Kevin Chen, Jennifer Tu, Alex Vandiver

Analyzing Network Traffic from a Class B Darknet

December 9, 2004

darknet@mit.edu

Contents

List of Tables

List of Figures

1 Introduction

Darknets are "a portion of routed, allocated IP space in which no active services or servers reside."[2] They are *dark* because packets go in but nothing comes out. All traffic present in a darknet is due to a misconfiguration, or, more likely, malicious probing. These networks are also called sinkholes, blackholes, or network telescopes.

Darknets can be used to fine-tune intrusion detection systems, serve as an early warning system before a worm's onslaught, or simply observe small or distant network events. Darknets can range in size from a single IP address to as large as you can possibly obtain.

This paper provides a summary of our experiences setting up a Class B darknet, a review of the tools we encountered, and some of our findings.

2 Prior work

There is some amount of work being done with darknets today. Most notably, these include Team Cymru, CAIDA, and the University of Michigan's Internet Motion Sensor project.

Team Cymru is "a group of technologists interested in making things better." They've assembled and provide a lot of documentation and guides, including an excellent introduction to darknets, as well as a brief guide to deploying a darknet. They recommend a few pieces of software for gathering router statistics and handling darknet data. Unfortunately, these recommendations are much more useful if you already know exactly what you're looking for. No tools are recommended to simply study general trends or patterns.

CAIDA is the Cooperative Association for Internet Data Analysis. They offer several papers and presentations on network telescopes and their utility in observing security events. They have developed the CoralReef software suite to analyze network traffic, and uses it as the basis of their network telescope monitoring system. CAIDA also provide some backscatter data (from February 2001) on request to researchers and CAIDA sponsors.

The University of Michigan's Internet Motion Sensor project collects darknets across the globe. Participants use the IMS software to collect data from their own darknets (which they have full access to), and can view aggregated, anonymized data from other participants. They do not have much public information beyond a description of the project.

There are other tools or methods available to observe malicious network traffic. Honeypots, for example, are machines (or what appear to be machines) set up for the explicit purpose of observing intrusion and subsequent use of a "compromised" machine. They can also be used to lure adversaries away from actual production servers. Darknets and honeypots serve two separate purposes, however. Honeypots are useful for *after* a would-be intruder has discovered a host. Darknets are useful for observing how the would-be intruder *before* he has discovered a victim.

Our work with darknets will hopefully make public more information about darknets and their

deployment. This paper should provide the reader with a good idea of what to expect out of a darknet, a more thorough guide to publicly available software, and an understanding of current trends.

3 Goals

Darknets are often run with several small network segments distributed across many subnets. This project, however, used a single large chunk of the MIT network. While we ended up with a clearer picture of what is occurring, this information is heavily MIT-specific. That is, we have a good idea of what's occurring locally, but may have a less realistic understanding of how the rest of the Internet may be functioning.

When we started this project, we were not completely sure what we would accomplish. There's very little information on what to expect out of a darknet. Simple facts were completely unknown to us. For example, we did not know how much disk space the packet log would use. (Although we could obtain an upper bound by looking at statistics from active portions of MIT's network, we still didn't have a good idea of what percentage might be "junk" until we started capturing data.)

One obvious application of a darknet is to use it as an extension to MIT's intrusion detection system. A lack of centralized vulnerability management and lack of centralized data about service deployment, however, makes it infeasible to take information learned from a darknet to decrease the attack risk across all systems at MIT.

Because there is a dearth of information about darknet tools and methods, one of the goals of our research was to examine the tools which were suggested or in common use for darknets, and analyze their utility. Finding few that satisfied our curiosity, the focus shifted to creating a basic set of tools to allow us to aggregate and visualize activity on the darknet.

4 Methods

4.1 Network architecture

We allocated and set up a class B network as our darknet, 18.x.0.0/16. This prefix maps to an MIT network backbone router in building W92. The router forwards packets to our logging server (running Debian Linux) over a GRE tunnel. We collected 17.5 gigabytes of recorded packets in a little over two weeks.

