新型蠕虫病毒—“SQL杀手”的分析和防范

The Analysis and Defense of Microsoft SQL Sapphire Worm

谭毓安（北京理工大学计算机科学与工程系 100081）

摘 要：“SQL Slammer”蠕虫在Windows系统中通过网络进行传播。该蠕虫利用Microsoft SQL Server 2000的缓冲区溢出漏洞获得系统控制权。蠕虫产生大量随机IP地址进行攻击，导致蠕虫的迅速传播并且形成整个互联网范围内的拒绝服务攻击，网络带宽大量被占用。本文讨论了其运行机制以及检测和清除方法。

关键词：蠕虫 病毒 缓冲区溢出 Windows

1 引言

2003年1月底，互联网上出现一种新型的蠕虫病毒，大量占用网络带宽，最终导致网络瘫痪。该蠕虫是利用Microsoft SQL Server 2000（以下称SQL Server）中的缓冲区溢出漏洞，通过向其解析端口1434发送包含恶意代码的数据包进行攻击。

由于“SQL Slammer”蠕虫具有极强的传播能力，造成了全球性的网络灾害。此次攻击是自“红色代码”蠕虫以来Internet遭受到的最严重攻击。这种蠕虫也被称为“强风”、“蓝宝石”、“2003蠕虫王”或“SQL地狱”。其攻击手段与2001年夏天肆虐全球的“红色代码”蠕虫非常类似，二者都是利用微软操作系统的漏洞向目标主机发送网络包，导致缓冲区溢出，并执行攻击代码。据统计，“SQL Slammer”爆发初期，仅需8.5秒被感染的主机数量就增加1倍，而“红色代码”需要37分钟。

2 “SQL杀手”病毒的特点

和“红色代码”一样，“SQL Slammer”也不将蠕虫信息写入被传染对象的文件中。它只存在于内存之中，不通过文件这一常规载体传染和存储，而是借助这个服务器的网络连接来传染其他的服务器，在计算机的内存之间不断进行复制。从传染途径上看，该蠕虫在运行SQL Server的服务器之间通过UDP 1434端口发送SQL请求进行传播。

除了运行SQL Server的Windows NT/2000系列服务器外，安装了Visual FoxPro、Backup Exec等其他软件的服务器也会被感染，因为这些软件中包含有Microsoft Data Engine 2000(微软数据库引擎)，而SQL Server内嵌在该引擎中。

“SQL Slammer”通过缓冲区溢出取得系统控制权后，就开始产生随机IP地址发送自身。由于发送数据包占用了大量系统资源和网络带宽，形成UDP风暴，感染了该蠕虫的网络性能会急剧下降，另外，蠕虫的扩散占用了整个Internet网络上的大量带宽。因此，形成了全球范围内的拒绝服务攻击。在高峰时，每秒钟产生6千多万个网络包。随着蠕虫的爆发，网络管理者在主机和路由器上采取措施将目标端口为1434的UDP网络包全部阻塞，因此攻击网络包的数量呈逐步下降趋势。

3 传染过程

“SQL Slammer”利用了微软公司SQL Server的一个漏洞（公告编号: MS02-039）[1]。该漏洞是在2002年7月由Next Generation Security公司发现的[2]。

该蠕虫利用的端口是UDP 1434，该端口提供SQL Server 解析服务。SQL Server支持在单一物理主机上运行多个SQL服务器的实例，但是多个实例不能使用同一个SQL标准服务会话端口(TCP 1433)，所以SQL Server解析服务监听UDP 1434端口来提供一种查询机制，向客户端返回各SQL服务实例对应的网络端口。

SQL Server解析服务在UDP 1434端口接收UDP包。包的第一个字节为0x04时，SQL监视线程会使用UDP包后面的信息来尝试打开注册表中的一个键值。例如，接收到的UDP数据包是\x04\x41\x41\x41，SQL服务程序就调用sprintf生成如下格式的字符串：

“HKLM\Software\Microsoft\Microsoft SQL Server\AAA\MSSQLServer\CurrentVersion”。再作为注册键读取有关参数。

由于SQL Server 解析服务程序在调用sprintf时没有进行长度检查，如果UDP数据包字节 \ x04后面的字符串超过一定长度后，将产生缓冲区溢出。“SQL Slammer”就是利用这一漏洞，通过在这个UDP包后追加包含攻击代码的额外数据，当解析服务程序调用sprintf时，会发生基于堆栈的缓冲区溢出。它使用0x42B0C9DC覆盖了ssnetlib.dll中一个函数的返回地址，该返回地址（0x42B0C9DC）指向sqlsort.dll中数据区中的“FF E4”字节，而“FF E4”正好是“JMP ESP”这条指令的机器码。缓冲区溢出时内存布局如图1所示。

42B0C9DC	FF E4 “JMP ESP”
缓冲区HKLM \ Software \ Microsoft \ Microsoft	
	SQL Server \
	04 01 01 01 01 01 01 01 01 01 01 01 01 01
函数返回地址	42B0C9DC
	EB 0E 01 01 01 01 01 01 01 70 AE 42 01
	\ MSSQLServer \ CurrentVersion

