SLIPPING IN THE WINDOW: TCP RESET ATTACKS

By:

Paul A. Watson
CISSP, CISM, CCSP, CCSE, MCSE+Security, etc.

Technical Whitepaper

www.terrorist.net

October 30, 2003

Latest Revision: December 25, 2003

ABSTRACT

SLIPPING IN THE WINDOW: TCP
RESET ATTACKS

By: Paul A. Watson

The threats posed by TCP injection attacks have long been a concern for Internet security researchers. The original TCP specification (USC, 1981) included features that originally intended to prevent reception of duplicate or disordered packets, but also provided protection against injection and spoofing attacks. The 32-bit sequence number ensures that received packets can be pieced together into the proper order, but also provided a significant hurdle for those seeking to inject false data into unseen TCP data steams. Although the TCP Reset attack has been recognized as a potential threat for years, little has been written on the subject and there appears very little understanding of the risks. This paper is intended to examine the real-world risks presented by TCP Reset attacks.

TABLE OF CONTENTS

A Brief Background....4
Separating Fact from FUD....5
TCP Window Considerations....7
The TCP Reset Attack....10
Window Scaling....10
OS Window Size Observations....12
Realistic Expectations and Equations....13
Initial Sequence Number Prediction....14
TCP Source Port Considerations....16
Practical Attack Requirements Assuming Ideal ISN....19
Observed Results of Blind TCP RESET Attacks....23
TCP Reset Testing Software Used....24
Test Results for 250 Packets Per Second (Upstream DSL Speed)....25
Test Results for 4,370 Packets Per Second (T-1 Speed)....26
Protecting From Attack....27
Appendix A: Adjusting Default TCP Window Size....29
Table of Figures....31
Bibliography....32

A Brief Background

For many years, numerous researchers have discussed in depth a wide variety of attacks against the TCP protocol. One particular attack method, the sequence number attack, has received much attention. The various sequence number attacks include blind spoofing, session hijacking, and packet injection. The 1994 "Christmas day attack" (Shimomura, 1995) on the home network of security researcher Tsutomu Shimomura was the first widely publicized incident involving TCP spoofing. News of the attack was slow to spread at first, with the first Usenet posting (Watson, 1995) detailing the attack appearing just under a month after the incident, and a CERT advisory (CA-1995-01, 1995) following shortly thereafter. The attack eventually generated an enormous amount of interest and research in sequence number prediction and other TCP spoofing attacks, as well as national media attention.

The reason many of these attacks were possible was due in large part to poor initial sequence number (ISN) selection. The poor ISN selection resulted in sequence numbers that were simple to predict, thereby providing the attacker a window of opportunity. Over the years, this window of opportunity has slowly closed, however, as many vendors eventually adopted stronger ISN selection methods. With nearly random sequence numbers, an attacker might be required to generate billions of TCP packets in a very short time frame in order to successfully implement another "Christmas attack." An even greater hurdle was the fact that bandwidth limitations prevented the attacker from being able to transmit these billions of packets in a timely enough manner as to make the attack likely to succeed. As a result, sequence number attacks have been widely regarded by many as no longer posing any significant risk.

During the time that vendors were strengthening ISN selection algorithms, something else was happening as well. The Internet grew dramatically. Tier-1 Internet providers upgraded peering connections from Megabit links to Gigabit links. Businesses upgraded their old, slow links to DS3's and higher. Home users moved from slow dial-up connectivity to broadband. Bandwidth had become cheap and abundant. Yet, despite this abundance of bandwidth the feasibility of most sequence attacks remain, for all practical purposes, infeasible.

One particular sequence attack, however, will benefit from the current abundance of bandwidth and become a potentially dangerous attack vector; the TCP Reset attack. A TCP Reset attack is a denial of service attack in which the perpetrator attempts to prematurely terminate a victim's active TCP session. This does not present a serious risk for many connections, but could create significant damage and or disruption if used effectively. The disruption of a Web browser's TCP connection presents little risk or impact in most cases. However, the termination of BGP peering links or other crucial communications could be immensely disruptive and costly.

