“DeadHat.B Analysis”

By Patrik Sternudd

Written during Spring 2004, as a part of my GCIA Practical Assignment. Revised September 2004.

Table of Contents

1. Summary

The attack is a worm targeting backdoors left in place by various MyDoom malwares[1]. Symantec's name is W32.HLLW.Deadhat.B [25]. It is reputed to have several attack vectors, but my main concern is its method of portscanning and then uploading a file for execution. When executed by the MyDoom backdor, it inactivates the MyDoom binary, and adds itself to the registry to get automatically started after a system reboot. As its name indicate, it targets Win32 systems.

It does have some characteristics that provides for easy identification:

- It scans TCP ports 1080, 3127, 3128, 10080.
- Source port is usually fixed to 22002.
- It establishes a new backdoor on port 2766.
- The TTLs are varying wildly between packets.

The worm does not cause any immediate damage (such as destroying data). However, the scanning seem to be quite network intense, and might cause denial of service conditions, especially at sites with limited bandwidth.

2. Network Design

The detect was captured at an undisclosed site. Below (figure 1) is a sanitised network diagram showing involved hosts. This section effectively replaces “Source of trace” for the remaining two detects. Detects are recorded by the Snort sensor which is running Snort [0] version 2.1.0 on a GNU/Linux system.

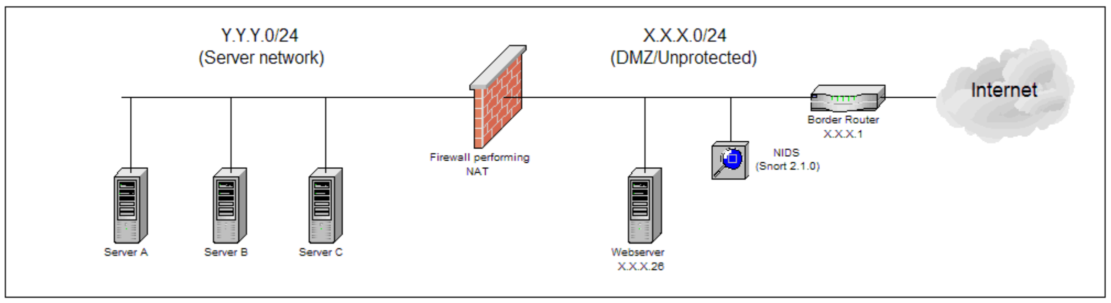

Fig 1 Network layout at an undisclosed site

Please be aware that IP addresses has been obfuscated where appropriate to protect the company in question.

1 It is getting harder and harder to say if something is a virus or a worm since they use techniques from both categories. Thus the term malware is becoming more and more favoured. For my part, I still think a worm is something that propagates through the network without user intervention (such as clicking on attachments or viewing it in the preview pane), so that will be my meaning whenever I use the term 'worm'.

3. It began as a simple Proxy Scan

The following detects where found in the Snort alert file. Well, in fact, several more of these were found, these are only the first four in a sequence of about 85.

```
[**] [1:615:5] SCAN SOCKS Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/31-11:55:18.474173 218.5.20.107:22002 -> X.X.X.2:1080
TCP TTL:108 TOS:0x0 ID:64668 IpLen:20 DgmLen:40
******S* Seq: 0x4812  Ack: 0x6F7D  Win: 0x7A67  TcpLen: 20
[Xref => http://help.undernet.org/proxyscan/]

[**] [1:618:5] SCAN Squid Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/31-11:55:18.674354 218.5.20.107:22002 -> X.X.X.2:3128
TCP TTL:109 TOS:0x0 ID:64680 IpLen:20 DgmLen:40
******S* Seq: 0x3789  Ack: 0x69C0  Win: 0x7EC9  TcpLen: 20

[**] [1:615:5] SCAN SOCKS Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/31-11:55:18.874769 218.5.20.107:22002 -> X.X.X.3:1080
TCP TTL:122 TOS:0x0 ID:64690 IpLen:20 DgmLen:40
******S* Seq: 0x2700  Ack: 0x6403  Win: 0x32B  TcpLen: 20
[Xref => http://help.undernet.org/proxyscan/]

[**] [1:618:5] SCAN Squid Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/31-11:55:19.074799 218.5.20.107:22002 -> X.X.X.3:3128
TCP TTL:123 TOS:0x0 ID:64700 IpLen:20 DgmLen:40
******S* Seq: 0x7476  Ack: 0x691  Win: 0x4D56  TcpLen: 20
```

