

Dynamic Port Scanning

An Integration of ARP Poisoning into Port Scanning To Dynamically Spoof Source IP
AR (ar[at]securebits[dot]org) HK (hk[at]securebits[dot]org)
SecureBits [<http://www.securebits.org>]

Introduction

Dynamic Port Scanning is a new methodology that aims to dynamically spoof the source IP of the scanning machine. What is meant by “dynamic spoofing” is that each TCP or UDP scan packet has a randomly generated IP address. However, that IP address must fall within the local subnet IP range of the scanning machine. The underlying implementation of such methodology is solely dependent on the integration of ARP poisoning/spoofing into the scanning process.

ARP poisoning/spoofing has been in place since the creation of TCP/IP protocols. It has been known and used for network traffic sniffing and interception in switched network. However, this paper will show how ARP poisoning/spoofing could be used in conjunction with port scanning to achieve the dynamic way of spoofing the source IP of the scanning machine.

In general, Dynamic Port Scanning [DPS] is implemented by ensuring that the ARP cache of the target host or even the default gateway is poisoned by fake IP/MAC entry which allows scan reply packets to carry the MAC address of the scanning machine. Although the reply packet is destined to a fake IP address (i.e. the spoofed IP), the placement of scanner’s MAC address as destination MAC address in the reply packet enables that packet to arrive correctly at the scanning machine. The process of poisoning the remote ARP cache is done for each TCP/UDP scan packet that carries a spoofed IP address.

This paper will discuss deeply the process of Dynamic Port Scanning [DPS]. First, the paper examines current methodologies used in spoofing the source IP while scanning. Then, it will describe the new methodology along with TCP scan types. Finally, a new open-source tool called Dynamic Port Scanner [DPS], which does exactly the technique described in this paper, is outlined.

Overview of Current Spoofing Techniques

Currently, there are three methods an attacker can use to spoof the source IP of the scanning machine and/or to disturb the attention of system and security administrators. These techniques are: Normal Spoofing Scan, Decoy Scan, and Distributed Scan. In the following subsections, each of these techniques is described along with their advantages and disadvantages.

Normal Spoofing Scan

This is the simplest among all other technique. All the attacker needs to do is to spoof the source IP of the scanning machine to any other IP without worrying about anything

else. That spoofed IP is used for all scan packets. Also, that spoofed IP can be any valid IP address and does not have to be within the subnet IP range of the scanning machine.

This normal spoofing could be done with the (-S) switch of nmap tool:

```
# nmap -sS -S 217.64.121.34 -P0 -p 1-1024 64.23.16.21
```

However, this technique suffers from a major drawback. That is, there will be no results since all replies will be forwarded to the spoofed IP. The scanning machine will never receive any of those replies. One reason an attacker might attempt such type of spoofing is to fool the scanned target into thinking that somebody else – probably a competitor – is scanning them. The attacker here is not concerned about the replies or about the port status of the target.

Advantages of this spoofing technique are:

- Freedom of spoofing. The attacker is not bounded by a specific range of IPs.
- No wasted or unneeded initiated packets. The attacker sends one TCP/UDP packet per port.
- No tracing of the original scanner. Detection of the scanning machine is impossible at the IP layer.

Disadvantages of this technique are:

- No replies. There will be no reply packets arriving at the scanning machine.
- No results. Since replies are not received, the attacker won't know port status.

Decoy Port Scan

Decoy scan works by sending more than one packet per port. All of these packets carry spoofed source IPs except one packet, which carries the original scanner IP address. By doing so, the attacker guarantees at least one reply packet which is the reply to the scan packet carrying the correct IP address. All other replies will not reach the scanning machine. This scan type is done using the (-D) switch of nmap tool as follows:

```
# nmap -sS -P0 -D217.89.54.23,64.56.23.21,98.76.54.32 -p1-1024 10.10.10.10
```

Decoy port scan is done to make detection of the original scanner harder. The administrator of the scanned target cannot tell exactly which one of the used IPs is the real scanner's IP. However, if all IPs were investigated, investigation could lead to the original scanning IP.

