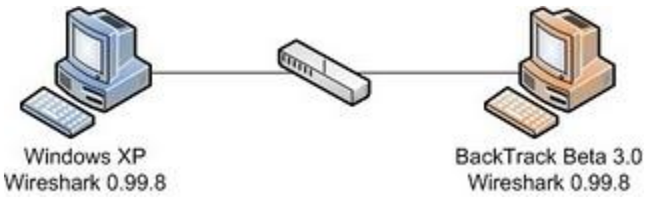
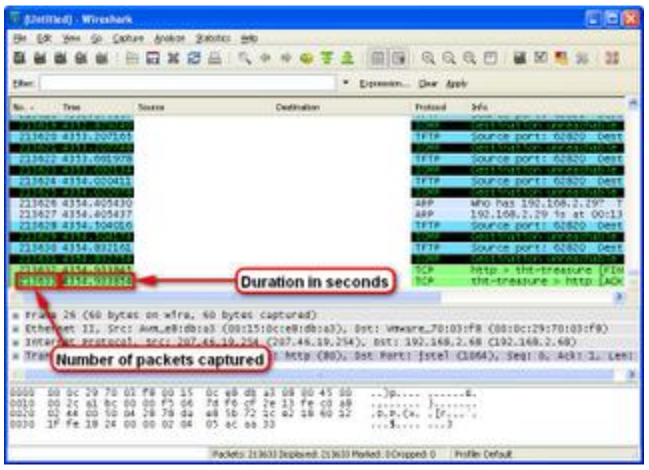
NMAP Detection and Countermeasures

Who is still enjoying the freedom? :)
Good, it seems like my lessons didn't work yet :)
Just kidding. I want you always to be safe, you heard that?

We dived into the scanning phase by learning nmap scans techniques, and today we'll see how these scans can be detected using the filters on Wireshark and what the countermeasures are.

For this article I created a lab with 2 PCs; 1 XP machines and 1 BackTrack beta 3.0
I also ran the wireshark for 1 hour and 10 minutes of capturing traffic, this traffic included Web browsing and NMAP Scans.





As you can see, Wireshark captured 213633 Packets in 4354 seconds. But on a real functional network these numbers are very humble; the real numbers will be scary. You have to try it yourself.

So imagine with me that you put Wireshark on your network which consists of 50 PCs all connected to the Internet, all of them are online all of the time, and your network has been scanned by a bad guy, how can you check that using Wireshark (assuming you don't have an Intrusion Detection installed and assuming you don't know what type of scan the bad guy used)?

Before we start, there are just 2 things I want to clarify:
1- Each protocol has a number assigned to it, these numbers are assigned by an organization called IANA (Internet Assigned Numbers Authority), IANA is responsible for the global coordination of the DNS Root, IP addressing, and other Internet protocol resources.
For example TCP is assigned the decimal number 6; UDP is assigned the decimal number 17, while IP is assigned the decimal number 4, and so on. For the whole list of the Assigned Internet Protocol Number, please visit the IANA website http://www.iana.org/assignments/protocol-numbers

2- Remember, in the Whois article we talked about the TCP flags, and that there are 8 flags; FIN, SYN, RST, PSH, ACK, URG, ECE, CWR
These flags have decimal numbers as well assigned to them:
FIN = 1
SYN = 2
RST = 4
PSH = 8
ACK = 16
URG = 32
ECE = 64
CWR = 128

So for example, if we want the SYN/ACK flag decimal value, we add 2 (which is the decimal value of the SYN flag) to 16 (which is the decimal value of the ACK flag), so the result would be 18.
What about the XMAS scan? From the article "Scanning using Nmap - Part 1" we learned that the XMAS scan sets the FIN, PSH and URG flags, so if we add 1 + 8 + 32, then the decimal values of the flags is 41.

Don't worry about these numbers; we will understand them as soon as we start analyzing the traffic.

TCP Connect Scan (Plain Vanilla)

"TCP Connect Scan" or "Plain Vanilla" attempts to complete the whole 3-Way handshake with each target host.
The attacker sends a SYN to the target, if the target's port is open and it responded with a SYN/ACK, and then the attacker will send the last ACK and tear down the connection using the RST.

Threshold:

The TCP 3-Way handshake is very normal to see a lot on your network's traffic, but if this kind of traffic is explosive and the number of them is extremely high per second on the network, then you have to investigate it

and check the IP responsible for these scans.
You are the only one who can specify this threshold, because you are the only one who knows your network’s traffic.

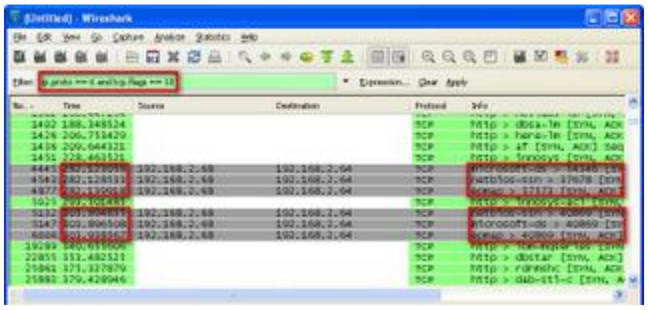
Keep Wireshark running for a whole working day, this will give you an average idea about the traffic on your network, and I said average because one day the network users might be busy working :) so the traffic gets less, or the opposite.

