基於 HTTP 協定的跳頻隱藏通道

林士正 劉振緒 林裕欽
逢甲資工所 逢甲資工所 逢甲資工所
e-mail： e-mail： e-mail：
m9489593@fcu.edu.tw liuj@fcu.edu.tw m9430122@fcu.edu.tw

摘要

基於 HTTP 協定的隱藏通道已有許多研究，然而它們大多只使用了單一一種基於 HTTP 協定的資料隱藏方法。在本篇文章中，我們提出了跳頻隱藏通道這個方法，讓我們的隱藏通道能切換多種基於 HTTP 協定的資料隱藏方法，使得我們的隱藏通道更加難以被偵測出來。另外我們將介紹現有基於 HTTP 協定的隱藏通道的偵測方法，並說明我們的方法能夠更加有效躲避偵測的原因。

關鍵詞: 隱藏通道、HTTP 協定、網路安全。

Abstract

HTTP-based covert channels had been studies in past few years. However, most of them employed only one HTTP-based encapsulation method. In this paper, we present a new covert channel scheme that employs multiple HTTP-based encapsulation methods. Our scheme uses the frequency-hopping concept to switch among multiple encapsulation methods. Detection of HTTP-based covert channel will be introduced and we will explain why our covert channel can escape from detection.

Keywords: covert channel, HTTP protocol, network security.

1. 前言

HTTP 是目前在網際網路上應用最廣泛的協定之一。許多的網路服務都是架構在 HTTP 協定之上，所以網際網路管理者通常不會關閉 HTTP 連線，也就是不會關閉埠 80(port 80)。這也代表如果我們希望利用網路做一些不易被人發現的工作，利用 HTTP 將是一個可行的選擇。隱藏通道為架構在一般資料串流(如 TCP,HTTP)上的秘密通訊管道，隱藏通道通常能跨越多種存取控制/監視報告系統。在過去幾年，基於網路架構的隱藏通道之相關研究已經被發表多次。這些研究架構在幾種不同的協定上,包括 TCP/IPv4[10,28], ICMP[16,29,32], DNS[18,26]和 HTTP[3,6,8,9,12,22,25]。然而在 OSI 階層中，低於應用層的隱藏通道工具，他們的藏入資訊方法大多是利用 IP packet identification fields, TCP ACK numbers, ICMP echo message 或者 DNS request/response messages 等方法，然而這些方法都有著一個共通的缺點，那就是能藏入的資料量相當有限。因此考慮到可以藏入的資料量，應用層協定應該較為適合建立隱藏通道。如同我之前所述，HTTP 協定是目前網路上使用最為廣泛而且使用者使用最為頻繁的協定。如果我們的隱藏通道可以使用多種架構在 HTTP 協定上的方法，進而利用這些方法模擬一般使用者的網路瀏覽模式，讓我們的隱藏通道與一般使用者瀏覽網頁時的網路流量類似，那將會使得我們的隱藏通道不易被偵測出來。

Firepass[12]，是一種建立網路通道的工具，我們可以利用它來穿透防火牆。它的原理是將資料藏入合法的HTTP POST請求中。然而這個工具有一個問題，那就是這個工具只使用HTTP POST請求這種藏入資料的方法，而且在一般的使用者瀏覽網頁時，通常不會頻繁的使用HTTP POST請求，一種因此這個工具有較為容易被入侵偵測系統偵測出來的缺點。

Cooking channels[9]允許使用者利用HTTP cookies建立一個通訊通道。這個工具的缺點也是只有利用一種藏入資料的方法。

Cctt[6]是一個利用多種隱藏通道的工具，它利用多種技術將資料藏入資料流中以便通過網路存取控制。然而這種工具利用網路應用層以外的協定，這個工具使用的協定包括了HTTP、TCP和UDP。

我認為如果我們能利用架構於HTTP協定上的不同藏入方法隱藏我們的資料，並且讓我們的網路通訊流量看起來像是一般使用者在瀏覽網頁，那將會使得我們的隱藏通道不易被如入侵偵測系統等監督系統偵測出來。在這篇文章中首先會先簡單介紹HTTP協定，接者會講解我們所使用的幾種藏入方法。再來提出我們切換不同藏入方法所使用的機制。由於能否預防偵測是評估隱藏通道最重要標準，接著便對基於HTTP協定的隱藏通道偵測提出討論。最後是我們的實作與所遭遇的挑戰。