The following rules are included in the scan.rules file, which is part of the installation:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 3128 (msg:"SCAN Squid Proxy attempt";
    stateless; flags:S,12; classtype:attempted-recon; sid:618; rev:5;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 1080 (msg:"SCAN SOCKS Proxy attempt";
    stateless; flags:S,12; reference:url,help.undernet.org/proxyscan/;
    classtype:attempted-recon; sid:615; rev:5;)
```

The rules checks for inbound TCP packets with the SYN flag set (ignoring ECN bits), and does not care about prior state of the session (this is done with the *stateless* keyword).The packets the rules are triggering on is showed in figure 2.

```
11:55:18.474173 IP 218.5.20.107.22002 > X.X.X.2.1080: S
11:55:18.674354 IP 218.5.20.107.22002 > X.X.X.2.3128: S
11:55:18.874769 IP 218.5.20.107.22002 > X.X.X.3.1080: S
11:55:19.074799 IP 218.5.20.107.22002 > X.X.X.3.3128: S
```

Fig 2 Packets triggering the Snort rules.

4. Indications of packet crafting

The one thing that really caught my interest was the source port, which was constant, and not increasing, as it should be for most normal applications. My initial thought was that I was seeing some sort of proxy scanning generated by a tool like nmap[2] which allows for user modified source ports.

It was at this stage I thought this might be worth to write about, and I took a look at the Intrusions mailinglist to see if it had already been covered. I found some other reports (these are covered in the correlations section), but no in-depth analysis. One of the posts there gave me an additional clue. It was not only the ports 3128 and 1080 that was being targeted, but also 3127 and 10080. I got access to the firewall logs, and managed to verify that this indeed was the case.

Excerpt from Firewall Log (the log does not show TTLs)							
Date	***Time***	***Action***	***Proto***	***SRC IP***	***DST IP***	***DPORT***	***SPORT***
6Apr2004	16:29:39	drop	tcp	62.201.64.66	10.1.2.2	3127	22002
6Apr2004	16:29:39	drop	tcp	62.201.64.66	10.1.2.2	1080	22002
6Apr2004	16:29:39	drop	tcp	62.201.64.66	10.1.2.2	10080	22002
6Apr2004	16:29:39	drop	tcp	62.201.64.66	10.1.2.2	3128	22002

A new snort rule was added to capture the complete scan. The complete scan pattern is shown in figure 19. The TTL values are really interesting.

```
07:34:12.514505 IP (ttl 114, length: 40) 192.168.1.5.22002 > X.X.X.2.3127: S
07:34:12.568048 IP (ttl 111, length: 40) 192.168.1.5.22002 > X.X.X.2.1080: S
07:34:12.645453 IP (ttl 127, length: 40) 192.168.1.5.22002 > X.X.X.2.10080: S
07:34:12.712233 IP (ttl 124, length: 40) 192.168.1.5.22002 > X.X.X.2.3128: S

