CERT Advisory CA-2001-09 Statistical Weaknesses in TCP /IP Initial Sequence Numbers

Original release date: May 01, 2001
Last revised: Feb 28, 2005
Source: CERT/CC

A complete revision history can be found at the end of this file.

Systems Affected

- Systems using TCP stacks which have not incorporated RFC1948 or equivalent improvements
- Systems not using cryptographically-secure network protocols like IPSec

Overview

Attacks against TCP initial sequence number (ISN) generation have been discussed for some time now. The reality of such attacks led to the widespread use of pseudo-random number generators (PRNGs) to introduce some randomness when producing ISNs used in TCP connections. Previous implementation defects in PRNGs led to predictable ISNs despite some efforts to obscure them. The defects were fixed and thought sufficient to limit a remote attacker's ability to attempt ISN guessing. It has long been recognized that the ability to know or predict ISNs can lead to manipulation or spoofing of TCP connections. What was not previously illustrated was just how predictable one commonly used method of partially randomizing new connection ISNs is in some modern TCP/IP implementations.

A new vulnerability has been identified (CERT VU#498440, CVE CAN-2001-0328) which is present when using random increments to constantly increase TCP ISN values over time. Because of the implications of the Central Limit Theorem, adding a series of numbers together provides insufficient variance in the range of likely ISN values allowing an attacker to disrupt or hijack existing TCP connections or spoof future connections against vulnerable TCP/IP stack implementations. Systems relying on random increments to make ISN numbers harder to guess are still vulnerable to statistical attack.

I. Description

Some History

In 1985, Bob Morris first identified potential security concerns [ref_morris] with the TCP protocol. One of his observations was that if a TCP sequence number could be predicted, an attacker could "complete" a TCP handshake with a victim server without ever receiving any responses from the server. One result of the creation of such a "phantom" connection would be to spoof a trusted host on a local network.

In 1989, Steve Bellovin [ref_bellovin] observed that the "Morris" attack could be adapted to attack client connections by simulating unavailable servers and proposed solutions for strengthening TCP ISN generators. In 1995, the CERT Coordination Center issued CA-1995-01, which first reported the widespread use of such attacks on the Internet at large.

Later in 1995, as part of RFC1948, Bellovin noted:

> The initial sequence numbers are intended to be more or less random. More precisely, RFC 793 specifies that the 32-bit counter be incremented by 1 in the low-order position about every 4 microseconds. Instead, Berkeley-derived kernels increment it by a constant every second, and by another constant for each new connection. Thus, if you open a connection to a machine, you know to a very high degree of confidence what sequence number it will use for its next connection. And therein lies the attack.
>
> Also in 1995, work by Laurent Joncheray [ref_joncheray] further describes how an attacker could actively hijack a TCP connection. If the current sequence number is known exactly and an attacker's TCP packet sniffer and generator is located on the network path followed by the connection, victim TCP connections could be redirected.
>
> In his recently published paper on this issue, [ref_newsham] Tim Newsham of Guardent, Inc. summarizes the more generalized attack as follows:
>
> > 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.
> >
> > Many TCP/IP implementers turned to incrementing the global tcp_iss [TCP Initial Send Sequence number, a.k.a., an ISN] variable using pseudo-random variables instead of constants. Unfortunately, the randomness of the pseudo-random-number generators (PRNGs) used to generate the "random" increments was sometimes lacking (see CVE-1999-0077, CVE-2000-0328, CAN-2000-0916, CAN-2001-0288, among others). As noted in RFC1750:
> >
> > > It is important to keep in mind that the requirement is for data that an adversary has a very low probability of guessing or determining. This will fail if pseudo-random data is used which only meets traditional statistical tests for randomness or which is based on limited range sources, such as clocks. Frequently such random quantities are determinable by an adversary searching through an embarrassingly small space of possibilities.
> > >
> > > Eastlake, Crocker, and Schiller were focused on randomness in cryptographic systems, but their observation was equally applicable in any system which relies on random number generation for security. It has been noted in the past that using such poor PRNGs can lead to smaller search spaces and make TCP ISN generators susceptible to practical brute-force attacks.

