# Global Information Assurance Certification Paper

**Secure Shell daemon crc32 compensation attack detector vulnerability.**
***Where have all the good programmers gone.***
Tim Yeager
December 24, 2001

## Introduction

'Twas the night before Christmas and all across the computer floor. The drives were all humming and ready to store. The systems safely locked down. Operating systems up to date. The latest patches were installed. Just hope it wasn't too late. The systems are hardened and access is tight. Only Secure Shell will let you in remotely tonight. Extreme amounts of caffeine in beverages were drank. All to protect from the hacker's pranks. Time be merry with little amateur rhymes. Relax and settle down to sleep for a time. Then with a clatter as a pager goes off and again it's now time to jump back to work. For it seems the hackers never sleep and the security job is never done. Reality sets in and it's time for more fun.

The bad news that the latest applied patches closed one vulnerability but created another. Another grim reminder of software bugs and patches. This lapse of vetting software for security is simply ponderous. What is happening in the software development arena to allow this malaise? This time was better than the last when an attacker found the exposure before it was fixed. Time to check the mailing lists, security sites and vendor sites for fixes. Crack open your favorite beverage and we can walk this through.

The whimsical introduction, while fun, is meant to emphasize the need in security for eternal vigilance. This paper will attempt to underscore the criticality, through the example of the recent security problems with the Secure Shell software, for the need to improve software to diminish security bugs and thus reduce the need for so many patches. For commonality between the various flavors the use of the term Secure Shell is used throughout this paper as opposed to SSH1 protocol or any vendor specific name. All instances used in conjunction with a vulnerability here are specifically related to the SSH1 protocol. The SSH2 protocol is safe from the vulnerabilities listed here unless it allows for SSH1 fallback. As with many pieces of software, upgrading to the latest version of Secure Shell may be delayed due to circumstances beyond security team control. Many times the security team and system administrators must work with patches upon patches to make systems secure. Thus, the added importance of getting the software right in the first place at least as far as security is concerned.

## Summary

As recently as December 13, 2001 the CERT/CC published an advisory CA-2001-35 about recent activity against Secure Shell daemons. Initially there was a problem with Secure Shell specifically the SSH1 protocol which uses a cyclic redundancy check (CRC) algorithm to perform checksums for data integrity. Weakness with this process could allow for modification of packets by the attacker. The victim would have no way to know a packet had been modified. CORE-SDI, an information security company, named the problem "ssh insertion attack" and released a patch to detect and disable attacks trying to exploit this vulnerability. The patch is

contained in a file called deattack.c. A software bug in the code of deattack.c introduced a remote buffer overflow vulnerability. That vulnerability is typically known as the Secure Shell daemon crc32 compensation attack detector or deattack patch vulnerability. There has now been a patch to the patch that eliminates the patch-induced vulnerability.

This is an example of a problem with poor software design for security and not fully vetting patches that are rushed into production. A better integrity check would have eliminated the initial need for a patch and therefore eliminated the second vulnerability entirely. Even with the first vulnerability a fully vetted patch with an emphasis on security would possibly have eliminated the need for a second patch.

As background I will give a brief history and description of Secure Shell and the two vulnerabilities listed above relating to the Secure Shell SSH1 protocol. Along with the reasons one might still be using the SSH1 protocol. For those forced to use SSH1 protocol the news that there is a new vulnerability is not a pleasant thought. Enduring patch after patch is enough to drive one to drink too many drinks filled with large amounts of caffeine.

**Background**

Secure Shell was written and introduced in 1995 by Tatu Ylonen of the University of Finland. What started as an academic project has blossomed into a well respected security enhancement. Secure Shell is mandatory for remote access in some instances.

In 1998 SSH Communications Security (www.ssh.com) commercialized SSH Secure Shell. Currently there are numerous flavors of Secure Shell available from various sources including an open source version from OpenSSH.org (www.openssh.org). Both SSH1 and SSH2 protocols are currently available. Although typically more recent versions of a protocol tend to have better security since they benefit from the incorporation of previous versions security fixes. All those that can should upgrade to the latest version of the Secure Shell protocol.

