Defense Techniques

Firewall, Intrusion Detection
Security Testing

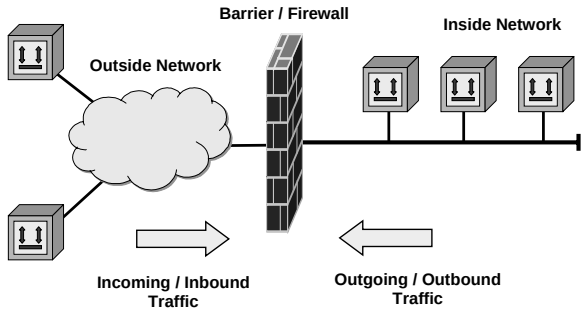
Agenda

- **Firewall Techniques**
  - Introduction
  - Packet-Level FW
  - Stateful Inspection FW
  - Application Level (Proxy) FW
  - Circuit Level FW
  - DMZ
  - Cisco ACL's
- **Intrusion Detection**
- **Security Testing**

Protect or Not Protect Your Network ?

- **Unprotected network**
  - Security have to be implemented on each host
    - Single vulnerable host would violate whole the network security
  - Administrative nightmare
    - In case of a huge number of machines running a mix of various versions of operating systems
- **Therefore protect your internal network**
  - Place a barrier at the borders of trusted, inside network
    - At the so called perimeter towards the Internet
  - Barrier provides a single point of access control
    - Helps with system monitoring and simplifies management
  - Such a barrier is called firewall (FW)

Firewall



Firewall Principles

- **Inside network is trusted**
- **Outside network is potentially malicious**
- **All traffic from inside to outside and vice versa**
  - Must pass through the firewall
- **Only authorized traffic will be allowed to pass**
  - What is authorized is defined by the network security policy of your company
- **The firewall must be well protected**
  - Immune to any kind of penetration
  - FW based on a trusted system with a secure operating system

Firewall Limitations and Types

- **Principle problems with any kind of FW**
  - If you want access from the outside you must let traffic in or you want access to the outside you must let traffic out
    - Open certain TCP/UDP ports, trust certain IP addresses
    - Malicious / unwanted traffic may disguise behind allowed traffic
  - You must trust your internal network
    - FW cannot project against internal threats
  - If the single entry point of FW is bypassed by any dial-in facilities (RAS) the firewall cannot provide protection
- **Different types**
  - Packet Level FW (Stateless)
  - Stateful Inspection FW
  - Application Level / Proxy FW
  - Circuit Level FW (e.g. SOCKS)

Agenda

- **Firewall Techniques**
  - Introduction
  - Packet-Level FW
  - Stateful Inspection FW
  - Application Level (Proxy) FW
  - Circuit Level FW
  - DMZ
  - Cisco ACL's
- **Intrusion Detection**
- **Security Testing**

Packet Level Firewall

- **Static packet filtering based on filter rules**
  - Decision what can pass the barrier is based on certain header fields of intercepted packets
    - MAC header (ether-type, source MAC address, destination MAC address)
    - IP header (source IP address, destination IP address, protocol type)
    - ICMP header (code type)
    - TCP header (source port, destination port, flags (SYN, ACK))
    - UDP header (source port, destination port)
- **Typically available on L3 routers (L2 switches), but nowadays also on Linux, Windows**
  - E.g. Cisco's famous access control lists (ACL)
  - E.g. iptables, ipchains, Windows XP SP2

Packet Level Firewall Usage 1

- **Can secure inside hosts against**
  - Unwanted traffic, simple attacks and certain DoS attacks
    - E.g. ICMP echo request, ICMP redirect request, ICMP unreachable, not supported UDP/TCP ports, IP source routing, SYN flooding
- **Can limit services inside hosts can get from the outside**
- **Can secure against IP spoofing**
  - Source address of inbound traffic is checked against inside used IP addresses -> if yes then traffic is blocked
    - Prevention technique
  - Source address of outbound traffic is checked if it contains inside used IP addresses -> if no then traffic is blocked
    - Be a good Internet citizen

Packet Level Firewall Usage 2

- **Can filter IP private address range**
  - 10.0.0.0, 172.16.0.0 - 172.31.255.255, 192.168.X.X
- **Can filter IP loopback address**
  - 127.X.X.X
- **Can filter IP multicast address range**
- **Can filter IP experimental address range**
- **Can filter APIPA**
  - (automatic private IP address) 169.254.x.x…used by Microsoft
- **All these addresses**
  - should not appear from the outside in the source address of a datagram
- **Can block incoming TCP connections**
  - Usual method: No TCP SYN flag allowed in inbound packets and hence avoid three-way handshake of TCP connection establishment from the outside network
    - E.g. Cisco's "established" keyword in ACL