2.相關資料

2.1. HTTP協定

HTTP（HyperText Transfer Protocol）[13,31]是網際網路上應用最為廣泛的一種網路傳輸協議。所有的WWW文件都必須遵守這個標準。設計HTTP最初的目的是為了提供一種發佈和接收HTML頁面的方法。

HTTP的發展是全球資訊網協會和Internet工作小組合作的結果，在一系列的RFC發佈中確定了最終版本，其中最著名的是RFC 2616[13]。在RFC 2616中定義了HTTP/1.1這個今天普遍使用的版本。

HTTP是一個用於在客戶端和伺服器間請求和應答的協議。一個HTTP的客戶端，例如一個web瀏覽器，通過建立一個到遠端主機埠（預設埠為80）的連接，初始化一個請求。一個HTTP伺服器通過監聽埠等待客戶端發送一個請求序列， 就像「GET / HTTP/1.1」(用來請求網頁伺服器的預設頁面)，有選擇的接收像email一樣的MIME消息，此消息中包含了大量用來描述請求各個方面的信息表頭序列。接收到一個請求序列後（如果需要的話，還包含有訊息本體)，伺服器會發回一個應答消息，例如「200 OK」，同時發回一個它自己的消息，此消息的主體可能是被請求的文件、錯誤消息或者其他的一些信息。

HTTP不同於其他基於TCP的協議，例如FTP。在HTTP中，一旦一個特殊的請求（或者請求的相關序列）完成，連接通常被中斷。由於缺乏持久連接的特性，當需要保持用戶狀態時，對網頁設計者來說，會偶然產生一些問題。而大部分這些問題的解決方法為使用「cookies」。

伺服器可以利用Cookies包含信息的任意性來篩選並經常性維護這些信息，以判斷在HTTP傳輸中的狀態。Cookies最典型的應用是判定註冊用戶是否已經登錄網站，用戶可能會得到提示，是否在下一次進入此網站時保留用戶信息以便簡化登錄手續，這些都是Cookies的功用。另一個重要應用場合是「購物車」之類處理。用戶可能會在一段時間內在同一家網站的不同頁面中選擇不同的商品，這些信息都會寫入Cookies，以便在最後付款時提取信息。

HTTP由唯一資源定位器(URI, Uniform Resource Identifiers)定位。創造這種地址定位的語法為了HTML的連結。

HTTP 定義了八種請求方法,用來表示出客戶端所希望執行的資源。以下就簡單介紹這八種請求方法。

(1) GET

用來請求特定的資源。為現今在網際網路上使用最為廣泛的請求方法。

(2) HEAD

用來請求與 GET 一樣的回應。然而回應並不包含回應訊息本體(response body)。

(3) DELETE

刪除特定資源。

(4) POST

傳遞資料到特定的資源。而資料存放在請求訊息本體(request body)裡。

(5) PUT

上傳特定資源。

(6) TRACE

請求伺服器回應前一次所接受到的請求訊息,使我們能判斷我們上一次的請求訊息再傳送的過程中是否被中介伺服器改變過。

(7) OPTIONS

請求伺服器告知所支援的 HTTP 方法。

(8) CONNECT

當使用 proxy 伺服器時,可利用這個請求轉換為 SSL tunnel。

一般在使用 HTTP 伺服器時,GET 請求方法與 POST 請求方法是使用較為頻繁的,也因此若是我們可以將資訊隱藏於這兩種請求方法,應該較不容易被懷疑。

3. 資料隱藏方法

多種基於 HTTP 協定的資料隱藏方法將會被使用於我們的研究,其中某些方法是使用現有的技術,包括了 Firepass[12]所使用的 HTTP POST 請求資料隱藏方法與 HTTP 回應訊息本體資料隱藏方法,Cooking channels[9]所使用的 HTTP 請求 cookie 資料隱藏方法與 HTTP set cookie 資料隱藏方法。

以下便一一講解我們所用到的資料隱藏方法。請注意由於我們的研究是架構於 HTTP 協定,因此客戶端到伺服器端所的資料隱藏於請求訊息,伺服器端到客戶端的資料隱藏於回應訊息,以下將分分別敘述。

3.1 HTTP 請求資料隱藏方法

如同 2.1 節末所述,我們的隱藏通道將隱藏於 HTTP 協定中的 GET 與 POST 請求。由於 GET 協定並不包含訊息本體(message body),因此我們使用 GET 請求來隱藏我們的資料時,只能將隱藏資料藏於 GET 請求的表頭(header),3.1.1 節到 3.1.4 節提出了四種隱藏資料於 GET 表頭的方法。而 POST 請求方法包含訊息本體,我們可以直接將隱藏資訊藏於訊息本體,在 3.1.5 節提出了隱藏資料於 POST 請求的方法。

