GUARDENT℠

The Problem with Random Increments

WHITE PAPER

Timothy N. Newsham
February 2001

GUARDENT℠

Terms and Conditions of Use

No Warranties/Limitation of Liability

Guardent makes no representations with respect to the information provided in this research paper, and such information is being provided "AS IS." The information in this advisory may change without notice. This research paper may also contain links to other websites. These links are provided for convenience only, and Guardent is not responsible for the information on such websites. If you choose to utilize such links, you do so at your own risk.

Guardent hereby disclaims all warranties, including the warranties with respect to this information, including any implied warranties of merchantability, fitness for a particular purpose and non-infringement. In no event will Guardent be liable for any damages whatsoever arising out of or in connection with the use or distribution of this information, including any special, consequential or other damages.

Introduction

Predictability in the Initial Sequence Numbers (ISN) used in a TCP/IP implementation can open up security holes. These numbers are used during the initial connection establishment of TCP: The initiator selects an ISN and sends it in a SYN packet to the responder. The responder selects its own ISN, and sends back a SYN packet to the initiator using its ISN and acknowledging the sequence number of the initiator. Finally the initiator responds by acknowledging the sequence number sent by the responder [Post]. It was pointed out in [Morr] and [Bell2] that an attacker can blindly forge a connection using any source address if the ISN value of the responder can be guessed. In order for such an attack to succeed, the exact value of the responder's ISN must be guessed.

To combat this flaw, many TCP/IP implementers have adjusted their designs to generate ISN values that are harder to guess. One method of achieving this is to increment the value used as an ISN by a random amount for every new connection (and also at periodic intervals). Examples of systems using this method are FreeBSD [Free] and OpenBSD [Open].

In this paper, we will analyze the properties of randomly incrementing ISN values and discuss some security weaknesses that remain in the TCP/IP stack when this method of ISN generation is used.

Properties of Random Increments

Random increments make it difficult to guess the next value a TCP implementation will choose for its next ISN, but they don't destroy all knowledge of what the next ISN will be. Clearly, after a single random increment, the range of the next ISN is still known. Surprisingly, even after many random increments, much can still be said about the range and distribution of ISN values.

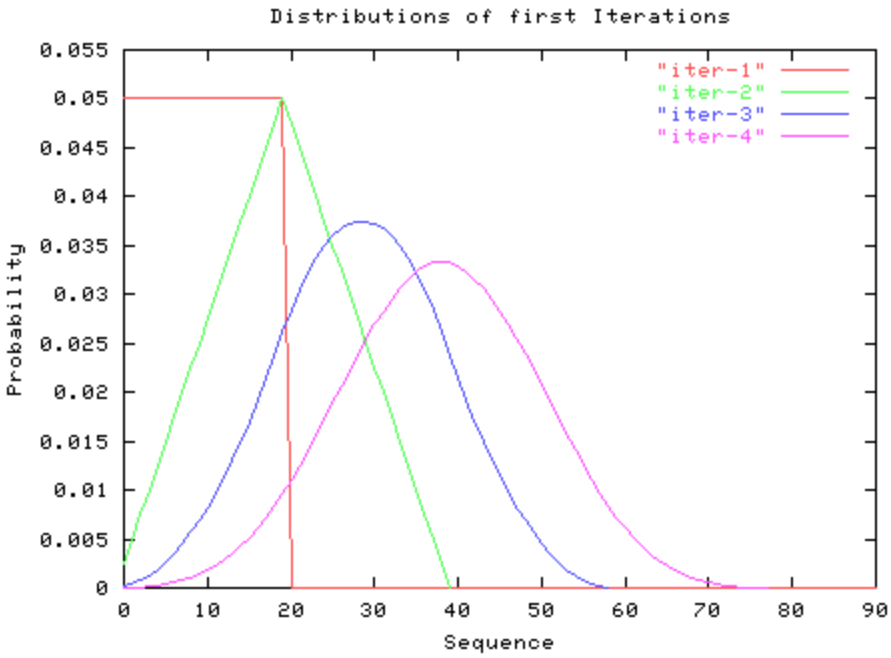
To illustrate this more clearly, let us assume that a random increment is applied with an even distribution of numbers from zero to nineteen (20 values). We will take, as the starting value, the sequence number of zero. Initially the distribution of possible ISN values is the single value zero, which has a probability of 100% (or 1.0) with all other values having a probability of 0%. After one random increment, the distribution of probable ISN values is evenly spread from zero to nineteen, each value having a probability of 5% (or 1/20).

