

ICMP Based Remote OS TCP/IP Stack Fingerprinting Techniques

Ofir Arkin and Fyodor Yarochkin
(Phrack Magazine Classic)

ICMP Based Fingerprinting Approach

TCP based remote OS fingerprinting is quite old(*1) and well-known these days, here we would like to introduce an alternative method to determine an OS remotely based on ICMP responses which are received from the host. Certain accuracy level has been achieved with different platforms, which, with some systems or or classes of platforms (i.g. Win*), is significantly more precise than demonstrated with TCP based fingerprinting methods.

As mentioned above TCP based method, ICMP fingerprinting utilizes several tests to perform remote OS TCP/IP stack probe, but unlike TCP fingerprinting, a number of tests required to identify an OS could vary from 1 to 4 (as of current development stage).

ICMP fingerprinting method is based on certain discoveries on differences of ICMP replies from various operating systems (mostly due to incorrect, or inconsistent implementation), which were found by Ofir Arkin during his "ICMP Usage in Scanning" research project. Later these discoveries were summarised into a logical decisions tree which Ofir entitled "X project" and practically implemented in 'Xprobe' tool.

Information/Noise Ratio with ICMP Fingerprints

As it's been noted, the number of datagrams we need to send and receive in order to remotely fingerprint a targeted machine with ICMP based probes is small. Very 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 classes of operating systems). The maximum datagrams which our tool will use at the current stage of development, is four. This is the same number of replies we will need to analyse. This makes ICMP based fingerprinting very time-efficient.

ICMP based probes could be crafted to be very stealthy. As on the moment, no malformed/broken/corrupted datagrams are used to identify remote OS type, unlike the common fingerprinting methods. Current core analysis targets validation of received ICMP responses on valid packets, rather than crafting invalid packets themselves. Heaps of such packets appear in an average network on daily basis and very few IDS systems are tuned to detect such traffic (and those which are, presumably are very noisy and badly configured).

Why it still works?

Inheritable mess among various TCP/IP stack implementations with ICMP handling implementations which implement different RFC standards (original RFC 792, additional RFC 1122, etc), partial or incomplete ICMP support (various ICMP requests are not supported everywhere), low significance of ICMP Error messages data (who verifies all the fields of the original datagram?!), mistakes and misunderstanding in ICMP protocol implementation made our method viable.

What do we Fingerprint

Several OS-specific differences are being utilized in ICMP based fingerprinting to identify remote operating system type:

IP fields of an 'offending' datagram to be examined:

* IP total length field

Some operating systems (i.g. BSD family) will add 20 bytes (sizeof(ipheader)) to the original IP total length field (which occurs due to internal processing mistakes of the datagram, please note when the

ICMP Based Remote OS TCP/IP Stack Fingerprinting Techniques

Ofir Arkin and Fyodor Yarochkin
(Phrack Magazine Classic)

same packet is read from SOCK_RAW the same behaviour is seen: returned packet ip_len field is off by 20 bytes).

Some other operating systems will decrease 20 bytes from the original IP total length field value of the offending packet.

Third group of systems will echo this field correctly.

* IP ID

some systems are seen not to echo this field correctly. (bit order of the field is changed).

* 3 bits flags and offset

some systems are seen not to echo this field correctly. (bit order of the field is changed).

* IP header checksum

Some operating systems will miscalculate this field, others just zero it out. Third group of the systems echoes this field correctly.

* UDP header checksum (in case of UDP datagram)

The same thing could happen with UDP checksum header.

IP headers of responded ICMP packet:

* Precedence bits

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' [cite{rfc791}](#), which is 3-bit 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, the 'MBZ' (must be zero), is unused and must be zero. Routers and hosts ignore this last field. This field is 1 bit long. The TOS Bits and MBZ fields are being replaced by the DiffServ mechanism for QoS.

