



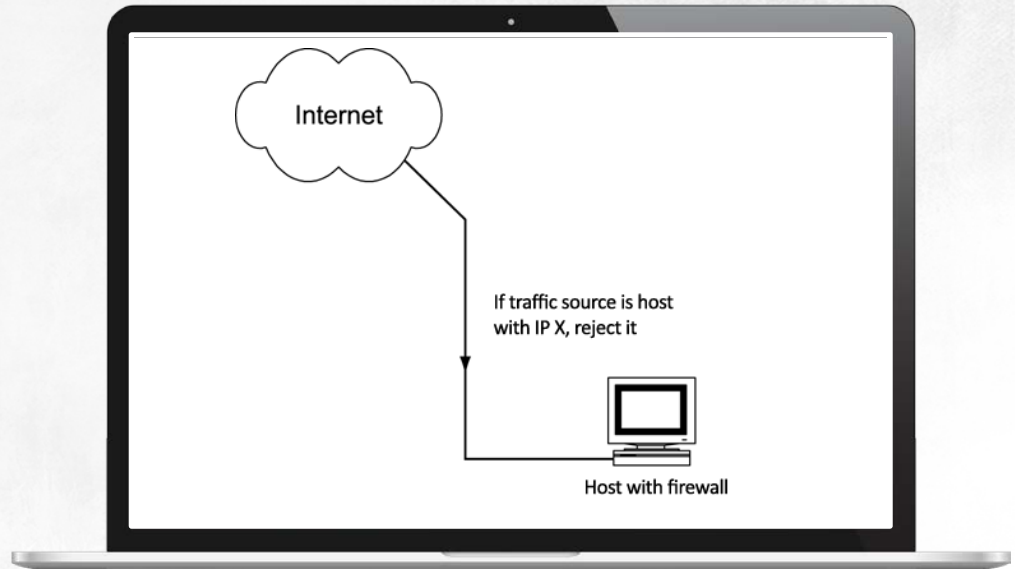
# Monitoring Transmitted Data



# Firewalls and Configuring Firewalls

The primary role of a firewall is to monitor and filter inbound and outbound traffic across hosts or networks. They do this based on a set of defined rules, or the requirements that need to be met to process a packet

A typical firewall rule is conditional: If the requirement A is satisfied, start action B. For example: If a source IP address is from the W.X.Y.Z network, block the packet

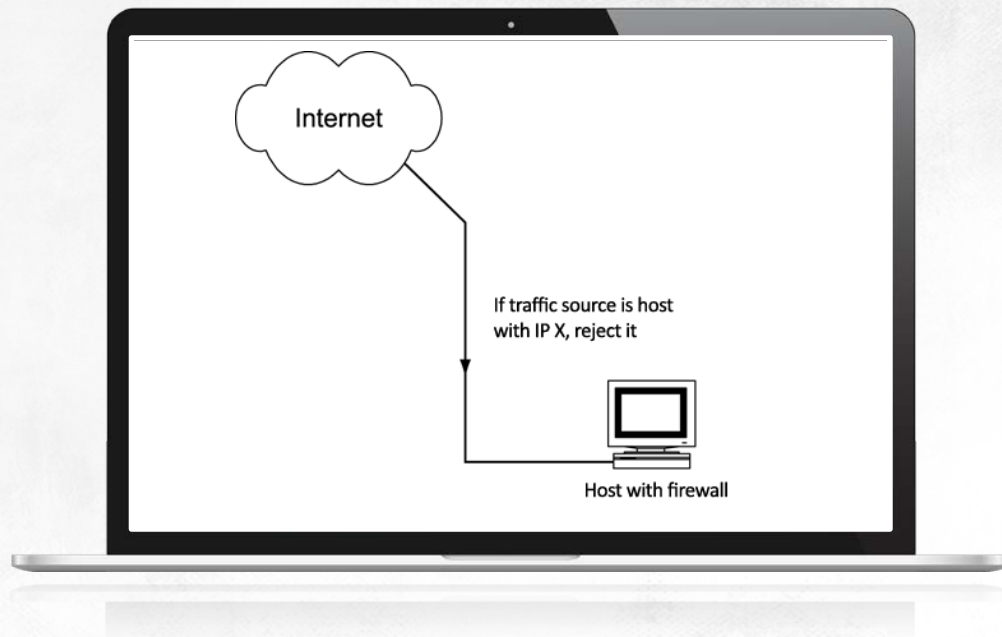


# Firewalls and Configuring Firewalls

## Firewall rules may be based on:

- A default REJECT policy: everything is blocked unless specified otherwise
- A default ACCEPT policy: everything is allowed unless specified otherwise

The first and foremost feature of firewalls is its ability to control and decide what IP addresses can be accepted





# Firewalls and Configuring Firewalls

**The first firewalls were created** in an environment that did not know or need Network Address Translation and that had all newly networked hosts having correct Internet addresses

**The introduction of address translation** led to the expansion of the packet filtering function in firewalls. Before, they would only control packets based on their source and destination IP addresses. After, they could also filter traffic based on the rest of header fields of network layer protocols and filter packets based on the destination connection addresses for computers that used the NAT mechanism

**The next milestone in firewall development** was the ability to filter packets based on transport layer protocol headers (TCP and UDP headers). This led to the creation of firewall rules for source and destination port numbers, OSI model layer four protocols and for types of transmitted messages

# Firewalls and **Configuring Firewalls**

**Depending on the mechanism** they use for filtering transport layer messages, firewalls are divided into static (stateless) and dynamic (monitoring the state of active sessions)

