Chatter on the Wire:

A look at excessive network traffic and what it can mean to network security.

by Eric Kollmann
aka xnih
v.1.0
6 August 2005

Dedicated to

- My wife and children that I've been away from for the last year while serving in Iraq and will continue to be away from for the next few months
- To those who have helped me in my research on this and other products over the years
- To those who have written the programs mentioned and written the papers that a lot of this is based on
- To all those that continue to push security forward

Table of Contents

Introduction

The man that just came out of the building is Joe, he is an overworked, under paid and under appreciated employee at one of the nations largest IT companies. He has access to all the software the company is currently developing, and has hinted, to a yet unknown source that he may be willing to sell as many of the programs and other thing he can get his hands on, for a few bucks, of course. It isn't that he hates his company, or that he has any ill will towards them, he just wants to make some quick cash and is looking for something to do. In short he is bored and broke.

Stan, a man in a long trench coat, smoking a cigarette and standing in the shadows, is waiting for his victim to walk out of the building across the street. .He's been hired by an unknown source to follow Joe and make sure there are no police of FBI agents involved.

As the victim emerges from the building, Stan slowly stubs his cigarette out and meanders after him. He doesn't want to get to close and tip the guy off that he is following him…..

This is a cheap dime store Spy novel, correct?

Maybe it is, maybe it isn't, lets look at how someone might get information about your company, about your network or about products you are selling?

How does a spy find out information? There are a few main ways:

- Straight-out breaking into a building/computer/whatever and stealing the info
- Recruiting other people to do it or who already have inside access
- Listening in on conversations while people are sharing sensitive information

And a ton of others ways, but ultimately, planning, planning, planning. It most likely isn't a 5 minute deal.

This happens in everyday locations, around the world, on computer networks all the time. In the weekly SANS report there are articles about break-ins at Universities or Corporations in every report. Information about social security numbers or other personal data being stolen on a monthly, if not weekly, or even daily basis. In most cases information is not provided on exactly how the intrusion took place, only that it did. But it basically comes down to 2 things, either they actively went after the information, leaving clues behind as to their entrance and the deed, or they silent gather the information in the background until they had enough to get in and out without being noticed. Either way, the information is gone, but in one case you may not even know that it has happened, in the other there are probably easily noticed clues that may help investigators.

Since we are hearing about it, most likely this was an active attack or the intruder wasn't quite as quiet and didn't clean up quite as well as they thought they did. That or you really don't know what they have!

I know what you are probably thinking; this is yet another paper on Active and Passive network OS detection and Scanning Techniques OR this is a scare tactic to finally get management to listen. Well it is and it isn't, on both cases. It started, primarily as a paper on passive fingerprinting, that dips a bit into this and that along the way trying to give you a broad enough understanding of everything that has come before so that the new stuff makes sense. Without understanding how they are doing it, or what has happened in the past, parts of the new ideas or techniques will mean little to you. Perhaps in the end they will mean little to you anyway. I won't go into the specifics of all of the different types of active/passive OS detection techniques, but I will cover some of the major and unique ones. Not to mention ones I've written programs to attempt to do. And, by the way, if it helps frighten your management into finally doing something, let me know, and please send a few kickbacks to me on whatever you finally get them to buy!

Both Active and Passive fingerprinting have their own strengths and their own weaknesses. One leaves noticeable tracks in the sand and makes it much easier to track, while the other is much stealthier, leaving little to nothing behind and you may never know it has happened. Where one, provides instant results and the other takes time, days, weeks, months, or even years to collect everything you may need.

Nothing located in this paper is anything completely new and unseen before. It is built upon hours, weeks and months of other peoples work and their dedication to the art of determining what devices are located on our networks. It is work that has been done by system administrators, white hats, black hats, grey hats, and common every day end users. All who just want to understand more about what is out there on their network, how to better determine that and perhaps hide that. It is work that has been built upon by many individuals over the years, some which will be mentioned, others who will not.

Again, there is nothing secret in here, all of this work can be verified against your own networks. It can be used as a stepping stone for the next program. Most techniques have been in use for a few years at least, where others perhaps only a few months. I'll try to bring together as much of the different techniques in one central place, with links to the products, research papers, or articles that this is all based on. Some things I will expand on greatly because they interested me or were found to be lacking in the original research. Others will be glossed over because there is a ton of other papers and books about them already and I just want to point you in the right direction or I plain found them either boring or so technical that I just couldn't bother trying to understand them (I am a bit lazy at times).

Hopefully this is useful for the next generation of fingerprinting tools, if not, oh well, it helped pass the time here in the desert.

Section I - Active Scanning

I've personally always been interested in knowing what is out on the network and have used many active scanning utilities over the years to determine this. There are many products out there that are widely used and we will discuss some of them. There are also a lot of home grown apps out there that, either haven't received a lot of "press time" or just have been well hidden which we will also discuss.

Active scanners as a whole have the following traits:
Strengths:

- Immediate Results
- Highly Accurate (most cases where firewalls are not involved)

Weaknesses:

- Very noisy
- High bandwidth usage
- Easily traceable

Each active scanner, itself, has its own strengths and weaknesses beyond the generalized ones above. Some of them are getting much better at what they do as more and more people utilize them and provide the programmer(s) with feedback.

I think the feedback programmers have received and the growing security concern are some of the number one reasons these programs have grown so much in the past few years. Initially a programmer may come up with an idea, but if only a few people are providing them with feedback, signatures, etc, they have a very small world in which their program runs and signatures are generated from. With this small world the program has little chance to expand.

As computer security has grown, the use of these programs has also grown with it. With that growth, more people are using it, providing feedback and signatures and with that info the programs have grown much more accurate. With more accuracy, more people start to use and rely on them. It is a nice circle.

Enough about the field as a whole, onto some of the programs themselves…

NMAP[1], of course, is the one people and books go on and on about, it uses some nice techniques to determine the OS and is quite reliable most of the time. Almost any network security book will talk about NMAP and I believe there is actually a book or two dedicated exclusively on how to use NMAP. It uses TCP/IP fingerprinting of the IP stack using some of the following techniques:

- FIN probe
- BOGUS flag probe
- TCP ISN Sampling
- IPID sampling
- TCP Timestamp Don't Fragment bit
- TCP Initial Window
- ACK Value
- ICMP Error Message Quenching
- ICMP Message Quoting
- ICMP Error message echoing integrity
- Type of Service
- Fragmentation Handling
- TCP Options
- Exploit Chronology
- SYN Flood Resistance

If you want to know more about the above scan types you can read all about them at NMAP's Remote OS Detection Page[2].

Note: To do most of the tricks that NMAP uses, on a windows based machine, you need something like WinPCAP or RAW Sockets, since most of the tweaks that need done aren't easily accessible using winsock.dll. When I first started using RAW sockets, (prior to Windows XP SP2 which broke some of this ability), I attempted some of these features in a program I called VisionX, more on that program later on. Some of the features that have been implemented in NMAP are fairly easy to duplicate without knowing a ton about packet manipulation, others aren't, as I have found out over the years.

NMAP is a widely used program that has been ported to many platforms. The use of many of these techniques can be seen throughout the field of OS Fingerprinting. Some of these same tests will be seen again, not on the active side, but on the passive side.

If you want to know more about NMAP pick up any security book and there is probably a whole chapter set aside of it.

---

[1] NMAP is written by Fyodor and can be found at: http://www.insecure.org/**nmap**/

[2] More on NMAP's remote OS Detection can be found at: http://www.insecure.org/nmap/nmap-fingerprinting-article.html

Xprobe[3] is one I've read about and followed a lot of the research on how it was done, but it is one that I have never played with. I'm not a huge Windows fan by any means, but I've never been a huge *nix fan either. I'm a Novell Admin, or was, for years, dabbled in Linux, HP-UX and whatnot, but back to being lazy, I got tired of having to install this library to get productX to work, only to find out that to get libraryY to work this libraryZ needed installed, and on top of that this version of productX, didn't work with that version of libraryY. Why all the babbling, well last I checked, Xprobe only runs on *nix, so I haven't actually run it (again, I'm lazy sometimes).

Anyway, I found out about Ofir's page sometime in early 2001 when his paper 'ICMP Usage in Scanning v2.5' was out. Within a few months of printing the 200 odd page paper, he of course, released v3.0 of it and I got to turn around and kill another tree printing it. There are lots of trees out there, what was one more?

That paper has been invaluable! The information provided is all about using the ICMP protocol to fingerprint an OS. Just like every other RFC out there, every vendor has chosen their own interpretation of what certain things mean and implemented them slightly different. The ability to use these differences, in the way they respond, to ICMP traffic provides us with a way to uniquely identify them.

Xprobe uses the following tests:

- ICMP Echo (normal ping)
- TTL
- ICMP Echo (invalid code)
- ICMP Timestamp
- ICMP Address
- ICMP Info Request
- ICMP Port Unreachable

When most people hear the term 'ICMP traffic' they jump directly to ping packets. True ping packets are ICMP traffic, but they are not the only type of ICMP traffic. You have Echo requests (ping packets), Timestamp, address mask, info request and a number of others. As you can see above, the program sends 2 separate ICMP Echo Requests to the machine. The first one is a typical packet that you would expect to see on the network, nothing "special" about it. A typical ping packet looks like this:

```
⊟ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x485c (correct)
    Identifier: 0x0200
    Sequence number: 0x0300
    Data (32 bytes)
```

[3] XProbe is written by Ofir Arkin and Fyodor Yarochkin and can be found at: http://www.sys-security.com/

Notice how Code: 0? This is the default value in an ICMP ping packet.

The second ICMP Ping packet that was sent has the code value changed to a non-zero value. Different OS's will do different things with this. Some will echo back what was sent (like the RFC states), others, contrary to the RFC will set it to zero in the Reply packet. Examples of OS's that do it contrary to the RFC are Windows and Novell Netware.

One recent observation on my part is that, on windows at least, the ID field seems to always be 0x0200. Not sure what other OS's use by default, but ICMP traffic, like other protocols can be used to help, at least a bit, to differentiate between OS's. (Later test have shown 0x0100, 0x0300, and 0x0400, but more on this later in the paper about ICMP traffic where we will also look at some other default values and some possible changes or other identifying features of ICMP traffic that I don't recall in Ofir's paper).

