化的變數，整個程式的流程有可能發生錯誤，而給予入侵者可趁之機。因此，在 Cyclone 語法中，所有的變數，尤其是指標變數都必須在宣告時給予初始值。
5. 動態記憶體配置不當有可能導致系統資源不足，進而造成系統損毀。為了避免動態記憶體配置不當，Cyclone 加入了 garbage collector，並且改以 new 取代 C 語法原有的 malloc 和 relloc 函式。
6. 在傳統的 C 語法中，switch 語法常常用在多條件判斷的選擇上。但是，如果程式設計師在撰寫 switch 語法時忘了加入 break 或是 default 敘述，switch 語法的功能就可能發生錯誤。因此，Cyclone 要求程式設計師必須明確加上 break、fallthru 和 default 敘述。
7. 為了避免發生 dangling pointer 問題 [8]，Cyclone 加入了 region 的設計。如果參數或回傳值有參考到 heap，Cyclone 會強迫使用者將所有相關的參數都宣告為 region，讓記憶體區塊不會被釋放。
8. Cyclone 語法中利用多型和 union 語法來取代傳統 void*的 C 語法。可以避免發生在 function pointer 上的錯誤。

四、 系統測試與效能比較

我們將上述移植完成的 WU-FTPD 安裝在 Intel Pentium II 350MHz 個人電腦上，其作業系統為 Linux 2.4。我們首先利用 You [9] 所設計的 WU-FTPD 漏洞入侵工具進行 WU-FTPD off-by-one error 漏洞的檢測，執行畫面如圖 2。圖 2 (a)是 You 以 WU-FTPD off-by-one error 漏洞入侵工具測試尚未 patch 的 WU-FTPD 的執行



(a) 尚未 patch 過的 WU-FTPD　　(b) Cyclone 版本的 WU-FTPD

圖 2：WU-FTPD off-by-one error 漏洞測試

表 2：WU-FTPD 移植前後程式碼行數比較

檔案名稱	C	Cyclone	檔案名稱	C	Cyclone
access.c	1533	1552	hostacc.c	361	368
acl.c	180	184	loadavg.c	493	493
auth.c	105	105	logwtmp.c	200	206
authenticate.c	76	76	paths.c	231	251
ckconfig.c	170	177	popen.c	255	260
conversions.c	200	205	private.c	331	338
domain.c	752	771	proto.h	295	299
extensions.c	2355	2370	rdservers.c	109	110
ftpcmd.y	1939	1963	realpath.c	367	368
ftpcount.c	418	428	restrict.c	189	190
ftpd.c	7549	7627	routevector.c	579	584
ftprestart.c	317	330	sigfix.c	81	81
ftpshut.c	478	494	test.loadavg.c	9	9
getpwnam.c	156	159	timeout.c	64	64
glob.c	639	641	wu_fnmatch.c	202	204
hard.loop.c	4	4	COPYRIGHT.c	59	59
			Total	20696	20970

畫面 [10]，在圖中可以看到最後入侵工具順利的開啟了一個 shell。圖 2 (b) 則是以相同的入侵工具測試 Cyclone 版本的 WU-FTPD 的執行畫面。從圖中可以看到入侵工具在入侵時會產生軟體中斷，使入侵無法成功。因此，將網路軟體移植到 Cyclone 後確實是安全的。

為了研究 Cyclone 對於網路軟體效能的影響，我們在 Ethernet 環境上實際操作，對系統進行效能的測試。我們將移植前後的 WU-FTPD 進行效能上的比較，比較的項目總共有兩項，分別是移植前後的程式碼大小和移植前後的檔案傳輸效能比較。經過我們的比較，如表 2 所示，移植後的 WU-FTPD 整體程式碼由 20696 行成長為 20970 行，成長不到 1%（移植的過程並未進行程式碼最佳化）。由這個數據可以得知，將網路軟體移植到 Cyclone 語言並不會增加太多的程式碼。

WU-FTPD 檔案傳輸效能的比較如圖 3 所示，我們以 dkftpbench [2] 進行效能測試。dkftpbench 會自動建立連線，抓取檔案並計算檔案傳輸的效能。從圖中看得出來，移植前後，兩者的效能幾乎完全沒有差別。相較於為了改善安全性而花費的時間來說，檔案傳輸造成在 I/O 上的延遲反而佔了絕大部分。因此可以看出以 Cyclone 移植網路軟體並不會嚴重影響網路軟體本身的效能。

綜合以上的實驗結果，我們可以發現將網路軟體移植到 Cyclone 語言確實可以保證軟體的安全性。同時，雖然 Cyclone 會需要多做安全性防護而可能降低軟體的執行效能，