However, new research demonstrates that the algorithm implemented to generate ISN values in many TCP/IP stacks is statistically weak and susceptible to attack even when the PRNG is adequately randomizing its increments. The problem lies in the use of increments themselves, random or otherwise, to advance an ISN counter, making statistical guessing practical.

Some Fresh Analysis: Guardent

Tim Newsham of Guardent, Inc. has written a paper titled "The Problem with Random Increments" [ref_newsham] concerning an observed statistical weakness in initial sequence number generation for TCP connections. Newsham explains how incrementing the ISN by a series of pseudo-random amounts is insufficient to protect some TCP implementations from a practical ISN guessing attack in some real-world situations. Such attacks would not rely on data sniffed from a victim site but only on one or two ISN samples collected by previous connections made to a victim site. Newsham's statistical analyses provide a theoretical backdrop for practical attacks, drawing attention once again to the protocol analysis documented by Steve Bellovin (building on work pioneered by Robert Morris) in RFC1948.

Newsham points out that the current popular use of random increments to obscure an ISN series still contains enough statistical information to be useful to an attacker, making ISN guessing practical enough to lead to TCP connection disruption or manipulation. This attack is possible because an attacker can still predict within "a suitable range of values" what the next (or a previous) ISN for a given TCP connection may be. This range can be derived when looking at the normal distribution that naturally arises when adding a large number of values together (random or otherwise) due to expected values governed by the Central Limit Theorem [ref_clt]:

> Roughly, the central limit theorem states that the distribution of the sum of a large number of independent, identically distributed variables will be approximately normal, regardless of the underlying distribution.

In addition to statistical analysis of this weakness, Newsham's paper demonstrates the weakness inherent in one specific TCP/IP implementation. In other recently-published research, Michal Zalewski of BindView surveys over 20 different ISN generators included in many of the most widely available operating systems on the Internet today. Their work shows in graphic detail how observable this statistical weakness is.

Some Fresh Empirical Evidence: BindView

Analysts at BindView have produced interesting research that analyzes the patterns many of the most popular TCP/IP stacks produce when producing ISNs. In a paper titled "Strange Attractors and TCP/IP Sequence Number Analysis," [ref_zalewski] author Michal Zalewski uses phase analysis to show patterns of correlation within sets of 32-bit numbers generated by many popular operating systems' TCP ISN generators. As Zalewski explains:

> Our approach is built upon this widely accepted observation about attractors:
>
> If a sequence exhibits strong attractor behavior, then future values in the sequence will be close to the values used to construct previous points in the attractor.
>
> Our goal is to construct a spoofing set, and, later, to calculate its relative quality by empirically calculating the probability of making the correct ISN prediction against our test data. For the purpose of ISN generators comparison , we established a limit of guess set size at the level of 5,000 elements, which is considered a limit for trivial attacks that does not require excessive network bandwidth or processing power and can be conducted within few seconds.
> (A "spoofing set" is defined as "a set of guessed values for ISNs that are used to construct a packet flood that is intended to corrupt some established TCP connections." Please see [ref_zalewski] for more information about phase space analysis and attractor reconstruction).
>
> In effect, using this technique for data visualization, they are able to highlight emergent patterns of correlation. Such correlation, when present in TCP ISN generators, can dramatically shrink the set of numbers that need to be guessed in order to attack a TCP session.
>
> Since the sequence number for TCP sessions is stored in packet headers using 32-bits of data, it was generally assumed that an attacker would have a very small chance of correctly guessing a sequence number to attack established (or to-be established) connections. BindView's research shows attackers actually have much smaller bit-spaces to guess within due to dependencies on system clocks and other implementation defects.
>
> Zalewski further notes in his paper [ref_zalewski]:
>
> > What comes to our attention is that most every implementation described above, except maybe current OpenBSD and Linux, has more or less serious flaws that make short-time TCP sequence number prediction attacks possible. Solaris 7 and 8 with tcp_strong_iss set to 2 results are a clear sign there are a lot of things to do for system vendors. We applied relatively loose measures, classifying attacks as "feasible" if they can be accomplished using relatively low bandwidth and a reasonable amount of time. But, as network speeds are constantly growing, it would be not a problem for an attacker having access to powerful enough uplink to search the entire 32-bit ISN space in several hours, assuming a local LAN connection to the victim host and assuming the network doesn't crash, although an attack could be throttled to compensate.