RFC 1812 Requires following for IP Version 4 Routers:

"4.3.2.5 TOS and Precedence

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

ICMP Based Remote OS TCP/IP Stack Fingerprinting Techniques

Ofir Arkin and Fyodor Yarochkin
(Phrack Magazine Classic)

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 Kernel 2.0.x, 2.2.x, 2.4.x will act as routers and will 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 bits echoing

Some TCP/IP stacks will echo DF bit with ICMP Error datagrams, others (like linux) will copy the whole octet completely, zeroing certain bits, others will ignore this field and set their own.

* IP ID field (linux 2.4.0 - 2.4.4 kernels)

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 later fixed with Linux Kernels 2.4.5 and up.

* IP ttl field (ttl distance to the target has to be precalculated to guarantee accuracy).

"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".

The field value is decreased at each point that the IP header is being processed. RFC 791 states that this field decrease reflects the time spent processing the datagram. The field value is measured in units of seconds. The RFC also states that the maximum time to live value can be set to 255 seconds, which equals to 4.25 minutes. The datagram must be discarded if this field value equals zero - before reaching its destination.

Relating to this field as a measure to assess time is a bit misleading. Some routers may process the datagram faster than a second, and some may process the datagram longer than a second.

The real intention is to have an upper bound to the datagram lifetime, so infinite loops of undelivered datagrams will not jam the Internet.

Having a bound to the datagram lifetime help us to prevent old duplicates to arrive after a certain time elapsed. So when we retransmit a piece of information which was not previously delivered we can be assured that the older duplicate is already discarded and will not interfere with the process.

The IP TTL field value with ICMP has two separate values, one for ICMP query messages and one for ICMP query replies.

The IP TTL field value helps us identify certain operating systems and groups of operating systems. It also provides us with the simplest means to add another check criterion when we are querying other host(s) or listening to traffic (sniffing).

TTL-based fingerprinting requires a TTL distance to the done to be precalculated in advance (unless a fingerprinting of a local network based system is performed system).

ICMP Based Remote OS TCP/IP Stack Fingerprinting Techniques

Ofir Arkin and Fyodor Yarochkin
(Phrack Magazine Classic)

The ICMP Error messages will use values used by ICMP query request messages.

* TOS field

RFC 1349 defines the usage of the Type-of-Service field with the ICMP messages. It distinguishes between ICMP error messages (Destination Unreachable, Source Quench, Redirect, Time Exceeded, and Parameter Problem), ICMP query messages (Echo, Router Solicitation, Timestamp, Information request, Address Mask request) and ICMP 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 specify 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.

ICMP headers of responded ICMP packet:

* ICMP Error Message Quoting Size:

All ICMP error messages consist of an IP header, an ICMP header and certain amount of data of the original datagram, which triggered the error (aka offending datagram).

According to RFC 792 only 64 bits (8 octets) of original datagram are supposed to be included in the ICMP error message. However RFC 1122 (issued later) recommends up to 576 octets to be quoted.

Most of "older" TCP stack implementations will include 8 octets into ICMP Error message. Linux/HPUX 11.x, Solaris, MacOS and others will include more.

Noticably interesting is the fact that Solaris engineers probably couldn't not read RFC properly (since instead of 64 bits Solaris 2.x includes 64 octets (512 bits) of the original datagram).

* ICMP error Message echoing integrity

Another artifact which has been noticed is that some stack implementations, when sending back an ICMP error message, may alter the offending packet's IP header and the underlying protocol data, which is echoed back with the ICMP error message.

Since mistakes, made by TCP/IP stack programmers are different and specific to an operating system, an analysis of these mistakes could give a potential attacker a a possibilty to make assumptions about

ICMP Based Remote OS TCP/IP Stack Fingerprinting Techniques

Ofir Arkin and Fyodor Yarochkin
(Phrack Magazine Classic)

the target operating system type.

Additional tweaks and twists:

- * Using difererent from zero code fields in ICMP echo requests

When an ICMP code field value different than zero (0) is sent with an ICMP Echo request message (type 8), operating systems that will answer our query with an ICMP Echo reply message that are based on one of the Microsoft based operating systems will send back an ICMP code field value of zero with their ICMP Echo Reply. Other operating systems (and networking devices) will echo back the ICMP code field value we were using with the ICMP Echo Request.

