

Monitoring Network Sludge: Using Intrusion Detection to Monitor Network Traffic

Sarah Diesburg
University of Northern Iowa
sarahm@uni.edu

Abstract

This paper explores the usefulness of using an Intrusion Detection System (IDS) to track potentially illegitimate traffic entering the University of Northern Iowa's (UNI's) network. I will demonstrate the tracking of four specific types of potentially illegitimate traffic with Snort¹ (an open-source IDS) by discussing the traffic in detail, comparing the traffic frequency results with global traffic patterns logged by the Internet Storm Center², and discussing post-project concerns.

1 Introduction

At the time of this writing, the average survival time of an unprotected system is 16 minutes. This means that, on average, a system directly connected to the Internet without any sort of firewall will have about 16 minutes until it experiences its first unauthorized network connection attempt or probe. Many unprotected and protected systems become infected or attacked each day.

I am interested in tracking such Internet traffic through the use of an Intrusion Detection System (IDS). An IDS is an application running on a system (usually placed at the entrance to a network) that analyzes network traffic for matches against a user defined rule set and performs several actions based upon what it sees. Usually these actions consist of logging any traffic that matches any rule in the predefined rule set while passively letting the traffic through, although an IDS can be easily configured to block the traffic as well.

This paper will attempt to demonstrate the usefulness of using an IDS to passively monitor four forms of illegitimate network traffic entering the University of Northern Iowa's (UNI's) network between October 1, 2004 and November 31, 2004. The monitored traffic will be discussed in detail, as well as analyzed and compared to other global traffic patterns. I will also consider a few post-project concerns, or things that I now see could have been done differently to better facilitate the project as a whole. First, though, I should discuss the specific hardware and resources I used throughout the course of the project.

1 Snort can be found at <http://www.snort.org>.

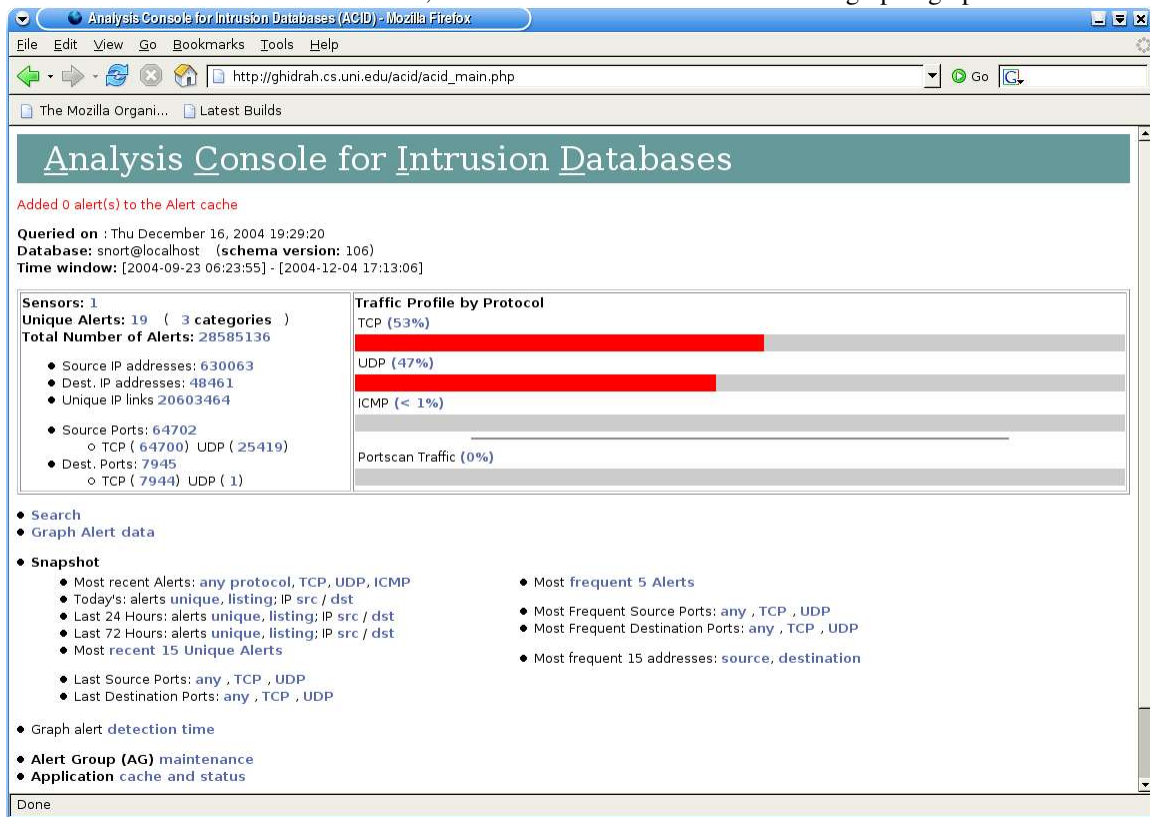
2 The Internet Storm Center can be found at <http://isc.sans.org>