**Another milestone** is filtering packets based on transmitted data rather than packet headers  
While application firewalls offer satisfactory protection from packet fragmentation or port tunnelling attacks, their basic flaw is that they widely use the ACCEPT policy as the default setting

**Encrypting transmitted data** is another element detrimental to firewall effectiveness

# Firewalls and Configuring Firewalls

While flawed, firewalls are still widely used to provide a variety of helpful functions:

- Controlling traffic between internal and external networks
- Blocking control and diagnostic messages at the enterprise network perimeter
- Controlling traffic between a corporate network's subnetworks
- Blocking access to specified servers or network services
- Logging connection history for example for further analysis





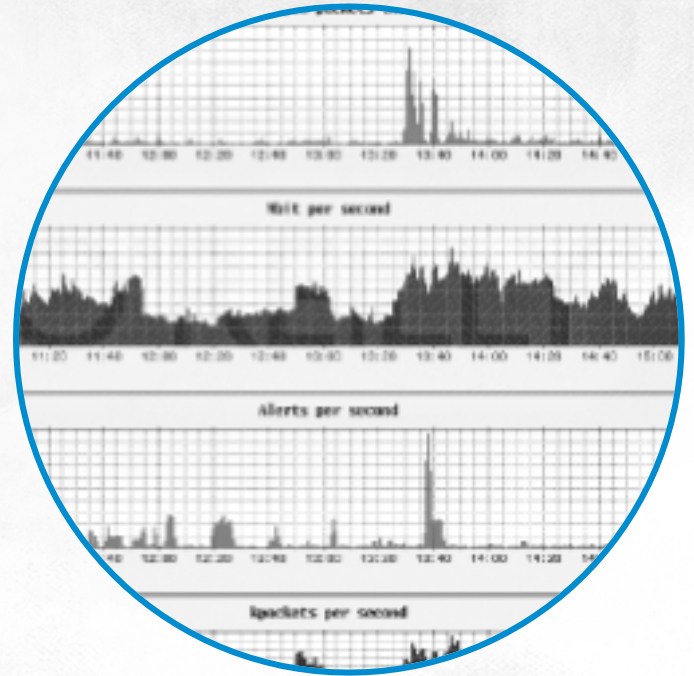
# Network intrusion detection systems

Network intrusion detection systems (NIDS) analyse data transmitted across a network in real time to automatically detect intrusions

They rely on the assumption that an intrusion involves a sequence of correlated actions that share some common traits

A NIDS can detect intrusions in two ways:

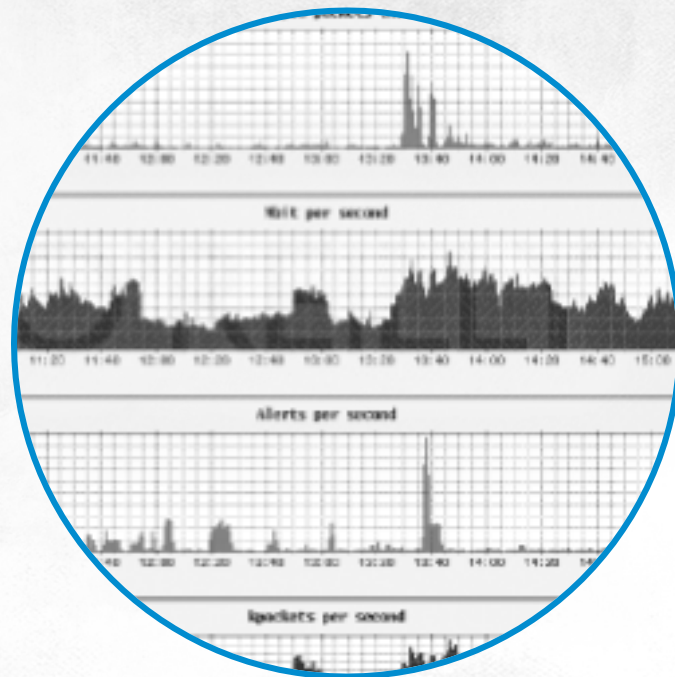
- Using intrusion characteristics, or by comparing intercepted data against reference data stored in a NIDS database (attack signatures)
- By detecting non-standard behaviours, or by comparing monitored network traffic against normal user activity references



# Network intrusion detection systems

After an attack is discovered, the NIDS may respond to it in an active or passive manner:

- An active response is stopping the attacker's session by injecting the FIN or RESET messages or logging out the attacker automatically
- A passive response is simply logging the information about the intrusion

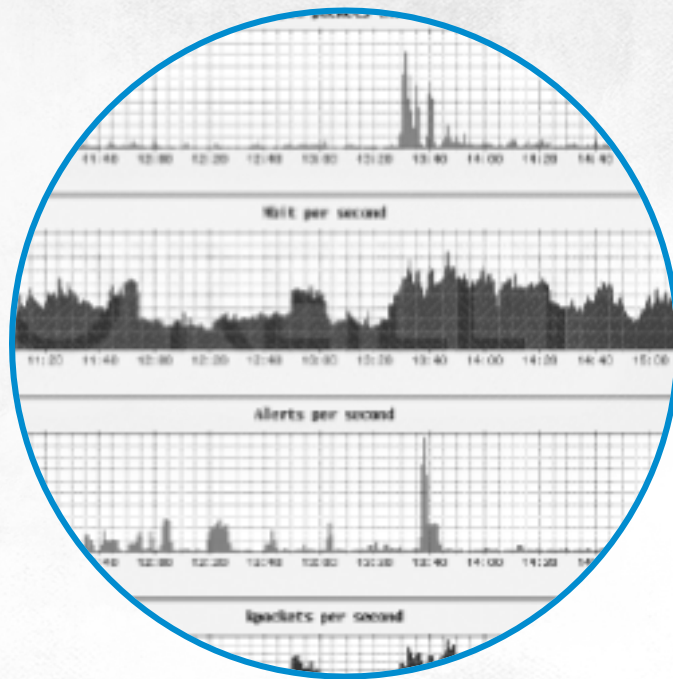




# Network intrusion detection systems

A **Honey Pot** is another IDS type used to collect information about intrusions. It simulates popular services and waits on for attack attempts