3.1.1 HTTP GET 請求 URI 資料隱藏方法

根據 HTTP 協定,HTTP 表頭的 URI 欄位是使用者可以自己定義的欄位之一,而且當一般使用者在瀏覽網頁時,HTTP GET 請求是最常被使用的協定,假如我們可以將隱藏訊息加入 HTTP GET 請求表頭的 URI 欄位,那將會使得我們隱藏訊息的意圖較不容易被偵測出來。

舉例而言,當使用者瀏覽某個擁有多張影像的網頁時通常會發送出多個 GET 請求,因此我們可以將隱藏訊息藏入於 URI 欄位的影像名稱或路徑中,例如我們的隱藏通道要傳遞”our cover channel start”的訊息,這時我們便可以傳遞包含以下 URI 的 GET 請求”GET /our/cover/channel/start.gif”。

在我們隱藏訊息於URI時有一點事項要注

意，那就是我們的路徑階層或檔案名稱不宜過長，若是URI的長度過長，將會有被偵測出來的風險。因此當我們所要隱藏的資料量較多時，我們可以將隱藏資料分散於多個GET請求傳遞，延續上一個例子，當我們的隱藏通道要傳遞”our cover channel start”的訊息，這時我們便可以傳遞兩次GET請求，”

GET /our/cover.gif

GET /channel/start.jpg

”。

就如同我前面所述，當使用者在瀏覽單一網頁的時候，通常會送出多個GET請求，因此當我們在短時間內送出多個GET請求，這個行為並不會被視為異常。

3.1.2 HTTP GET請求CGI資料隱藏方法

在CGI(Common Getaway Interface)協定裡的 script-URI 也是一個適合傳遞隱藏訊息的位址。根據 RFC 3875 [27] 所述“the mapping form client request URI to choice of script is defined by the particular server implementation and its configuration.”這代表我們可以利用script-URI傳遞任意訊息。例如我們希望傳遞隱藏訊息”our cover channel start”，我們可以將訊息隱藏於 script-URI 中，傳送出如下的

“GET /test?FHSS=cover&channel=start”。

這裡有一點我們需要注意的。根據RFC2396[1]的第二節所定義，針對URL的編碼有一些保留字元，例如?、@、;等字元。當我們的隱藏資料保含這些字元時，我們需要把這些保留字元轉換成以一個百分比符號%開頭並接著兩個十六進位數字的樣式。例如?= %3F, @= %40, and ;= %3B。

這個方法有者與HTTP GET 請求URI資料隱藏方法相同的缺點，那就是URI的長度不宜過長，若是URI的長度過長，將會有被偵測出來的風險。因此當我們所要隱藏的資料量較多時，我們可以將隱藏資料分散於多個GET請求傳遞。

利用這種方法讓我們的 GET 請求不只有3.1.1節所述，只能將隱藏資訊隱藏於URI的檔案路徑以及檔案名稱中，還能隱藏資訊於script-URI 中。混合使用 HTTP GET 請求URI資料隱藏方法與HTTP GET 請求CGI 資料隱藏方法這兩種方法，可以使我們利用GET方法所請求的 URI 更貼近真正使用者瀏覽網路時的行為，就算入侵偵測系統針對 URI 欄位作偵測，也不容易發現異常。

3.1.3 HTTP Referer URI資料隱藏方法

根據HTTP協定,Referer欄位指定用戶端最後一個頁面的URI位址。例如，如果用戶訪問頁面A，然後點擊在頁面A上到頁面B的鏈結，訪問頁面 B 的 HTTP 請求會包括一個Referer 欄位，該欄位會包括這樣的資訊“這個請求是來自於頁面A”。

因此我們可以利用前面兩節所述的URI隱藏方法，將隱藏訊息藏入Referer欄位

3.1.4 HTTP 請求cookie資料隱藏方法

根據 RFC 2109 [19] 所述“the name of the state information (“cookie”) is NAME, and its value is VALUE. (…) The VALUE is opaque to the user agent and may be anything the origin server chooses to send, (…)”，這代表在HTTP協定中,cookie基本上是可以讓使用者自行定義的，因此我們可以將資料隱藏於cookie中。舉例而言，當我們要利用 cookie 傳遞”our cover channel start”時，我們可以在 HTTP 表頭裡定義如下的cookie，