The work done by Guardent and BindView illustrates that not all current TCP/IP ISN generators have implemented the suggestions made by Steve Bellovin in RFC1948 to address prediction-based ISN attacks, or provided a equivalent fixes. In particular, TCP/IP stacks based on operating system software which has not previously incorporated RFC1948 or equivalent fixes will be susceptible to classic TCP hijacking in the absence of other cryptographically secure hardening (i.e., when not using IPSec or an equivalent secure networking technology). Much work remains to be done to ensure the systems deployed using TCP today and tomorrow have strengthened their ISN generators using RFC1948 recommendations or equivalent fixes.

II. Impact

If the ISN of an existing or future TCP connection can be determined within some practical range, a malicious agent may be able to close or hijack the TCP connections. If the ISNs of future connections of a system are guessed exactly, an agent may be able to "complete" a TCP three-way handshake, establish a phantom connection, and spoof TCP packets delivered to a victim.

The ability to spoof TCP packets may lead to other types of system compromise, depending on the use of IP-based authentication protocols. Examples of such attacks have been previously described in
CA-1995-01 and CA-1996-21.

III. Solution

The design of TCP specified by Jon Postel in
RFC793 specifically addressed the possibility of old packets from prior instantiations of a connection being accepted as valid during new instantiations of the same connection, i.e., with the same 4-tuple of <local host, local port, remote host, remote port>:

> To avoid confusion we must prevent segments from one incarnation of a connection from being used while the same sequence numbers may still be present in the network from an earlier incarnation. We want to assure this, even if a TCP crashes and loses all knowledge of the sequence numbers it has been using. When new connections are created, an initial sequence number (ISN) generator is employed which selects a new 32-bit ISN. The generator is bound to a (possibly fictitious) 32-bit clock whose low order bit is incremented roughly every 4 microseconds. Thus, the ISN cycles approximately every 4.55 hours. Since we assume that segments will stay in the network no more than the Maximum Segment Lifetime (MSL) and that the MSL is less than 4.55 hours we can reasonably assume that ISN's will be unique. Several criteria need to be kept in mind when evaluating each of the following solutions to this problem:
>
> 1. Does the soulution address the security concerns identified in this advisory?
> 2. How well does the solution conform for TCP reliability and interoperability requirements?
> 3. How easily can the solution be implemented?
> 4. How much of a performance cost is associated with the solution?
> 5. How well will the solution stand the test of time?
>
> In the discussions following the initial report of this statistical weakness, several approaches to solving this issue were identified. All have various strengths and weaknesses themselves. Many have been implemented independently by various vendors in response to other reported weaknesses in specific ISN generators.

Deploy and Use Cryptographically Secure Protocols

TCP initial sequence numbers were not designed to provide proof against TCP connection attacks. The lack of cryptographically-strong security options for the TCP header itself is a deficiency that technologies like IPSec try to address. It must be noted that in the final analysis, if an attacker has the ability to see unencrypted TCP traffic generated from a site, that site is vulnerable to various TCP attacks - not just those mentioned here. The only definitive proof against all forms of TCP attack is end-to-end cryptographic solutions like those outlined in various IPSec documents.

The key idea with an end-to-end cryptographic solution is that there is some secure verification that a given packet belongs in a particular stream. However, the communications layer at which this cryptography is implemented will determine its effectiveness in repelling ISN based attacks. Solutions that operate above the Transport Layer (OSI Layer 4), such as SSL/TLS and SSH1/SSH2, only prevent arbitrary packets from being inserted into a session. They are unable to prevent a connection reset (denial of service) since the connection handling will be done by a lower level protocol (i.e., TCP). On the other hand, Network Layer (OSI Layer 3) cryptographic solutions such as IPSec prevent both arbitrary packets entering a transport-layer stream and connection resets because connection management is directly integrated into the secure Network Layer security model.

The solutions presented above have the desirable attribute of not requiring any changes to the TCP protocol or implementations to be made. Some sites may want to investigate hardening the TCP transport layer itself though. RFC2385 ("Protection of BGP Sessions via the TCP MD5 Signature Option") and other technologies provide options for adding cryptographic protection within the TCP header at the cost of some potential denial of service, interoperability, and performance issues.

