Analysis of the "SQL Slammer" worm and its effects on Indiana University and related institutions

Gregory Travis
Ed Balas
David Ripley
Steven Wallace

Advanced Network Management Lab
Cybersecurity Initiative
Indiana University
Bloomington, Indiana

INTRODUCTION

On November 2nd 1988 Robert Morris, then a Cornell University computer science graduate student, released the first Internet worm. Morris's Worm, as it was known, exploited known flaws in the finger and sendmail services as well as in common webs of trust inherent in the rlogin facility. The worm's only activity was that of replicating itself to as many hosts as possible. Towards that end, the worm searched local files (such as /etc/hosts) to identify machines to infect as well as scanning likely addresses in the local network. The worm did not damage files or otherwise disrupt operation of the infected machines; however the traffic volume generated by its replication attempts severely disrupted the global Internet, local enterprise networks, and the processing ability of the infected machines themselves. The Morris worm infected roughly 10 percent of Internet computers and cost an estimated 100 million dollars (156 million in 2003 dollars) to clean up.

Like the Morris worm, Slammer's only disruptive activity was the traffic associated with its replication. SQL Slammer infected less than one in a thousand Internet computers, but its effect was much more dramatic. Slammer targeted random hosts, which is relatively inefficient, however a Slammer infected computer would try as many as 25,000 target addresses a second. The simplicity of the infection method, which required only a single packet to infect a vulnerable computer and, like Morris, exploited a known vulnerability, combined with the speed at which potential computers where probed, allowed Slammer to reach global proportions in less than eight minutes (it doubled in size every 8 seconds). Current estimates put the cost of Slammer at approximately one billion dollars – an order of magnitude more expensive than the Morris worm in constant dollars.

Fifteen years after the first worm the Internet has grown from 60,000 computers, based mostly in higher education and research facilities, to its current 200,000,000 computers permeating all aspect of information technology. Despite the Internet becoming part of the critical information technology for business and government world-wide, a single packet just 376 characters in length targeted at a well known and preventable vulnerability, caused a global information disruption. Unfortunately, the fundamental vulnerability of the Internet to these types of attacks has not changed but our reliance on the Internet as a trusted information technology infrastructure for business, for defense, for crisis management, for healthcare etc. has.

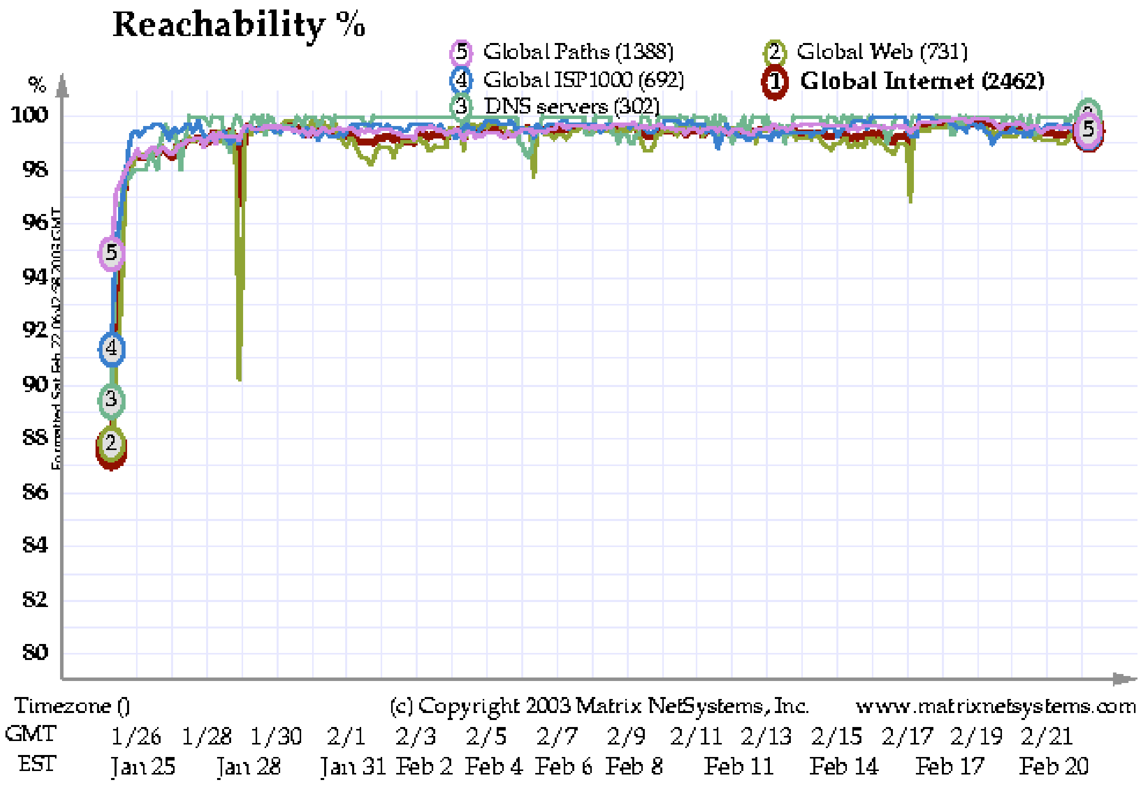
The Internet has remained a best effort network with end-to-end transparency. While these attributes have supported the innovation of applications and the ability to cost-effectively grow the Internet from heterogeneous parts, it also represents its Achilles heal. Requiring individual computers to be free from Slammer-like vulnerabilities as a prerequisite for the secure and reliable functioning of the Internet is simply unattainable. For the Internet to evolve into a more robust and dependable communications infrastructure, both of these attributes (best effort and end-to-end transparency) will need modification.