”cookie: our=cover, channel=start”。

然而根據”How to cook a covert channel“[9]，利用cookie隱藏訊息有著許多的限制。由於客戶端的cookie是由伺服器利用set cookie控制，基本上客戶端沒有接收到set cookie，是不會更改cookie，不過監控者不太可能監控所有的 set cookie 與cookie並察覺某個cookie未接收set cookie而變更(也許這個客戶端上次接收到 set cookie是十年前的事)，而且cookie是有時效性的，所以客戶端偶而傳送不同的 cookie到伺服器是不容易被察覺的。但若是監控者發現某段時間內客戶端沒有接收到伺服器的set cookie 指令，卻頻繁的更動 cookie 訊息，那我們的隱藏通道將有被偵測出來的可能，因此cookie通常用來傳遞我們隱藏通道的控制訊息，並不適合用來隱藏並傳遞大量資料。並且在使用cookie隱藏訊息時需要跟伺服器端的set cookie回應密切配合，才能避免被懷疑。

3.1.5 HTTP POST請求資料隱藏方法

HTTP POST 請求資料隱藏方法利用 HTTP協定中的 POST 請求來隱藏資料。由於 POST請求方法包含訊息本體(message body)，我們可以將隱藏資料存放在請求訊息本體裡。HTTP POST請求資料隱藏方法的好處是當我們存放資料到訊息本體(message body)，基本上是沒有長度限制的。然而在一般使用者瀏覽網頁時，基本上使用到 HTTP POST 請求的機會相對較小，若是在我們的隱藏通道傳遞過程中，HTTP POST請求出現次數過於頻繁，將導致被入侵偵測系統發現的可能。

因此在我們的隱藏通道裡，HTTP POST請求資料隱藏方法將不會被頻繁使用，舉例而言，在我們的隱藏資訊傳遞過程中，可能每使用五次其他資料隱藏方法(例如三次 HTTP GET 請求 URI 資料隱藏方法與兩次 HTTP Referer URI 資料隱藏方法)後，才會使用一次HTTP POST請求資料隱藏方法。

3.2 HTTP 回應資料隱藏方法

3.2.1 HTTP 回應訊息本體資料隱藏方法

HTTP 回應訊息本體資料隱藏方法將資料隱藏於 HTTP 協定中的回應訊息本體(response message body)。舉例而言，當我們要傳遞”our cover channel start”時，我們可以直接在回應訊息本體中加入”our cover channel start”的訊息。

3.2.2 回應檔案資料隱藏方法

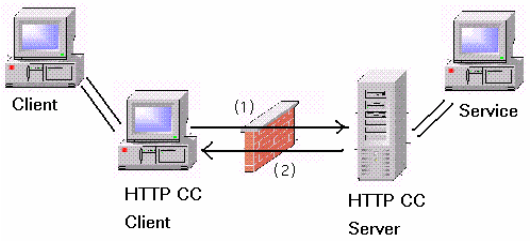
當使用者瀏覽網頁時，時常會請求伺服器端回傳某些檔案文件。因此我們可以將隱藏訊息藏於檔案中傳遞。關於將機密訊息藏入數位媒體的技術已有許多相關研究[4,14,15,20,21,23,24,30]。因此對於回應檔案資料隱藏方法，我們有許多隱藏資料於檔案的技術選擇。

舉例而言，我們可以利用Y.-K. Lee 等人[21]所提及的方法，將我們所要傳遞的資訊隱藏於JPEG影像中，再將影像傳回給客戶端。

3.2.3 HTTP set cookie資料隱藏方法

HTTP set cookie資料隱藏方法與HTTP 請求 cookie 資料隱藏方法基本上都是利用cookie來隱藏資料。他們的不同之處在於HTTP請求 cookie 資料隱藏方法將資料隱藏於客戶端傳送到伺服器端的請求表頭中的 cookie 欄位，而HTTP set cookie資料隱藏方法將資料隱藏於伺服器到客戶端的回應表頭中的 set cookie欄位。而且HTTP set cookie資料隱藏方法並沒有3.1.4節所提出的限制，因為伺服

器端頻繁傳送不同的 set cookie 值是被允許的。例如我們在瀏覽購物網站時，若頻繁的更動購物車裡面的內容，將會使得伺服器一直傳送不同的 set cookie 值給客戶端。因此若是我們較為頻繁使用 HTTP set cookie 資料隱藏方法，這個行為並不會被視為異常。