Because these systems don't handle authorised requests, all connection attempts must be investigated as suspicious



# Why are firewalls and nids **not enough?**

## Four-way handshake

If you have paid more than ten thousand dollars on a firewall, odds are it's good... or is it? The studies made by BreakingPoint Systems and NSS Labs undermine this belief

**TCP is a session protocol:** all TCP connections start with establishing a session through a three-way handshake, the exchange of SYN, SYN/ACK and ACK packets between the initiator and the target host as defined in RFC 793  
Point 3.3 of RFC 793 contains a good description of establishing a TCP session through exchanging four messages:

- The initiator (client) sends the SYN packet
- The recipient (server) responds with an acknowledgment (ACK)
- The recipient sends the SYN packet
- The initiator acknowledges the session has been established by sending the ACK

**Since steps 2 and 3 may be combined,** all implementations of TCP in fact use the three-way handshake  
The standard however describes it as a four-step procedure

**In theory,** the initiator should accept the ACK packet silently and acknowledge the reception of the SYN packet, which leads to a TCP session being established

# Why are firewalls and nids **not enough**?

## Four-way handshake

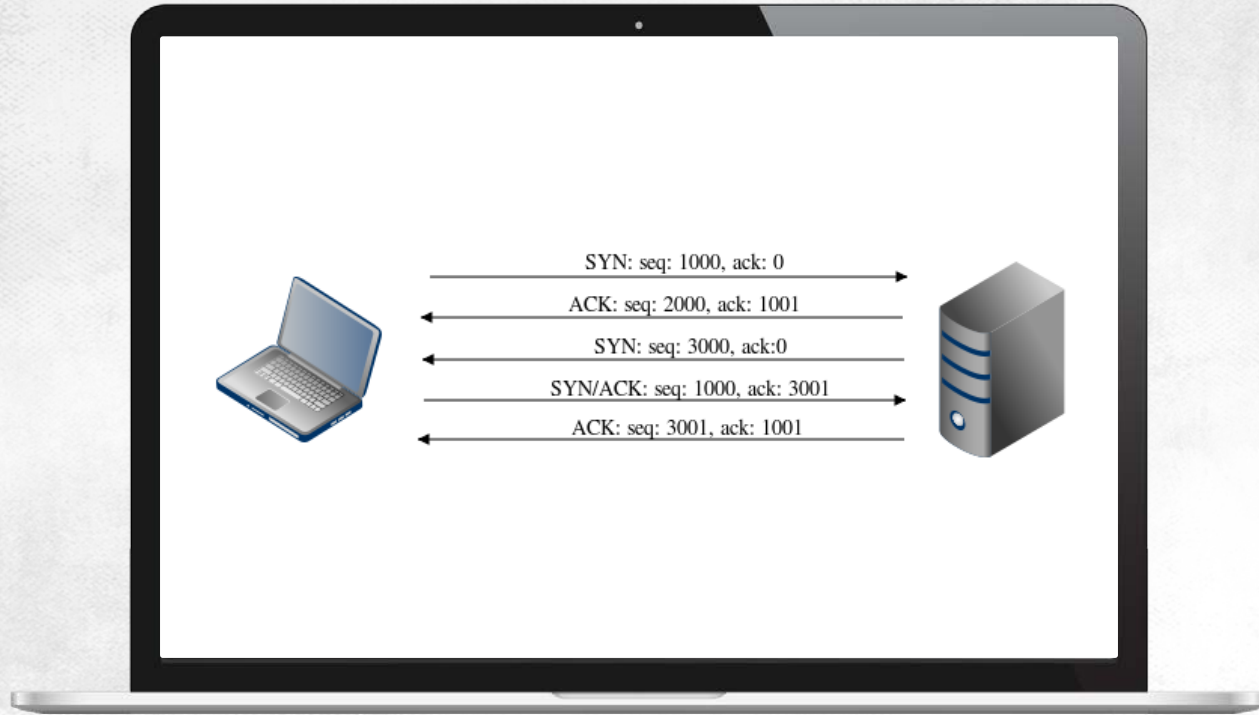
### The practice is different:

- The client sends the SYN with a pseudorandom sequence number
- The server replies with the ACK with an increased acknowledgment number. The packet's sequence number is pseudorandom, but the SYN bit is not set. As expected, the client does not acknowledge the reception of this packet
- Next, the server sends the SYN with a pseudorandom sequence number and a correct acknowledgment number
- Instead of acknowledging the reception of this packet and establishing a TCP session through the ACK message, the client sends a SYN/ACK packet which reuses the previous sequence number and increases the acknowledgement number by one
- The server responds correctly to the reception of the SYN/ACK and sends the ACK acknowledging the TCP session has been established



# Why are firewalls and nids **not enough?**

## Four-way handshake



# Why are firewalls and nids **not enough**?

## Four-way handshake

**Since in this scenario** it is the client, not the server, that sends SYN/ACK packets, the devices that analyse TCP sessions by looking at the headers will assume that it was the server that establishes a session with the client: the direction of the TCP session will be determined wrongly