Knowing this we can next look at the payload sent with a typical Windows client ping, which is: ABCDEFGHIJKLMNOPQRSTUVWABCDEFGHI
Unix on the other hand does not use alphabetic codes for most of it, but starts if off with a timestamp and then fills it with other info.

So by looking at ICMP Ping traffic we can differentiate between Windows and non Windows machines. Assuming of course they are using the built in ping utility. Anyone can write a utility that masquerades as another OS's. There is nothing that requires an OS to send a specific ICMP ping packet out. For that matter, any type of test, passive or active can be fooled by a tweaking of the responses that the device gives. That is just the nature of the game.

Next we will look at TTL, which most fingerprint utilities will use to some extent in their analysis of the remote system, this is not unique to XProbe. Each hop through a router decrements it by 1, so you have to have a general idea how many hops away the machine is for it to help. Most machines, on a local network will only be 1 or 2 hops away from you at the most. Even though it may only be 2 hops away on a LAN it may be 10-20 hops away from you if it is a machine on the internet. If you don't know you can normally just assume it falls into the 32, 64, 128, or 255 framework.

Some devices use "odd" TTL values such as 15, 60, 150, etc. In general this does two things:

- In the case where it is easy to determine what the base value was (i.e. your local LAN) this just adds to the uniqueness of the device and makes it that much easier to identify.
- It makes it that much easier to identify!

(More on Default TTL can be found in a later chapter all about TTL values).

ICMP Timestamp, Address, and Info request packets all help to differentiate between systems even more. Some builds of an OS may support Timestamp Requests/Replies, while others may not. The same goes for Address and Info request packets.

ICMP Port Unreachable packets add one more place to get info from. Different OS's, just like all of the above features, send different info back in the Port Unreachable packet.

Note: Almost everything that Xprobe does, packet wise, I've been able to implement. I actually wrote a program, prior to Xprobe's release that used as many of the features as possible from Ofir's paper. I actually only sent 4 packets in my program at the time. ICMP Echo, TimeStamp Request, Address Request, and Info Request. Just using those 4 packets, and the info it generated, a large part of our network could be mapped out to a base OS, or a class of OSes. In my program, due to a small computer base to test against, certain OSes were harder to differentiate between than others. Most types of *nix machines, at least to my program, all looked the same. One thing I did start to implement was the ability to masquerade as other OSes with the ICMP traffic that was sent out, but due to my inability, at the time to change all of the parameters in the packet that I wanted, this project fell to the way side and was later replaced by VisionX. The technology was rolled over into the new product, but wasn't expanded on much.

Xprobe was also the first program to use, as they describe it, "fuzzy logic". They give the OS a score based on how it matches up to a set of tests and gives you a probability of which OS it is. Instead of locking you into a static list, saying it must pass, all these tests to be this OS.

It will be interesting to see how certain hotfixes and SP's change their IP stacks, either on purpose to help hide their OS, or to fix broken functionality in them. I haven't had a chance to keep up with the latest MS hotfixes, but I know at least one of them, I believe MS05-019 had to do with ICMP issues. Will the fixing of that issue change the way the OS responds to ICMP packets. If it does, will it be in an easy enough way to determine it is still the same OS, but with or without that hotfix?

The reason this could be an issue, assuming it is simple to tell, either on this hotfix or others, is that most of the vulnerabilities anymore seem to be oriented towards "remote compromise". There have been many tools released over the past few years to detect if a specific hotfix is installed, such as MS03-026, or MS04-011, will these same types of tools be released for MS05-0xx patches that were released in April 2005? Or will it turn out that these attack vectors and not as easily used?

Note: Information about Xprobe is all with version 0.2. 0.2.3 has since been released in August 2005 which added some new scanning features. Also, Ofir has been busy with a new company call insightix, which can be found at www.insightix.com and their utility does a cross of passive and active scanning for network security. Still waiting for info to be sent to me about it beyond their default .pdf files. Not sure if it is going to ever make it to me or not.

GFI's Languard Network Security Scanner[4], is one that I've been using/following for years now. (Perhaps this is because I worked for them for a time being there, writing documentation, testing, etc?) This program is primarily used for network scanning by administrators for hotfix checking on MS networks these days, but it has a ton of other features in it and I believe it has a wider base in detection than some people may give it credit for! Most people only seem to be using it to push/scan for hotfixes though. As for OS detection with it, it uses some of the same ICMP tricks Xprobe uses, but not all of them. It also does banner grabbing, SNMP scans, SMB scans and others.

You can do simple scans of your network with null sessions, to see what anyone else could see if they were to scan you from the internet. Even though this type of scanning should gradually be on the decline, as more people move to XP in the windows world and implement the built in firewall, at least for home users, I believe it will still be with us for some time to come. This type of scan may still be useful, in some cases against corporations, at least scans from the "inside" since many networks have implemented GPO's to shutdown the default firewall, either to keep it from interfering with their own internal scans or to keep it from causing issues with home grown network apps. The Null Session scan is still useful on most networks, but due to its wide use over the years, hopefully, it isn't nearly as useful as it once was. I won't hold my breath though, I'm sure you can scan quite a few networks and return way too much info!

You can also specify a specific user (say Joe.Smith, with no special rights) or you can specify to use an administrative account. Either way, assuming you make a valid connection to the machine, LNSS will then do registry reads, file scans, etc, all depending on the profile used to determine what OS (exact OS unlike some other programs mentioned that can only get a general guess, LNSS will/can read the info straight from the Windows Registry assuming it has rights) and it will also check for what hotfixes are installed by verifying the file versions of specific files.

Again, depending on the profile being used it may also run scripts against your FTP, SMTP, Web, etc Servers and check for vulnerabilities there also. Not to mention gather user lists, group lists, etc. All of this through Windows API calls. Some of these things can be done through the Null session, others you must have appropriate rights to do.

Version 1.1 was the first one I played with, so I've been with it since its early days, watching it go to 2.0, 2.25, 3.0..3.3, betas of 4, and then into 5 where I sort of bowed out due to other commitments (like this one that has me in Iraq now). It has come a long way to say the least, and it was fun being in on the testing of it, all of those years! They are currently in version 6 which I have had little opportunity to play with except for a few minutes here or there. I assume version 7 will be out in the next few months, or at least be in a beta format before long.

[4] GFI's website is located at: http://www.gfi.com

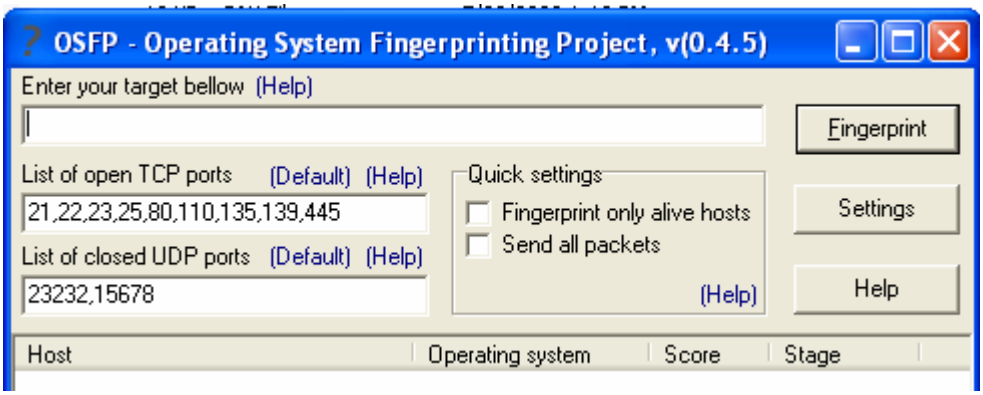
Note: Most of what LNSS could do through version 3.3 I have reproduced in my own program that I currently am using to scan our networks here. One shortfall, in my opinion, in LNSS is that it uses Microsoft's mssecure.xml file. This is a blessing and a curse. It saves them time, but also costs them time. In my opinion the file contains too much information for products that are not supported and is too big and cumbersome to work with. To add support for something that wasn't in there, originally, one has to modify the file in 3 places. I much prefer my approach to a specific file per product. That way, if files need updated, all the user has to do is download that one file that will range in size form 1-10K compared to one huge file that is 250K or more compressed (it is normally around 2 megabytes uncompressed). Some of that may have been taken care of in the latest versions though, I have been out of the loop for awhile now.

OSFP[5], was released for a very short time. Not sure how many people ever downloaded it, but I know I added signature files to it for awhile there. The last version I had was 0.4.5, at which time many cool features had been added. It also had the ability to identify the OS, in as little as one packet, normally within three. It used the following tests:

- TCP SYN request to an open port
- TCP ACK+PUSH request to an open port
- UDP request to a closed port

Perhaps these tests look a bit familiar? Sort of a cross between NMAP and Xprobe and some of the features that make them both very effective at what they do. It didn't stop at just using the best tests from each of the other products, it implemented the ability to take the info returned by the program, compare it to the "static" list it knew about, take the weight each answer had, and predict what the OS was in a similar way to Xprobe.

To do these tests with the least amount of packets you'd need to know an open TCP port and a closed UDP port. To help in this it has a set of default ports for both cases. By using this it, of course, increases the packets sent though:



One issue with OSFP was that it used RAW Sockets to send packets. Prior to XP SP2 this was not an issue, but with the release of SP2, the use of RAW sockets on the Windows platform is no longer a feasible option for what needs done.

I believe this is part of the reason that OSFP was short lived. Another reason was, in my opinion, it was a research product to see if it could be done back in late 2002 through the middle of 2003 (ouch has it been that long already?). For the product to continue it would need to be rewritten utilizing WinPCAP or something besides RAW sockets.

Though, through a little bit of "google-ing" I did find a copy of it out there on the lovely internet, got to love it, programs never cease to exist. Granted the copy I found had a CRC error in the .exe. Oh well, I still have the original, what do I need a copy from the internet for!

[5] OSFP was written by blad3, who has gone by other names over the years, the program which is no longer there used to be found at: http://www.blad3.ro

Note: I worked a bit on duplicating this program also, though I was having trouble with RAW sockets at the time. I could read the info returning just fine, but the ability to send the TCP packet in Delphi 6 was kicking my butt due to a PacketBuffer Byte vs. String issue. I fixed it late one night, got an updated version of the RAW socket files from the author sometime later, wasn't paying attention, and overwrote the version I'd fixed the issue in, and by the time I got around to playing with it again had forgotten all about how I fixed it. So that pretty much ended that project.