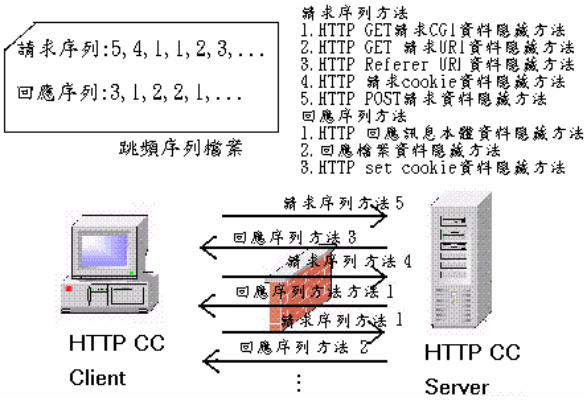
圖一 基於 HTTP 協定的跳頻隱藏通道架構圖

4. 基於 HTTP 協定的跳頻隱藏通道

我們的方法「基於 HTTP 協定的跳頻隱藏通道」如圖一所示，當 client 希望獲得某個 service 而不希望這個連線被發現，或者連線被存取控制系統擋住時，便可利用我們的隱藏通道，將兩端互相溝通的訊息藏入符合 HTTP 協定的請求與回應訊息中，並利用 port 80 傳遞，傳遞的過程在監控者看起來與一般的網頁瀏覽訊息一樣。「基於 HTTP 協定的跳頻隱藏通道」由 HTTP CC(Cover channel) Client 與 HTTP CC(Cover Channel) Server 組成，模擬正常客戶端與伺服器端的 HTTP 請求與回應，並將欲隱藏的資料利用前面第 3 節所述的幾種方法隱藏於 HTTP 的請求與回應訊息中。在我們隱藏通道中，使用了之前所述的五種請求隱藏方法與三種回應隱藏方法。

圖一的(1)代表在請求的訊息裡，我們可以使用五種隱藏方法，包括 HTTP GET 請求 CGI 資料隱藏方法、HTTP GET 請求 URI 資料隱藏方法、HTTP Referer URI 資料隱藏方法、HTTP POST 請求資料隱藏方法與 HTTP 請求 cookie 資料隱藏方法。圖一的(2)代表在回應的訊息裡，我們可以使用三種隱藏方法，包括 HTTP 回應訊息本體資料隱藏方法、回應檔案資料隱藏方法以及 HTTP set cookie 資料隱藏方法。

我們的方法「基於 HTTP 協定的跳頻隱藏通道」使用了「跳頻序列」的技術，這個技術是從 FHSS(Frequency-hopping Spread Spectrum)得到靈感，FHSS 是一種應用在無線電的技術，利用傳送者與接收者預先定義好的近似隨機序列(pseudorandom sequence)連續在不同頻率的頻道上切換。應用這個觀念在我們的研究上，我們可以將不同的資料隱藏方法視為不同的頻道，並依據事先所定義好的「跳頻序列」交替切換五種請求隱藏方法與三種回應隱藏方法。



圖二 跳頻序列

舉例來說，如圖二所示，我們設定請求序列方法 1 代表 HTTP GET 請求 CGI 資料隱藏方法、2 代表 HTTP GET 請求 URI 資料隱藏方法、3 代表 HTTP Referer URI 資料隱藏方法、4 代表 HTTP 請求 cookie 資料隱藏方法、5 代表 HTTP POST 請求資料隱藏方法，回應序列方法 1 代表 HTTP 回應訊息本體資料隱藏方法、2 代表回應檔案資料隱藏方法、3 代表 HTTP set cookie 資料隱藏方法。假設我們的的「跳頻序列」包含了請求序列:5,4,1,1,2,3,…與回應序列:3,1,2,2,1,…，第一次傳遞訊息時，根據請求序列，第一個數字 5 代表使用 HTTP POST 請求資料隱藏方法，而根據回應序列，第一個

數字3代表HTTP set cookie資料隱藏方法，那當HTTP CC Client第一次傳遞訊息給HTTP CC Server時將會使用HTTP POST請求資料隱藏方法，而HTTP CC Server第一次傳遞訊息給HTTP CC Client時將會使用HTTP set cookie資料隱藏方法。第二次傳遞訊息時，根據請求序列，第二個數字4代表使用HTTP請求cookie資料隱藏方法，而根據回應序列，第二個數字1代表HTTP回應訊息本體資料隱藏方法，那當HTTP CC Client第二次傳遞訊息給HTTP CC Server時將會使用HTTP set cookie資料隱藏方法，而HTTP CC Server第二次傳遞訊息給HTTP CC Client時將會使用HTTP回應訊息本體資料隱藏方法。以此類推。