The distribution of values has spread out and lowered. Mathematically, what happened is that the initial distribution underwent a convolution with the random distribution used by the increment:

$$f'[n] = \sum_i f[n]r[i-n]$$

where *f* is the distribution before the increment, *r* is the distribution that the increment is drawn from, and *f'* is the distribution after the increment.

When further random increments occur, this convolution is again repeated. After four increments, the distribution quickly degenerates into a Gaussian distribution. This can be seen graphically in Figure 1. This is not a property of the distribution we started with, but is a general principle of probability theory known as the Central Limit Theorem [Hoel]. Had a random increment with a different distribution been used, we would eventually see the same result after several random increments.



As random increments are applied to the ISN value, the range of possible values spread, and the distribution becomes Gaussian.

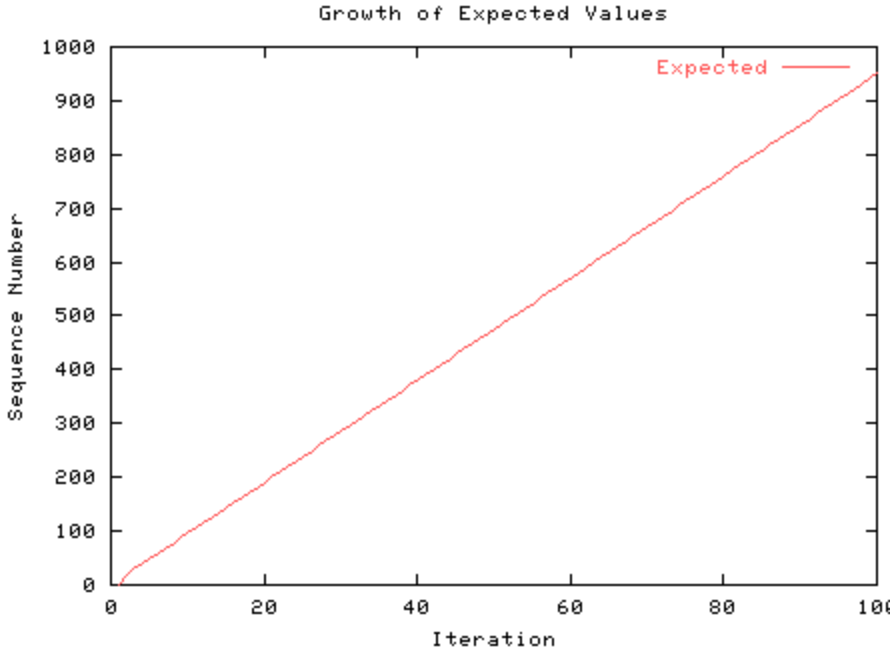
As we continue to apply random increments, the Gaussian distribution continues to spread and flatten. In the following two plots, we show how the expected value and the standard deviation of the distributions grow with continued random increments. We observe that the expected value grows linearly.