GRE tunneling is similar to IP-in-IP tunneling, in that it provides a way to encapsulate and forward IP packets between routers. This is most often used to create private networks which appear to be adjacent in network topology, even though they are in reality separated by the public Internet. In our case, one endpoint of the tunnel was the backbone router in building W92 and the other was our computer in building 32. The GRE tunnel served to encapsulate packets to our darknet, and forward them to our computer, where we could sniff them.

4.2 Hardware and kernel

We set up a Pentium III 700 MHz machine with 256 MB of RAM as a minimal Debian machine. Its hard drive was partitioned with a 2 GB root partition, 0.5 GB swap partition, and a 17.5 GB `/data` partition, which we used to store the network traffic we received.

To set up the GRE tunnel, we needed to recompile the Linux kernel to include support it. We chose the latest version of the Linux 2.4 kernel, 2.4.27. Then, we used the Debian package `iproute` and followed the instructions in section 5.3 of the "Linux Advanced Routing & Traffic Control HOWTO."[1]

4.3 Packet capture

To record our data, we used `tcpdump` to log to a series of packet capture files. These files (often also called "`pcap` files," for the library that writes them) are a common format readable by a variety of programs, which meant we could later feed this data into other programs for analysis.

We also recorded packets using Snort (see section 5.1) to a MySQL database. However, during peak traffic times, with nearly 200 packets observed per second, the MySQL server was unable to handle the load, and hence dropped nearly all of the traffic. As such, we were happy that the raw tcpdump files were available, as they allowed us to reconstruct the events afterwards.

5 Tools review

We asked several sources for recommendations of tools for analyzing data gathered from darknets, in addition to doing research ourselves. These sources varied from personal contacts in MIT and other universities to UNISOG[1] to Zanshin Security[2]. The software that was proposed varied from mildly useful to impossible to use; many suffered from lack of useful statistics or non-intuitive and under-documented interfaces. There was certainly no one "Darknet tool" we could feed all our packet capture files to with an "Analyze this!" button.

Our analysis of the tools — in the context of darknets — follows, sorted roughly from most useful to least useful.

5.1 Snort
http://www.snort.org/

Snort is an intrusion detection system that uses pre-defined rules to analyze incoming network traffic. A set of rules is included with the software and additional custom rules can be added as

[1] UNIversity Security Operation Group, a mailing list "intended for university and other academic institution system operators to discuss security issues specific to the academic environment."

[2] Zanshin Security is a small security consulting firm based in Boston, MA.

well. Snort processes network traffic or tcpdump files and logs packets that match these rules into a database.

For non-darknet intrusion-detection, this is potentially useful, though the rules that come with the program are very limited. The data we obtained with the rules that were included with snort showed that 99% of our alerts were simply ICMP requests. The included rules seem to be very specialized, focusing on specific attacks that happened and looking for characteristics of those attacks in each packet. However, for the most part, snort as originally configured gave us no useful information.

The database logging feature, however, was quite useful, and we added a rule that would catch all packets. This allowed us to perform database queries on all packets received so that we could develop some of our own tools for analyzing the data (see section 6.3). The ability to state that *all* packets were of note was the most important feature of snort.

5.2 ACID
http://acidlab.sourceforge.net/

ACID is the Analysis Console for Intrusion Databases. It uses snort's database to present the information from the database in a graphical format using a web page. It can summarize the alerts seen, give details on each individual alert, and generate graphs.

Because the default rules snort files do not provide useful groupings, ACID by default is also not very useful. Given additional comprehensive snort rules, however, ACID could prove its worth. However, without the effort of creating large numbers of customized snort rules, ACID is of limited utility. Furthermore, ACID was most probably not designed for the sheer volume of data in question — it became unusable after just a few days, because of the size of the database.

5.3 CoralReef
http://www.caida.org/tools/measurement/coralreef/

CAIDA, the Cooperative Association for Internet Data Analysis, has developed the CoralReef software suite to analyze network traffic, and uses it as the basis of their network telescope monitoring system. They use this to monitor their network, which they proudly claim occupies 1/256th of the public Internet.

We only had only a short amount of time to investigate this tool, and were unable to compile it, though the demonstration of the software on CAIDA's web site looked quite promising. Unlike the other tools which only gave limited general information, CoralReef can also sort information by tuples, which can give much more insight into the traffic received.

