Global Information Assurance Certification Paper

Joanne_Pilker_GSEC.doc
Certification and Assignment: GSEC Version 1.4b
Title of Paper: MS SQL Slammer/Sapphire Worm

INTRODUCTION:

On Saturday, January 25, 2003 the Slammer Worm, also known as Sapphire, and SQL Hell began to spread across the internet infecting more than 90 percent of the vulnerable systems within 10 minutes and took many small networks offline with its scanning. Spreading hundreds of times faster than any other worm Slammer was ruthless and quick. It is a self propagating worm. It would locate a vulnerable system, infect it and than the infected system would begin to propagate. Slammer was doubling in size every 8.5 seconds and within 3 minutes it had scanned approximately 55 million IP addresses per second causing disruption of the internet [8]. It infected more than 75,000 hosts. This worm takes advantage of the buffer overflow vulnerability in Microsoft SQL Server 2000 and Microsoft SQL Server Desktop Engine 2000 (MSDE). The Microsoft SQL Server 2000 Desktop Engine (MSDE 2000) is a database engine built and based on core SQL Server technology. With support for single and dual-processor desktop computers, MSDE 2000 is a storage engine and query processor for desktop extensions of enterprise applications [13]. The Slammer Worm is known to be the fastest worm in history. It is important that we understand what this worm is, it's history, how is works, how to identify it, and the steps to resolve the problems it creates [2] [9].

WHAT IS A WORM?

A worm is a computer program that has the ability to copy itself. Worms are malicious self-propagating code. Worms move from system to system through the network using the network to infect each system by replicating itself. The worm can expand from a single copy quickly using the network.

A virus is a piece of software code that attaches itself to a real program. It is a piece of code embedded in a legitimate program. It piggybacks on top of a document or program. Once attached it spreads when the document or program is run or downloaded. The virus loads into memory and looks around for other programs in order to add its code to the program. When the new program executes the virus repeats this process infecting and reproducing itself over and over. A virus may attach itself to a game, someone downloads and runs the game, when executed the virus looks for more programs to attach to.

A worm is independent and able to propagate itself, but it needs the network in order to copy and expand to other systems. Unlike a virus, worms do not need to attach to a host program. A virus is dependent upon other programs to spread and infect other systems when the programs are executed.

WHAT IS THE SLAMMER WORM?

It is a memory resident worm that is designed to propagate, and does not appear to have any additional payload. The code is written exclusively in x86 assembly Slammer remains exclusively in the memory and makes no changes to any infected machine's system files. The code has a simple set of instructions that are no bigger than a short paragraph; it only contains 376 bytes with the following strings "h.dllhel32hkernQhounthickChGetTf", "hws2", "Qhsockf" and "toQhsend" [4] [5]. Analyzing Slammer's design reveals its knowledge of both computers and the internet. A technical description of the worm's source code is available [7].

HISTORY OF A WORM:

The first worm programs were designed to enable better usage of the network. In 1971 a program called a worm was written by Bob Thomas for the air traffic controllers. It notified operators when control of a certain airplane moved from one computer to another. This program was called "creeper" and it did not reproduce itself. It traveled from screen to screen displaying the message "I'm Creeper! Catch me if you can!".

In 1980 John Shock and Jon Hepps of Xerox created a worm to perform a variety of repetitive tasks. The worm was useful, but a dangerous tool if not used correctly. One evening the worm malfunctioned and systems began to crash. Xerox had to create an antivirus program to prevent the worm from crashing the systems.

On November of 1988 the Morris Internet worm was released and broke into about 6000 systems using a known Unix backdoor. Patches for the backdoor were available but had not been installed. Robert Morris Jr. a graduate at Cornell wrote this self-replicating worm and injected it into the Internet. Morris's worm infected multiple types of machines, attacked multiple security holes such as Unix sendmail program. The worm was replicating faster than anticipated and Morris wrote instructions for killing the worm and to prevent reinfection but was unable to send the program because the network was clogged. A team at Berkeley provided steps that would retard the spread of the worm. After a few days things returned to normal and everyone wanted know who did this. Evidence pointed to Robert T. Morris, he was convicted, sentenced to three years probation, 400 hours community service and fined $10,050 and lost his supervisor position [17].