Soon after that SP2 came out, I found out TCP Packets in RAW mode were either no longer available or severely limited, so I moved onto other projects such as my network enumeration program for Windows that still used RAW sockets for the ICMP and SMB Scanning (UDP) part of it since they were not affected by the new SP. The use of RAW sockets greatly speed up the initial detection time for a class B or for the matter a class C network. With the ability to only wait 1 millisecond (or if I thought the card could keep up with inbound/outbound traffic I could set it to a 0 millisecond pause) between packets I can scan an entire network for both ICMP and SMB in mere seconds, compared to quite a few minutes. No waiting to see if they respond back or not, just send and forget, while listening for any packets directed back at me. If someone is talking back to my machine, evidently they are alive, so add them to the list. It has the possibility of adding a few false positives (people outside who I wanted to scan), but is worth the time saving features. I've since read of a way to fix that too, but haven't had the time since I've moved onto what prompted this whole paper which will be mentioned more later, but so you have a name, it is currently called Satori. .

VisionX, is a program I wrote a few years ago. It took bits and pieces of the above programs and rolled them all together into one program. It primarily did the following:

- ICMP OS Detection
- TCP Port Scanning
- UDP Port Scanning
- Banner Grabbing
- SNMP Scanning
- SNMP Brute Force Community Name Guesser

The ICMP OS Detection was actually written before Xprobe 0.1 was, or at least about the same time. I'm not sure I had rolled that into VisionX yet, but I had written my original ICMP detection kit at that point. It was all based off the paper mentioned before in the Xprobe section. VisionX did ok at identifying objects on the network, but the TCP/UDP scanning was wanting to say the least. I didn't know about raw sockets at the time, so it was very slow on the scanning process and it had some other major memory glitches in it that I never bothered to track down. But it gave me a place to start. Without some of that code I probably wouldn't be where I am on other projects.

Regardless of the simplicity or the duplication from other programs that may be out there each of my programs helped me to understand different concepts about programs that were on the market.

The main shortcoming on most of my programs, such as VisionX, besides bad programming skills on my part, was that they all added to the network traffic, providing people with clues that they were being scanned or probed. For those administrators who were checking logs I normally got an earful about my IP showing up in their logs yet again! Fortunately for me, this didn't happen that often, not me scanning the network, but administrators checking their logs on a regular basis. Sad, but true.

Active Scanner Conclusion is this…

These products all generate a lot of noise to determine what the remote device is. Some generate a lot more than others depending on what the end result you want is. If all you want is the OS this can be done in as little as 1-3 packet(s) in some cases assuming you know a little about the network. Information is gathered via TCP/UDP packets, responses on ICMP packets, SNMP OID info, banner grabbing, etc. But as mentioned, they are all quite noisy and point right back at who sent them. When using them you've put a big sign out that says "Here I am, I'm scanning the network!!!!" Even with the ability to spoof the source MAC/IP on some programs the noise generated on the wire is still there and because of the decoys it is greater. These decoy scans may get the "Hey I'm here" message away from you a bit, but they are still adding traffic to the net.

Active Scanners have the ability to pinpoint the remote devices OS better, in most cases, than Passive products, but at a cost of bandwidth and a lack of stealth since any IDS should be capable of being tweaked to determine exactly what type of scan is coming in. With this in mind, we need to look at the other part of this equation, passive scanning, but first…..

Section II - Default TTL

Somewhere in between active and passive OS fingerprinting you need to take a look at the default Time to Live (TTL) on an IP packet. Both active and passive utilities will utilize the TTL in their determination of the OS. One problem with this is that there are many utilities that will allow you to change the default TTL, so utilizing the TTL alone will not give you 100% reliability, but it may give you a starting point on what OS the remote device is running. Based on active scans with my ICMP project here is a list of devices and their TTL in returned packets (note that, 'returned' packets, not 'sent' packets, this is a major issue and will be discussed more later):

3Com Hub	255
AIX 3.2-4.3	255
APC Web/SNMP Management Card	60
Apple LaserWriter 8500	128
Asante Intrastack 6014 DSB	255
AXIS 2100 Network Camera	64
Axis 560 (5600?) PS	15
BSD	255
Canon Fiery Printer/PS	255
Cisco 1900	64
Cisco 1900C	255
Cisco 2800 Switch	15
Cisco 2900 Series Switch IOS 11.2 and 12.0	255
Cisco 7500 Router	255
Cisco Catalyst 5000 Series	60
Cisco Catalyst 6000 Series	60
Cisco IP Phone	64
CoBox for Recognition Systems	128
Compaq Tru64 v5.0	64
Dec Digital Alpha	64
Dell Power Connect 3024 Switch	255
DiscZerver	64
HP 700 X-Term	255
HP Ethertwist Hub Plus	32
HP J2603A AdvStack Hub	255
HP J4813A ProCurve Switch 2524	64
HP JetDirect Device Firmware <= X.8.32 (old devices)	60
HP Jetdirect Firmware L.2x.x Series (new devices)	64
HP ProCurve Switch 2424M	64
HP ProCurve Switch 2524	64
HP-UX 9000/300	255
HP-UX 9000/700	255
HP-UX 9000/800	255
HP-UX V10.20	255
HP-UX V11.0	255
Irix 6.2-6.5	64
Irix 6.5.3-6.5.8	255
Lexmark Optra S 1855 PS	255

LinkSys Print Server	32
Linux Kernal 2.0	64
Linux Kernel 2.2.x	255
Linux Kernel 2.4.x	255
Macintosh	255
Macintosh OS X	255
MetaSys - HVAC Stuff	126
Micro Annex MICRO-XL-UX	255
Novell Netware 5.1 or below	128
Novell Netware 6.0 or greater	128
OpenVMS V7.1-2	255
Optical Hand Scanner/Time Clock	126
OS 2	255
QMS MagiColor PS	15
Savin 2070 DP EB-70	255
Savin 2070 EB-70 PS	255
Savin 2535/2235 Network Printer	255
SmartChoice Register on Win98SE	128
Solaris 2.5.1-2.8	255
SunOS 4.1	255
SunOS 5.6	255
Ultrix 4.2-4.5	255
Ultrix 4.2-4.5	255
USR8000 broadband router	64
Web V Networks Web Cam	64
Windows 2000	128
Windows 2003	128
Windows 95 Original	32
Windows 95A or B w/winsock 2	128
Windows 98 or 98SE	128
Windows ME	128
Windows NT 4	128
Windows XP	128
Wireless Access Point	64
Wireless AP-1000 from Lucent	126
Xerox Document Centre (Photocopier)	255

For the most part the default TTL holds along the line of 32, 64, 128, 255, but in some cases odd IP stacks will implement something outside of the norm. If the machine is on your local network this can be a dead giveaway to its OS, or at least a greater help in determining what it is. When the machine is out on the internet “somewhere” you would have to turn around and do a traceroute to it, determine how many hops away it is, and based on that figure out the TTL.

If you see a TTL of 126, you’ll most likely assume the device is 2 hops away, but in reality it could be an AP-1000 from Lucent or a MetaSys HVAC system on your local subnet. Taking the IP address into account (and the subnet mask) along with the type of network when determining the TTL, from the source, is the key.

Most of the data above is a couple of years old now (or more), new firmware could have come out for the AP-1000 to fix this along with many of the others you see here. But as a general rule of thumb you can assume a TTL of:

- 255 – Some form of *nix or a network device (switch/router/hub).
- 128 – Windows (winsock 2 and above on 95 and the whole NT/2K family) or Novell Netware (at least through 6.0, haven't tested anything recently).
- 64 – Some form of linux, network device (switch/router/hub), or printer
- 60 – Some printer or network device (switch/router/hub)
- 32 – Something old if it is an OS (such as Win95 with the original winsock version), otherwise printer/hub
- 15 – Some type of printer

As you can see, just basing it off TTL is not going to get you real close to an OS, but it may break it down into smaller groups.

There is a good paper, though quite old now, located at http://secfr.nerim.net/docs/fingerprint/en/ttl_default.html which goes into some other information on Default TTL and what values are safe for the TTL and what aren't. It also takes a look at the TTL on packets depending on if they are TCP or UDP packets. This gets a bit away from what this chapter is about, but as we'll see later depending on the protocol and if you are sending the initial request or if you are sending a response your TTL may be different.

Section III - Passive OS Detection

Passive OS detection is going to be, by nature, completely quiet, therefore they are not going to leave any type of a fingerprint of their own on the network they are monitoring. There are products out there that claim to be able to detect if a machine is in promiscuous mode and I can't say if they work or not since I have never tested one, but this can be thwarted, in some cases by simply making a receive only cable. If all you want to do is monitor the network, and not interact with it, why would you want to be able to transmit onto it. Granted without the ability to transmit onto the network, you may be missing a vital group of packets that are needed to help identify the remote machine.

The problem with passive programs is that they only hear what is being sent to them or what is being multicast or broadcast out to everyone. Assuming you are the network engineer you can incorporate the passive scanner onto a switched port that is setup for monitoring, or if you are listening for "other means" you could do something like 'ARP Poisoning', but since we wanted to be passive (read quite!) this would defeat the purpose of putting a passive listener out there. The use of any IDS system should detect ARP Poisoning. For those not aware of what this is I'd recommend doing some research on it because it can be a serious security issue in certain circumstances and is beyond the scope of this paper.

p0f[6] is probably the first program most people think of with passive scanners. It is now in version 2.0. p0f uses the following 3 types of TCP packets to determine the remote OS:

- SYN
- SYN+ACK
- RST+

Each type of test has its own associated fingerprint file. Below is a list of tests that p0f uses:

- Window Size
- Overall Packet Size
- Initial TTL
- Don't Fragment Flag
- Maximum Segment Size
- Window Scaling
- Timestamp
- Selective ACK Permitted
- NOP Option
- Other/Unrecognized Options
- EOL Option
- Sequence of the Options
- Quirks (about 10 of them)

All of the above are read off of TCP Syn Packets. The Syn+Ack packets and RST+ packets have slightly different features and quirks they are looking at.

As mentioned before, in the Active Scanning section, each vendor has implemented the IP stack their own way, as they read the RFC's, it isn't enough to break the protocol, just enough to give someone with plenty of time on their hands, and access to tons of systems, to look at the differences long enough to be able to determine how to differentiate between them.