Filter:

The filter we are going to apply to check if a TCP Connect Scan occurred on our network is:
ip.proto == 6 and tcp.flags == 18

We chose 6 for the ip.proto because this is the Assigned Internet Protocol Number for the TCP protocol, and we chose 18 for the tcp.flags because 18 represents the decimal value of the SYN/ACK flag

TCP Connect Scan on Wireshark:


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: ip.proto == 6 and tcp.flags == 18
Expression... Clear Apply

```
No. .  Time         Source         Destination    Protocol  Info
 1402  188.348524                                 TCP       http > dbsa-lm [SYN, ACK
 1426  206.711429                                 TCP       http > hera-lm [SYN, ACK
 1436  209.644321                                 TCP       http > af [SYN, ACK] Seq
 1451  228.463521                                 TCP       http > innosys [SYN, ACK
 4445  [illegible]  192.168.2.68   192.168.2.64   TCP       microsoft-ds > 34346 [S
 4543  [illegible]  192.168.2.68   192.168.2.64   TCP       netbios-ssn > 37678 [SYN
 4877  [illegible]  192.168.2.68   192.168.2.64   TCP       epmap > 37171 [SYN, ACK
 5025  [illegible]                                TCP       http > innosys-acl [SYN,
 5132  [illegible]  192.168.2.68   192.168.2.64   TCP       netbios-ssn > 40869 [SYN
 5147  [illegible]  192.168.2.68   192.168.2.64   TCP       microsoft-ds > 40869 [S
 6006  [illegible]  192.168.2.68   192.168.2.64   TCP       epmap > 40869 [SYN, ACK]
19299  [illegible]                                TCP       [illegible]
22855  351.482521                                 TCP       http > dbstar [SYN, ACK]
25861  371.327879                                 TCP       http > rdrmshc [SYN, ACK
25881  379.428946                                 TCP       http > dab-sti-c [SYN, A
```

Because my lab is small compared to a real network, the filter is not so obvious, but as you can that the target (192.168.2.64) sends responses back within (1 second) to the attacker (192.168.2.68) telling him what ports are open.

TCP SYN Scan (Half Open)

TCP SYN scan is a little bit stealthier than the previous scan, because it uses a different technique. The attacker sends a SYN to the targets, if the target’s port is open and it responded with a SYN/ACK, then the attacker will immediately tear down the connection using the RST.

Threshold:

As we know, SYN Scan starts as the 3-Way handshake, but instead of completing the handshake, it terminates the connection with a RST flag. So this kind of traffic might appear to be normal, but you have to notice the number of the Half Open connections, if the SYN packets are greater than the SYN/ACK packets, then there is something wrong.

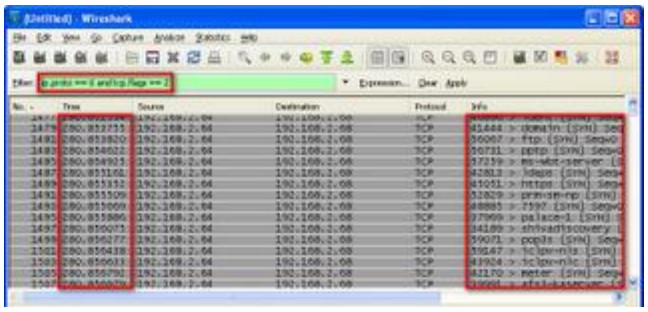
Filter:

The filter we are going to apply to check if a TCP SYN Scan occurred on our network is:

ip.proto == 6 and tcp.flags == 2

We chose 6 for the ip.proto because this is the Assigned Internet Protocol Number for the TCP protocol, and we chose 2 for the tcp.flags because 2 represents the decimal value of the SYN flag.

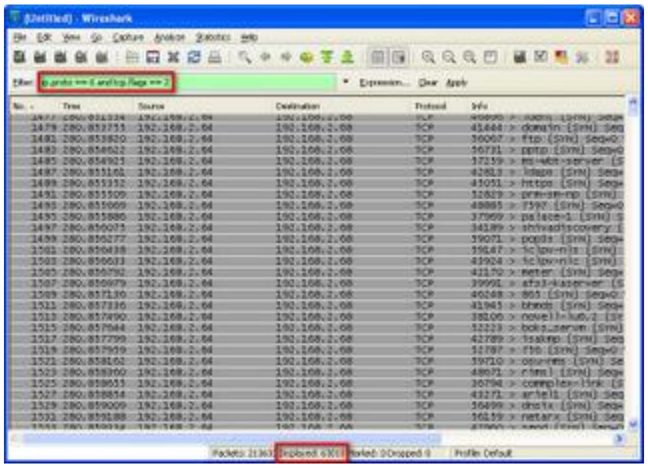
TCP SYN Scan on Wireshark:


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

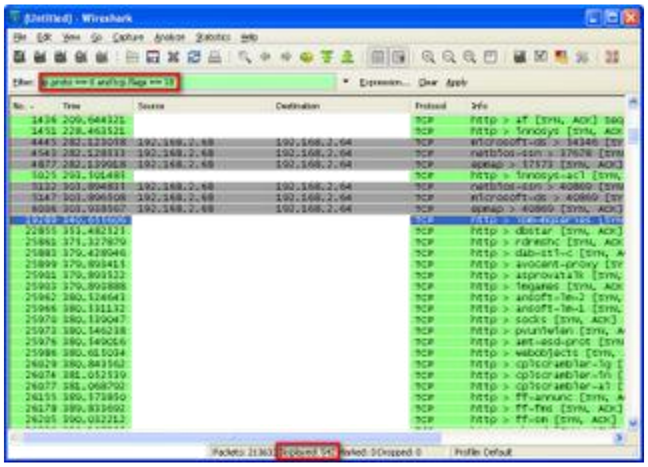
```
Filter: ip.proto == 6 and tcp.flags == 2

No.   Time        Source        Destination   Protocol Info
1479  280.852733  192.168.2.64  192.168.2.68  TCP      41444 > domain [SYN] Seq
1481  280.853820  192.168.2.64  192.168.2.68  TCP      56067 > ftp [SYN] Seq=0
1483  280.854622  192.168.2.64  192.168.2.68  TCP      56731 > pptp [SYN] Seq=0
1485  280.854923  192.168.2.64  192.168.2.68  TCP      37159 > ms-wbt-server [S
1487  280.855161  192.168.2.64  192.168.2.68  TCP      42813 > ldaps [SYN] Seq=
1489  280.855352  192.168.2.64  192.168.2.68  TCP      45051 > https [SYN] Seq=
1491  280.855509  192.168.2.64  192.168.2.68  TCP      32829 > prm-sm-np [SYN]
1493  280.855069  192.168.2.64  192.168.2.68  TCP      48885 > 7597 [SYN] Seq=0
1495  280.855886  192.168.2.64  192.168.2.68  TCP      37969 > palace-1 [SYN] S
1497  280.856075  192.168.2.64  192.168.2.68  TCP      34189 > shivadiscovery [
1499  280.856277  192.168.2.64  192.168.2.68  TCP      59071 > pop3s [SYN] Seq=
1501  280.856438  192.168.2.64  192.168.2.68  TCP      59147 > tclprv-nls [SYN]
1503  280.856633  192.168.2.64  192.168.2.68  TCP      43924 > tclprv-nlc [SYN]
1505  280.856792  192.168.2.64  192.168.2.68  TCP      42170 > meter [SYN] Seq=
```

The attacker here is 192.168.2.64, and he is sending the target 192.168.2.68 a lot of SYN packets in a very small time zone, this for sure indicates a SYN Scan.

Let’s compare the number of SYN flagged packets to the SYN/ACK flagged ones.


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
Filter: ip.proto == 6 and tcp.flags == 2

No.   Time        Source        Destination   Protocol Info
1479  280.852733  192.168.2.64  192.168.2.68  TCP      41444 > domain [SYN] Seq
1481  280.853820  192.168.2.64  192.168.2.68  TCP      56067 > ftp [SYN] Seq=0
1483  280.854622  192.168.2.64  192.168.2.68  TCP      56731 > pptp [SYN] Seq=0
1485  280.854923  192.168.2.64  192.168.2.68  TCP      37159 > ms-wbt-server [S
1487  280.855161  192.168.2.64  192.168.2.68  TCP      42813 > ldaps [SYN] Seq=
1489  280.855352  192.168.2.64  192.168.2.68  TCP      45051 > https [SYN] Seq=
1491  280.855509  192.168.2.64  192.168.2.68  TCP      32829 > prm-sm-np [SYN]
1493  280.855069  192.168.2.64  192.168.2.68  TCP      48885 > 7597 [SYN] Seq=0
1495  280.855886  192.168.2.64  192.168.2.68  TCP      37969 > palace-1 [SYN] S
1497  280.856075  192.168.2.64  192.168.2.68  TCP      34189 > shivadiscovery [
1499  280.856277  192.168.2.64  192.168.2.68  TCP      59071 > pop3s [SYN] Seq=
1501  280.856438  192.168.2.64  192.168.2.68  TCP      59147 > tclprv-nls [SYN]
1503  280.856633  192.168.2.64  192.168.2.68  TCP      43924 > tclprv-nlc [SYN]
1505  280.856792  192.168.2.64  192.168.2.68  TCP      42170 > meter [SYN] Seq=
1507  280.856979  192.168.2.64  192.168.2.68  TCP      39991 > afs3-kaserver [S
1509  280.857130  192.168.2.64  192.168.2.68  TCP      46248 > 865 [SYN] Seq=0
1511  280.857336  192.168.2.64  192.168.2.68  TCP      41965 > bhmds [SYN] Seq=
1513  280.857490  192.168.2.64  192.168.2.68  TCP      38106 > novell-lu6.2 [SY
1515  280.857644  192.168.2.64  192.168.2.68  TCP      32223 > boks_servm [SYN]
1517  280.857799  192.168.2.64  192.168.2.68  TCP      42789 > isakmp [SYN] Seq
1519  280.857959  192.168.2.64  192.168.2.68  TCP      51787 > ftp [SYN] Seq=0
1521  280.858162  192.168.2.64  192.168.2.68  TCP      59710 > osu-nms [SYN] Se
1523  280.858360  192.168.2.64  192.168.2.68  TCP      48671 > rtmp [SYN] Seq=
1525  280.858633  192.168.2.64  192.168.2.68  TCP      36794 > complex-link [S
1527  280.858834  192.168.2.64  192.168.2.68  TCP      43271 > ariel1 [SYN] Seq
1529  280.859009  192.168.2.64  192.168.2.68  TCP      56499 > dnsix [SYN] Seq=
1531  280.859188  192.168.2.64  192.168.2.68  TCP      56159 > netarx [SYN] Seq

