MONITORING THE ARP PROTOCOL ON LOCAL AREA NETWORKS

by David Fuselier
August 1, 2003
GIAC Security Essentials Certification (GSEC)
Practical Assignment: Version 1.4b (Option 1)

1. ABSTRACT

This practical assignment is a research paper on how to use the ARP protocol to monitor local area networks. The ARP protocol can be used to take an inventory of computers on the network, alert when new machines are connected, inspect ARP traffic for computers that are scanning the network, and if you record the ARP traffic, it can provide a tool to help clean up after worm attacks and provide a record of communications to explain other events.

The practical begins with a description of the ARP protocol, packet formats, and characteristics of the protocol. The next section contains a survey of different ARP monitoring programs that are available such as arpwatch[7] and tcpdump[8].

And then, almost like a hobby, I wanted to write an ARP monitor for Windows 2000[10]. There are not many tools available for Windows 2000 (w2k). I hoped to learn something about ARP and get some experience with it on the network. I started with windump[9], because it is so excellent. Then I wrote two vbs scripts, ARP discovery and ARP traffic. ARP discovery listens to the ARP requests and displays the computers it has not seen before. ARP traffic displays the source and destination addresses like a basic network analyzer. If the ARP traffic is logged to a file, it becomes a diagnostic tool for determining which computers are involved in security events.

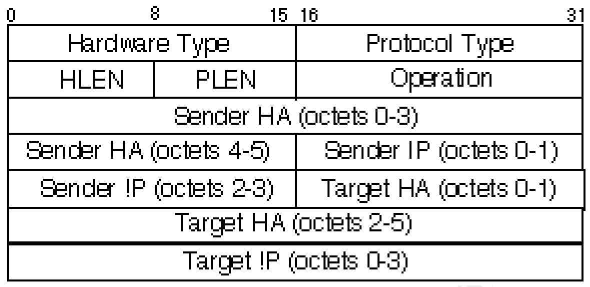
The last section talks about monitoring wireless networks. The same principles for monitoring wired networks apply to wireless if you can get windump to show ARP traffic on your wireless computer.

These tools by no means provide comprehensive network security. It would be possible to conceal a computer from the ARP monitoring by not broadcasting ARP packets.

2. A BRIEF DESCRIPTION OF THE ARP PROTOCOL

The ARP protocol is used to convert 32 bit IP addresses into 48 bit ethernet addresses. ARP works by broadcasting a request packet to all hosts on an ethernet. The request packet contains the IP address that the sender wants to communicate with. Most hosts ignore the packet. The target machine recognizes the IP address in the packet as its own, and returns a reply packet with the ethernet address[1]. The ARP protocol is described in RFC 826 and was originally designed for the DEC/Intel/Xerox 10Mbit ethernet.[2]

The format of an ARP packet is show in the following diagram. The credit for this diagram belongs to Fairhurst Gory, “Address Resolution Protocol (arp).” Jan 2001[4].



The following chart is a description of the ARP fields. The credit for this chart belongs to: Yegappan Lakshmanan. “ARP – Q&A.” 17-Feb-1999.[3]

```
+---------+--------------------------------------------------------+
|Ethernet |For a ARP request, source MAC address is the MAC        |
|Header   |address of the host sending the ARP request,            |
|         |destination MAC address is the Ethernet broadcast       |
|         |address (FF:FF:FF:FF:FF:FF), frame type field is 0x806.|
|         |For ARP reply, source MAC address is the MAC address of|
|         |the host replying to the ARP request, destination MAC   |
|         |address is the MAC address of the host that sent the    |
|         |ARP request, and the frame type field is 0x806.         |
+---------+--------------------------------------------------------+
|Hardware |Type of the hardware MAC address which is being mapped.|
|Address  |For Ethernet the value of this field is 1.              |
|Type     |                                                        |
+---------+--------------------------------------------------------+
|Protocol |Type of the protocol address to which the MAC address   |
|Address  |is mapped.  For IP address the value of this field is   |
|Type     |0x800.                                                  |
+---------+--------------------------------------------------------+
|Hardware |Size of the hardware MAC address.  For Ethernet, the    |
|Address  |value of this field is 6.                               |
|Size     |                                                        |
+---------+--------------------------------------------------------+
|Protocol |Size of the protocol address.  For IP, the value of     |
|Address  |this field is 4.                                        |
|Size     |                                                        |
+---------+--------------------------------------------------------+
|Operation|Type of operation being performed.  The value of this   |
|         |field can be 1 (ARP request), 2 (ARP reply)            |
+---------+--------------------------------------------------------+
|Source   |The hardware MAC address of the host sending the ARP    |
|MAC      |request or reply.  This is same as the source MAC       |
|address  |address present in the Ethernet header.                 |
+---------+--------------------------------------------------------+
|Source   |The IP address of the host sending the ARP request or   |
|IP       |reply.                                                  |
|address  |                                                        |
+---------+--------------------------------------------------------+
|Target   |The hardware MAC address of the host receiving the ARP |
|MAC      |request or reply.  This is same as the destination MAC |
|address  |address present in the Ethernet header.               |
+---------+-------------------------------------------------------+
|Target   |The IP address of the host receiving the ARP request   |
|IP       |or reply.                                              |
|address  |                                                       |
+---------+-------------------------------------------------------+
```

