Global Information Assurance Certification Paper

Miguel Dilaj
GIAC Security Essentials (GSEC) Certification Practical Assignment
Version 1.4b – Option 2
Submitted on Dec 16, 2003

The Unintentional Criminal: DDoS from the inside!

Summary

The criminals will most likely be using a compromised machine as an operations base from which they may launch a computer attack.
This paper will highlight the IT Security problems resulting from the economic constraints on an ISP in a developing country and of their impact everywhere. I will use the example of a Distributed Denial of Service (thereafter DDoS) attack to demonstrate the process of its detection, analysis and protection.
I will also discuss ways of raising the level of IT Security with minimum investment that could be attractive for low-budget companies.
At the end it will be concluded that the lack of budget is not a good reason to reduce the security level, thus jeopardizing the security and continuity of operation of everyone else, because there is always a way to enhance the security level using free or low-cost solutions.
It will also be clear the need to have IT staff with a sound understanding of the security process, even in small companies.

The Unintentional Criminal: DDoS from the inside!

Background

The incident described in this paper took place during the summer of 2002 in a medium-sized ISP in an important city in South America,.
I had a key role in the resolution of the problem, having been hired to discover the source of a problem and to solve it.

Following this situation, I posted its details (after proper sanitization) in the Spanish newsgroup news:es.comp.hackers in March 2002 (and reposted by request one year later). It was presented as a QUIZ, in order to improve the knowledge of participants of this group. The archives (in Spanish) of the QUIZ and the answers can be found in Google Groups, by searching for MEGA+QUIZ in this group:

URL: http://groups.google.com/groups?hl=en&lr=&ie=ISO-8859-1&scoring=d&q=MEGA+QUIZ&btnG=Google+Search&meta=group%3Des.comp.hackers

(Take into account that the URL was split in 2 lines above).

1. THE INITIAL SITUATION

1.1. The First Contact

In February 2002 I received an email from a friend who had attended a training course that I had given, telling me that his ISP was having serious connectivity problems, together with apparent strange behaviour in their main Cisco router. He was desperately looking for someone to find a solution.

He gave me the name and phone number of the ISP's Chief Technical Officer (thereafter CTO), as well as some tit-bits of information about the problem. The CPU usage in the main router climbed suddenly to 99% and, after a couple of seconds, connectivity to the upstream provider was lost. At this time, my friend had no other information, so I phoned the ISP's CTO. After a short introduction he confirmed the problem as it had been initially explained.

My first concerns were the model of the main Cisco router that was causing the problems and the IOS (Internetwork Operating System Software) version, as well as how much to charge for my consulting services. I didn't get any useful information about the network infrastructure on the phone and so I arranged an interview with the CTO that same afternoon. I also asked him to provide some network diagrams.
So far I had only learned that the router was a Cisco 2610 with IOS 11.3T and so I used the time looking for any known vulnerabilities in SecurityFocus' Vulnerabilities database.[1]

The information until now could indicate some kind of Denial of Service (hereafter DoS) attack. At least Cisco IOS 11.3T was vulnerable to 2 DoS attacks:

"Cisco IOS BGP Transitive Attribute Denial of Service Vulnerability"
BID: 2733 (published May 10, 2001).[2]

Cisco devices running IOS before version 12.1 (with some exceptions, see SecurityFocus for details) are vulnerable to a DoS caused by the impossibility for the IOS to handle an exceptional condition, when a faulty or crafted BGP-4 (Border Gateway Protocol version 4) UPDATE message containing an unrecognized transitive attribute. The result can be a crash of the IOS. Cisco devices not running BGP are not vulnerable.[3]
This was discovered due to faulty implementation of BGP by a vendor, and no DoS tool based on it has been reported to BugTraq so far, but the possibility to craft such packets has to be considered by anyone administering Cisco equipment (taking into account that BGP-4 uses TCP as its transport protocol, normally on port 179/tcp, and the standard defined in RFC-1771, crafting such packets must be trivial).
Cisco has provided upgrades for the affected versions of the IOS.[4]

"Cisco Local Interface ARP Denial of Service Vulnerability"
BID: 3547 (published November 15, 2001).[5]

Cisco devices running IOS up to and including 12.2 (again, see SecurityFocus for details) are vulnerable to a DoS from the local network caused by the improper

handling of a specially crafted ARP (Address Resolution Protocol) request claiming a different MAC (Media Access Control) address for the device's local broadcast interface. If the device is vulnerable, it will replace its own MAC address by the one supplied in the packet, and stopping to receive any more ARP requests. After a short time (when the ARP table information expires) connectivity will be lost for all clients.[6]
This was initially demonstrated during a Black Hat[7] conference, and no DoS tool based on it has been reported to BugTraq so far, but the possibility to craft such packets has to be considered by anyone administering Cisco equipment (crafting ARP packets is trivial, and can be done with tools like Packet Excalibur[8], the packetgenerator of the P.A.T.H. Project[9], and many others).
Cisco has provided upgrades for the affected versions of the IOS.

***Note**: there are two additional Denial of Service conditions on Cisco IOS 11.3T, under BID: 4132 (published Feb 12, 2002)[10] and BID: 8211 (published Jul 16, 2003)[11]. Both of those were discovered AFTER the problem discussed in this paper, so I briefly point to the resources where you can get more detailed information.*

The first one affects Cisco IOS up to and including version 12.2 (see SecurityFocus for details), and is caused by specially crafted SNMPv1 (Simple Network Management Protocol version 1) messages than can cause the device to reset. An exploit for that is available in BugTraq.[12]
Cisco has provided upgrades for the affected versions of the IOS.[13]

The second vulnerability affects Cisco IOS up to and including version 12.2 (see SecurityFocus for details), and is caused by a sequence of specially crafted IPv4 (Internet Protocol version 4) packets with specific protocol fields. There are exploits available in BugTraq, and also a discussion on how to use widely available tools like hping[14] to reproduce the DoS condition.
Cisco has provided upgrades for the affected versions of the IOS, and also some specific workaround directions.[15]

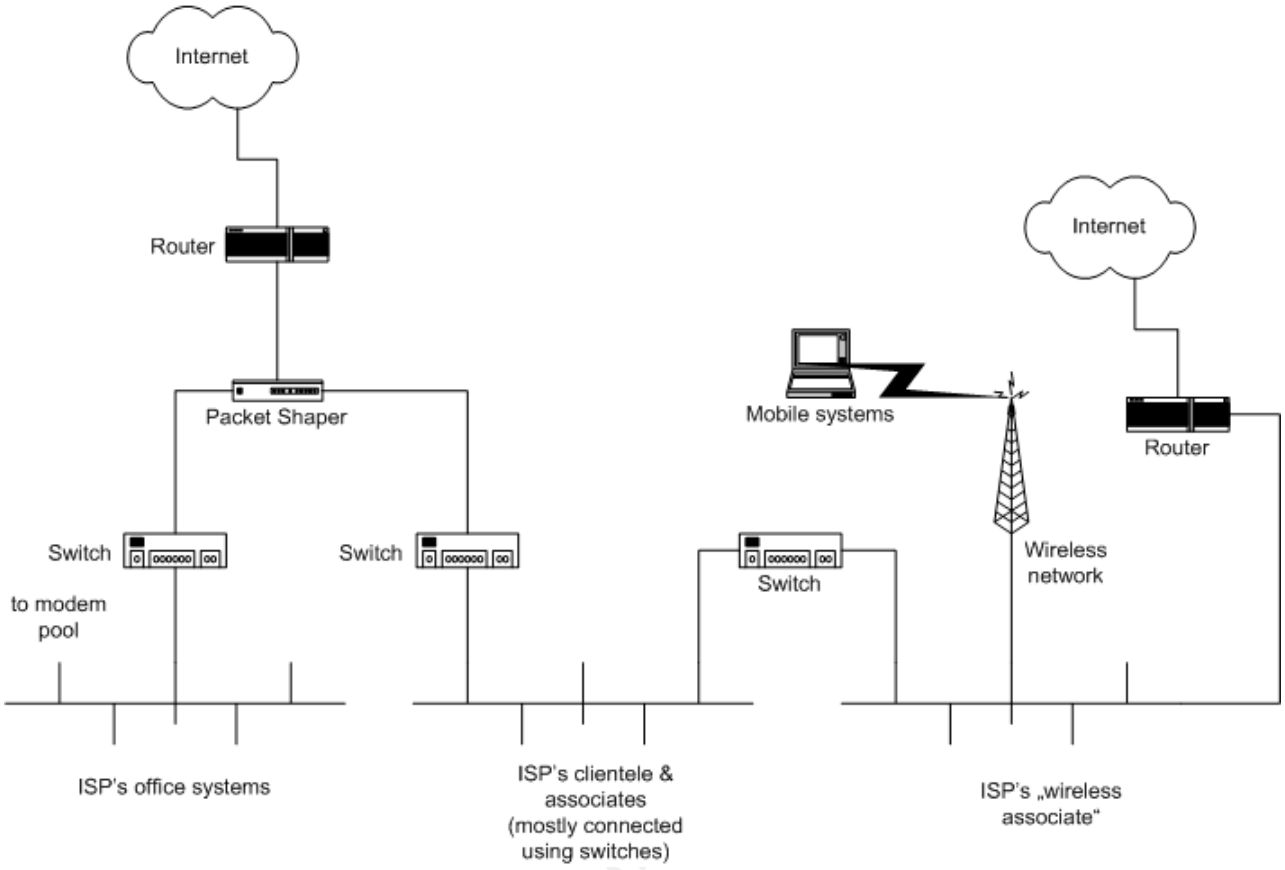
*SecurityFocus is a good reference site. **Use it!***

1.2. The Interview

During my interview with the CTO I told him that I needed to gather more information, either from existing logs or by continuous monitoring.

