

# [zz]“西廂計劃”原理小解



從 [farseerfc.wordpress.com](http://farseerfc.wordpress.com) 導入

好神奇的想法，先存着，以後慢慢研究

原文：<http://blog.youxu.info/2010/03/14/west-chamber/>

待月西廂下，迎風戶半開。隔牆花影動，疑是玉人來。

最近推上最流行的一個關鍵詞是“西廂計劃”，這個計劃名字取得很浪漫，客戶端叫做張生，對，就是西廂記裏面那個翻牆去見崔鶯鶯小姐的張生；顯然，服務器端必然叫做崔鶯鶯。客戶端的張生是最重要的部件，可以不依賴於服務端工作。因西廂計劃的作者只是簡要的介紹了一下原理，其他報道又語焉不詳，我當時就覺得很好奇，花了昨天一個晚上詳細讀了一下源代碼，終於知道怎麼回事了，覺得原理非常漂亮，所以寫篇文章介紹總結一下。

先說大方向。大家都知道，連接被重置的本質，是因收到了破壞連接的一個 TCP Reset 包。以前劍橋大學有人實驗過，客戶端和服務器都

忽略 Reset, 則通信可以不受影響。但是這個方法其實只有理論價值, 因為絕大多數服務器都不可能忽略 Reset 的 (比如 Linux, 需要 root 權限配置 iptables, 而且這本身也把正常的 Reset 給忽略了)。只要服務器不忽略 Reset, 客戶端再怎麼弄都沒用, 因為服務器會停止發送數據, Reset 這條連接。所以, 很多報道說西廂計劃是忽略 Reset, 我從源代碼來看應該不是這樣。在我看來, 西廂計劃是利用了牆的一個可能的弱點-牆只在連接發起的時候把一個 TCP 連接加入監聽序列, 如果牆認為這個連接終止了, 就會從監聽序列中去掉這條記錄, 這樣, 這條連接上後續的包就不會被監聽。西廂計劃就是讓牆“認為”這個連接終止的一個絕妙的方法。只要牆認為這個連接兩端都是死老虎, 牆就不會觸發關鍵詞檢測, 其後所有的數據, 都不存在連接被重置的問題了。

如何讓一個連接置之死地而後生, 就是西廂計劃那幫黑客神奇的地方了。這也不是一日之功。首先, 這幫牛人發現, 牆的是一個入侵檢測系統, 把含有關鍵字的包當成一種“入侵”來對待。採取這種設計有很多好處, 但缺點是入侵檢測系統可能具有的問題, 牆都可能有。西廂計劃主頁上那篇著名的論文就是講這些七七八八的漏洞的。可以說處理這些七七八八的漏洞是非常困難的, 迫使牆的設計者“拆東牆, 補西牆”。這樣補來補去, 外表看起來好像很牛逼的牆, 其實有很多本質上無法簡單修補的漏洞, 其中有一個致命的, 就是 TCP 連接狀態的判定問題。出於入侵檢測系統這種設計的侷限, 牆沒有, 也沒辦法準確判定一條 TCP 連接的狀態, 而只是根據兩邊收到的數據來“推測”連接的狀態。而所有的關鍵詞檢測功能, 都是基於“連接還活着”的這個推測的結果的。因為牆的規則是在連接發起的時候開始對這條連接的檢測, 在連接終止的時候停止對這條連接的檢測, 所以, 一旦對連接的狀態推測錯誤, 把還活着的連接當成已經關閉的連接, 牆就會放棄對這條連接上隨後所有的包的檢測, 他們都會都透明的穿過牆的入侵檢測。