There have been many products that have either incorporated the fingerprint technology of p0f (version 1) or were built using the same general idea (some before, some after) such as: Ettercap, Siphon and others.

All of these programs basically listen for TCP packets with options and compare that to the static list they have. I haven't played a lot with these in a few years, but as I recall, unlike Xprobe and OSFP, they use a static list to determine what OS it is, which due to the strict type of test it is, with little "wiggle" room, this may be for the best. Many OS's appear to show the same general type of fingerprint, but not until you take it all together do you get an exact OS.

[6] p0f can be found: http://lcamtuf.coredump.cx/p0f.shtml

DHCP Listener[7] is a java program, which appears to be the proof of concept code they originally used in testing. It listens for DHCPREQUEST or DHCPDISCOVER packets and uses the options requested to fingerprint the remote OS/Device. They have incorporated a DHCP fingerprint module into their custom software that tracks machines at their university (DHCP Listener was the POC I believe to get the ball rolling). They provided a nice write-up about it for the February 2005 issue of Sys Admin Magazine. The article can be found at inisipd.com.[8]

Reading through their article and glancing at their code they are using the following main 2 things to identify the OS:

- Option 55 in the DHCP Request
- Option 60 in the DHCP Request

The DHCP Listener program has ~20 objects that it can identify with Option 55 (the Java program does not appear to use Option 60, but it is discussed in their paper).

Option 55 is the Parameter Request List from the client. It lists what features the client would like to know about on the network. You ask "How can we use this to identify the machine?" Well just like every other fingerprint feature so far, guess what, this protocol is no different. Well that isn't entirely true, since there is no specific rules on what a client should request or in what order. In this case, part of this, of course, has to do with the fact that different OS's are going to need different information. A windows machine will probably want to know where its Netbios servers are (options 44, 46 and 47), where a Mac OS X machine won't really care about those. So in this case, it isn't as much about vendors doing their own thing compared to what the RFC says, but vendors doing their own thing, because they need different info.

Option 60 is the Vendor Specific Code. Originally I thought only Windows implemented this. You'll see MSFT 5.0 and MSFT 98 in most Windows DHCPREQUEST packets and you don't see it in Mac OS X packets, but after some digging, I did found that other OS's and more importantly devices do use it. There aren't a lot that I've been able to find, but there have been a few. In the case of Windows 2000/XP and I assume 2003 we should find MSFT 5.0. In Windows 98 (SE only it appears) and ME we will see MSFT 98.

Note: Another feature that can be used, that DHCP Listener does not do, is just utilizing the Options that are requested as a whole. This may not provide you with an exact OS match (take for example the whole Windows family, which all have the same options as a whole they request: 53,61,12,60,55), but this gets us down to a family of OS's immediately. Also, this may be all that you get since a client does not have to request option 55.

---

[7] POC program can be found at: http://www.insipid.com/

[8] Sys Admin Magazine article about DHCP Fingerprinting: http://www.insipid.com/NGDHCP.pdf

Section IV - Where to go from here with Passive OS fingerprinting

The main problem with the p0f and other passive programs like p0f out there, is that they only look at TCP packets, and only packets with TCPDataOptions . We have a ton of other TCP packets out there that hold a ton of useful information. We also have another third of the IP protocol, that little thing known as UDP. Not to mention ICMP for yet another third of it. We saw a small portion of UDP being utilized in the DHCP Listener program, but even that didn't use everything, that was in that packet, to the fullest potential possible.

Ok, so now we need to look at more TCP and UDP packets, what else is there? What about 802.3 packets? I know everybody is going to IP these days, but there are a ton of programs out there that utilize IEEE 802.3 to send info, and they are just as chatty as the rest.

It is time to start looking at something new…. (maybe 20+ pages was a long intro just to get to this point in the paper that is all about chatty protocols and why this was written in the first place, but oh well, we are here now and there is good info above!)

I started looking through packet captures for unique packets on the network I'm on right now in Iraq. I long ago realized that people in general are way to talkative to people they have just met or know nothing about. This is always an issue in war time for OPSEC (Operational Security), it is just as big of an issue when you are in sensitive work areas in the private sector, and it is also an issue with computer networks. A quick packet capture, on any network will show you that machines are chatty and will provide someone "new" (a computer they know nothing about) info about their OS, name, IP, MAC, locally logged on user, etc.

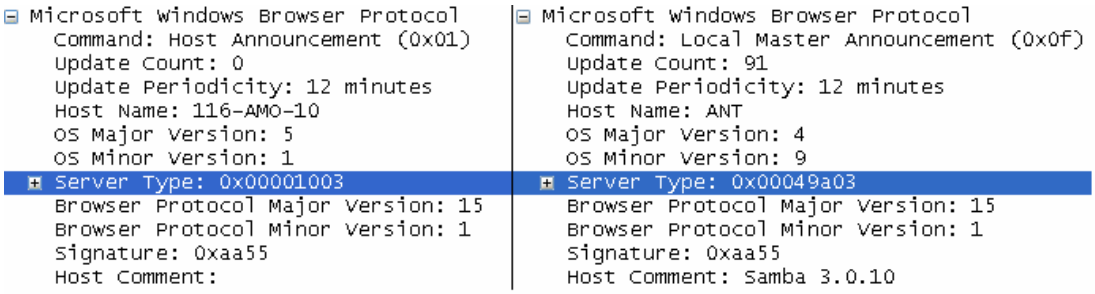
Each OS has its own chatty nature and protocols it use's, some have been cleaned up, a bit, over the years, others don't appear to have been at all, and yet other protocols are being added that are designed to make life easier, but have added even more noise to the line.

We are going to look at a few of the main OS's out there and how they broadcast out everything they know about themselves to anyone willing to listen. For the normal end user, on a secure network, this might be a good thing, but is it worth the risk?

Microsoft Machines

We'll start with Microsoft Machines and all the traffic they spew on the network. Fire up your favorite packet capture utility and you'll see plenty of these packets on a MS network. In my case I use Ethereal[9] and all the packet captures you've seen so far and will see from here on out will be from there, along with some of the packet names used.

Anyway, glancing at a packet capture of a MS network you will see a ton of SMB and BROWSER packets, each one, possibly, full of information about a machine on the network. The two packets below show two separate OS's. The one on the left is Windows XP, the one on the right is FreeBSD 4.11. Your thinking 4.11, why does it say 4.9 then? We'll talk about that more here in a minute. The one on the right also tells us a bit more, it provides us with info in the Host Comment section, telling us it is running Samba 3.0.10



Microsoft Windows Browser Packets, that are also Host Announcement packets, tell you the following:

- OS
- Services the machine offers
- And in some cases, the version of Samba running on it

In the case, on the left, it is a Windows XP Workstation (OS = 5.1). The machine isn't providing Print Services or Novell/Apple/SQL etc services as we can see from the Server Type (Expanding the Server type will show you 1 or 0 to a bunch of services, 1 means 'on' or available, 0 means 'off' or not a service it provides).

In the case, on the right, it is a FreeBSD 4.11 machine, that lovely one with 4.9 that is causing us a bit of confusion, well that is providing the following services: Workstation, Server, Print Server, Xenix Server, NT Workstation, NT Server, Master Browser.

OS Major/Minor versions that I've seen so far, for Windows, are:

- 5.0 – Windows 2000
- 5.1 – Windows XP
- 5.2 – Windows 2003

[9] Ethereal's homepage is: http://www.ethereal.com

- 4.0 – Windows NT 4.0

I don't know if BROWSER packets actually existed prior to NT 3.5, I'd have to think hard to remember when I started using specific protocols. It seems NetBEUI was around way back when, but how "way back" and "when" are we talking about. More things to look into one of these days.

The addition of new information, about FreeBSD originally had thrown a small kink into the project (that little issue of it being version 4.11 and not 4.9). Not a bad kink really since this is why the project was started, but I originally thought, what will a machine with version 5.0 of FreeBSD show (does that version currently exist, if not, what about when it is created), will it show 5.0 for the version info and if so, how do we incorporate this change into the program? Do we need to instead of focusing on the OS Major/Minor version, to start looking at the Server Type information along with the use of OS Version? Will this help provide more light on the subject? I think it will and may try to incorporate it into future builds. So time to start parsing the Server Type field also! The reason to also parse the server info field, is that for non windows machines, they normally advertise as Xenix Server. So even if we were to have 2 OS's (Windows and a *nix variant) with the same OS version, we would still be able to differentiate between them in these cases at least.

Anyway, with the addition of more info, from more testers, it appears that every machine we've seen so far with Samba installed on it shows either 4.5 or 4.9. So in the case of 4.5 and 4.9, we can't use this to determine what OS is on the machine, but we can use it to determine software installed on that machine (we know Samba is installed on it). If we happen to pick up a few other packets we may even find out what version of Samba (as we did above with the Host Description field) or with the LANManager field.

As you can see, by just listening on the wire we can get the base OS, but we can't tell SP level (for Windows) from Browser. Most everything, so far, that I've seen has 15.1 for the Browser Protocol, this is different on most pre-Windows2k OS's and may also be on the 9x series, or older versions of Samba, but so far I have not had the opportunity to get my legacy software out and start testing that theory.

Note: Some versions of NT4 will have a browser version of 15.1 and others 21.4. Originally the only machines I saw which advertised as 4.0 and 15.1 together were Novell Netware machines running Native File Access, but as always happens as you collect more intel is that you find that things overlap. So we've now seen some NT4 machines with 15.1 and others with 21.4, what causes the difference? Don't know, could be SP level or a hotfix, or perhaps it is server vs workstation. More on this is a few paragraphs.

As for FreeBSD we also see the same version in its packets (15.1), so this tends to make me lean towards the side that use of the Browser Protocol version, in fingerprinting, is not going to be useful, or is it?

Take a look at these 2 packets:

```
Microsoft Windows Browser Protocol
   Command: Local Master Announcement (0x0f)
   Update Count: 163
   Update Periodicity: 30 minutes
   Host Name: 8N1TH01
   OS Major Version: 4
   OS Minor Version: 0
 + Server Type: 0x00452003
   Browser Protocol Major Version: 21
   Browser Protocol Minor Version: 4
   Signature: 0xaa55
   Host Comment: PreferredCustomer
```

```
Microsoft Windows Browser Protocol
   Command: Host Announcement (0x01)
   Update Count: 3
   Update Periodicity: 12 minutes
   Host Name: BRN-3825F5
   OS Major Version: 1
   OS Minor Version: 51
 + Server Type: 0x00002003
   Browser Protocol Major Version: 11
   Browser Protocol Minor Version: 3
   Signature: 0xaa55
   Host Comment:
```

Browser Protocol of 21.4 = 4.0 for windows?
Browser Protocol of 11.3 = 3.11 for Windows?