ARP will work across a bridge. Bridges will propagate the ARP broadcasts and bridge the replies.[1]

A router will block ARP packets.[1] This means an ARP monitor is required on each separate network segment to detect new ethernet addresses.

Network switches will pass the ARP traffic because it is broadcast traffic. This means that any computer on the network can see the ARP broadcast traffic. The reply packets are usually returned directly to the requesting computer, and network switches will block this unless it is addressed to your station.

3. A SURVEY OF ARP MONITORING PROGRAMS

The survey of ARP monitoring programs includes two excellent programs, arpwatch, and tcpdump.

Arpwatch is from Lawrence Berkeley National Laboratory.[5] Arpwatch runs on a lot of platforms: AIX, BSDI, DG-UX, FreeBSD, HP-UX, IRIX, Linux, NetBSD, OpenBSD, SCO, Solaris, SunOS, True64 UNIX, Ultrix, UNIX.[7] Whenever a new machine is connected to the network, arpwatch sends an email showing the ethernet address, and the vendor of the ethernet card. Here is what the email looks like:

```
     Date: Wed, 15 Oct 2003 13:57:17 -1000
     From: Arpwatch <administrators@university.edu>
     To: administrators@university.edu
     Subject: new station (station)

                 hostname: station
               ip address: 1.1.2.3
         ethernet address: 0:0:0:aa:aa:aa
          ethernet vendor: <unknown>
                timestamp: Wednesday, October 15, 2003
13:57:17
```

Arpwatch also writes to syslog which is shown here.

```
Oct 17 09:00:02 university.edu arpwatch: new station 0.0.0.0 0:0:0:aa:aa:aa
```

Arpwatch will also provide an alert if a station's ethernet address changes for any given IP address. An example follows. Most of the ethernet address changes will be expected. Replacing a computer will generate one of these alerts. This will also alert you to machines trying to boot up with the same IP as machines that are already running. I saw this one time in syslog. Another computer was trying to boot with the same IP address as the main file server. This really gets your attention. As it turns out, a computer was using bootp and was assigned the wrong IP address.

```
Date: Wed, 05 Nov 2003 08:45:38 -1000
From: Arpwatch <administrators@university.edu>
To: administrators@university.edu
Subject: changed ethernet address (host1)

            hostname: host1
          ip address: 10.20.30.40
    ethernet address: 0:0:0:a:b:c
     ethernet vendor: <unknown>
old ethernet address: 0:0:0:a:b:d
 old ethernet vendor: <unknown>
           timestamp: Wednesday, November 5, 2003 8:45:31
  previous timestamp: Monday, November 3, 2003 16:35:58
               delta: 1 Day
```

The next program is tcpdump. Tcpdump can show several different protocols, but the arp option will show the ARP traffic

This is an example of tcpdump on a Solaris 8 workstation. This example illustrates the ARP request/reply sequence. Pc4 is requesting the IP address for server1.

```
Solaris1 % sudo tcpdump arp
tcpdump: listening on hme0
22:17:08.941698 arp who-has server1 tell pc4
22:17:23.272951 arp who-has server1 tell pc42
```

This is another example from a linux machine. –n means show the ip address, -e shows the ethernet addresses, -tttt shows a date and time. Notice that the request packet has the destination ethernet address in it. Normally, the broadcast address would be there. It makes sense though if you want to refresh the cache using a directed transmit instead of a broadcast to cut down on ARP broadcast traffic. ARP request packets that are sent directly to another computer will not be seen by an ARP monitor running on a remote machine.

```
Linux1 % sudo tcpdump -netttt arp
tcpdump: listening on eth0
```

```
10/18/2003 08:21:27.760563 0:0:0:8:8:8 0:0:0:0:2:4 arp 42:
arp who-has 0.0.0.2 tell 0.0.0.0
10/18/2003 08:21:27.760737 0:0:0:0:2:4 0:0:0:8:8:8 arp 60:
arp reply 0.0.0.2 is-at 0:0:0:0:2:4
```

This is an ARP request packet with the broadcast address from a linux machine running tcpdump.

```
Linux1 % sudo tcpdump -netttt arp
tcpdump: listening on eth0
10/18/2003 08:41:49.848607 0:0:0:0:0:1a Broadcast arp 60:
arp who-has 0.0.0.1 (Broadcast) tell 0.0.0.15
```

4. ARP DISCOVERY

The ARP discovery program started out as a simple idea. I just wanted to be alerted when a new computer was connected to the network. This is the same way that arpwatch works. The ARP discovery program would have to inspect each ARP packet and if the ethernet address is not in the list, then add it to the list and report it as a new computer. If the computer is in the list, then ignore it. I ended up using vbs scripting, not because it was a particularly good choice, but because it was available on w2k and XP systems by default.