Packet Level Firewall Limitation 1

- **Most network communication responses are stimulated by requests**
  - So we have to let in the responses in order to communicate
  - But forged packets which look like harmless responses are still let in
- **In principle all packets which match the filter rule and are allowed will pass**
  - Malicious packets may hide behind allowed TCP/UDP ports
- **Very strict filter rules**
  - May be an administrative nightmare and tend to be complex

Packet Level Firewall Limitation 2

- **Filter rules must often allow more than what is be necessary for a certain communication**
  - E.g. inside client want access to outside servers
    - Think about the TCP client port range
    - Often all TCP destination ports and all IP source addresses must be passed through to let TCP replies from servers in (scenario ##)
  - Special attention to Cisco's "established" keyword in ACL
    - There is no state management of a TCP session
    - Just inbound TCP segments with SYN (only) set are blocked
    - Inbound TCP segments with ACK set are let in (especially a problem with above mentioned case of scenario ##

Packet Level Firewall Limitation 3

- **Ports are open permanently to allow inbound traffic**
  - Security vulnerability
  - Not adequate with certain applications which dynamically negotiated port numbers
- **IP Fragmentation**
  - Can't check TCP/UDP ports in a fragmented IP datagram
  - So the decision is
    - Either drop such the packet or let it through
  - If you let it in then together a vulnerability problem may arise (TCP overwrite attack)
    - The first fragment will fulfill all rules but the second fragment uses on offset to override fields (e.g. TCP ports) of the first fragment
    - Fragments are reassembled at the destination host and hence a malicious packet will reached the host at a port which is not permitted to pass the firewall

Packet Level Firewall Limitation 4

- **Filtering UDP segments is a problem**
  - Because of stateless behavior of UDP requests/replies
  - So very often the decision on a packet level FW is to block UDP traffic generally
- **Some services can't be filtered at all**
  - Think about IPSec encrypted traffic
  - Check of TCP/UDP ports in encrypted payload of an IP datagram is not even possible at the firewall
- **Simple NAT**
  - May provide similar security for certain scenarios

Institute of Computer Technology - Vienna University of Technology

L91C - Defense Techniques (FW, IDS, IPS)

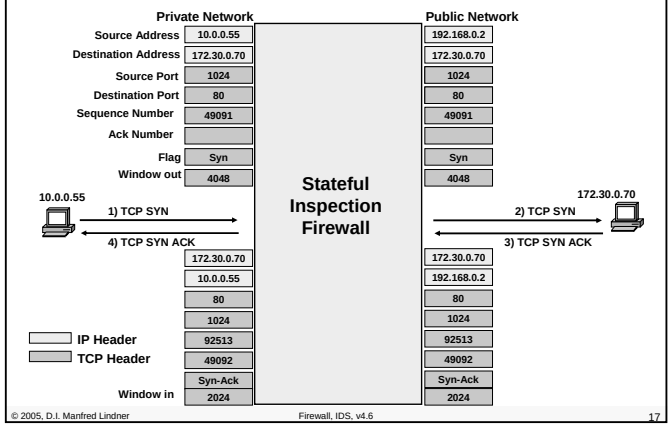
Agenda

- **Firewall Techniques**
  - Introduction
  - Packet-Level FW
  - Stateful Inspection FW
  - Application Level (Proxy) FW
  - Circuit Level FW
  - DMZ
  - Cisco ACL's
- **Intrusion Detection**
- **Security Testing**