The use of cryptographically secure protocols has several advantages over other possible solutions to this problem. Protection against hijacking and disruption are provided by the cryptography, while the TCP layer is free to return to a simple increasing sequence number mechanism, providing the greatest level of reliability. The performance, durability, and practicality of implementation will vary according to the protocol selected, but IPSec in particular appears to have a number of positive attributes in this regard.

Use RFC1948 Implementations

In RFC1948, Bellovin observed that if the 32-bit ISN space could be segmented across all the ports available to a system, collecting sample ISNs from one connection could yield little or no information about the ISNs being generated in other connections. Breaking the reliance on a global ISN pool by using cryptographically hashed secrets and [IP, port] 4-tuples effectivly eliminates TCP ISN attacks by remote users (unless, of course, attackers able to sniff traffic on a local network segment).

Newsham notes in his paper [ref_newsham]:

> RFC 1948 [ref1] proposes a method of TCP ISN generation that is not vulnerable to ISN guessing attacks. The solution proposed 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 [...] generated by increments to this offset. ISN values generated in this way are not vulnerable to ISN range prediction methods outlined in this paper since an attacker cannot gain knowledge of the ISN space for any connection identifiers he cannot directly observe.
>
> Once the global ISN space becomes segmented among all the TCP ports available on a system, attacking TCP ISNs remotely becomes impractical. However, it should be noted that even when using RFC1948 implementations, some forms of ISN attack remain viable under very specific conditions, as discussed in further detail below.
>
> In addition, using a cryptographically strong hash function to perform this segmentation may lead to longer TCP connection establishment time. Some implementors (like those of the Linux kernel) have chosen to use a reduced-round MD4 hash function to provide a "good enough" solution from a security standpoint to keep performance degradation to a minimum. One cost of weakening the hash algorithm is the need to re-key the generator every few minutes. Each time a re-keying occurs, security is strengthened, but other reliability issues identified in RFC 793 become a concern.
>
> It had been understood (but not widely noted) that ISNs generated by a "strictly-compliant" RFC1948 generator would still allow ISN guessing attacks to be made against previously-owned IP addresses. If an attacker could "own" an IP address used by a potential victim at some point afterward, given enough sample ISNs collected within the shared [IP, port] 4-tuple ISN space, an attacker could make reasonable guesses about the ISNs of subsequent connections.
>
> This is because strict RFC1948 suggests the following algorithm:
>
> ```
>         ISN = M + F(sip, sport, dip, dport, <some secret>)
> ```
>
> where
>
> ```
>         ISN   = 32-bit initial sequence number
>         M     = monotonically increasing clock/counter
>         F     = crypto hash (typically MD4 or MD5)
>         sip   = source IP
>         sport = source port
>         dip   = destination IP
>         dport = destination port
>
>         <some secret> = an optional fifth input into the
> hash function
>                         to make remote IP attacks unfeasible.
> ```

For the ISN itself to monotonically (constantly) increase, **F()** needs to remain fairly static. So the <some secret> envisioned by Bellovin was a system-specific value (such as boot time, a passphrase, initial random value, etc) which would infrequently change. Each time it changes, the value of **F()** (a hash) changes and there is no guarantee that subsequent ISNs will be sufficiently distanced from the previous value assigned, raising the potential RFC793 reliability concern again.

When viewed from the perspective of a particular [IP, port] 4-tuple, the ISN sequence is predictable and therefore subject to practical attacks. When looking at the Solaris `tcp_strong_iss` generator (RFC1948) from the perspective of a remote IP attacker, for example, the ISNs generated appear random. However, the Zalewski paper analyzes data which looks at both the remote and same-IP address attack vectors. Their data confirms the same-IP attack vector against Solaris `tcp_strong_iss=2` (RFC1948) is a practical attack.

The Linux TCP implementors avoided this issue by rekeying <some secret> every five minutes. Unfortunately, this breaks the monotonicity of the algorithm, weakening the iron-clad reliability guarantee that Bellovin was hoping to preserve by segmenting the ISN space among ports in the first place.