Secure Shell is commonly described on many FAQs as a drop in replacement for the Berkeley 'r' commands. The 'r' commands (rlogin, rsh, rcp) enable remote system communication and were first introduced in the BSD (Berkeley Standard Distribution) Unix operating system. The Secure Shell server listens on port 22/TCP and is not to be confused with telnet which runs on port 23/TCP.

The main benefits of using Secure Shell is end-to-end encryption and strong authentication. The end-to-end encryption is important so that no traffic passes over the network unencrypted and vulnerable to a sniffer. The strong authentication reduces the risk of use of impersonated credentials. Added benefits include automatic display setup for accessing the remote system and the ability to tunnel other service like X over Secure Shell. The ability to automatically or as many users have called "automagically" set the display is a nice feature for users that need to display graphics from a remote server on their local desktop. Instead of having to know how to modify the display setting Secure Shell does it automatically.

Secure Shell is considered a recommended addition to help improve remote security access. It has been adopted widely so a vulnerability could potentially be devastating.

**A vulnerability found. A patch is born.**

In June of 1998 Core-SDI released information on what they called the "ssh insertion attack." The attack is derived from an inherent weakness in the type of integrity checksum Secure Shell performs on packets. Specifically an attacker would have to craft a packet with his own data. The data portion the packet the attacker creates would need to produce a valid checksum field once the packet is decrypted. The attacker would then need to insert his own data packets into an existing Secure Shell client and server session. This is a complex attack which relies on the ability to compute a valid checksum field. In this case the crc32 (32 bit cyclic redundancy check) checksum for the attacker crafted packet. The valid checksum field makes the victim believe it to be valid. The attacker must also perform some sort of network attack to be able to inject his packets into the current session stream of Secure Shell.

From the descriptions of this vulnerability as a CVE (Common Vulnerabilities and Exposures) candidate at cve.mitre.org (CAN-1999-1085) and Core-SDI it seems to be a complex and difficult attack. I have greatly simplified the explanation here since the attack itself is not the main thrust of the paper but the mere fact that there is now a vulnerability in code that is widely accepted and used as a security enhancing piece of software. Core-SDI released a patch to detect and react to the attack since a direct fix would create a new version that would not be backward compatible with other SSH1 protocols. Many vendors adopted the patch. The patch was added to the Secure Shell source to protect against this kind of attack.

**Oops, the patch creates a vulnerability. Time for another patch.**

On February 8, 2001 over a year and a half after the patch from Core-SDI was released a new vulnerability with the Core-SDI patch was discovered. This new vulnerability was dubbed by a RAZOR Bindview Advisory the "Remote vulnerability in SSH daemon crc32 compensation attack detector". The Advisory is also listed on the CVE list as CVE-2001-0144. Many versions of Secure Shell that either use or support the SSH1 protocol were vulnerable. The problem is located in the detect_attack() function of the deattack.c file. Within the detect_attack() function a 16-bit variable is used in a place that could be assigned data from a 32-bit variable. The 16-bit variable could be "overflowed" and typically would end up having a value of zero. Based on the how the C programming language function that allocates memory (malloc()) reacts to the value placed in this 16-bit variable the attacker could gain access to change memory. The Secure Shell software could then interpret the attacker-changed memory as valid and perform desired duties for the attacker. The attacker could potentially use this to gain root. This new vulnerability gave rise to a new patch for the previous patch. The fix was essentially to change the 16-bit variable to a 32-bit variable in the source code.

The initial report from Bindview said that they were "not aware of working exploits for this vulnerability". However, as early as February 21, 2001 a working "SSH CRC-32 Compensation Attack Detector Vulnerability Exploit" was listed on Bugtraq at SecurityFocus.com. A detailed analysis of an attack which used an exploit to compromise a system which ran the vulnerable

SSH CRC-32 Compensation attack detector is available at The SANS Institute Incidents.org Handler's Diary from November 12, 2001. The attack described there also lists what happens to the system once a successful attack takes place. Trojans are installed and someone else owns the system. The notes from the Handler's Diary give good advice on what to look for on a questionable system and where to look for modified files from that particular attack.