Separating Fact from FUD

The TCP Reset attack and its risks are greatly misunderstood by many researchers. A presentation at the 2003 Las Vegas Blackhat briefings, the premier Information Security Convention, illustrates this lack of understanding. In their presentation "BGP Vulnerability Testing: Separating Fact from FUD" (Convery & Franz, 2003), Sean Convery and Matthew Franz of Cisco CIAG presented some excellent research on a variety of attacks on the BGP protocol. One of the 10 attack vectors discussed focused on the feasibility of TCP Resets and Sequence number guessing. While the vast

majority of their presentation was superb, their presentation contained calculations and statistics regarding TCP Reset attacks that missed critical factors and resulted in highly erroneous statements of fact. This happens to almost all researchers at some point or another, and they are typically corrected and revised when reviewed by peers. Yet when presented to an overflowing auditorium of Information security and Internet researchers, there was no dispute or questioning of the facts provided. The obvious conclusion is that the security community does not understand the fundamentals of TCP Reset attacks.

The following slide from their presentation most obviously demonstrates the presentations misstatements regarding TCP Reset attacks. Convery and Franz used the slide in Figure 1 below to summarize their estimations of requirements for a successful TCP Reset attack.

TCP Resets Time Requirements

- A theoretical blind attack @ 1 million pps ~ 30 minutes just to guess the sequence number (assuming a correct guess after iterating through 50% of the space).

$(2^{32}/2)/1{,}000{,}000$ = # of seconds

- Our tool was able to generate 62,500pps* ~ 9 hours
- Since the attacker won't know which side is 179 vs. a high port multiply these numbers by 2.
- With source port randomization, this goes to 4 years in the first example (1 mil. pps to guess 1 48 bit number and 142 years assuming 62,500pps and needing to guess both sides):

((248/2)/62,500)x2 = # of seconds

*What sort of event is 62.5kpps on your router?

Figure 1: Convery & Franz Presentation Slide

The first bullet point in the slide appears to be based on the following mathematical assumption:

((Potential seq. numbers/packets per second)/seconds)/2 (half the numerical space) = Time required

Which translates to the following equation:

((4,294,967,295 / 1,000,000) / 60) / 2 = 35.79 minutes

The time estimation of 30 minutes that is stated in the first bullet point is then used in a following point to assert than an attack could realistically take up to 142 years. This estimation is accurate numerically, but only if one assumes that the attacker must guess every possible sequence number in order to generate an acceptable Reset packet.

TCP Window Considerations

The time requirement is grossly overstated. This is due to the fact the TCP Window sizing was not considered in the calculations. The window size is a critical component in the risks presented by TCP Reset attacks. This is due to the RFC-793 (USC, 1981) requirement that a TCP session should consider any TCP packet with a sequence number within the range of the expected sequence number to the expected sequence number plus window size to be accepted as valid (see figure 1). The authors of RFC-793 intended for the possibility that packets could arrive in a non-sequenced order, such that packet 3 arrives before packet 1 and 2.

Additionally, RFC-793 states that in the event a Reset flag is received, the receiver should immediately terminate the connection (see figure 3). This is of particular interest in the case for TCP Reset attacks, since even an out of order packet containing a Reset flag is considered valid and acted upon immediately without waiting for the previously sequenced packets to arrive. This provides

the Reset attacker with an exceptionally expanded opportunity over other types of TCP data injection attacks.

```
receive window


This represents the sequence numbers the local (receiving) TCP
is willing to receive.  Thus, the local TCP considers that
segments overlapping the range RCV.NXT to RCV.NXT + RCV.WND - 1
carry acceptable data or control.

Segments containing sequence numbers entirely outside of this
range are considered duplicates and discarded.
```

Figure 2: RFC-793 Receive Window

```
RST

A control bit (reset), occupying no sequence space, indicating
that the receiver should delete the connection without further
interaction.  The receiver can determine, based on the sequence
number and acknowledgment fields of the incoming segment,
whether it should honor the reset command or ignore it.  In no
case does receipt of a segment containing RST give rise to a
RST in response.
```

Figure 3: RFC-793 RST Flag

Therefore, the time estimation offered by Convery and Franz is dramatically overstated. The size of the overstatement is variable, depending on the particular window sizes selected by the given TCP endpoints.

The TCP header allocates two bytes, or 16 bits, for the window size. This allows for window sizes to be adjustable up to 65,535. A window provides a way for a TCP talker to notify the remote end of the amount of octets (bytes) it

is capable of accepting. For example, if a operating system maintains a receive buffer for incoming TCP data of 32,768 bytes, it will notify the remote system of this by specifying a window of 32,768 during the initial 3-way handshake which establishes the TCP session. As each packet of data is received, the receiver will acknowledge the receipt of the data and indicate if the amount of free buffer space has changed by either increasing or decreasing the window size. This is known as a sliding window. If the application on the receiving host fails to process the data in the receiving buffer, it may eventually fill to capacity. If the receive buffer becomes full, the receiver can notify the sender by setting a receive window size of zero (known as closing the window), effectively asking the sender to not send any more data until further notice.

