The Trouble with UDP Scanning

John Kristoff

DePaul University
Chicago, IL
jtk@depaul.edu

March 11, 2002

Abstract

Actively probing a TCP/IP host with network packets can offer a great deal of insight into the remote host's capabilities, configuration and even vulnerabilities. Along with other common protocols, user datagram protocol (UDP) probes can and have been used to help determine a remote system's characteristics. However, as we show in this paper, using UDP probes is susceptible to failure in a number of ways and often cannot be used reliably. In addition, we will examine UDP scanning techniques and some common misinterpretations of the results. Finally, we'll discuss the use of UDP application specific scanning in order to achieve more accurate probing results.

1 Introduction

Since at least 1993 when Christopher Klaus posted the source code to a tool called the Internet Security Scanner (ISS) on Usenet, numerous software tools have been written to help administrators remotely probe TCP/IP hosts for common vulnerabilities accessible over the network. While many newer tools have been written or used with malicious intent, being able to remotely determine system characteristics through the use of network packets has become popular for both administrators and attackers ever since. Today remotely probing, also known simply as *scanning*, may involve the use of sophisticated protocol and application semantics to infer specific information about a host or network. Perhaps the most widely known scanning tool is Fyodor's *nmap*, which uses low level packet routines to audit a remote host.[1]

Recent scanning tools such as *nmap* have been built for general purpose remote system mapping and others such as *ScanSSH* have been built to probe for specific versions of applications running on end hosts.[2] Going a step further, some tools have incorporated the ability to test for a complete set of vulnerabilities on end systems.[3][4][5]

Tools such as *nmap*, which use low level packet probes to discover system characteristics rely solely on IP, ICMP, UDP and TCP protocols for system mapping. Low level packet probes can be amazingly accurate.[1] However, scanning tools can suffer from false positives and false negatives.

In this paper we focus our attention on the problem of correctly scanning host systems with UDP specific probes. The trouble with UDP scanning is that the results can be misleading. The technical details of remote system probing using UDP will help the reader understand how to interpret their own scanning results.

The remainder of this paper is organized as follows. Section 2 provides an overview of the user datagram protocol. The focus of Section 3 is on how network protocols can be used to remotely probe end systems, including how UDP scanning compares to ICMP and TCP scanning. Section 4 details possible failure scenarios that a UDP scanning session may incur. Section 5 discusses the use of UDP application layer scanning. Section 6 is the conclusion of the paper.

2 An Overview of UDP

The user datagram protocol (UDP) is perhaps one of the simplest of all TCP/IP suite protocols.[6] The specification is only three pages long, the header format contains only four fields (not including the message data) and two of the four fields are optional. UDP provides the minimal functionality needed to use IP's raw datagram delivery services. A UDP message contains a pair of source and destination port fields, a length field and an optional checksum that verifies the entire UDP message including data (see Figure 1).

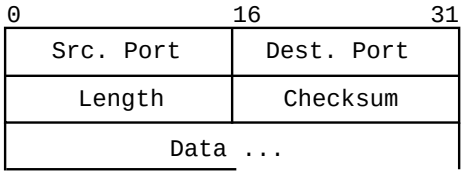

Figure 1 - User Datagram Protocol

The 16-bit source port field is optional, set to zero in the absence of an associated source process to which the receiver can reply, but it is almost always used even if response packets are not expected.

The 16-bit destination port field is required and it identifies the remote UDP process on the receiving system.

The 16-bit length field is the length of the UDP message including the header and data fields. The minimum length is eight and is measured in octets (e.g. 0x08).

The 16-bit checksum is the 16-bit one's complement of the one's complement sum of the psuedo IP header, UDP header and UDP data. If necessary, the UDP

[1] On a local subnet at DePaul *nmap* can correctly identify over 90% of all host systems.

data is padded with zeroes-filled octets to make a multiple of two octets. While optional, it is recommended that applications always compute the checksum on transmission. Applications not computing a checksum will fill the field with zeroes. Receivers must verify the checksum if one is present.

UDP is a connectionless, unreliable message delivery protocol. That is, there is no predetermined setup procedure for two communicating UDP processes and when the sending UDP process formats the message, it delivers the message to the network and forgets about it. Likewise at the receiver, a UDP message is handed to the appropriate destination process without regard for the sending host. It is up to receiving application to properly demultiplex UDP messages if the receiver must send reply messages.

Lacking reliability, UDP messages may be lost due to error or congestion without any notification to the sender or receiver. Applications using UDP should be willing to accept the trade-off of reliability for simplicity or must provide their own application-level reliability mechanisms. A number of applications can take advantage of UDP's lightweight design if they do not require sophisticated transport layer service. For example, real-time multimedia services would typically prefer to avoid lost message recovery, because in this case, recovery would not benefit a stream as it is being played out in real-time.

A number of applications use UDP as their transport layer protocol. DNS, SNMP, TFTP, DHCP, NTP and RIP are among the class of protocols that make use of UDP's simple message delivery service. For a more complete list of UDP-based applications the Internet Assigned Numbers Authority (IANA) maintains the well known, registered, dynamic and private port number assignments at *http://www.iana.org/assignments/port-numbers*.

