

## Nmap from an **Ethical** Hacker's View Part 1



By **Kirby** **Tucker**

**Editor's Note:** **Kirby** is a long time contributor and supporter of EH-Net. So when he came to me with the idea to do a more approachable tutorial series on Nmap, it was a no brainer. Be sure to share your thoughts on this article and Nmap in general in the forum discussion.

What, another Nmap tutorial? Yes that's true, but I am hoping to approach it a little differently than what I have seen available. I want to describe Nmap from the viewpoint of a hacker and at the same time give a clear, step-by-step method of attaining a good level of proficiency. After completing this 2 Part Series and having practiced the techniques described, one should not only be able to sit at a "roundtable" discussion with advanced security professionals and "hold their own" in a discussion concerning Nmap, but also utilize this great tool in their own network.

[Discuss in Forums](#)

 [Digg This Story](#)

 [del.icio.us](#)

 [\(8\)](#)



### Introduction

So why learn Nmap? After all, you have been using expensive scanners like GFI LANguard, and it appears to give you a good view of your network. Well yes, there are a number of very good commercial scanners that do a reasonable job of showing you what's open and what's vulnerable on your network. However, the problems with commercial scanners is that they display everything inside the network from the admin's perspective and they usually only reveal flagrantly exposed ports with tcp connect type scans. Obviously this is not how an attacker from the outside will perceive your network. A skilled attacker might attempt different customized scanning techniques that might involve changing timing options or fragmenting his attack in order to slip through a firewall. These techniques are available in Nmap but usually not in other scanners, even when they incorporate Nmap. It's interesting to note that premium tools like Core Impact incorporate the free utility Nmap as one of their network discovery tools!

Nmap is without question the finest scanner available for the security professional and the scanner of choice for the vast majority of hackers. Logic dictates that you should view your network with the same tools as 99% of the attackers and see what they are going to see. If I had to secure a network, and I was given a choice of only using one tool - it

would be Nmap. Why is that? If I feel confident that, after all my best stealthy scans, I can't slip by my firewall or IDS, then I can feel reasonably secure about limiting a remote attack. The reality is, if the attacker can't connect to my network, he is dead in the water. So it really doesn't matter at that point if my software is exploitable or not. Think of Nmap as the security guard that goes around and checks all the doors on your house to make sure they are locked and locked tight.

## **The Basics**

Let's start with the basics. Nmap stands for Network Mapper. It was written by Gordon Lyon who is better known as Fyodor, a pseudonym he uses to honor one of his favorite authors, Fyodor Dostoyevsky. It's a free, open source tool available at <http://www.insecure.org/> with versions for Windows and Linux and is ubiquitous in its use. The latest version is tracked on a massive monitoring display at the National Security Agency at Fort Mead, Maryland. You might even see it used in movies such as The Matrix Reloaded, when the character Trinity used it to make a breach. It is so well known that the FBI subpoenaed Fyodor, so they could access information from his server logs of his download site. They were hoping to have a log showing certain individuals that have downloaded Nmap. It was recommended that privacy conscious users should use a proxy server to download the latest version.

As mentioned earlier, Nmap is available for both Windows and Linux, so the obvious question is which is the best OS for running Nmap? I am not anti-Windows, but the case for running Nmap on Linux has merit. Nmap's development is first completed on Linux and later ported to Windows. Fyodor has been very public about his lack of love for Windows, and much of the development for Windows has been done by other developers. As an example, the [GUI UMIT](#) that is without question the best front end for Nmap running in Windows. Don't waste much time on NmapFE. It hasn't been updated since 2002, and the latest features of Nmap are not accessible.

Microsoft can make sudden changes to their OS that can impair the working status of Nmap, and there are a few features not available in Windows. With the implementation of Windows XP Service Pack 2, Microsoft removed the ability to create TCP frames through raw sockets. To create a "workaround" for these SP2 raw socket problems, Nmap was modified to create raw Ethernet frames instead of raw TCP/IP frames. This fix allows most of the Nmap options to work properly, although Nmap's raw socket functions can now only create frames on Ethernet networks. Microsoft also implemented TCP/IP stack changes to Windows XP SP2 that limits the number of simultaneous outbound TCP connections, which in turn has a limiting effect on Nmap's TCP connect scan that tries to create a large number of connections. Having said that, Windows has developed into an effective platform for Nmap. Certainly it's easier to work in a Windows environment, if your network is completely Windows.