There are six steps to install and run the ARP discovery script.

1. Install WinPcap[11] by running setup.exe. Information concerning WinPcap can be found at http://winpcap.polito.it/
2. Create the directory c:\windump
3. Copy the file windump.exe[9] to c:\windump. Information concerning windump.exe can be found at http://winpcap.polito.it/
4. Copy the file http://standards.ieee.org/regauth/oui/oui.txt[6] to c:\windump. This file contains a list of the ethernet adapter vendors.
5. Create a file called arpdiscover.vbs in c:\windump. Arpdiscover.vbs is a text file that should contain the arpdiscover.vbs source listing in appendix A. You can use notepad and cut and paste the text from appendix A.
6. Open a command window, cd to c:\windump, and type cscript arpdiscover.vbs. The ARP monitor will start running. Press control-C to exit the ARP discovery program.

This is a listing of the files in c:\windump. The file arp.txt is created by the ARP discovery program and contains a list of unique computers on the network.

```
c:\windump>dir
 Volume in drive C has no label.
 Volume Serial Number is 907E-F78E
 Directory of c:\windump

10/18/2003  03:17 PM    <DIR>          .
10/18/2003  03:17 PM    <DIR>          ..
10/18/2003  02:41 PM             1,000 arp.txt
10/18/2003  02:39 PM             8,669 arpdiscover.vbs
10/15/2003  06:33 AM             2,784 arptraffic.vbs
10/17/2003  11:58 PM         1,122,009 oui.txt
08/04/2003  08:50 PM           397,312 WinDump.exe
```

The ARP discovery main screen looks like this:

```
c:\windump>cscript arpdiscovery.vbs
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

ARP Discovery
Read 0 addresses from arp.txt.
-------------------------------------------------------------------------------
Date  Time  MAC Address       IP address      Host             Vendor         R
-------------------------------------------------------------------------------
10/18 15:24 0:0:0:0:0:54      10.10.11.1      cpe-10-10-11-1   Cisco Systems
10/18 15:24 0:0:0:0:0:54      10.13.12.1      user-888bn88     Cisco Systems
10/18 15:24 0:0:0:0:0:54      10.12.14.1      cpe-10-12-14-1   Cisco Systems
10/18 15:24 0:0:0:0:0:54      10.11.15.193    aolclient-10-11- Cisco Systems
10/18 15:24 0:0:0:0:0:fa      10.10.10.29     cpe-10-10-10-29  CNet Technolog
10/18 15:24 0:0:0:0:0:8c      10.10.10.30     cpe-10-10-10-30  Cisco Systems *
```

You can get rid of the Windows Script Host header by using the command cscript/nologo arpdiscovery.vbs.

It is easy to create two shortcuts to run the ARP display and traffic programs. The commands are cscript/nologo arpdiscovery.vbs and cscript/nologo arptraffic.vbs. This is the unofficial icon for the shortcut.

The ARP discovery banner is next followed by the number of addresses read from arp.txt. If arp.txt is not found in c:\windump, the program will create a new file. In this example there are no entries in arp.txt. Each time you run ARP discovery, it will append any new computers to the list. If you want to rediscover the addresses on the network, just delete or rename arp.txt and it will start collecting addresses all over again.

The column headings are self explanatory except for the R column. The R column indicates whether the ARP packet is a request or a reply. The majority of ARP traffic is request packets that are broadcasted. If the R flag is blank, the packet is a request. If the R flag is a *, this indicates a reply. There are very few replies and most of them will be to your machine.

There are two options for the date and time stamps. Windump shows the date and time in GMT. The program can also generate a local date and time stamp. You can control this in the ARP discovery script. The local timestamp is not as accurate as the GMT timestamp, because the program generates the timestamp when the packet is written to the screen instead of when it was taken off the network by windump.

The MAC address refers to the source MAC address in the ARP packet. This is the computer sending the request or reply packet. The IP address is the source IP address in the ARP packet.

The host name is resolved by running nslookup and passing the source IP address. If this field is blank, then nslookup did not return a name. There is no host file lookup.