2 Hardware and Resources

My first order of business was to secure permission to pursue this project. Once I had permission, I needed a way to either observe the traffic entering UNI's network first-hand, or otherwise receive logs from someone already doing this. It turned out that the Systems and Operations Manager of UNI, Ken Connelly, was already monitoring the network traffic entering UNI with an IDS, and it was trivial for him to send me daily logs with the information I requested.

Now that I had binary logs in tcpdump format pertaining to the specific instances of network traffic I was interested in, I now had to find applications and hardware to parse this data into an analyzable format. For this task I decided to use Snort v2.2.0, an open-source IDS, along with MySQL v4.0.22 for processing my logs and storing them in a database. I also used Acidlab v0.9.6b23, which is a open-source, PHP-based analysis engine used to search and process a database of security events generated by various IDS's, firewalls, and network monitoring tools. Acidlab came in very handy to graphically view and chart my data, as seen in figure 1 below.

Figure 1: Acidlab shows a graphical comparison of TCP/UDP/ICMP port hits and miscellaneous statistics, as well as a multitude of retrieval and graphing options.



I installed and used these applications originally on a Debian-based system with a 2GHz processor and 1G of RAM. After my database became too large and memory-intensive to handle, I moved it to another Debian-based system with two Intel XEON 2.20GHz CPU's and 1G of RAM.

3 Monitored Traffic and Results

It was tough to decide what specific instances of network traffic to analyze. I couldn't watch all the traffic due to security concerns, and narrowing down the list of candidates was a hard task at best. I wanted to choose traffic that would be interesting to monitor as well as to learn about. Therefore, my investigations in September consisted of watching global network trends through the Internet Storm Center's (ISC's) website and being subscribed to UNI's local port-scan activity mailing list. The ISC is a volunteer organization that gathers millions of intrusion detection log entries every day from sensors covering over 500,000 IP addresses in over 50 countries. UNI's local port-scan activity mailing list is a private mailing list which sends out daily emails containing valuable information on which specific ports at UNI are being probed the most.

The illegitimate traffic I finally chose to monitor consisted of:

- a Dabber Worm Propagation Attempt on TCP port 9898,
- a Radmin Server Vulnerability Probe on TCP port 4899,
- a DameWare Remote Administration Buffer Overflow on TCP port 6129,
- and a MS-SQL Worm Propagation Attempt (Slammer) on UDP port 1434.

To enable the IDS to correctly capture this data, I had to come up with 4 signatures, or snort rules. The IDS then compared all incoming traffic to these snort rules, looking for a match. When a match was found, an alert was generated. Each snort rule will be discussed in detail in the proper section below.

3.1 Dabber Worm Propagation

Dabber is a worm discovered in May of 2004 which spreads by exploiting a vulnerability component of the Sasser worm in Windows 2000 and Windows XP. In other words, this worm will only affect systems already infected with the Sasser worm. Once infected, Dabber actually deletes the registry keys of Sasser and other viruses, while opening up a backdoor listener on TCP port 9898. I thought this virus was interesting in that it completely relies on the presence of an older worm to propagate, and I expected network frequencies of this worm to decrease in time as more system became patched. Below is the snort rule I used to track traffic on this port:

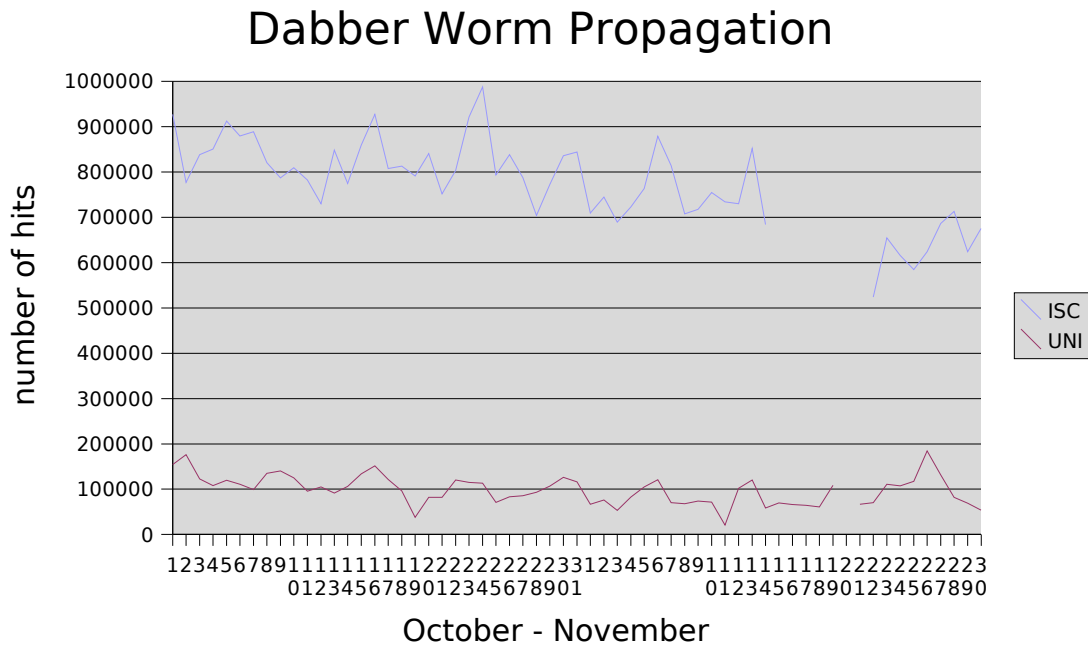
```
alert tcp $EXTERNAL_NET any -> $HOME_NET 9898 (msg:"Dabber backdoor attempt"; classtype:misc-attack;)
```

This rule means that:

- I would like to generate an `alert` if this rule is matched,
- I am looking for a `tcp` packet,
- This packet may come from any IP with any source port as long as it has a destination port of `9898`,
- I would like its alert message to read "`Dabber backdoor attempt`",
- and it has a classtype of `misc-attack`.

Figure 2 below contrasts the global connection attempts of TCP port 9898 to local connection attempts entering UNI's network.

Figure 2: The frequency of incoming connection attempts to TCP port 9898 to UNI's network contrasted with global data.



It seems that the general shape of are similar, although UNI's frequency is well below global average. Both frequencies also seem to be declining, which may mean that there are fewer Sasser-infected systems.

3.2 Radmin Server Vulnerability

Radmin³ (pronounced “remote administrator”) is a very fast, very powerful remote administrator server available to Windows 95 and above. It is a commercial product that allows a user to visually and remotely take control of the computer on which it is installed with administrative rights and privileges. By default, Radmin uses TCP port 4899 and password authentication for remote access. The default port can easily be changed, and it is in fact recommended to do so. In September, the number of probes for

³ Radmin can be found at <http://www.radmin.com>.

this service was extremely high without a known exploit in both the UNI port-scan emails and in the ISC data. Thinking that an exploit may surface in the next few months, I decided to watch for this traffic using the snort rule below:

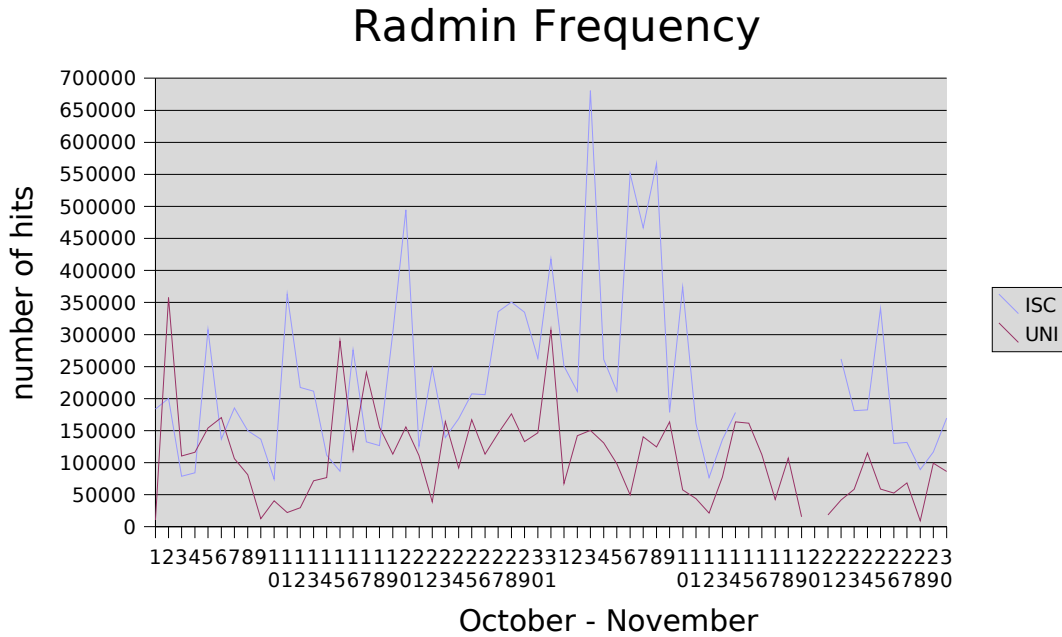
```
alert tcp $EXTERNAL_NET any -> $HOME_NET 4899 (msg:"Radmin connection attempt"; classtype:attempted-admin;)
```