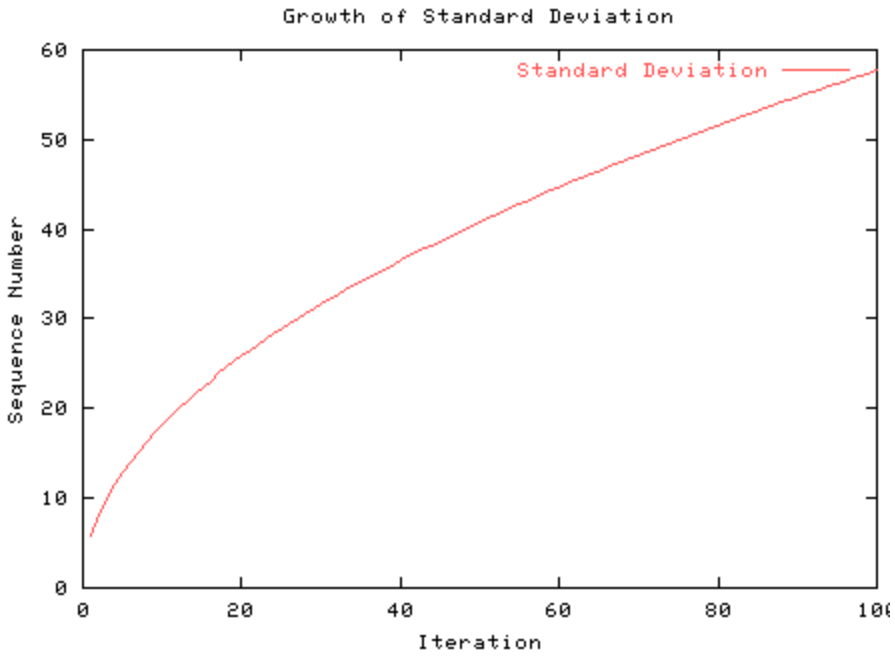
The growth of the standard deviation is proportional to x ^ 1/2. Functions that approximate these values are:

$$m[n] = m_1 n$$

$$s[n] = s_1 \sqrt{n}$$

where *m1* and *s1* are the expected value and standard deviation of the random increment, respectively. For a mathematical derivation of these equations, refer to the appendix. Formulas for computing *m1* and *s1* for uniform distributions are also derived.





Computing ISN Ranges

To compute ISN values in practice, one must know the details of the ISN generation. In this section we will discuss the specifics of the OpenBSD [Open].

The OpenBSD system uses random increments to update its Initial Sequence Number sent by the TCP/IP stack. There are three events that can cause the ISN number to be updated. The first is the reception of an incoming connection request. The second is initiation of a new outgoing connection. The last is a timer event that fires periodically. These are listed in the following table.

Event	Location
Connection Request	tcp_input.c
Connection Initiation	tcp_usrreq.c
Timer (2 Hz)	tcp_timer.c

Each of these updates causes a random increment of size TCP_ISSINCR, or 125 * 1024, plus a definite increment of one. The value of the increment is drawn from a uniform distribution. The timer increment is implemented in such a way that the total increment per second will have a size of 2 * TCP_ISSINCR, even if the timer frequency is changed.

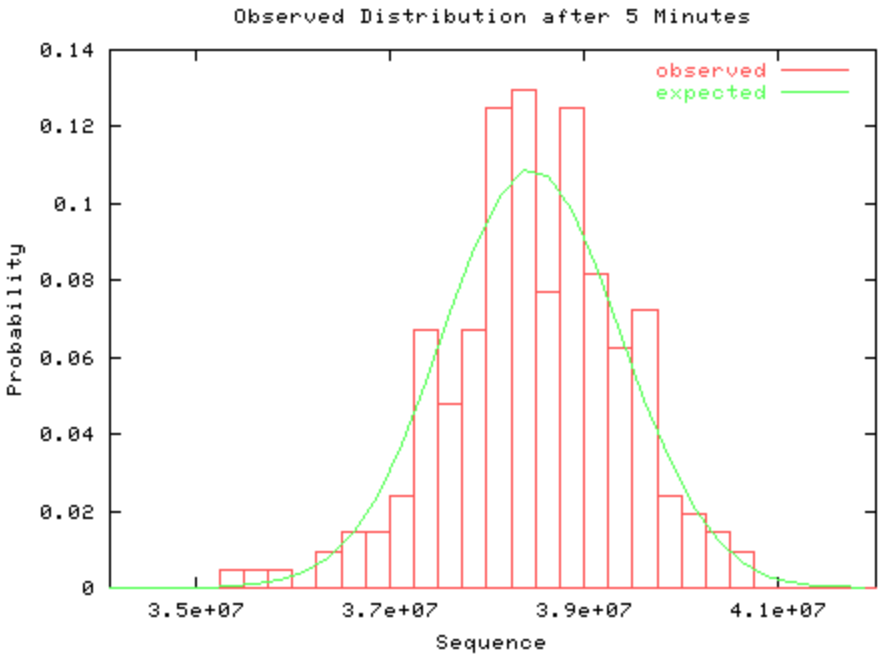
We can easily derive equations for the mean and standard deviation of the distribution as a function of the number of increments applied by using the properties derived in the previous section. First, we must calculate the properties of the random increment. Using the formulas derived in the appendix, we derive the expected value of each increment as 64000 and the standard deviation as 36950. Note: to compensate for the definite increment of one in each increment, the expected value is one greater than the value given by the equation in the appendix. When these numbers are applied to the approximations derived in the appendix we get:

$$m[n] = 64000n$$

$$s[n] = 36950\sqrt{n}$$

where *n* is the number of increment operations that have occurred.