**The server-sent ACK packet** doesn't establish the session either: even if it is passed on, the session will only be established after the exchange of four messages (SYN, SYN, SYN/ACK and ACK) is completed

# Why are firewalls and nids **not** enough?

## Four-way handshake

The consequences of this prove disastrous for both NIDS systems and firewalls:

- Only one of the three NIDS systems tested by BreakingPoint Systems was able to discover a buffer overflow attack that exploited an ActiveX control during the establishing of a TCP session between a client and web server in the manner described above (exchanging four or five messages). What's more, the one successful system did not analyse TCP session states
- The results firewalls got were even worse. BreakingPoint Systems checked the performance of default firewalls in Windows, Linux and Apple systems. None were able to block the transmissions sent over a reverse TCP session. Nearly a year later NSS Labs tested six professional firewalls to see if they can effectively filter data sent over reverse TCP sessions. It turned out that five out of the six firewalls could not block the attacks despite the fact the same attacks launched over standard TCP sessions were detected and stopped

**IT organizations** worldwide have relied on third-party testing and been misled  
Vik Phatak, CTO, NSS Labs

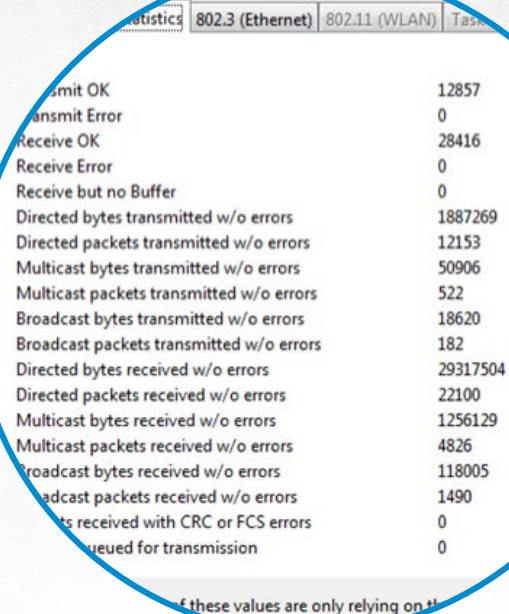


# Wireshark

**Administrators should perform** manual checks and monitor the data transmitted across the networks they manage

**Being able to monitor packets** on your own is necessary to be able to pick the right settings for firewalls and intrusion detection systems

**One of the most popular** network analysers is the free Wireshark



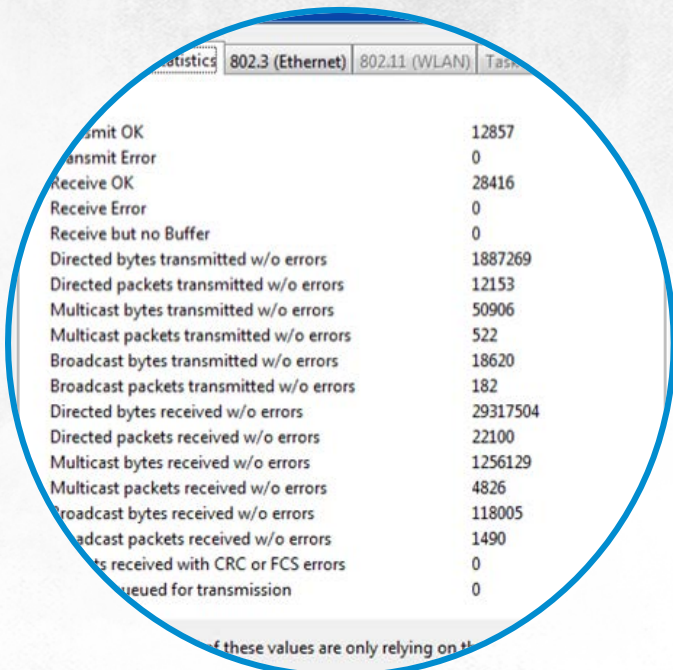
Interface	Statistics
802.3 (Ethernet)	12857
802.11 (WLAN)	0
802.3 (Ethernet)	28416
802.11 (WLAN)	0
802.3 (Ethernet)	0
802.11 (WLAN)	0
802.3 (Ethernet)	1887269
802.11 (WLAN)	12153
802.3 (Ethernet)	50906
802.11 (WLAN)	522
802.3 (Ethernet)	18620
802.11 (WLAN)	182
802.3 (Ethernet)	29317504
802.11 (WLAN)	22100
802.3 (Ethernet)	1256129
802.11 (WLAN)	4826
802.3 (Ethernet)	118005
802.11 (WLAN)	1490
802.3 (Ethernet)	0
802.11 (WLAN)	0

if these values are only relying on the

# Wireshark

Wireshark lets you gather data transmitted over all popular networks, including Ethernet, Bluetooth , Token Ring and ATM

The WinPcap library is used to collect data in Windows, while to capture packets sent over Wi-Fi networks you will need an AirPcap adapter