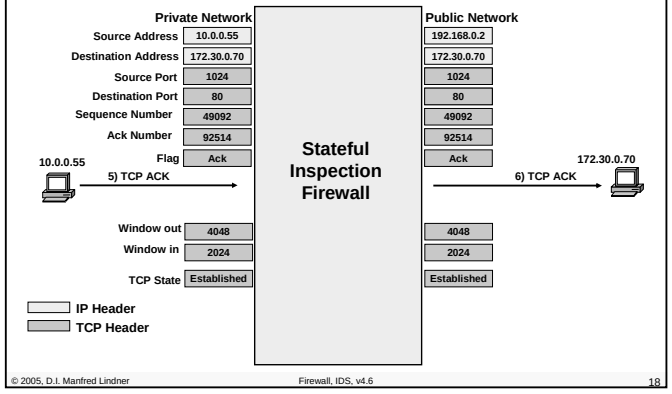
Stateful (Inspection) Firewall

- **Stateful inspection**
  - Provides state management additionally to basic function of a packet level firewall
  - Remembers (the initiating) outbound traffic so only valid responses are let in
  - Creates filter rules (or better exceptions) on demand
    - Dynamic ACL's are used
  - Actually monitors the TCP connections and records the important TCP state values in a table
  - Checks if all TCP fields are in the expected range
    - Sequence number
    - Acknowledgement number
    - Window field
- **Mandatory part of "real" firewall boxes**
  - Like Checkpoint's FW1 or Cisco's PIX

TCP State Management with NAT (Establishment by SYN – SYN ACK – ACK sequence) 1



TCP State Management with NAT (Establishment by SYN – SYN ACK – ACK sequence) 2



Stateful (Inspection) Firewall Usage

- **Far better protection then normal stateless packet level firewall**
  - E.g. against TCP hijacking
- **More effective against DoS attacks**
  - E.g. by limiting the number of half-opened TCP sessions to a certain amount based on time / per host
- **May do some DoS prevention**
  - By sending RST TCP messages to endpoints of half-opened TCP sessions to free up resources of a waiting server
- **May generate alarms**
  - When certain incidents happen
  - When certain thresholds are reached

Stateful (Inspection) Firewall on Routers

- **Trend to be become part of "normal" network components like routers**
  - E.g. with Cisco's IOS "reflexive" ACL feature
  - E.g. with Cisco's IOS "Firewall Feature" set and CBAC (Context Base Access Control list)
  - Remark:
    - CABC makes stateful inspection not only for TCP/UDP but can additionally check network application communication for valid commands
    - E.g. valid FTP, SMTP, RPC
    - E.g. HTTP Java Applet Blocking
    - E.g. SIP, H.323, RTSP
  - With such features a normal router can operate even up to the network application level

Stateful (Inspection) Firewall

- **Less problems with stateless UDP traffic**
  - Because such a FW can remember initiator and therefore can check corresponding fields (e.g. port numbers) in replies
  - Usually done within certain time limits after initial UDP request was sent
  - Note:
    - even with this UDP leaves some security problems like UDP spoofing, UDP hijacking
- **A real good stateful inspection firewall**
  - Operates on higher performance level than packet level firewalls or proxy firewalls
  - Provides failover techniques
    - Hot standby with uninterrupted operation that means standby FW knows the context / states of the active FW

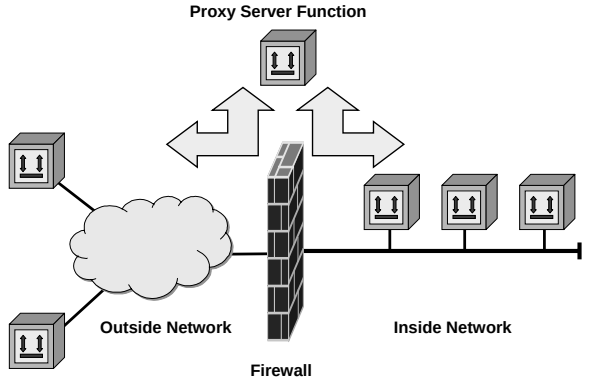
Stateful (Inspection) Firewall Limitation

- **Exposure of inside IP address**
  - Stateful inspection firewall does not change the IP address from inbound to outbound
- **Still we trust some IP addresses**
  - But who is the actual user?
  - Therefore we would need a kind of authentication system
- **Therefore some firewall implemenations combine**
  - Proxy authentication and stateful FW together
  - NAT (PAT) and stateful FW together

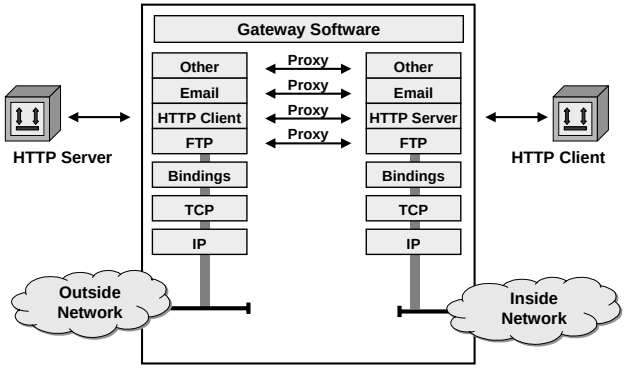
Agenda

- **Firewall Techniques**
  - Introduction
  - Packet-Level FW
  - Stateful Inspection FW
  - Application Level (Proxy) FW
  - Circuit Level FW
  - DMZ
  - Cisco ACL's
- **Intrusion Detection**
- **Security Testing**