This rule means that:

- I would like to generate an `alert` if this rule is matched,
- I am looking for a `tcp` packet,
- this packet may come from any IP with any source port as long as it has a destination port of 4899,
- I would like its alert message to read "Radmin connection attempt",
- and it has a classtype of `attempted-admin`.

Figure 3 below contrasts the global connection attempts of TCP port 4899 to local connection attempts entering UNI's network:

Figure 3: The frequency of incoming connection attempts to TCP port 4899 to UNI's network contrasted with global data.



Out of all my project graphs, this graph shows the best local-to-global correlation. Both sets of data have large spikes and lows, and each seem to occur roughly at the same time. It seems that global trends for this traffic would well predict local frequencies. UNI's overall frequencies are only slightly smaller in number to that of the global average.

Recently I have learned of a possible reason for the recent increase of Radmin activity on this port. It seems that a system with Radmin v2.1 or earlier installed is vulnerable to a hole created by running "Spybot – Search & Destroy", a free spyware detector and removal application. If run on such a system, Spybot will detect Radmin as the Haxdor-H trojan and request to have this "trojan" removed. If the user continues, Spybot will remove the Radmin registry settings which causes Radmin to return to listening on its default port (TCP 4899), causes all logging to be turned off, and causes all passwords become blank! The worst part is that Radmin v2.1 or earlier will *allow* users to log in with a blank password. Thus it would become very simple for anyone to simply log into a system where this scenario occurred and take complete control.

I found the above information mentioned on official forums which I have listed at the end of this paper for those more curious of this issue. Fortunately, the official Radmin website does not offer any version below 2.2, which also unfortunately meant that I was unable to test this vulnerability for myself.