5.4 ntop
http://www.ntop.org/

ntop shows a summary of network usage, and summarizes data by protocol via a web interface. It displays traffic per port and traffic per host, and also flags hosts it considers suspicious based on the amount of traffic they have.

ntop turned out to be a useful program to get general summary information from, but did not show us the more interesting trends that we found with our own programs. It also had the problem that when we attempted to feed it a 2 GB packet log, it took up all 2 GB of memory and 3.5 GB of swap. At that time, the ntop webserver was very non-responsive (likely because of the large amount of swap used), and eventually the entire computer froze and had to be rebooted. We later repeated this process with a smaller 228 MB file covering a 24-hour period, and ntop was much more responsive.

5.5 Argus
http://www.qosient.com/argus/

Argus is the Audit Record Generation and Utilization System. Several people recommended using Argus to gather our data. Argus runs with a server-client model; the `argus` program gathers data, and others connect to it to report and analyze the data. For instance, the the `ra` program connects to `argus` and spews the data, causing `argus` to act like a tcpdump proxy.

These programs that analyze the data, however, are poorly documented on both the web site and the `man` pages (in some cases, not even having `man` pages). Because of this, we ended up not using Argus, especially since it seems that it can only provide general data summary information (if we were to figure out how to use these programs).

It is, however, considered somewhat of a standard and three people independently recommended it to us, so given additional time, it may be worth investigating further.

5.6 Other tools

Other tools that were recommended included Sentaurus[3], Snortsnarf[4], Firestorm[5], and Prelude[6]. Sentaurus and Snortsnarf provided functionality that other tools already gave us. Prelude added no functionality over Snort for darknet purposes.

Even so, these tools may be worth looking into again in the future, either as an alternative to some of the ones we used, or when they are more fully developed.

[3] http://www.demarc.com/products/sentarus/software/
[4] http://www.snort.org/dl/contrib/data_analysis/snortsnarf/
[5] http://www.scaramanga.co.uk/firestorm/
[6] http://www.prelude-ids.org/

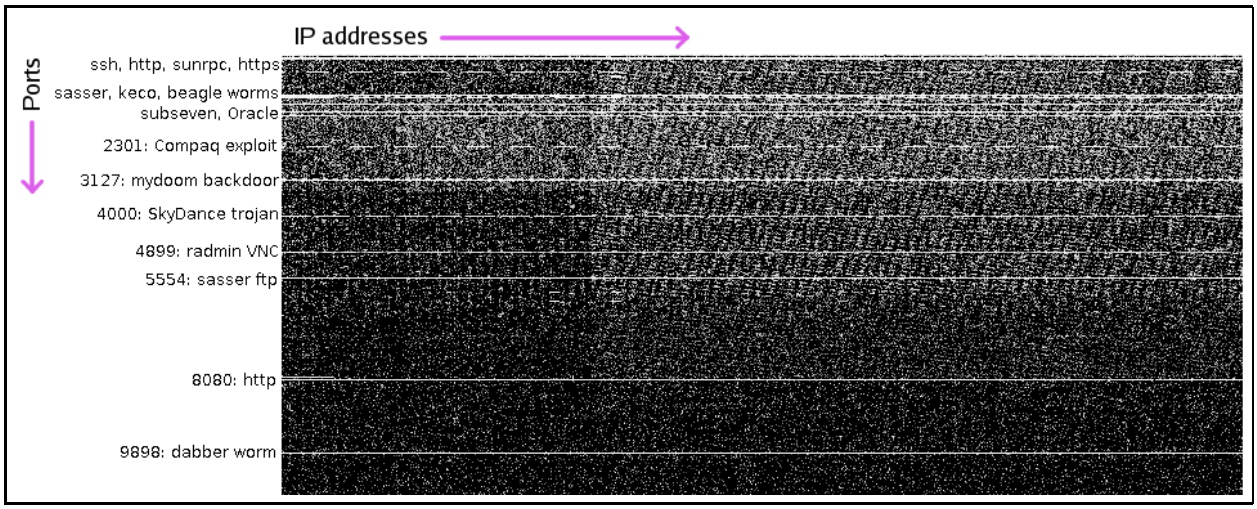

Figure 1: TCP activity, broken down by destination host and port

6 Results

When we started trying to actually make sense of the large pile of data we had amassed, we found that the tools available weren't able to show us very interesting trends. We were able to get some general information out of the basic tools, but ended up writing several Perl scripts to graph data points to actually analyze and visualize our data. With this visualization, we could actually gain insight into the data and find things that general summaries just couldn't tell us.