Application Level / Proxy Firewall



Application Level Firewall



Application Level Firewall

- **Inside client request are directed to a proxy function**
  - Which open and maintain communication to the requested outside server on behalf of the inside client
    - User authentication and authorization may be checked
      - Can be done for both directions (inbound, outbound)
    - Session state information is maintained
    - Can do caching of information replies (e.g. HTTP proxy)
- **Proxy appears**
  - As endpoint for a certain application from both inside and outside
- **Some Problems:**
  - Relatively slow under full load
  - Support of new applications must be installed at the proxy (may be difficult under operation)
  - Single point of failure

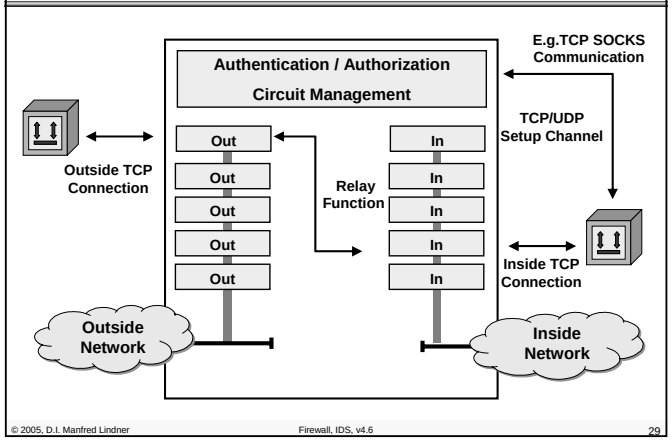
Agenda

- **Firewall Techniques**
  - Introduction
  - Packet-Level FW
  - Stateful Inspection FW
  - Application Level (Proxy) FW
  - Circuit Level FW
  - DMZ
  - Cisco ACL's
- **Intrusion Detection**
- **Security Testing**

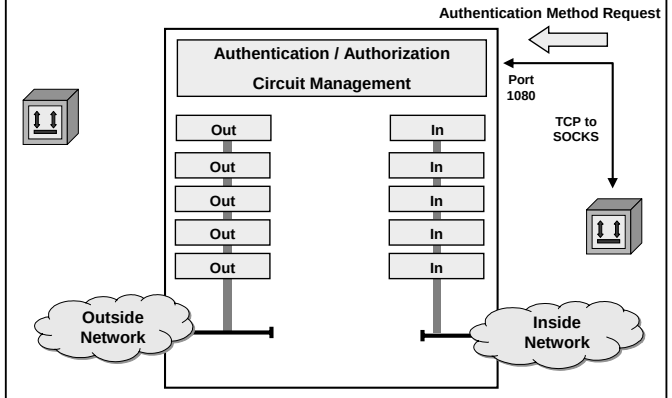
Circuit Level Firewall

- **Avoids complexity of application level FW**
  - A new application means a new proxy function in such a application level FW
- **Method:**
  - Just act as connection relay function on TCP or UDP layer but not on the application layer
  - “Shim layer” between application and transport layer
  - TCP/UDP connection request is signaled via out-band channel and after authentication is successfully performed the appropriate TCP/UDP connections (two) are established
  - After that TCP/UDP segments are just relayed without any examination of content
- **Example:**
  - SOCKS (RFC 1928, 1929 (Auth U/PW), 1961 (Auth GSS-API), 1508/1509 (GSS-API obs.), 2743/2744 (GSS-API act.)
  - usually port 1080 is used to reach the SOCKS server for the TCP/UDP setup channel