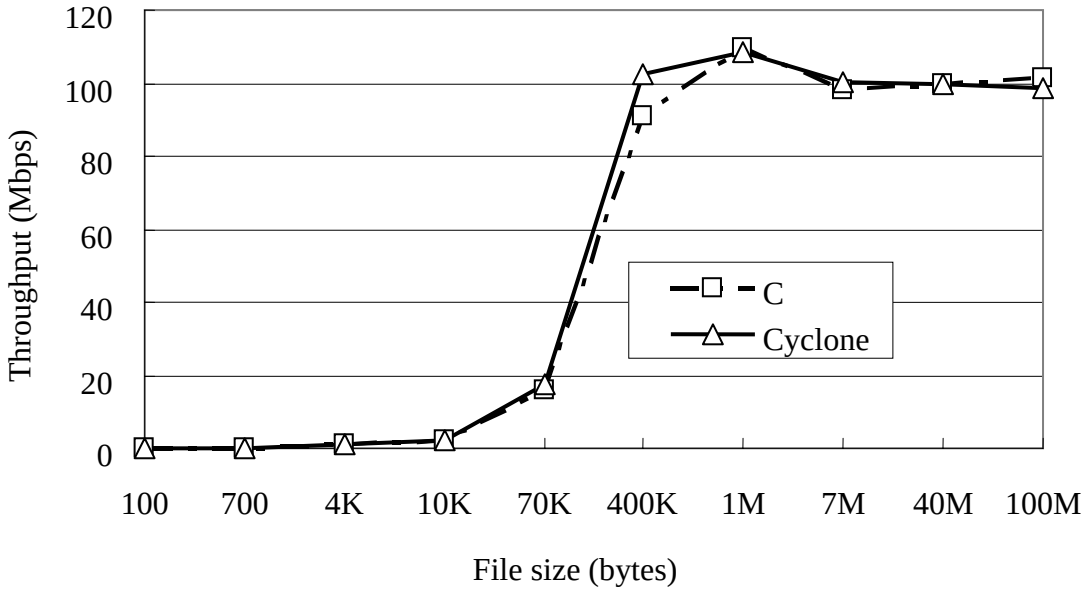


圖 3：WU-FTPD 移植前後的檔案傳輸效能比較

但是從我們的實驗數據中發現，相較於系統進行 I/O 操作所花費的時間來說，Cyclone 對網路軟體所造成的效能負擔是可以被忽略的。

五、 結論與未來工作

電腦安全一直是資訊界關心的重要議題。由於軟體充斥著各式各樣的安全漏洞，每一年造成的經濟損失都十分巨大。為了從根本解決軟體安全的問題，我們在本論文中探討以安全性高階程式語言移植網路軟體，並研究移植後在軟體安全性和效能上的影響。我們實際以 Cyclone 安全性高階程式語言移植漏洞很多的 WU-FTPD 伺服器，並以 WU-FTPD off-by-one error 入侵工具實際測試移植後的 WU-FTPD。實驗結果顯示，移植後的 WU-FTPD 可以通過安全性測試。至於在 WU-FTPD 的效能上，移植前後無論在程式碼大小或是檔案傳輸效能上的改變均不大。

在我們移植 WU-FTPD 伺服器的過程中，我們發現雖然以 Cyclone 安全性高階程式語言移植軟體可以提高軟體的安全性，但是移植軟體的過程卻會耗費大量時間。此外，雖然 Cyclone 內已經包含了用以轉譯 C 語言原始碼的 buildlib 工具，buildlib 工具卻仍舊只能提供半自動化的轉譯。因此，為了落實提升軟體安全性的目標，設計更容易使用的轉譯工具以進行大規模軟體移植是有必要的。在未來，我們期望能夠設計一套完全自動化將軟體移植到 Cyclone 語言的工具，以降低移植軟體時所需要耗費的人力。另一方面，並針對安全性高階程式語言的機制進行更進一步的探討。

六、 參考文獻

1. *Cyclone User's Manual*, version 0.82, Aug. 2004. http://www.research.att.com/projects/cyclone/online-manual/.
2. D. Kegel, *DKFTPBench*, http://www.kegel.com/dkftpbench/.
3. G. Hoglund and G. McGraw, *Exploiting Software: How to Break Code*, Addison-Wesley, 2004.
4. M. Howard and D. LeBlanc, *Writing Secure Code*, 2nd ed., Microsoft Press, 2004.
5. T. Jim, G. Morrisett, D. Grossman, M. Hicks, J. Cheney, and Y. Wang, "Cyclone: A Safe Dialect of C," *Proceedings of the 2002 USENIX Annual Technical Conference (USENIX'02)*, Monterey, California, June 2002.
6. D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the Slammer Worm," *IEEE Security and Privacy Magazine*, vol. 1, no. 4, July/Aug. 2003, pp. 33-39.
7. C. Myers and B. O'Connor, *Wuarchive-FTP*,. http://www.wu-ftpd.org/.
8. R. Sethi, *Programming Languages: Concepts and Construct*s, 2nd ed., Addison-Wesley, 1996.
9. D.-H. You, *0x82-WOOoou~Happy_new.c - wu-ftpd v2.6.2 off-by-one remote 0day root exploit*, ver. 0.0.5, Aug. 2003. http://x82.inetcop.org/.
10. D.-H. You, *wu-ftpd-2.6.2 off-by-one remote exploit*, Aug. 2003. http://www.securityfocus.com/archive/1/331723