3 Basic Scanning Techniques

Since a UDP process does not reply to received UDP messages on its own, a pure UDP scanner is impossible to build without the help of other protocol mechanisms. For low level packet probing tools such as *nmap*, the technique used is to monitor for returning ICMP port unreachable messages. An ICMP port unreachable message should be sent in response to a packet that cannot be demultiplexed to a receiving UDP process.

Therefore, a UDP scan is the combination of sending UDP probe packets and monitoring for returning ICMP port unreachable messages. If a port unreachable message is returned, the UDP port being probed is closed. Otherwise, if no unreachable message is received, the sender assumes the port is open. Figure 2 demonstrates this process.

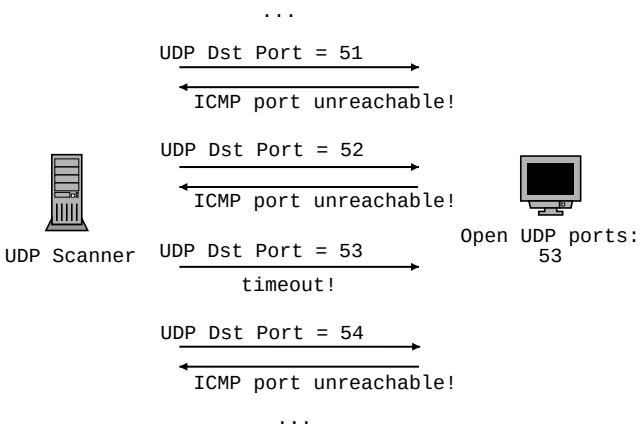

Figure 2 - UDP Scanning Technique

This basic technique is pretty much it for UDP port scanning. Later in the paper we'll discuss failure modes with this simple UDP scanning technique as well as how UDP application specific probes may be able to detect active UDP processes more accurately. ICMP mapping and TCP port probing on the other hand can use a number of alternative scanning methods. Furthermore, both ICMP and TCP can solicit responses from the remote system, which if returned can be a very reliable mechanism for system scanning compared to using UDP scanning probes and ICMP replies (or lack thereof).

The most basic TCP scanning technique is to issue a standard *connect()* request to the destination port. If the TCP destination port is available, the reliable connection startup, called the *3-way handshake*, should proceed normally. If the 3-way handshake completes successfully, the port is open. Unlike a UDP scan, a TCP listening process will acknowledge valid packets sent to it (unless some type of advanced filtering or application security were configured to limit TCP communications to or from a selected set of hosts and/or ports). In addition, TCP may respond to other packet probe types. For example Section 3.4 of RFC 793 says TCP must return a RST when a segment arrives on a closed port.[2][7] For a detailed description of various TCP port scanning techniques, *nmap's* documentation provides a fairly complete account of TCP probing strategies.[1]

While ICMP lacks TCP's reliable acknowledged delivery service, it does solicit various responses to numerous types of ICMP message queries. For example, an ICMP echo request message solicits an ICMP echo reply from the probed host.[8] For an excellent analysis of ICMP scanning techniques see Ofir Arikin's *ICMP Usage in Scanning* whitepaper.[9]

[2] Although some systems do violate this behavior.

4 UDP Scanning Failure Scenarios

While UDP should not be dismissed for use in scanning, its operation and the scanning results it generates need to be fully understood by those using them. Only until UDP's failure modes in scanning are realized can the results be interpreted properly.

As noted in the previous section, UDP scanning relies on the presence or absence of ICMP port unreachable replies from the probed host. To understand the most fundamental failure scenario, examine what RFC 1122, Requirements for Internet Hosts says in section 3.2.2.1[10]:

```
A host SHOULD generate Destination Unreachable messages with
code:

2    (Protocol Unreachable), when the designated transport
     protocol is not supported,; or

3    (Port Unreachable), when the designated transport
     protocol (e.g., UDP) is unable to demultiplex the
     datagram but has no protocol mechanism to inform the
     sender.
```

Note the use of the word *SHOULD*. While it is recommended that a host generate an ICMP unreachable message to a probe packet received on a closed UDP port, it is not required. In fact, there are classes of systems that send unreachables by default and others that do not. Therefore, for systems that do not send ICMP unreachable messages, standard UDP scanning as described earlier will fail. A basic UDP scanner that probes a host who does not return ICMP port unreachable messages can only assume that all UDP ports are open on the remote system.

ICMP unreachables may fail to return for a number of other reasons. First, filtering between the scanner and probed host may be preventing unreachable messages from returning. Filtering may also prevent the original UDP probes from reaching their destination. Novice users of UDP scanning tools are often fooled into thinking remote systems have large numbers of open UDP ports when firewalls block either UDP probes or returning ICMP replies.

Second, many systems allow an administrator to alter the behavior of their protocols including the ability to disable ICMP unreachable messages. On a system that would normally return unreachable messages, silence may again lead a novice into believing UDP ports are open when they are in fact not.

