

a remote active OS fingerprinting tool using ICMP

by Ofir Arkin

Ofir Arkin is the Founder of the Sys-Security Group, a free computer security research body. In his free time he enjoys doing computer security research. His publications and work is available from the group's web site, <http://www.sys-security.com>.



ofir@sys-security.com

During the winter of 2000 I started researching the Internet Control Message Protocol (ICMP). The protocol goals and features were outlined in RFC 792 (and then later in RFCs 1122, 1256, 1349, 1812) as a means to send error messages for nontransient error conditions, and to provide a way to probe the network in order to determine general characteristics about it. My goal was to go through the relevant RFCs quickly and then continue with other more interesting protocols. Instead, I found that ICMP can be used to fingerprint operating systems.

Techniques for OS fingerprinting using TCP packets already exist and are well known: the nmap and queso tools are examples. As I continued to discover idiosyncrasies in the responses of different operating systems to small but legal tweaks to ICMP packets, I published information about the way these packets could be used to determine the type of operating system in use at a particular destination not filtering incoming (and in some cases, outgoing) ICMP packets. The first fruit of this research was a paper that can be found at <http://www.sys-security.com/html/>.

A large portion of the research paper is dedicated to active operating system fingerprinting techniques that I have discovered during the research project. Using active OS fingerprinting methods with ICMP requires less traffic initiation from the prober's machine in determining the underlying operating system of a targeted host. With most of the fingerprinting methods, only one datagram can be enough to accomplish this.

For quite some time people have asked me for an automated tool that will correlate some of the active OS fingerprinting methods I have discovered using ICMP. But the final push for the tool was done by J.D. Glaser, a good friend of mine, who asked me if I could use these ICMP fingerprinting methods to differentiate between Microsoft-based operating systems. Less than three hours later I had a little logic drawn, and tested, that could differentiate between the different Microsoft-based OSes using only one to three ICMP queries.

X and Xprobe

X is the logic which combines various remote active OS fingerprinting methods using ICMP into a simple, fast, efficient, and powerful way to detect an underlying operating system a host is using. Xprobe is the tool I wrote with Fyodor Yarochkin (fygrave@tigerteam.net) which automates X.

Xprobe is an alternative to some tools which are heavily dependent on TCP for remote active OS fingerprinting. This is especially true when trying to identify some Microsoft-based operating systems, when TCP is the protocol being used with the fingerprinting process. Since the TCP implementation with Windows XP, 2000 and ME, and with Windows NT4 and 98/98SE are so close, we are unable to differentiate between these Microsoft-based OS groups when using TCP with a remote active OS fingerprinting process. And this is only an example.

The number of datagrams we need to send and receive in order to remotely fingerprint a targeted machine with Xprobe is small. In fact we can send one datagram and

receive one reply and this will help us identify up to eight different operating systems (or groups of operating systems). The maximum datagrams the tool will send is four. This is the same number of replies we will need. This makes Xprobe very fast as well.

Xprobe probes can be very stealthy. Xprobe does not send any malformed datagrams to detect a remote OS type, unlike the common fingerprinting methods. Xprobe analyzes the remote OS TCP/IP stack responses for valid packets. Heaps of such packets appear in an average network on a daily basis and very few IDS systems are tuned to detect such traffic (and those which are, presumably, are very badly configured).

Usually people see the types of datagrams being used by Xprobe as evidence of a simple host detection attempt; in fact, the replying machines were not only detected, but their underlying operating systems were revealed as well. In the future, Xprobe will use actual application data with its probes. This will help in disguising the real intentions of the probes.

Xprobe might change the traditional intelligence-gathering approach. With the traditional approach we need to detect the availability of a host (using a host-detection method), find a service it is running (using port scanning), and then identify the underlying operating system (with a remote active OS fingerprinting tool). If the targeted machine is running a service that is known to be vulnerable, it may allow a malicious computer attacker to execute a remote exploit in order to gain unauthorized access to the targeted machine.

Xprobe takes advantage of several remote active OS fingerprinting methods discovered during the ICMP Usage in Scanning research project. Some operating system stacks do not correctly echo several field values within the same ICMP error message. This enables Xprobe to use multiple echoing integrity tests with just one ICMP error message sent by a targeted machine.

ICMP ERROR MESSAGE QUOTING SIZE

Each ICMP error message includes the IP header and at least the first eight data bytes of the datagram that triggered the error (the offending datagram); more than eight bytes may be sent according to RFC 1122. Most of the operating systems will quote the offending packet's IP header and the first eight data bytes of the datagram that triggered the error. Several operating systems and networking devices will echo more than eight data bytes. Examples of operating systems that quote more include: Linux based on kernel 2.0.x/2.2.x/2.4.x, Sun Solaris 2.x, HPUX 11.x, MacOS 7.x-9.x (10.x not checked), Nokia boxes, Foundry Switches (and other OSes and several networking devices).