Advantages of Decoy scan are:

- Results are guaranteed. Since reply packets arrive at the scanning machine, the attacker can have true results of port status.
- Freedom of spoofing. Each spoofed IP used in the decoy is not bounded by any set of IPs

Disadvantages of Decoy scan are:

- Detection is not impossible though it is hard. Since all used IPs are logged in a way or another on the target system, heavy investigation could lead to the original attacker.
- Lots of traffic. Since for each scanned port there are many packets, this will increase the traffic flow.

Distributed Scan

Distributed scan works by dividing the scanning scope among multiple attack platforms. In such case, each attack platform performs a normal scan for a small range of port numbers. Although this is not 100% spoofing mechanism, it increases the overhead of the system administrator on the other side to trace back the attacker [e.g. there could be hundreds of originating IPs.] Furthermore, those originating IPs could be compromised hosts of innocent people.

Advantages of Distributed Scan are:

- It minimizes the scan time since multiple scanning platforms are working in parallel.
- Tracing back the attacker is a little hard since there are many originating IPs appearing in the logs of the scanned network.

Disadvantages of Distributed Scan are:

- It requires that the attacker is controlling, in one way or another, all the scanning platforms.
- There is no real spoofing in this technique since all the IP addresses appearing in the scanned network are truly the IPs of the scanning machine.

Integrating ARP Poisoning with Port Scanning

The concept of Dynamic Port Scanning is based on the integration of ARP poisoning/spoofing into the scanning process. However, before describing the technique of Dynamic Port Scanning, a brief review of ARP poisoning will be presented.

Overview of ARP Poisoning

ARP poisoning/spoofing has been used and known since the invention of TCP/IP. It exploits an implementation flaw in ARP protocol. Since ARP is a stateless protocol, that is, a host does not keep track of ARP requests or ARP replies, it is easy for an attacker to poison the ARP cache of a host with false IP/MAC entry. In the old days, it was enough to send one fake ARP reply to poison the remote host. This would poison WIN95, WIN98, WIN ME, and Cisco routers. Check the illustration below:

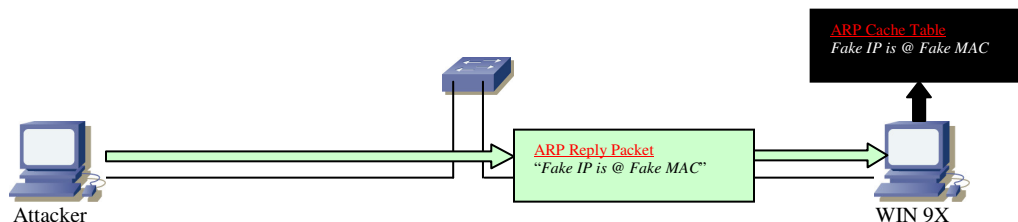


Figure1: For win9x and Cisco routers, one fake ARP reply is sufficient to poison the ARP cache.

Nowadays, modern Operating Systems (i.e. Linux, WIN2K, and WINXP) cannot get poisoned with just one fake ARP reply. It is necessary that a fake packet (i.e. ICMP, TCP, ARP ...) containing the false IP/MAC is sent prior to sending the fake ARP reply. For example, an attacker can send an ICMP ECHO request followed by an ARP reply; both of these packets contain the same fake IP/MAC addresses. Another way is to send a fake ARP Request packet followed by a fake ARP reply. Check the illustration below:

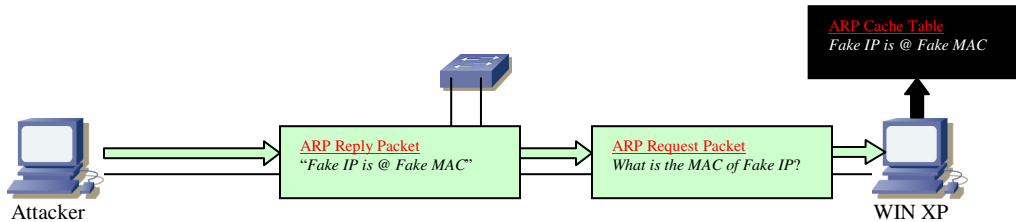


Figure2: For WinNT/2K/XP and Linux, two fake ARP packets are needed to poison the ARP cache.

Once the ARP cache of a host is poisoned, any packet intended to be transmitted to the spoofed IP, it will carry the MAC address associated with that IP in the ARP cache. That MAC address is usually the attacker's MAC address. Thus, the packet will arrive to the attacker's machine although the IP address is different. This method of sending two fake ARP packets will be integrated within Dynamic Port Scanning [DPS] technique.