## **Scanning with Nmap**

Before we even think of scanning, we need a reasonable understanding of the four

protocols of the internet. IP, TCP, UDP and ICMP are the protocols that Nmap utilizes, and most everything that occurs in Nmap is based on these. Nmap performs four steps during normal scanning. If we have specified a hostname for the scan, it uses a DNS lookup to discover the IP address. The next step is to do an "Nmap ping" on the target. This is not the same as an ICMP echo request and is a bit more complex. Step three is a reverse DNS lookup which provides us a host name. Often a reverse DNS look up can provide additional information. Finally in step four, Nmap performs the scan we specified.

One important caveat about using Nmap (and not often discussed), is that by running it too aggressively, you could unintentionally DoS your own system. Yes, Nmap can be turned into an effective little DoS tool, so use it wisely. Just remember that the default setting won't hurt anything, so play with those first before going onto more complex scans. The default Nmap scan can make up to 1500 queries to a remote device, and there are 15 different scan methods built into Nmap. You can run it very loud or under the radar. You can send a single packet or thousands of packets. You really have a lot of control.

To really use Nmap effectively, you need to run it as an Administrator or from an account with Root privileges. If run as a non-privileged user, many of Nmap's best options will be unavailable. Also, I recommend running a packet sniffer like [Wireshark](#) as you begin to play with different scans. By analyzing packet captures during your scan, you will get a lot of insight into the inner working of TCP/IP and the scanning process as a whole.

The basic syntax of an Nmap command is straightforward:

**Nmap (command) (switch) IP of target**

**Example: Nmap -sS -v 192.168.1.105**

Each command can be customized by the optional switch we add. In Part 2 of this article we will delve deeper into the switch options, but for now we will focus on the default commands accompanied with the -v option. Verbose is most commonly defined as long-winded and excessively wordy. For this reason, the "verbose" switch is always a good option to include, especially when first learning our way around Nmap.

## Scan Types

These are the 15 core scans of Nmap and all are considered to be essential for the Certified **Ethical** Hacker (CEH) exam by EC-Council. Let's look briefly at each one and try to get a basic grasp of their purpose. The Internet is already filled with more than enough technical descriptions of each scan, so, instead of rehashing a thousand articles, let's focus on their practical aspects.

**And now for a little detail...**

## **TCP SYN Scan (-sS)**

**Example: nmap -sS -v 192.168.1.100**

The TCP SYN scan uses common port-identification methods without completing the TCP handshake process. When an open port is identified, the TCP handshake is reset before it can be completed. This is often referred to as "half open" scanning. If a scan type is not specified on the command line and Nmap is run from root or administrator, the TCP SYN scan is used by default. This scan works on all operating systems and is a little less noisy than a TCP connect scan. It is considered not as stealthy as it was in the past, because many IDSs are set up to detect SYN scanning. Unfortunately it's not a good choice for OS detection, because it can only provide information on open, closed or filtered ports. In the past you could just SYN scan an entire network and not be seen. Not true today because today's firewalls and intrusion detection systems can detect SYN scans. The trick is to alter the default SYN scan with options such as timing, fragmenting, etc. If done correctly, you can become virtually undetectable, but we will save that for Part 2. For now let's stay with the default settings.

## **TCP Connect Scan (-sT)**

**Example: nmap -sT -v 192.168.1.100**

Unlike the TCP SYN scan (-sS), the TCP Connect scan uses the normal TCP three-way handshake to determine if a port is available. It then uses the same TCP handshake connection that every other TCP-based application uses on the network. It is considered the most reliable scan, but the loudest. It is the least used by a knowledgeable hacker. If this kind of scan is picked up in the logs, more than likely it's someone new to hacking and is unaware of how noisy a TCP connect scan can be. However there could be times a hacker might use this scan from a hijacked computer. If this scan is run against most servers today, it will not only set off a warning but will also log the connection and the IP address. If the server were human, the log would read "Connect attempt by 'Script Kiddie'!"