ICMP ERROR MESSAGE ECHOING INTEGRITY

When sending back an ICMP error message, some stack implementations may alter the offending packet's IP header and the underlying protocol's data, which is echoed back with the ICMP error message. The only two field values we expect to be changed are the IP time-to-live field value and the IP header checksum. The IP time-to-live (TTL) field value changes because the field is decreased by one, each time the IP header is being processed. The IP header checksum is recalculated each time the IP TTL field value is decreased. With Xprobe we take advantage of ICMP Port Unreachable error messages triggered by UDP datagrams sent to closed UDP ports. Xprobe examines several IP header and UDP-related field values of the offending packet being echoed with the ICMP error message, for some types of alternation.

Xprobe probes can be very stealthy.

IP TOTAL LENGTH FIELD

Some operating systems add 20 bytes to the original IP total length field value of the offending packet in the data echoed with the IP header of the offending packet in the ICMP error message. Other operating systems subtract 20 bytes from the original IP total length field value of the offending packet. And some operating systems echo correctly this field value.

IP ID

Some operating systems do not echo the IP ID field value correctly with their ICMP error messages. They will change the bit order with the value echoed. Other operating systems will correctly echo this field value. Linux machines based on kernel 2.4.0–2.4.4 will set the IP identification field value with their ICMP query request and reply messages to a value of zero. This was fixed with Linux kernels 2.4.5 and up.

FRAGMENTATION FLAGS AND OFFSET FIELDS

Some operating systems do not echo the fragmentation flags and offset field values correctly with their ICMP error messages. They will change the bit order with these fields.

IP HEADER CHECKSUM

Some operating systems will miscalculate the IP header checksum of the offending packet echoed back with the ICMP error message. Some operating systems will zero out the IP header checksum of the offending packet echoed back with the ICMP error message. Other operating systems will correctly echo this field value.

UDP HEADER CHECKSUM

Some operating systems miscalculate the UDP header checksum of the offending packet echoed back with the ICMP error message. Some operating systems will zero out the UDP header checksum of the offending packet echoed back with the ICMP error message. Other operating systems will correctly echo this field value.

PRECEDENCE BITS ISSUES WITH ICMP ERROR MESSAGES

Each IP datagram has an 8-bit field called the TOS byte, which represents the IP support for prioritization and type-of-service handling. The TOS byte consists of three fields. The precedence field, which is 3 bits long, is intended to prioritize the IP datagram. It has eight levels of prioritization. Higher priority traffic should be sent before lower-priority traffic. The second field, 4 bits long, is the type-of-service field. It is intended to describe how the network should make tradeoffs between throughput, delay, reliability, and cost in routing an IP datagram. The last field is unused and must be zero. Routers and hosts ignore this last field. This field is 1 bit long.

RFC 1812 sets requirements for IPv4 routers, and this affects the TOS and precedence bits. ICMP Source Quench error messages, if sent at all, **MUST** have their IP precedence field set to the same value as the IP precedence field in the packet that provoked the sending of the ICMP Source Quench message. All other ICMP error messages (Destination Unreachable, Redirect, Time Exceeded, and Parameter Problem) **SHOULD** have their precedence value set to 6 (INTERNETWORK CONTROL) or 7 (NETWORK CONTROL). The IP precedence value for these error messages **MAY** be settable.

Linux kernels 2.0.x, 2.2.x, 2.4.x will act as routers and set their precedence bits field value to 0xc0 with ICMP error messages. Networking devices that will act the same will be Cisco routers based on IOS 11.x–12.x and Foundry Networks switches.

DF BIT ECHOING WITH ICMP ERROR MESSAGES

Some operating systems set the DF (don't fragment) bit in error quoting when the DF bit is set with the offending packet. Some OSs will not.

THE IP TIME-TO-LIVE FIELD VALUE WITH ICMP MESSAGES

The sender sets the time-to-live field to a value that represents the maximum time the datagram is allowed to travel on the Internet. In practice, the TTL gets decremented each time a packet passes through a router or IP stack. The TTL field value with ICMP has two separate values, one for ICMP query messages and one for ICMP query replies. The TTL field value helps identify certain operating systems and groups of operating systems. It also provides the simplest means to add another check criterion when we are querying other hosts or listening to traffic (sniffing).

USING CODE FIELD VALUES DIFFERENT FROM ZERO WITH ICMP ECHO REQUESTS

When an ICMP code field value different from zero is sent with an ICMP Echo Request message (type 8), operating systems that answer the query with an ICMP Echo Reply message based on one of the Microsoft-based operating systems send back an ICMP code field value of zero with their ICMP Echo Reply. Other operating systems (and networking devices) echo back the ICMP code field value that was used with the ICMP Echo Request.