Dynamic Port Scanning

After the previous overview of ARP poisoning/spoofing, we will describe now the Dynamic Port Scanning [DPS] mechanism. In general, the number TCP/UDP scan packets required to perform a port-scan is the same as the number of the ports needed to be scanned. For example, if we want to scan 100 ports on a remote machine, our scan will require 100 TCP/UDP scan packets. Note, this is always the case except in using TCP Vanilla [i.e. Full Connect] scan.

Now, the mechanism of Dynamic Port Scanning will work as follows:

- [1] Before sending any TCP/UDP scan method, a random IP address which falls within the local subnet range of the scanning machine is generated. This random IP address is inserted in the "source IP address" of the TCP/UDP scan packet; particularly, in the "Source IP" field of IP header.
- [2] The target machine IP address is checked to see if it is within the local subnet or not. If it is within the local subnet, this IP is marked as "ARP-Poisoning IP." If it is not within the local subnet, the IP address of the gateway is picked up [i.e. the IP of the router] and marked as "ARP-Poisoning IP"
- [3] Two "fake" ARP packets are sent to the "ARP-Poisoning IP." The first packet is an "ARP Request" while the second is an "ARP Reply". Both of these two packets contain the exact IP/MAC address combination in their fields. The addresses are as follows:

- Source IP: The randomly-generated fake IP (step 1)
- Destination IP: The “ARP-Poisoning IP” (step 2)
- Source MAC: MAC address of the attack machine
- Destination MAC: MAC address of the “ARP-Poisoning IP”

[4] Following the two ARP packets, the TCP/UDP scan packet is sent carrying the following data in its fields:

- Source IP: The randomly-generated fake IP (step 1)
- Destination IP: the IP address of the target machine
- Source Port: randomly generated port number
- Destination Port: port number to scan

[5] The reply of the scan packet is now guaranteed to reach back to the scanning machine. If the target machine is within the local subnet, it will send the packet directly to MAC address of the scanning machine despite the fact that the destination IP address will be the “randomly-generated fake IP.” If the target is not within the local subnet, the reply packet will be sent back to the “randomly-generated fake IP” and the gateway [i.e. router] will take care of placing the MAC address of the scanning machine in the destination MAC once it consults its ARP cache table.

The same process is repeated with every scan packet except step 2, it is done only once at the first time.

Here is the same process again, but, in charts:

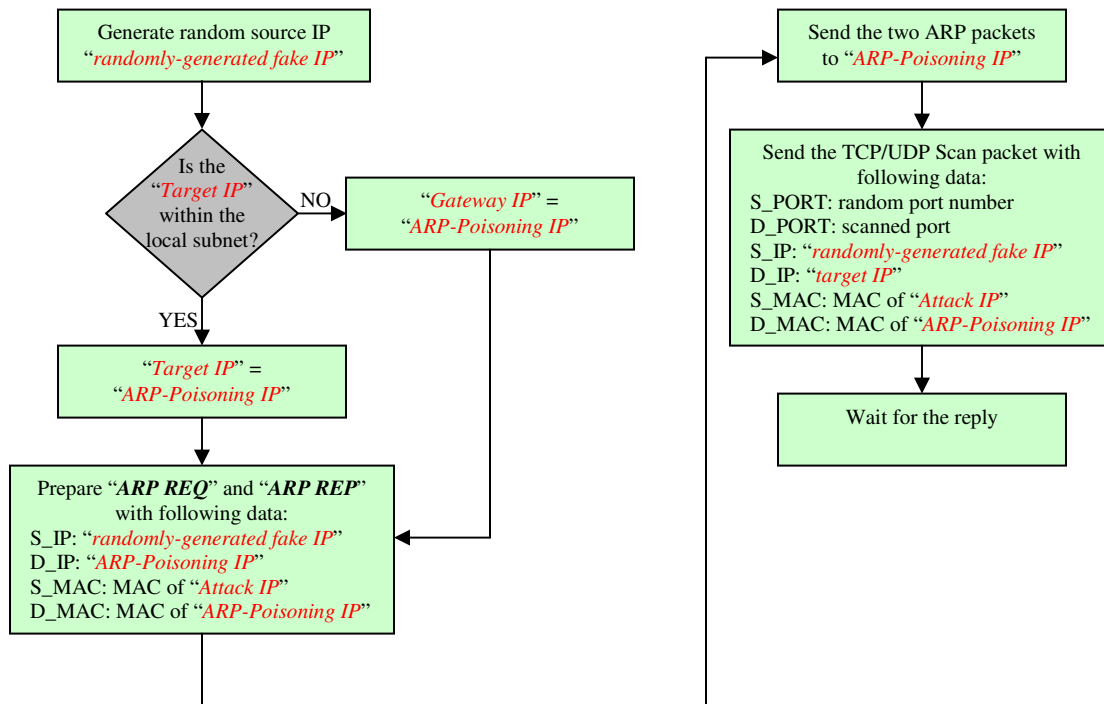


Figure3: A flowchart describing the internal mechanism of Dynamic Port Scanning [DPS]

To graphically visualize what goes on the network, the network monitoring tool, “EtherApe” was installed to see the difference between Normal Spoofing Scan [i.e. one spoofed IP] and Dynamic Scan [i.e. Dynamically-generated spoofed IPs], the following figure shows the results:

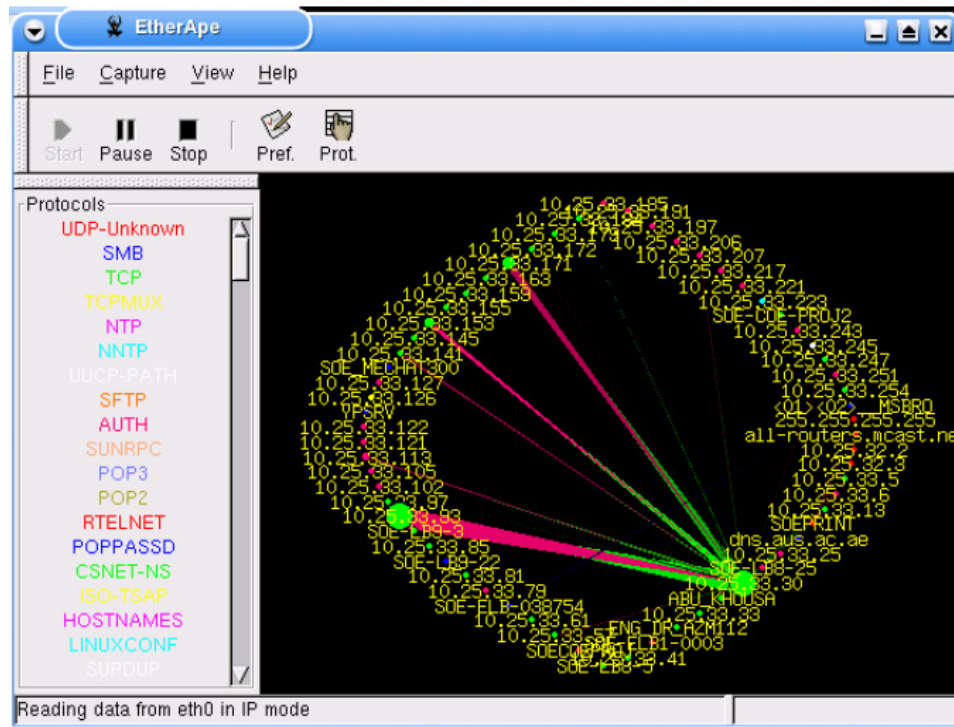


Figure4: EtherApe visualizes the difference between “normal-spoofe” scan and “Dynamic-Spoof” scan

We can see that the IP 10.25.33.93 is scanning a wide-range of ports on 10.25.33.30. An administrator can directly notice a huge traffic coming from 10.25.33.93 since “EtherApe” shows a **thick** line representing a high number of packets. On the other hand, we see there are single packets, represented by **thin lines**, being sent from different IP addresses towards 10.25.33.30. In reality, those lines are actually “fake” or “virtual” since those IP addresses are not sending anything. It is our DPS machine that is sending all those packets, but, with randomly and dynamically generated IP addresses.