I explained the likelihood of a DoS attack and of the vulnerability of Cisco IOS 11.3T to 2 to such attacks, but I was unable to confirm my suspicions with so little information.

The CTO showed me some hand-drawn diagrams of the network. I was immediately shocked. I've tried to reproduce them here as well as I can remember them -this is not for the faint of heart.



The ISP Network

Most of the clientele of this ISP were small companies connected via LAN/WAN rather than individuals, connected via LAN/WAN. There were only a few dialup users (with dedicated lines) that used to connect to the modem pool located in the "ISP's office systems" branch.

Isn't there something missing? What about inserting this symbol somewhere? Perhaps you would use it in more than once?

This wonderful piece of network engineering was created using Microsoft Windows 2000 and different Linux distributions on the ISP systems, but almost anything imaginable on the ISP's clientele. Due to the fact that everything was connected to everything else simply with switches, the ISP and their clientele can be considered simply as a big and heterogeneous network. It seems that there was a merging with the "wireless associate" a short time ago, and this resulted in having 2 contact points with the Internet (2... if we don't discuss wireless issues). The 2 entry points were with the same upstream provider.

Let me discuss briefly the pitfalls of the way this network was interconnected.

First, when 2 or more networks are connected using switches or hubs, they simply become a bigger network.

I can hear someone saying: "Yes, but switches are far better for security than hubs, because in a shared network (using hubs) anyone can sniff all the traffic!".

That's partially true, but no cigar. It's true that in a shared network all traffic goes to all machines, and that anyone that was able to install a sniffer can capture all the traffic. It's also true that in a switched network all traffic (except broadcast) goes only to the destination system it was intended to be sent. BUT… switches do that collecting information about the MAC addresses of the networked devices, and creating tables to correlate that MAC address to the proper IP address. Then it's easy to determine to which port a certain packet has to be sent, the switch simply has to send to the proper MAC address!

What about if someone is able to modify the information stored in the tables in the switch? In this case the individual able to do the modification will also be able to direct any traffic (or ALL traffic) to his/her machine. This wonderful technique is named ARP poisoning, is widely known and there're a lot of tools to do that. Example tools are dsniff[16] (that also integrates a sniffer) and ARP0c/WCI[17] (one of my favourites, just add your preferred sniffer and stir).

There're some switches with the functionality to try to avoid spurious changes to the ARP table, but even in those case sometimes is possible to flood the switch with requests, thus causing a small "DoS" to the ARP caching functionality, when this happens, usually the switch will fail to work as a hub (to avoid disrupting connectivity).

Said the above, 2 or more networks that want to remain as 2 or more *separate* networks must be connected using routers or gateways properly configured.

Second, I already mentioned the lack of firewall(s). Firewalls are used both to separate and PROTECT networks, applying Access Control Lists (ACLs) at different levels depending on the implementation.

The most basic firewall is named a packet filter, and only allows or blocks the passage of network packets according to certain criteria (for example source or destination IP address, source or destination MAC address, source or destination port, etc., or a nice mix of all the above).

In our particular case, and as will be discussed shortly, proper ACLs in the existing equipment were not present, nor one or more dedicated firewalls. This lead to the possibility of anyone to send and receive packets, both internally and externally (as long as the upstream provider allows that).

A good network implementation should use several firewalls, at least one at each entry point from the Internet to block traffic with some invalid characteristics (for example invalid or private source IP address, loopback addresses, etc.). Only traffic destined to the addresses in the ISP's range should be allowed. All outgoing traffic must came from the ISP's address range (this is known as 'outgoing filtering' as is quite important in the issue discussed in this paper).

Then, there should be firewalls to segregate different portions of the ISP network. The network office should be private to the ISP staff, and not accessible from outside this area (not even from the ISP servers or the clientele). The servers that are deemed to be accessible from the Internet should be in what is called a DeMilitarized Zone (DMZ, see an example in the Linux IPCHAINS-HOWTO).[18]
The clientele range should be also separate and protection by a firewall is desirable (this must be flexible enough for the requirements of the clientele, but stringent enough to avoid "strange" packets coming out, for example packets with a source address not corresponding to the clientele range).

I started asking questions. It seemed that all the ISP servers that could be accessed directly from the Internet were updated with the same security patches, but we couldn't say the same for the routers, and of course nothing was clear about the systems of the clientele. The list of services provided on the ISP systems was extensive, covering HTTP, HTTPS, SMTP, POP3, IRC, DNS and many more.

The problem that motivated our interview was that the router on the upper left was losing Internet connectivity from time to time. This happened mostly at night and during such periods the CPU usage climbed to ~99%.

There was a WhatsUp Gold[19] monitoring host in the ISP's office network, but the logs were not very useful, other than showing the exact times of the periods in which the router lost connectivity.

A closer look at the Cisco router showed some basic ACLs, good enough to block an ICMP based DoS attack. The logs in the router itself didn't show anything abnormal, except the periods of losing connectivity. The router was not configured with egress filtering, so spoofed outgoing traffic was possible.

I didn't find anything else of any use to identify the problem. There were no more logs, no more filters/ACLs, and of course no Network Intrusion Detection System (hereafter NIDS). We decided to investigate the source of the problem before making any changes to the infrastructure.

Even whilst we were still looking for information the problem appeared again. Nothing worthwhile came from it, other than the fact that we were able to confirm the increase in CPU usage in the router with the '`show processes cpu`' command. The output at this time was probably similar to this one:

```
router#show processes cpu
      CPU utilization for five seconds: 94%/4%; one minute: 7%; five minutes: 6%
       PID Runtime(ms)   Invoked  uSecs    5Sec   1Min   5Min TTY Process
         1         873     32678     12   0.00%  0.00%  0.00%   0 Load Meter
         2        2527      1288   2533   0.83%  1.66%  0.48%   0 Exec
```

(…rest of output omitted for brevity…)

2. HOW WE PROCEEDED

2.1. Preparing the ground

It was essential to get more information and that it be sufficiently detailed to show all the traffic reaching the router just before and during the shortage periods.

I did this by connecting a sniffing system as close as possible to the router. I used a spare computer with Linux and Ethereal[20] at negligible cost. The main switch didn't have a monitoring port, and was not directly connected to the router (the traffic shaper was in the middle), so we decided to explore the readily available hardware, and ended by connecting a small hub between the router and the packet shaper. The Linux sniffing system was connected to the hub, so being able to capture anything. I configured the system with the latest security patches from the vendor and without any services active.

Note: *Just as a minor comment, the so called "Switched Port Analyzer" (SPAN) present in some Cisco Catalyst models and in some other brands and models is a very convenient functionality if you need to monitor (i.e.: sniff) anything on your network.*[21]

Ethereal is a very good and flexible network analyzer (for us, that means "more than simply a sniffer"). It can be downloaded for free, has versions for a wide range of Operating Systems, and can be used from the GUI or command line. The possibility to use it from the command line is making it a very powerful tool, even if the traffic has to be captured to do the analysis using the GUI (perhaps on a different machine). The syntax of the command line version of Ethereal (named `tethereal`) is as follows:

```
tethereal [ -a capture autostop condition ] ... [ -b number of
ring buffer files [:duration] ] [ -c count ] [ -d <layer
type==<selector>,<decode-as protocol> ]> [ -D ] [ -f capture
filter expression ][ -F file format ] [ -h ] [ -i interface ]
[ -l ] [ -n ] [ -N resolving flags ] [ -o preference setting ]
... [ -p ] [ -q ] [ -r infile ] [ -R display filter expression
] [ -s snaplen ] [ -S ] [ -t time stamp format ] [ -v ] [ -V ]
[ -w savefile ] [ -x ] [ -z statistics-string ] [ filter
expression ]
```

A bit messy? Don't worry, I'll try to highlight the most important switches:

`-h`
provides help options, show the version and exit

`-c count`
the number of packets to capture is specified

`-i interface`
interface to monitor, when there's only 1 interface card, this one is used
You can check the available interfaces with '`tethereal -D`'

`-v`
print tethereal's version and exit

`-n`
disables network object name resolution

`-p`
DON'T put the interface in promiscuous mode (used to capture traffic destined to the local machine only)

`-w savefile`
writes the packet data to '`savefile`' (this is raw data, no decoding of the information is saved)

`-x`
will print an hex[adecimal] and ASCII dump of the packet data

The above is just scratching the surface of the functionality of Ethereal, read the documentation and experiment in your lab.