OS version 1.51, and Browser Protocol of 11.3 An interesting issue with this device is that Server Type says this is a WFW Host, so far this is the only ones I've seen this in. All of them appear to be the same types of machines. Upon further checking we found these devices were actually print servers, brother print servers to be exact with their own built in SMB support.

And just when you thought you had it figured out, here comes a packet like this one:

```
Microsoft Windows Browser Protocol
   Command: Host Announcement (0x01)
   Update Count: 0
   Update Periodicity: 12 minutes
   Host Name: COAT-NW1_W
   OS Major Version: 4
   OS Minor Version: 0
 + Server Type: 0x00403003
   Browser Protocol Major Version: 15
   Browser Protocol Minor Version: 1
   Signature: 0xaa55
   Host Comment:
```

Well this is where it comes in useful to know your networks naming convention, even if you don't know all the systems. This machine is actually not a Windows NT 4.0 machine (notice the Browser Protocol of 15.1?) this is actually a Novell Netware machine, I assume running, Native File Access (NFA) for Netware which allows you to connect to a Netware machine without having the Novell 32 bit client installed.

So what do we learn from this? We can't take OS Major.Minor version, without also looking at Browser Protocol Major.Minor version! And as noted in the note: above even when we take both the OS Major.Minor and Browser Major.Minor we aren't guaranteed to get a perfect match. It may be that we need to utilize the Server Type at that point in time also.

Another interesting device is this one.

```
⊟ Microsoft Windows Browser Protocol
    Command: Host Announcement (0x01)
    Update Count: 0
    Update Periodicity: 1 minute
    Host Name: MARCHING BAND
    OS Major Version: 2
    OS Minor Version: 2
  ⊞ Server Type: 0x00000a03
    Browser Protocol Major Version: 15
    Browser Protocol Minor Version: 1
    Signature: 0xaa55
    Host Comment:
```

It claims it is a Print Server (in the Server Type field) and it very well may be. It is either a network printer or a Macintosh if memory serves me (the Marching Band part of the Music department is very big on using Macintosh computers). I'm pretty sure it is a printer, but can't recall specifics at this time, perhaps a printer/copier system?

We don't just need to look at Ethernet II SMB packets, we can also look at 802.3 SMB packets, such as the one of the left below:

```
⊞ IEEE 802.3 Ethernet
⊞ Logical-Link Control
⊞ NetBIOS
⊞ SMB (Server Message Block Protocol)
⊞ SMB MailSlot Protocol
⊟ Microsoft Windows Browser Protocol
    Command: Host Announcement (0x01)
    Update Count: 0
  ⊞ Server Type: 0x00000203
    OS Major Version: 1
    OS Minor Version: 0
    Update Periodicity: 1 minute
    Server Name: RNP7DF9C7
    Host Comment:
```

```
⊟ Microsoft Windows Browser Protocol
    Command: Host Announcement (0x01)
    Update Count: 0
    Update Periodicity: 12 minutes
    Host Name: RNP7DF9C7
    OS Major Version: 1
    OS Minor Version: 0
  ⊞ Server Type: 0x00000203
    Browser Protocol Major Version: 0
    Browser Protocol Minor Version: 0
    Signature: 0xaa55
    Host Comment:
```

The one on the right appears pretty much the same, but it is an IP packet and gives us a Browser Protocol version of 0.0 Both packets are from the same machine as you can see in the Server Name and Host Name respectively.

These appear to be Ricoh NIC's and are advertising themselves as Workstation, Server, and Printer Server Queues. I can't say for 100% certainty that they are print servers, but other ones that I have seen that advertise as 1.0 and 0.0 have been linksys switches with a USB PS built into them, so I assume that these are also.

Next area of study is on **SMB Logon Protocol** packets:

```
Microsoft Windows Logon Protocol (Old)
    Command: SAM LOGON request from client (0x12)
    Request Count: 0
    Unicode Computer Name: LS-ARMY-OBMC01
    User Name:
    Mailslot Name: \MAILSLOT\NET\GETDC828
  + Account control  = 0x0000
    Domain SID Size: 0
    NT Version: 11
    LMNT Token: 0xffff (Windows NT Networking)
    LM20 Token: 0xffff (LanMan 2.0 or higher)
```

The versions of 'NT Version' I've seen are

- 11
- 536870923
- 553648139 (Windows 2000 Server SP4, maybe others)

Do these tell us an actual version number? In at least one case it may, as noted above. Again, more research is needed into this part of the packet. What does NT Version really tell us here? Perhaps nothing, but perhaps, like the DHCP Options as a whole, it may at least give us a base OS Family such as the NT Kernel vs the 9x Kernel.

We do get the Computer name and a MailslotName which may come in useful later. The MailSlotName does provide us with a bit of info here. DC828 may indicate the name of a Domain Controller?

NBNS Packets

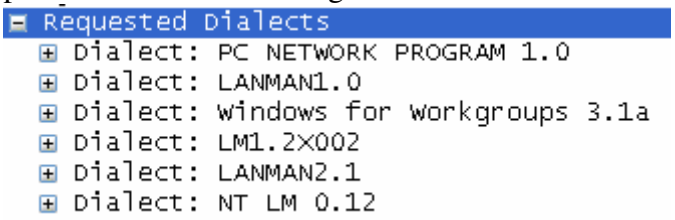
```
NetBIOS Name Service
    Transaction ID: 0x8cfe
  + Flags: 0x0110 (Name query)
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  - Queries
    - XWARE.CJB.NET<00>: type NB, class IN
        Name: XWARE.CJB.NET<00> (Workstation/Redirector)
        Type: NB
        Class: IN
```

With these packets we can see who on the network the machine is trying to communicate with or find out who is in what groups (workgroups/domains). These may just be looking for WORKGROUP, WPAD, etc. But this tells us who the machine is trying to talk to. In most cases this may not mean much, but other times this info could be useful. Take for instance the WPAD query. When I first saw this I promptly went out and looked for a workgroup called WPAD, when that didn't show up, I did a quick search on google (got to love google). WPAD queries are sent when Internet Explorer is configured to Auto Detect Proxies. So ok, we now find out every machine out there that is configured to AutoDetect Proxy settings.

Complete Sidebar: What would happen if we setup our rogue machine to answer these proxy requests? We setup a little relay program that answers the query, takes traffic on port XYZ and relays it all to wherever they wanted to go, in the process looking for any type of username/password that they may decide to enter. They have no idea that they are going through a proxy, no idea that basically a Man-in-the-Middle attack has happened and the best part about it, we didn't even have to flood a chunk of the network with ARP Poisoning or something such as that to accomplish this. Nor did we have to do anything to the primary DNS server to get them to route info to us instead of where they wanted to. All we had to do was gather info and answer that one little packet. (Granted most of this is speculation on my part, so far I've not read that it works this easy, but I don't see why this couldn't be done fairly simply).

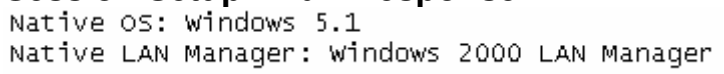
SMB packets

Any time you go to connect to another MS machine they do their little protocol dance and provide us with something like:

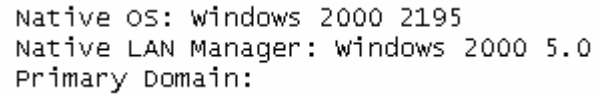


Can we fingerprint the OS by the dialects it supports? Unfortunetly, again I don't have enough research or big enough test bed to know. But with this we can find out what languages this machine speaks. Perhaps one of them has a known bug, or is easier to crack than another one. If so, when it goes to talk to you, if you have a program set to only allow that "weaker" dialect you may find it easier to compromise the system afterwards.

Session Setup AndX Response



Session Setup AndX Request



With both Session Setup and Request/Response packets we find out two main things

- Base OS
- LAN Manger version of the conversation going on between the two machines

We know we can use the information about the Base OS, here is a list of what I've found so far either on my own network or by searching the internet for packet captures:

- Windows Server 2003 3790
- Windows Server 2003 3790 Service Pack 1
- Windows 5.1

- Windows 2002 Service Pack 1 2600
- Windows 2002 2600 Service Pack 1 (what is different with these machines?)
- Windows 2002 Service Pack 2 2600
- Windows 2000 2195
- Windows NT 4.0
- Unix

And for LAN Manager:

- Windows Server 2003 5.2
- Windows 2000 LAN Manager
- Windows 2002 5.1
- Windows 2000 5.0
- NT LAN Manager 4.0
- Samba 2.0.0 for IRIX
- Samba 2.0.5a for IRIX
- Samba 2.2.7a-SuSE
- Samba 2.2.8a-SuSE
- Samba 3.0.4-SuSE
- Samba 3.0.2-7.FC1
- Samba 3.0.4-1.FC1
- Samba 3.0.6-2.FC1
- Samba 3.0.7-2.FC1
- Samba 3.0.10-1.fc3

And a ton of other Samba ones

Ok, what can we learn from that Session Setup and Request/Response? Originally, you probably thought not much, but as you can see, with Windows alone you have the possibility of learning the exact SP level of the machine. I say "possibility" because in some instances it doesn't always provide the exact version. So far this has not been the case with the Native OS field, but the Native LAN Manager field sometimes indicates a prior version (Windows XP will say Windows 2000 2195). Is this because of the handshake before, did the older OS not understand something, so we stepped down to a easier/older version? Again, unknown at this time.

For Unix machines, using samba, we may be able to determine the type of *nix it is, such as in the case of SuSE, Fedora, and IRIX, assuming they are running the precompiled versions of SAMBA for them. If they have taken the source and compiled it themselves all we will know is the version they are running. This may be enough to manipulate later, depending on how out of date the version is that they are running.

Also, depending on what packets are captured we can get the username being used and the encrypted password. There are many other programs out there that use/abuse this, I don't plan on bothering to even look at it in my captures, or at least I haven't bothered so far. If you are interested in that side of it you may want to look at Cain and Abel.