The ethernet vendor is obtained by taking the first three bytes of the source MAC address, and looking it up in the file oui.txt[6] and extracting the vendor's name. This is an example entry from oui.txt:

```
00-00-F8   (hex)            DIGITAL EQUIPMENT CORPORATION
0000F8     (base 16)        DIGITAL EQUIPMENT CORPORATION
                            LKG 1-2/A19
                            550 KING STREET LITTLETON  MA  01460-128
                            01460
```

Oui.txt is pretty big at 1096K. You can vastly decrease the size of the file and increase performance by removing the address lines. You only need the lines with the keyword (hex).

Notice the ethernet address has capital letters with leading zeros. Well windump has lowercase letters with no leading zero. So I had to convert the windump addresses before they could be used to search the oui.txt for the vendor. The code is pretty ugly.

The last line of ARP discovery shows the current ARP packet on the network. This line will overwrite itself with the current ARP packet, until a new computer is found. Then it writes out the new computer entry and the current ARP packet display line moves down one line.

In the above example, notice the ethernet addresses are all the same, but the IP addresses are different. This is a trace from a cable modem. It looks like an ARP bridge or proxy. Since the unique IP addresses represent new computers on the network, there is a switch in the program called showmultip. If showmultip = “on” then the program will display each duplicate ethernet addresses and the unique IP addresses. If showmultip is set to “off” then ARP discovery will show an ethernet address only once the first time it sees it.

If you let ARP discovery run, it will build a list of new computers in the file arp.txt. This is an easy way to take an inventory of the network. The longer you run discovery, the more comprehensive the inventory of computers.

After you have built a list of computers, if you restart ARP discovery the screen will only show new computers that are attached to the network. This is the alert feature. In its simplest form, the alert is a new entry on the console and an entry in arp.txt. The program could also send email or write to a syslog server. But for now, the program will just send a beep to the console. You can turn the beep off in the script by setting the bell variable to off which is the default.

There is one problem with the ARP discovery script. If windump is run continuously, the ARP discovery program will buffer the input and delay the output until several ARP packets are captured. If the ARP traffic is slow, you don't see any output initially. If windump is run repeatedly, one time for each packet, the program is lightning fast, but with heavy ARP traffic the script will lose packets. The default is to run windump continuously so no packets are lost. But the output will be delayed until a number of ARP records are read. I worked around this problem by running windump repeatedly until the first duplicate address is seen. After that the program runs windump continuously.

5. ARP TRAFFIC

While ARP discovery shows only the source computers in the ARP packet, the ARP traffic program shows both the source and destination IP addresses. By watching the ARP traffic you can get a good idea of which computers are talking to each other. Certain traffic patterns can alert you to a computer that is scanning the network. The traffic will also show misconfigured computers repeatedly sending ARP requests for computers that are not answering or computers that are sending ARP requests for their own address.

Tcpdump is the definitive ARP traffic monitor. One day at work, a Macintosh user told me that ntop was showing a lot of network traffic from an unrelated windows notebook computer. I ran tcpdump on the Macintosh and saw a lot of ARP requests from a third computer. This third computer was infected with the welchia worm. And the unrelated windows notebook also had welchia. The stupid welchia worm was trying to break into the Macintosh. It was easy to spot the infected computer with tcpdump because it was requesting blocks of sequential addresses.

I reworked the ARP discovery program and created a second script called ARP traffic. It is much simpler than ARP discovery. The program does not do any processing except display a subset of the windump output. It displays a list of source and destination addresses to make it easy to scan for patterns. I considered analyzing the output to look for slow scans, but did not get that far. This is the main screen for ARP traffic.

```
c:\windump>cscript arptraffic.vbs
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

ARP Traffic Monitor
-------------------------------------------------------------------------------
Date  Time  MAC Address       Source IP       Destination IP
-------------------------------------------------------------------------------
10/19 11:34 0:0:0:0:0:54      10.10.140.1     10.10.140.205
10/19 11:34 0:0:0:0:0:54      192.70.12.1     192.70.14.108
10/19 11:34 0:0:0:0:0:54      192.70.12.1     192.70.14.204
10/19 11:34 0:0:0:0:0:54      10.10.248.1     10.10.55.160
10/19 11:34 0:0:0:0:0:54      10.13.221.1     10.13.221.90
10/19 11:34 0:0:0:0:0:54      10.10.248.1     10.10.55.75
10/19 11:34 0:0:0:0:0:54      10.10.248.1     10.10.253.30
10/19 11:34 0:0:0:0:0:54      10.10.248.1     10.10.55.105
```

The example above is from a cable modem. Notice that 10.10.248.1 is talking to a lot of different computers. On your own network, you might notice two unlikely computers talking to each other. I can be surprising to see which computers are talking.