Packets: 21363 Displayed: 6301 Marked: 0 Dropped: 0    Profile: Default
```



```
Filter: ip.proto == 6 and tcp.flags == 18

No. -  Time         Source          Destination     Protocol  Info
 1436  209.644321                                   TCP       http > af [SYN, ACK] Seq
 1451  228.463521                                   TCP       http > innosys [SYN, ACK
 4445  282.123098   192.168.2.68    192.168.2.64    TCP       microsoft-ds > 34346 [SY
 4543  282.128313   192.168.2.68    192.168.2.64    TCP       netbios-ssn > 37678 [SYN
 4877  282.139018   192.168.2.68    192.168.2.64    TCP       epmap > 37573 [SYN, ACK]
 5025  293.101481                                   TCP       http > innosys-acl [SYN,
 5122  303.894831   192.168.2.68    192.168.2.64    TCP       netbios-ssn > 40869 [SYN
 5147  303.896508   192.168.2.68    192.168.2.64    TCP       microsoft-ds > 40869 [SY
 6086  303.938507   192.168.2.68    192.168.2.64    TCP       epmap > 40869 [SYN, ACK]
[illegible]                                         TCP       http > [illegible]
22855  351.482121                                   TCP       http > dbstar [SYN, ACK]
25861  371.327879                                   TCP       http > rdmnshc [SYN, ACK
25883  379.428946                                   TCP       http > dab-sti-c [SYN, A
25899  379.893411                                   TCP       http > avocent-proxy [SY
25901  379.893522                                   TCP       http > asprovatalk [SYN,
25903  379.893888                                   TCP       http > leganes [SYN, ACK
25962  380.124643                                   TCP       http > ansoft-lm-2 [SYN,
25966  380.131132                                   TCP       http > ansoft-lm-1 [SYN,
25970  380.139047                                   TCP       http > socks [SYN, ACK]
25973  380.146238                                   TCP       http > pvuniwien [SYN, A
25976  380.149066                                   TCP       http > amt-esd-prot [SYN
25986  380.615034                                   TCP       http > webobjects [SYN,
26029  380.843562                                   TCP       http > cplscrambler-lg [
26074  381.052539                                   TCP       http > cplscrambler-in [
26077  381.068792                                   TCP       http > cplscrambler-al [
26155  389.171850                                   TCP       http > ff-annunc [SYN, A
26178  389.833692                                   TCP       http > ff-fms [SYN, ACK]
26205  390.032212                                   TCP       http > ff-sm [SYN, ACK]

Packets: 213631 Displayed: 542 Marked: 0 Dropped: 0    Profile: Default
```

(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Expression... Clear Apply

Wow, did you see that? When I applied the SYN flag filter, Wireshark displayed 63018 packets.
While when I applied the SYN/ACK flag filter, Wireshark displayed 542 packets. There is a huge difference between both numbers, this difference indicates the huge amount of Half Open connections.

TCP FIN Scan

The FIN Scan breaks the rule of TCP connection establishment because it sends an unexpected packet at the start of the connection, which is the FIN flag.

Threshold:

FIN flags are part of any communication between 2 hosts, because this communication has to be ended at a moment, but if you see an explosive number of FIN flagged packets without a previous established connection, then take care of that.

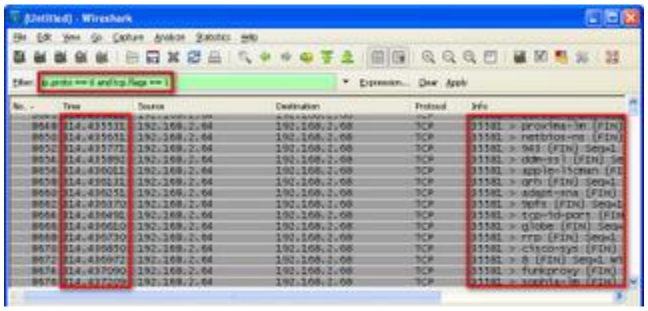
Filter:

The filter we are going to apply to check if a TCP FIN Scan occurred on our network is:

```
ip.proto == 6 and tcp.flags == 1
```

We chose 6 for the ip.proto because this is the Assigned Internet Protocol Number for the TCP protocol, and we chose 1 for the tcp.flags because 1 represents the decimal value of the SYN flag.

TCP FIN Scan on Wireshark:


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: ip.proto == 6 and tcp.flags == 1
Expression... Clear Apply