Mac OS X

Ok, we've had fun looking at MS machines, what about those lovely new Unix versions of the Mac OS?

Rendezvous

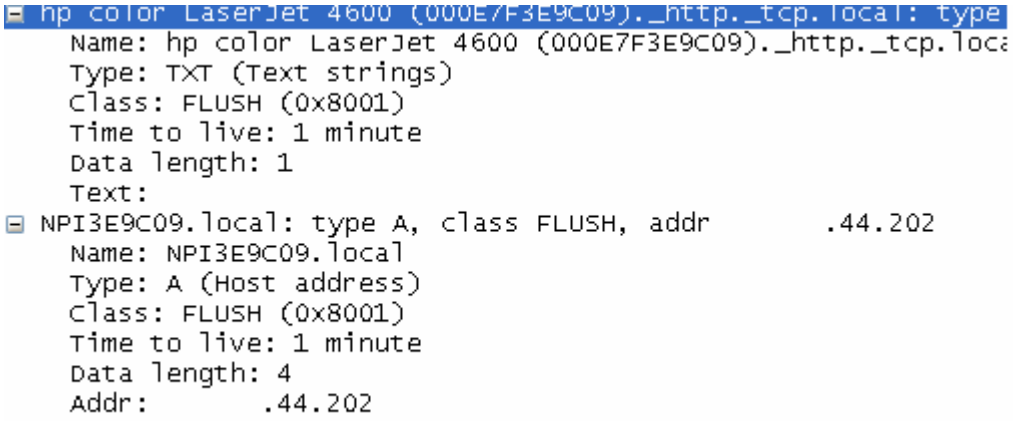
My favorite new product/protocol, that I just found out about, is Rendezvous (now known as: Bonjour) it sends out multicast packets on UPD 5353 that Ethereal calls MDNS packets. Basically, as you'll see, this protocol is more than happy to share everything it knows about you to the world. In a perfect network, where we know it is 100% secure, we know everyone who is on the network, and we trust them all, this isn't an issue, but unfortunately, we don't know who else is on the network, and we shouldn't trust everyone else on the network.

```
Ann-Marie-        :-Computer.local: type HINFO, class FLUSH, CPU PowerBook
  Name: Ann-Marie-        -Computer.local
  Type: HINFO (Host information)
  Class: FLUSH (0x8001)
  Time to live: 4 minutes
  Data length: 80
  CPU: PowerBook6,5
  OS: Mac OS X 10.3.5 (7P216), mDNSResponder-58.8 (Apr 24 2004 00:15:05)
```

Note: When I first looked at this protocol I missed a major little issue, all that MDNS is, at least to a point is DNS on another port that provides HINFO type packets. Not that it provides HINFO packets, but just that it provides them more often than DNS does.

I removed the young lady's last name here since she had done what most people do when they get a machine; they personalize it with their own name. So her machine constantly sits there and "talks" on the network telling everyone that it is here, it is a PowerBook running OS X 10.3.5. Is this too much info? Is it enough to make you worry? Well this isn't all that Ann-Marie's machine told us, but it is the most revealing.

Next we have this printer, it decided it want to provide us with a little info too:



Minor Sidebar: So now we have the IP address (first 2 octets removed), the name, NPIE9C09, the type of printer, HP Color LaserJet 4600, and its MAC address. Lets assume this was a printer with either a HD or memory that I could store files on, instead of just print to.

I've always wondered how vulnerable printers are to "attack" and/or abuse. Years ago I found a nice little "feature" on SNMP enabled printers, but more of that in the SNMP section to come.

So now that I know the printers IP and what type it is I could use/abuse it to do multiple things, perhaps only for a short period of time, if I'm storing stuff in volatile ram that will be reset when it is rebooted, but there are other places it may be stored and then utilized. A great little program to play with HP printers is HIJetter[10], but this is getting off topic. All of this information, again, is just being broadcast out.

Rendezvous, is available on the Microsoft platform also, it does not appear to be unique to Mac OS X, but that is where I first ran into it.

I'm sure OS X has plenty of other chatty protocols, but I haven't had a chance to look into it much. Take for example one talkative protocol, such as appletalk. Unfortunately, I'm not on a network that has many Mac's on it, so I can't look into it much at this time. But between Appletalk ARP packets that we can use to build a database of name to MAC's and ZIP (Zone Information Protocol) to get info about Appletalk zones we have plenty of things to look at in the Macintosh world too!

---

[10] HiJetter can be found at: http://www.phenoelit.de/hp/

Devices and other protocols

DHCP

Before we looked a bit at DHCP Listener and what it did, now we will go beyond that and look at everything we can find out via DHCP.

Here is a quick list I've noticed so far on my network:
Windows XP

DHCP Options:

- 53,61,12,60,55
- 53,61,50,12,81,60,55

DHCP Option 55:

- 1,15,3,6,44,46,47,31,33,249,43,252
- 1,15,3,6,44,46,47,31,33,249,43
- 1,15,3,6,44,46,47,31,33,249,43,252,12

Windows 2000

DHCP Options:

- 53,61,12,60,55

DHCP Option 55:

- 1,15,3,6,44,46,47,31,33,43

MacOS X

DHCP Options:

- 53,55,57,61,50,51

DHCP Option 55:

- 1,3,6,15,112,113,78,79,95,252

As you can see DHCP Options, 53,61,12,60,55 are what both Windows XP and 2000 request, but to differentiate them more we can go to the Option 55 and fingerprint it. (Actually, so far, every Windows OS tested has this one specific fingerprint). We probably could just use the DHCP Option 55 to ID them alone, but by broadening it we may find matches to products we didn't know about before. In many cases, as you can see with the Windows XP example above, there are multiple DHCP Option 55 request lists that a machine may send.

Looking at Windows XP, it sometimes requests 252 and 12 also, but other times it does not. The nice thing is the rest of the options are all the same. More research needs to be done to see if those extra options help to indicate a different SP level, or different services that may be running on the machine, but at this time there is not enough data to say one way or the other why the extra Options in 55 exist.

To do proper fingerprinting using this method, simply comparing the first X options may not work in all cases on Option 55. According to the fingerprint file from inisipid, Windows 2000 and ME have the same basic fingerprint, ME just requests one more feature in Option 55. Therefore, just like with p0f we need to do exact fingerprinting with this, we can't, at least easily, say that 9 out of 10 are the same, therefore it must be OS xyz. We may do this as a best guess, but there is no guarantee it is anywhere close. This could, of course, be due to a fairly small pool in the test pool used.

Note: I would have never even stumbled across DHCP fingerprinting had it not been for a packet capture I was looking at only 2 months after the article on DHCP Fingerprinting was published. I just happened to notice one DHCPREQUEST that was different than the rest, at the time I didn't have Option 55 expanded, all I could see was the Options as a whole the machines were requesting. That one packet was all it took, along with 20 seconds on google, to expand my passive program that much farther.

I was a little sad to see that there was already research done in this area since I was hoping to have finally found something "new" instead of just repeating what everyone else had done.

Now into the nitty gritty of DHCP, here are the main options that we've seen requested:

- 1 – Subnet Mask
- 3 – Router
- 12 – Hostname – provides the name of the machine
- 17 – Root Path
- 40 – NIS Domain
- 43 – Vendor Specific – depending on the vendor, different type of info may be sent in this field
- 50 – Address Request – What's the point of sending a request if we don't ask for an address?
- 51 – Address Time – How long of a lease time
- 53 – DHCP Message Type - 9 types, typical ones are: Discover, Offer, Request, Acknowledgement
- 54 – DHCP Server ID
- 55 – Parameter List – Sends the list of options the client wants to know, more about this later
- 57 – DHCP Max Message Size
- 60 – Class ID
- 61 – Client ID
- 77 – User Class
- 81 – Client FQDN
- 82 – Relay Agent Information
- 93 – Client System
- 94 – Client NDI
- 97 – UUID/GUID
- 116 – Auto Config

- 150 – Unknown
- 220 – Unknown
- 221 – Unknown
- 251 – Private

More can be found at: http://www.iana.org/assignments/bootp-dhcp-parameters

Some of these are requested by all clients, others are specific to each OS or device. So these options in themselves can give us a better idea of what device is out there even if we don't have an exact match. In other cases, as we've already seen the order in which they are requested is also a tell-tale sign.

DHCP Request, Options the Remote machine is requesting (all options are listed even if they were in multiple packets which may skew data):

	Windows 95	Windows 95a	Windows 95b	Windows 98	Windows 98 SE	Windows ME	Windows NT 3.5	Windwos NT 3.51	Windows NT 4.0	Windows 2000 Gold	Windows 2000 SP1	Windows 2000 SP2	Windows 2000 SP3	Windows 2000 SP4	Windows XP Gold	Windows XP SP1	Windows XP SP2	Windows 2003 Gold	Windows 2003 SP1	Etherboot 5.0	Etherboot 5.2	Mac OS ?	Mac OS X	empeg:mecury	Linux 2.4.8-pre4 i568	Linux 2.4.20-br20 ZEUS	Linux 2.4.21-202-default i686	Debain 3 unstable	PXEClient	IBMWARP_V4.1	Cisco IP Phone
1																								X							
3																								X							
12					X	X				X					X	X	X							X			X				X
17																								X							
40																								X							
43																						X									X
50																							X		X		X			X	X
51																							X	X	X	X	X	X			
53					X	X				X					X	X	X			X	X	X	X	X	X	X	X	X	X	X	X	X
54															X	X	X														
55					X	X				X					X	X	X			X	X	X	X		X	X	X	X	X	X	X	X
57																				X	X		X		X	X	X		X		
60					X	X				X					X	X	X			X	X	X		X	X	X	X		X	X	X	X
61					X	X				X					X	X	X						X	X	X	X	X			X	X
77										X					X	X	X													X	
81										X					X	X	X														
82																												X			
93																												X			
94																												X			
97																												X			
116															X	X	X														
150																					X										
220																						X									
221																						X									
251										X																					

Possible Promising Uses of DHCP that didn't pan out…

Now what happens, instead of just looking at the DHCP Request and Discover packets if we look at the Inform packets, is there anything different or more importantly is there anything useful… Yes there is useful info, in some cases, but is there anything specific that will help us in OS Fingerprinting, it doesn't appear so.

So far the best I can see that we can get from Inform packets is on a Novell Netware network where it lets you know about NDS specific info, but beyond knowing they have the Netware 32 Client installed we don't seem to find out much.