Typically, the TCP window is at least several kilobytes, which allows many packets to be transmitted before the sender must wait for an acknowledgement reply. Typically the receiver acknowledges the data and the application continues to read the data so the window stays wide open. Without this windowing capability TCP would at best be painfully slow and at worst unusable over links slower than a couple of megabits per second.

Since the window size presents an expanded opportunity for the malicious hacker to inject bogus Reset control packets, the most obvious solution is to reduce the window size to 1 octet, thereby ensuring the an exact sequence number must be guessed. However, this would introduce severe inefficiencies in the transmission of data, as two entire TCP/IP packets would have to be sent for the receipt of only a single byte of actual data. If efficiency were the only consideration, a window size of 65,535 would allow for the largest transmission of data using the traditional RFC-793 specification.

The TCP Reset Attack

The basis of the TCP Reset attack is the utilization of the TCP window size to reduce the number of sequence numbers that must be guessed. In a traditional sequence number attack, the exact sequence number must be known in order for the spoofed packet to be considered valid and accepted by the receiving TCP endpoint. The TCP Reset attack is made possible due to the requirements that a TCP endpoint must accept out of order packets that are within the range of a window size, and the fact that Reset flags should be processed immediately.

This reduces the number of sequence number guesses the attack must make by a factor equivalent to the active window size. Each sequence number guess made by the attacker can be simply incremented by the receiving connections window size. In a traditional TCP packet with a Window size of 65,535, this means that instead of having to spoof 4,294,967,295 packets with different sequence numbers, the attacker is only required to send 65,535 Reset packets in order to successfully reset the active TCP connection. Worse yet, with RFC-1323 Extensions, the attack could be even easier.

Window Scaling

TCP Window scaling was added to the TCP specification in RFC-1323 "TCP Extensions for High Performance" (Jacobson, 1992). This option defines an implicit scale factor, which is used to multiply the window size value found in a TCP header to obtain the true window size. It allows for windows up to 1,073,741,823 bytes in size by adding a TCP scale factor that effectively expands the 16-bit window to 32 bits. For practical purposes the scaled windows are limited to 30 effective bits, or a scale factor of 14. This is primarily due to numeric wrap-around issues.

Window scaling was added to allow for rapid transmission of data on long fat networks (LFN). These are typically connections with very high bandwidth, but also high latency. Networks with satellite connections are one example of a LFN, since satellite links always have high propagation delays but typically have high bandwidth.

The risks of TCP Reset attacks are greatly increased when window scaling is utilized. This is true even if only one end of the TCP connection utilized window scaling. If both sides of the link are not configured to support window scaling, then the default of 65,535 bytes will typically apply as the maximum window size. On a Cisco-to-Cisco BGP connection, this would result in the typical window size of 16,384 being expanded to 65,536. This would decrease the number of packets required for a successful TCP reset attack by a factor of nearly 4. Moreover, if the scale factor and window size were configured for maximum performance, only 4 sequence number guesses would be required for 100% effectiveness in a TCP Reset attack.

OS Window Size Observations

As previously noted, one key component of a Reset attack is window size. Since various vendors may choose to implement varying methods of determining window size, some baseline information was gathered by analyzing the Initial Window size of various TCP connections from assorted equipment on-hand in the lab. The TCP sessions were generated using the BGP protocol on Cisco devices, and Telnet protocol on other systems. The packets required for a successful Reset are based on the equation**: (2^32 / Initial Window Size)**

Operating System	Initial Window Size	Packets Required
Windows 2000 5.00.2195 SP4	**64512**	**66,576**
Windows XP Home Edition SP1	**64240**	**66,858**
HP-UX 11	**32768**	**131,071**
Nokia IPSO 3.6-FCS6	**16384**	**262,143**
Cisco 12.2(8)	**16384**	**262,143**
Cisco 12.1(5)	**16384**	**262,143**
Cisco 12.0(7)	**16384**	**262,143**
Cisco 12.0(8)	**16384**	**262,143**
Windows 2000 5.00.2195 SP1	**16384**	**262,143**
Windows 2000 5.00.2195 SP3	**16384**	**262,143**
Linux 2.4.18	**5840**	**735,439**

Operating System	Initial Window Size	Packets Required
Efficient Networks 5861 (DSL) v5.3.20	**4096**	**1,048,575**

Figure 4: Initial Window Size Selection

As noted earlier, a larger window will provide greater transmission efficiency but also expand the opportunity for spoofed TCP Reset attacks. It is interesting to note that all four versions of Cisco IOS (ranging from 12.0 through 12.2) tested had exactly the same initial window size. Originally, the Cisco TCP window sizes were obtained using the telnet protocol, which had a window size of 4192. At the suggestion of Matthew Franz of Cisco CIAG, the Cisco devices were re-tested using BGP as the TCP protocol and the much larger 16,384-byte window size shown above was obtained. Due to the dangers inherent in TCP Resets against BGP and other routing protocols, this was especially noteworthy.