TIMELINE

On Saturday, January 25th, at approximately 12:30AM (EST), a computer worm was injected into the global Internet. The exact point, or points, of injection remain unknown but it is known that the worm is in the form of a 376-byte User Datagram Protocol (UDP) packet destined for port 1434 on a candidate machine. Candidate machines are those machines running an un-patched version of Microsoft's SQL Server software or various Microsoft and other vendor products which incorporate SQL Server technology. Once infected candidate machines rapidly (see measurement section below) begin issuing new infection packets randomly targeted at other potential candidate hosts within the 32-bit IPv4 IP address range. The worm is colloquially referred to by various names including SQL Slammer, Sapphire, and SQL Hell. Throughout this document we will refer to it as SQL Slammer.

The primary characteristic of the worm is its extraordinary rate of propagation. It is estimated that it reached its full level of global internet infection within ten minutes of release. At its maximum (reached on Sunday, January 26th) approximately 120,000 individual computers worldwide were infected and those computers generated an aggregate of over 1 terabit/second of infection traffic (ANML estimate).

Figure 1, Global Internet Reachability (courtesy Matrix Systems)



At the point of maximum infection traffic the worm caused a loss of approximately 15% of hosts on the internet where loss is defined by a lack of reachability to the host (see figure 1 above). This loss was attributed both to network overload due to data as well as specific infrastructure failures (see below).

Response to the worm was rapid with restoration to approximately 98% reachability by noon EST on the 25th. This recovery was made possible through two primary approaches, first the firewalling of port 1433/1434 at institutional and organizational borders as well as the direct physical disconnect of potential candidate hosts. Although the maximum number of infected hosts was not realized until Sunday the 26th, the maximum amount of damage to the Internet infrastructure was contained much earlier.

MEASUREMENT

We desired to independently measure and characterize the Worm's infection method as well as propagation performance. Towards that end, ANML set up an isolated test network consisting of a custom-built infection and measurement (IAM) system and a standard vendor-supplied host system. The custom-built system comprised dual AMD Athlon 2100+MP processor and 1 megabyte of main memory. It was running the Linux operating system. The host system was a Gateway Pentium III system running at 1Ghz and having 256 megabytes of main memory. It was running Windows NT server and an unpatched version of Microsoft's SQL Server database server. Both systems had Intel PRO 100 100Mb/s Ethernet adaptors which we have found in previous lab evaluations to be excellent performing Network Interface Cards (NICs) under both Windows and Linux environments. The systems were directly connected together with a standard CAT 5e Ethernet crossover cable.

We repeated the tests several times. At the beginning of each test a single UDP packet was sent from the IAM to the host system and infection was confirmed typically within one to three hundredths of a second. Infection was determined by the detection at the IAM of an outbound packet from the host destined for any IP address and having a destination port of 1434 (SQL Server's UDP management port). A typical exchange is reproduced here:

```
16:19:07.010339 10.0.0.1.32808 > 10.0.0.2.1434:  udp 376 (DF)
16:19:07.027434 10.0.0.2.1033 > 192.112.252.149.1434:  udp 376
```

The first line is the infected UDP packet being sent from the IAM while the second line shows the now-infected host attempting to further infect other machines on the internet. The difference is shown by the two timestamps: one at 16:19:07.010339 and the other less than 0.03s later. Even when the additional overhead of an Address Resolution Protocol (ARP) request is factored in, as would be typical where infector and infectee machines are not in regular contact with each other, the time delay between the issue of an infection packet and infection is minimal as the following demonstrates:

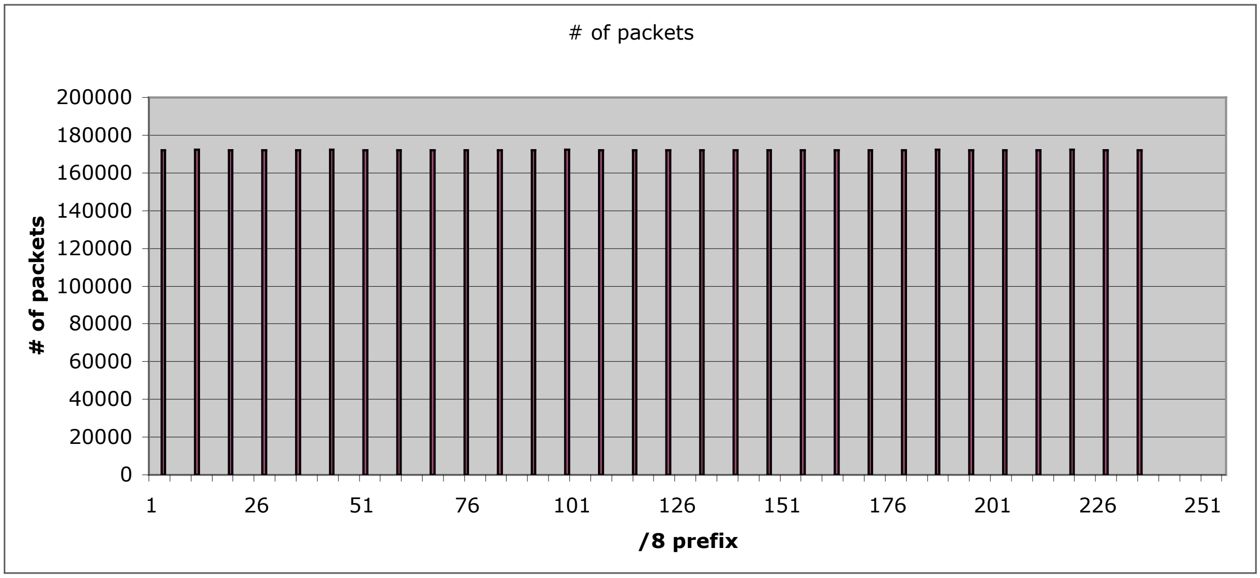
```
16:30:09.914604 10.0.0.1.32809 > 10.0.0.2.1434:  udp 376 (DF)
16:30:09.934518 arp who-has 10.0.0.1 tell 10.0.0.2
16:30:09.934571 arp reply 10.0.0.1 is-at 0:2:b3:8a:b7:8b
16:30:09.934752 10.0.0.2.1033 > 142.255.244.89.1434:  udp 376
```

Once infected our host machine was able to almost saturate the 100Mb/s Ethernet with outgoing packets despite the relatively small size of the SQL Slammer infection packet. Our 1Ghz host was easily able to generate over 25,000 infection packets per second or roughly 75Mb/s.