What about the cases when the DHCP server sends out a DHCP Offer for an IP address that is already in use on the network, this should generate a DHCP NAK packet from the host that is currently using that IP, will this info provide us with something useful?

```
⊟ Bootstrap Protocol
    Message type: Boot Reply (2)
    Hardware type: Ethernet
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x53f18687
    Seconds elapsed: 0
  ⊞ Bootp flags: 0x8000 (Broadcast)
    Client IP address: 0.0.0.0 (0.0.0.0)
    Your (client) IP address: 0.0.0.0 (0.0.0.0)
    Next server IP address: 0.0.0.0 (0.0.0.0)
    Relay agent IP address:        .144.1 (       .144.1)
    Client MAC address: 00:0e:35:bb:50:60 (       .171.214)
    Server host name not given
    Boot file name not given
    Magic cookie: (OK)
    Option 53: DHCP Message Type = DHCP NAK
    Option 54: Server Identifier =        .2.52
    End Option
    Padding
```

At least with this machine, no, we don't get any useful info back, or at least none that can be used for fingerprinting purposes, or can it? We would need more NAK packets to compare once we know the OS of the machine in question to know for sure. Perhaps other OS's will send more than just Option 53 and 54, but for that we'll need to wait until someone provides me with more NAK packets.

Cisco Discovery Protocol (CDP):

Now we are going to do a little jump, we are leaving the IP world, to an extent, and jumping over to IEEE 802.3. Here we find a very useful protocol CDP, the problem, it babbles just like everything else we have talked about. It wants to tell anyone and everyone all about everything that it has. It just wants to share info, is that so wrong?

```
⊞ Device ID:        _CORE
⊟ Addresses
    Type: Addresses (0x0002)
    Length: 17
    Number of addresses: 1
  ⊞ IP address:      .0.2
⊞ Port ID: FastEthernet0/17
⊞ Capabilities
⊟ Software Version
    Type: Software version (0x0005)
    Length: 226
    Software Version: Cisco Internetwork Operating System Software
                      IOS (tm) C3500XL Software (C3500XL-C3H2S-M), Version 12.0(5)WC9a
                      Copyright (c) 1986-2004 by cisco Systems, Inc.
                      Compiled Tue 13-Jan-04 11:26 by
⊞ Platform: cisco WS-C3524-XL
⊞ Protocol Hello: Cluster Management
⊞ VTP Management Domain:
⊞ Native VLAN: 1
⊞ Duplex: Full
```

Here you can get the IP addresses of the switches/routers, the Software version running on them, port info, duplexing and exact type of hardware. Normally you'd have to know the SNMP community name and IP address to query the OID to see what this is, but instead, with CDP turned on, it is out there for anyone to listen as the switches go upon the merry way to announce this to the world. With the number of Buffer Overflows in different IOS versions, I'm not sure I'd want to be advertising to the world what IOS version I'm on! But hey, that is just me.

Service Advertisement Packets (SAP)

Also in the IEEE 802.3 world we have SAP packets, they can help provide useful info about a device also:

```
Service Advertisement Protocol
  General Response
⊟ Server Name: 000E7FDCCE7C80DZNPIDCCE7C
    Server Type: Intel Netport 2 or HP JetDirect or HP Quicksilver (0x030C)
    Network: 00 00 0C 0E
    Node: 00:0e:7f:dc:ce:7c
    Socket: HP LaserJet/QuickSilver (0x400c)
    Intermediate Networks: 1
```

This is a bit generic, but it may be more than we currently had for this MAC. The one issue with this is that we have to somehow correlate this MAC with the IP address. Otherwise all we have is a pretty MAC and have no idea where on the network it resides.

SNMP

Jumping back into the realm of IP and running away from 802.3 for now, we find ourselves back in the IP realm, looking at UDP packets headed for port 161, better know as SNMP packets. To make SNMP work, all we need to know is one little thing, at least with SNMPv1 and I believe v2, haven't looked at v2 or v3 very much. Anyway, all we need is that community string, that simple thing gives us the keys to the kingdom, at least in some sense to that device. A quick listen on most networks will show:

```
Version: 1 (0)
Community: public
PDU type: GET (0)
Request Id: 0x0008c077
Error Status: NO ERROR (0)
Error Index: 0
Object identifier 1: 1.3.6.1.2.1.1.3.0 (SNMPv2-MIB::sysUpTime.0)
Value: NULL
Object identifier 2: 1.3.6.1.2.1.1.3.0 (SNMPv2-MIB::sysUpTime.0)
Value: NULL
```

This may not be a lot of information to start with, but since the Community name is passed in clear text, we have a starting point to abuse the switched network. According to most books, and based on general human nature, the default Community name of 'public' is never changed. This may just be the read community name, but guess what, the write community name, by default, and again, normally never changed, is 'private'. Now wasn't that hard?

Depending on the device we are talking about this may get the attacker nowhere. But certain HP printers did the following, for no apparent reason I was able to ever track down.

- Store their jetdirect password in a place that anyone with the community name could read in clear text.
- Store their jetdirect password in a place that anyone with the community name could read by simply converting it from HEX to ASCII

Slightly off track: I actually emailed HP about this 4-5 years ago or so when I first noticed it, got a quick response back that development had been notified and besides adding it to a query in LNSS that I published I pretty much forgot about it until it was brought up about 2 years ago in a notice on full disclosure or other mailing list. The author went into more details and explanations than I had when I first found it, but I found it funny that something that I'd seen/known about for so long was just then getting noticed in the full disclosure mailing lists (in other words, odds are this had been being abused for a long time!). Actually, it was mentioned again in an article about hacking printers in between those 2 events (me publishing it in LNSS and the full disclosure notice by someone else). I know I forwarded it onto him and he added a bit about it to his Defcon presentation[11] since I got recognition in his presentation

[11] http://packetstorm.linuxsecurity.com/defcon10/dc10-mattison/printers.pdf

paper (think this was post the main presentation and was for the upcoming year, been too long now).

Scanning a NT/2k/2k3 or Novell Netware Server that has SNMP enabled will give you users, loaded modules, services, etc. So this isn't harmless information by any means. If the service isn't being used, it should be turned off by default. Make the administrators go find it if they want to use it!

The point, developers have put information, available with a clear text "password" available via SNMP that probably shouldn't be. Hopefully they are cleaning this up as more and more things are being published to mailing lists, but, how many old printers with out of date firmware do you have in your organization?

Here is a list of SNMP Community names that I've seen floating around in packet captures that I've had access to:

- public (no surprise here)
- internal (HP JD Cards I believe)
- access

UPnP

This one falls a bit into the middle ground between active and passive. A device will broadcast out searching or notifying about services it offers. Here is a quick example:

Right now, all we know from this is that the device 192.168.0.50 is advertising the fact that it supports UPnP. To find out more about the device we would have to go to: http://192.168.0.50:5678/igd.xml The other bit of info we actually do posses here is that it is an Internet Gateway Device (igd.xml).

Where this crosses the line from passive, into active, or sits somewhere in between is when we actually go to the web address listed and retrieve that file to find out more about the device. In a normal network load/flow this shouldn't be an issue or stand out at all, since this device was actively telling us it was out there and this is how to find out more info.

Grabbing the igd.xml file from the device we see:

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
        <specVersion>
```

```
                <major>1</major>
                <minor>0</minor>
        </specVersion>
        <URLBase>http://192.168.0.50:5678</URLBase>
        <device>
                <deviceType>urn:schemas-upnp-org:device:InternetGatewayDevice:1</deviceType>
                <presentationURL>http://192.168.0.50:80</presentationURL>
                <friendlyName>NAT-Gateway</friendlyName>
                <manufacturer>U.S. Robotics Corporation</manufacturer>
                <manufacturerURL>http://www.usr.com</manufacturerURL>
                <modelDescription>Wireless Internet Gateway Device</modelDescription>
                <modelName>802.11g Wireless Router</modelName>
                <modelNumber>USR 8054</modelNumber>
    <UDN>uuid:upnp-InternetGatewayDevice-1_0-12345678900001</UDN>
    <UPC>123456789001</UPC>
```

So what do we know now? We have a Wireless AP that supports 802.11g that is from US Robotics and is model #8054.

If you want more info about UPnP a good starting source is www.upnp.org

HTTP Traffic

There is a good paper on some of this at:
http://net-square.com/httprint/httprint_paper.html
This article is more related to active fingerprinting, but the idea remains the same in what we can pull from a passive packet capture.

When you make a connection to a remote HTTPD you will normally see something like this within a packet capture:

```
⊟ Hypertext Transfer Protocol
  ⊞ HTTP/1.0 200 OK\r\n
    Date: Thu, 05 May 2005 05:40:35 GMT\r\n
    Server: Apache/1.3.26 (Unix) PHP/4.3.1\r\n
    X-Powered-By: PHP/4.3.1\r\n
    Content-Type: text/html\r\n
    Age: 1071\r\n
```

We can find out a few main things here:

- Server Type, in this case Apache 1.3.26
- X-Powered-By PHP 4.3.1

Another Unix machine:

```
⊞ HTTP/1.1 200 OK\r\n
  Date: Tue, 17 May 2005 20:40:56 GMT\r\n
  Server: Apache/1.3.28 (Unix) mod_ssl/2.8.15 OpenSSL/0.9.7c\r\n
  Pragma: no-cache\r\n
  Expires: Thu, 01 Jan 1970 00:00:00 GMT\r\n
  Cache-Control: must-revalidate\r\n
  Connection: Close\r\n
  Transfer-Encoding: chunked\r\n
```

Something a little new here:

- In this one we find out that it is running 2.8.15 SSL and Open SSL 0.9.7c

Or for Windows machines we have this:

```
Hypertext Transfer Protocol
  HTTP/1.0 200 OK\r\n
    Date: Thu, 05 May 2005 05:58:50 GMT\r\n
    Server: Microsoft-IIS/6.0\r\n
    P3P: CP='ALL IND DSP COR ADM CONo CUR CUS
    level: T10\r\n
    X-Powered-By: ASP.NET\r\n
    Content-Length: 1762\r\n
    Content-Type: text/html\r\n
    Cache-Control: private\r\n
    Age: 0\r\n
```

In this one we can determine the following:

- Server type – Microsoft IIS 6.0
- X-Powered-By – ASP.net

In either case, we can profile the system we are connecting to and determine a base OS that is most likely running on the server. This is not always the case or it may be a very vague idea of what OS, such as Unix, but in other cases we may not believe the info it provides, such as in this case:

```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    Cache-Control: private\r\n
    Content-Type: text/html\r\n
    Server: commodore64-HTTPD/1.1\r\n
    Date: Tue, 17 May 2005 20:37:33 GMT\r\n
    Connection: close\r\n
```

Now it could just be me, but I don't think they are really running an HTTPD on a Commodore64, but who knows these days, anything is possible!

Ok, so now we know about the remote systems we connect to, what about the local system that is connecting to it?