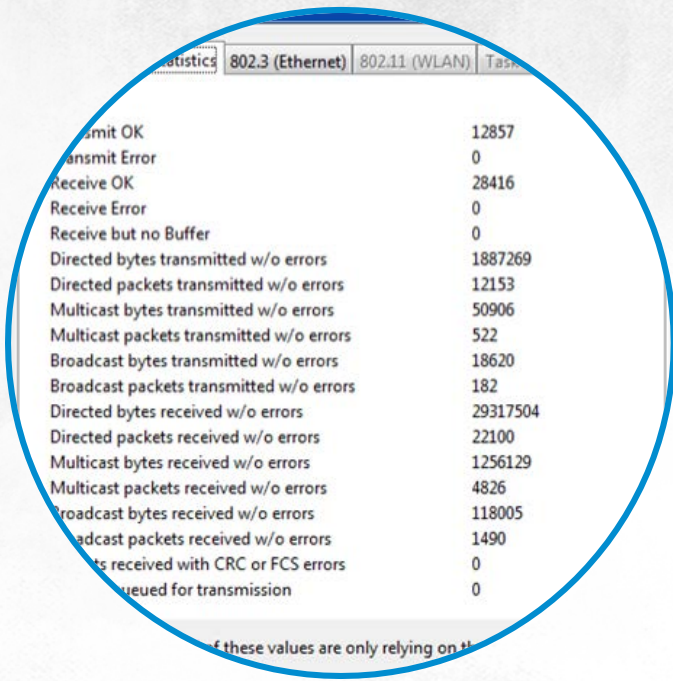


Statistics	802.3 (Ethernet)	802.11 (WLAN)	Task
Transmit OK	12857		
Transmit Error	0		
Receive OK	28416		
Receive Error	0		
Receive but no Buffer	0		
Directed bytes transmitted w/o errors	1887269		
Directed packets transmitted w/o errors	12153		
Multicast bytes transmitted w/o errors	50906		
Multicast packets transmitted w/o errors	522		
Broadcast bytes transmitted w/o errors	18620		
Broadcast packets transmitted w/o errors	182		
Directed bytes received w/o errors	29317504		
Directed packets received w/o errors	22100		
Multicast bytes received w/o errors	1256129		
Multicast packets received w/o errors	4826		
Broadcast bytes received w/o errors	118005		
Broadcast packets received w/o errors	1490		
Packets received with CRC or FCS errors	0		
Packets queued for transmission	0		

if these values are only relying on the

# Wireshark

Wireshark also retrieves the most critical data about all available network interfaces, including a summary report on the data sent through them



Statistics	802.3 (Ethernet)	802.11 (WLAN)	Task
Transmit OK	12857		
Transmit Error	0		
Receive OK	28416		
Receive Error	0		
Receive but no Buffer	0		
Directed bytes transmitted w/o errors	1887269		
Directed packets transmitted w/o errors	12153		
Multicast bytes transmitted w/o errors	50906		
Multicast packets transmitted w/o errors	522		
Broadcast bytes transmitted w/o errors	18620		
Broadcast packets transmitted w/o errors	182		
Directed bytes received w/o errors	29317504		
Directed packets received w/o errors	22100		
Multicast bytes received w/o errors	1256129		
Multicast packets received w/o errors	4826		
Broadcast bytes received w/o errors	118005		
Broadcast packets received w/o errors	1490		
Packets received with CRC or FCS errors	0		
Packets queued for transmission	0		

If these values are only relying on the



# Wireshark

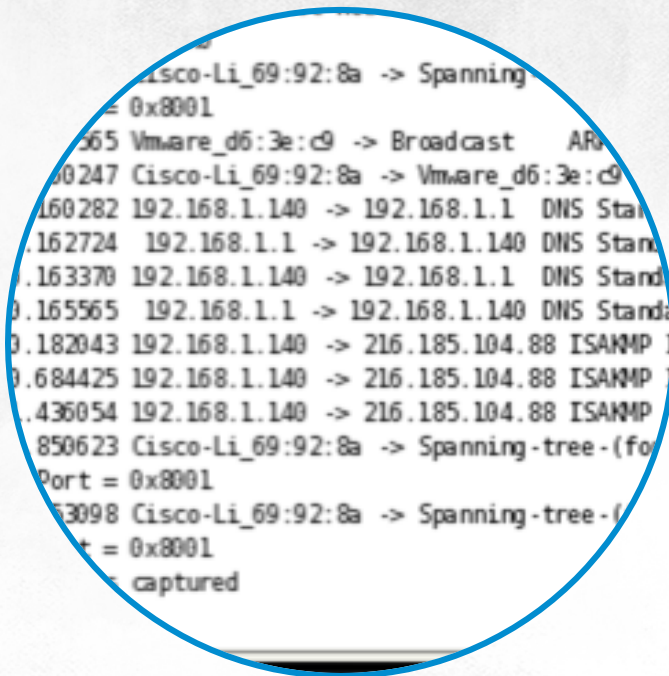
If there is simply too much data

You may have **DROPS** shown in your status bar

If you cannot reduce the amount of collected data using capture filters, try to minimise the number of operations run by the program

In the capture options window:

- Uncheck Update list of packets in real time
- Uncheck Automatic scrolling in live capture
- Uncheck every box in the Name resolution section