图 1 “SQL Slammer”攻击成功时的内存布局

“SQL Slammer”蠕虫的长度极小，仅仅是一段376个字节(1+96+4+275=376)的数据。而“红色代码”的长度约为4KB，“尼姆达”蠕虫的长度达到60KB。另外，“SQL Slammer”通过UDP端口进行传播，不象“红色代码”那样需要建立TCP连接。因此，其传播速度比“红色代码”要快。图2列出了“SQL Slammer”的完整数据。

```
0000  04010101 01010101 01010101 01010101  ................
0010  01010101 01010101 01010101 01010101  ................
0020  01010101 01010101 01010101 01010101  ................
0030  01010101 01010101 01010101 01010101  ................
0040  01010101 01010101 01010101 01010101  ................
0050  01010101 01010101 01010101 01010101  ................
0060  01DCC9B0 42EB0E01 01010101 010170AE  ....B.........p.
0070  420170AE 42909090 90909090 9068DCC9  B.p.B........h..
0080  B042B801 01010131 C9B11850 E2FD3501  .B.....1...Pay5.
0090  01010550 89E55168 2E646C6C 68656C33  ...P..Qh.dllhel3
00a0  32686B65 726E5168 6F756E74 6869636B  2hkernQhounthick
00b0  43684765 745466B9 6C6C5168 33322E64  ChGetTf1llQh32.d
00c0  68777332 5F66B965 74516873 6F636B66  hws2_f1etQhsockf
00d0  B9746F51 6873656E 64BE1810 AE428D45  1toQhsend....B.E
00e0  D450FF16 508D45E0 508D45F0 50FF1650  .P..P.E.P.EeP..P
00f0  BE1010AE 428B1E8B 033D558B EC517405  ....B....=U..Qt.
0100  BE1C10AE 42FF16FF D031C951 515081F1  ....B...D1.QQP..
0110  0301049B 81F10101 0101518D 45CC508B  ..........Q.E...
0130  C4508B45 C050FF16 89C609DB 81F33C61  .P.E.P.........a
0150  C1E20829 C28D0490 01D88945 B46A108D  .a.).......E...j
0160  45B05031 C9516681 F1780151 8D450350  E.P1.Qf..x.Q.E.P
0170  8B45AC50 FFD6EBCA                    .E.P....
```

图 2 “SQL Slammer”网络包

4 运行过程

“SQL Slammer”通过覆盖堆栈中的返回地址，“RET”指令不能返回到原来的调用者，而是执行0x42B0C9DC处的“JMP ESP”指令，而ESP指向的正是“SQL Slammer”蠕虫偏移0x65字节处的运行代码。这些程序以SQL Server进程的权限在系统中执行[3]。其运行过程为：

（1）将0x42B0C9DC压栈，再将0x01010101压栈24次，最后将0x04000000压栈。

（2）执行第1步后，将ESP保存到EBP中，而EBP+3此时指向“SQL Slammer”的完整数据(图2)，在第8步中作为网络数据包被发送出去。

（3）在sqlsort.dll的IAT表中取得LoadLibrary函数的地址。

（4）调用LoadLibrary和GetProcAddress 得到3个函数的入口地址：kernel32.dll中的GetTickCount函数、ws2_32.dll中的socket和sendto函数。

（5）使用GetTickCount获得系统启动后经历的毫秒数。

（6）调用socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)创建一个套接字。

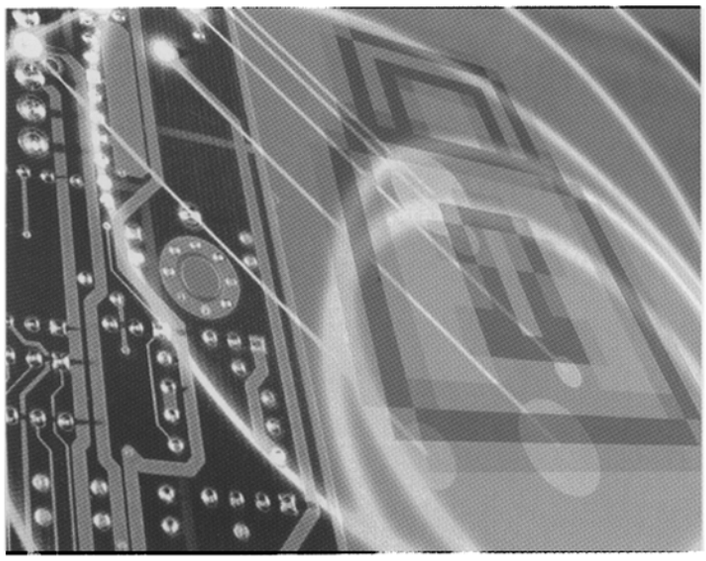
（7）以毫秒数为依据生成一个随机的IP地址。

（8）以该IP地址为目标，调用sendto发送SQL解析请求UDP网络包。目标端口为1434，缓冲区地址为EBP+3，长度为376字节。因此，如果目标服务器上也运行了SQL Server服务，就会将这个UDP网络包作为SQL Server解析请求，从而被蠕虫入侵。目标机被入侵之后，目标机再次重复“SQL Slammer”的运行过程，在网络上继续扩散。