Comments on Dynamic Port Scanning:

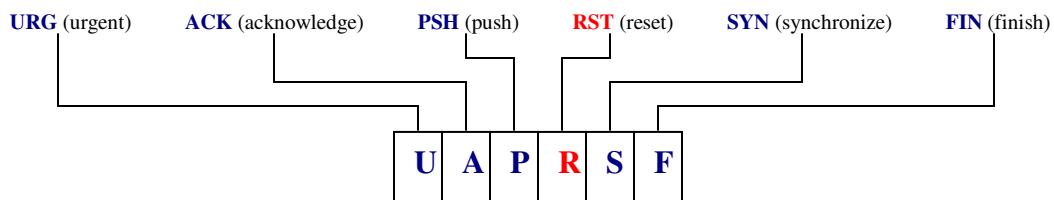
As one can see, DPS is very efficient in spoofing the scanning source IP while maintaining the receiving of return packets. There is no need for the real scanning IP even to show up once since all scan packets can carry “spoofed” IP. Also, there is no need in sending “unneded” packets to disturb the process of “tracing-back.” Furthermore, the spoofed IPs used could be completely off-line in the subnet. In other words, some of the IPs that will appear in the system logs of the scanned network will be “imaginary-existent”, which makes finding the attacker even harder since those IPs are actually never used in the local subnet of the scanning machine. However, the process of

DPS is limited only by the range of IPs within the local subnet; and this makes it a powerful and suitable tool for inside-attackers, that is, attackers within corporate networks, companies, universities, and so on.

1-packet-based TCP Stealth Scan Types

For the completion of this paper, we will describe the types of TCP scans which are based on 1-scan-packet. This of course excludes the TCP Vanilla Scan. Also, we will see those scan types in scanning both Linux and Windows based machine, and how the reply can differ between the two.

1-Packet TCP scan is based on the “control flags” of the TCP header, which are:



By combining different sets of those control flags, we can have different type. With the exception of **RST** flag, which is used to reset a connection, we can have up to 10 types [Note: only 5 of these 10 are implemented in “nmap”]. The following is brief description of those 10 types and their results on both Windows and Linux machines:

[1] TCP NULL Scan [_____]]

The results of sending this scan packet are as follows:

- Linux machine Open Port : no reply
- Linux machine Closed Port : reply contains **ACK** and **RST** bits
- Windows machine Open Port : reply contains **ACK** and **RST** bits
- Windows machine Closed Port : reply contains **ACK** and **RST** bits

[2] TCP FIN Scan [_____]]

The results of sending this scan packet are as follows:

- Linux machine Open Port : no reply
- Linux machine Closed Port : reply contains **ACK** and **RST** bits
- Windows machine Open Port : reply contains **ACK** and **RST** bits
- Windows machine Closed Port : reply contains **ACK** and **RST** bits

[3] TCP SYN Scan [_____]]

The results of sending this scan packet are as follows:

- Linux machine Open Port : reply contains **ACK** and **SYN** bits
- Linux machine Closed Port : reply contains **ACK** and **RST** bits
- Windows machine Open Port : reply contains **ACK** and **SYN** bits
- Windows machine Closed Port : reply contains **ACK** and **RST** bits

[4] *TCP PSH Scan* [**P**]

The results of sending this scan packet are as follows:

- Linux machine Open Port : no reply
- Linux machine Closed Port : reply contains **ACK** and **RST** bits
- Windows machine Open Port : reply contains **ACK** and **RST** bits
- Windows machine Closed Port : reply contains **ACK** and **RST** bits

[5] *TCP ACK Scan* [**A**]

The results of sending this scan packet are as follows:

- Linux machine Open Port : reply contains only **RST** bit
- Linux machine Closed Port : reply contains only **RST** bit
- Windows machine Open Port : reply contains only **RST** bit
- Windows machine Closed Port : reply contains only **RST** bit

[6] *TCP URG Scan* [**U**]

The results of sending this scan packet are as follows:

- Linux machine Open Port : no reply
- Linux machine Closed Port : reply contains **ACK** and **RST** bits
- Windows machine Open Port : reply contains **ACK** and **RST** bits
- Windows machine Closed Port : reply contains **ACK** and **RST** bits

[7] *TCP XMAS Scan* [**U** **P** **F**]

The results of sending this scan packet are as follows:

- Linux machine Open Port : no reply
- Linux machine Closed Port : reply contains **ACK** and **RST** bits
- Windows machine Open Port : reply contains **ACK** and **RST** bits
- Windows machine Closed Port : reply contains **ACK** and **RST** bits