07:34:12.785866 IP (ttl 109, length: 40) 192.168.1.5.22002 > X.X.X.3.3127: S
07:34:13.005955 IP (ttl 125, length: 40) 192.168.1.5.22002 > X.X.X.3.1080: S
07:34:13.098609 IP (ttl 114, length: 40) 192.168.1.5.22002 > X.X.X.3.10080: S
07:34:13.232869 IP (ttl 119, length: 40) 192.168.1.5.22002 > X.X.X.3.3128: S
```

Fig 3 Scanning pattern – notice the TTLs.

I do expect the TTL to vary, but not that much. It would indicate an incredibly unstable connection. And when I see the same pattern from three different locations (not showed in the trace), I go “hmm!”. The word “crafted” comes to mind. But why? Since someone obviously thought it worth the effort, there is probably some reason behind it. My guess is that it is supposed to stop us analysts from pinpointing the location, or distance. It might be wrong, but it is the only thing I can come up with right now.

It is always worth to ask oneself whether the source adress is spoofed or not. This is even more important when there are evidence of other IP header fields being crafted. At this point it was hard to make the call, but subsequent analysis showed that this was a worm attempting to spread, and it was using TCP to communicate. The network traces confirmed that a two-part communication took place. As such,

2 Available from http://www.insecure.org/nmap [26]

my conclusion must be that this traffic not is being spoofed. This conclusion was later verified when running the worm in a contained test environment.

5. Providing some bait

I got curious, and let a host listen on port 3127 over a night, and captured all data with a sniffer. The scan hit, my host dutifully answered SYN-ACK, and got a RST back. Immediately after, another connection was made from the same source, but now with a normal source port (2241). Figure 20 shows the beginning of the conversion (the TWH has already taken place).

```
01:02:49.827280 IP 67.97.205.104.2241 > X.X.X.X.3127: P 1:2(1) ack 1

0x0000   4500 0029 ba90 4000 7306 XXXX 4361 cd68        E..)..@.s.@*Ca.h
0x0010   XXXX XXXX 08c1 0c37 79e3 c5e2 581c 190b        .B9....7y...X...
0x0020   5018 faf0 XXXX 0000 8500 0000 0000             P.............