## **The Xmas Tree Scan (-sX), FIN Scan (-sF), The Null Scan (-sN)**

**Example: nmap -sX -v 192.168.1.100**

**nmap -sF -v 192.168.1.100**

**nmap -sN -v 192.168.1.100**

These three scans are usually lumped together as "Stealth Scans" and really are more targeted towards Linux than Windows. On Windows all ports will appear to be closed regardless of their actual state. These are called "Stealth Scans," because they send a single frame to a TCP port without any TCP handshaking or additional packet transfers. The Xmas Tree Scan sends a TCP frame to a remote device with the URG, PUSH, and

FIN flags set. This is called a Xmas Tree Scan because of the alternating bits turned on and off in the flags byte (00101001) much like the lights of a Christmas tree. The FIN Scan's "Stealth" frames are sent to a device without first going through the normal TCP handshaking. If a TCP session isn't active, the session can't be formally closed! The Null Scan turns off all flags, creating a lack of TCP flags that should never occur in the real world. While its possible that the scans might not show up in certain application logs, they will be seen by even the most basic modern firewall including the rather simple firewall utilized in Windows XP SP2.

### **Ping Scan (-sP)**

**Example: nmap -sP -v 192.168.1.100**

This ping is a bit more complex than just a simple ICMP echo request. It first attempts an ICMP echo request packet. If no response is received, Nmap will then attempt a TCP ping. The ping scan is one of the quickest scans that Nmap performs, since no actual ports are queried. Unlike a port scan where thousands of packets are transferred between two stations, a ping scan requires only two frames. The ICMP echo request is very harmless and these frames are very common on most networks. Unless many IP addresses are chosen for a simultaneous scan, the ICMP echo request will probably not be noticeable. However, most administrators will filter all ICMP packets on their network ingress and egress points. A ping scan is also very fast, since there are only two frames required to complete it. This scan is useful for locating active devices or determining if ICMP is passing through a firewall. If this does occur, you can feel confident you can either make a breach of that network or it's a honey pot of some sort.

### **Version Detection (-sV)**

**Example: nmap -sV -v 192.168.1.100**

This scan can provide valuable information about the specific service running on an open port. For instance, it might inform you that the version of the http service running on port 80 is Apache 2.0.53 win 32. From there we can determine if we have the opportunity to exploit it or not.

### **UDP Scan (-sU)**

**Example: nmap -sU -v 192.168.1.100**

UDP has no need for "handshaking." With the UDP protocol, packets are sent and received without warning. Since there's no overhead of a TCP handshake, the UDP scan is inherently less "noisy." This scan relies on receiving an ICMP port unreachable message to determine if the port is closed. However, if ICMP is responding to each unavailable port, the number of total frames can exceed a TCP scan by about 30%. This scan operates very efficiently on Windows-based devices, because Microsoft-based operating systems do not usually implement any type of ICMP rate limiting. It's not really

a scan of choice for the hacker, because it can produce too many false positives when ICMP is blocked by the firewall. Where this scan might have value is if the system has already been compromised. For example, say your UDP scan reveals a Trojan running on a port that utilizes UDP. Most administrators block UDP leaking out of the network, but, if this is an internal scan of a network, you might find some exploitable services like SNMP, etc.

### **IP Protocol Scan (-sO)**

**Example: nmap -sO -v 192.168.1.100**

The IP protocol scan attempts to determine the IP protocols in use by the remote target such as ICMP, TCP, and UDP. It may help to determine routers, switches or printers on the network, if those devices have been configured with an additional IP address. This scan has the same limitations as the UDP Scan due to its reliance on the ICMP unreachable message.

### **ACK Scan (-sA)**

**Example: nmap -sA -v 192.168.1.100**

At face value this scan appears to be rather limiting. Because the ACK scan only provides a "filtered" or "unfiltered" response, it never connects to an application to confirm an "open" state. However it has value in determining firewall rulesets. It can do a reasonable job in discovering if a firewall is stateful or stateless.