There is one problem with the ARP traffic script. If windump is run continuously, the ARP traffic program will buffer the input and delay the output until several ARP packets are captured. If the ARP traffic is slow, you don't see any output initially. If windump is run repeatedly, one time for each packet, the program is lightning fast but with heavy ARP traffic the program will lose packets. The default is to run windump continuously so no packets are lost. But the output will be delayed until a number of ARP records are read.

The ARP traffic program is suitable for recording ARP requests to keep a log of which machines are contacting each other. I use this as a tool to find computers infected with the blaster and welchia worms. The way this works is if you see an antivirus alert on a computer, you can check the ARP traffic log for entries to find out which computers were connected. If you know about the worm alert right away, issue a netstat –a to see which computers are connected. It is easy to record the ARP traffic output with the following command.

```
C:\windump> cscript arptraffic.vbs > filename.ext
```

6. MONITORING WIRELESS NETWORKS

I have an IBM Thinkpad T40 with a built-in wireless network card. The notebook has two stickers with different MAC addresses on the bottom. One sticker was for the ethernet adapter and one was for the wireless ethernet adapter. Monitoring ARP should work the same way for both wired and wireless networks

Since the T40 has both an ethernet and wireless network adapter it is necessary to tell windump which interface to use. It turns out windump has a very useful option, –D, that shows you a numbered list of network interfaces on the computer. First type windump –D to see the interfaces. Then type windump –ix where x = the network number, or you can use the far more tedious network name as shown in this example.

```
15CC14C7-8D35-4703-B528-C2D90FFB51FC
C:\windump>windump -D
1.\Device\NPF_{ } (Intel(R) PRO/Wireless LAN
2100 3B Mini PCI Adapter (Microsoft's Packet Scheduler) )
2.\Device\NPF_{51AE200D-E35F-5D29-C170-62C3672E8462} (Intel(R)
PRO/1000MT Mobile Connection (Microsoft's Packet Scheduler) )

C:\windump>windump -i \Device\NPF_{26DD25D8-9D46-5814-C639-
D2E0100C620D}
windump: listening on \Device\NPF_{26DD25D8-9D46-5814-C639-
D2E0100C620D}
```

Notice there is no output after the windump command. For some reason windump would not display the traffic. The WinPcap web site FAQ suggested it might be the wireless adapter: “Wireless adapters are not granted to work: some of them are not detected, other don't support promiscuous mode. In the best case, WinPcap is able to see an Ethernet emulation and not the real transiting packets.”[12]

Plan B was to install another wireless adapter and I installed an Orinoco silver wireless card and tried windump again. This time windump worked. Now there are network interfaces on this notebook, 2 wireless and one ethernet.

In order to get arpdiscovery.vbs to work I had to edit the arpdiscovery.vbs file to change the interface number in the windump command. The windump command in this case was windump –netttt –i3 arp. To switch between multiple interfaces, you have to edit arpdiscovery.vbs and arptraffic.vbs to specify the desired network number. The default network adapter is -i1.

Once the correct wireless network adapter was selected, arpdiscovery and arptraffic were working the same way they did on a wired network. The only problem was that the ARP traffic on our wireless network was very light and the arpdiscovery program output would lag behind windump.

So what happens if there is multiple access points in a building? Is there any way to tell which computers are connected to specific wireless access points? In our case, we have multiple access points connected to a single network switch. Since ARP is broadcasted, each access point will see the ARP request packets. There is no way to distinguish which access point is sending the ARP request. The ARP reply packets may or may not be seen depending on who they are addressed to. If the wireless access points are separated by a router or firewall with a different subnet, then you could monitor each network individually. A simpler way to monitor wireless access points is to use the wireless access point software logs to see what machines are connecting to a specific wireless access point.