01:02:49.827374 IP X.X.X.X.3127 > 67.97.205.104.2241: . ack 2 win

01:02:49.840318 IP 67.97.205.104.2241 > X.X.X.X.3127: . 2:1462(1460) ack 1

0x0000   4500 05dc ba91 4000 7306 XXXX 4361 cd68        E.....@.s.:vCa.h
0x0010   XXXX XXXX 08c1 0c37 79e3 c5e3 581c 190b        .B9....7y...X...
0x0020   5010 faf0 XXXX 0000 133c 9ea2 4d5a 9000        P........<..MZ..
0x0030   0300 0000 0400 0000 ffff 0000 b800 0000        ................
0x0040   0000 0000 4000 0000 0000 0000 0000 0000        ....@...........
0x0050   0000 0000 0000 0000 0000 0000 0000 0000        ................
0x0060   0000 0000 0000 0000 f000 0000 0e1f ba0e        ................
0x0070   00b4 09cd 21b8 014c cd21 5468 6973 2070        ....!..L.!This.p
0x0080   726f 6772 616d 2063 616e 6e6f 7420 6265        rogram.cannot.be
0x0090   2072 756e 2069 6e20 444f 5320 6d6f 6465        .run.in.DOS.mode
```

Fig 4 Data exchange after a scanned port proved to be open.

As can be seen, first thing after the TWH, there comes a packet (PSH) with only one byte: 0x85. In the next packet, the following bytes comes: 0x13, 0x3C, 0x9E and 0xA2. I am not sure what these are for, probably some kind of instruction or length. After those, however, things begin to clear up. The next two bytes is 0x4D and 0x5A. Or, as it is in ASCII: MZ (anyone hear bells ringing?) Oh, and the string "*This program cannot be run in DOS mode"* is there too.

I was able to extract a fully functional PE[3] from the tcpdump logfile. The file size was 56 832 bytes, which is consistent with Symantec's report.

The next thing that is supposed to happen is that the uploaded binary is to be executed by a backdoor already in place (netcat on a non-windows system was not obliged to comply with the wishes of the worm).

3 Portable Executable

However, if the file were to be executed on the Windows system, it would do several things, and few of them good (I set up an isolated network with a fresh Windows XP system to act as my sacrifice box).

Commence Scanning

Immediately after execution, it began the scanning. A certain pattern can be seen:

- It randomizes five sets of two bytes (denoted *X* and Y) used for the first two fields of the IP addresses. Then it begins to scan X.Y.0.0 on all four ports, continues to X.Y.0.1, and so on, for all five sets simultaneously.

- When X.Y.0.255 is reached, it goes over to X.Y.1.0 and starts over. In other words, each set contains 65,536 addresses. It spews out approximately one packet every 0.02 second.

- I do not know what happens when it reaches X.Y.255.255. I assume one of:
  - New randomize (this is where I would place my bet).
  - Fall silent.
  - Increase the second octet (and ultimately the first one).

- The initial analysis that the TTLs were forged was quite correct. In the lab, i.e. on the originating subnet, I saw no TTL greater than 141, and no lesser than 122. This is also where I verified it did not spoof the source addresses.

Shutdown MyDoom processes

A copy of MyDoom-A [27] was retrieved from a nearby virus quarantine, and was executed on the system (after the date was set to January[4]). A listening port on address tcp/3127 appeared.

Then the new executable was run again. Port 3127 disappeared. However, the files installed by MyDoom was left intact, and so was the registry entries). It only stopped the backdoor process.

My findings disagree with Symantec's description, as the MyDoom registry entries or files not were removed. However, the MyDoom-A process will be started before Deadhat.B after reboot, which means it will be shut down again.

No other versions of MyDoom were tested, but I do not doubt they would be stopped as well (why else scan for the other three ports?)

Ensure survival through system reboots

It copied itself to C:\WINDOWS\SYSTEM32\MSGSRV32.EXE and created a registry entry in the Run folder to reload after reboot.

4 MyDoom-A will not spread after February 12 [27]

Establish new backdoor

It closes one hole, but opens another. A new listening port is opened at tcp/2766. When connected to with telnet or netcat, it responds with:
“SSH-2.0-OpenSSH_3.7.1p2 \m/”

It looks almost correct. Except the fact that normal OpenSSH servers do not terminate with “\m/”.

And it is not SSH either, of course. When trying to connect with a SSH client, I get a "connection closed" message back.

Final observations

An odd thing: There were also scans going to the same target ports, but without TTL or source port crafting. Those packets were very few in comparision, but did still appear. Why? I have no idea. The TTL on these were the OS default of 128.

I am not very good at debugging executables myself (which is why it is very neat to have friends who are). It appears the running process has capability to detect if someone is debugging it. Result: the logged in XP user was suddenly logged off! Harmless, but still quite fun, if one indeed is allowed to say that about malware. This is probably due to the compression/encryption rather than the worm itself. According to Panda[5], tElock v0.98 was used for this purpose.

5 http://www.pandasoftware.com/virus_info/encyclopedia/overview.aspx?lst=det&idvirus=44590 [28]

6. Correlations

The scanning pattern has been reported in various postings from Ken Connelly, of which the below post is the first one (dated February 12):

http://cert.uni-stuttgart.de/archive/intrusions/2004/02/msg00105.html [29]

The first [public] query on what this traffic actually was came from Michael Schwartzkopff in February 20 (although only ports 1080 and 3128 were reported):

http://cert.uni-stuttgart.de/archive/intrusions/2004/02/msg00126.html [30]

Finally, Carl Gibbons reports the pattern as well (March 8):

http://cert.uni-stuttgart.de/archive/intrusions/2004/03/msg00032.html [31]

As the scanning is made by a worm, I felt it would not add anything to the analysis by attempting to correlate source addresses. It suffice to say that most of the scans I have seen comes from various ISP:s, distributed on several countries.

Symantec, in addition to most other antivirus vendors, has a description of this worm. Unfortunately, their analysis does not mention the portscanning behaviour more than to state "*Attempts to connect to sequential IP addresses on TCP ports 3127, 3128, and 1080 (ports that the Mydoom worm used)*". I see no mention of port 10080 (I am confident that it is the same binary, as it does have the same file size and otherwise matches the described behaviour).
Even so, Symantec has the most accurate description I have found.

http://securityresponse.symantec.com/avcenter/venc/data/ (line wrapped)
w32.hllw.deadhat.b.html [25]

I did post an analysis on this worm on the intrusions list, in hope it could be useful to somebody before the completion of this practical. This posting is available in DShield's archive, URL listed below:

http://www.dshield.org/pipermail/intrusions/2004-April/007888.php [32]

7. Defensive recommendations

Some of the source addresses comes from the reserved RFC 1918 [33] address space. That tells us two things. There is not enough egress filtering being done near the source. This is outside of our control; however, as these packets also arrive unmolested, we can also also assume there is a lack of ingress filtering on our part. This in turn could mean there is insufficient egress filtering as well. The recommendation, then, is to ensure that both egress and ingress filtering is performed (at the very least, anti-spoofing measures should be employed – there is no reason to allow RFC 1918 addresses coming into the network, nor should any source addresses not part of the network be let out onto the Internet).

Another thing would be to create new Snort rules, and configure them to watch for internal system responding on ports 1080, 3127, 3128, 10080 and 2766 (Any Squid proxies would have to be exempt for the ports they actually use).

It might also be prudent to watch for such traffic going out. This could indicate an internal compromise (by way of IRC, P2P, or other means[6]), or simply an internal user scanning external systems (which is a bad thing in almost all cases).