上面只是想法, 具體到 TCP 協議實現這一層, 就要只迷惑牆, 還不能觸及我要通信的服務器。最理想的情況下, 在任何有效通信之前, 就能讓牆出現錯誤判斷, 這些, 就需要對 TCP 協議有深刻理解了。西廂計劃的那幫黑客, 居然真的去讀 TCP 幾百頁的 RFC, 還居然就發現了方法 (這裏我假設讀者都知道 TCP 的三次握手過程和序列號每次加一的規則)。我們都知道, 三次握手的時候, 在收到服務器的 SYN/ACK 的時候, 客戶端如果發送 ACK 並且序列號+1 就算建立連接了, 但是客戶端如果發送一個序列號沒 +1 的 FIN (表示連接終止, 但是服務器知道, 這時候連接還沒建立呢, FIN 這個包狀態是錯的, 加上序列號也是錯

的，服務器自己一判斷，就知道這個包是壞包，按照標準協議，服務器隨手丟棄了這個包），但這個包，過牆的時候，在牆看來，是表示連接終止的（牆是 *made in china*，是比較山寨的，不維護連接狀態，並且，牆並沒有記下剛纔服務器出去的 **SYN/ACK** 的序列號，所以牆不知道序列號錯了）。所以，牆很高興的理解連接終止，舒了一口氣去重置其他連接了，而這個連接，就成了殭屍，牆不管你客戶端了，而這時候，好戲纔剛剛開始。

事實上，牆是雙向檢測的（或者說對每個包都檢測的），因此，對服務器和客戶端實現相同的對待方法，所以，牆不管客戶端還不行，假如服務端有關鍵詞傳給客戶端，牆還是有可能要發飆的（這裏說有可能，因我也不知道）。所以，最好的辦法就是，讓服務端也給牆一個終止連接的標誌就好了。可是這個說起來簡單，做起來難，怎麼能讓不受自己控制的服務器發一個自己想要的包呢？西廂計劃的那幫黑客，再次去讀幾百頁的 RFC，令人驚訝的發現，他們居然在 RFC 上發現了一個可以用的特性。我們上面說了，三次握手的時候，在收到 **SYN/ACK** 後，客戶端要給服務器發送一個序列號+1 的 **ACK**，可是，假如我不+1 呢，直接發 **ACK** 包給服務器。牆已經認你客戶端是死老虎了，不理你了，不知道你搞什麼飛機，讓這個 **ACK** 過了。可是服務器一看，不對啊，你給我的不是我期待的那個序列號，RFC 上說了，TCP 包如果序列號錯了的話，就回復一個 **Reset**。所以，服務器就回復了一個 **Reset**。這個 **Reset** 過牆的時候，牆一看樂了，服務器也終止連接了，好吧，兩邊都是死老虎了，我就不監聽這條連接了。而至於客戶端，這個服務器過來的 **Reset** 非常好識別，忽略就是。隨後，客戶端開始正確的發送 **ACK**，至此，三次握手成功，真正的好戲開始，而牆則認客戶端和服務器都是死老虎，直接放過。所以，張生就這樣透明的過了牆。至於過牆以後所有的事情，《西廂記》裏面都有記載，各位讀者自行買書學習。

現在的西廂計劃客戶端，即“張生”模塊的防連接重置的原理就是這樣，服務器端，即鶯鶯模塊的實現也是類似的。防 DNS 那個，不懂 DNS 協議，所以看不懂。我猜想，因開發人員都是黑客，所以自然喜歡用最經得起折騰和高度定製的 Linux 開發。現在看西廂計劃的實現，因依賴於 Linux 內核模塊 **netfilter**，在 Linux 上如魚得水，但往其他平臺的移植可能是個亟待解決的問題。我覺得，在其他平臺上，可以通過 **libpcap** 和 **libnet**，在用戶態實現相同的功能，就是有點麻煩而已，有興趣的懂網絡的可以照西廂計劃原理，在家自行做出此功能；當然，全中國人民都用 Linux 最好：)

PS 1: 據說是西廂計劃一個作者畫的原理圖：<http://img.ly/Dli>

PS 2: 我對 TCP 的理解僅限於課本，如果上面的對技術的理解有錯，請大家指出。

PS 3: 有些漏洞，可能是設計上本質缺陷，不是那麼容易修復的。

PS 4: 除了最後一個圖，本文沒有其他相關鏈接，如需相關資料，自行Google。