On March 1999 the Melissa worm appeared. This macro virus is propagated by email. It infected machines with Microsoft Word 97 and Word 2000. The virus resides in the attachment of an email and when a user opens the attachment it looks through all Outlook address books coping itself to the first 50 individuals. The worm spread quickly and soon overwhelmed some email servers [18].

On May 8, 2000 the “Love Letter” worm spread to more than 500,000 individual systems. Network degradation was apparent at several sites from the mail, file and web traffic generated by the “Love Letter” worm. This worm infects in several ways to include electronic mail, Windows file sharing, IRC, USENET news and web pages. The worm attempts to copy itself using Microsoft Outlook to all entries in all the address books when it executes. Some characteristics of the worm are subject “ILOVEYOU”, attachment named “LOVE-LETTER-FOR-YOU.TXT.VBS” and the body reads “kindly check the attached LOVELETTER coming from me [19].

The next major worm is the Code Red. It is self-propagating malicious code that exploits a known vulnerability in Microsoft’s Internet Information Server. The buffer overflow vulnerability allows the worm to execute code within the IIS Server. It achieved complete infection in about twelve hours [20].

The last worm is the Nimda worm known as W32/Nimda worm and it is propagated through email. It is a mass mailing worm that uses multiple methods to spread itself. It sends itself out by email, searches for open network shares, and attempts to copy it to unpatched Microsoft IIS web servers [21].

SLAMMER’S HISTORY:

Slammer launched Saturday, January 25, 2003 about 12:30 AM (EST); it spread across the Internet in a matter of hours. Slammer owes its speed to an Internet protocol UDP that is light and fast. The majority of systems were infected within 10 minutes; about five of the thirteen major hubs were targeted. The worm was so fast it attacked the entire internet all at once creating a major denial of service. A denial of service is an explicit attempt to prevent users of a service from utilizing that service. The impact of a denial of service attack can disable the network and consume scarce resources. In addition to network bandwidth, intruders may be able to consume other resources that your systems need in order to operate. For example, in many systems, a limited number of data structures are available to hold process information (process identifiers, process table entries, process slots, etc.). An intruder may be able to consume these data structures by writing a simple program or script that does nothing but repeatedly create copies of itself [15]. Initially the first thought as the denial of service progressed was to block port 1434 as a preventive measure until further analysis and investigative forensics could be completed and a better approach implemented. Slammer reached its full scanning rate in about three minutes, but its rate slowed down due to the fact that a major portion of the internet did not have enough bandwidth to support it. Although Slammer is a very simple worm it is a major milestone in the evolution of computer worms. The prospect of further development of this worm is inevitable [3].

HOW SLAMMER WORKS:

Slammer's method of spreading is by random scanning IP addresses to infect, and finding all vulnerable hosts. The scanning produces a large number of packets aimed at User Datagram Protocol (UDP) port 1434, an older and less secure protocol. It spreads using a buffer overflow; a well known flaw in Microsoft SQL Server and MSDE 2000 and related packages. The overflow occurs in a DLL (ssnetlib.dll) implementing the SQL Server Resolution Service. When a data string is written into memory and its length is not checked by the program this causes a buffer overflow. The program's own code is overwritten by the tail end of the data, if the string is too long [4]. This vulnerability was exploited using a single packet to UDP port 1434. SQL Server establishes a listener service the proxy between the client and the database and uses the UDP port 1434 to ensure that you can correctly access and browse named instances. It's a simple, datagram-oriented, transport layer protocol used for Web sites, email and file downloads. For each output operation by a process only one UDP datagram is produced and this causes one IP datagram to be sent. It is not reliable and sends the datagrams the application writes to the IP layer with no guarantee that they will reach their destination. A UDP packet does not require a handshake before exchanging information. It does not require sender and receiver to acknowledge each other; it carries a message in a single, one-way packet [10]. Once machines are infected by the worm they randomly scan IP addresses and eventually find vulnerable systems to infect. Machines infected by the worm overwhelmed the network with its scanning, infecting, and the newly infected machines scanning again.