（9）跳转到第7步。

系统被感染后，在第7步至第9步进入一个无限循环，不断在网络上扩散蠕虫代码。由于UDP不需要与目标主机建立TCP连接，在一个100Mbps Internet出口的主机，每秒可重复发送攻击包3万个。

数据中包括两个“01 70 AE 42”序列，表示偏移地址0x42AE7001。这是因为sprintf所引用的缓冲区后面还有数据指针，“SQL Slammer”在发送攻击代码时，也会同时覆盖这个数据指针。如果数据指针指向了一个非法地址，函数访问这个指针时就会出现异



常，而被异常处理程序捕获，函数不可能正常返回，也就无法执行0x42B0C9DC处的指令了。因此，攻击数据包中的“01 70 AE 42”的作用是为了防止出现异常。

5 “SQL 杀手”病毒的检测和防范

“SQL 杀手”病毒的传染对象是运行SQL Server的Windows系统。根据其运行特点，有以下几种手段来检查系统中是否“SQL杀手”病毒：

（1）在任务管理器的进程列表中查找是否有sqlservr.exe进程，或者检查系统中是否安装有“SQL Server”服务。

（2）检查是否出现负载显著增加（CPU/网络）的现象。

（3）可用“netstat -a -p udp 100”命令检查是否存在对许多外部IP地址的1434端口的连接。

（4）启动网络监视程序（如Sniffer Pro）进行检查，查找是否有向目的端口为UDP 1434发送大量数据的主机。

（5）使用各安全厂商提供的监测工具。

为防止系统被“SQL 杀手”病毒感染，有以下两种措施：

① 下载并安装针对该漏洞的微软补丁。可以安装Microsoft SQL Server 2000 SP3来升级SQL Server，或者安装编号为40602的补丁。

② 在边界防火墙或者路由器上阻塞外部对内和内部对外的UDP/1434端口的访问。

在发现系统被病毒感染后，由于该病毒并没有修改系统中的任何文件或数据，因此不需要专门的清除工具。其清除过程为：

- 切断被感染主机的网络连接，防止在清除病毒的过程中再次受到攻击。
- 重新启动系统。之后，内存中的“SQL 杀手”病毒不复存在。
- 下载并安装针对该漏洞的微软补丁。
- 恢复被感染主机的网络连接。

6 结束语

2001年夏天出现的红色代码蠕虫[4]，利用了IIS的漏洞造成网络阻塞，而早在红色代码病毒发作前几个月微软就发布了此漏洞的补丁程序。如果IIS服务器的系统管理员及时安装了补丁程序，红色代码病毒根本就不会在全世界范围内发作。这次“SQL 杀手”病毒的状况与红色代码有着惊人的相似。微软在2002年7月得知这一漏洞后立即进行了公告并发布了相应的补丁，在这之后长达6个月的时间中，安装SQL Server的系统并没有及时进行升级或修补，给造成“SQL 杀手”造成了可乘之机。另外，许多系统并没有直接安装SQL Server数据库软件，而是由于系统中安装的其他软件（如Veritas备份工具）中默认安装了SQL数据库引擎，从而也受到“SQL 杀手”的攻击，而用户本身并没有意识到系统中存在SQL Server的运行程序。

以红色代码、求职信和“SQL 杀手”等为代表的恶意程序的出现，预示了未来病毒的一个发展趋势：针对主流操作系统及其关键应用（如Internet服务器、数据库服务器、邮件服务器、代理服务器等）的漏洞、通过Internet进行传播，形成拒绝服务(DoS)攻击造成网络瘫痪。这些恶意程序显示出病毒编制者的“高超”水平及他们对系统漏洞的极度关注。

“SQL 杀手”的大规模爆发促使SQL Server用户去安装SP3以弥补此漏洞，因此该蠕虫的影响已经迅速减弱。但操作系统和其他关键应用中仍然存在其他的一些未知漏洞，会不断地被发现和修补。因此，根本的解决办法是：从厂商的角度，在软件开发过程中要严格遵循安全标准，最大限度地杜绝缓冲区溢出等漏洞的出现；从用户的角度，应该及时关注软件的安全漏洞并采取相应的措施。采取预防措施所花费的精力和代价要比遭受攻击后再进行补救要小得多，正所谓“防患于未然”。

参考文献

1 http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS02-039.asp。

2 http://www.ngssoftware.com/vna/ms-sql.txt。

3 http://www.eeye.com/html/Research/Flash/sapphire.txt。

4 谭毓安，新型网络病毒—红色代码病毒的分析和防范，计算机系统应用，2002 年第 2 期。