### **Window Scan (-sW)**

**Example: nmap -sW -v 192.168.1.100**

The Window Scan is similar to an ACK Scan, but the Window Scan has the advantage of sometimes identifying open ports. The network traffic is kept to a minimum, and the scan itself looks relatively harmless. The Window Scan doesn't open a session, so there's no application log (unless there is a firewall or IDS in place).

### **RPC Scan (-sR)**

**Example: nmap -v -sR 192.168.1.100**

This scan displays detailed RPC application and version information if the remote device is running an RPC-based application, regardless of whether an RPC application is running on an unexpected port.

### **List Scan (-sL)**

### **Example: nmap -sL -v 192.168.1.100**

The List Scan simply lists the IP addresses that would normally be actively scanned. The idea here is to print a list of IPs without pinging or scanning the host. For large network audits, this process could be the difference between an efficient scan and one with hours of wasted time. Therefore, it has value as a pre-scan troubleshooter.

### **Idle Scan (-sI)**

#### **Example: nmap -sI -v 192.168.1.105 192.168.1.100 -P0**

The Idle Scan is used to gather information using another station on the network, and it will appear that the scanning process is coming from this third-party rather than the attacker. The advantage of this scan is the possible stealth factor. By stealth I don't mean flying under the radar and not being logged, but by hiding the identity of the attacker. A destination station will never see the IP address of the attacker directly. If the zombie used as the go between doesn't have any logging application, this can be reasonably untraceable. The key to this scan is in the choice of the "zombie host," and it should provide consistent and predictable IP identification (IPID) values and should have low traffic. Also, as long as the attacker and the zombie are both outside the network, then it is less likely that spoofed packets are going to be blocked by a firewall or IDS. However it is important to remember that an Idle Scan only locates ports. It can't provide any application version information or operating system information. Good targets are printers, routers and Windows computers as they make good zombie targets.

### **FTP Bounce Attack -b**

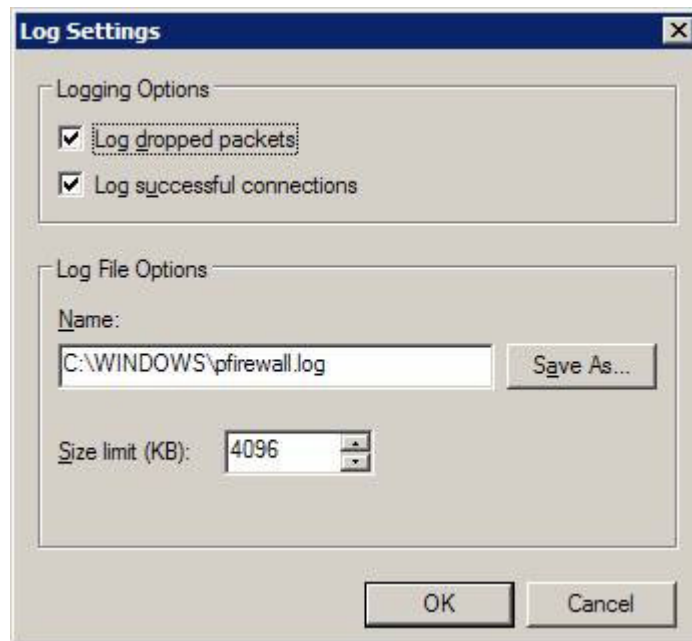
#### **Example: nmap -v -b anonymous: [anopass@192.168.1.8](mailto:anopass@192.168.1.8) 192.168.1.100 -P0**

This attack is a bit dated and most likely won't work with modern FTP servers. The FTP bounce attack wouldn't be possible if it weren't for passive mode FTP being available. The idea was to connect to a FTP server and from there data could be sent anywhere. It was great for its time and is a good example of how you can be creative with your scanning.