```
No. -  Time        Source          Destination     Protocol  Info
8648   314.435531  192.168.2.64    192.168.2.68    TCP       31381 > proxima-lm [FIN]
8650   314.435651  192.168.2.64    192.168.2.68    TCP       31381 > netbios-ns [FIN]
8652   314.435771  192.168.2.64    192.168.2.68    TCP       31381 > 943 [FIN] Seq=1
8654   314.435892  192.168.2.64    192.168.2.68    TCP       31381 > ddm-ssl [FIN] Se
8656   314.436011  192.168.2.64    192.168.2.68    TCP       31381 > apple-licman [FI
8658   314.436131  192.168.2.64    192.168.2.68    TCP       31381 > qrh [FIN] Seq=1
8660   314.436251  192.168.2.64    192.168.2.68    TCP       31381 > adapt-sna [FIN]
8662   314.436370  192.168.2.64    192.168.2.68    TCP       31381 > 9pfs [FIN] Seq=1
8664   314.436490  192.168.2.64    192.168.2.68    TCP       31381 > tcp-id-port [FIN
8666   314.436610  192.168.2.64    192.168.2.68    TCP       31381 > globe [FIN] Seq=
8668   314.436730  192.168.2.64    192.168.2.68    TCP       31381 > rrp [FIN] Seq=1
8670   314.436850  192.168.2.64    192.168.2.68    TCP       31381 > cisco-sys [FIN]
8672   314.436972  192.168.2.64    192.168.2.68    TCP       31381 > 8 [FIN] Seq=1 Wi
8674   314.437090  192.168.2.64    192.168.2.68    TCP       31381 > funkproxy [FIN]
8676   314.437209  192.168.2.64    192.168.2.68    TCP       31381 > sophia-lm [FIN]
```

The attacker here is 192.168.2.64, and he is sending the target 192.168.2.68 a lot of FIN packets in a very small time zone, this for sure indicates a FIN Scan.

TCP XMAS Scan

The XMAS Scan breaks the rule of TCP connection establishment because it sends an unexpected packet at the start of the connection, by setting the FIN, PSH and URG flags.

Threshold:

XMAS packets should never be seen on your network, so if you see a single XMAS flagged packet, then someone is scanning your network.

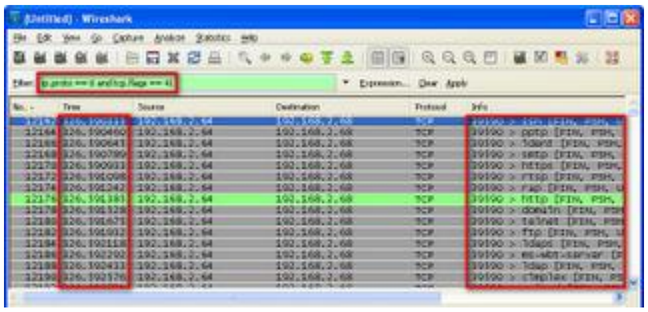
Filter:

The filter we are going to apply to check if a TCP XMAS Scan occurred on our network is:

ip.proto == 6 and tcp.flags == 41

We chose 6 for the ip.proto because this is the Assigned Internet Protocol Number for the TCP protocol, and we chose 41 for the tcp.flags because 41 represents the decimal value of the (FIN + PSH + URG flags).

TCP XMAS Scan on Wireshark:


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: ip.proto == 6 and tcp.flags == 41
Expression... Clear Apply

```
No. -  Time        Source          Destination     Protocol  Info
12162  326.190311  192.168.2.64    192.168.2.68    TCP       39190 > [illegible] [FIN, PSH,
12164  326.190460  192.168.2.64    192.168.2.68    TCP       39190 > pptp [FIN, PSH,
12166  326.190641  192.168.2.64    192.168.2.68    TCP       39190 > [illegible] [FIN, PSH,
12168  326.190789  192.168.2.64    192.168.2.68    TCP       39190 > smtp [FIN, PSH,
12170  326.190931  192.168.2.64    192.168.2.68    TCP       39190 > https [FIN, PSH,
12172  326.191098  192.168.2.64    192.168.2.68    TCP       39190 > rtsp [FIN, PSH,
12174  326.191242  192.168.2.64    192.168.2.68    TCP       39190 > rap [FIN, PSH, U
12176  326.191381  192.168.2.64    192.168.2.68    TCP       39190 > http [FIN, PSH,
12178  326.191528  192.168.2.64    192.168.2.68    TCP       39190 > domain [FIN, PSH
12180  326.191671  192.168.2.64    192.168.2.68    TCP       39190 > telnet [FIN, PSH
12182  326.191912  192.168.2.64    192.168.2.68    TCP       39190 > ftp [FIN, PSH, U
12184  326.192118  192.168.2.64    192.168.2.68    TCP       39190 > ldaps [FIN, PSH,
12186  326.192292  192.168.2.64    192.168.2.68    TCP       39190 > ms-wbt-server [F
12188  326.192431  192.168.2.64    192.168.2.68    TCP       39190 > ldap [FIN, PSH,
12190  326.192576  192.168.2.64    192.168.2.68    TCP       39190 > cimplex [FIN, PS
```

As we said, you should never ever see an XMAS packet on your network for any reason, and as you can see in the picture the attacker 192.168.2.64 is doing an XMAS Scan against 192.168.2.68.

TCP NULL Scan

The NULL Scan breaks the rule of TCP connection establishment because it sends an unexpected packet at the start of the connection, by all flags from the packets.

Threshold:

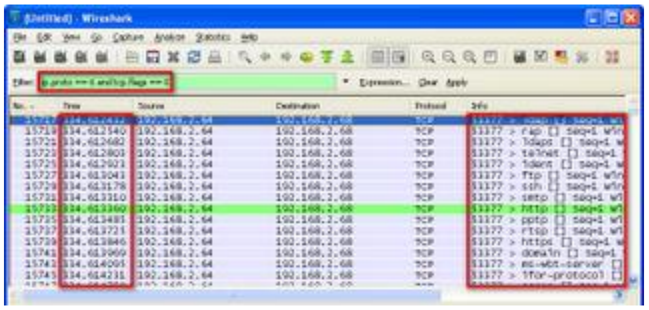
NULL packets should never be seen on your network, so if you see a single NULL flagged packet, then someone is scanning your network.

Filter:

The filter we are going to apply to check if a TCP XMAS Scan occurred on our network is:
ip.proto == 6 and tcp.flags == 0

We chose 6 for the ip.proto because this is the Assigned Internet Protocol Number for the TCP protocol, and we chose 0 for the tcp.flags because 0 means that all flags are removed.

TCP NULL Scan on Wireshark:


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
Filter: ip.proto == 6 and tcp.flags == 0

No.    Time        Source         Destination    Protocol  Info
15717  334.612412  192.168.2.64   192.168.2.68   TCP       53177 > [illegible] [] Seq=1 Win
15719  334.612540  192.168.2.64   192.168.2.68   TCP       53177 > rap [] Seq=1 Win
15721  334.612682  192.168.2.64   192.168.2.68   TCP       53177 > ldaps [] Seq=1 W
15723  334.612803  192.168.2.64   192.168.2.68   TCP       53177 > telnet [] Seq=1
15725  334.612921  192.168.2.64   192.168.2.68   TCP       53177 > ident [] Seq=1 W
15727  334.613041  192.168.2.64   192.168.2.68   TCP       53177 > ftp [] Seq=1 Win
15729  334.613178  192.168.2.64   192.168.2.68   TCP       53177 > ssh [] Seq=1 Win
15731  334.613310  192.168.2.64   192.168.2.68   TCP       53177 > smtp [] Seq=1 Wi
15733  334.613360  192.168.2.64   192.168.2.68   TCP       53177 > http [] Seq=1 Wi
15735  334.613481  192.168.2.64   192.168.2.68   TCP       53177 > pptp [] Seq=1 Wi
15737  334.613721  192.168.2.64   192.168.2.68   TCP       53177 > rtsp [] Seq=1 Wi
15739  334.613846  192.168.2.64   192.168.2.68   TCP       53177 > https [] Seq=1 W
15741  334.613969  192.168.2.64   192.168.2.68   TCP       53177 > domain [] Seq=1
15743  334.614095  192.168.2.64   192.168.2.68   TCP       53177 > ms-wbt-server []
15745  334.614231  192.168.2.64   192.168.2.68   TCP       53177 > tfor-protocol []
```