Although the ISP was not an important one, I expected the level of traffic to be very high and so capturing the contents of all of the packets was not practical. I used the following command to capture only the headers. We left it running overnight with the following syntax, in order to send all packet decode information that's usually shown on screen to a log file:

```
# tethereal > logfile.headers
```

We did this because we had no pattern about the time of the shortages, except that they usually occurred at night. I instructed the night operator to press CTRL-C to stop the capture no more than 10 minutes later than a network shortage, to avoid capturing unnecessary traffic, and to have an idea of the time of the incident.

Nothing happened the first night, but when we repeated the exercise the following night, we managed to capture something quite interesting.

2.2. The Hunt

We got 30 MB of logs, covering the "hot" period from just before the problem, up to some minutes after everything returned to normality. I'll post here only 40 headers of those 30 MB, thus making life easier for the readers of this paper.

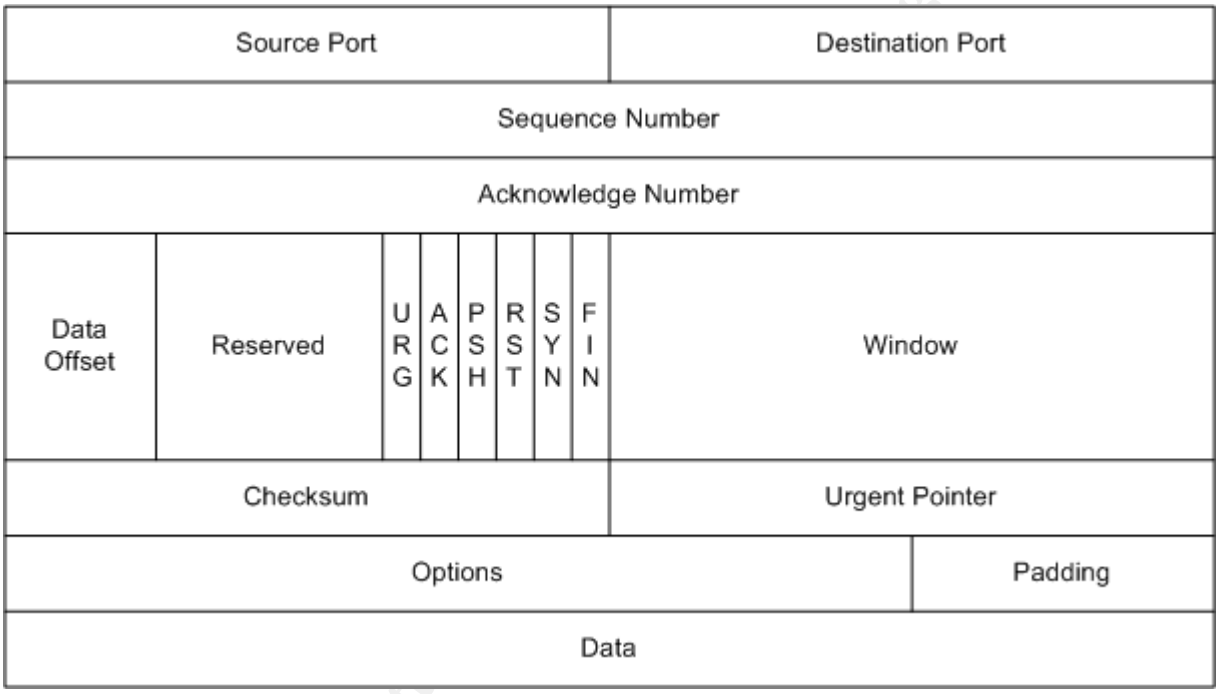
```
  1   0.000000 161.232.248.52 -> 200.206.aaa.bbb TCP 4218 > 40417 [ACK]
Seq=3452355670 Ack=0 Win=15987 Len=0
  2   0.000000 49.210.67.107 -> 200.181.ccc.ddd TCP 4474 > 2187 [ACK]
Seq=3469132886 Ack=0 Win=15987 Len=0
  3   0.000000 51.87.122.36 -> 200.206.aaa.bbb TCP 4730 > 60709 [ACK]
Seq=3485910102 Ack=0 Win=15987 Len=0
  4   0.000000 145.245.150.11 -> 200.181.ccc.ddd TCP 4986 > 57941 [ACK]
Seq=3502687318 Ack=0 Win=15987 Len=0
  5   0.000000 184.157.5.89 -> 200.206.aaa.bbb TCP 5242 > 54690 [ACK]
Seq=3519464534 Ack=0 Win=15987 Len=0
```

```
  6   0.000000 133.33.248.59 -> 200.181.ccc.ddd TCP 5498 > 45386 [ACK]
Seq=3536241750 Ack=0 Win=15987 Len=0
  7   0.000000 10.132.87.107 -> 200.206.aaa.bbb TCP 5754 > 13668 [ACK]
Seq=3553018966 Ack=0 Win=15987 Len=0
  8   0.000000 38.34.196.27 -> 200.181.ccc.ddd TCP 6010 > 10986 [ACK]
Seq=3569796182 Ack=0 Win=15987 Len=0
  9   0.000000 157.244.87.56 -> 200.206.aaa.bbb TCP 6266 > 46309 [ACK]
Seq=3586573398 Ack=0 Win=15987 Len=0
 10   0.000000 239.98.57.67 -> 200.181.ccc.ddd TCP 6522 > 4362 [ACK]
Seq=3603350614 Ack=0 Win=15987 Len=0
 11   0.000000 12.23.246.51 -> 200.206.aaa.bbb TCP 6778 > 35128 [ACK]
Seq=3620127830 Ack=0 Win=15987 Len=0
 12   0.000000 255.204.187.86 -> 200.181.ccc.ddd TCP 7034 > 3173 [ACK]
Seq=3636905046 Ack=0 Win=15987 Len=0
 13   0.000000 79.207.253.50 -> 200.206.aaa.bbb TCP 7290 > 27578 [ACK]
Seq=3653682262 Ack=0 Win=15987 Len=0
 14   0.000000 158.218.169.3 -> 200.181.ccc.ddd TCP 7546 > 46194 [ACK]
Seq=3670459478 Ack=0 Win=15987 Len=0
 15   0.000000 56.129.105.88 -> 200.206.aaa.bbb TCP 7802 > 44372 [ACK]
Seq=3687236694 Ack=0 Win=15987 Len=0
 16   0.000000 36.139.138.41 -> 200.181.ccc.ddd TCP 8058 > 55657 [ACK]
Seq=3704013910 Ack=0 Win=15987 Len=0
 17   0.000000 74.54.202.117 -> 200.206.aaa.bbb TCP 8314 > 21853 [ACK]
Seq=3720791126 Ack=0 Win=15987 Len=0
 18   0.000000 226.244.24.86 -> 200.181.ccc.ddd TCP 8570 > 32141 [ACK]
Seq=3737568342 Ack=0 Win=15987 Len=0
 19   0.000000 67.131.102.61 -> 200.206.aaa.bbb TCP 8826 > 29674 [ACK]
Seq=3754345558 Ack=0 Win=15987 Len=0
 20   0.000000 95.227.104.39 -> 200.181.ccc.ddd TCP 9082 > 64288 [ACK]
Seq=3771122774 Ack=0 Win=15987 Len=0
 21   0.000000 72.141.62.75 -> 200.206.aaa.bbb TCP 9338 > 58372 [ACK]
Seq=3787899990 Ack=0 Win=15987 Len=0
 22   0.000000 172.107.199.12 -> 200.181.ccc.ddd TCP 9594 > 21009 [ACK]
Seq=3804677206 Ack=0 Win=15987 Len=0
 23   0.000000 25.105.192.123 -> 200.206.aaa.bbb TCP 9850 > 53901 [ACK]
Seq=3821454422 Ack=0 Win=15987 Len=0
 24   0.000000 124.251.111.39 -> 200.181.ccc.ddd TCP 10106 > 46685 [ACK]
Seq=3838231638 Ack=0 Win=15987 Len=0
 25   0.000000 134.115.194.37 -> 200.206.aaa.bbb TCP 10362 > 27742 [ACK]
Seq=3855008854 Ack=0 Win=15987 Len=0
 26   0.000000 200.103.157.95 -> 200.181.ccc.ddd TCP 10618 > 37770 [ACK]
Seq=3871786070 Ack=0 Win=15987 Len=0
 27   0.000000 245.150.112.72 -> 200.206.aaa.bbb TCP 10874 > 50996 [ACK]
Seq=3888563286 Ack=0 Win=15987 Len=0
 28   0.000000 159.239.144.125 -> 200.181.ccc.ddd TCP 11130 > 17766 [ACK]
Seq=3905340502 Ack=0 Win=15987 Len=0
 29   0.000000 50.239.253.55 -> 200.206.aaa.bbb TCP 11386 > 15818 [ACK]
Seq=3922117718 Ack=0 Win=15987 Len=0
 30   0.000000 249.216.33.46 -> 200.181.ccc.ddd TCP 11642 > 27248 [ACK]
Seq=3938894934 Ack=0 Win=15987 Len=0
 31   0.000000  234.30.71.2 -> 200.206.aaa.bbb TCP 11898 > 7780 [ACK]
Seq=3955672150 Ack=0 Win=15987 Len=0
 32   0.000000 68.218.201.29 -> 200.181.ccc.ddd TCP 12154 > 13397 [ACK]
Seq=3972449366 Ack=0 Win=15987 Len=0
 33   0.000000 115.193.122.108 -> 200.206.aaa.bbb TCP 12410 > 9935 [ACK]
Seq=3989226582 Ack=0 Win=15987 Len=0
 34   0.000000 177.226.85.18 -> 200.181.ccc.ddd TCP 12666 > 46660 [ACK]
Seq=4006003798 Ack=0 Win=15987 Len=0
 35   0.000000 153.185.146.85 -> 200.206.aaa.bbb TCP 12922 > 4294 [ACK]
Seq=4022781014 Ack=0 Win=15987 Len=0
 36   0.000000 177.101.77.64 -> 200.181.ccc.ddd TCP 13178 > 57670 [ACK]
Seq=4039558230 Ack=0 Win=15987 Len=0
 37   0.000000 244.202.31.29 -> 200.206.aaa.bbb TCP 13434 > 24017 [ACK]
```

```
Seq=4056335446 Ack=0 Win=15987 Len=0
 38   0.000000 52.88.103.87 -> 200.181.ccc.ddd TCP 13690 > 3380 [ACK]
Seq=4073112662 Ack=0 Win=15987 Len=0
 39   0.000000 48.95.160.117 -> 200.206.aaa.bbb TCP 13946 > 45139 [ACK]
Seq=4089889878 Ack=0 Win=15987 Len=0
 40   0.000000 196.145.248.76 -> 200.181.ccc.ddd TCP 14202 > 46802 [ACK]
Seq=4106667094 Ack=0 Win=15987 Len=0
```