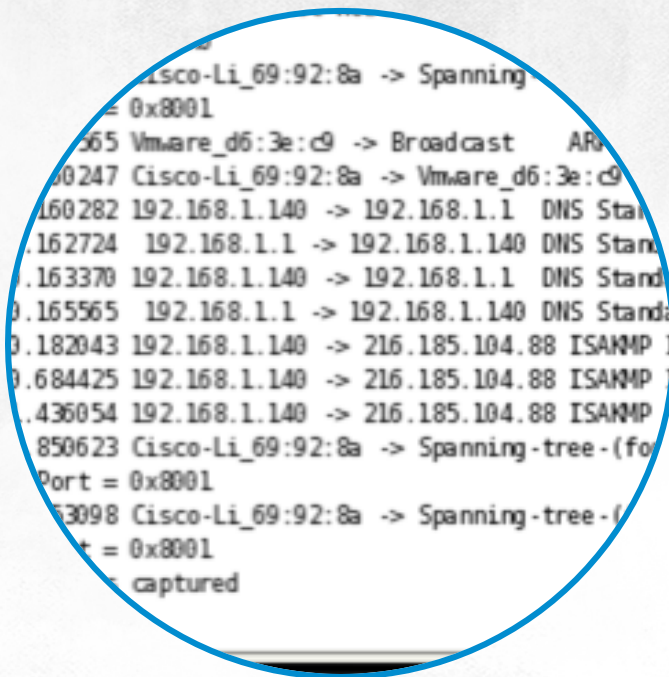
# Wireshark

If there is simply too much data

If packets are still being dropped, select Preferences in the Edit menu and check the Protocols section:

- Select IP protocol and uncheck the Validate the IP checksum if possible and Reassemble fragmented IP datagrams boxes
- Select TCP protocol and uncheck Analyze TCP sequence numbers

If packets are still being dropped, run Wireshark in the command-line interface (the tshark program)



# Wireshark

## If there is simply too much data

### You may have DROPS shown in your status bar

If you cannot reduce the amount of collected data using capture filters, try to minimise the number of operations run by the program

In the capture options window:

- Uncheck Update list of packets in real time
- Uncheck Automatic scrolling in live capture
- Uncheck every box in the Name resolution section

If packets are still being dropped, select Preferences in the Edit menu and check the Protocols section:

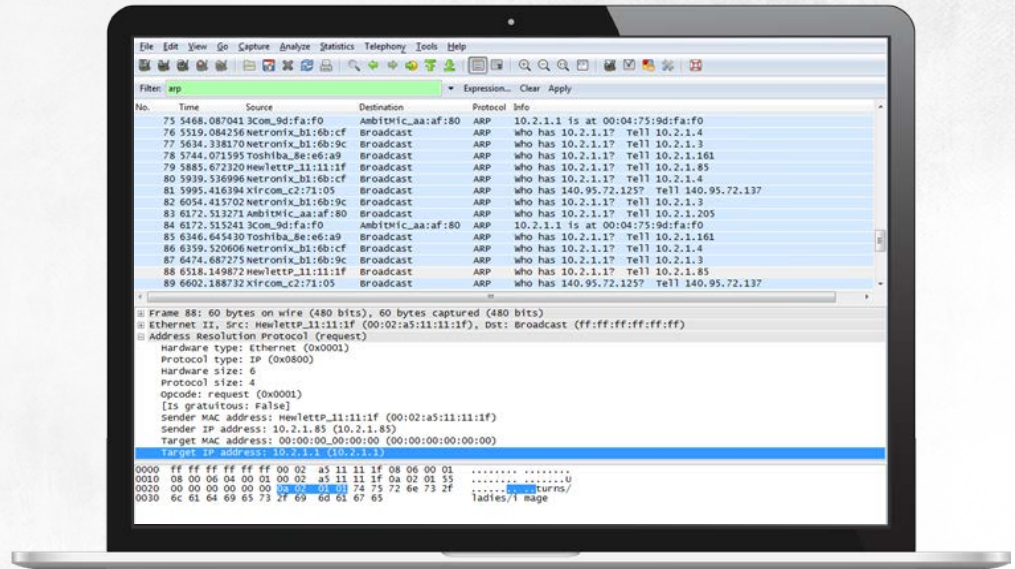
- Select IP protocol and uncheck the Validate the IP checksum if possible and Reassemble fragmented IP datagrams boxes
- Select TCP protocol and uncheck Analyze TCP sequence numbers
- If packets are still being dropped, run Wireshark in the command-line interface (the tshark program)



# Wireshark

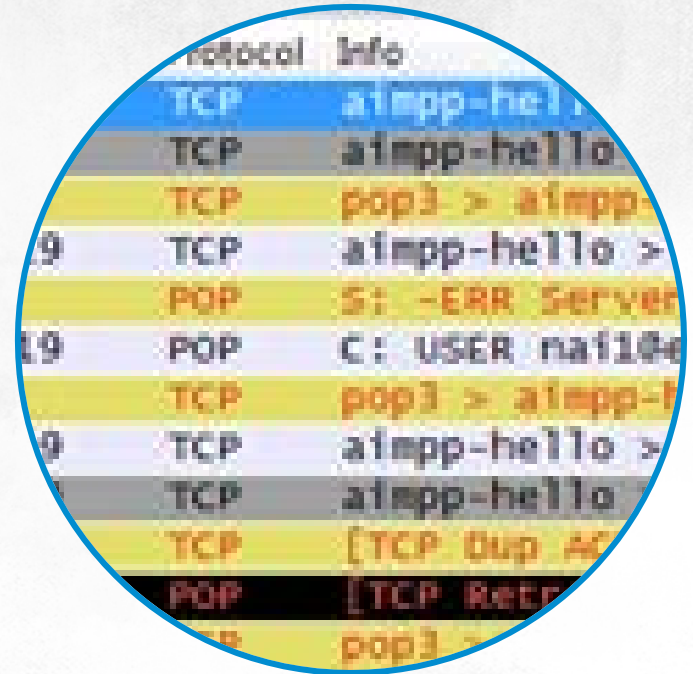
You can make **Wireshark** fit your needs

- Consider adding the time elapsed between the current and the previous packet column
- Apart from capture filters, Wireshark also offers filtering of displayed captured packets
- Display filters is not just sorting packets by protocol or host names: they can sort packets by any field
- Display filters can also be used for defining your own packet colouring rules