As we said, you should never ever see an NULL packet on your network for any reason, and as you can see in the picture the attacker 192.168.2.64 is doing an NULL Scan against 192.168.2.68.

TCP ACK Scan

The idea behind the TCP ACK scan is very simple and very smart; I will give you an analogy to get how it is working.
We don’t know each other, right?
Imagine I met you once in the street, and suddenly I went to you and said “hey man, where have you been all of this time? Not even a single mail,

shame on you?” :)
What will you think? You will say “This man knows me for sure, but probably I don’t remember him”, and then you will start answering me “Oh, I’m fine, and sorry for not sending you mails but I was very busy the last few weeks. I got a baby and…” and you will start talking friendly.

TCP ACK Scan almost works the same, it sends an ACK to the target’s ports, the target will think “it seems like I started a connection with this computer before, let’s answer him”

Threshold:

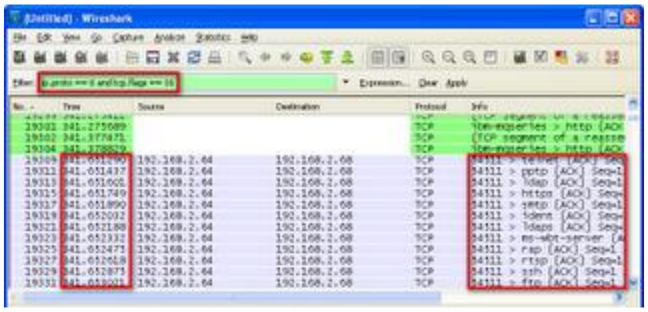
As we know, ACK is the last packet in the 3-Way handshake, thus seeing ACK packet on the network is normal, but if you see an extreme high number of them, then an ACK scan is occurring.

Filter:

The filter we are going to apply to check if a TCP SYN Scan occurred on our network is:
`ip.proto == 6 and tcp.flags == 16`

We chose 6 for the ip.proto because this is the Assigned Internet Protocol Number for the TCP protocol, and we chose 16 for the tcp.flags because 16 represents the decimal value of the ACK flag.

TCP ACK Scan on Wireshark:


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: ip.proto == 6 and tcp.flags == 16
Expression... Clear Apply

```
No. -  Time        Source         Destination    Protocol  Info
19301  341.275089                                TCP       3com-amp3 > http [ACK
19302  341.377471                                TCP       [TCP segment of a reasse
19304  341.378829                                TCP       3com-amp3 > http [ACK
19309  341.651290  192.168.2.64   192.168.2.68   TCP       54311 > telnet [ACK] Seq
19311  341.651437  192.168.2.64   192.168.2.68   TCP       54311 > pptp [ACK] Seq=1
19313  341.651601  192.168.2.64   192.168.2.68   TCP       54311 > ldap [ACK] Seq=1
19315  341.651749  192.168.2.64   192.168.2.68   TCP       54311 > https [ACK] Seq=
19317  341.651890  192.168.2.64   192.168.2.68   TCP       54311 > smtp [ACK] Seq=1
19319  341.652032  192.168.2.64   192.168.2.68   TCP       54311 > ident [ACK] Seq=
19321  341.652188  192.168.2.64   192.168.2.68   TCP       54311 > ldaps [ACK] Seq=
19323  341.652332  192.168.2.64   192.168.2.68   TCP       54311 > ms-wbt-server [A
19325  341.652473  192.168.2.64   192.168.2.68   TCP       54311 > rap [ACK] Seq=1
19327  341.652618  192.168.2.64   192.168.2.68   TCP       54311 > rtsp [ACK] Seq=1
19329  341.652873  192.168.2.64   192.168.2.68   TCP       54311 > ssh [ACK] Seq=1
19331  341.653021  192.168.2.64   192.168.2.68   TCP       54311 > ftp [ACK] Seq=1
```

The attacker here is 192.168.2.64, and he is sending the target 192.168.2.68 a lot of ACK packets in a very small time zone, this for sure indicates an ACK Scan.

UDP Scan

Because UDP is simpler than TCP; no 3-Way handshaking, no Flags, no Sequence numbers, so the UDP scan is very simplified.
The attacker sends a UDP packet to each port on the Target. There might be

here 3 responses; an ICMP Port Unreachable (which indicates a closed port), no response (which means the port might be open or filtered by firewall), or a UDP response

Threshold:

These packets are not supposed to be seen on the network, so whenever you see them, they mean something bad.