Another observation worthy of mentioning, is that the various Microsoft Windows operating systems appears to have increased the Window size in the most recent versions and patch levels. This was most likely intended to improve performance, but also increases the ease with which TCP Reset attacks against those systems can be accomplished.

Realistic Expectations and Equations

Assuming that most Cisco IOS versions initially select a window size of 16,384, and have perfectly random initial sequence number selection (discussed in more detail later), we can greatly reduced the time and packets required for the TCP Reset attack by an equivalent factor. As such, we can revise Convery and Franz's equation to the following, in order to more

accurately demonstrates that time and packets required to execute a TCP Reset attack.

(((4,294,967,295 / 1,000,000) / 16,384) / 60) / 2 = .00218 (approximately 1/10 second)

Convery and Franz's equation estimated a successful attack (50% success rate) would require 35 minutes at 1 million packets per second. With the revised equation, which includes TCP Window size, the time required is reduced to less than ½ second. The difference is both staggering and dramatic. If we consider that in practical use, Convery and Franz indicated they could only generate 62,500 packets per second as opposed to 1 million, we can determine a realistic assumption as follows:

(((4,294,967,295 / 62,500) / 16,384) / 60) / 2 = .0291 (approximately 1.7 seconds)

Convery and Franz attempt to assure us that we could detect this attack, since an event of 62,500 packets per second over 9 hours would clearly be noticed. However, since the actual attack could be completed at the rate of 62,500 packets per second in as little as 1.7 seconds (50% success rate), the potential for administrative observation is extremely low.

Initial Sequence Number Prediction

Initial sequence number selection is of significant interest to the attacker, since the majority of the attack efforts discussed up to this point have been based on the assumption that initial sequence number selection is perfectly random. This is never the case in practical use however, since computers are notoriously bad at generating random numbers. The generation of initial

sequence numbers was the subject of a paper from Michal Zalewski of Bindview entitled "Strange Attractors and TCP/IP Sequence Number Analysis – One Year Later (Zalewski, 2003)." While Cisco's IOS code has historically had problems with ISN generation, according to Zalewski this has improved dramatically in recent versions. This makes sequence prediction less of a factor now than it has historically been for Cisco routers. In fact, Sean Convery of Cisco has noted that any advantages of sequence number prediction are practically irrelevant, given the ease with which TCP Reset attacks can be performed using the methods outline in this paper.

What is important to note regarding ISN prediction, is the dramatic risk presented to the numerous operating systems (Win95, 98, NT, 2000, AIX, HP/UX, IRIX, MacOS, and others) that have severe inadequacies. A TCP Reset attack against the systems mentioned above is dramatically easier due to ISN prediction, as well as much larger TCP window selection. These two factors can result in TCP Reset attacks that are trivial to perform, since the attacker can easily predict sequence number selection as well as providing an expanded window size to allow for reduced guess attempts within the predicted sequence range.

Steven Bellovin introduced RFC-1948 (Bellovin, 1996) in an effort to provide a very intelligent a logical method for defeating most sequence number prediction attacks. Unfortunately, it appears the suggestions in RFC-1948 have not been widely adopted.

TCP Source Port Considerations

A successful TCP Reset attack requires the attacker to either have knowledge of a valid 4-tuple, or to be able to make an accurate guess. The 4-tuple consists of source IP address, TCP source port, destination IP address, and destination TCP port. The source and destination IP addresses are obvious as they are the TCP speakers whose conversation the attacker will be spoofing. The destination port is also obvious, since a TCP RESET attack will be against a particular TCP protocol, such as BGP where the attacker could assume the destination port of 179.

The only difficult part of this 4-tuple is the TCP source port, since it varies with each new TCP session. To the casual observer, it would appear that the additional requirement of a correct source port would increase the difficulty of the attack by a factor or 16 (65,536 possible combinations of 16 bits). Even Convery and Franz suggest that Pseudo-random source ports would increase the difficulty of an attack by increasing the numerical attack space "from 2^{32} to 2^{48}". Unfortunately, pseudo-random source port usage is not used today, and quite likely will not be used due to the additional overhead and difficulties that can be introduced.