6.1 Basic statistics

Our darknet received 12 packets per second, on average, with peak traffic bordering on 200 packets per second. Over the period of 10 days, we received approximately 16 million packets (2^{64}), totaling more than 2GB of data.

Figure 1 shows the total traffic we received to our darknet over several weeks, with more traffic being represented by brighter areas. The vertical axis marks ports (lower-numbered at the top, higher-numbered at the bottom), and the horizontal axis marks IP addresses (lower to the left, higher to the right). Specific ports of note are labeled.

Table 1 is a list of the top ten TCP ports by activity in our darknet, along with the services, exploits, or malware associated with each[7]. Table 2 shows a few other common TCP ports receiving high amounts of traffic.

Table 3 shows the top ten sources of UDP port 137 (NetBIOS, or Windows file sharing) traffic come from a variety of geographic locations.

[7]Information gathered in part from DShield.org[3]. See Appendix for more information

TCP		UDP	
1433	Microsoft SQL server	137	NetBIOS Name Service
9898	Dabber worm backdoor	38293	Norton AV (host discovery)
5554	Sasser FTP server	1026	Windows messaging
1023	Sasser	1027	Windows messaging
80	http, many trojans	161	SNMP
4899	Windows remote admin	9191	Windows telnet overflow backdoor
22	ssh	4672	eMule file sharing
8080	http, some trojans	53	DNS
20168	Lovgate worm	11218	unknown
4000	SkyDance worm	9212	unknown

Table 1: Top 10 TCP and UDP ports, by number of packets

Rank by activity	Port	Use
12	443	https
20	111	Sun RPC (Remote Procedure Call)
27	21	ftp, AudioGalaxy, many trojans
31	1	TCP port service multiplexer[5]
38	31337	BackOrifice, many other trojans

Table 2: Other well-known commonly attacked ports

6.2 Analysis

One of the first things that we quickly noticed is that there is a visible diurnal cycle of traffic, as seen in Figure 2. There are vertical regions of higher brightness — these correspond to more traffic. These patterns brighten and repeat over period of a day. This may be because of attackers attempting to avoid people with day-jobs or attackers in other time zones scanning during their daytime hours.

Each day at 11:30am EST, as Figure 3 shows, we get a huge spike in traffic. This traffic targets ports 1023, 5554, and 9898, the first two of which correspond to the Sasser worm, and the last of which correspond to the Dabber worm backdoor. This is obviously a case of a worm awakening at a pre-specified time and bombarding the Internet with packets, looking for vulnerabilities. There is a much smaller spike an hour later, at 12:30pm EST, is a smaller peak — those hosts whose clocks are off by an hour.

The host on our darknet that got the most traffic was overwhelmingly 18.x.252.128. This seemed a bit unusual, as one would expect traffic to be approximately equally distributed at all the hosts. After investigating, we found that the traffic this host was getting was backscatter — packets being sent to this host in response to spoofed packets supposedly originating from this IP address. The source of all these packets were IP addresses located in China, so it appears that a denial of service attack was aimed at various machines in China.