The attack has moved fully from theoretical to useable for the attacker. Remember all of this was caused by using a 16-bit variable instead of a 32-bit variable which only became an issue because of the need for the previous patch.

**A relative first.**

Newsbytes reported a possible first in conjunction with the Secure Shell CRC-32 Compensation Attack Detector Vulnerability. In an October 18, 2001 article by Brian McWilliams, entitled "Hackers Put A Price Tag On New Attack Tool" the author discusses the possibility that some hackers were trying to sell copies of a new script to attack the Secure Shell CRC-32 Compensation Attack Detector Vulnerability. Since typically hackers trade exploits rather than take monetary compensation for them this would be a first according to what was reported. From a security practitioners point of view one would wonder how fantastically valuable it is to gain access into your company if someone is willing to spend money for that an exploit. Again to emphasize the need to reduce software security exposures and be more proactive when vetting software for security since we may never be able to eliminate software security holes entirely.

**Why use SSH1 protocol ?**

After reading all this discussion one might ask the question "Why not just upgrade and stop using the SSH1 protocol? As mentioned before the security team usually has to work within the organizational framework and can not always dictate and enforce what software is used and when upgrades occur. Discussions I've had with others that still use SSH1 protocol lead to software compatibility issues with Secure Shell. Some flavors of Kerberos may not be fully supported in some version other than the SSH1 protocol. Custom additions may have been added over time to build compatibility. This is mainly an issue of conversion time and could be undertaken given a reasonable changeover period. In some instances the more restrictive issue is the license. The SSH1 protocol has been in place for a long time and has the least restrictive license (pre-version 1.2.31). Most decisions are driven by monetary concerns in any organization. However, it would be sad to say that the whole organization may be compromised because it was too big a step to change from SSH1 protocol to the most recent protocol because of license restrictions. However, it may still be the only driving factor in some organizations.

**Helpful Tool**

The Center for Information Technology Integration at the University of Michigan has made a tool available called ScanSSH. The tool is part of a project "ScanSSH – Scanning the Internet for SSH Servers". The tool can be used by security team and attackers alike to find vulnerable running Secure Shell daemons. ScanSSH is available at http://www.monkey.org/~provos/scanssh. The tool scans IP addresses it is given for the

software version of the Secure Shell daemons. This information is helpful to identify which versions are vulnerable and which are safe. It scans very quickly so it is ideal for scanning large networks that run Secure Shell. For further information on the "ScanSSH – Scanning the Internet for SSH Servers" project please see the reference section.

**Conclusion**

Certainly there are other examples of vulnerabilities in secure shell. This was a simple double whammy example that helps to underscore the overall point. Many other examples of vulnerabilities that are simply software flaws are enough to frustrate even the person in the office with the least amount of caffeine in them. Attackers will keep coming from various angles. When the security team and system administrators attention is constantly focused on getting patches out something has to give. Other security measures may be diminished and suddenly an attacker has an opening.

As security practitioners we may wish for safer software. We can fight to get more security vetting in the software that is produced or as some have said "unleashed". If allowed we can help participate in teams that examine new software for security flaws. We can publicize fixes and share security knowledge. However, the reality is software is getting even more complex. Programming on large projects is a difficult task at best. Many times to a programmer simply getting the program to run is a major success. A programming language like C++ is supposed to enable the programmer to write better code as compared to the C programming language. It does not mean that the programmer will write better code or that the overall interaction of all the objects within the code will still be secure. Turnover and the constant demand for new features can lead to maintenance issues that lead to code that can not maintain any semblance of security. It is a wonder that the systems survive attack as long as they do. In the race to the marketplace software vetting for security concerns tend to be left behind and attackers get to do the vetting at the expense of the software user.