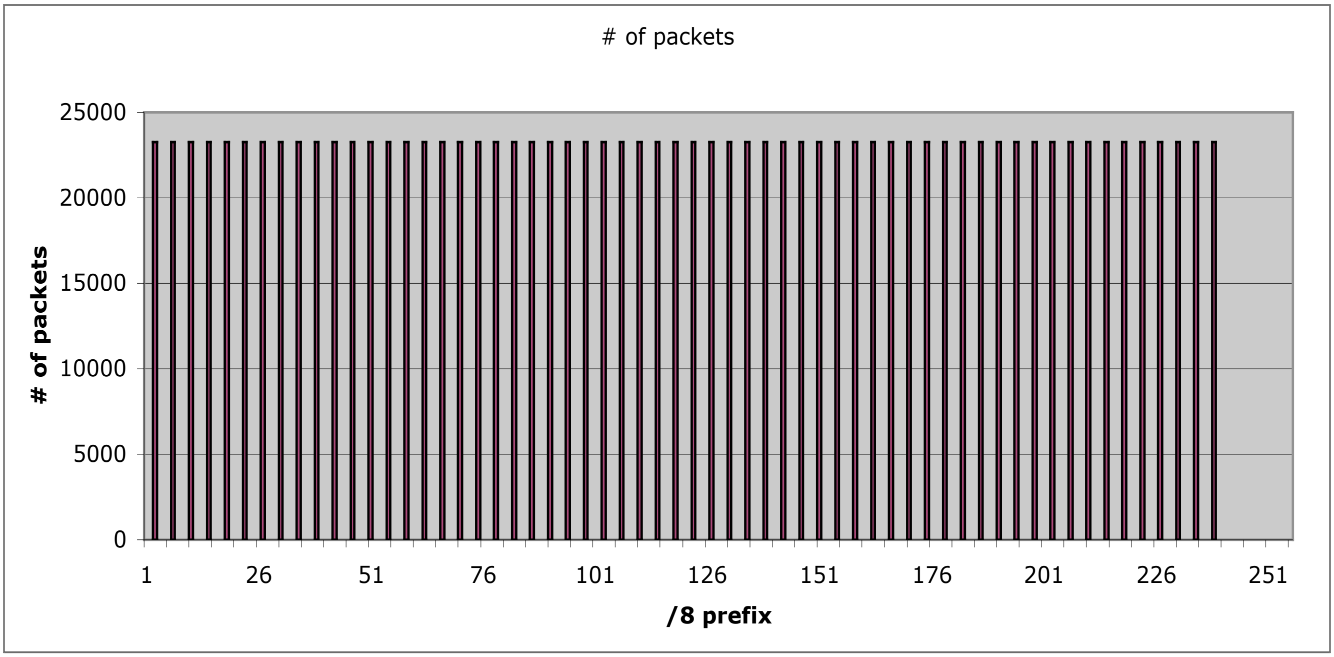
In our lab we confirmed what had already been reported elsewhere regarding the Worm's choice of source and destination IP addresses. In the case of the source address the Worm made no attempt to perform so-called "source spoofing" and always used the actual source IP address of the infected machine. In the case of destination IP addresses we found the same rigidity as others regarding the randomness of the IP number and we found significant variability between runs as a result of differing seeds. In particular, the algorithm employed reliably generated IP numbers within the entire 2^32 range of IPv4 addresses but also tended to cluster in well-defined groups within the range. How the clusters were distributed was a function of the initial seed on each infected machine with the effect that a single infected machine would repeatedly "hit" groups of the same IP numbers within the entire IP range. However, as the worm multiplied and spread to multiple hosts, each using a different seed, the effect on the internet as a whole would be that virtually every IP number would have been vulnerable.

The following are graphs of the IP distributions over three separate infections. Note that no attempt was made by the worm to avoid obvious IP ranges such as the "unroutable" or private ranges at 10.0.0.0/8, 172.16.0.0/16, and 192.168.0.0/16/24. Also the worm did not attempt to avoid the multicast range of 224.0.0.0 through 239.0.0.0 and it is this characteristic that led to most of the worm's actual damage as we will see in the next section. Note also the even distribution of IP addresses with the clusters mentioned earlier. Finally note that the worm, probably because of an ideiosyncrasy of its random number generator, did not generate any IP numbers above the range 230.0.0.0