7. APPENDIX A arpdiscovery.vbs

```
'----------------------------------------------------
'ARP Discovery
'Show new MAC addresses on a network
'by monitoring ARP traffic. Tested on w2k/xp.
'David Fuselier
'Oct 18, 2003
'----------------------------------------------------
'Constants
'----------------------------------------------------
Const ForReading = 1, ForWriting = 2, ForAppending = 8
Set objShell = WScript.CreateObject("WScript.Shell")
'Create memory cache for mac and ip address
nextcacheposition = 0
foundincache = 0
Dim ipcache ()
Redim Preserve ipcache(nextcacheposition)
Dim cache ()
Redim Preserve cache(nextcacheposition)
'
Dim objFSO, objTextFile
Dim windumprec
mac = ""
ip  = ""
cnt = 1
found = 0
host = Space(16)
vendor = Space(14)
Done = 0
bell = "off"
showmultip = "on"
'-----------------------------------------------------
'Check if arp.txt is there
'-----------------------------------------------------
Set objFSO = CreateObject("Scripting.FileSystemObject")
If objFSO.FileExists("arp.txt") Then
  Set objFolder = objFSO.GetFile("arp.txt")
'-----------------------------------------------------
' Read existing mac and ip from arp.txt into cache
'-----------------------------------------------------
  Set objFSO = CreateObject("Scripting.FileSystemObject")
  Set objTextFile = objFSO.OpenTextFile ("arp.txt", ForReading)
  Do Until objTextFile.AtEndOfStream
    strNextLine = objTextFile.Readline
    arrServiceList = Split(strNextLine , " ")
    cache(nextcacheposition) = arrServiceList(2)
    ipcache(nextcacheposition) = Trim(Mid(strNextLine,31,15))
'    ipcache(nextcacheposition) = arrServiceList(3)
    nextcacheposition = nextcacheposition + 1
    Redim Preserve cache(nextcacheposition)
    Redim Preserve ipcache(nextcacheposition)
  Loop
  objTextFile.Close
Else
'-----------------------------------------------------
' Create a new arp.txt
'-----------------------------------------------------
  Set objFSO = CreateObject("Scripting.FileSystemObject")
  Set objFile = objFSO.CreateTextFile("arp.txt", True)
  objFile.Close
End If
'-----------------------------------------------------
' Display Banner
'-----------------------------------------------------
Wscript.Echo "ARP Discovery"
Wscript.Echo "Read " & Ubound(cache) & " addresses from arp.txt."
Wscript.Echo "-----------------------------------------------------------------------------"
Wscript.Echo "Date  Time  MAC Address       IP address      Host             Vendor         R"
Wscript.Echo "-----------------------------------------------------------------------------"
'-----------------------------------------------------
'Read each line from windump
'-----------------------------------------------------
Do Until Done = 1
If found = 0 Then
  Set objExecObject = objShell.Exec("%comspec% /c windump -netttt –i1 -c1 arp")
Else
  Set objExecObject = objShell.Exec("%comspec% /c windump -netttt –i1 arp")
End If
Do Until objExecObject.StdOut.AtEndOfStream
    windumprec = objExecObject.StdOut.ReadLine()
'    Wscript.Echo windumprec
'------------------------------------------------------
' Extract date, time, mac and IP from windump record
'------------------------------------------------------
  If InStr(windumprec,"tell") Then
    dt  = txtarg (windumprec,1)
    dt  = Left(dt,Len(dt)-5)
    tm  = txtarg (windumprec,2)
    tm  = Left (tm, Instr(tm,".")-1)
    tm  = Left (tm,5) & " "
    mac = txtarg (windumprec,3)
    ip  = txtarg (windumprec,11)
    if ip = "tell" Then
      ip = txtarg (windumprec,12)
    End If
    reply = " "
  End If
  If InStr(windumprec,"reply") Then
    dt  = txtarg (windumprec,1)
    dt  = Left(dt,Len(dt)-5)
    tm  = txtarg (windumprec,2)
    tm  = Left (tm, Instr(tm,".")-1)
    tm  = Left (tm,5) & " "
    mac = txtarg (windumprec,11)
    ip  = txtarg (windumprec,9)
    reply = "*"
  End If
'-----------------------------------------------------
' The timestamp from windump is GMT
' This block will log a local timestamp.
'-----------------------------------------------------
    dt = Month(Now) & "/"
    If Len(dt) = 2 Then
      dt = "0" & dt
    End If
    If Len(Day(Now)) = 1 Then
      dt = dt & "0" & Day(Now)
    Else
      dt = dt & Day(Now)
    End If

    tm = Hour(Now) & ":"
    If Len(tm) = 2 Then
      tm = "0" & tm
    End If
    If Len(Minute(Now)) = 1 Then
      tm = tm & "0" & Minute(Now)
    Else
      tm = tm & Minute(Now)
    End If
'-----------------------------------------------------
' Write current MAC line (overwrites itself)
'-----------------------------------------------------
  WScript.StdOut.Write dt & " " & tm & " " & mac & Space(18-Len(mac)) & ip  & _
    Space(15-Len(ip)) & Space(33) & reply & VbCr
'  Note: any delay will slow down the response of the arpmonitor
'  WScript.sleep 20
'  Wscript.Echo dt & tm & mac & ip
'-----------------------------------------------------
' Check if ethernet address is in the cache
' Packet is logged if IP is different
'-----------------------------------------------------
  foundincache = 0
  For i = 0 to Ubound(cache)
    If mac = cache(i) Then
      foundincache = 1
    End If
  Next
' Check ip.
  If foundincache = 1 Then
   foundip = 0
   For i = 0 to Ubound(ipcache)
      If ip = ipcache(i) and mac = cache(i) Then
        foundip = 1
      End If
   Next
     If foundip = 0 Then
       If showmultip = "on" Then
         foundincache = 0
       End If
     End If
  End If

'-----------------------------------------------------
'Found in Cache
'-----------------------------------------------------
  If foundincache = 1 Then
    found = 1
    foundincache = 0
'    Wscript.Echo dt & tm & mac & ip & " F"
  ElseIf foundincache = 0 Then
'------------------------------------------------------
' Not in Cache Add entry to cache, increase cache size by 1
'------------------------------------------------------
    cache(nextcacheposition) = mac
    ipcache(nextcacheposition) = ip
    nextcacheposition = nextcacheposition + 1
    ReDim Preserve cache(nextcacheposition)
    ReDim Preserve ipcache(nextcacheposition)
'------------------------------------------------------
'Lookup the vendor in oui.txt
'------------------------------------------------------
vendor = Space(8)
ouimac = Ucase(macfmt(mac))
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objFile = objFSO.OpenTextFile("oui.txt", 1)
Do Until objFile.AtEndOfStream
    strLine = objFile.ReadLine
    If (Instr(strLine,"(hex)")) Then
      If Left(strLine,8) = ouimac Then
         vendor = Right(strLine,Len(strLine)-18)
         If Len(vendor) > 14 Then
           vendor = Left(vendor,14)
         ElseIf Len(vendor) < 8 Then
           vendor = vendor & Space(14-Len(vendor))
         End If
      End If
    End If
Loop
objFile.Close
'------------------------------------------------------
'Lookup Host
'------------------------------------------------------
  host = Space(16)
  Set nslookup = objShell.Exec("%comspec% /c nslookup" & " " & ip)
  Do Until nslookup.StdOut.AtEndOfStream
    nsrec = nslookup.StdOut.ReadLine()
'     WScript.Echo nsrec
    If InStr(nsrec, "Name:") Then
        host = Left(nsrec,InStr(nsrec, ".")-1)
        host = Right(host,Len(host)-9)
        If Len(host) > 16 Then
          host = Left(host,16)
        End If
        If Len(host) < 16 Then
          host = host + space(16-Len(host))
        End If
    End If
  Loop
'------------------------------------------------------
' Log entry to console
'------------------------------------------------------
    logentry = dt & " " & tm & " " & mac & Space(18-Len(mac)) & ip & Space (15-Len(ip)) & _
      Space(1) & host & Space(1) & vendor & " " & reply
    If bell = "on" Then
      Wscript.Echo logentry & Chr(7)
    Else
      Wscript.Echo logentry
    End If
'------------------------------------------------------
' Log entry to arp.txt
'------------------------------------------------------
    Set objFSO = CreateObject("Scripting.FileSystemObject")
    Set objTextFile = objFSO.OpenTextFile ("arp.txt", ForAppending, True)
    Set colServices = GetObject("winmgmts:").ExecQuery ("SELECT * FROM Win32_Service")
    objTextFile.WriteLine(logentry)
    objTextFile.Close
  End If
```

```
Loop
Loop
'----------------------------------------------------
' Functions
'----------------------------------------------------
Function txtarg (ByVal string, ByVal argnum)
  TestArray = Split(string , " ")
  txtarg = TestArray (argnum-1)
End Function
'----------------------------------------------------
'Format the windump mac address to match oui.txt format
'Have to add leading zeros for compare to work.
'----------------------------------------------------
Function macfmt (ByVal string)
  macarray = Split(string, ":")
  For cnt = 0 to 2
    If Len(macarray(cnt)) = 1 Then
      macarray(cnt) = "0" & macarray(cnt)
    End If
  Next
  macfmt = macarray(0) &  "-" & macarray(1) & "-" & macarray(2)
End Function
```