One thing worth mentioning is that both 200.206.aaa.bbb and 200.181.ccc.ddd were not machines in the ISP's address range.

I'll explain what we can see in those packets. Use the following schematic drawing of a TCP packet as reference.



A TCP packet

Basically, the explanation of the information that Ethereal decoded for a packet is as follows:

```
  1   0.000000 161.232.248.52 -> 200.206.aaa.bbb TCP 4218 > 40417 [ACK]
Seq=3452355670 Ack=0 Win=15987 Len=0
```

1 is the packet number within this Ethereal capture. In this case we are looking at the first packet.

0.000000 is the timestamp in seconds at which this packet was captured, in relation to the total time of the capture. We can see that this packet was captured at time zero, because is the first packet. In the particular incident discussed you can also see that all 40 packets came within the same period (that means that they came sequentially, but in a VERY short period of time).

161.232.248.52 is the source IP address of the packet. At this point you don't know if this is the IP address of the origin system, a router or gateway, etc.

The symbol **->** helps to clarify the direction of the traffic. In this case was sent from 161.232.248.52 to 200.206.aaa.bbb

200.206.aaa.bbb is the destination IP address of the packet.

TCP is the protocol used by this packet.

4218 is the source port in the origin host.

40417 is the destination port in the destination host.

[ACK] is one of the flags of the TCP protocol in this packet. The complete list of flags (in the order they're located in a TCP packet) together with a short explanation is as follows:

URG: means that the content of the Urgent Pointer field is significant. From the Internet Standard 0007 "Transmission Control Protocol"[22]:

> *"[The Urgent Pointer] field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment. The urgent pointer points to the sequence number of the octet following the urgent data. This field is only be interpreted in segments with the URG control bit set".*

ACK: means that the content of the Acknowledge Number field is significant. From the Internet Standard 0007 "Transmission Control Protocol"[22]:

> *"If the ACK control bit is set [the Acknowledge Number] field contains the value of the next sequence number the sender of the segment is expecting to receive. Once a connection is established this is always sent".*

PSH: Push Function. From the Internet Standard 0007 "Transmission Control Protocol"[22]:

> *"If the PUSH flag is set, the data must be transmitted promptly to the receiver, and the PUSH bit will be set in the last TCP segment created from the buffer.*
>
> *…*
>
> *When a receiving TCP sees the PUSH flag, it must not wait for more data from the sending TCP before passing the data to the receiving process".*

RST: Reset the connection. This flag is used to forcefully terminate a connection, instead of following the proper "negotiation" to end it.

SYN: Synchronize sequence numbers. This flag is usually seen only during the initiation of a connection, during the so named "3 way handshake" depicted here:

```
A --> B  SYN
A <-- B  SYN / ACK
A --> B  ACK
```

(In the example above I suppose that the connection CAN be initiated, i.e. the destination port in system B is open).
The SYN flag name came from the fact that during the 3 way handshake both systems inform each other of their respective sequence numbers (more on that below). System A tells B his sequence number, then B acknowledges that (this is why the ACK flag is set) and informs A about his own sequence number. Finally A acknowledges that.
For the sake of completeness, here's what happens when the destination port in B is closed:

```
A --> B  SYN
A <-- B  RST / ACK
```

Can you see the use of the RST flag?

FIN: No more data from sender. This flag is used to negotiate the termination of an established TCP connection instead of forcefully resetting it with the RST flag.

`Seq=3452355670` is the sequence number carried by this packet. In our example is the sequence number of the origin system.

`Ack=0` is the acknowledge number carried by this packet. In our example is zero because there's nothing to acknowledge at this point because the packet is not part of an ongoing connection. During normal traffic, this field will be the destination sequence number + 1 (in fact, the next sequence number the sender of the packet is expecting to receive).

`Win=15987` is the maximum number of bytes (more properly octets) that the sender of this packet will accept, counting from the byte indicated by the acknowledge number field. Can be considered kind of a "buffer size" per packet.

`Len=0` indicates the length of the data in the packet. In all the packets in the capture above is always zero.

For a very detailed description of TCP functionality, flags, fields, usage, etc., please read the Internet Standard 0007 "Transmission Control Protocol".[22]

☞ ***Hint:*** *when visiting http://www.rfc-editor.org, use the opportunity to learn about the internals of AT LEAST the 3 basic protocols we use: ICMP, UDP and TCP.*

What doesn't need to be explained after a quick view at the capture is that the source addresses of those packets are spoofed. You can see any imaginable address there,

but all packets are VERY similar: TCP packets with the ACK flag set, a Window size of 15987, and no information inside, seemingly from random source ports to random destination ports.

In some future captures the destination addresses were different, otherwise the general pattern was the same.

Let's analyze facts:

- The packets look so similar that we can suppose that they come from the same process on a single machine or device, instead of different sources. This also makes us more confident in our guess that the source addresses are spoofed.
- The speed at which the packets are sent is also not good. Even without taking into account all the other evidence, it's too fast for any decent communication.
- The packets didn't have the SYN flag set, so they are not intended to initiate a connection with the other side.
- They carry no data, so there's no clear purpose on them (at least not a good purpose).
- Finally, all of them have the ACK flag set. It has not been discussed so far, but some unsophisticated firewalls simply allow all traffic with the ACK flag set to go through, because this is what you've to expect for most of the traffic during a normal communication.

A lot of incriminating evidence…

By now, an expert would be quite sure of the culprit, but for those who are still learning the business, this had started smelling of a DoS attack, specifically like a couple of them that have a very similar pattern: `stream` and `raped`.

The authors of the excellent series of books Hacking Exposed wrote that:

> Both attacks [stream and raped] are resource-starvation attacks taking advantage of the operating system's inability to manage all the malformed packets sent to it at once...
> ... The stream.c attack works by sending TCP ACK packets to a series of ports with random sequence numbers and random source IP addresses. The raped.c attack works by sending TCP ACK packets with spoofed source IP addresses.
> (McClure, Stuart; Scambray, Joel; Kurtz, George; p. 517–519, 4th Edition)[23]

I quote here the 4th Edition of the book, but this paragraph has been there since, at least, the 2nd Edition. The page numbers mentioned in the references correspond to the 4th Edition.

I'm not going to repeat here what is a DoS, a DDoS, or how those particular attacks operate, because those topics have been discussed and explained in detail a lot of times. For those interested in some advanced reading, please review the following papers:

Dittrich, David; Weaver, George; Dietrich, Sven; Long, Neil.
"The "mstream" distributed denial of service attack tool". 1 May 2000.
URL: http://staff.washington.edu/dittrich/misc/mstream.analysis.txt[24]

Murphy, Michael. "mstream – DDoS – Plain and Simple?".
GCIH (GIAC Certified Incident Handling Analyst (GCIH)). 12 Mar 2003.
URL: http://www.giac.org/practical/Michael_Murphy_GCIH.doc[25]

At the time of this incident, only the first of those two papers was available, but I mention the GCIH practical because it's a very good paper on this subject.