[8] *TCP XMAS1 Scan* [**P** **F**]

The results of sending this scan packet are as follows:

- Linux machine Open Port : no reply
- Linux machine Closed Port : reply contains **ACK** and **RST** bits
- Windows machine Open Port : reply contains **ACK** and **RST** bits
- Windows machine Closed Port : reply contains **ACK** and **RST** bits

[9] *TCP XMAS2 Scan* [**U** **F**]

The results of sending this scan packet are as follows:

- Linux machine Open Port : no reply
- Linux machine Closed Port : reply contains **ACK** and **RST** bits
- Windows machine Open Port : reply contains **ACK** and **RST** bits
- Windows machine Closed Port : reply contains **ACK** and **RST** bits

[10] *TCP XMAS3 Scan* [**U** **P**]

The results of sending this scan packet are as follows:

- Linux machine Open Port : no reply
- Linux machine Closed Port : reply contains **ACK** and **RST** bits
- Windows machine Open Port : reply contains **ACK** and **RST** bits
- Windows machine Closed Port : reply contains **ACK** and **RST** bits

The Tool of Trade

A tool demonstrating the concept of Dynamic Port Scanning has been authored, that is, “dps v1.0”, available for download at [<http://www.securebits.org/tools/dps-1.0.tar.gz>] The tool is built on “*libnet*” packet creation library, and “*libpcap*” packet capture library.

Dynamic Port Scanning Prevention

Since DPS is heavily dependent on ARP-Poisoning, then, the best method to defeat it is through the protection against ARP-Poisoning. There is no out-of-the-box solution to prevent ARP-Poisoning; however, we believe that a combination of multiple measures can aid in prevention as well as detection of ARP-Poisoning. We will discuss three measures that a sys admin should deploy in his network:

[1] *The deployment of Port-Disabling feature on switches*

Recent switches come with “Port-Disabling” option in case of detecting any malicious activities on that port. Among those activities is the change of IP Address of the machine attached to that port. Since DPS requires that packets are sent with “fake” IP addresses, a switch can detect this behavior and disable the switch port immediately. The only way to bypass such measure is to increase time-gap between packets sent with different IP addresses. If the time-gap is long enough so that the switch cache is timed-out, it could lead to a situation where that attacker can still use DPS, but, it will take longer time.

[2] *Installing ARPWatch package on the server*

“arpwatch” is a software package that monitors MAC/IP pairs in the network and reports any suspicious behavior. It is always recommended that the sys admin installs it on different subnets to monitor MAC/IP pair changes on the network.

[3] *Configuring static ARP entries on the machines*

Static ARP entries can be the best measure to protect against ARP-Poisoning. However, it can be a nightmare. However, if the network is almost stable (i.e. changes of IPs and machines are minimal), the sys admin can maintain a small perl or shell script that runs once a day and probe the IP/MAC combination of live systems and add static entries for them on the servers, located on that subnet, as well as on the gateway [i.e. router]. Although DPS can use unallocated IPs in subnet, “arpwatch” should take care of reporting them in such case.

Conclusion

We have seen in this paper how integrating ARP-Poisoning with Port Scanning can lead to a stealth scan where the IP of the scanning machine does not show either in the logs of the scanned machine or in the logs of an IDS. Also, we have discussed the pros and cons of such technique and how to prevent and detect DPS.

Reference

- [1] *Building Open Source Network Security Tools*, by Mike Schiffman.
[http://www.amazon.com/gp/product/0471205443/qid=1047410022/sr=8-3/ref=sr_8_3/002-2750037-0800055?n=507846&s=books&v=glance]
- [2] *Libnet Packet Creation/Injection Platform*, by Mike Schiffman.
[<http://www.packetfactory.net/projects/libnet/>]
- [3] *Nmap Port Scanner tool*, by Fyodor.
[<http://www.insecure.org>]
- [4] *The Art of Scanning*, by Fyodor.
[Phrack Magazine - Volume 7, Issue 51 September 01, 1997 - article 11]
- [5] *libpcap: the packet capturing library*
[<http://www.tcpdump.org/>]
- [6] *arpwatch: the monitor program for keeping track of ethernet/ip address pairings.*
[<http://ee.lbl.gov/>]
- [7] *EtherApe: a graphical network monitor.*
[<http://etherape.sourceforge.net/>]