## **Running Scans**

Ok we have gone through the basic scans and now have some knowledge of what they do. The next step is to actually run these scans and get a feel for their output. A good place to start is to scan a Windows box on your home network. Turn off any firewalls and begin your scans. Make sure you are running Wireshark as you scan to help give meaning to what's going on. After you have run your scans with the firewall off, run them again with the XP firewall on and make sure logging is enabled. The idea here is even if XP's firewall can detect your scan, then certainly an enterprise firewall will! This is a great place to spend some time reviewing your work.

Turn on logging by going to the Advance Tab on Windows Firewall and then click on Settings for the security logging:



Here is an example of a basic -sT scan:

```
C:\Program Files\Nmap>nmap -sT 192.168.1.101
```

Starting Nmap 4.21ALPHA4 ( <http://insecure.org> ) at 2007-08-19 14:57 US Mountain Standard Time

Interesting ports on 192.168.1.101:

Not shown: 1701 filtered ports

PORT	STATE	SERVICE
------	-------	---------

135/tcp	open	msrpc
---------	------	-------

139/tcp	open	netbios-ssn
---------	------	-------------

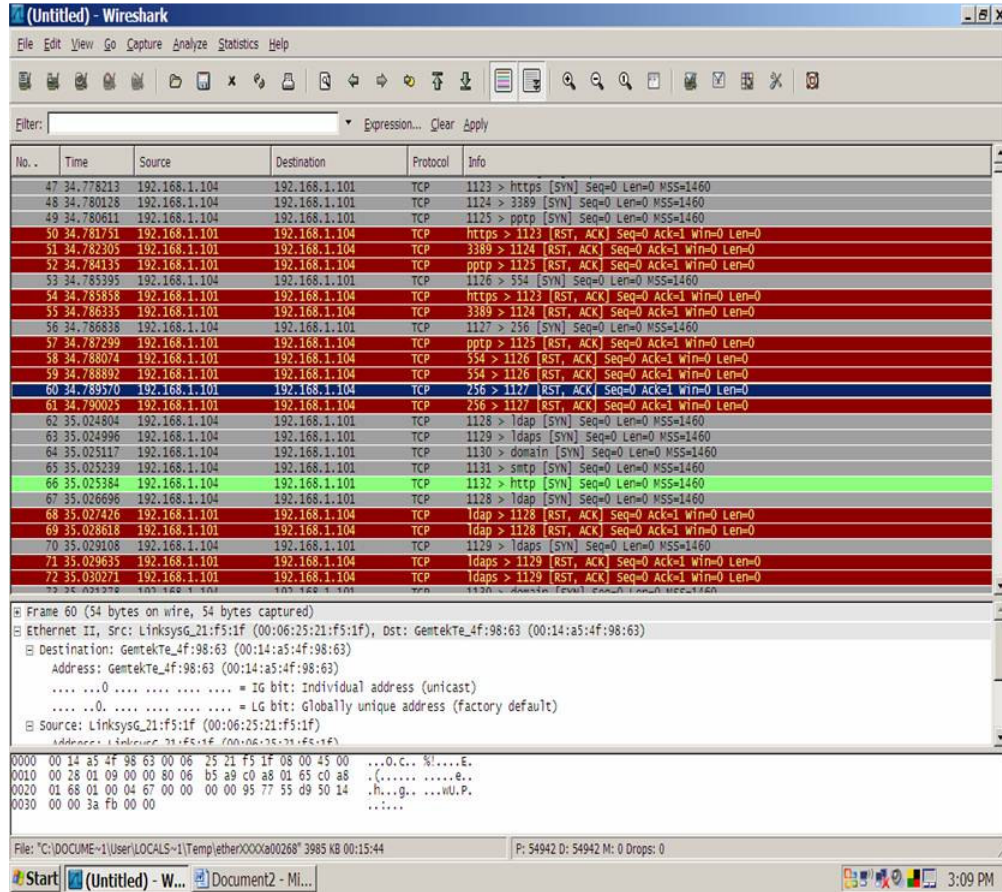
2105/tcp	open	eklogin
----------	------	---------

MAC Address: 00:06:25:21:F5:1F (The Linksys Group)