These approximations agree quite well with observed behavior. We recorded 209 ISN values generated by an OpenBSD 2.8 machine at five-minute intervals. During this period no connections were made to or from the machine, other than the connection used to probe the ISN. There should have been 601 random increments performed between each probe. We calculated the size of the actual ISN increment between probes and collected them into bins with a spacing of 250000. The following plot shows the total fraction of increments that fell into each bin. The graph also shows the total fraction of increments that we expect to see in each bin using the approximation functions we derived. The two curves agree quite well.



The approximations we derived can now be put to use to make predictions about ISN values that will be used in the future or ISN values that have been used in the past. Most important, we would like to know what the likely range of ISN values was or will be at a different time. First, we must calculate the number of increment operations that have or that will have occurred. We know that two increment operations happen every second. We don't know exactly how many increments have occurred due to incoming and outgoing connections to the machine. Fortunately, we can guess at a range of likely values for these numbers. We then compute the number of increments as the time difference plus the number of connections initiated or received.

$$n = 2\mathrm{num}_{seconds} + \mathrm{num}_{connects}$$

Having the number of increments, we compute the expected increment and the standard deviation. We can then predict the range of ISN values by considering the sequence numbers within a few standard deviations of the current ISN plus the expected increment value:

$$\begin{aligned} \mathrm{range}_{min} &= \mathrm{ISN} + m[n] - fs[n] \\ \mathrm{range}_{max} &= \mathrm{ISN} + m[n] + fs[n] \end{aligned}$$

where *f/2* is the number of standard deviations around the mean we wish to consider. Because the distribution is Gaussian, we can select *f* to allow us to predict the sequence number to whatever accuracy we desire.

The following values were derived from a normal distribution table:

Factor f	Certainty
0.68	50 %
1.65	90 %
1.96	95 %
2.58	99 %

If the exact value of *n* was not known (for example, if the time of the connection was not known with certainty, or the number of connections was not known), we simply repeat the computations for the two extreme guesses:

$$\begin{aligned} \mathrm{range}_{min} &= \mathrm{ISN} + m[n_{min}] - fs[n_{min}] \\ \mathrm{range}_{max} &= \mathrm{ISN} + m[n_{max}] + fs[n_{max}] \end{aligned}$$

To illustrate these concepts, we present an example. We know that a connection was made between 25 and 30 minutes ago, and we estimate that no more than 50 connections have been made to the machine since. We choose a range of increment iterations that we think are likely:

$$\begin{aligned} n_{min} &= 25 \cdot 60 \cdot 2 + 0 = 3000 \\ n_{max} &= 30 \cdot 60 \cdot 2 + 50 = 3650 \end{aligned}$$

We then compute the range of probable ISN values that would have been realized, with 95% confidence:

$$\begin{aligned} \mathrm{range}_{min} &= \mathrm{ISN} - (64000 \cdot 3000 - 1.96 \cdot 36950\sqrt{3000}) \\ &= \mathrm{ISN} - 188033283 \\ \mathrm{range}_{max} &= \mathrm{ISN} - (64000 \cdot 3650 + 1.96 \cdot 36950\sqrt{3000}) \\ &= \mathrm{ISN} - 237975391 \end{aligned}$$

Note that we subtracted from the ISN, since we are computing a previous ISN value. Because of this, the ranges are also backwards. It is also worth noting that even though more than 25 minutes have gone by, the ISN values are still known, with a high level of confidence, to be in a range of only 49942108 values. This is only 1.2% of the range of total possible sequence numbers.