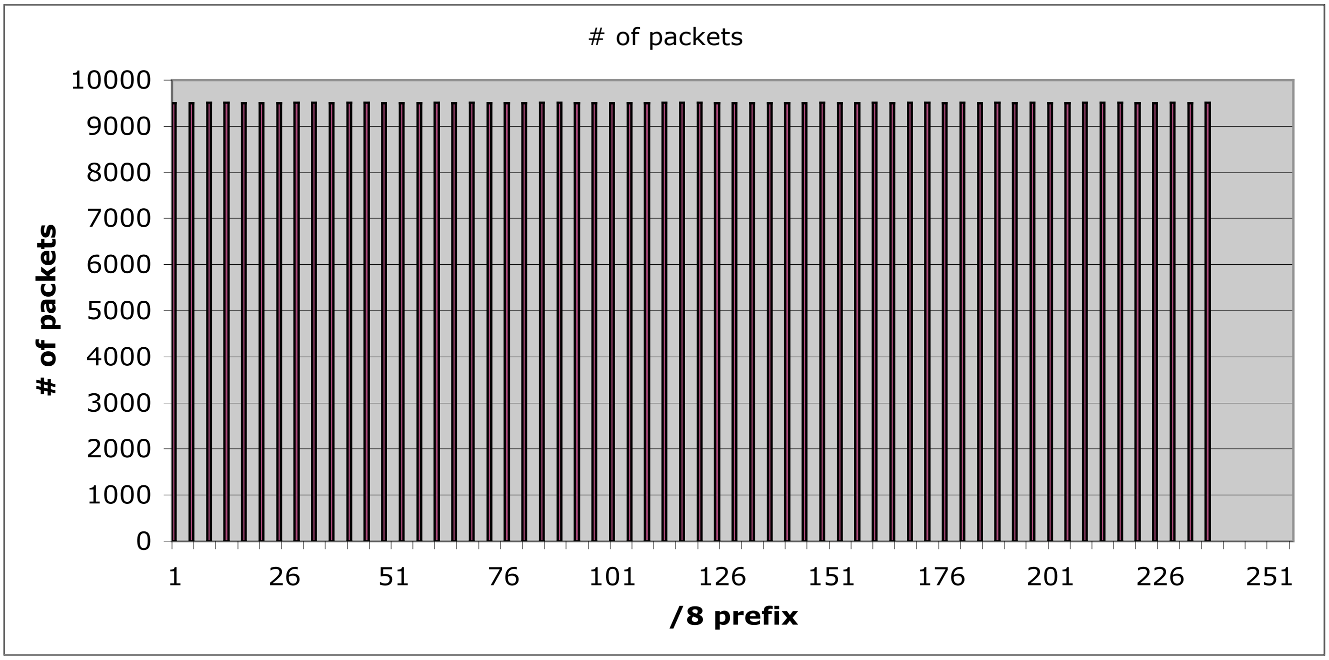
RUN 1



RUN 2


# of packets
25000
20000
15000
10000
5000
0
# of packets
1
26
51
76
101
126
151
176
201
226
251
/8 prefix

RUN 3


# of packets
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
# of packets
1
26
51
76
101
126
151
176
201
226
251
/8 prefix

EFFECT OF THE WORM

Effect on Indiana University

The worm was detected at Indiana University's Global Network Operation's Center (GNOC) at approximately 3AM EST, almost three hours after it had been released. The most immediate effect was the destabilization of the university's Juniper M20 router located in Indianapolis. Major symptoms of the destabilization included hung maintenance sessions as well as multiple memory faults. This resulted in loss of connectivity between the university and the commodity Internet as well as connections to the Abilene (Internet2) network and numerous other statewide connections. It also resulted in the loss of NETFLOW data to ANML's DDoS monitoring systems.

It was at first believed that the loss of the M20 was due to route instability, particularly with regard to BGP sessions, as the global internet attempted to reconfigure itself in the wake of a massive amount of traffic. Later analysis however did not support such a conclusion as it was discovered that the major commodity providers did not suffer the kind of failures which would cause route disruptions in their infrastructure. Analysis of the Abilene network also indicated that it did not suffer from the amount of traffic generated. In fact, although traffic was higher than normal on Abilene during the attack, it was nowhere near the network's ultimate capacity. In fact, the delta was so low that the increased amount of traffic alone was insufficient to trip our DDoS detectors.

It was ultimately discovered that the M20 was brought down not by excessive traffic or by route instability but by the ingress of multicast state from the Abilene network. Abilene, like most research networks, fully supports IP multicast services while most commercial commodity network providers still do not. As we showed above, the worm's random IP algorithm regularly generated destination IP addresses in the IP multicast space. Because most networks do not support multicast IP most worm packets being sent to multicast destinations died close to their source. However, infected systems on networks which supported multicast allowed those packets out. If they reached Abilene they were able to transit the globe

Effect due to Multicast state creation

Impact of Slammer on router control planes

The classic high-speed router model includes a data plane used to forward IP packets and a control plane used to exchange router-to-router information such as routing protocols (e.g. OSPF, BGP, IS-IS). Normally all packets sent between computers in the Internet pass through the data plane only and do not need to be sent to the control plane. Since routers are designed to forward packets within the data plane at wire-speed, extreme use of the network, whether from the propagation of a worm or more legitimate intense use, does not cause them to crash or otherwise fail. However any network traffic which requires control plane intervention usually requires the dedication of limited resources such as memory, router CPU, or internal router bandwidth. Thus any traffic to the router which enters the control plane is potentially the source of a denial of service attack. Router operating software carefully monitors the amount of control plane activity to ensure that no attack is possible. However, in the case of IU's M20, there existed a method by which IP multicast traffic could overwhelm the control plane and ultimately cause the router to fail.

IP Multicast

Slammer, like the RameN worm, targeted a wide range of IP addresses including those reserved for IP multicast transmissions. Unlike normal IP unicast transmissions, IP multicast packets have a direct effect on the router control plane. When an IP multicast packet is transmitted the network must signal to adjoining networks that there is a new source for this particular multicast group (multicast destination address represent groups of listeners). Routers must keep a list of the active groups within their networks and exchange this information periodically with other routers.