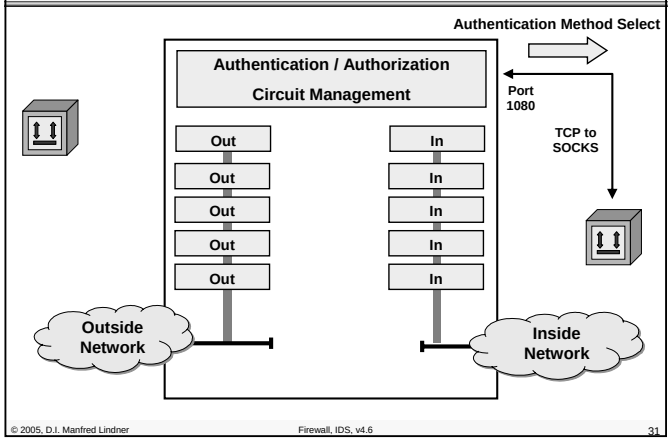
Circuit Level Firewall



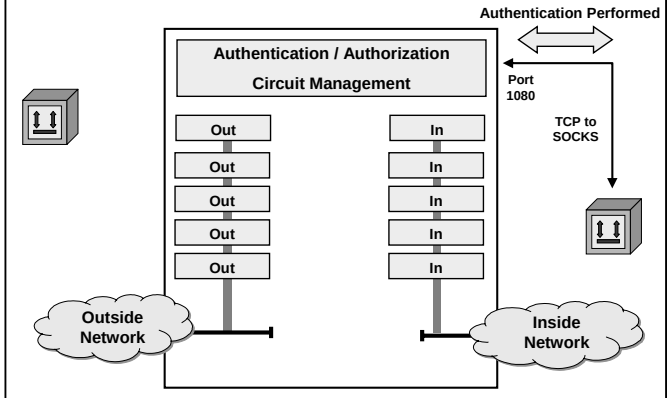
Circuit Level Firewall with SOCKS 1



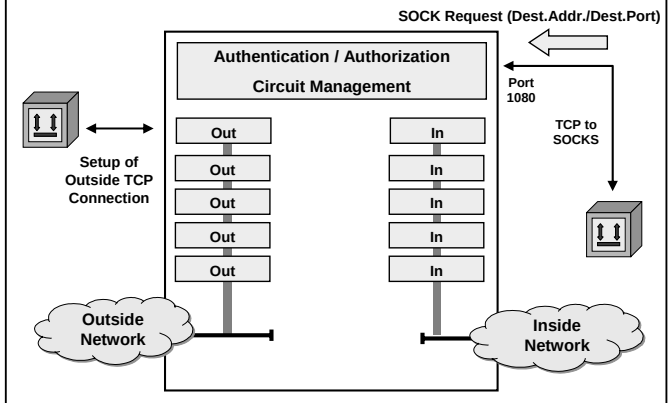
Circuit Level Firewall with SOCKS 2



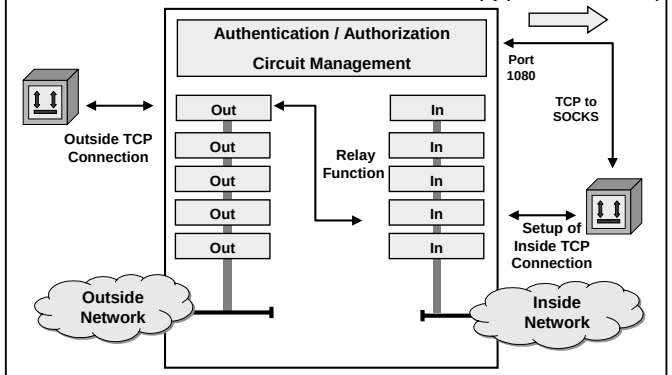
Circuit Level Firewall with SOCKS 3



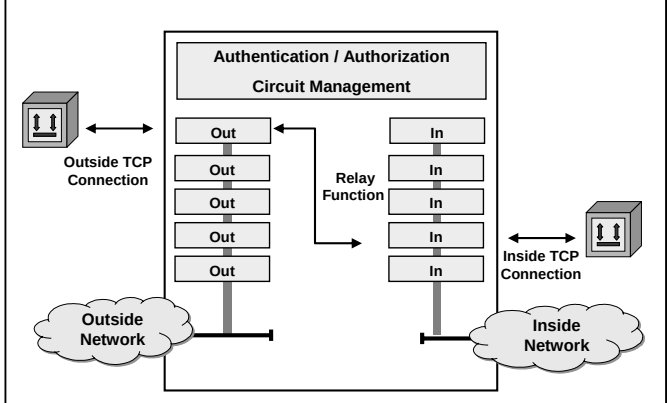
Circuit Level Firewall with SOCKS 4



Circuit Level Firewall with SOCKS 5



Circuit Level Firewall with SOCKS 6



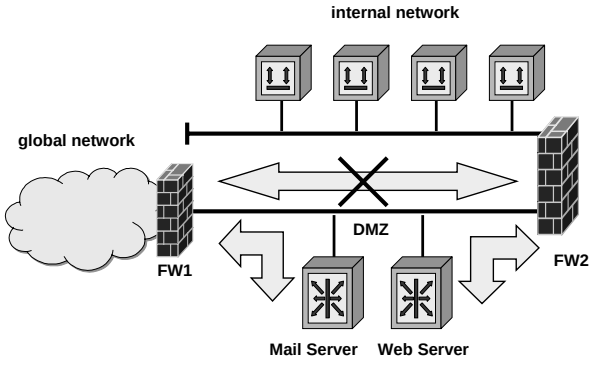
Agenda

- **Firewall Techniques**
  - Introduction
  - Packet-Level FW
  - Stateful Inspection FW
  - Application Level (Proxy) FW
  - Circuit Level FW
  - DMZ
  - Cisco ACL's
- **Intrusion Detection**
- **Security Testing**

DMZ

- **DMZ – De-Militarized Zone**
  - Network area between two packet filter firewalls
    - One firewall FW1 only allows traffic from outside to the servers (sometimes called "**bastion hosts**") and vice versa
      - F1 refuses anything to forward from the global network unless the destination address is your bastion host and refuses anything to forward to the global network unless the source address is the bastion host
    - Second firewall FW2 only allows traffic from inside to the servers and vice versa
      - F2 refuses anything to forward from your internal network unless the destination address is your bastion host and refuses anything to forward to your internal network unless the source address is the bastion host
  - Separates external and internal network
  - Contains hosts that provide
    - External services (e. g. web server, DNS) and
    - Application gateways for internal clients
  - When bastions hosts are compromised
    - Traffic of internal network cannot be sniffed
    - Protection by firewall F2

DMZ