5. 偵測

我們之所以需要建立隱藏通道，最重要的便是希望我們傳送資料的過程是隱密且不為人知的，因此能否被偵測出來便成為評估隱藏通道的標準。

根據Simon Castro[8]等人所述，偵測基於HTTP協定的隱藏通道大致分為三類，包括(1)基於特徵模式的偵測方法(2)基於協定的偵測方法(3)基於行為模式的偵測方法。以下針對這三種方法詳細講解:基於特徵模式的偵測方法主要是利用建立與更新具有某些明顯攻擊特徵的資料庫，藉由比對網路的流量與資料庫，從中找出可能的攻擊特徵並通知系統管理者。經由基於特徵模式的偵測方法，可以歸納出其中許多攻擊特徵是屬違反正常的協定定義，因此衍生出基於協定的偵測方式，此種方式專注於偵測網路流量封包是否有明顯違反協定的定義。若是隱藏通道設計者較為注意設計的細節以及協定的定義，將可以避開上述這兩種方式的偵測。最後一種基於行為模式的偵測，首先利用統計，機率，類神經網路等方法建立正常使用者（包含使用者、工作站、伺服器和網路流量等）的行為模型，再判斷目前的資料流量是否與正常使用者的行為模式相符，進而偵測出網路流量的異常行為模式。隱藏通道較不容易避開此種偵測方式。

之前的一些基於HTTP協定的隱藏通道程式，通常都會產生一些與正常使用者所不同的網路流量，這些程式將有被前述(3)基於行為模式的偵測方式偵測出來的可能。K.Borders[3]提出了一種基於行為模式的偵測方式，利用請求出現的頻率、頻寬的使用、請求的延遲時間和傳送流量的大小等資訊建立一個模型，藉此比對出可能的異常行為。因此藉由這兩篇論文我們可以得知，如何使我們隱藏通道的網路流量貼近正常使用者的行為模式，將是能否避開偵測的關鍵。

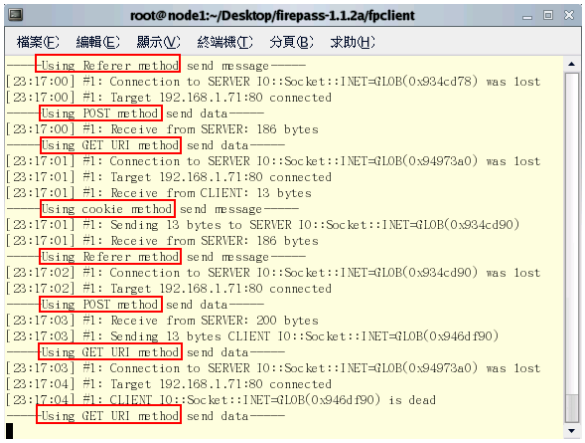
我們的方法「基於HTTP協定的跳頻隱藏通道」利用「跳頻序列」來預防偵測。首先，由於我們利用多種的資料隱藏方法來傳遞訊息，這將可以避免某種請求或回應方法使用過於頻繁。舉例而言，在一般使用者瀏覽網路時，並不會經常使用HTTP POST請求，若是頻繁的始用HTTP POST請求傳遞隱藏訊息，將會提高被偵測出來的可能性，因此我們使用多種資料隱藏方法來減低使用HTTP POST請求的機率。

接著，由於我們使用「跳頻序列」來決定我們所使用的資料隱藏方法，若是我們可以仿照(3)基於行為模式的偵測方法中所提及建構正常使用者行為模型的方式，先經由探測現行的網路流量，取得請求出現的頻率、頻寬的使用、請求的延遲時間和傳送流量的大小等資訊，再使用統計、機率、類神經網路[17]，或者SVM(Support Vector Machine)[2,5,11]等方法建構一個正常使用者的網路流量模型，再設定各種隱藏方法的使用頻率，最後利用「跳頻序列」使我們的隱藏通道模擬正常使用者的行為，那將會使得我們的隱藏通道更難被偵測出來。例如，我們經由統計發現在某個網路環

境下，使用者瀏覽網頁時，下載影像檔案的次數相當頻繁，我們便可以調高我們「跳頻序列」中使用 HTTP GET 請求 URI 資料隱藏方法和回應檔案資料隱藏方法的使用率，藉以躲避被判斷為異常行為的可能。