```
GET /CCMAdmin/phonelist.asp?findBy=pattern&match=contains&pattern=5763&rows=20&wildcards=on&utilityList= HTTP/1.1\r\n
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint, application/vnd.ms-excel, a
Referer: http://cmi/CCMAdmin/phoneconfig.asp?pkid={FD1F1B5D-1E54-458C-A89C-AF4A57FB0AB8}&status=uc\r\n
Accept-Language: en-us\r\n
Accept-Encoding: gzip, deflate\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; Avant Browser 7.0.0.7 [avantbrowser.com]; (R1 1.5))\r\n
Host: cmi\r\n
Connection: Keep-Alive\r\n
Cookie: phonelistLastQuery=findBy%3Dpattern%26match%3Dcontains%26pattern%3D5763%26rows%3D20%26wildcards%3Don%26utilityL
```

As you can see here the User-Agent is Mozilla 4.0, compatible with MSIE 6.0, Windows NT 5.1, Avant Browser 7.0.0.7

```
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7) Gecko/20040803 Firefox/0.9.3\r\n

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.1.4322; .NET CLR 2.0.50215; MSVS 8.0.50215)\r\n
```

```
User-Agent: Symantec LiveUpdate\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5)\r\n
```

As you can see from the above few there are many different types of User Agents that you may see on the network. Each one may provide a bit different fingerprint option. In the case of Symantec LiveUpdate , we don't know much more about the system besides the fact that it is running Symantec, either AV and/or Firewall software, and is doing an update at this point in time.

In the case of a few others, we get Windows NT 5.x, which would lead me to believe it is the OS that is running on the system in most cases, but not all! Since I'm pretty sure the first one listed with Avant Browser is actually a Cisco IP phone.

Depending on if we are looking at GET, POST, etc packets the information in the User-Agent field may be different. Most, if not all of the ones above are from GET packets.

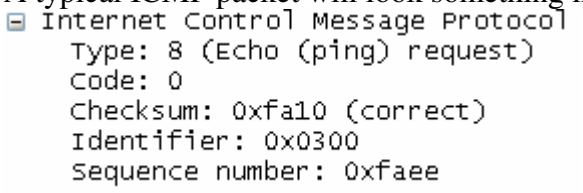
We can also see nice things like >NET CLR 1.1.4322, which means it has MS .NET 1.1 installed.

So again we see that we can use quite a bit of different types of traffic to get a better feel for what OS is running on a system, or we can see what products are running on a machine.

ICMP

Next we will start looking at ICMP traffic. Again, a common packet on the network, that at first glance you'd assume is useless in fingerprinting a device, but on the contrary, at least in certain OS's it is useful to distinguish one OS from another.

A typical ICMP packet will look something like this:



You will have a Type, Code, Checksum, Identifier, and a Sequence Number, along with Data, in most cases.

The Type field we will be looking at is 8 and 0, 8 being a request, 0 being a reply to that request. In most cases Code should be 0, but as we saw in the Xprobe section, by setting this to a nonzero in an active scan we can help differentiate between OS's, but that is off topic for here.

What we will look at from here is the Identifier field and the Sequence Number field.

Windows OS's will, when using the built in ping function, send packets out with an Identifier of 0x0200, 0x0300, or 0x0400. The machine appears to always send the same ID field, so this may be OS or SP dependent.

The data field for a Windows machine is 32 bytes in length and consists of ABCDEFGHIJKLMNOPQRSTUVWABCDEFGHI

The data field for a *nix machine has a timestamp value in there, some other stuff, and then 1234567

Common Identifiers are:

- 0x4757 Novell Netware
- 0x1b04 ???
- 0x0000 Linux Kernel 2.4x (replies only)
- 0x0100 Unknown (abcd for data)
- 0x0100 Windows (rare?, possible NT4?)
- 0x0200 Windows (most common)
- 0x0300 Windows (2nd most common)
- 0x0400 Windows (3rd most common)

Added to this we can look at the Sequence Number, with Windows machines we see the sequence increment by 0x0100 (256) on each ICMP packet, on other OS's we see other values.

In the case of Novell Netware, sometimes it will pick a Sequence number and stick with it through the current set of pings it does, other times it appears to increment by 0x0100. Not sure what the differences are here, but a nice area for more study. We can easily identify a Novell box though by its ID field of 0x4757

Other OS's, on the other hand, appear to have completely random ID fields.

One interesting thing to note is that some machines appear to do a completely random Identifier field, but then turn around and use 0x0000 for their sequence number.

Another thing to look at here is the TTL value on the IP packet. 32, 128, 255, etc…. The IP TTL here will depend on if it is a Response or a Request. The windows 9x family and up through NT4 will set the TTL to 32 in a request packet, where 2000+ will set it to 128. *The TTL value will depend on if it is a Request or a Response*. Other info on ICMP can be found in ICMP Scanning 3.0 paper by Ofir Arkin at the link previous mentioned in the Xprobe section of the paper.

Further Info: I hope to add more to this section later, but on an ending note, if there is data involved, at least in a Windows ping, the TTL is different than when it sends a ping packet with no data. So just because we have a packet that has a TTL of X, we may also need to verify the contents of the data that was sent with it before we utilize this information!

Section V - Conclusion

Bandwidth for most people is free, at least for all intent and purposes. Most people are not paying per packet that flows across the wire, so they aren't worried about a little extra traffic here, a little there, but as seen above some protocols provide a lot of info that either could help to compromise a network or at least helps to congest it. At the university, where I normally work full time, we went in and turned IPX and Appletalk off on almost every printer. We also changed the default time printers polled for new print jobs. In most cases we increased it by a factor of 10. So instead of polling for new print jobs every 6 seconds it was every 60. Most end users will never notice the difference, especially on a network printer since they have to walk to it, but you've just saved 9 packets from going out on the wire over that minute time frame that may have not been needed. You say, so what, what's nine packets…..

How does this tie back into what we've been talking about? It is all about what and how much we are broadcasting out onto the network all day long. Some of it may be needed for machines to know/understand what is going on and what resources are available, but a central server could just as easily provide that info. When you need to know where a printer is, request it, don't have it broadcast out all day that "I'm here, abuse me". We had WINS, we have SLP, etc. Are any of them the best, don't know. But if DHCP were to push the tag to tell you who to request/send info to and this gave you a "one stop" repository we wouldn't need BROWSER election packets, Dialect packets, etc floating around in every few packets we send. My machine, sitting here while I type this away, is busy chatting away with the domain controller, other machines on the network that are in workgroups wanting to become the next elected master browser, etc. All stuff that I would normally know nothing about, nor care, nor for that matter, should my computer care while I type this. The only time it should come to any type of importance is if I was about to print and didn't know where my printer was. Then again I'm of the opinion that the user should get off their lazy butt, walk down to the printer, get the IP, walk back and add it themselves the first time. But again, we've digressed.

It seems to me, instead of worry about jumping to 1 GB (or 10GB) network connections we should first look at what we can get off the network that is congesting it and making us need to look at greater speeds. To send my word document to the server in 0.2 secs compared to 2 seconds may be more of an issue with less protocols chatting on the network than spending thousands and thousands of dollars on infrastructure changes. By cutting down some of the chatter this in turn will make it harder for anyone listening to determine what is out there and possibly be exploitable.

Information floating across the network really is like a party line that some of you may still remember in your neighborhoods (before my time, and if you are reading this, most likely before yours, but there are still a few old timers out there that know what I'm talking about). The information is freely available on the wire for anyone with the expertise to install a packet capture program, parse it for the type of packet they need/want, and then turn around and have the gumption to use it to exploit you.

Or in the case of the party line, just pick up the phone and listen in. Not quite that simple, but close! In some cases you can't do anything about this, but in others you can by enforcing what protocols are allowed on the network and blocking those that aren't, at the router, so they at least don't propagate any farther than they have to.

I've repeatedly mentioned programs that I've written that imitate what other commercial products do, most of the time poor imitations, but what sparked this whole paper was a program, I was working on, that went farther than any of these other programs (hey I had to finally get out and write my own stuff one of these days, though I really am a crappy programmer). Almost everything that is mentioned in here for passive OS fingerprinting is incorporated into my latest program Satori. Stuff that isn't currently incorporated I plan to add to a future build if time permits. It listens in the background, captures packet after packet, looks for ones it likes (Browser, Session Setup and Response/Request, CDP, DHCP packets, SNMP, along with a poor mans version of p0f that I haven't had a chance to dig into very deeply), plus a few others that I haven't mentioned in the paper. Basically any packet on the network that may help identify either the OS/device or a product running on that device that I've had access to I've parsed and tried to incorporate. All of this just so that I could see what is on the network, which as I mentioned at the beginning is something I've always been interested in.

I've toyed with the idea of putting active OS fingerprinting options into it also. You find a machine you don't know what it is, right click, choose if you want to send ICMP tests, OSFP type tests, etc, sort of an all in one network detection utility. When I started writing this paper there was nothing on the market like that, but around June/July a company released a product that, to my knowledge, does just this. Again, just a little late on my part to getting it to the market!

A few months left over here, just don't know, maybe for the first time in a long time, I'll actually finish a large programming project and use it!

Hope you got something out of the babbling! Send comments, questions, or derogatory remarks to: eric.kollmann@us.army.mil or xnih@cableone.net or erickollmann@boisestate.edu The first address should be good, at least until I get back stateside. The other two… I don't check often, but I do check from time to time.

To find the latest version of this paper or to check out Satori, that uses almost everything mentioned in this paper and a few other tricks, try my website at:
http://myweb.cableone.net/xnih
Direct downloads:
Paper – http://myweb.cableone.net/xnih/download/os%20fingerprint.pdf
Satori – http://myweb.cableoen.net/xnih/download/satori.zip