Filter:

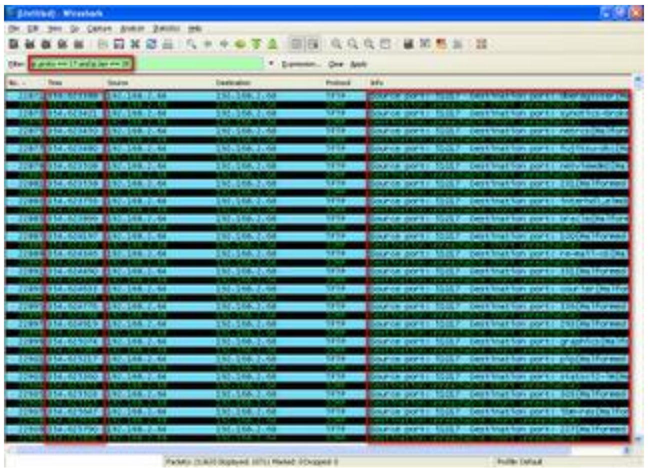
The filter we are going to apply to check if a TCP SYN Scan occurred on our network is:

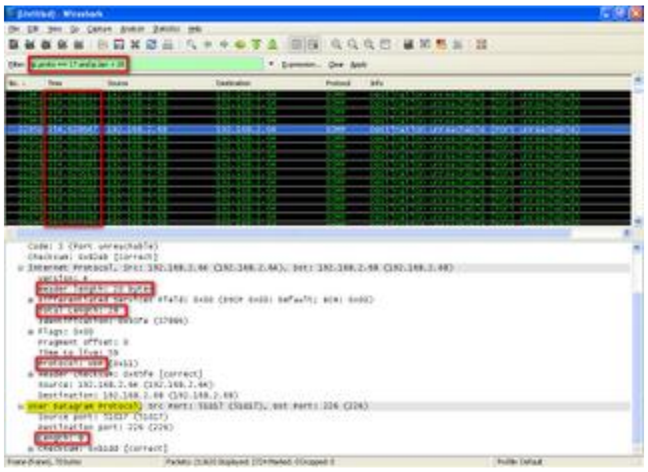
```
ip.proto == 17 and ip.len = 28
```

Or you can change the equal sign to “Greater Than”, the reason is that some scans can add junk data after the UDP packet, instead of sending an IP datagram with no data.

We chose 17 for the ip.proto because this is the Assigned Internet Protocol Number for the UDP protocol, and an IP Total Length (Specifies the length, in bytes, of the entire IP packet, including the data and header) equal or greater than 28, and we chose 28 because the length of the IP header is 20 bytes and the length of the UDP header is 8 bytes, so 20+8=28.

TCP UDP Scan on Wireshark:





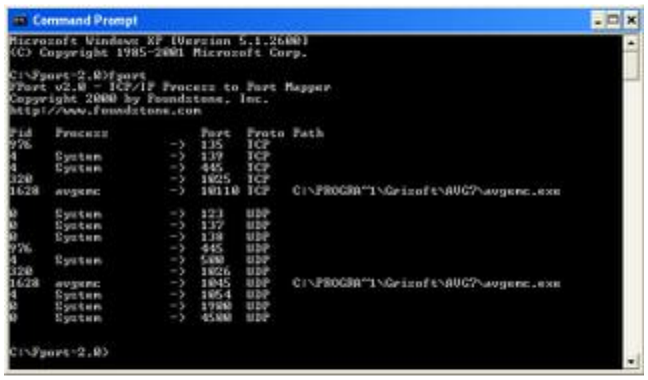
The attacker here is 192.168.2.64, and he is sending the target 192.168.2.68 a lot of UDP packets in a very small time zone, this for sure indicates an UDP Scan.
And the target 192.168.2.68 kept responding with an ICMP message “Destination Unreachable – Port Unreachable” which indicates a closed port.

After we learned how to detect a Scan, we have to know how can we defend or avoid these kind of attacks.

Port Scanning Countermeasures:

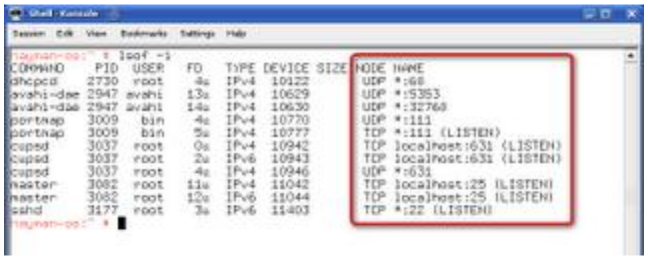
1- The first and most obvious countermeasure is to close all of the unwanted ports, most of the administrators (whether Sysadmins or Netadmins) install by default, this type of installation is popular because it’s the easiest, why should I bother myself by trying to find the open ports? How to close these ports? What if closing a port causes me problems with the big boss because something suddenly stopped? And a lot of these excuses.