Under normal conditions the Internet has state for a few thousand multicast transmitters. By randomly creating packets with destination IPs in the multicast address space the SQL Slammer worm caused this state to increase by an order of magnitude. This increase was outside the capability of many routers on the Internet, including Indiana University's M20 in Indianapolis, and had a drastic affect on the stability of networks. Numerous campus networks in the Internet2 community, as well as the largest research and education backbone in Europe, where disrupted in part due to the increased state required by the bogus multicast transmitters.

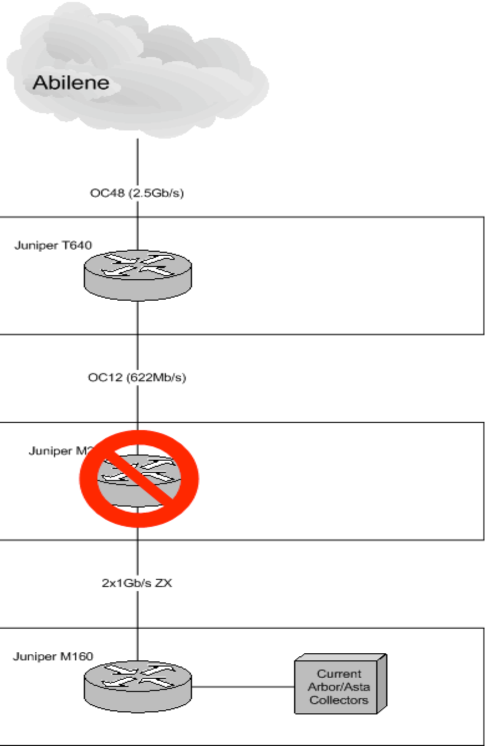
As we mentioned earlier, SQL Slammer was not the first worm to cause damage as a result of scanning into the multicast address space. The RameN worm had a similar, and in some ways more intense affect on routers in the research and education network environment due to its sequential scanning of IP addresses as opposed to Slammers pseudo random pattern. At the time of the RameN worm the Abilene network was composed of Cisco GSR routers (it is now composed of Juniper T640s). After the effect of the additional multicast state was understood, Cisco added a software feature that gave network operators a tool to limit the amount of multicast state that would be exchanged between networks, thereby providing a form of firewall from a future incident. Indiana University’s Juniper M20 router had not yet had a similar software feature added by the vendor and, as a result, it failed in a manner similar to the earlier Cisco router failures.

Effect on DdoS monitoring capability

The laboratory's ability to detect and track DDoS events across the Abilene backbone consists of the following infrastructure: The Abilene core routers of which there are twelve, NETFLOW collection devices provided by Arbor and Asta, and the Arbor and Asta aggregation and analysis systems. The Abilene core routers are distributed geographically across the continental United States while the Arbor and Asta equipment is centrally located in Indiana in both Bloomington and Indianapolis.

Each Abilene core router generates a constant stream of NETFLOW records as the router detects individual streams (either TCP or UDP) across its interfaces. Those NETFLOW records are sent to mirroring servers located in the same physical space as the routers and those mirroring servers then duplicate the flows and send them to all customers of Abilene NETFLOW records, including ANML's Arbor and Asta equipment. ANML's Arbor and Asta equipment then analyzes the NETFLOW records comparing them against historic levels as well as analyzing them for the signatures of known DDoS attacks. Either a departure from normal traffic levels or the detection of a signature causes a DDoS event to be logged in the systems.

As outlined above, the effect of large amounts of multicast state generation, as a side-effect of the worm probing into the multicast address space, was to disable the university's M20 router located in Indianapolis. This router currently sits between the Abilene network and the Indiana University Purdue University Indianapolis (IUPUI) campus network as well as the Indiana University Bloomington (IUB) campus network (see Figure XX below) and all of ANML's Asta and Arbor infrastructure currently sits behind the M20.



Because the M20 failed ANML's DDoS infrastructure stopped receiving NETFLOW packets from the Abilene core routers at approximately 3AM EST (the same time that the general effects of the worm were becoming felt at IU). From 3AM until approximately 9AM we received sporadic NETFLOW packets and saw UDP traffic to destination port 1434 among the NETFLOW records. However, because our Arbor and Asta gear had not been programmed to look for the Slammer signature and because the observed Abilene traffic was not significantly above historic levels our equipment generated no significant alerts.

We are currently re-evaluating our NETFLOW disbursement and collection architecture. One obvious solution is to physically move our Asta and Arbor collectors so that they are adjacent to the Abilene core routers. That would avoid having to ship NETFLOW records across the Abilene network and to Indiana and would, presumably, avoid another situation in which we lose collection and monitoring capability because of a general connectivity failure that is the result of either human error or another attack like SQL Slammer. Another option is to move the Asta and Arbor collectors to the other side of the M20 while continuing to house them in Indiana (Indianapolis, actually. The collectors located in Bloomington would move to Indianapolis).

