



## Identifying Open UDP Ports in NET+OS Applications

## 1 Document History

Date	Version	Change Description	
10/1/2010	V1.0	Initial Entry	

## 2 Table of Contents

1	Document History.....	2
2	Table of Contents.....	3
3	Introduction.....	4
3.1	Problem Solved.....	4
3.2	Scope.....	4
4	Basics.....	4
4.1	Analyzing ports using nmap.....	6
4.2	What if I'd like to close some of these ports?.....	6
5	Conclusion.....	7

### **3 Introduction**

We have received a number of questions asking for the identification of open UDP ports in Digi NET+OS applications. This is especially true for customers whose applications might not explicitly be opening UDP ports. This paper's intent is to identify UDP ports opened by the NET+OS operating system in hopes of allaying any confusion relative to these ports.

#### **3.1 Problem Solved**

Identify ports created by the NET+OS operating system, and tie those ports to NET+OS components.

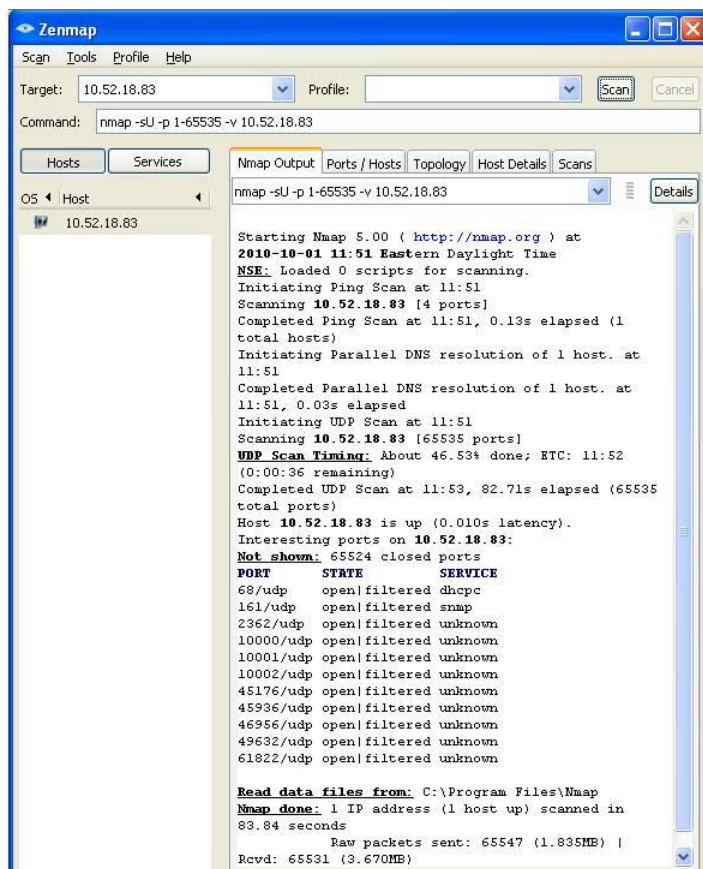
#### **3.2 Scope**

- The one and only scope of this paper is to identify UDP ports created by the NET+OS operating system, to aid in the understanding of what goes on “under the hood.”

### **4 Basics**

The ports identified by this document are relative to the sample application nasnmpd as defined in the directory \src\examples of a NET+OS operating environment installation. The screen shot below shows the output from running a port scan against a Digi development board using the nmap product.

## Identifying Open UDP Ports in NET+OS Applications



```
Starting Nmap 5.00 ( http://nmap.org ) at
2010-10-01 11:51 Eastern Daylight Time
NSE: Loaded 0 scripts for scanning.
Initiating Ping Scan at 11:51
Scanning 10.52.18.83 [4 ports]
Completed Ping Scan at 11:51, 0.13s elapsed (1
total hosts)
Initiating Parallel DNS resolution of 1 host. at
11:51
Completed Parallel DNS resolution of 1 host. at
11:51, 0.03s elapsed
Initiating UDP Scan at 11:51
Scanning 10.52.18.83 [65535 ports]
UDP Scan Timing: About 46.53% done; ETC: 11:52
(0:00:36 remaining)
Completed UDP Scan at 11:53, 82.71s elapsed (65535
total ports)
Host 10.52.18.83 is up (0.010s latency).
Interesting ports on 10.52.18.83:
Not shown: 65524 closed ports
PORT      STATE      SERVICE
68/udp    open|filtered dhcpc
161/udp   open|filtered snmp
2362/udp  open|filtered unknown
10000/udp open|filtered unknown
10001/udp open|filtered unknown
10002/udp open|filtered unknown
45176/udp open|filtered unknown
45936/udp open|filtered unknown
46956/udp open|filtered unknown
49632/udp open|filtered unknown
61822/udp open|filtered unknown

Read data files from: C:\Program Files\Nmap
Nmap done: 1 IP address (1 host up) scanned in
83.84 seconds
Raw packets sent: 65547 (1.835MB) |
Rcvd: 65531 (3.670MB)
```

#### **4.1 Analyzing ports using nmap**

You'll notice that in the port scan output above, that nmap identified 11 open UDP ports. We will now identify each port, according to its usage.

- 68 – DHCP
- 161 – SNMP agent
- 2362 – ADDP
- 46956 – DNS client
- 10000 – port created by the application for listening for UDP requests. Causes a trap to be sent out by the application.
- 10001 – port created by the application for listening for UDP requests. Causes a trap to be sent out by the application.
- 10002 – port created by the application for listening for UDP requests. Causes a trap to be sent out by the application.
- 45176 – serial port. The Treck stack allows the serial ports to be handled by the select all. It does this by considering it a pseudo UDP port. You cannot actually perform any network activity on it.
- 49632 – serial port. The Treck stack allows the serial ports to be handled by the select all. It does this by considering it a pseudo UDP port. You cannot actually perform any network activity on it.
- 45936 – trap port started by SNMP for sending out such traps as coldstart and authentication traps
- 61822 – trap port created within the application as a result of calling `naSnmpSendTrap()`.

It needs to be noted that many of these ports are ephemeral ports. That is, they are allocated by the operating system when the client requests a port and thus you may see different port numbers used from one run to another. Also many of these ports are created by the service that is using them at startup, and stays tied to that port until you reboot your system. This is done for efficiency as opposed to closing and opening a new port each time a trap needs to be sent.

#### **4.2 What if I'd like to close some of these ports?**

Digi's NET+OS development environment gives the developer some level of flexibility in configuring his or her system. Some of this configuration will cause the associated UDP ports to be closed. For example, if you configure ADDP out of your system, the ADDP port will not be open. If you use a static IP address, as opposed to using DHCP for acquiring an IP address, the port associated with DHCP will not be opened. If you do not include SNMP or traps in your application, then the ports associated with SNMP and traps will not be opened. If you do not create any serial ports, then the two ports associated with your system's serial ports will not be opened.

Needless to say, these are all application specific decisions, and as such, the final decision on how to manage these resources is up to you the developer and is outside of the scope of this paper.

### **5 Conclusion**

Hopefully this paper has cleared up some of the confusion caused by performing a port scan against your Digi product and finding certain ports open. You can, as needed, discontinue the use of certain components of your Digi NET+OS development system and thus cause certain ports to be closed. This would be an individual customer decision. That is, it is up to you.