Some have proposed that the following algorithm may be a better answer to this issue:

```
M   = M + R(t)
ISN = M + F(sip, sport, dip, dport, <some secret> )
```

where

```
R(t)   = some random value changing over time
```

This is essentially adding a random increment to the RFC1948 result. This makes most attacks impractical, but still theoretically possible. (It would still be "RFC1948-compliant" as well ... RFC1948 makes as few assumptions about the **F()** incrementing function as possible, requiring only that the connection [IP, port] 4-tuple be inputs to the function and that it be practically irreversible.) However, the "problem" of random increments was what brought this issue back into the spotlight to begin with.

Use Some Other Non-RFC1948 Approaches

A more direct solution chosen by some TCP implementors is to simply feed random numbers directly into the ISN generator itself. That is, given a 32-bit space to choose from, assign:

```
ISN = R(t)
```

Solutions which essentially randomize the ISN seem to mitigate against the practical guessing attack once and for all (assuming strong pseudo-random number generation). However, a purely-random approach allows for overlapping sequence numbers among subsequently-generated TCP connnections sharing [IP, port] 4-tuples. For example, a random generator can produce the same ISN value three times in a row. This runs contrary to multiple RFC assumptions about monotonically increasing ISNs (RFC 793, RFC 1185, RFC 1323, RFC1948, possibly others as well). It is unclear what practical effect this will have on the long-term reliability guarantees the TCP protocol makes or is assumed to make.

Another novel approach introduced by Niels Provos of the OpenBSD group tries to strike a balance between the fully-random and segmented (RFC1948) approaches:

```
ISN = ((PRNG(t)) << 16) + R(t)
```

where

```
PRNG(t) = a pseudo-randomly ordered list of
          sequentially-generated 16-bit numbers
R(t)    = a 16-bit random number generator
          with its msb always set to zero
```

(This formula is an approximation of the results the OpenBSD implementation actually generates. Please see their actual code at: http://www.openbsd.org/cgi-bin/cvsweb/src/sys/netinet/tcp_subr.c) What the Provos implementation effectively does is generate a psuedo-random sequence that will not generate duplicate ISN values within a given time period. Additionally, each ISN value generated is guaranteed to be at least 32K away from other ISN values. This avoids the purely-random ISN collision problem, as well as makes a stronger attempt to keep sequence number spaces of subsequent [IP, port] 4-tuple

connections from overlapping. It also avoids the use of a cryptographic hash which could degrade performance. However, monotonicity is lost, potentially causing reliability problems, and the generator may leak information about the system's global ISN state.

Further discussion and analysis on the importance of such attributes needs to occur in order to ascertain the characteristics present in each ISN generator implemented. Empirical evidence provided by BindView *ma* indicate that from a predictability standpoint, the solutions are roughly equivalent when viewed from a remote attackers perspective. It is unclear at the time of this writing what the security, performance, and reliability tradeoffs truly are.

Appendix A. - Vendor Information

This appendix contains information provided by vendors for this advisory. When vendors report new information to the CERT/CC, we update this section and note the changes in our revision history. If a particular vendor is not listed below, we have not received their comments.

Cisco Systems

Cisco systems now use a completely random ISN generator.

Please see the following for more details:

http://www.cisco.com/warp/public/707/ios-tcp-isn-random-pub.shtml

Compaq Computer Corporation

At the time this document was written, Compaq is investigating the potential impact to Compaq's Tru64 UNIX and OPENVMS operating systems. Compaq views the problem to be a concern of moderate severity. Compaq implementations of TCP/IP sequence randomization for Tru64 UNIX for Alpha and OpenVMS for Alpha follow current practices for implementation of TCP/IP initial sequence numbers.

If and when further information becomes available Compaq will provide notice of the completion/availability of any necessary patches or tuning recommendations through AES services (DIA, DSNlink FLASH and posted to the Services WEB page) and be available from your normal Compaq Global Services Support channel. You may subscribe to several operating system patch mailing lists to receive notices of new patches at:

http://www.support.compaq.com/patches/mailing-list.shtml

FreeBSD, Inc.

FreeBSD has adopted the code and algorithm used by OpenBSD 2.8-current in FreeBSD 4.3-RELEASE and later, and this release is therefore believed not to be vulnerable to the problems described in this advisory (for patches and information relating to older releases see FreeBSD Security Advisory 01:39). We intend to develop code in the near future implementing RFC 1948 to provide a more complete solution.

Fujitsu

Fujitsu is currently working on the patches for the UXP /V operating system to address the vulnerabilities reported in
VU#498440.