None of the proposed alternatives is without a downside. Moving the collectors out to the core router sites involves securing rack space and connectivity to the routers as well as acquiring significant additional equipment – equipment that must be compatible with the 48 volt DC supplies that are standard in the TELCO environments in which the Abilene core routers exist. There are also increased difficulties involved in maintaining the equipment from a distance. Finally, we need to plan for the possibility of another attack which does not impact the collector's ability to gather NETFLOW records but does prevent ANML from reaching the collectors with the net result that the additional cost of a distributed collection architecture bears no fruit with regard to higher system reliability. In fact one of the common issues brought up by network engineers as they attempted to ameliorate the effects of the worm was the difficulty in communicating with remote equipment.

EFFECT ON OTHER INSTUTUTIONS

In an effort to obtain a perspective of the effects of the worm on a broad regional basis, we polled network engineers at a number of network centers nationwide (California, Georgia, New York, and Indiana). This expanded our qualitative knowledge of the attack and allowed us to determine if there was any variation in the severity of the effects of the worm as it propagated geographically.

Of the centers polled all first became aware of difficulties related to the worm between 1 and 2AM EST. This was in line with Indiana University's experience as well as various experiences reported in the media. The California Research Network (CalREN/CNIC) in California reported the earliest problems, starting around midnight. As it is now generally accepted that the worm reached maximum effectiveness within ten minutes of release we conclude that differences in the reporting time can be attributed to human response times.

Our research indicates that, in contrast to media reports, the worm's effect on individual networks varied greatly. As we reported earlier, the Abilene network was almost completely unaffected (although it was a significant "carrier" of the worm). Through queries we discovered that networks such as IPLS, CalREN/CENIC and Southern Crossroads (SOX) all suffered from significant problems, but others including New York State Educational Research Network (NYSERNet), Mid Atlantic Crossroads (MAX), and George Washington University (GWU) claimed that the worm had little or no impact on their network performance. Moreover, on the networks that were affected issues were more likely in general to be internal as opposed to external problems with connections to other networks (the "outside world). Where external connections were hindered they usually were the result of internal issues e.g. border routers overloaded by too many packets trying to get out, rather than the network connections between sites being over capacity, or overloading of internal links. Again the experience of other institutions in this regard is similar IU's M20 failure which caused it to lose connectivity with the commodity internet as well as Abilene and others.

There were few instances of routers failing from CPU overload. Two sites (SOX and GWU) suffered from ATM failures due to overloading. Of the institutions we surveyed the worst situation seems to have been IPLS, with infected machines filling external links, and a number of other problems which could be traced back to Multicast Source Discovery Protocol (MSDP) storms from several machines -- a consequence of the random number generator on one or more machines generating destination IP addresses in the multicast space.

Once the source and nature of the problem has been established, almost all affected networks were able to restore normal operations very quickly. Difficulties with in-band access due to network overloading were encountered by Indianapolis (IPLS) and SOX and CalREN/CENIC had difficulties tracking down infected machines (apparently due to a bug in some Cisco equipment logging incorrect port numbers). Another issue of note was the practice of sites downstream of the network centers installing filters to block inbound traffic while not realizing that they also were *generating* huge volumes of outbound traffic -- contributing to the difficulties further downstream.

The single consistent account we received was that effective communication between network centers and their clients was essential. For example, GWU experienced almost no problems at all, due largely to timely notification from upstream providers allowing them to have filters in place very early. And while NyserNET didn't experience any problems themselves, downstream networks (that they could have protected) failed to notify them of any difficulties until their own situation was out of hand at which point the load on the network made diagnosis and rectification difficult (a situation complicated by the extremely fast propagation of the worm). Congruent to IU's experience it appeared that the most serious problems were caused by the generation of destination IP addresses in the multicast domain while limited actual implementation of multicast mitigated these problems. Had the worm's random number generator been correctly implemented, there would probably have been a wider incidence of multicast-related problems although the severity at individual sites would have been reduced.

MITIGATION

As we hinted in our introduction there is nothing particularly sophisticated about SQL Slammer and it is this fact more than any other that is most troubling. Fifteen years after the release of the Morris worm the internet was heavily disrupted by a similar worm which existed for one reason only: to replicate itself as fast as possible. Like the Morris worm, SQL Slammer does not attempt to destroy data, to infect programs on permanent storage, or to extract sensitive information. It exists simply to try and re-create itself and it is this simplicity, manifested in its rate of propagation, which brought the internet once again to its knees for a short time on Saturday, January 26th.