3.3 DameWare Remote Administration Buffer Overflow

DameWare Mini Remote Control⁴ is a commercial product which allows remote system management and administration of Windows NT/2000/XP/2003 networks. An exploit was found in versions 3.72 and prior in November of 2003 which takes advantage of an instance in the remote authentication process where no bounds checking is done to overwrite the return address on the stack and have the process call the attacker's code. This vulnerability has been patched and does not exist in the current application version. Even though this was an older exploit, I was still curious about the slow but steady traffic frequencies I noted in September. To watch for this traffic, I used the snort rule below:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 6129 (msg:"Dameware
Remote Administration buffer overflow attempt";
classtype:attempted-admin;)
```

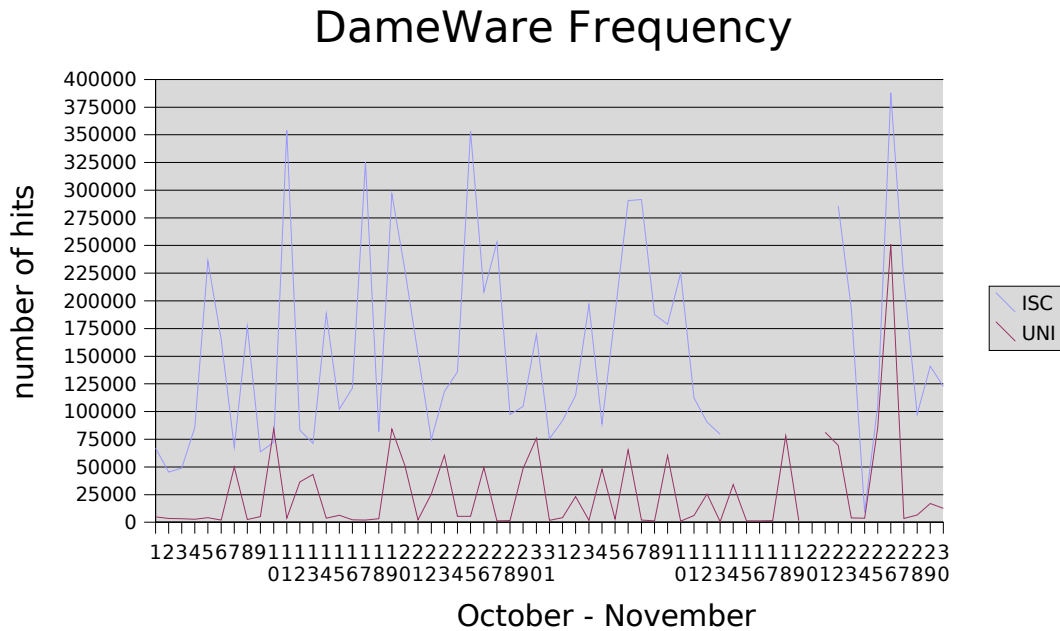
This rule means that:

- I would like to generate an `alert` if this rule is matched,
- I am looking for a `tcp` packet,
- this packet may come from any IP with any source port as long as it has a destination port of `6129`,
- I would like its alert message to read "Dameware Remote Administration buffer overflow attempt",
- and it has a classtype of `attempted-admin`.

Figure 4 below contrasts the global connection attempts of TCP port 6129 to local connection attempts entering UNI's network:

⁴ DameWare Mini Remote Control can be found at <http://www.dameware.com>.

Figure 4: The frequency of incoming connection attempts to TCP port 6129 to UNI's network contrasted with global data.



Both sets of data seem to look very spikey, although the spikes do not necessarily occur at the same time. Of greatest note is the spike which occurred on November 26th. This day had the highest frequency of hits for *both* data sets, which may possibly mean something significant.

3.4 MS-SQL Worm Propagation (Slammer)

Slammer was released in January of 2003, yet still tops many lists in the sheer frequency of its traffic. This worm is interested in compromising Microsoft SQL Servers and exploits a buffer overflow in the Resolution Service. A patch has since been released for MS SQL Server, yet many unpatched systems still obviously exist. This traffic interests me because of the extremely large amount of it that won't seem to go away. Below is a snort rule that catches this specific traffic:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 1434 (msg:"MS-SQL Worm propagation attempt"; content:"|04|"; depth:1; content:"|81 F1 03 01 04 9B 81 F1 01|"; content:"sock"; content:"send";)
```

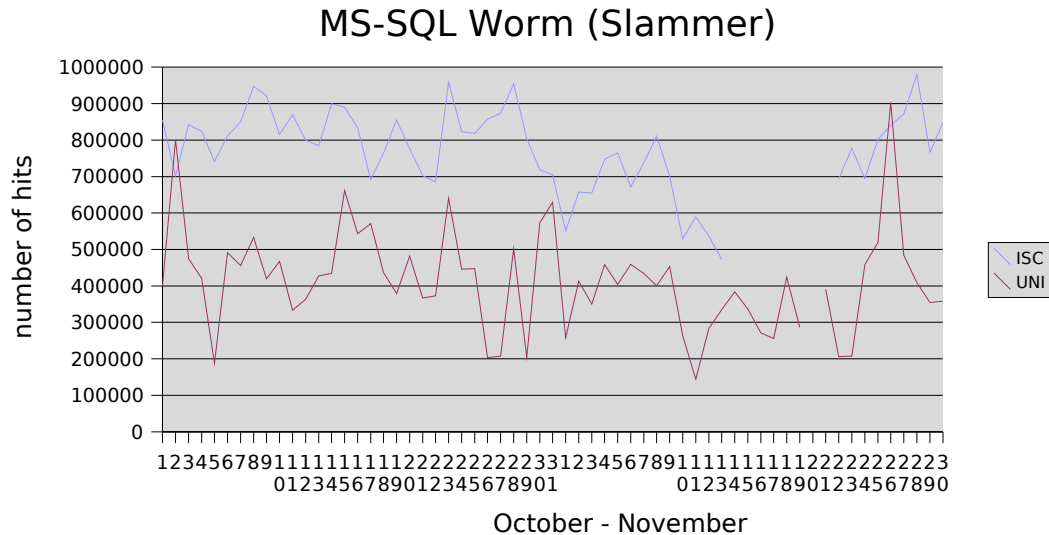
This rule means that:

- I would like to generate an alert if this rule is matched,
- I am looking for a udp packet,
- this packet may come from any IP with any source port as long as it has a destination port of 1434,
- I would like its alert message to read "MS-SQL Worm propagation attempt",

- and it has specific hex characters and words in specific places in the payload of the packet.