The patches will be made available with the following ID numbers:

OS Version,PTF level	patch ID
UXP/V V20L10 X01021	UX28164
UXP/V V20L10 X00091	UX28163
UXP/V V10L20 X01041	UX15529

Hewlett-Packard Company

Date: Thu Aug 29 20:52:48 2002

```
The following tcp
randomizations are now
available:

        HP-UX releases
11.00, 11.04, and 11.11
(11i):
             - HP
randomization
             - RFC
1948 ISN randomization


        For HP
randomization on
releases:
            HP-UX
11.00:       PHNE_22397
or subsequent,
            HP-UX
11.11:       default
mode.

         For RFC 1948
ISN randomization
            HP-UX
11.00:       PHNE_26771
or subsequent,
            HP-UX
11.04:       PHNE_26101
or subsequent,
            HP-UX
11.11:       PHNE_25644
or subsequent.



 To enable tcp
randomization on HP-UX
11.00, 11.04, and 11.11
(11i):

------------------------
------------------------
----------------------
--

  HP randomization

     HP-UX release
11.00:
     Install PHNE_22397
or subsequent.  The HP
randomization will
     then be the
default tcp
randomization.

       NOTE: This patch
has dependencies.


     HP-UX release
11.11 (11i):
     No patch is
required.  The HP
randomization has
always been
```

implemented in HP-UX 11.11 (11i) and is the default tcp randomization.

RFC 1948 ISN randomization

HP-UX 11.00: Apply PHNE_26771 or subsequent.
HP-UX 11.04: Apply PHNE_26101 or subsequent.
HP-UX 11.11 (11i): Apply PHNE_25644 or subsequent.

Once the appropriate patch has been applied the RFC 1948 ISN randomization can be enabled on HP-UX 11.00, 11.04 and 11.11 by executing the following command as root:

ndd -set /dev /tcp tcp_isn_passphrase
where is any length character string. Only the first 32 characters will be

retained. If the passphrase is changed the system should be rebooted.

NOTE: RFC 1948 ISN randomization is not available on HP-UX release 10.20. Customers who want RFC 1948 ISN randomization should upgrade to HP-UX 11.X and apply necessary patches as discussed herein.

For the the legacy 10.20 release:

HP created a tunable kernel parameter that can enable two levels of randomization. This randomization feature requires a TRANSPORT patch level of:

For S700 platform: PHNE_17096 or greater
For S800 platform: PHNE_17097 or greater

The tunable kernel parameter is set as follows using the "nettune"

```
program:

    tcp_random_seq set
to 0  (Standard TCP
sequencing)
    tcp_random_seq set
to 1  (Random TCP
sequencing)
    tcp_random_seq set
to 2  (Increased Random
TCP sequencing)

  and requires a reboot.
--
```

IBM Corporation

We have studied the document written by Guardent regarding vulnerabilities caused by statistical analysis of random increments, that may allow a malicious user to predict the next sequence of chosen TCP connections.

IBM's AIX operating system should not be vulnerable as we have implemented RFC 1948 in our source coding. According to Guardent, we do not expect an exploit described in the document to affect our AIX OS because we employ RFC 1948.

Linux

The Linux kernel has used a variant of RFC1948 by default since 1996. Please see:

http://lxr.linux.no/source/drivers/char/ChangeLog#L258
http://lxr.linux.no/source/drivers/char/random.c#L1855

OpenBSD

post-2.8 we no longer use random increments, but a much more sophisticated way.

SGI

SGI implemented RFC 1948 with MD5 on IRIX 6.5.3 and above using the tcpiss_md5 tunable kernel parameter, but the default is disabled. To enablee tcpiss_md5 kernel parameter, use the following command as root:

```
        # /usr/sbin/systune -b
tcpiss_md5 1
```

To verify RFC 1948 has been enabled in IRIX, use the following command as root:

```
        # /usr/sbin/systune
tcpiss_md5
```

This should return:

```
        tcpiss_md5 = 1 (0x1)
```

The latest IRIX 6.5 Maintenance Releases can be obtained from the URL:

http://support.sgi.com/colls/patches/tools/relstream/index.html

An SGI security advisory will be issued for this issue via the normal SGI security information distribution methods including the *wiretap* mailing list and http://www.sgi.com/support/security/ .