# Wireshark

Both the capture and display filters are extremely effective in detecting suspicious and potentially harmful packets

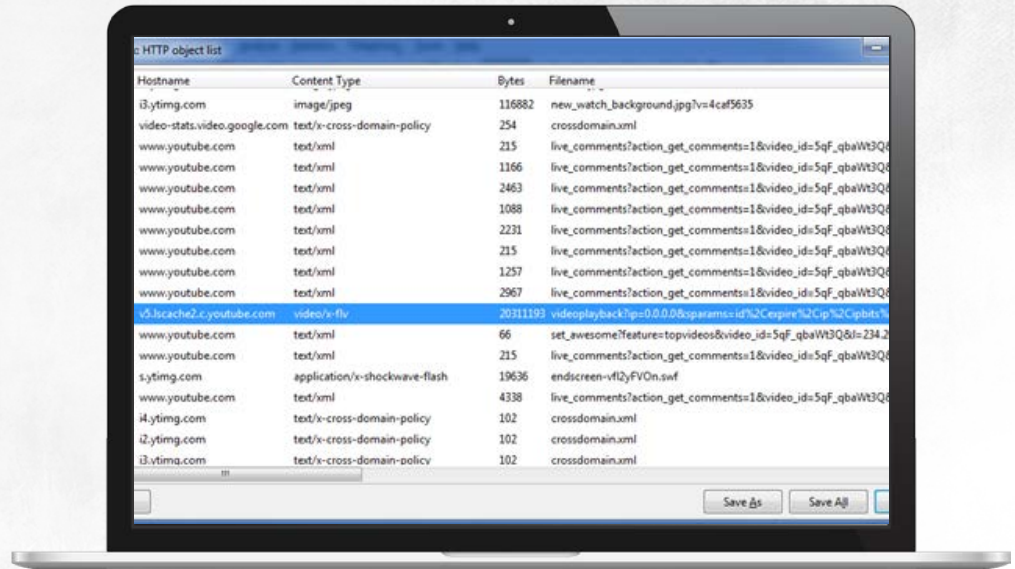


No.	Time	Protocol	Info
		TCP	aimpp-hello
		TCP	aimpp-hello
		TCP	pop3 > aimpp-
19		TCP	aimpp-hello >
		POP	S: -ERR Server
19		POP	C: USER nait10e
		TCP	pop3 > aimpp-
9		TCP	aimpp-hello >
		TCP	aimpp-hello
		TCP	[TCP Dup ACK
		POP	[TCP Retr
		POP	pop3 >

# Wireshark

You can investigate attack attempts and monitor and document user activity using the reassemble feature, which lets you rebuild higher layer protocol packets from a selected TCP session

Flow Graph, IO Graph and TCP Stream Graph are indispensable for identifying hosts that exchange transmissions, for estimating the amounts of data sent and detecting non-standard interactions