6. 實作

經由前兩節分別介紹我們的基於 HTTP 協定的跳頻隱藏通道方法與預防偵測理論後，以下分兩小節講解我們的實作。



圖三 HTTP CC Client 實作執行過程

6.1 實作基於 HTTP 協定的跳頻隱藏通道

由於我們的基於 HTTP 協定的跳頻隱藏通道架構於 HTTP CC Client 與 HTTP CC Server，以模擬正常客戶端與伺服器端的 HTTP 請求與回應，並將隱藏訊息隱藏於 HTTP 的請求與回應訊息中。這個架構與 Firepass[12]的架構相同，再加上 Firepass 以 Perl 語言實作，Perl 語言有很好的網路與字串處理能力，所以我們的實作以 Firepass 程式為基礎，加入了多種的資料隱藏方法，並將跳頻序列的實作加入程式中。

Firepass 只有實作了 HTTP POST 請求資料隱藏方法與 HTTP 回應訊息本體資料隱藏方法。我們的目的是使用第 3 節所提及的五種請求資料隱藏方法與三種回應資料隱藏方法，並且可以藉由讀取一個紀錄跳頻序列的檔案，切換多種不同的資料隱藏方法。圖三顯示了我們的程式執行時，在 HTTP CC Client 傳送隱藏資訊到 HTTP CC Server 的過程中，交替使用了多種的資料隱藏方法。

在實作的過程中，有一個較為特別的地方，那就是如 3.1.2 小節所述，針對 URL 的編碼有一些保留字元。為了處理這個問題，我們使用了 LWP(Library for Web access in Perl)函式庫中的 uri_escape 方法來解決這個問題。

接下來的挑戰便是如何實作一個方法能自動建立模擬使用者行為的跳頻序列。

6.2 實作自動模擬使用者行為的跳頻序列

如同上一小節末所說的，接下來的研究重點便是如何實作一個方法能自動建立模擬使用者行為的跳頻序列，以達到預防偵測的目的。我們的目的是讓我們的實作能在任意的網路環境下，探測並蒐集正常使用者瀏覽網頁時的網路流量，取得請求出現的頻率、頻寬的使用、請求的延遲時間和傳送流量的大小等資訊，再使用統計、機率、類神經網路[17]，或者 SVM(Support Vector Machine)[2,5,11]等方法建構一個正常使用者的網路流量模型，進而自動建立模擬使用者行為的跳頻序列，然後套用到我們的隱藏通道上。為了達到這個目的，我們先以逢甲大學的網路環境為代表，藉由研究逢甲大學伺服器的記錄檔，統計出逢甲大學學生使用網頁的行為模式，接著將結果導入我們的跳頻序列，以讓我們的隱藏通道能夠貼近使用者的行為模式，並從研究問題的過程中，找出一個實作方法，能用來在所有網路環境下建構使用者的行為模型。

在我們完成自動模擬使用者行為的跳頻序列的實作後，每當我們到一個陌生的網路環境時，我們可以先探測並蒐集正常使用者瀏覽

網頁時的網路流量，進而使用我們的方法自動建立模擬使用者行為的跳頻序列，然後套用到我們的隱藏通道上。如此便能讓我們的隱藏通道流量貼近正常使用者瀏覽網頁時的網路流量。

7. 結論

我們所提出的方法「基於 HTTP 協定的跳頻隱藏通道」，利用了多種符合 HTTP 規範訊息格式的資料隱藏方法來隱藏資訊，並利用探測與蒐集正常使用者瀏覽網頁時的網路流量建立模擬使用者行為的「跳頻序列」，最後使用「跳頻序列」來切換不同的資料隱藏方法，以讓我們的隱藏通道模擬使用者的正常行為模式，使得我們的隱藏通道在傳遞隱藏資訊的過程中，在監控者看起來，與一般使用者瀏覽網頁時的網路流量相同，進而避免被偵測出異常的可能。

我們的「基於 HTTP 協定的跳頻隱藏通道」與現有基於 HTTP 協定的隱藏通道的技術相比，將使得隱藏通道的使用更加不容易被偵測出來。

我們所遇到的挑戰是如何提出有效的方法，以建構出精確模擬正常使用者行為的「跳頻序列」，這部份將是我們後續研究的重點。

理論上，我們所提出的「基於 HTTP 協定的跳頻隱藏通道」將能有效的躲避偵測，然而我們還需要實驗的數據來驗證，因此我們後續將會盡快提出實驗數據來證明我們的研究。

參考文獻

[1] T. Berners-Lee, R. Fielding, L. Masinter, “RFC2396: Uniform Resource Identifiers (URI): Generic Syntax ”, 1998.