In this context SQL Slammer represents not a lesson regarding new technology but a lesson on things forgotten. What are those things? First that buffer overrun exploits happen. There was nothing inherent about Microsoft's SQL Server which made it a host for the worm other than the fact that it possessed a buffer overrun vulnerability – a vulnerability it shares with hundreds of other programs; programs from commercial vendors, programs from academia, programs from research, etc. There is nothing that prevents a variant of SQL Slammer being adapted to take advantage of a buffer overflow in another program with a similar vulnerability. The lesson here is that buffer overflow problems continue to be, fifteen years after Morris, a significant security issue affecting internetworked computers. What steps that can be taken to reduce or eliminate buffer overflow exploits need to be taken and strengthened. In particular vendors need to be ever more pro-active in their support of buffer overrun control. Reports are that Microsoft's fix to SQL Server, while available for over six months prior to Slammer, wasn't widely deployed due to the difficulty of applying the necessary patches. Indeed, even Microsoft's own internal networks and machines were infected as a result of its own personnel apparently shunning the task of installing the fix.

The next lesson is perhaps more a casualty of SQL Slammer than a lesson per se. The casualty is the end of an era of permissive connections where networks, institutions, network providers, etc. permissively allow arbitrary port connections between computers. SQL Slammer propagated by finding computers listening on UDP port 1434. The vast majority of those computers probably didn't need to allow connections outside their own local subnet, much less outside their associated institution. Had institutions or departments already implemented firewall policies where connections were by default denied, as opposed to allowed as is common today, the worm's spread would have been a fraction of what it was.

We note, however, that to be effective firewalling has to be completely pervasive. SQL Slammer required the passage of only a single 376 byte UDP packet, a miniscule amount of traffic, to infect an entire enterprise network. The implication, for large enterprises that have tens if not hundreds of internetwork connection, is that every single point of possible ingress needs to be locked down. For example, we discussed the effect of the worm with a large international pharmaceutical company – a company that spent a significant amount of time and resources controlling the spread of the worm within its internal network. The question arose: how did the worm get into the internal network in the first place? That's a question that may never be known as it became clear that the internal network of this multinational corporation spanned multiple continents each with tens of hundreds of points of contact between the internal network and external ones. Ingress of the worm required that only a single one of those points allow a single 376 byte UDP packet, which appeared destined for an entirely legitimate port, ingress.

Another lesson is that of maintaining sufficient communication and control backchannels, using paths separate from the main lines over which normal network traffic is expected to transit. As reported, early attempts to even understand the situation were hampered by communication failures into infrastructure equipment. For example, network engineers repeatedly encountered troubles even using TELNET or SSH to gain command-line control of routers. Medium and long-term analysis was hampered by a loss of telemetry data from distant routers, such as the experience that ANML had with its NETFLOW feeds. In discussions with a large diversified global manufacturer we learned that their primary issue with regard to SQL Slammer was a loss of communication between network engineers, hampering control. Their primary action item resulting from SQL Slammer was decidedly low-tech: the installation of facsimile connections between all critical global locations to act as a failsafe backup to the normal corporate email communication channels.

The final lesson is to be ever vigilant over potential and actual single points of failure. Indiana University's experience with the worm would have been quite different had the worm not been able to fail IU's M20 router due to insufficient safeguards against the creation of excessive multicast state in the router. Adding salt to the wound is the fact that a previous worm, the RameN worm, had already demonstrated this exact vulnerability with the result that at least one router vendor had applied fixes to its operating software. Yet just as buffer overrun exploits still plague internetworks a decade and a half after Morris, multicast exploits continue to cause damage two years after RameN. It is difficult to see how "non revenue" priorities can be increased for router and software vendors but investigation into how to tie architectural and design flaws which currently damage only consumers and other third parties back to the vendor's shareholders may yield significant advances in this area.

FOR FURTHER READING

Analysis of the Sapphire Worm - A joint effort of CAIDA, ICSI, Silicon Defense, UC Berkeley EECS and UC San Diego CSE (http://www.caida.org/analysis/security/sapphire/)

The Morris Worm: how it Affected Computer Security and Lessons Learned by it (http://www.sans.org/rr/malicious/morris.php)

Slammer Worm Resources (http://www.microsoft.com/sql/techinfo/administration/2000/security/slammer.asp)

Advanced Network Management Lab (http://www.anml.iu.edu)