The scanning attack on all the IP addresses (over 55 million scans per second) caused a denial of service [8]. There are several types of denial of service attacks, attempts to flood the network, disrupt connections between two systems, to prevent an individual from accessing a service and disrupt service of a specific system. Slammer definitely caused a denial of service by flooding the network. The worms scanning speed was limited by the bandwidth available to uninfected host. Bandwidth consumption is when an intruder generates a large number of packets directed at your network. In a matter of just a few hours, it shut down Bank of America Corp. ATMs, cancellation of airline flights, and blacked out an emergency call center in Seattle. [11] The following will show a step by step flow of the execution of the worm:

STEP ONE

Get Inside

Slammer masquerades as a single UDP packet, one that would normally be a harmless request to find a specific database service. The first byte in the string - 04 - tells SQL Server that the data

following it is the name of the online database being sought. Microsoft's tech specs dictate that this name be at most 16 bytes long and end in a telltale 00. But in the Slammer packet, the bytes run on, craftily coded so there is no 00 among them. As a result, the SQL software pastes the whole thing into memory

STEP TWO

Reprogram the Machine

The initial string of 01 characters spills past the 128 bytes of memory reserved for the SQL Server request and into the computer's stack next door. "Stack" is programmer-speak for an orderly list of information the computer shuffles to remind itself what to do next, like tidy paperwork on a desk. The first thing the computer does after opening Slammer's too-long UDP "request" is overwrite its own stack with new instructions that Slammer has disguised as a routine query. The computer reprograms itself without realizing it.

STEP THREE

Choose Victims at Random

Slammer generates a random IP address, targeting another computer that could be anywhere on the Internet. To randomize, Slammer deploys a time-honored programmer's trick: It looks up the number of milliseconds that have elapsed on the CPU's system clock since it was booted and interprets the number as an IP address.

STEP FOUR

Replicate

The envelope is addressed, now it just needs to be stuffed. Slammer points to its own code as the data to send. The infected computer writes out a new copy of the worm and licks the UDP stamp.

STEP FIVE

Repeat

After sending off the first tainted packet, Slammer loops around immediately to send another to a different computer. It doesn't

> waste a single millisecond. Instead of making another call to the system clock to get the time, it just shuffles the bits of the IP address already in memory to create a new one. Slammer's one bug is buried here: The reshuffling leaves a few digits in the address unchanged. It hardly matters, though, since the computer is now spewing packets as fast as its network cable can carry them away. A home PC could cram a couple hundred copies onto its broadband link every second. Corporate data centers became nasty breeding grounds, launching tens of thousands per second. Slammer commandeered just 75,000 SQL machines. But because it replicated so fast, the worm was able to take down millions more, kicking them offline with a flood of meaningless traffic [4].

The Slammer worm lost momentum because the scanning was so aggressive that it diminished its growth. Newly infected systems had to compete with other infected systems for scarce bandwidth.

HOW TO IDENTIFY THE WORM:

Identification of the worm took place through three separate entities:

The first being sheer volume which reflected itself in the initial stages of the infection whereas 75, 000 known hosts were attacked. There was probably more infected host causing major disruptions in network reliability.

The second method of identification was that all UDP connections attacked port 1434 causing each attacked host to spend so much time on port 1434 that its services were degraded.

The third area of identification was by the size of the packet. Each UDP packet sent from the hostile to the target was 376 bytes [6].

Below is an actual output from a tcpdump that collected data from the Slammer Worm attack. This was collected from commercially available Intrusion Detection Systems Software. This dump reflects how an infected system scans other systems to find vulnerable host targeting UDP port 1434. It also reflects the size of the packet 376 bytes.

Incident Details:

15:01:49.340254 128.xx.xx.xx.2679 > 216.xxx.xx.xx.1434: udp 376
15:01:49.340395 128.xx.xx.xx.2679 > 24.xxx.xx.xxx.1434: udp 376
15:01:49.340826 128.xx.xx.xx.2679 > 216.xxx.xxx.xxx.1434: udp 376
15:01:49.340829 128.xx.xx.xx.2679 > 152.xxx.xxx.xxx.1434: udp 376

15:01:49.340835 128.xx.xx.xx.2679 > 24.xx.xxx.xxx.1434: udp 376

RESOLVING THE PROBLEMS SLAMMER CREATED:

The first course of action was to block inbound and outbound traffic destined for UDP port 1434 to help reduce the impact of the worm.

Rebooting the system will remove the worm from memory. The system must be patched or it will be re-infected.