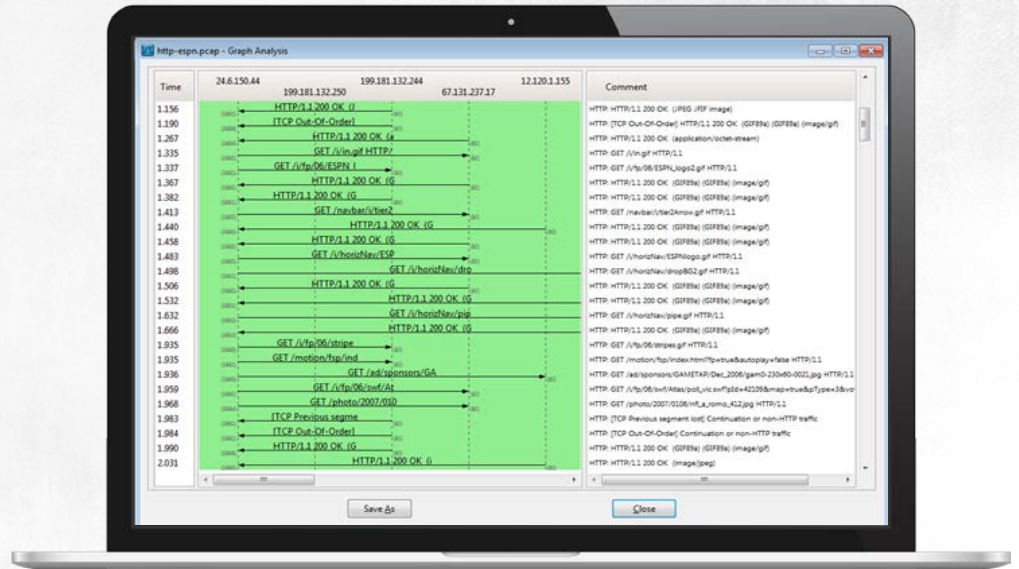


# Wireshark

**Flow Graphs** can also help you detect machines that are riddled with unwanted software

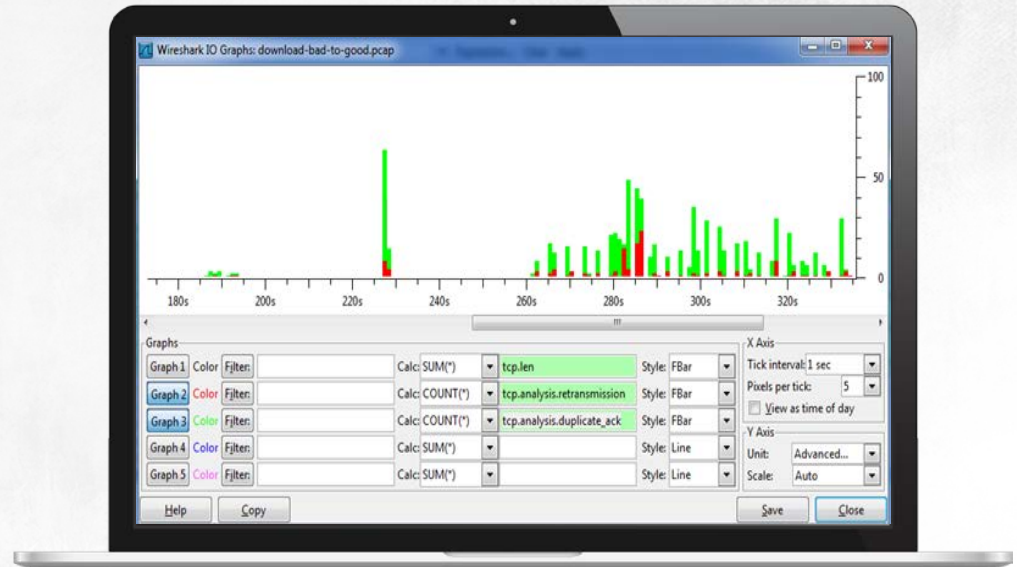
**IO Graphs** are great for analysing the amount and speed of transmissions between hosts

**They can enable you** to filter and highlight the data you present using some standard display filters



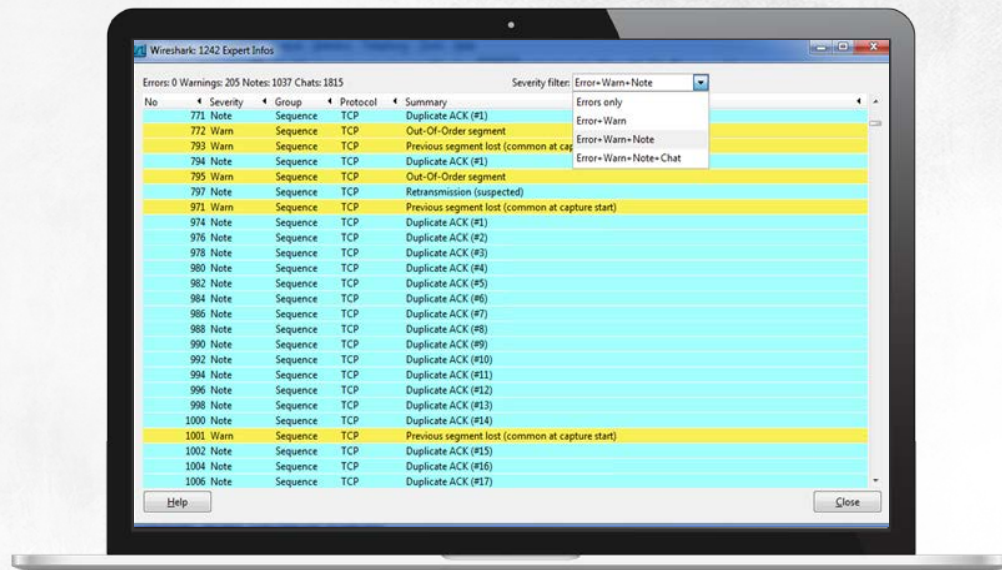
# Wireshark

**Advanced IO Graphs** give you full control over the data in the graph and aggregation methods: for example, they let you sum up and calculate mean values for transmitted packets



# Wireshark

Wireshark can also help you diagnose network problems through its expert functions: two dialogue boxes containing the output of analysing captured packets, with emphasis placed on non-typical or unusual communication patterns

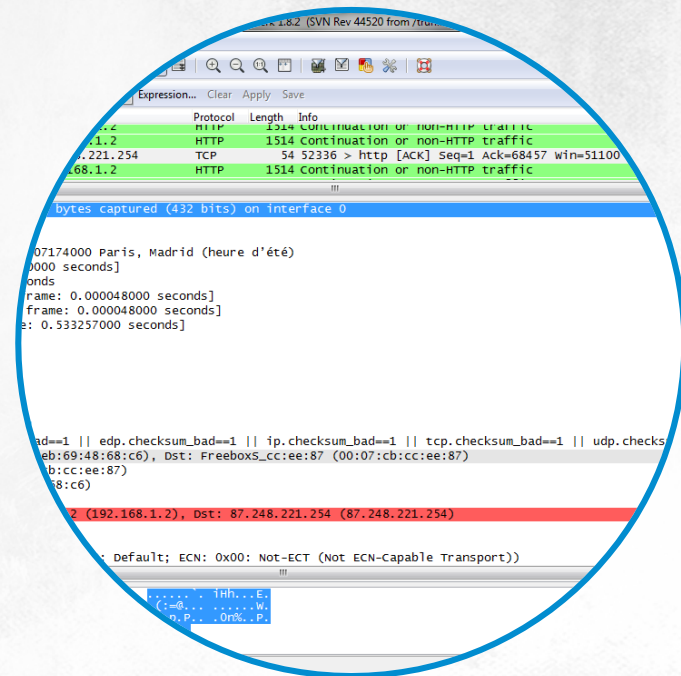




# exercise

## Monitoring traffic using Wiresharke

- FTP server password cracking
- Video files downloaded by users
- ICMP, TCP and UDP scanning
- Protocol scanning (IP scanning)
- Starting infected computers
- Redirecting through web servers
- SQL server password cracking



THANKS