Nmap finished: 1 IP address (1 host up) scanned in 190.469 second

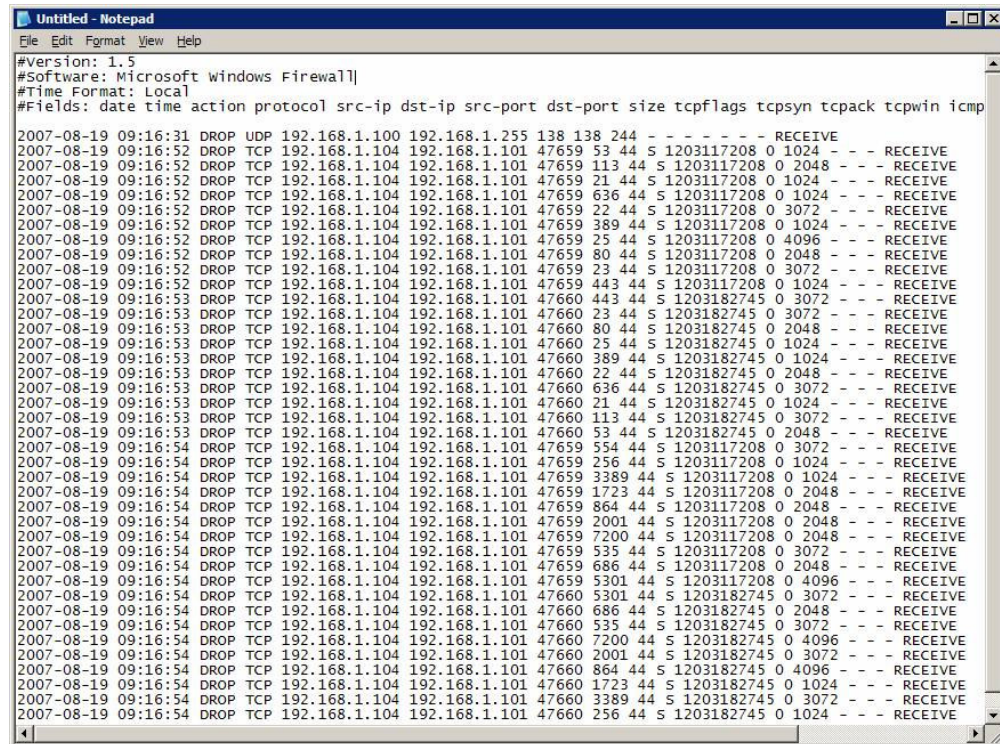
Don't forget to run wireshark:





Click Image for Larger Version

**Remember to run all your scans with the firewall off first, and then run them again with the firewall and logging on:**



```
#version: 1.5
#software: Microsoft Windows Firewall
#time format: Local
#fields: date time action protocol src-ip dst-ip src-port dst-port size tcpflags tcpsyn tcppack tcpwin icmp

2007-08-19 09:16:31 DROP UDP 192.168.1.100 192.168.1.255 138 138 244 - - - - - RECEIVE
2007-08-19 09:16:52 DROP TCP 192.168.1.104 192.168.1.101 47659 53 44 S 1203117208 0 1024 - - - RECEIVE
2007-08-19 09:16:52 DROP TCP 192.168.1.104 192.168.1.101 47659 113 44 S 1203117208 0 2048 - - - RECEIVE
2007-08-19 09:16:52 DROP TCP 192.168.1.104 192.168.1.101 47659 21 44 S 1203117208 0 1024 - - - RECEIVE
2007-08-19 09:16:52 DROP TCP 192.168.1.104 192.168.1.101 47659 636 44 S 1203117208 0 1024 - - - RECEIVE
2007-08-19 09:16:52 DROP TCP 192.168.1.104 192.168.1.101 47659 22 44 S 1203117208 0 3072 - - - RECEIVE
2007-08-19 09:16:52 DROP TCP 192.168.1.104 192.168.1.101 47659 389 44 S 1203117208 0 1024 - - - RECEIVE
2007-08-19 09:16:52 DROP TCP 192.168.1.104 192.168.1.101 47659 25 44 S 1203117208 0 4096 - - - RECEIVE
2007-08-19 09:16:52 DROP TCP 192.168.1.104 192.168.1.101 47659 80 44 S 1203117208 0 2048 - - - RECEIVE
2007-08-19 09:16:52 DROP TCP 192.168.1.104 192.168.1.101 47659 23 44 S 1203117208 0 3072 - - - RECEIVE
2007-08-19 09:16:52 DROP TCP 192.168.1.104 192.168.1.101 47659 443 44 S 1203117208 0 1024 - - - RECEIVE
2007-08-19 09:16:53 DROP TCP 192.168.1.104 192.168.1.101 47660 443 44 S 1203182745 0 3072 - - - RECEIVE
2007-08-19 09:16:53 DROP TCP 192.168.1.104 192.168.1.101 47660 23 44 S 1203182745 0 3072 - - - RECEIVE
2007-08-19 09:16:53 DROP TCP 192.168.1.104 192.168.1.101 47660 80 44 S 1203182745 0 2048 - - - RECEIVE
2007-08-19 09:16:53 DROP TCP 192.168.1.104 192.168.1.101 47660 25 44 S 1203182745 0 1024 - - - RECEIVE
2007-08-19 09:16:53 DROP TCP 192.168.1.104 192.168.1.101 47660 389 44 S 1203182745 0 1024 - - - RECEIVE
2007-08-19 09:16:53 DROP TCP 192.168.1.104 192.168.1.101 47660 22 44 S 1203182745 0 2048 - - - RECEIVE
2007-08-19 09:16:53 DROP TCP 192.168.1.104 192.168.1.101 47660 636 44 S 1203182745 0 3072 - - - RECEIVE
2007-08-19 09:16:53 DROP TCP 192.168.1.104 192.168.1.101 47660 21 44 S 1203182745 0 1024 - - - RECEIVE
2007-08-19 09:16:53 DROP TCP 192.168.1.104 192.168.1.101 47660 113 44 S 1203182745 0 3072 - - - RECEIVE
2007-08-19 09:16:53 DROP TCP 192.168.1.104 192.168.1.101 47660 53 44 S 1203182745 0 2048 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47659 534 44 S 1203117208 0 3072 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47659 256 44 S 1203117208 0 1024 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47659 3389 44 S 1203117208 0 1024 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47659 1723 44 S 1203117208 0 2048 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47659 864 44 S 1203117208 0 2048 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47659 2001 44 S 1203117208 0 2048 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47659 7200 44 S 1203117208 0 2048 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47659 535 44 S 1203117208 0 3072 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47659 686 44 S 1203117208 0 2048 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47659 5301 44 S 1203117208 0 4096 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47660 5301 44 S 1203182745 0 3072 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47660 686 44 S 1203182745 0 2048 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47660 535 44 S 1203182745 0 3072 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47660 7200 44 S 1203182745 0 4096 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47660 2001 44 S 1203182745 0 3072 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47660 864 44 S 1203182745 0 4096 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47660 1723 44 S 1203182745 0 1024 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47660 3389 44 S 1203182745 0 3072 - - - RECEIVE
2007-08-19 09:16:54 DROP TCP 192.168.1.104 192.168.1.101 47660 256 44 S 1203182745 0 1024 - - - RECEIVE
```

[Click Image for Larger Version](#)

## Conclusion

This should give you a great running start with Nmap. We covered not only what each scan does, but, more importantly, we gave you some valuable insight as to what an **ethical** hacker should be thinking when scanning. If the bad guys are thinking it, then you should, too. In Part 2 we will discuss all the special options we can add to our Nmap scans to really bring it to a much higher level.

---

**Kirby Tucker** AKA EH-Net Member "Kev" has been involved with the security industry since 1994. He likes to think of himself as an "old school" hacker and has been a great supporter of the concept of "**ethical** hacking". He has a passion for all aspects of hacking, including hardware as well as software. His day to day work involves anything from designing high-end computers to whitebox / blackbox penetration testing. With a diverse clientele ranging from medical, educational and law enforcement, life has been anything but dull.