Sun Microsystems, Inc.

Sun implemented RFC 1948 beginning with Solaris 2.6, but it isn't turned on by default. On Solaris 2.6, 7 and 8, edit /etc/default/inetinit to set TCP_STRONG_ISS to **2**.

On a running system, use:

```
ndd -set /dev/tcp
tcp_strong_iss 2
```

Appendix B. - References

1. Postel, J., "RFC 793: *TRANSMISSION CONTROL PROTOCOL: DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION*," September 1981. ftp://ftp.isi.edu/in-notes/rfc793.txt
2. Eastlake, D., Crocker, S., Schiller, J., "RFC 1750: *Randomness Recommendations for Security*," December 1994. ftp://ftp.isi.edu/in-notes/rfc1750.txt
3. Bellovin, S., "RFC 1948: *Defending Against Sequence Number Attacks*," May 1996. ftp://ftp.isi.edu/in-notes/rfc1948.txt
4. Heffernan, A., "RFC 2385: *Protection of BGP Sessions via the TCP MD5 Signature Option*," August 1998. ftp://ftp.isi.edu/in-notes/rfc2385.txt
5. Thayer, R., Doraswamy, N., Glenn, R., "RFC 2411: *IP Security Document Roadmap*," November 1998. ftp://ftp.isi.edu/in-notes/rfc2411.txt
6. CERT® Advisory CA-1995-01: *IP Spoofing Attacks and Hijacked Terminal Connections* http://www.cert.org/advisories/CA-1995-01.html
7. CERT® Advisory CA-1996-21: *TCP SYN Flooding and IP Spoofing Attacks* http://www.cert.org/advisories/CA-1996-21.html

8. *A Weakness in the 4.2BSD UNIX TCP/IP Software*, Morris, R., Computing Science Technical Report No 117, ATT Bell Laboratories, Murray Hill, New Jersey, 1985. http://www.pdos.lcs.mit.edu/~rtm/papers/117.pdf

9. *Security Problems in the TCP/IP Protocol Suite*, Bellovin, S., Computer Communications Review, April 1989. http://www.research.att.com/~smb/papers/ipext.ps http://www.research.att.com/~smb/papers/ipext.pdf

10. *Simple Active Attack Against TCP*, Joncheray, L., Proceedings, 5th USENIX UNIX Security Symposium, June 1995. http://www.usenix.com/publications/library/proceedings/security95/full_papers/joncheray.txt

11. Newsham, T., "Guardent White Paper: *The Problem with Random Increments*," February 2001. http://www.guardent.com/comp_news_tcp.html

12. Zalewski, M., "Razor Paper: *Strange Attractors and TCP/IP Sequence Number Analysis*," April 2001. http://razor.bindview.com/publish/papers/tcpseq.html

13. Virtual Laboratories in Probability and Statistics, Random Samples Section 5: The Central Limit Theorem

14. CVE-1999-0077
15. CVE-2000-0328
16. CAN-2000-0916
17. CAN-2001-0288
18. CAN-2001-0328

19. Havrilla, J., "CERT Vulnerability Note VU#498440: *Multiple TCP/IP implementations may use statistically predictable initial sequence numbers*", March 2001. https://www.kb.cert.org/vuls/id/498440

The CERT/CC thanks Guardent, Inc. and BindView for their invaluable contributions to this advisory. We also thank all the vendors who participated in the discussion about this vulnerability and proposed solutions.

We also thank the following people for their individual contributions to this advisory:

- Steve Bellovin, AT&T Labs
- Kris Kennaway, FreeBSD
- Mark Loveless, Bindview
- Tim Newsham, Guardent, Inc.
- Niels Provos, OpenBSD
- Damir Rajnovic, Cisco
- Theo de Raadt, OpenBSD
- Theodore Tso, MIT

Authors: Jeffrey S. Havrilla, Cory F. Cohen, Roman Danyliw, and Art Manion.

Revision History

```
May 01, 2001:  Initial
release
May 10, 2001:  Updated
vendor statement from HP
Sep 13, 2002:  Updated
vendor statement made Thu
Aug 29 20:52:48 2002 from HP
Feb 28, 2005:  Updated
Morris reference
```