Third, UDP messages or ICMP unreachable reply messages can be lost in transit when less than perfect network conditions exist. While generally rare in most circumstances, the lack of reliability in UDP and ICMP make this a possible failure scenario since lost UDP or ICMP messages will not be retransmitted without application layer intervention.

Fourth, like loss, UDP and ICMP messages can be subject to error in transit. Errored packets are generally as good as lost since they may be misrouted or dropped before reaching their destination. Even if errored packets do reach their

destination, the packet contents are unreliable and should be silently discarded. As in lost packets, errored UDP and ICMP packets will not be retransmitted unless an upper layer application implements its own recovery mechanisms.

Fifth, many hosts implement an ICMP message rate limiting recommendation given in Section 4.3.2.8 of RFC 1812, Requirements for IP Version 4 Routers.[11] If UDP port probing against a particular host is done too quickly, the probed host may not send all ICMP unreachable messages it otherwise would have. Exceeding the solicitation rate for ICMP unreachable messages therefore may fail to properly identify closed UDP ports if ICMP messages are not sent.

SATAN was one of the first scanners to implement tactics that reduced UDP port probing false positives by implementing checks for ICMP replies, congestion avoidance and round trip time estimates.[12] Other tools such as *pscan*, *netcat* and *nmap* attempt to avoid false positives by mimicking *SATAN's* techniques or by using more crude false positive checks.[13][14][1]

5 UDP Application Scanning

While some UDP applications do not ever respond to properly formatted UDP messages (e.g. SYSLOG), many do (e.g. SNMP, DNS, TFTP and NTP). Making UDP scanning more accurate requires UDP application specific scanners. If UDP packets can generate UDP application replies on the remote host UDP specific scans will accomplish a great deal more reliability in remote UDP port probing than today's basic UDP port scanners. The challenge for UDP application specific scanning may be to come up with the proper UDP application message formats that will solicit replies from remote systems. While some UDP application specific scanners exist, many only check for specific well known UDP application vulnerabilities or are still in the crude development stages.

While UDP application specific probing is not completely resilient to failure scenarios, they are significantly more reliable than basic UDP port probing that most scanning tools current employ. However, an additional failure scenario exists with application specific probes. Which UDP destination port do you send an application specific probe to? While standard port assignments exist, many applications use dynamically generated UDP ports. In addition, over-zealous administrators may have setup standard UDP applications on non-standard ports as an attempt to hide from remote probes. While this security by obscurity technique is prone to failure, it does extend the UDP application specific scanning time considerably.

6 Conclusions and Future Work

Clearly UDP scanning is susceptible to failure and it is easy for results to be misinterpreted. While UDP scanning can be a useful tool, it is important for users of UDP scanning tools to understand UDP scanning limitations and recognize when results by scanning tools may be misleading. While tools such as

nmap are able to get the most out of low level packet probes, more sophisticated UDP application scanning mechanisms are needed to reliably ascertain remote UDP process listeners.

UDP scanning is inherently difficult. We believe the only way to make UDP scanning more reliable is to build application specific scanners. Some tools already exist, but currently are not widely used as they are often system or even application implementation specific. In addition, performing UDP application scanning sweeps against potentially 64K ports is cumbersome. So while reliability may potentially increase with application specific scanning, so must the scanner's bandwidth and patience.

7 Acknowledgments

This paper results from a long standing request by Damir Rajnovic. Long overdue, but better late than never. We hope this paper addresses the needs expressed in those original discussions. In addition, the initial collaboration began with Nils Magnus, Klaus Moeller and Rob Thomas. While this paper would have been better with more of their input, it would have been worse if they had not initially been involved. Finally, thanks to the FIRST team members who were able to provide last minute feedback that helped improve this paper.

References

[1] Fyodor. The Network Mapper (nmap), Available at http://www.insecure.org/nmap/.

[2] Niels Provos. ScanSSH, Available at http://www.monkey.org/~provos/scanssh/.

[3] W. Venema and D. Farmer. Security Administrator Tool for Analyzing Networks (SATAN). Available at http://www.porcupine.org/satan/.

[4] Internet Security Systems. Internet Scanner, Available at http://www.iss.net.

[5] Nessus Security Scanner. Available at http://www.nessus.org.

[6] J. Postel. User Datagram Protocol. RFC 768, August 1980.

[7] J. Postel. Transmission Control Protocol. RFC 793, September 1981.

[8] J. Postel. Internet Control Message Protocol. RFC 792, September 1981.

[9] O. Arkin. ICMP Usage in Scanning - The Complete Know How, Version 3.0. Available at http://www.sys-security.com/html/papers.html.

[10] R. Braden. Requirements for Internet Hosts – Communications Layers. RFC 1122, October 1989.

[11] F. Baker. Requirements for IP Version 4 Routers. RFCC 1812, June 1995.

[12] Wietse Venema. Private communications, March 2002.

[13] pluvius. pscan.c version 1.2. January, 1995.

[14] Hobbit. Netcat v1.10. Available at http://www.atstake.com/research/tools/.