The first paper is the "de facto" bible on this DDoS tool, and at this time I've already read it, so provisionally identifying the problem was easy.

Taking that into account we can be inclined to identify the problem in our network as an outgoing stream or mstream attack, but due to the fact that the source code of the programs used to create the packets is readily available and has been modified several times, we can't be sure until we find the program itself. To do that, we need to identify the origin system first, and we can't do that using the IP address, because all source addresses in the packets during the attack have being spoofed.

There are several options, like installing a NIDS like Snort with the signatures to detect the initiation of such attacks, or scanning all the systems looking for strange ports open. However, being short of time and money, and also aware of the size, complexity, and the heterogeneous nature of the network, I decided to repeat the traffic capture during an attack, but including all of the contents of the packets as well. This can lead to tremendous logs, but I felt confident that, if done only during the attack, we can keep the log size at a minimum. The hardware setup was not modified, but this time I typed the command:

```
# tethereal -w > logfile.raw
```

without pressing ENTER! I instructed the operators (mainly the night operator) to be ready to run to this machine and press ENTER if the WhatsUp Gold starts showing anything strange, and to press CTRL-C a couple minutes after the start of the problem.

2.3. The Fox Hideout

We made a nice capture of the next attack, and I used the GUI interface of Ethereal to do the analysis. I could have used '`tethereal -x < logfile.raw`' as in the printouts here for the same purpose from the shell, but I really like Ethereal's GUI!

```
  1   0.000000 161.232.248.52 -> 200.206.aaa.bbb TCP 4218 > 40417 [ACK]
Seq=3452355670 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 6ded 0000 d606 6b68 a1e8 f834 c8ce   .(m.....kh...4..
  20  XXXX 107a 9de1 cdc6 c456 0000 0000 5010   ...z.....V....P.
  30  3e73 2576 0000 0000 0000 0000             >s%v........
```

```
  2   0.000000 49.210.67.107 -> 200.181.ccc.ddd TCP 4474 > 2187 [ACK]
Seq=3469132886 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 6eed 0000 d606 6c05 31d2 436b c8b5   .(n.....l.1.Ck..
  20  XXXX 117a 088b cec6 c456 0000 0000 5010   ...z.....V....P.
  30  3e73 ba69 0000 0000 0000 0000             >s.i........

  3   0.000000 51.87.122.36 -> 200.206.aaa.bbb TCP 4730 > 60709 [ACK]
Seq=3485910102 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 6fed 0000 d606 560a 3357 7a24 c8ce   .(o.....V.3Wz$..
  20  XXXX 127a ed25 cfc6 c456 0000 0000 5010   ...z.%...V....P.
  30  3e73 bed3 0000 0000 0000 0000             >s..........

  4   0.000000 145.245.150.11 -> 200.181.ccc.ddd TCP 4986 > 57941 [ACK]
Seq=3502687318 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 70ed 0000 d606 b741 91f5 960b c8b5   .(p......A......
  20  XXXX 137a e255 d0c6 c456 0000 0000 5010   ...z.U...V....P.
  30  3e73 29db 0000 0000 0000 0000             >s).........

  5   0.000000 184.157.5.89 -> 200.206.aaa.bbb TCP 5242 > 54690 [ACK]
Seq=3519464534 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 71ed 0000 d606 438f b89d 0559 c8ce   .(q.....C....Y..
  20  XXXX 147a d5a2 d1c6 c456 0000 0000 5010   ...z.....V....P.
  30  3e73 c1db 0000 0000 0000 0000             >s..........

  6   0.000000 133.33.248.59 -> 200.181.ccc.ddd TCP 5498 > 45386 [ACK]
Seq=3536241750 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 72ed 0000 d606 5fe5 8521 f83b c8b5   .(r....._..!.;..
  20  XXXX 157a b14a d2c6 c456 0000 0000 5010   ...z.J...V....P.
  30  3e73 018a 0000 0000 0000 0000             >s..........

  7   0.000000 10.132.87.107 -> 200.206.aaa.bbb TCP 5754 > 13668 [ACK]
Seq=3553018966 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 73ed 0000 d606 9d96 0a84 576b c8ce   .(s.........Wk..
  20  XXXX 167a 3564 d3c6 c456 0000 0000 5010   ...z5d...V....P.
  30  3e73 ba21 0000 0000 0000 0000             >s.!........

  8   0.000000 38.34.196.27 -> 200.181.ccc.ddd TCP 6010 > 10986 [ACK]
Seq=3569796182 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 74ed 0000 d606 f104 2622 c41b c8b5   .(t.......&"....
  20  XXXX 177a 2aea d4c6 c456 0000 0000 5010   ...z*....V....P.
  30  3e73 170a 0000 0000 0000 0000             >s..........

  9   0.000000 157.244.87.56 -> 200.206.aaa.bbb TCP 6266 > 46309 [ACK]
Seq=3586573398 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 75ed 0000 d606 0859 9df4 5738 c8ce   .(u......Y..W8..
  20  XXXX 187a b4e5 d5c6 c456 0000 0000 5010   ...z.....V....P.
  30  3e73 a362 0000 0000 0000 0000             >s.b........
```

```
 10   0.000000 239.98.57.67 -> 200.181.ccc.ddd TCP 6522 > 4362 [ACK]
Seq=3603350614 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 76ed 0000 d606 b09c ef62 3943 c8b5   .(v........b9C..
  20  XXXX 197a 110a d6c6 c456 0000 0000 5010   ...z.....V....P.
  30  3e73 ee81 0000 0000 0000 0000             >s..........

 11   0.000000 12.23.246.51 -> 200.206.aaa.bbb TCP 6778 > 35128 [ACK]
Seq=3620127830 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 77ed 0000 d606 f93a 0c17 f633 c8ce   .(w.......:...3..
  20  XXXX 1a7a 8938 d7c6 c456 0000 0000 5010   ...z.8...V....P.
  30  3e73 bdf1 0000 0000 0000 0000             >s..........

 12   0.000000 255.204.187.86 -> 200.181.ccc.ddd TCP 7034 > 3173 [ACK]
Seq=3636905046 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 78ed 0000 d606 1c1f ffcc bb56 c8b5   .(x..........V..
  20  XXXX 1b7a 0c65 d8c6 c456 0000 0000 5010   ...z.e...V....P.
  30  3e73 5ca9 0000 0000 0000 0000             >s\.........

 13   0.000000 79.207.253.50 -> 200.206.aaa.bbb TCP 7290 > 27578 [ACK]
Seq=3653682262 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 79ed 0000 d606 ac83 4fcf fd32 c8ce   .(y.......O..2..
  20  XXXX 1c7a 6bba d9c6 c456 0000 0000 5010   ...zk....V....P.
  30  3e73 8cb8 0000 0000 0000 0000             >s..........

 14   0.000000 158.218.169.3 -> 200.181.ccc.ddd TCP 7546 > 46194 [ACK]
Seq=3670459478 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 7aed 0000 d606 8d64 9eda a903 c8b5   .(z......d......
  20  XXXX 1d7a b472 dac6 c456 0000 0000 5010   ...z.r...V....P.
  30  3e73 23e1 0000 0000 0000 0000             >s#.........

 15   0.000000 56.129.105.88 -> 200.206.aaa.bbb TCP 7802 > 44372 [ACK]
Seq=3687236694 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 7bed 0000 d606 55ac 3881 6958 c8ce   .({.....U.8.iX..
  20  XXXX 1e7a ad54 dbc6 c456 0000 0000 5010   ...z.T...V....P.
  30  3e73 f246 0000 0000 0000 0000             >s.F........

 16   0.000000 36.139.138.41 -> 200.181.ccc.ddd TCP 8058 > 55657 [ACK]
Seq=3704013910 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 7ced 0000 d606 248e 248b 8a29 c8b5   .(|.....$.$..)..
  20  XXXX 1f7a d969 dcc6 c456 0000 0000 5010   ...z.i...V....P.
  30  3e73 9413 0000 0000 0000 0000             >s..........

 17   0.000000 74.54.202.117 -> 200.206.aaa.bbb TCP 8314 > 21853 [ACK]
Seq=3720791126 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 7ded 0000 d606 e0d9 4a36 ca75 c8ce   .(}.......J6.u..
  20  XXXX 207a 555d ddc6 c456 0000 0000 5010   .. zU]...V....P.
```

```
  30  3e73 d36b 0000 0000 0000 0000            >s.k........

 18   0.000000 226.244.24.86 -> 200.181.ccc.ddd TCP 8570 > 32141 [ACK]
Seq=3737568342 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 7eed 0000 d606 d5f7 e2f4 1856 c8b5   .(~..........V..
  20  XXXX 217a 7d8d dec6 c456 0000 0000 5010   ..!z}....V....P.
  30  3e73 9f59 0000 0000 0000 0000            >s.Y........

 19   0.000000 67.131.102.61 -> 200.206.aaa.bbb TCP 8826 > 29674 [ACK]