Another noticeable pattern is the set of diagonal lines highlighted in Figure 4. Some of the machines

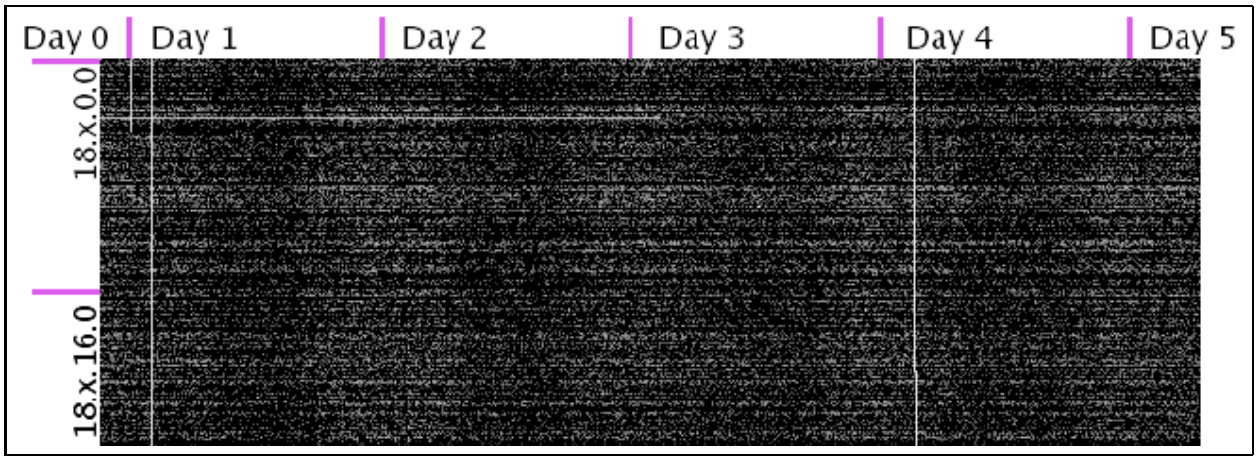

Figure 2: Daily variations in ICMP traffic

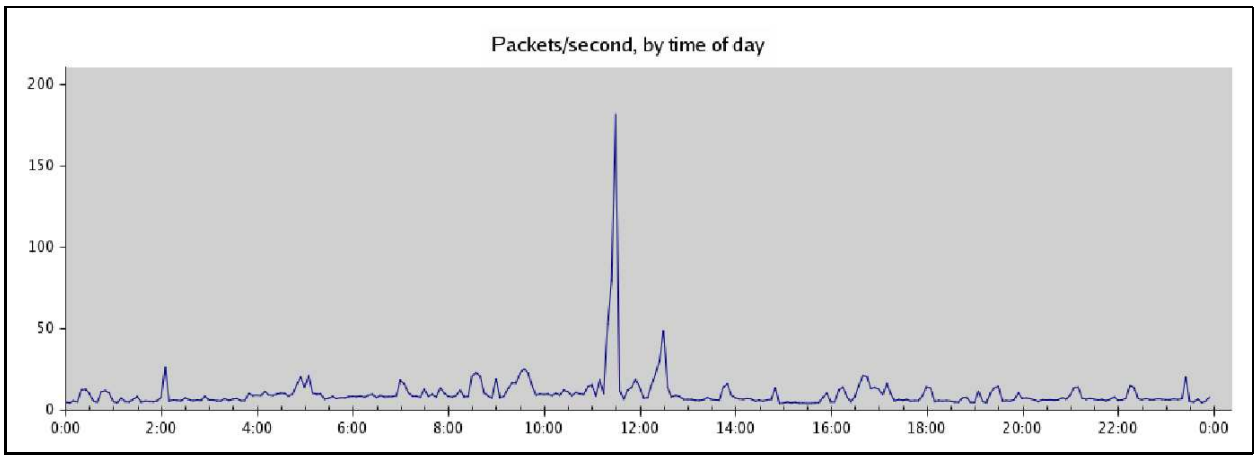

Figure 3: Average weekly traffic patterns show a clear event at 11:30am EST

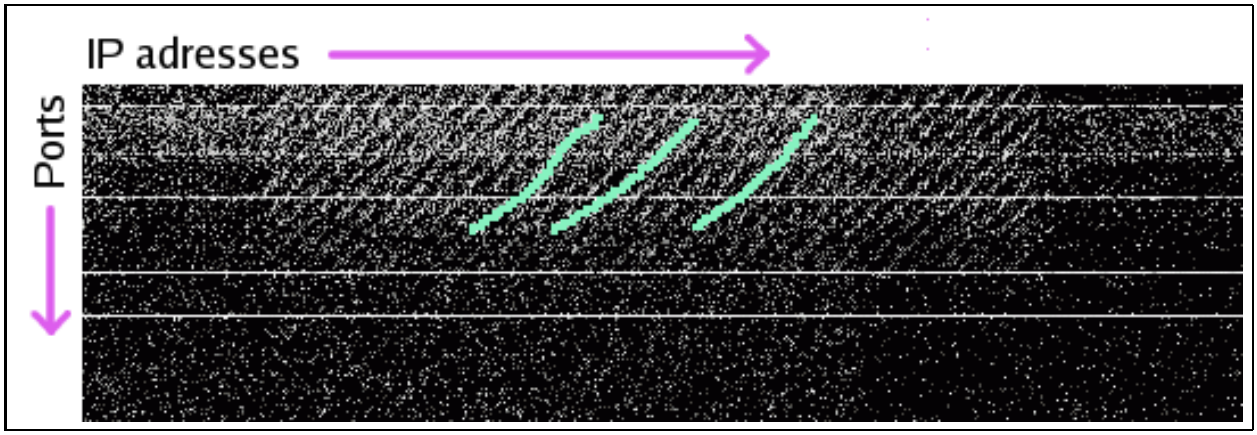

Figure 4: Diagonal patterns in port and host attack patterns are indications of a poor randomizer at work

IP address	Location
4.226.39.193	Level 3 Communications, Broomfield, CO
69.19.20.18	Hughes Network Systems, Germantown, MD
201.225.190.37	Cable & Wireless Panama, Panama
62.5.133.162	Moscow, Russia
81.40.184.43	Telefonica de Espana, Spain
220.107.163.236	Open Computer Network, Japan
200.44.180.91	CANTV Net, Caracas, Venezuela
62.42.208.102	Madrid, Spain
66.255.146.154	USLEC Corp, Charlotte, NC
210.212.95.66	CGM Data Networks, India

Table 3: Hosts generating the most requests on UDP port 137, NetBIOS

probing our darknet were picking random IP addresses and ports to scan. They were, however, using a poor randomizer, resulting in the diagonal patterns.

We were originally granted a much smaller, class C darknet. This was unsufficient; the darknet's ability to detect events is proportional to the fraction of the public Internet that it occupies. As such, interesting trends are simply invisible with a smaller class C network space; several of the randomized scans, which covered nearly half of the class B, would have been indistinguishable from background noise. A large quantity of the backscatter observed (such as that directed at the unfortunate 18.x.252.128), would have been completely missed because we might not have been observing the correct subnet.

6.3 Perl scripts

By and large, most currently-available tools are not tailored to darknets — they do not assume that all traffic is malicious, in general. In fact, many are forced to err on the side of information security, and *not* log borderline suspicious packets, as they may be real data. Another common problem with the currently-available tools is that they are simply not suited to the volume of information which flows through a class B darknet.