This is a known vulnerability; Microsoft has had a patch available since July 2002. Microsoft has released a recent service pack (Service Pack 3) that will fix this vulnerability. Download and apply Service Pack 3 from Microsoft then reboot the server [5].

PREVENTIVE MEASURES:

Snort is a lightweight network intrusion detection system, capable of performing near real-time traffic analysis and packet logging on IP networks. Using this intrusion detection system, an effective Snort rule can be applied to detect the signature of the slammer worm. It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more. Snort uses flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes a modular plugin architecture. Snort has a near real-time alerting capability as well, incorporating alerting mechanisms for syslog, a user specified file, a UNIX socket, or WinPopup messages to Windows clients using Samba's client [16]. Using Snort to identify previous worm attacks or future attacks will help to retain the integrity of your network. Apply this Snort rule it will help to detect the Slammer Worm. Below is a copy of the rule [12].

SNORT rule (by Chris Brenton):

```
alert udp $EXTERNAL_NET any -> $HOME_NET 1434 (msg:"SQL
Sapphire Worm";
dsize:>300; content: "|726e 5168 6f75 6e74 6869 636b 4368 4765|";
offset: 150; depth: 75;)
```

Most personal computers are unlikely to be vulnerable to the Slammer worm, most personal computers do not run Microsoft SQL Server. A quick way to ensure you are not running Microsoft SQL Server is to enter the following command:[1]

At the MS-DOS prompt window:

netstat –an find “1434”

After entering the command and if lines containing 1434 are printed to the screen your system may be vulnerable.

Apply latest patches and service packs available from Microsoft to all systems.

Know what applications are installed on your systems. Some administrators are not aware MDSE 2000 is installed on their system it is often incorporated into other Microsoft software products and third-party applications such as medical imaging software used to catalog MRI and CAT scans [14]. MSDE is installed as part of the following:

* SQL Server 2000 (Developer, Standard, and Enterprise Editions)
* Visual Studio .NET (Architect, Developer, and Professional Editions)
* ASP.NET Web Matrix Tool
* Office XP (various versions)
* MSDN (various subscription levels)
* Access 2002
* Visual FoxPro 7.0/8.0

For a complete list, see:
http://www.sqlsecurity.com/DesktopDefault.aspx?tabindex=10&tabid=13

To determine if your system is not running the vulnerable code Microsoft recommends the following steps on Windows XP and 2000 systems:

Right-click on the My Computer icon;
Select Manage;
Double-Click on Services; and
Double-Click Services.

If you see items beginning with the string "MSSQL" in the list of running services, MSDE is running [14].

Run the most current anti-virus software throughout the network.

Disable any unused or unneeded network services.

Routinely examine physical security of servers, routers, etc.

Observe system performance-unusual levels of disk and network activity.

Use of a firewall will help manage all internet access.

Establish password policies.

Filtering traffic may prevent infected systems from compromise other systems and can help reduce the strain on individual systems.

CONCLUSION:

It is possible this could have been prevented if the patch that was released in July 2002 was applied. The worm was easily distinguishable with its scanning, its destination of 1434 port/udp and the presence of 376 bytes UDP packet. If port 1434 is not needed recommendation is to block this port. Slammer negatively impacted the Internet by causing poor network performance. However within our monitored network environment that this paper was based on the actual severity of the attack was greatly reduced due to instantaneous action to block port 1434. High speed worms had been predicted but with the Slammer attack it is no longer just a theory but a reality. The high speed worm presents problems for system administrators; there is no way for them to respond to a worm of this speed. As of today no one has laid claim to the Slammer Worm attack and to acknowledge development of the code. We do not know when and if it will happen again. The total cost for lost revenue is more than $1 billion [4].