Seq=3754345558 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 7fed 0000 d606 49c5 4383 663d c8ce   .(......I.C.f=..
  20  XXXX 227a 73ea dfc6 c456 0000 0000 5010   .."zs....V....P.
  30  3e73 1bca 0000 0000 0000 0000            >s..........

 20   0.000000 95.227.104.39 -> 200.181.ccc.ddd TCP 9082 > 64288 [ACK]
Seq=3771122774 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 80ed 0000 d606 0738 5fe3 6827 c8b5   .(.......8_.h'..
  20  XXXX 237a fb20 e0c6 c456 0000 0000 5010   ..#z. ...V....P.
  30  3e73 5106 0000 0000 0000 0000            >sQ.........

 21   0.000000 72.141.62.75 -> 200.206.aaa.bbb TCP 9338 > 58372 [ACK]
Seq=3787899990 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 81ed 0000 d606 6aad 488d 3e4b c8ce   .(......j.H.>K..
  20  XXXX 247a e404 e1c6 c456 0000 0000 5010   ..$z.....V....P.
  30  3e73 ca97 0000 0000 0000 0000            >s..........

 22   0.000000 172.107.199.12 -> 200.181.ccc.ddd TCP 9594 > 21009 [ACK]
Seq=3804677206 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 82ed 0000 d606 59ca ac6b c70c c8b5   .(......Y..k....
  20  XXXX 257a 5211 e2c6 c456 0000 0000 5010   ..%zR....V....P.
  30  3e73 4aa8 0000 0000 0000 0000            >sJ.........

 23   0.000000 25.105.192.123 -> 200.206.aaa.bbb TCP 9850 > 53901 [ACK]
Seq=3821454422 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 83ed 0000 d606 15a1 1969 c07b c8ce   .(.........i.{..
  20  XXXX 267a d28d e3c6 c456 0000 0000 5010   ..&z.....V....P.
  30  3e73 8502 0000 0000 0000 0000            >s..........

 24   0.000000 124.251.111.39 -> 200.181.ccc.ddd TCP 10106 > 46685 [ACK]
Seq=3838231638 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 84ed 0000 d606 df1f 7cfb 6f27 c8b5   .(........|.o'..
  20  XXXX 277a b65d e4c6 c456 0000 0000 5010   ..'z.]...V....P.
  30  3e73 69b1 0000 0000 0000 0000            >si.........

 25   0.000000 134.115.194.37 -> 200.206.aaa.bbb TCP 10362 > 27742 [ACK]
Seq=3855008854 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 85ed 0000 d606 a4ec 8673 c225 c8ce   .(.........s.%..
```

```
  20  XXXX 287a 6c5e e5c6 c456 0000 0000 5010   ..(zl^...V....P.
  30  3e73 787d 0000 0000 0000 0000             >sx}........

 26   0.000000 200.103.157.95 -> 200.181.ccc.ddd TCP 10618 > 37770 [ACK]
Seq=3871786070 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 86ed 0000 d606 637b c867 9d5f c8b5   .(......c{.g._..
  20  XXXX 297a 938a e6c6 c456 0000 0000 5010   ..)z.....V....P.
  30  3e73 0ee0 0000 0000 0000 0000             >s..........

 27   0.000000 245.150.112.72 -> 200.206.aaa.bbb TCP 10874 > 50996 [ACK]
Seq=3888563286 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 87ed 0000 d606 85a6 f596 7048 c8ce   .(..........pH..
  20  XXXX 2a7a c734 e7c6 c456 0000 0000 5010   ..*z.4...V....P.
  30  3e73 fc60 0000 0000 0000 0000             >s.`........

 28   0.000000 159.239.144.125 -> 200.181.ccc.ddd TCP 11130 > 17766 [ACK]
Seq=3905340502 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 88ed 0000 d606 96d5 9fef 907d c8b5   .(...........}..
  20  XXXX 2b7a 4566 e8c6 c456 0000 0000 5010   ..+zEf...V....P.
  30  3e73 8e5e 0000 0000 0000 0000             >s.^........

 29   0.000000 50.239.253.55 -> 200.206.aaa.bbb TCP 11386 > 15818 [ACK]
Seq=3922117718 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 89ed 0000 d606 b95e 32ef fd37 c8ce   .(.......^2..7..
  20  XXXX 2c7a 3dca e9c6 c456 0000 0000 5010   ..,z=....V....P.
  30  3e73 b783 0000 0000 0000 0000             >s..........

 30   0.000000 249.216.33.46 -> 200.181.ccc.ddd TCP 11642 > 27248 [ACK]
Seq=3938894934 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 8aed 0000 d606 aa3b f9d8 212e c8b5   .(.......;..!...
  20  XXXX 2d7a 6a70 eac6 c456 0000 0000 5010   ..-zjp...V....P.
  30  3e73 7aba 0000 0000 0000 0000             >sz.........

 31   0.000000  234.30.71.2 -> 200.206.aaa.bbb TCP 11898 > 7780 [ACK]
Seq=3955672150 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 8bed 0000 d606 b664 ea1e 4702 c8ce   .(.......d..G...
  20  XXXX 2e7a 1e64 ebc6 c456 0000 0000 5010   ...z.d...V....P.
  30  3e73 d1ef 0000 0000 0000 0000             >s..........

 32   0.000000 68.218.201.29 -> 200.181.ccc.ddd TCP 12154 > 13397 [ACK]
Seq=3972449366 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 8ced 0000 d606 b54a 44da c91d c8b5   .(.......JD.....
  20  XXXX 2f7a 3455 ecc6 c456 0000 0000 5010   ../z4U...V....P.
  30  3e73 b9e4 0000 0000 0000 0000             >s..........

 33   0.000000 115.193.122.108 -> 200.206.aaa.bbb TCP 12410 > 9935 [ACK]
Seq=3989226582 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
```

```
  10  0028 8ded 0000 d606 f757 73c1 7a6c c8ce   .(.......Ws.zl..
  20  XXXX 307a 26cf edc6 c456 0000 0000 5010   ..0z&....V....P.
  30  3e73 0878 0000 0000 0000 0000             >s.x........

 34   0.000000 177.226.85.18 -> 200.181.ccc.ddd TCP 12666 > 46660 [ACK]
Seq=4006003798 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 8eed 0000 d606 ba4d b1e2 5512 c8b5   .(.......M..U...
  20  XXXX 317a b644 eec6 c456 0000 0000 5010   ..1z.D...V....P.
  30  3e73 3af8 0000 0000 0000 0000             >s:.........

 35   0.000000 153.185.146.85 -> 200.206.aaa.bbb TCP 12922 > 4294 [ACK]
Seq=4022781014 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 8fed 0000 d606 b776 99b9 9255 c8ce   .(.......v...U..
  20  XXXX 327a 10c6 efc6 c456 0000 0000 5010   ..2z.....V....P.
  30  3e73 dc9f 0000 0000 0000 0000             >s.........

 36   0.000000 177.101.77.64 -> 200.181.ccc.ddd TCP 13178 > 57670 [ACK]
Seq=4039558230 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 90ed 0000 d606 c09c b165 4d40 c8b5   .(.........eM@..
  20  XXXX 337a e146 f0c6 c456 0000 0000 5010   ..3z.F...V....P.
  30  3e73 1445 0000 0000 0000 0000             >s.E........

 37   0.000000 244.202.31.29 -> 200.206.aaa.bbb TCP 13434 > 24017 [ACK]
Seq=4056335446 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 91ed 0000 d606 cd9d f4ca 1f1d c8ce   .(..............
  20  XXXX 347a 5dd1 f1c6 c456 0000 0000 5010   ..4z]....V....P.
  30  3e73 a3bb 0000 0000 0000 0000             >s.........

 38   0.000000 52.88.103.87 -> 200.181.ccc.ddd TCP 13690 > 3380 [ACK]
Seq=4073112662 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 92ed 0000 d606 2193 3458 6757 c8b5   .(......!.4XgW..
  20  XXXX 357a 0d34 f2c6 c456 0000 0000 5010   ..5z.4...V....P.
  30  3e73 474e 0000 0000 0000 0000             >sGN........

 39   0.000000 48.95.160.117 -> 200.206.aaa.bbb TCP 13946 > 45139 [ACK]
Seq=4089889878 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 93ed 0000 d606 0eb1 305f a075 c8ce   .(........0_.u..
  20  XXXX 367a b053 f3c6 c456 0000 0000 5010   ..6z.S...V....P.
  30  3e73 904c 0000 0000 0000 0000             >s.L........

 40   0.000000 196.145.248.76 -> 200.181.ccc.ddd TCP 14202 > 46802 [ACK]
Seq=4106667094 Ack=0 Win=15987 Len=0

   0  0002 161f d4e0 0020 780d 1c17 0800 4508   ....... x.....E.
  10  0028 94ed 0000 d606 fe63 c491 f84c c8b5   .(.......c...L..
  20  XXXX 377a b6d2 f4c6 c456 0000 0000 5010   ..7z.....V....P.
  30  3e73 7880 0000 0000 0000 0000             >sx.........
```

In this capture we are looking at a bit more than the line decoded by Ethereal. To start with, we can see the data content of the packets, that bring us very interesting

information, but what's more important: the MAC address of the origin system can be clearly be seen in positions 06 – 09 in all packets: 00:20:78:0d (numbering the positions starts with 00). This is because Ethereal captures the entire Ethernet II frame from the network, and when decoding the TCP packet shows only the information that is usually more relevant for that traffic, but ALL the information in the frame is there, including the MAC addresses of the source and destination device within that network segment.