Due to the lack of available tools which seemed to do an adequate job of visualizing the data to the darknet, we decided to write our own analysis tools. These were Perl scripts which used the database written to by snort. This database contained all of the information that was included in the tcpdump capture files, as snort was set up to log packets that didn't fit any established rule. This is possible in a darknet, but not in normal networks, where innocent traffic should *not* be logged.

The database provided a powerful tool to aggregate and index information. Tools exist that enable direct reading and parsing of tcpdump files, but the database allowed some of difficult work of aggregation to be performed by a program specifically designed for such. A customized set of indexes were added to the database to further improve search speeds. Despite this, query results were often too big to process at once, and needed to be split into several chunks. It was also not

unusual for particularly complex queries to take longer than an hour. The results, however, were highly useful in our data analysis; all of the preceding figures and tables were generated wither by direct database query, or by a suite of Perl scripts that generated images based on the data returned by such queries.

7 Future work

There's a lot more work we would like to accomplish. For one, we only based our analysis on IP, TCP, and UDP packet headers, and did not look at the more telling bits of the packets — the TCP or IP flags, the TCP sequence numbers, of the payload data. By doing this we would be able to detect shellcode and buffer overflow attacks, possibly identify and differentiate individual attackers and scans. In addition, although the Perl scripts provided useful information, it is already old by the time it is compiled. What would be even better would be a program that can perform real-time analysis and reporting. An even more applicable feature of this program would be the capability of generating packet signatures on the fly, to be able to categorize data, and hence realize the appearance of new worms based on their inability to fit pre-existing signatures[8].