[1] Worm Watch 2003 “SQL Sapphire/SQL-Hell Slammer Worm News” 25Jan2003
URL: http://grc.com/worms/25-01-03.htm

[2] “The Spread of the Sapphire/Slammer Worm”
URL: http://www.caida.org/outreach/papers/2003/sapphire/sapphire.html

[3] Analysis: SQL Slammer “Advisory; SQL slammer” 26Jan2003
URL: http://www.robertgraham.com/journal/030126-sqlslammer.html

[4] Wired Issue 11.07 “Slammed!” July2003
URL: http://www.wired.com/wired/archive/11.07/slammer_pr.html

[5] McAfee Security – Virus Information Library “Virus Profile” 30Jan2003, 4Mar2003
URL: http://www.vil.mcafee.com/dispVirus.asp?virus_k=99992

[6] Symantec Security Response “W32.SQLExp.Worm” 4Feb2003
URL: http://www/symantec.com/avcenter/venc/data/w32.sqlexp.worm.html

[7] Symantec DeepSight Threat Management System Threat Analysis “SQLExp SQL Server Worm Analysis” 28Jan2004
URL: http://securityresponse.symantec.com/avcenter/Analysis-SQLEXP.pdf

[8] vnunet com Bugwatch “Son of Slammer on the cards” 18Jun2003
URL: http://www.vnunet.com/News/1141694

{9} SANS “Critical Vulnerability Analysis(CVa) –Volume 2,Issue# “MS-SQL Server Worm”
http://www.sans.org/newsletters/cva/vol2_3.php

[10]W. Richard Stevens. TCP/IP The Protocols Volume ! . Addison-Wesley, April 2000 (pages 143-145)

[11] Washingtonpost.com, TechNews.com, “Internet Worm Unearths New Holes” 6Jun2003
http://www.washingtonpost.com/ac2/wp-dyn?pagename=article&...

[12] Internet Storm Center, “Analysis Port 1434 MS-SQL Worm” 25Jan2003
http://isc.incidents.org/

[13] Microsoft SQL Server, “MSDE 2000 Product Overview” 02Jun2003
http://www.mictosoft.com/sql/msde/productinfo/overview.asp

[14] Extreme Tech, “SQL Slammer” Can Affect Other Microsoft Software 29Jan2003
http://www.extremetech.com/article2/0,3973,852194,00.asp

[15] CERT Coordination Center “Denial of Service Attacks”
http://www.cert.org/tech_tips/denial_of_service.html

[16] Datanerds “Snort” 14Nov2002
http://www.datanerds.net/~mike/snort.html

[17] Massachusetts Institute of Technology, “The Robert Morris Internet Worm”25AUG1995
http://www.swiss.ai.mit.edu/6805/articles/morris-worm.html

[18] CERT Coordination Center, CERT Advisory CA-1999-04 “Melissa Macro Virus 17Mar1999
http://www.cert.org/advisories/CA-1999-04.html

[19] CERT Coordination Center, CERT Advisory CA-2000-04 “Love Letter Worm 4MAY2000
http://www.cert.org/advisories/CA-2000-04.html

[20] CERT Coordination Center, CERT Advisory CA-2001-19 "Code Red" Worm Exploiting Buffer Overflow In IIS Indexing Service DLL 19Jul2001
http://www.cert.org/advisories/CA-2001-19.html

[21] CERT Coordination Center, CERT Advisory CA-2001-26 “Nimda Worm” 18Sep2001
http://www.cert.org/advisories/CA-2001-26.html

Last Updated: May 26th, 2017

Upcoming Training

SANS Stockholm 2017	Stockholm, Sweden	May 29, 2017 - Jun 03, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 2017	Live Event
Community SANS Ottawa SEC401	Ottawa, ON	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS San Francisco Summer 2017	San Francisco, CA	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS Charlotte 2017	Charlotte, NC	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Rocky Mountain 2017 - SEC401: Security Essentials Bootcamp Style	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
SANS Secure Europe 2017	Amsterdam, Netherlands	Jun 12, 2017 - Jun 20, 2017	Live Event
Community SANS Portland SEC401	Portland, OR	Jun 12, 2017 - Jun 17, 2017	Community SANS
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Minneapolis 2017	Minneapolis, MN	Jun 19, 2017 - Jun 24, 2017	Live Event
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Cyber Defence Canberra 2017	Canberra, Australia	Jun 26, 2017 - Jul 08, 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 Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Minneapolis SEC401	Minneapolis, MN	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
Community SANS Phoenix SEC401	Phoenix, AZ	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Munich Summer 2017	Munich, Germany	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 Colorado Springs SEC401	Colorado Springs, CO	Jul 17, 2017 - Jul 22, 2017	Community SANS
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 Charleston SEC401	Charleston, SC	Jul 24, 2017 - Jul 29, 2017	Community SANS
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 Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event