Since it may be impossible to fully understand every piece of code utilized on the system and have the time and resources to devote to completely reviewing and testing all patch code before it is placed on the system. Peer review of open source software. Open source software whose source code is put forth for public scrutiny may help force better software since attackers also have access to the source code. Having access to the source code can also help in vulnerability discovery and patch development. This reduces the lag time that can be incurred if the software user has to wait for a vendor to release a patch. Keeping abreast of the latest vulnerabilities is a next best practice to having better software. That way a vulnerability has the potential to be addressed before it is exploited.

**References**

1. Core SDI, "SSH1 CRC-32 compensation attack detector vulnerability", February 8, 2001, URL: http://www.core-sdi.com/pressroom/advisories_desplegado.php?idx=81&idxsection=10 (December 24, 2001)

2. Core SDI, "SSH INSERTION ATTACK", June 11, 1998,
URL: http://www.core-sdi.com/soft/ssh/ssh-advisory.txt (December 24, 2001)

3. Michal Zalewski, "Remote vulnerability in SSH daemon crc32 compensation attack detector", February 8, 2001,
URL: http://razor.bindview.com/publish/advisories/adv_ssh1crc.html (December 24, 2001)

4. Paul Festa, "Got hacked? Blame it on the software", November 30, 2001,
URL: http://www.zdnet.com/zdnn/stories/news/0,4586,2829102,00.html (December 24, 2001)

5. Provos, Niels, "ScanSSH – Scanning the Internet for SSH Servers", October 2, 2001, URL: http://www.citi.umich.edu/techreports/reports/citi-tr-01-13.pdf.

**More Reference**

Network ICE, Port Knowledgebase resource page,
URL: http://www.networkice.com/Advice/Exploits/Ports/22/default.htm

David J. Bianco, "Integer Overflow Attack Against SSH Version 1 Attack Detectors", March 1, 2001, URL: http://www.sans.org/infosecFAQ/encryption/integer.htm

CERT/CC, UNIX Security Checklist v2.0, "Section 3.3 Encryption and Strong Authentication", URL: http://www.cert.org/tech_tips/usc20.html

CVE, "CAN-1999-1085 SSH insertion attack", September, 9, 2001,
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-1085

CVE, "CVE-2001-0144 CORE SDI SSH1 CRC-32 compensation attack detector vulnerability", May 7, 2001, URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0144

CERT/CC Vulnerability Note VU#945216, "SSH CRC32 attack detection code contains remote integer overflow", October 24, 2001, URL: http://www.kb.cert.org/vuls/id/945216

CERT/CC, "CERT Advisory CA-2001-35 Recent Activity Against Secure Shell Daemons", December 13, 2001, URL: http://www.cert.org/advisories/CA-2001-35.html

Internet Security Systems, "SSH protocol 1.5 deattack.c allows memory to be overwritten", February 8, 2001, URL: http://xforce.iss.net/static/6083.php

Hugo Dias, "SSH CRC-32 Compensation Attack Detector Vulnerability Exploit", February 21, 2001, URL: http://www.securityfocus.com/archive/1/164133

Incident Handler, "SSH CRC32 Compensation Attack Detector Exploit Code Analyzed", November 12, 2001, URL: http://www.incidents.org/diary.php?id=16

Alex Salkever, Business Week, “Is Open-Source Security Software Safe?”, December 11, 2001, URL: http://www.securityfocus.com/news/297

The Secure Shell FAQ, URL: http://www.employees.org/~satch/ssh/faq/

Clement Clarke Moore, Attributed, “A Visit From St. Nick”, 1822, URL: http://www.nyise.org/moore/