Moreover, ephemeral source ports are not actually selected from the full 16-bit range, but a smaller subset range. Ports 1-1024 are reserved for privileged processes, and ports 49152-65535 are reserved for private ports. In some cases, operating systems have even reserved ports 5001-65535 for private usage, leaving only 3977 ports for dynamic kernel assignment.

Currently, existing source port selection is so predictable as to make guessing reasonably trivial; even for the blind TCP spoofing attacker. The following chart represents observations of source port selection from various Operating Systems.

OPERATING SYSTEM	OBSERVED INITIAL SOURCE PORT	OBSERVED NEXT SOURCE PORT SELECTION METHOD
Cisco 12.2(8)	**11000**	**Increment by 1**
Cisco 12.1(5)	**48642**	**Increment by 512**
Cisco 12.0(7)	**23106**	**Increment by 512**
Cisco 12.0(8)	**11778**	**Increment by 512**
Windows 2000 5.00.2195 SP4	**1038 / 1060**	**Increment by 1**
Windows 2000 5.00.2195 SP3	**1060**	**Increment by 1**
Windows XP Home Edition SP1	**1050**	**Increment by 1**
Linux 2.4.18	**32770**	**Increment by 1**
Nokia IPSO 3.6-FCS6	**1038**	**Increment by 1**

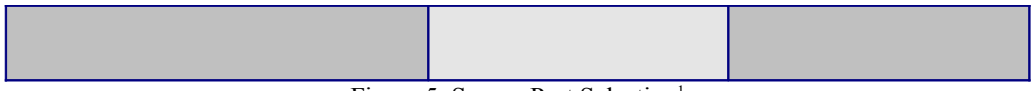

Figure 5: Source Port Selection[1]

Windows based operating systems appear extremely predictable, almost always starting new TCP source port selections just after the reserved ports and incrementing by one with each new connection. Cisco devices, however, appear more dispersed. This is not as problematic as it first seems. Since particular versions of IOS appear to have distinct preferences, knowledge of the target device IOS would be helpful to the attacker, but not required. Moreover, this distinctive behavior could potentially help an attacker by allowing IOS fingerprinting if the attacker could coax an outbound connection from the device within a reasonable time after a reboot. A reasonable assumption can be made on the number of connections that the device may be presumed to initiate over a given time. This assumption should be based on the number of BGP neighbors (if known, assumed otherwise), and any other roles of functions that may be performed on the attacked device.

If the router IOS version is unknown (presumed), then one simple method for guessing source port selection could be to estimate the number of BGP neighbors for the device. Half of this number of BGP neighbors can be assumed to be initiators, and the other half as receivers, since BGP allows for either end to initiate communication. A range of estimated source ports can be generated by increasing the initial source port by 512 (or 1 depending on the IOS version) for each assumed initiated connection and then estimating other

[1] It should be noted that the source port data was obtained through observational analysis only, and not intended to be definitive by any means.

variables, such as number of times the TCP connection may have been broken and re-initialized since the last reboot. Finally, if the IOS version is unknown, add the ranges of source port numbers to all the known source port starting numbers for the various IOS versions that are assumed may be running on the attacked device. The size of the "guess range" will vary, depending on the desired level of success, but will most likely be less than 50 in most cases. This is greatly encouraging since the assumed range of guessed source ports is more than 48,000 (reserving the 1,024 low ports, and high ports from 49,152 - 65,535.)

As is always true in hacking and network attacks, the more you know about your target, the more likely you are to succeed. This fundamental rule is equally true here. In an ideal situation, an attacker would be able to deduce the IOS version, number of BGP peers, network management method used to monitor the router, and possibly the last reboot time of the router. Each of these factors can be determined or accurately estimated using existing techniques and methods.

Practical Attack Requirements Assuming Ideal ISN

Now that the theoretical aspects of a TCP Reset attack have been addressed, it is also important to consider the practical. While the 1,000,000 packets per second attack suggested by Convery and Franz is ideal for theoretical assumptions, it does not translate to a world where attackers may have limited bandwidth resources. Many attackers, however, are quite capable of launching distributed and coordinated attacks from more than one compromised system, achieving the capability to transmit packets at a much higher rate than a typical dial-up or broadband Internet users.

The following charts were prepared which reasonably reflect the potential for success, based on a variety of connectivity speeds. All time requirement results are based pure sequence number guessing only (no sequence number prediction attempts). Packets are calculated as 20 bytes for the IP header, plus 20 bytes for the TCP header, or 40 bytes (320 bits) total. Data link is not considered in packet size and will vary based on differing types of physical links. Window scaling options are not used.