8. APPENDIX B arptraffic.vbs

```
'---------------------------------------------------
'ARP Monitor
'Show new MAC addresses on a network
'by monitoring ARP traffic. Tested on w2k/XP.
'David Fuselier
'Aug 6, 2003
'---------------------------------------------------
'Constants
'---------------------------------------------------
Done = 0
Set objShell = WScript.CreateObject("WScript.Shell")
Dim objFSO, objTextFile
Dim windumprec
Const ForReading = 1, ForWriting = 2, ForAppending = 8
mac = ""
ip  = ""
cnt = 1
found = 0
host = Space(22)
vendor = Space(8)

'----------------------------------------------------
' Display Banner
'----------------------------------------------------
Wscript.Echo "ARP Traffic Monitor "
Wscript.Echo "-------------------------------------------------------------------------------"
Wscript.Echo "Date  Time  MAC Address       Source IP       Destination IP                  "
Wscript.Echo "-------------------------------------------------------------------------------"
'----------------------------------------------------
'Read each line from windump
'----------------------------------------------------
Do Until Done = 1
  Set objExecObject = objShell.Exec("%comspec% /c windump -netttt -i1 arp")
  Do Until objExecObject.StdOut.AtEndOfStream
    windumprec = objExecObject.StdOut.ReadLine()
'    Wscript.Echo windumprec
'-----------------------------------------------------
```

```
' Extract date, time, mac and IP from windump record
'-----------------------------------------------------
  If InStr(windumprec,"tell") Then
    dt  = txtarg (windumprec,1)
    dt  = Left(dt,Len(dt)-5)
    tm  = txtarg (windumprec,2)
    tm  = Left (tm, Instr(tm,".")-1)
    tm  = Left (tm,5) & " "
    mac = txtarg (windumprec,3)
    dip = txtarg (windumprec,9)
    ip  = txtarg (windumprec,11)
    if ip = "tell" Then
      ip = txtarg (windumprec,12)
      dip = txtarg(windumprec,9)
    End If
  End If
  If InStr(windumprec,"reply") Then
    dt  = txtarg (windumprec,1)
    dt  = Left(dt,Len(dt)-5)
    tm  = txtarg (windumprec,2)
    tm  = Left (tm, Instr(tm,".")-1)
    tm  = Left (tm,5) & " "
    mac = txtarg (windumprec,11)
    ip  = txtarg (windumprec,9)
    dip = Space(16)
  End If
'----------------------------------------------------
' The timestamp from windump is GMT
' This block will show a local timestamp.
'----------------------------------------------------
    dt = Month(Now) & "/"
    If Len(dt) = 2 Then
      dt = "0" & dt
    End If
    If Len(Day(Now)) = 1 Then
      dt = dt & "0" & Day(Now)
    Else
      dt = dt & Day(Now)
    End If

    tm = Hour(Now) & ":"
    If Len(tm) = 2 Then
      tm = "0" & tm
    End If
    If Len(Minute(Now)) = 1 Then
      tm = tm & "0" & Minute(Now)
    Else
      tm = tm & Minute(Now)
    End If
'-----------------------------------------------------
' Log entry to console
'-----------------------------------------------------
    logentry = dt & " " & tm & " " & mac & Space(18-Len(mac)) & ip & Space (16-Len(ip)) & dip
    Wscript.Echo logentry
Loop
Loop

'----------------------------------------------------
' Functions
'----------------------------------------------------
Function txtarg (ByVal string, ByVal argnum)
  TestArray = Split(string , " ")
  txtarg = TestArray (argnum-1)
```

```
End Function
```