First, we have to detect the open ports, and to do that:
For Windows users, I like to use Fport (ex Foundstone, McAfee recently) and TCPview (ex Sysinternals, Microsoft recently)



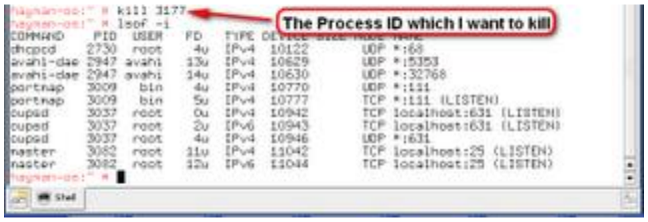
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Fport-2.0>fport
FPort v2.0 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid   Process            Port  Proto Path
976                  ->  135   TCP
4     System         ->  139   TCP
4     System         ->  445   TCP
320                  ->  1025  TCP
1628  avgemc         ->  10110 TCP   C:\PROGRA~1\Grisoft\AVG7\avgemc.exe

0     System         ->  123   UDP
0     System         ->  137   UDP
0     System         ->  138   UDP
976                  ->  445   UDP
4     System         ->  500   UDP
320                  ->  1026  UDP
1628  avgemc         ->  1045  UDP   C:\PROGRA~1\Grisoft\AVG7\avgemc.exe
4     System         ->  1054  UDP
0     System         ->  1900  UDP
0     System         ->  4500  UDP

C:\Fport-2.0>
```

For Linux users, I like the command line `lsof` “List Open Files”

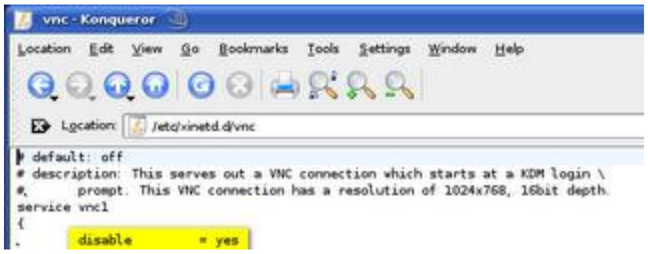


```
haynan-os:~ # lsof -i
COMMAND     PID  USER    FD    TYPE DEVICE SIZE NODE NAME
dhcpcd     2730  root    4u    IPv4  10122       UDP *:68
avahi-dae  2947 avahi   13u    IPv4  10629       UDP *:5353
avahi-dae  2947 avahi   14u    IPv4  10630       UDP *:32768
portmap    3009   bin    4u    IPv4  10770       UDP *:111
portmap    3009   bin    5u    IPv4  10777       TCP *:111 (LISTEN)
cupsd      3037  root    0u    IPv4  10942       TCP localhost:631 (LISTEN)
cupsd      3037  root    2u    IPv6  10943       TCP localhost:631 (LISTEN)
cupsd      3037  root    4u    IPv4  10946       UDP *:631
master     3082  root   11u    IPv4  11042       TCP localhost:25 (LISTEN)
master     3082  root   12u    IPv6  11044       TCP localhost:25 (LISTEN)
sshd       3177  root    3u    IPv6  11403       TCP *:22 (LISTEN)
haynan-os:~ #
```

After finding the unwanted ports, now it’s time to close the process which is using this port (if the process or service not needed).
On Windows, you can use [Pskill](#)(ex Sysinternals, Microsoft recently), the command line `kill` or `taskkill`, or use “Services” management console from the “Administrative Tools” in the “Control Panel”
On Linux, you can use the `kill` command line with the specified PID (Process ID), this command line is exactly like the Windows taskkill command, used to kill the chosen process till the next restart.
If you want to close the process permanently, then use the “Services” management console on Windows or edit the /etc/xinited.d/[service] on Linux and include this line `disable = yes`



```
haynan-os:~ # kill 3177
haynan-os:~ # lsof -i
COMMAND     PID  USER    FD    TYPE DEVICE SIZE NODE NAME
dhcpcd     2730  root    4u    IPv4  10122       UDP *:68
avahi-dae  2947 avahi   13u    IPv4  10629       UDP *:5353
avahi-dae  2947 avahi   14u    IPv4  10630       UDP *:32768
portmap    3009   bin    4u    IPv4  10770       UDP *:111
portmap    3009   bin    5u    IPv4  10777       TCP *:111 (LISTEN)
cupsd      3037  root    0u    IPv4  10942       TCP localhost:631 (LISTEN)
cupsd      3037  root    2u    IPv6  10943       TCP localhost:631 (LISTEN)
cupsd      3037  root    4u    IPv4  10946       UDP *:631
master     3082  root   11u    IPv4  11042       TCP localhost:25 (LISTEN)
master     3082  root   12u    IPv6  11044       TCP localhost:25 (LISTEN)
haynan-os:~ #
```



Location: /etc/xinetd.d/vnc

```
# default: off
# description: This serves out a VNC connection which starts at a KDM login \
#,       prompt. This VNC connection has a resolution of 1024x768, 16bit depth.
service vnc1
{
        disable         = yes
```

2- The second way that will help you defend yourself is to attack yourself before the hacker does. What I mean here is to try using the scanning tools yourself against your network, this way you will be able to see in reality how your network is reacting towards attacks.
BUT, 2 things you have to notice before doing that:
The first is to make sure that you have an approval for doing that, your boss might not be as kind as my boss :)

The second thing to notice, is that scanning tools are creating extra traffic on your network, because they are sending and receiving packets, this will eat from your traffic bandwidth, thus slowing down your network performance. So for that, just monitor your network performance while you are scanning.

3- Use Stateful Packet filter and Proxy devices
Normally, there are 3 types of filtering devices: Static Packet filter, Stateful Packet filter, and Proxy.
The Static Packet firewalls (such as Cisco Routers) are used to block simple traffic depending on simple filters, such as filtering according to the IP address.

While Stateful (such as Cisco PIX Firewall and Checkpoint Firewall) and Proxies keep records of earlier packets, for example if I'm sending you an ACK flagged packet, the Stateful filter device will check the records of the already-opened connections, if it finds that the ACK packet doesn't belong to a previous communication, then this packet will be dropped.