In the first chart (Figure 6), a window size of 65,535 is used, as this is the best-case scenario for traditional TCP packets (without RFC-1323 Extensions). The following calculation is used:

(((4,294,967,295 / 65,535) * 320) / bandwidth in mpbs) = seconds required

Internet Connectivity	Packets Required	Time Requirement: known source port	Time Requirement: guessing 50 source ports
56kbps (dialup)	**65,537 (*50)**	**374 seconds (6 min.)**	**18,700 (5.2 hours)**
80kbps (DSL)	**65,537 (*50)**	**262 seconds (4.3 min.)**	**13,107 (3.6 hours)**
256kbps (DSL)	**65,537 (*50)**	**81 seconds (1 min.)**	**4,050 (1.1 hours)**

Internet Connectivity	Packets Required	Time Requirement: known source port	Time Requirement: guessing 50 source ports
1.54kbps (T1)	**65,537 (*50)**	**13.6 seconds**	**680 (11 minutes)**
45mbps (DS3)	**65,537 (*50)**	**1/2 second**	**25 seconds**
155mbps (OC3)	**65,537 (*50)**	**1/10 second**	**5 seconds**

Figure 6: Bandwidth and Time Requirements for a 65k Window

In the second chart (Figure 7), a window size of 16,384 is used, as this is the window size typically used on Cisco routers for traditional TCP packets (without RFC-1323 Extensions). The following calculation is used:

(((4,294,967,295 / 16,384) * 320) / bandwidth in mpbs) = seconds required

Internet Connectivity	Packets Required	Time Requirement	50 Source Ports Time Requirement
56kbps (dialup)	**262,143 (*50)**	**1497 seconds (90 min)**	**74,850 (20.8 hours)**
256kbps (DSL)	**262,143 (*50)**	**327 seconds (20 min)**	**16,350 (4.5 hours)**
1.54kbps (T1)	**262,143 (*50)**	**54 seconds**	**2,700 (45 minutes)**
45mbps (DS3)	**262,143 (*50)**	**1.8 seconds**	**90 seconds**
155mbps (OC3)	**262,143 (*50)**	**.5 seconds**	**25 seconds**

Figure 7: Bandwidth and Time Requirements for a 16,384 Byte Window

Most attackers would not have access to Internet links at speeds such as an OC3. However, it should be pointed out that although an attacker may not have a single dedicated link at the speeds suggested above, it is reasonable to consider that an attacker may have multiple links available with an aggregate

bandwidth equal to, or higher, than those listed above. It would be trivial for an attacker to compartmentalize a TCP Reset attack into small pieces and initiate it as a distributed attack.

Observed Results of Blind TCP RESET Attacks

A series of blind TCP Reset attacks were performed and the relevant results were recorded in the following chart. The attacks were conducted using tools developed specifically to test the practical application of TCP Reset attacks using the methods outlined above. This tool will be made available shortly.

The testing environment consisted of the hardware and software shown in Figure 8. All systems were connected to the Cisco 2950 switch at 100 full duplex. The attacker system was not permitted to sniff or view the sequence numbers used in the TCP connection between the Initiating and Target hosts. The TCP Source port was provided to the attacking system in order to obtain accurate results for a single port attack. Results can be multiplied to obtain attack results requiring multiple source port guesses. The attacker used brute force sequence number guessing, starting at 0 and working up through the entire 32-bit sequence number space.

SYSTEM	HARDWARE	SOFTWARE
Initiating Host	Pentium 4 1.70ghz	Windows XP SP1
Target Host A	Intel Celeron 2.2ghz	Windows 2000 SP4
Target Host B	Cisco 2621 router	12.0(7) T
Attacker System	AMD Athlon XP 2700+	Linux 2.4.18

SYSTEM	HARDWARE	SOFTWARE
Network Fabric	Cisco Catalyst 2950	12.1(13) EA1

Figure 8: Hardware and Software

TCP Reset Testing Software Used

The spoofed TCP Reset packets were generated with a program called `reset-tcp.c` and written specifically for these tests. It utilizes the libnet-1.1.1 library (Schiffman, 2003). This program will be made available on the TERRORIST.NET (http://www.terrorist.net) website. The program code is not ISO-31337 complaint (not an official ISO, but a term I coined for 'script-kiddies usability') so please read the code and comments before compiling and running.