This technique can be used to narrow down the range of possible sequence numbers, but for how long? We can calculate the maximum number of seconds that must expire before the range of possible ISN values is *p* percent of the total sequence space. This is done by assuming that increments occur only due to clock ticks. The percentage *p* of the total sequence space that the ISN value is likely to be within, with 95% certainty after *s* seconds, is:

$$\begin{aligned} p &= \frac{100\text{range}}{2^{32}} \\ &= \frac{100 \cdot 2 \cdot 1.96m[2s]}{2^{32}} \\ &= \frac{100 \cdot 2 \cdot 1.96m_1\sqrt{2s_1}}{2^{32}} \end{aligned}$$

if we solve for *s* we get:

$$\begin{aligned} s &= \left(\frac{2^{32}p}{\sqrt{2} \cdot 100 \cdot 2 \cdot 1.96m_1}\right)^2 \\ &= \left(\frac{7747450}{m_1}\right)^2 p^2 \\ &= 43963p^2 \end{aligned}$$

This equation results in the following values:

% Sequence Space	Time
1	12 hours 13 minutes
10	50 days 21 hours
50	3 years 177 days

If the machine is accepting incoming connections or making outgoing connections, the amount of time it would take to spread over these ranges would be shorter.

Implications

Armed with a technique for guessing the range of an ISN, we now turn to practical applications. The ISN value is exchanged during initial TCP/IP session setup in a process known as a three-way

handshake. After two peers decide to establish a connection and exchange their ISN values, the session continues to use sequence numbers to keep track of the bytes sent. At any time in a connection, the sequence number to be sent next will be the ISN of the sender plus the number of bytes already sent.

TCP/IP is vulnerable to a number of attacks if the sequence numbers of the connection are known. If the ISN is known exactly, a forged three-way handshake can be performed without receiving any return traffic. This attack is discussed in [Morr] and [Bell2]. After a session is established, data and control packets can be inserted into the session if the proper sequence numbers are known. Each TCP implementation will keep track of the amount of space it is willing to buffer from the sender. This is known as the receive window. TCP implementations will accept data packets that are sent with sequence numbers within this window. Control packets such as FIN and RST packets are also processed if they are within this window Illegal control packets are also received if they are within the window, and result in the aborting of the established session.

As a result, if a sequence number within the receive window is known, an attacker can inject data into the session stream or terminate the connection. If the ISN value is known and the number of bytes sent already sent is known, an attacker can send a simple packet to inject data or kill the session. If these values are not known exactly, but an attacker can guess a suitable range of values, he can send out a number of packets with different sequence numbers in the range until one is accepted. The attacker need not send a packet for every sequence number, but can send packets with sequence numbers a window-size apart. If the appropriate range of sequence numbers is covered, one of these packets will be accepted. The total number of packets that needs to be sent is then given by the range to be covered divided by the fraction of the window size that is used as an increment.

To illustrate this more clearly, let us return to the example used in the previous section. After more than 25 minutes, it was noted that the initial sequence number must be within a range of 49942108 values. If we know that no more than 1 Megabyte of data has been sent over the connection, then the sequence number is likely to be within a range of 50990684 values. OpenBSD typically advertises a receive window of 16 kilobytes. If we assume that the receiver's buffers are no more than 50% full at the time of the attack, we can use an increment of 8000 between the sequence number of each packet sent out. To kill the connection, we need only generate 50990684 / 8000 or 6374 SYN packets within the guessed range. Not counting hardware encapsulation, this represents only 250 kilobytes of data, which would take no more than a few seconds to transmit over medium-bandwidth links.

Fixes

An alternate solution to fix the problems originally discovered by Robert Morris was proposed in RFC 1948 [Bell1]. This solution partitions the sequence space by connection identifiers. Each connection identifier, which is composed of the local address and port and the remote address and port of a connection, is assigned its own unique sequence space starting at an offset that is a function of the connection identifier. The function is chosen in such a way that it cannot be computed by an attacker. The ISN is then be generated by increments to this offset. ISN values generated in this way for one connection are independent of the ISN values generated for other connections. Because of this, observed ISN values cannot be used to infer ISN values of other connections. This method of ISN generation is not susceptible to the problems outlined in this paper.

TCP streams can also be cryptographically protected. One such method is outlined in RFC 2385 [Heff] as a stopgap measure for protecting BGP streams. Although the proposal has some weaknesses, it can effectively protect against attacks that inject data or control messages into TCP streams.