9. REFERENCES

[1] Brent Baccala. “Connected: an Internet Encyclopedia.” April 1997. URL: http://www.freesoft.org/CIE/Topics/61.htm
[2] Brent Baccala. “Connected: an Internet Encyclopedia.” April 1997. URL: http://www.freesoft.org/CIE/RFC/826/index.htm
[3] Yegappan Lakshmanan. “ARP – Q&A.” 17-Feb-1999. URL: http://www.geocities.com/SiliconValley/Vista/8672/network/arp.html
[4] Fairhurst Gory “Address Resolution Protocol (arp).” Jan 2001. URL: http://www.erg.abdn.ac.uk/users/gorry/course/inet-pages/arp.html
[5] Network Research Group, “LBNL’s Network Research Group.” September 2002. URL: http://www-nrg.ee.lbl.gov/
[6] IEEE Registration Authority, OUI.TXT, July 2003, URL: http://standards.ieee.org/regauth/oui/oui.txt
[7] Arpwatch, Lawrence Berkeley National Laboratory, URL: http://www.securityfocus.com/tools/142
[8] TCPdump, Van Jacobson, Craig Leres and Steven McCanne, all of the Lawrence Berkeley National Laboratory, University of California, Berkeley, CA. It is currently being maintained by tcpdump.org.
URL: http://www.tcpdump.org/
[9] Windump, Loris Degioanni, Fulvio Risso, Piero Viano, Navin Pai URL: http://windump.polito.it/misc/credits.htm
[10] Windows 2000, Microsoft Corporation, Redmond, WA URL: http://www.microsoft.com/windows2000/
[11] WinPCap, Loris Degioanni, Fulvio Risso, Piero Viano, Navin Pai URL: http://winpcap.polito.it/default.htm
[12] WinPcap, Loris Degioanni, Fulvio Risso, Piero Viano, Navin Pai URL: http://winpcap.polito.it/misc/faq.htm#Q-16