Last Updated: June 24th, 2017

# Upcoming Training


| | | | |
|---|---|---|---|
| SANS Cyber Defence Canberra 2017 | Canberra, Australia | Jun 26, 2017 - Jul 08, 2017 | Live Event |
| SANS Columbia, MD 2017 | Columbia, MD | Jun 26, 2017 - Jul 01, 2017 | Live Event |
| SANS Paris 2017 | Paris, France | Jun 26, 2017 - Jul 01, 2017 | Live Event |
| SANS London July 2017 | London, United Kingdom | Jul 03, 2017 - Jul 08, 2017 | Live Event |
| Cyber Defence Japan 2017 | Tokyo, Japan | Jul 05, 2017 - Jul 15, 2017 | Live Event |
| SANS Munich Summer 2017 | Munich, Germany | Jul 10, 2017 - Jul 15, 2017 | Live Event |
| SANS Cyber Defence Singapore 2017 | Singapore, Singapore | Jul 10, 2017 - Jul 15, 2017 | Live Event |
| Community SANS Phoenix SEC401 | Phoenix, AZ | Jul 10, 2017 - Jul 15, 2017 | Community SANS |
| SANS Los Angeles - Long Beach 2017 | Long Beach, CA | Jul 10, 2017 - Jul 15, 2017 | Live Event |
| Mentor Session - SEC401 | Ventura, CA | Jul 12, 2017 - Sep 13, 2017 | Mentor |
| Mentor Session - SEC401 | Macon, GA | Jul 12, 2017 - Aug 23, 2017 | Mentor |
| Community SANS Atlanta SEC401 | Atlanta, GA | Jul 17, 2017 - Jul 22, 2017 | Community SANS |
| SANSFIRE 2017 | Washington, DC | Jul 22, 2017 - Jul 29, 2017 | Live Event |
| SANSFIRE 2017 - SEC401: Security Essentials Bootcamp Style | Washington, DC | Jul 24, 2017 - Jul 29, 2017 | vLive |
| Community SANS Fort Lauderdale SEC401 | Fort Lauderdale, FL | Jul 31, 2017 - Aug 05, 2017 | Community SANS |
| SANS San Antonio 2017 | San Antonio, TX | Aug 06, 2017 - Aug 11, 2017 | Live Event |
| SANS Boston 2017 | Boston, MA | Aug 07, 2017 - Aug 12, 2017 | Live Event |
| SANS Prague 2017 | Prague, Czech Republic | Aug 07, 2017 - Aug 12, 2017 | Live Event |
| Community SANS Omaha SEC401* | Omaha, NE | Aug 14, 2017 - Aug 19, 2017 | Community SANS |
| SANS Salt Lake City 2017 | Salt Lake City, UT | Aug 14, 2017 - Aug 19, 2017 | Live Event |
| SANS New York City 2017 | New York City, NY | Aug 14, 2017 - Aug 19, 2017 | Live Event |
| Community SANS Trenton SEC401 | Trenton, NJ | Aug 21, 2017 - Aug 26, 2017 | Community SANS |
| SANS Virginia Beach 2017 | Virginia Beach, VA | Aug 21, 2017 - Sep 01, 2017 | Live Event |
| Community SANS San Diego SEC401 | San Diego, CA | Aug 21, 2017 - Aug 26, 2017 | Community SANS |
| Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style | Virginia Beach, VA | Aug 21, 2017 - Aug 26, 2017 | vLive |
| SANS Adelaide 2017 | Adelaide, Australia | Aug 21, 2017 - Aug 26, 2017 | Live Event |
| SANS Chicago 2017 | Chicago, IL | Aug 21, 2017 - Aug 26, 2017 | Live Event |
| Mentor Session - SEC401 | Minneapolis, MN | Aug 29, 2017 - Oct 10, 2017 | Mentor |
| SANS Tampa - Clearwater 2017 | Clearwater, FL | Sep 05, 2017 - Sep 10, 2017 | Live Event |
| SANS San Francisco Fall 2017 | San Francisco, CA | Sep 05, 2017 - Sep 10, 2017 | Live Event |
| Mentor Session - SEC401 | Edmonton, AB | Sep 06, 2017 - Oct 18, 2017 | Mentor |