A modified version of the Cisco TCP Test Tool (`ttt-1.3.1`) used in Convery and Franz's research was also created, which allows incrementing the sequence number by the specified window size for each subsequent packet. The Cisco program was significantly faster and more efficient at writing spoofed packets to the wire, achieving rates greater than 60,000 packets per second. The code changes have been submitted to Cisco for possible inclusion in their next release (should they desire to add my changes.) In any case, the modified version of `ttt-1.3.1` will also be made available on the TERRORIST.NET (http://www.terrorist.net) website.

Test Results for 250 Packets Per Second (Upstream DSL Speed)

For the first set of tests, spoofed packets were generated at a rate of approximately 250 packets per second. At 40 bytes per packet, this approximates 80,000 bits per second, or approximately the upstream bandwidth available to a low-end home DSL connection. The time required for covering the entire 32-bit sequence number range using a 65,535-byte window size was approximately 4 minutes 18 seconds. The results are displayed in Figure 9.

Target: Windows 2000	Start Time	Time of Reset	Time Required (seconds)
Test 1	164830	165231	401
Test 2	164835	165241	406
Test 3	165503	165920	417
Test 4	170041	170044	3
Test 5	171639	171654	15
Test 6	222221	222227	6
Test 7	222347	222431	84
Test 8	224751	224820	69
Test 9	231325	231441	116
Test 10	161545	161759	214
Anticipated window size	63000		
Packets per second	250		

Figure 9: Attack Test at 250 pps

Test Results for 4,370 Packets Per Second (T-1 Speed)

For the second set of tests, spoofed packets were generated at a rate of approximately 4,370 packets per second. At 40 bytes per packet, this approximates 1,398,400 bits per second, or approximately the bandwidth of a typical T-1 connection. The time required for covering the entire 32-bit sequence number range using a 65,535-byte window size was 15 seconds. The results are displayed in Figure 10.

Target: Windows 2000	Start Time	Time of Reset	Time Required (seconds)
Test 1	180038	180046	8
Test 2	180724	180728	4
Test 3	180812	180822	10
Test 4	180910	180920	10
Test 5	181210	181221	11
Test 6	205600	205607	7
Test 7	181408	181418	10
Test 8	181539	181550	11
Test 9	202329	202332	3
Test 10	202410	202414	4
Anticipated window size	63000		
Packets per second	4370		

Figure 10: Attack Test at 4,370 pps

As can be seen in the date above, the speed of a successful attack was based entirely on the value of the sequence number used in the TCP connection, since the testing program guessed sequence numbers in a purely linear fashion. Connections with low sequence numbers were quickly reset while connections with higher sequence numbers took longer. Overall, the results of each test set appear to confirm the estimations made earlier (see Figure 6).

Protecting From Attack

The TCP Reset attack is clearly a potential threat. The risk of the threat varies greatly depending on the target and the nature of the TCP session being attacked.

The following precautions are recommended.

RFC-2827 Filtering:

The most obvious defense against a TCP Reset attack is implementation of strong ingress/egress filters on border routers. This will only protect from attacks originating outside of the target network, but is the best defense available. RFC-2827 (Ferguson & Senie, 2000) provides best practices for configuring border filters to block spoofing attacks originating using ingress traffic filtering to prohibit DoS attacks which use forged IP addresses to be propagated from 'behind' an Internet Service Provider's (ISP) aggregation point.

RFC-2385 BGP TCP MD5 Signatures:

The BGP protocol is a likely target for TCP Reset attacks, and the impact could be dramatic. Fortunately, RFC-2385 (Heffernan, 1998) proposed the protection of BGP sessions through the use of TCP MD5 signatures in the TCP header options. This RFC was proposed specifically to guard against TCP Reset attacks. The TCP Header options include an MD5 signature in every packet and are checked prior to the acceptance and processing of any TCP packet, including Reset flags.

TCP Window Size Tuning:

Another option to reduced expose to TCP Reset attacks is careful tuning of TCP window size. Since window size has such a dramatic effect on the vulnerability of a system to a TCP Reset attack, careful tuning should be considered. Information on how to tune basic TCP windows size is provided in Appendix A, although it provides no guarantees that any given application will respect tuning of the default window size.

For example, while testing Cisco IOS windows sizes, the telnet connection utilized a window of 4,192 while BGP used a window size of 16,384. When the command "*`ip tcp window-size 100`*" was issued in IOS, telnet used the newly configured default window size of 100, while BGP continued to utilize a window size of 16,384.