Figure 5 below contrasts the global connection attempts of UDP port 1434 to local connection attempts entering UNI's network:

Figure 5: The frequency of incoming connection attempts to UDP port 1434 to UNI's network contrasted with global data.



It seems the UNI frequency is a bit more volatile than the global average. A slammer-infected machine is often characterized by large amounts of worm propagation attempts in very short periods, which would possibly explain the sharp spikes and lows. It also seems that, by this graph, the worm isn't going anywhere.

4 Post-Project Concerns

This project seemed to be riddled with problems. In this section I will only talk about the two largest problems I encountered as well as steps I could have taken to prevent them. These problems include less than optimal hardware to house a large database and the complete and utter loss of an entire dataset.

Towards the end of the project I noticed the query retrieval time of the University of Northern Iowa dataset became quite long. My database had grown to a whopping size of 28G, with over 28.5 million logged alerts by the end of the project. Unfortunately, the original system I housed this database on couldn't seem to handle the load, crashing with kernel problems and page faults which caused many corrupted database tables and long wait times while they were being fixed. I could have predicted the size of the end database early into the project and requested the use of a machine with faster processor(s) and more RAM to handle the heavy retrieval loads.

I also lost an entire dataset towards the end of the project! My project had originally consisted of comparing frequencies of selected network traffic entering the University of Northern Iowa *and* Cedar Falls Utilities(CFU), but unfortunately I had no user account to the CFU snort machine and had made no arrangement for data backups to be made. On December 6th of 2004 the CFU machine which housed all of the snort data, including the database and original binaries, was broken into and compromised. Unfortunately, due to CFU policy, all of the data had to be completely wiped off the machine.

I could have done numerous things to prevent the total loss of this data. For starters, I could have requested the snort data to be sent directly to me to be analyzed much like my arrangement with UNI. In this way I could have been in charge of my own data. I also could have requested backups to be made, either sent to me or another safe location.

5 Conclusion

In conclusion, most of the local UNI frequency data “generally” matched global data in terms of shape and stability. This was to be expected. If this were not the case, then either locally or globally something abnormal could have been occurring, perhaps in the form of new traffic. If I were comparing two sets of frequencies from similar local networks, I would venture to guess that both sets of frequency data would have been much more similar. It then could have been possible to pinpoint problem systems which appeared in both datasets.

I think this project demonstrates the importance of paying attention to network trends, and I think that IDS's are excellent ways to do this. Due to the project I am able to conclude, concerning the specific ports I observed, that UNI's traffic seems relatively normal. This project also enabled me to watch (through the Radmin traffic) the progress of a new sort of vulnerability unfold, as well as its repercussions and traffic patterns. Snort itself proved to be a very capable at intrusion detection, and Acidlab proved to be very insightful, although slow with large datasets.

References

Dabber

LURHQ Corporation. Dabber worm analysis. <http://www.lurhq.com/dabber.html>

DameWare Mini Remote Control

Beyond Security Ltd. Technical synopsis of exploit code and timeline of events.
<http://www.securiteam.com/windowsntfocus/6N00B1P95I.html>

K-OTIK Security. Full DameWare Mini Remote Control exploit code.
<http://www.k-otik.com/exploits/08.13.nfm-shatterdame.c.php>

MS-SQL Worm Propagation (Slammer)

Snort. Snort Signature Database. Contains detailed information on snort rule and worm.
<http://www.snort.org/snort-db/sid.html?sid=1%3A2003>

Radmin Links (in chronological order)

Virus.org. September posting of increase in traffic to port 4899 through the VulnWatch mailing list.
<http://lists.virus.org/vulnwatch-0209/msg00001.html>

Famatech Remote Administrator Troubleshooting Forum. Radmin/Spybot problem.
<http://www.radmin.com/support/forum/read.php?FID=11&TID=7697&PHPSESSID=ecd6f7b42924911795d94ba6343b9213>

Net-Integration. Official Spybot Search & Destroy Forum. Includes Spybot's response to Radmin/Spybot problem.
<http://forums.net-integration.net/index.php?act=ST&f=28&t=23482>