[2] B.E. Boser, I.M. Guyon, and V.N. Vapnik. “A training algorithm for optimal margin classifier”, ***In Proc.5th ACM Workshop on Computational Learning Theory***, pages 144-152, Pittsburgh, PA, July 1992.

[3] K. Borders, “Web Tap – Detecting Covert Web Traffic”, 2004, http://www.eecs.umich.edu/~aprakash/papers/borders-prakash-ccs04.pdf

[4] A. Brown, S-Tools version 4.0, 2000, ftp://ftp.funet.fi/pub/crypt/mirrors/idea.sec.dsi.unimi.it/code/s-tools4.zip

[5] C.J.C. Burges. “Simplified support vector decision rules”, ***In International Conference on Machine Learning***, pages 71-77. 1966.

[6] S. Castro, Covert Channel Tunneling Tool(cctt), 2003, http://gray-world.net/pr_cctt.shtml

[7] S. Castro, Cooking channels, 2006, http://gray-world.net/pr_cook_cc.shtml

[8] S. Castro, “Covert Channel and Tunneling over the HTTP protocol Detection: GW implementation theoretical design”, 2003, http://gray-world.net/projects/papers/html/cctde.html

[9] S. Castro, “How to cook a covert channel”, 2006, http://gray-world.net/projects/cooking_channels/hakin9_cooking_channels_en.pdf

[10] S. Cabuk, “IP cover timing channels: design and detection”, 2004, http://www.cs.jhu.edu/~fabian/courses/CS600.624/covert.pdf

[11] C. Cortes and V. Vapnik, Support vector networks, ***Machine Learning***, 20:1-25, 1995.

[12] A. Dyatlov, Firepass, 2003, http://gray-world.net/pr_firepass.shtml

[13] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, “RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1”, 1999.

[14] J. Fridrich, “A new steganographic method for palette-based images”, ***In Proceedings of IS&T PICS***, pp.285-289, Savannah, Georgia, 1999.

[15] J. Fridrich and R. Du, “Secure steganographic methods for palette-based images”, ***In Proceedings of 3rd International Workshop on Information Hiding***, pp.47-60, Dresden, Germany, 1999.

[16] T. Gil, IP-over-ICMP using ICMPTX HOWTO, 2005, http://thomer.com/icmptx/

[17] Abdi, H. “A neural network primer.” ***Journal of Biological Systems***, 2, 247-281, 1994.

[18] D. Kaminsky, IP-over-DNS tunneling using

Ozyman, 2004, http://www.doxpara.com/

[19] D. Kristol, L. Montulli, “RFC 2109: HTTP State Management Mechanism ”, 1997.

[20] Y.-K. Lee and L.-H. Chen, “High capacity image steganographic model”, ***IEE Proceedings Vision, Image and Signal Processing***, Vol. 147, Issue 3, pp.288-294, 2000.

[21] Y.-K. Lee and L.-H. Chen, “Secure Error-Free Steganography For JPEG Images,” ***I.J. Pattern Recognition and Artificial Intelligence (IJPRAI)***, Vol. 17, No. 6, pp. 967-982, 2003.

[22] P. Leboutillier, HTTunnel, 2005, http://sourceforge.net/projects/httunnel/

[23] L. M. Marvel, C. G. Boncelet, Jr. and C. T. Retter, “ Spread spectrum image steganography”, ***IEEE Trans. on Image Processing***, Vol. 8, No. 8, pp.1075-1083, 1999.

[24] R. Machado, Stego, http://www.stego.com

[25] P. Padgett, Corkscrew, 2001, http://www.agroman.net/corkscrew/

[26] T. Pietraszek, IP-over-DNS tunneling using DNScat, http://tadek.pietraszek.org/projects/DNScat/

[27] D. Robinson, K. Coar, “RFC 3875: The Common Gateway Interface (CGI) Version 1.1”, 2004.

[28] C. Rowland, Covert Channels in the TCP/IP Protocol Suide, 1996, http://www.firstmonday.org/issues/issue2_5/rowland/

[29] D. Stødle, ptunnel, 2005, http://www.cs.uit.no/~daniels/PingTunnel/

[30] D. Upham, Jstego, ftp://ftp.funet.fi/pub/crypt/steganography/

[31] Wikipedia, Hypertext Transfer Protocol, http://en.wikipedia.org/wiki/HTTP

[32] I. Zelenchuk, Skeeve, 2004, http://gray-world.net/poc_skeeve.shtml