The techniques outlined in this paper narrow down the range of sequence numbers used by a connection to an acceptable range that make attacks feasible. As networks become faster and computers use increasingly larger receive buffers, it will become more feasible to perform attacks without any information about what range of sequence numbers are in use. It may eventually be necessary to increase the size of TCP's sequence space to prevent such attacks. This could be done through the addition of a new TCP option similar to the TCP timestamp option outlined in RFC 1323 [Jaco].

The ultimate solution to network tampering is the deployment of widespread encryption and authentication between Internet peers. Although protocols protecting IP traffic between peers exist, they are currently not in widespread use due to political and technical issues that are not yet resolved.

Conclusions

Although random increments have been successful against protecting TCP initial sequence numbers against earlier attacks, we have discovered that they still leave weaknesses in TCP implementations that use them. Better methods of ISN generation have been available for some time and should be used. In the future, even the most sophisticated method of ISN generation may not be able to protect TCP from sequence number attacks. Although TCP can be extended to provide increased protection by enlarging its sequence space size, cryptographic protection between Internet peers would provide the most complete solution to network tampering.

Acknowledgements

I would like to thank Nate Lawson and Steve Bellovin for reviewing early drafts of this paper. Their comments helped improve the quality of this paper. I would also like to thank my employer, Guardent, for allowing this research to take place.

Appendix

Derivation of the Change in Distribution

In this section, we will derive approximation functions for the growth in the standard deviation and expected value of the distribution with the number of random increments. We begin by noting that after 25 iterations of random increments, the distribution can be approximated by a normal distribution *n(m_25, s_25 ^ 2)* with mean (or expected value) *m_25* and standard deviation *s_25*. This is a direct application of the Central Limit Theorem of probability theory. Let X_25 be a random number chosen from this distribution. It follows that after 50 iterations, the random increment is the sum of two such randomly chosen numbers: X_50 = X_25 + X_25. The sum of any two random variables chosen with from normal distributions *n(m1, s1 ^ 2)* and *n(m2, s2 ^ 2)* is *n(m1 + m2, s1 ^ 2 + s2 ^ 2)*. It follows that the distribution of X_50 is then *n(2 m_25, 2 s_25 ^ 2)*.

In general, the mean and standard deviation after *n* such random increments is given by:

$$m'[n] = m_{25}n$$

$$s'[n] = s_{25}\sqrt{n}$$

We interpolate this to individual iterations by:

$$\begin{aligned} m[n] &= m'[n/25] \\ &= m_{25}\frac{n}{25} \\ &= \frac{m_{25}}{25}n \end{aligned}$$

$$\begin{aligned} s[n] &= s'[n/25] \\ &= s_{25}\sqrt{\frac{n}{25}} \\ &= \frac{s_{25}}{\sqrt{25}}\sqrt{n} \end{aligned}$$

By noting that *m*[1] is *(m_25 / 25)* and *s*[1] is *(s_25 / sqrt(25))* we can rewrite this as:

$$m[n] = m_1 n$$

$$s[n] = s_1\sqrt{n}$$

Where *m1* and *s1* are the values *m[1]* and *s[1]* respectively.

We note experimentally that the values *m1* and *s1* are the expected value and standard deviation of the distribution of random values that the random increment was chosen from.

Properties of Uniform Distributions

In this section we derive the expected value and the standard deviation of uniform distributions of arbitrary widths. A uniform distribution has equal probability for all values over a specific range. We will call this range the width, or *w* for short. Because the sum of all values in the probability density function must equal 1.0, the probability density function for a uniform distribution of width *w* is given as:

$$f[n] = \begin{cases} 1/w & 0 \le n < w \\ 0 & \text{otherwise} \end{cases}$$

The *r*th moment of any distribution is given as:

$$\begin{aligned} \mathrm{EX}^r &= \sum_x x^r f[x] \\ &= \sum_{x=0}^{w-1} \frac{x^r}{w} \\ &= \frac{1}{w} \sum_{x=0}^{w-1} x^r \end{aligned}$$