8 Conclusion

Darknets provide a unique way of investigating the constant tumult of malicious background traffic that exists on the Internet. By observing network trends close to home, one can discover properties of remote attackers, to be better able to defend hosts against them. It also provides a "pulse" on the attacks that currently exist in the wild — their distribution, their propagation speed, and identifying signatures. While the only literal requirement is free IP addresses, and the results can be quite revealing, there are hidden costs to setting up a darknet — be they finding or writing proper analysis software, or storing the huge quantities of data. The cost is well worth the ability to take a bird's eye view of would-be intruders in their natural setting, however.

9 Acknowledgements

This project could not have happened without a generous network allocation from Jeff Schiller. We would also like to thank Zanshin Security for their assistance — their technical advice and introductions were invaluable. Thanks also to Chris Lesniewski for office space.

[8] If it comes with a beautiful dappled pony with cream-white mane and tail and a gilded harness, we'll be set for life.

References

[1] Bert Hubert, et al. Linux Advanced Routing & Traffic Control HOWTO, 2002. http://lartc.org/howto/.

[2] Team Cymru. The Team Cymru Darknet Project, 2004. http://www.cymru.com/Darknet/.

[3] DShield.org. DShield.org, 2004. *Handler's Diary* and port reports. http://www.dshield.org/.

[4] Dan Kaminsky. Black Ops of DNS, 2004. Presentation slides from The Black Hat Briefings 2004. http://www.doxpara.com/dns_bh/Black_Ops_DNS_BH_files/v3_document.htm.

[5] Network Working Group M. Lottor. RFC 1078 - TCP port service Multiplexer (TCPMUX), November 1988. http://www.faqs.org/rfcs/rfc1078.html.

[6] Networks Associates Technology, Inc. McAfee Virus Information Library, 2004. http://vil.nai.com/.

[7] SecurityFocus. Multiple Vendor Sun RPC xdr_array Buffer Overflow Vulnerability, July 2002. bugtraq ID 5356. http://www.securityfocus.com/bid/5356/info/.

Appendix

This appendix briefly covers various services (how they can be used in a naughty manner), exploits, vulnerabilities, etc. associated with the top ten TCP/UDP ports getting traffic on our darknet.

Port 80 used to serve web pages; often not blocked by firewalls, so a popular port for trojans

Port 8080 used as an alternate or secondary port for serving web pages

Dabber a worm that spreads by further infecting Sasser-infected machines; also acts as a TFTP server and sets up a backdoor[6]

DNS a service with some BIND vulnerability attacks, but most darknet traffic is possibly from abusing DNS machines for their caches[4]

eMule P2P software that allows users to publish IP addresses and ports of a peer (which may result in scans)

Microsoft SQL Server some traffic from an Slammer, an exploit worm from January 2003 still in circulation

MyDoom an email virus that contains its own SMTP engine to construct outgoing messages, the ability to copy itself to mapped drives, a backdoor component, a denial of service payload, and a payload of deleting files[6]

NetBIOS Name Service used by Windows machines to find names of other Windows machines it can set up file sharing with. Traffic to this port (137) can be from misconfiguration or malice. A source port greater than 1024 suggests malware and not misconfiguration[3]

Norton AntiVirus host discovery port 38923/UDP is used by a Norton AntiVirus application to perform a network discovery task, but is likely being used by an adversary as an unclever network scanner [3]

Sasser worm that spreads by exploiting a Windows vulnerability (MS04-011 - buffer overflow in `lsass.exe`); creates a remote shell and FTP server (for newly infected hosts to download copies of the worm from) [6]

SNMP protocol used for remote management of network devices; traffic to darknet probably consists of misconfiguration and script kiddies attempting to exploit an old vulnerability

SunRPC this remote procedure call suffers from many buffer overflow vulnerabilities[7]