The Microsoft based operating systems acts in contrast to RFC 792 guidelines which instruct the answering operating systems to only change the ICMP type to Echo reply (type 0), recalculate the checksums and send the ICMP Echo reply away.

- * Using DF bit echoing with ICMP query messages

As in case of ICMP Error messages, some tcp stacks will respond these queries, while the others: will not.

- * Other ICMP messages:

- * ICMP timestamp request
- * ICMP Information request
- * ICMP Address mask request

Some TCP/IP stacks support these messages and respond to some of these requests.

Xprobe Implementation

Currently Xprobe deploys hardcoded logic tree, developed by Ofir Arkin in 'Project X'. Initially a UDP datagram is being sent to a closed port in order to trigger ICMP Error message: ICMP unreachable/port unreach. (this sets up a limitation of having at least one port not filtered on target system with no service running, generically speaking other methods of triggering ICMP unreach packet could be used, this will be discussed further). Moreover, a few tests (icmp unreach content, DF bits, TOS ...) could be combined within a single query, since they do not affect results of each other.

Upon the receipt of ICMP unreachable datagram, 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.

Logic Tree

Quickly recapping the logic tree organization:

Initially all TCP/IP stack implementations are split into 2 groups, those which echo precedence bits back, and those which do not. Those which do echo precedence bits (linux 2.0.x, 2.2.x, 2.4.x, cisco IOS 11.x-12.x, Extreme Network Switches etc), being differentiated further based on ICMP error quoting size. (Linux sticks with RFC 1122 here and echoes up to 576 octets, while others in this subgroup echo only 64 bits (8 octets)). Further echo integrity checks are used to differentiate cisco routers from Extreme Network switches.

Time-to-live and IP ID fields of ICMP echo reply are being used to recognize version of linux kernel.

ICMP Based Remote OS TCP/IP Stack Fingerprinting Techniques

Ofir Arkin and Fyodor Yarochkin
(Phrack Magazine Classic)

The same approach is being used to recognize other TCP/IP stacks. Data echoing validation (amounts of octets of original datagram echoed, checksum validation, etc). If additional information is needed to differ two 'similar' IP stacks, additional query is being sent. (please refer to the diagram at <http://www.sys-security.com/html/projects/X.html> for more detailed explanation/graphical representation of the logic tree).

One of the serious problems with the logic tree, is that adding new operating system types to it becomes extremely painful. At times part of the whole logic tree has to be reworked to 'fit' a single description. Therefore a signature based fingerprinting method took our closer attention.

Signature Based Approach

Signature based approach is what we are currently focusing on and which we believe will be further, more stable, reliable and flexible method of remote ICMP based fingerprints.

Signature-based method is currently based on five different tests, which optionally could be included in each operating system fingerprint. Initially the systems with lesser amount of tests are being examined (normally starting with ICMP unreachable test).

If no single OS stack found matching received signature, those stacks which match a part, being grouped again, and another test (based on lesser amounts of tests issued principle) is chosen and executed. This verification is repeated until an OS stack, completely matching the signature is found, or we run out of tests.

Currently following tests are being deployed:

- * ICMP unreachable test (udp closed port based, host unreachable, network unreachable (for systems which are believed to be gateways)
- * ICMP echo request/reply test
- * ICMP timestamp request
- * ICMP information request
- * ICMP address mask request

Future Implementations/development

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 would heavily dependent on network topology (pre-test network mapping will take place).
- * Path-to-target test (to calculate hops distance to the target) filtering devices probes.
- * Future implementations will be using 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).

Code for Kids

Currently implemented code and further documentation is available at following locations:

- <http://www.sys-security.com/html/projects/X.html>
- <http://xprobe.sourceforge.net>
- <http://www.notlsd.net/xprobe/>