The expected value of *f* is the first moment EX, or:

$$\begin{aligned} m &= \mathrm{EX} \\ &= \frac{1}{w} \sum_{x=0}^{w-1} x \\ &= \frac{w-1}{2} \end{aligned}$$

The second moment is :

$$\begin{aligned} \mathrm{EX}^2 &= \frac{1}{w} \sum_{x=0}^{w-1} x^2 \\ &= \frac{(w-1)(2w-1)}{6} \end{aligned}$$

The standard deviation is then:

$$\begin{aligned} s^2 &= \mathrm{EX}^2 - (\mathrm{EX})^2 \\ &= \frac{(w-1)(2w-1)}{6} - \left(\frac{w-1}{2}\right)^2 \\ &= \frac{w^2-1}{12} \end{aligned}$$

$$s = \sqrt{\frac{w^2-1}{12}}$$

$$= \frac{\sqrt{3}}{6}\sqrt{w^2-1}$$

$$= 0.2887\sqrt{w^2-1}$$

Implementation Specific Constants

OpenBSD

The random increment used by OpenBSD was analyzed in a previous section. The resulting values of *m_1* and *s_1* are *64000* and *36950* respectively.

FreeBSD

FreeBSD increments its ISN value twice a second and once for each incoming and outgoing connection request. For each incoming and outgoing connection request, the ISN value is incremented by a definite amount of 31232 and by a random increment of size 65536. It is also incremented by twice this amount (31232 plus a random increment of size 131072) twice a second. If *m_c* and *s_c* are the mean and standard deviation of the first (smaller) increment, then the mean and standard deviation of the larger increment are *2 m_c* and *sqrt(2) s_c* respectively. Gathering the distributions for both of these increments into a single equation results in:

$$m = 2n_{seconds}m_c + n_{connects}(2m_c)$$

$$= 2m_c(n_{seconds} + n_{connects})$$

$$= 2m_cn$$

$$s = \sqrt{2n_{seconds}s_c^2 + n_{connects}(2s_c^2)}$$

$$= s_c\sqrt{2}\sqrt{n_{seconds} + n_{connects}}$$

$$= s_c\sqrt{2}\sqrt{n}$$

Where *n* is the sum of the number of incoming connections, outgoing connections and seconds. The values of m_c and s_c are 48384 and 18920 respectively. It follows then that *m_1* is *96768* and *s_1* is *26757*.

NetBSD

NetBSD [NetB] does not use a random increment to update its ISN value, but it does use a similar technique. Twice a second, and once for each incoming and outgoing connection, a definite increment is added to an internal ISN value. Actual ISN values are then generated by adding a random value to this internal variable. The result of this algorithm is that the range that the ISN is in is always known to 100% accuracy if the number of increments is known, and the ISN is always distributed evenly across this range. The size of the definite increment and the maximum size of the random value are *16777216* or *1/256* the of the total sequence space. Because the size is so large, this technique is more resilient to attack. An attacker would have to know the number of increments that have occurred to within a few increments to narrow down the sequence space significantly.

References

[Bell1] S. M. Bellovin. RFC 1948, Defending Against Sequence Number Attacks. May 1996.

[Bell2] S. M. Bellovin, *Security Problems in the TCP/IP Protocol Suite*, Computer Communications Review Vol 19, No 2, pp. 32-48, April 1989. (available at http://www.research.att.com/~smb/papers/ipext.ps

[Free] FreeBSD. http://www.freebsd.org/.

[Heff] A. Heffernan, RFC 2385, Protection of BGP Sessions via the TCP MD5 Signature Option. 1998.

[Hoel] P. Hoel, S. Port, C. Stone, Introduction to Probability Theory. Houghton Mifflin Company, 1971.

[Jaco] V. Jacobson, R. Braden, D. Borman, RFC 1323, TCP Extensions for High Performance. 1992.

[Morr] R. T. Morris, *A Weakness in the 4.2BSD Unix TCP/IP Software*, AT&T Bell Laboratories Computing Science Technical Report #117, February 1985. (available at ftp://research.att.com/dist/internet_security/117.ps.Z)

[NetB] NetBSD. http://www.netbsd.org/.

[Open] OpenBSD. http://www.openbsd.org/.

[Post] J. Postel, RFC 793, Tranmission Control Protocol. 1981.