The ISP's CEO already had the program MAC Address Discovery from the SolarWinds suit[26] in one of the monitoring machines, and so discovering the IP address of the system was quite straightforward, taking only some time.
The IP was a public one, and for a minute I was afraid that this IP was that of the other router, that could have complicated things a bit from the point of view of tracing the source of the attack, but we were lucky and the system was identified as a server owned by one the clients of the ISP. (Coincidences are wonderful sometimes: the client was my friend, the one that got in touch with me in the very beginning).

So we found our "Unintentional Criminal" in my friend's server. I didn't think for a minute that he was directly responsible, but somehow someone has gained access to the server and was using it as a weapon.

2.4. Where's the Fox?

We had found the fox hideout, but we still hadn't caught the fox. To do that, I checked once more the mountain of logs we had collected so far, looking for traffic directed to the problematic server just before the start of one of the attacks. This is what I found:

```
  1   0.000000 129.aa.bb.cc -> 200.xx.yy.zz UDP Source port: 1904
Destination port: 8000
  2   0.000000 200.xx.yy.zz -> 129.aa.bb.cc UDP Source port: 2216
Destination port: 8001
  3   0.000000 200.xx.yy.zz -> 64.aa.bb.cc UDP Source port: 2216
Destination port: 8001
```

And looking at the payload of the packets:

```
  1   0.000000 129.aa.bb.cc -> 200.xx.yy.zz UDP Source port: 1904
Destination port: 8000

   0  0020 780d 1c17 0002 161f d4e0 0800 4500   . x...........E.
  10  0049 04e5 0000 3311 6e83 81aa bbcc c8xx   .I....3.n......x
  20  yyzz 0770 1f40 0035 78a5 6d73 7472 6561   yz.p.@.5x.mstrea
  30  6d2f 3230 302e 3230 362e AAAA AA2e BBBB   m/200.206.AAA.BB
  40  BB3a 3230 302e 3138 312e AAAA AA2e BBBB   B:200.181.AAA.BB
  50  BB2f 3138 3030 0a                         B/1800.

  2   0.000000 200.xx.yy.zz -> 129.aa.bb.cc UDP Source port: 2216
Destination port: 8001

   0  0002 161f d4e0 0020 780d 1c17 0800 4500   ....... x.....E.
  10  0043 b778 0000 4011 aef5 c8xx yyzz 81aa   .C.x..@....xyz..
  20  bbcc 08a8 1f41 002f dd72 6e6f 7463 5374   .....A./.rnotcSt
  30  6172 7469 6e67 206d 7374 7265 616d 696e   arting mstreamin
  40  6720 7769 7468 2034 3020 6b62 2f73 2e2e   g with 40 kb/s..
  50  2e                                        .
```

```
  3   0.000000 200.xx.yy.zz -> 64.aa.bb.cc UDP Source port: 2216
Destination port: 8001

   0  0002 161f d4e0 0020 780d 1c17 0800 4500   ....... x.....E.
  10  0043 b779 0000 4011 12cd c8xx yyzz 40aa   .C.y..@....xyz@.
  20  bbcc 08a8 1f41 002f 414b 6e6f 7463 5374   .....A./AKnotcSt
  30  6172 7469 6e67 206d 7374 7265 616d 696e   arting mstreamin
  40  6720 7769 7468 2034 3020 6b62 2f73 2e2e   g with 40 kb/s..
  50  2e                                        .
```

From that it's quite clear that the attack was `mstream`, and that the attack order came from a host in an important University in the USA. A gentle "doorknob rattling" (i.e.: port scan) in this host showed a very insecure and obscure server (without even a name), that possible was the former hop for the real attacker to reach us.
The message indicating that the attack was initiated was sent to two hosts: the server in the University in the USA, and a host in Japan. A small investigation of this later host didn't bring any conclusive results, it was a dynamic IP address in an ISP.

So... it seems that we got no fox in this hunt, but at least we can solve the local problem, and made the people in the University aware of the situation.
Of course the pleasure of discovery, and to deal with complex IT security problems is something that I like, and this also counts.

3. AND WHAT WAS THE OUTCOME

3.1. Kicking the Trashcan

We put a rule in the ACL of the router blocking all traffic coming from the offending host in the University in the USA. We got the email of the coordinator there using the whois engine at NetworkSolutions[27], and sent her a short message detailing what we've got here. Regrettably we never got an answer back (don't think that this is related to the ACL mentioned above!).

I checked the internal compromised system remotely, and it come to be an unpatched Linux distribution. After some talks, it seems that it was installed for some testing a while ago, and forgotten after that (connected to the network!). In fact I was able to gain administrative access to this host at least exploiting 2 well known vulnerabilities.

Literally pulling the plug of the server and connecting its hard disk to a safe machine to perform a forensic analysis on it showed the following:

- a '`...`' directory under `/tmp`. This, together with putting the tools in some obscure directory somewhere, are very old but still effective ways to trivially hide files and programs. Can be easily defeated by a local security analyzer or system integrity tool, but is effective against human beings.
- inside this directory, the `mstream` DDoS executable (no source code), an IRC eggdropper (a program that can have many functionalities, like controlling IRC channels, acting as a robot in the same, serving files, cracking IRC channel passwords, etc.), and a few other hacking tools were found
- the original compromise of this system was probably due to one of the unpatched RPC or WU-FTPD vulnerabilities mentioned below

Multiple Linux Vendor rpc.statd Remote Format String Vulnerability[28]
SecurityFocus. 16 July 2000. URL: http://www.securityfocus.com/bid/1480

Wu-Ftpd Remote Format String Stack Overwrite Vulnerability[29]
SecurityFocus. 22 June 2000. URL: http://www.securityfocus.com/bid/1387

It is important to "pull the plug" (literally) if you intend to do forensics, because there's always the possibility that the attacker has installed some self-destruction mechanism to delete the hacking tools or wipe the entire hard disk if the owner tries to shut it down properly.

Taking into account the type of tools found, and the relatively simple way in which those were hidden, I felt inclined to suppose that this was not the work of a technically skilled individual. Not even a "serious" one, someone that were looking for important information. Just someone (I'm opposed to the classical image of the teenager hacker!) that has enough time to lurk in the IRC and that's interested in trying DoS attacks on seemingly random hosts on the Internet.

A technically skilled attacker could have used one of the so called rootkits to hide his/her presence in the compromised system. At the time of this writing there're 2

clearly different types of rootkits, those that simply replace tools in the system by modified ones, in order to hide the attacker's files, processes, network sockets, etc., and those that are kernel based. Modifying kernel structures in memory is simply the gates to absolute power in the compromised system, not only you've the power of the administrator user, but no one of your resources in that machine can be detected from within the running system. Everything, even integrity checking tools, has to pass through the kernel first. If you own the kernel, game over.

A "serious" attacker looking for data could have got all the information he/she was looking for (don't ask me, I don't know what other people could want), erase all traces of any compromise on that system, and went away.

Regrettably there were some passwords (for other systems) in a clear text file under a user in the `/home`, so we instructed the owner of the server to verify the change of those.

3.2. Rebuilding for the Future

The internal compromised host will be reinstalled from scratch, this time with all security updates needed, and won't be forgotten in a corner anymore.

At least 1 firewall in each entry point in the perimeter was badly needed, and it was decided together with the ISP's CTO that an implementation of Linux acting as a bridge+firewall[30] was the best solution, because it can be inserted directly at any point without changing anything.

Some diagrams for re-engineering of the network were started, oriented towards the creating of appropriate DMZ areas for ISP systems that need to be externally accessible, and another for the clientele.

I showed Snort to the CTO (on his own Windows 2000 workstation), and he was also positively impressed. In fact we detected someone (internal!) scanning web servers looking for IIS with the Unicode vulnerability! The installation of minimal NIDS using Snort on a dedicated Linux machine was also contemplated.