TOS ECHOING

RFC 1349 defines the use of the type-of-service field with ICMP messages. It distinguishes between ICMP error messages (Destination Unreachable, Source Quench, Redirect, Time Exceeded, and Parameter Problem), query messages (Echo, Router Solicitation, Timestamp, Information Request, Address Mask Request), and reply messages (Echo Reply, Router Advertisement, Timestamp Reply, Information Reply, Address Mask Reply). Simple rules are defined: an ICMP error message is always sent with the default TOS (0x0000). An ICMP request message may be sent with any value in the TOS field. A mechanism to allow the user to specify the TOS value to be used would be a useful feature in many applications that generate ICMP request messages. The RFC further specifies that although ICMP request messages are normally sent with the default TOS, there are sometimes good reasons why they would be sent with some other TOS value. An ICMP reply message is sent with the same value in the TOS field as was used in the corresponding ICMP request message. Some operating systems will ignore RFC 1349 when sending ICMP Echo Reply messages and will not send the same value in the TOS field as was used in the corresponding ICMP request message.

HOW DOES XPROBE WORKS?

Currently Xprobe deploys hardcoded logic tree. Initially a UDP datagram is being sent to a closed UDP port in order to trigger an ICMP Port Unreachable Error message. This sets up a limitation of having at least one port not being filtered on a target system with no service running.

Few tests can be combined within a single query, since they do not affect results of each other.

Upon the receipt of the ICMP Port unreachable error message, contents of the received datagram is examined and a diagnostics decision is made. If any further tests are required, according to the logic tree, further queries are sent. For a detailed explanation and graphical representation of the logic please go to: <http://www.sys-security.com/html/projects/X.html>.

As always, an example is worth a thousand words...

```
[root@godfather /root]# xprobe -v www.redhat.com
X probe ver. 0.0.2
-----
Interface: ppp0/213.8.195.154
LOG: Target: 216.148.218.195
LOG: Netmask: 255.255.255.255
LOG: probing: 216.148.218.195
TEST: UDP to 216.148.218.195:32132 [98 bytes] sent, waiting for response.
TREE: Cisco IOS 11.x-12.x! Extreme Network Switches.Linux
2.0.x!2.2.x!2.4.x.
TREE: Linux kernel 2.0.x!2.2.x!2.4.x! Based.
TREE: Linux kernel 2.2.x!2.4.x! Based.
TEST: ICMP echo request to 216.148.218.195 [68 bytes] sent, waiting for
response.
TREE: ICMP echo/echo reply are not filtered
FINAL:[ Linux 2.2.x/2.4.5+ kernel ]
[root@godfather /root]#
```

The number of tests in the output is the number of datagrams sent (represented by the word TEST in the output). With the example above of the RedHat Linux Web site, Xprobe sent only two datagrams to the target. It took Xprobe 700 milliseconds to figure out what the underlying operating system was. The time can be even faster, depending on the type of link you have. On local LANs, you will get the fastest results. For each internal logic test, Xprobe prints the word TREE in the output, representing a decision-tree criterion we check the replies against.

Many sites will block most incoming UDP packets (for good reason), but not all. When probing msdn.microsoft.com, we found that UDP port 53 (used for DNS queries) was allowed through, and we used the -p 53 option to specify that to Xprobe:

```
[root@godfather /root]# xprobe -v -p 53 msdn.microsoft.com
X probe ver. 0.0.2
-----
Interface: ppp0/x.x.x.x
LOG: Target: 207.46.196.115
LOG: Netmask: 255.255.255.255
LOG: probing: 207.46.196.115
TEST: UDP to 207.46.196.115:53 [98 bytes] sent, waiting for response.
TREE: IP total length field value is OK
TREE: Frag bits are OK
TEST: ICMP echo request to 207.46.196.115 [68 bytes] sent, waiting for
response.
TREE: Microsoft Windows Family TCP stack
TREE: Other Windows-based OS (ttl: 48)
FINAL:[ Windows 2k. SP1, SP2 ]
[root@godfather /root]#
```

This time we succeed in our identification. This also means that UDP port 53 on `msdn.microsoft.com` is closed, but not covered by the firewall. It also means that ICMP Port Unreachable error messages are allowed from internal Microsoft systems to the outside world. The example above shows you why it is important to use egress and ingress filtering on your firewall.

In the future if we will receive an ICMP Destination Unreachable (Communication Administratively Prohibited) error message from a filtering device protecting the targeted host, we will be able to fingerprint the filtering device as well. Compared to other remote active operating systems fingerprinting programs, Xprobe is a very efficient one. Just remember that the official release is only 0.0.2 and that there is still a long way to go and a lot of enhancements to introduce.

Future Development

The following issues are planned to be deployed (we always welcome discussions/suggestions though):

- Fingerprints database (currently being tested)
- Dynamic, AI-based logic (long-term project :)
- Tests will heavily depend on network topology (pre-test network mapping will take place).
- Path-to-target test (to calculate hops distance to the target)
- Filtering devices probes
- Logging capabilities
- DB fingerprints creator

Future implementations will use packets with actual application data to dismiss chances of being detected. Other network mapping capabilities shall be included (network role identification, search for closed UDP port, reachability tests, etc.). For more information, visit: <http://www.sys-security.com/html/projects/X.html>.