Appendix A: Adjusting Default TCP Window Size

The configurations below will only set the "default" TCP Window size. It is important to be aware that applications may choose to specify their TCP window size when creating the socket. If an application explicitly requests a window size, the OS will most likely honor it.

Windows 2000:

Tuning of Window size can be accomplished by adjusting registry settings. The registry keys of interest can be found in the registry at this location:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters

The names and data types are as follows;

GlobalMaxTcpWindowSize	REG_DWORD
TcpWindowSize	REG_DWORD

An excellent document titled "Microsoft Windows 2000 TCP/IP Implementation Details" (MacDonald and Barkley, 2000) provides detailed information on tuning Windows 2000 window sizes, as well as other TCP/IP settings.

Linux (2.4.x kernels):

The following two variables can be added to the `/etc/sysctl.conf` file. The names "`rmem`" and "`wmem`" correspond to receive and transmit respectively. After settings these values, execute the "`sysclt -p`" command to have them take effect.

```
net.core.rmem_default = [0-65535]

net.core.wmem_default = [0-65535]
```

Linux (2.2.x kernels):

The default transmit and receive buffers can be viewed or modified through the /proc filesystem.

```
echo [0-65536] > /proc/sys/net/core/rmem_default

echo [0-65536] > /proc/sys/net/core/wmem_default
```

Cisco IOS:

To adjust the default window size in Cisco IOS, enter enable mode and enter the following command in configuration mode:

```
router(config)#ip tcp window-size [0-65535]
```

Solaris:

Adjusting the default TCP window size in Solaris can be accomplished using the ndd command.

```
# ndd -set /dev/tcp tcp_xmit_hiwat [0-65535]

# ndd -set /dev/tcp tcp_recv_hiwat [0-65535]
```

Table of Figures

Figure 1: Convery & Franz Presentation Slide.....6
Figure 3: RFC-793 RST Flag.....8
Figure 4: Initial Window Size Selection.....13
Figure 5: Source Port Selection.....18
Figure 6: Bandwidth and Time Requirements for a 65k Window.....21
Figure 7: Bandwidth and Time Requirements for a 16,384 Byte Window.....22
Figure 8: Hardware and Software.....24
Figure 9: Attack Test at 250 pps.....25
Figure 10: Attack Test at 4,370 pps.....26

Bibliography

1. Convery, Sean and Franz, Matthew; "BGP Vulnerability Testing: Separating Fact from FUD", 2003, http://www.nanog.org/mtg-0306/pdf/franz.pdf

2. Schiffman, Mike D.; Libnet packet library, 2003, http://www.packetfactory.net/libnet

3. CA-01-09 CERT; The sequence number CERT advisory CA-01-09, 2001, http://www.cert.org/advisories/CA-2001-09.html

4. Zalewski, Michal; Strange Attractors and TCP/IP Sequence Number Analysis - One Year Later, 2002, http://lcamtuf.coredump.cx/newtcp/

5. Bellovin, Steve; RFC 1948 "Defending against Sequence number attacks", 1996, http://www.faqs.org/rfcs/rfc1948.html

6. Jacobson V.; RFC 1323 TCP Extension for High Performance, 1992, http://www.faqs.org/rfcs/rfc1323.html

7. USC, University of Southern California; RFC 793 "Transmission Control Protocol", 1981, http://www.faqs.org/rfcs/rfc793.html

8. PIX TCP Reset Advisory; 2000-07-11, http://www.cisco.com/warp/public/707/pixtcpreset-pub.shtml

9. Shimomura, Tsutomu; "Technical details of the attack described by Markoff in NYT", 25 Jan1995, http://www.networkcomputing.com/unixworld/security/001.add.html

10. Watson, Paul A. aka shade@corcom.com; alt.2600 Usenet Reply Post, January 1995, http://groups.google.com/groups?as_umsgid=26OOhertz@shade.com&output=gplain

11. CA-1995-01, CERT; IP Spoofing Attacks and Hijacked Terminal Connections, 23 Jan 1995, http://www.cert.org/advisories/CA-1995-01.html

12. MacDonald, Dave and Barkley Warren; Microsoft Windows 2000 TCP/IP Implementation Details, 2000, http://www.microsoft.com/windows2000/techinfo/howitworks/communications/networkbasics/tcpip_implement.asp

13. Ferguson P. & Senie D., RFC-2827 "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", 2000, http://www.faqs.org/rfcs/rfc2827.html

14. Heffernan, A., RFC-2385 "Protection of BGP Sessions via the TCP MD5 Signature Option", 1998, http://www.faqs.org/rfcs/rfc2385.html