At this time keeping the interest of the CTO was more and more difficult (that means that convincing him to invest in security now that that the problem was not "hot" anymore was almost impossible), so as a final word I recommended some templates for Security Policies that can be found in SANS website.[31]

3.3. Conclusions

Well... that was it. Simply a DDoS, but a very nasty one, for which there's no known patch (the authors of Hacking Exposed mentioned an unofficial patch for FreeBSD[32], but the link they provide isn't valid anymore, so I suppose that the patch has been incorporated in the main stream of FreeBSD). The impact of such DDoS attacks must not be minimized after the attack to at least 7 big companies during February 2000, as seen in these news headlines at CNN.com (they were particularly interested at that time, because their site was one of the targets):

"We can prevent those distributed denial of service attacks with 'egress filtering'"[33]
Livingston, Brian for CNN.com. 1 March 2000.
URL: http://edition.cnn.com/2000/TECH/computing/03/01/prevent.ddos.idg/index.html

"Avoiding future denial-of-service attacks"[34]
Denise Pappalardo for CNN.com. 23 February 2000.
URL: http://edition.cnn.com/2000/TECH/computing/02/23/isp.block.idg/index.html

It's also important to consider what could have happened if someone else would have found the problem... after being under attack. Our "Unintentional Criminal" could have had a big problem indeed.

Some people may think that incidents like this simply happen to anyone working in IT security. Perhaps this is true, but this is not the root of the problem.

As you can realize after reading this paper, this organization didn't have anyone with the experience to face the situation and solve it, and needed to hire me for that. This is particularly worrisome being a medium size ISP.

But don't blame this organization too much. In the Third World only the Big Companies (^{TM}) have the resources to invest in a good IT Security infrastructure (and sometimes they DO invest!), but Small Office Home Office (thereafter SOHO) companies have to rely on one or more "IT handymen" that know how to install and operate the servers and the infrastructure, but are not very well versed in IT Security.

For such small to medium size companies, money used in IT Security is money spent, not money invested (regrettably, also for some BIG companies!), and usually they are not in the best economic position when it comes to buy some expensive hardware, software or training (invariably with a price in US dollars, and an exchange rate against the company). This is why in this paper I tried to show how we used readily available hardware, Linux, and open source tools for almost anything, with the hope to inspire some people to do things better, knowing that this doesn't always mean at a higher cost. In the Open Source world even training shouldn't be big problem, as long as you're willing to train yourself!

References

[1] SecurityFocus. "Vulnerabilities".
URL: http://www.securityfocus.com/bid

[2] SecurityFocus. "Cisco IOS BGP Transitive Attribute Denial of Service Vulnerability". 10 May 2001.
URL: http://www.securityfocus.com/bid/2733

[3] Cisco Systems, Inc. "Cisco Security Advisory: Cisco IOS BGP Attribute Corruption Vulnerability". Rev. 1.1
10 May 2001.
URL: http://www.cisco.com/warp/public/707/ios-bgp-attr-corruption-pub.shtml

[4] Network Working Group. "RFC-1771: A Border Gateway Protocol 4 (BGP-4)". March 1995.
URL: ftp://ftp.rfc-editor.org/in-notes/rfc1771.txt

[5] SecurityFocus. "Cisco Local Interface ARP Denial of Service Vulnerability". 15 November 2001.
URL: http://www.securityfocus.com/bid/3547

[6] Cisco Systems, Inc. "Cisco Security Advisory: Cisco IOS ARP Table Overwrite Vulnerability". Rev. 1.3. Last updated 22 July 2002.
URL: http://www.cisco.com/warp/public/707/IOS-arp-overwrite-vuln-pub.shtml

[7] BlackHat®. "Black Hat Briefings, Training and Consulting Security".
URL: http://www.blackhat.com/main.html

[8] Security Bugware. "Packet Excalibur: A multi-platform graphical and scriptable network packet engine with extensible text based protocol descriptions".
URL: http://www.securitybugware.org/excalibur/

[9] Ballmann, Bastian; Krecher, Stefan. "P.A.T.H. -- Perl Advanced TCP Hijacking".
URL: http://p-a-t-h.sourceforge.net/html/index.php

[10] SecurityFocus. "Cisco IOS Malformed SNMP Message Denial of Service Vulnerabilities". 12 February 2002.
URL: http://www.securityfocus.com/bid/4132

[11] SecurityFocus. "Cisco IOS Malicious IPV4 Packet Sequence Denial Of Service Vulnerability". 16 July 2003.
URL: http://www.securityfocus.com/bid/8211

[12] Carnegie Mellon, Software Engineering Institute, CERT® Coordination Center. "CERT® Advisory CA-2002-03 Multiple Vulnerabilities in Many Implementations of the Simple Network Management Protocol (SNMP)". Last revised 18 August 2003.
URL: http://www.cert.org/advisories/CA-2002-03.html

[13] Cisco Systems, Inc. " Cisco Security Advisory: Malformed SNMP Message-Handling Vulnerabilities". Rev. 2.1. Last updated 14 March 2002.

URL: http://www.cisco.com/warp/public/707/cisco-malformed-snmp-msgs-pub.shtml

[14] Sanfilippo, Salvatore and team of contributors. "hping: a command-line oriented TCP/IP packet assembler/analyzer".
URL: http://www.hping.org

[15] Cisco Systems, Inc. " Cisco Security Advisory: Cisco IOS Interface Blocked by IPv4 Packets". Rev. 1.14. Last updated 4 September 2003.
URL: http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml

[16] Song, Dug. "dsniff: a collection of tools for network auditing and penetration testing".
URL: http://www.monkey.org/~dugsong/dsniff/

[17] Phenoelit group. "ARP0c connection interceptor".
URL: http://www.phenoelit.de/fr/tools.html (click on ARP0c/WCI on the left side)

[18] Russell, Rusty. "Linux IPCHAINS-HOWTO". Version 1.0.8. 4 July 2000.
URL: http://www.linuxdocs.org/HOWTOs/IPCHAINS-HOWTO-7.html

[19] IPSWITCH. "WhatsUp Gold".
URL: http://www.ipswitch.com/Products/WhatsUp/index.html

[20] Combs, Gerald and many contributors. "The Ethereal Network Analyzer".
URL: http://www.ethereal.com

[21] Cisco Systems, Inc. " Configuring the Catalyst Switched Port Analyzer (SPAN)".
URL: http://www.cisco.com/warp/public/473/41.html

[22] University of Southern California, Information Science Institute. "RFC-793: Transmission Control Protocol". September 1981.
URL: ftp://ftp.rfc-editor.org/in-notes/std/std7.txt

[23] McClure, Stuart; Scambray, Joel; Kurtz, George. Hacking Exposed: Network Security Secrets & Solutions (4th Edition). Berkeley: McGraw-Hill/Osborne, 2003. Pages 517 – 519.

[24] Dittrich, David; Weaver, George; Dietrich, Sven; Long, Neil.
"The "mstream" distributed denial of service attack tool". 1 May 2000.
URL: http://staff.washington.edu/dittrich/misc/mstream.analysis.txt

[25] Murphy, Michael. "mstream – DDoS – Plain and Simple?".
GCIH (GIAC Certified Incident Handling Analyst (GCIH)) – Certification Practicals. 12 Mar 2003.
URL: http://www.giac.org/practical/Michael_Murphy_GCIH.doc

[26] SolarWinds.net Network Management. "SolarWinds Engineer's Edition Toolset".
URL: http://www.solarwinds.net/Tools/Engineer/index.htm

[27] NetworkSolutions. "WHOIS Search".
URL: http://www.networksolutions.com/en_US/whois/index.jhtml

[28] SecurityFocus. " Multiple Linux Vendor rpc.statd Remote Format String Vulnerability". 16 July 2000.
URL: http://www.securityfocus.com/bid/1480

[29] SecurityFocus. " Wu-Ftpd Remote Format String Stack Overwrite Vulnerability". 22 June 2000.
URL: http://www.securityfocus.com/bid/1387

[30] Buytenhek, Lennert; Hemminger, Stephen.
"SourceForge Project: IEEE 802.1d ethernet bridging".
URL: http://sourceforge.net/projects/bridge/

[31] SANS. "Need an Example Policy or Template?".
The SANS Security Policy Project.
URL: http://www.sans.org/resources/policies/#template

[32] Unofficial patch for the stream.c/raped.c vulnerabilities for FreeBSD, as mentioned in Hacking Exposed (see reference 5 above).
URL: http://www.freebsd.org/~alfred/tcp_fix.diff (the link isn't valid anymore)

[33] Livingston, Brian for CNN.com. "We can prevent those distributed denial of service attacks with 'egress filtering'". 1 March 2000.
URL: http://edition.cnn.com/2000/TECH/computing/03/01/prevent.ddos.idg/index.html

[34] Pappalardo, Denise for CNN.com. "Avoiding future denial-of-service attacks". 23 February 2000.
URL: http://edition.cnn.com/2000/TECH/computing/02/23/isp.block.idg/index.html