Finally, as this worm targets backdoors left in place by email borne viruses, it would be in an organisation's best interest to ensure all Microsoft Windows systems have up-to-date antivirus definitions, and that its employees are receiving on-going security education. A policy concerning email attachments should also exist.

8. Multiple choice question

TTL can be used in many ways. Which of the following statements is true?

A) TTL is defined in both the TCP and UDP headers
B) The TTL values will always be identical for all packets within a TCP session.
C) 128 is a common value for packets leaving a host (i.e. the initial value)
D) A common misconception is that TTLs is used by the Windows tracert command.

Correct answer: C

(I am attempting to trick the unwary to answer “D”. Windows tracert does differ, truly enough, but only because it is using ICMP instead of UDP high-ports. TTL is manipulated in the same way on both platforms. Many UNIX traceroute commands has an option to use ICMP as well)

6 This is covered in more detail by the antivirus vendors.

9. References

[0] ”Snort”. URL: http://www.snort.org/ (25 Jun 2004)

[1] “Part 2 data” URL: http://www.incidents.org/logs/Raw/2003.12.15.tgz (24 Jun 2004)

[2] Martin, Ian. “SANS GCIA Practical Version 3.3”. Jul 2003.
URL: http://www.giac.org/practical/GCIA/Ian_Martin_GCIA.pdf (24 Jun 2004)

[3] “oui.txt”. URL: http://standards.ieee.org/regauth/oui/oui.txt (24 Jun 2004)

[4] “IEEE OUI and Company_id Assignments”
URL: http://standards.ieee.org/regauth/oui/index.shtml (24 Jun 2004)

[5] ”GCIA Raw data README”. Apr. 2004.
URL: http://www.incidents.org/logs/Raw/README (24 Jun 2004)

[6] Postel, Jon (editor). ”Internet Protocol”. Sep 1981.
URL: http://www.ietf.org/rfc/rfc0791.txt (24 Jun 2004)

[7] “Nessus”. URL: http://www.nessus.org (24 Jun 2004)

[8] “Source routed packets”. 2003.
URL: http://cgi.nessus.org/plugins/dump.php3?id=11834 (24 Jun 2004)

[9] “CAN-1999-0510”. 1999.
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0510 (24 Jun 2004)

[10] “Source Routing”. URL:
http://www.iss.net/security_center/advice/Underground (line wrapped)
/Hacking/Methods/Technical/Source_Routing/default.htm (24 Jun 2004)

[11] Novak, Judy et al. “TCP/IP for Intrusion Detection”. The SANS Institute. 2004.

[12] “IDS418 "SOURCE_ROUTE_UDP_LSRR"”.
URL: http://www.whitehats.com/info/IDS418 (24 Jun 2004)

[13] “CVE-1999-0909”. 2000.
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0909 (24 Jun 2004)

[14] “Microsoft Windows IP Source Routing Vulnerability”. Sep 1999.
URL: http://www.securityfocus.com/bid/646 (24 Jun 2004)

[15] “NAI-Sep201999: Windows IP Source Routing Vulnerability”. Sep 1999.
URL: http://www.securityfocus.com/advisories/1761 (24 Jun 2004)

[16] Postel, Jon (editor). ”Transmission Control Protocol”. Sep 1981.
URL: http://www.ietf.org/rfc/rfc0793.txt (24 Jun 2004)

[17] ”OpenSSH Security”. URL: http://www.openssh.org/security.html (25 Jun 2004)

[18] Cheswick, William & Bellovin, Steven. “Firewalls and Internet Security Repelling the Wily Hacker”. Reading. Addison-Wesley. 1994. ISBN 0-201-63357-4.

[19] Sternudd, Patrik. “[Intrusions] LOGS: GIAC GCIA Version 3.4 Practical Detect Patrik Sternudd”. Apr 2004. URL: http://www.dshield.org/pipermail/intrusions/2004-April/007902.php (24 Jun 2004)

[20] Sternudd, Patrik. “[Intrusions] LOGS: GIAC GCIA Version 3.4 Practical Detect Patrik Sternudd (resubmitted)”. Apr 2004.
URL: http://www.dshield.org/pipermail/intrusions/2004-April/007945.php (24 Jun 2004)

[21] “Unicode 4.0.1”. Mar 2004. URL: http://www.unicode.org/versions/Unicode4.0.1/ (24 Jun 2004)

[22] “Windows Workstation Service Remote Buffer Overflow”. Nov 2003.
URL: http://www.eeye.com/html/Research/Advisories/AD20031111.html (24 Jun 2004)

[23] “Microsoft Security Bulletin MS03-049”. Nov 2003. URL:
http://www.microsoft.com/technet/security/bulletin/MS03-049.mspx (24 Jun 2004)

[24] “CAN-2003-0812”. 2003.
URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0812 (24 Jun 2004)

[25] “W32.HLLW.Deadhat.B”. Feb 2004. URL:
http://securityresponse.symantec.com/avcenter/venc/data/w32.hllw.deadhat.b.html (24 Jun 2004)

[26] ”Nmap”. URL: http://www.insecure.org/nmap/ (24 Jun 2004)

[27] “W32.Mydoom.A@mm”. Jan 2004. URL:
http://securityresponse.symantec.com/avcenter/venc/data/w32.novarg.a@mm.html
(24 Jun 2004)

[28] “Deadhat.B”. Feb. 2004. URL: http://www.pandasoftware.com/virus_info (line wrapped)
/encyclopedia/overview.aspx?lst=det&idvirus=44590 (24 Jun 2004)

[29] Conelly, Ken. “[LOGS] Summary of large-scale portscanning detects”. Feb 2004.
URL: http://cert.uni-stuttgart.de/archive/intrusions/2004/02/msg00105.html (24 Jun 2004)

[30] Schwartzkopff, Michael. “Source port 22002”. Feb 2004.
URL: http://cert.uni-stuttgart.de/archive/intrusions/2004/02/msg00126.html (24 Jun 2004)

[31] Gibbons, Carl. “Re: [LOGS] Summary of large-scale portscanning detects”. Mar 2004.
URL: http://cert.uni-stuttgart.de/archive/intrusions/2004/03/msg00032.html (24 Jun 2004)

[32] Sternudd, Patrik. “[Intrusions] Source port 22002 (Worm targeting MyDoom backdoors)”.
Apr 2004. URL: http://www.dshield.org/pipermail/intrusions/2004-April/007888.php (24 Jun 2004)

[33] Rekhter, et al. ”Address Allocation for Private Internets”. Feb 1996.
URL: http://www.ietf.org/rfc/rfc1918.txt (23 Jun 2004)

[34] IANA. “Special-Use IPv4 Addresses”. Sep 2002.
URL: http://www.ietf.org/rfc/rfc3330.txt (23 Jun 2004)

[35] eDonkey2000. URL: http://www.edonkey2000.com/ (24 Jun 2004)

[36] Rautiainen, Sami. “F-Secure Virus Descriptions : Adore”. Apr 2001.
URL: http://www.f-secure.com/v-descs/adore.shtml (24 Jun 2004)

[37] “F-Secure Virus Descriptions : SdBot”. Nov 2002.
URL: http://www.f-secure.com/v-descs/sdbot.shtml (24 Jun 2004)

[38] “GIAC Logs”. URL: http://isc.sans.org/logs/ (24 Jun 2004)

[39] “IDS177 "NETBIOS-NAME-QUERY”.
URL: http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids177&view=research (24 Jun 2004)

[40] Beardsley, Tod. “Intrusion Detection and Analysis: Theory, Techniques, and Tools”. May 2002.
URL: http://www.giac.org/practical/Tod_Beardsley_GCIA.pdf (23 Jun 2004)

[41] “eDonkey2000 FAQ”.
URL: http://www.edonkey2000.com/documentation/donkeyfaq.html (24 Jun 2004)

[42] “eDonkey2000 FAQ – General”.
URL: http://www.edonkey2000.com/documentation/clientfaq.html (24 Jun 2004)

[43] “Novell Protocols”. URL: http://www.protocols.com/pbook/novel.htm#NCP (24 Jun 2004)

[44] “Overview of NetStorage”. Jun 2002.
URL: http://developer.novell.com/research/appnotes/2002/june/03/a0206033.htm (24 Jun 2004)

[45] Gordon, Les. “Intrusion Analysis - The Director's Cut!”. Nov 2002.
URL: http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc (24 Jun 2004)

[46] “broadband » Forums » Stopping Spam » Vast amount of spam linked to one "company"”.
May 2004. URL: http://www.dslreports.com/forum/remark,10137492~mode=flat (24 Jun 2004)

[47] “Security Incidents: Re: Outbreak of a virus on campus, scanning tcp 80/6129/1025/3127”.
Apr 2004. URL: http://seclists.org/lists/incidents/2004/Apr/0063.html (24 Jun 2004)

[48] Ramakrishnan, et al. ”The Addition of ECN to IP”. Sep 2001.
URL: http://www.ietf.org/rfc/rfc3168.txt (23 Jun 2004)

[49] Newport, Brandon. “Level Two Intrusion Detection In Depth GCIA Practical Assignment”.
May 2001. URL: http://www.giac.org/practical/Brandon_Newport_GCIA.zip (23 Jun 2004)

[50] ”MySQL Reference Manual”. URL: http://dev.mysql.com/doc/mysql/en/index.html (25 Jun 2004)

[51] Sternudd, Patrik. “insert_delims.c”. Jun 2004.
URL: http://hem.bredband.net/flank/source/insert_delims.c (24 Jun 2004)

[52] Sternudd, Patrik. “extract_dbout.c”. Jun 2004.
URL: http://hem.bredband.net/flank/source/extract_dbout.c (24 Jun 2004)

[53] ”Ports Database”. URL: http://www.portsdb.org/ (24 Jun 2004)

[54] Reynolds, Joyce K (editor). ”Assigned Numbers: RFC 1700 is replaced by an On-Line Database”.
Jan 2002. URL: http://www.ietf.org/rfc/rfc3232.txt (25 Jun 2004)

[55] IANA. ”TCP and UDP Port Numbers” 19 Mar 2004.
URL: http://www.iana.org/assignments/port-numbers (25 Jun 2004)