Agenda

- **Firewall Techniques**
  - Introduction
  - Packet-Level FW
  - Stateful Inspection FW
  - Application Level (Proxy) FW
  - Circuit Level FW
  - DMZ
  - Cisco ACL's
- **Intrusion Detection**
- **Security Testing**

Access Lists on Cisco Routers

- **Basic IOS feature in all router**
  - Stateless (static ACL)
    - Standard and extended access control lists (ACL's)
      - For IP, IPX, AppleTalk, MAC
      - Numbered
      - Named (since IOS 11.2)
  - Stateful (dynamic ACL)
    - Reflexive ACL's (since IOS 11.3)
- **IOS routers with "FW Feature Set"**
  - Stateless/stateful like in basic IOS feature
  - Additionally more powerful stateful filtering
    - Context Based Access Control (CBAC)

IP Access List Types

- **Standard**
  - Filtering based on the source address of a IP datagram only
- **Extended**
  - Filtering based on several attributes
    - IP protocol type (e.g. tcp, udp, icmp, ospf, eigrp, …)
    - IP source and IP destination address
    - TCP/UDP source and TCP/UDP destination port
    - ICMP and ICMP message type
    - ….

Numbered Access Lists

- **1 – 99 (IP standard)**
- **100 – 199 (IP extended)**
- **1300 – 1999 (IP standard second range)**
- **2000 – 2699 (IP extended second range)**
- **700 – 799 (MAC address filtering)**
- **1100 – 1199 (MAC address extended filtering)**

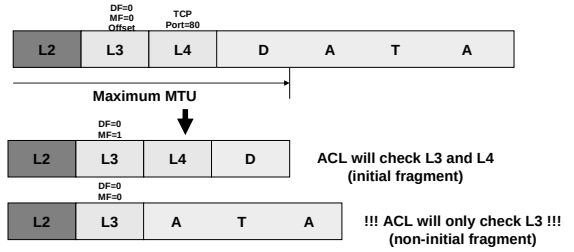
IP Standard ACL

- **Command syntax:**
  - access-list ***access-list-number*** **{**deny **|** permit**}** ***source*** **[*source-wildcard*] [**log**]**
- **Command parameters**:
  - ***access-list-number***
    - **Number of an access list. This is a decimal number from 1 to 99 or from 1300 to 1999.**
  - deny
    - **Denies access if the conditions are matched.**
  - permit
    - **Permits access if the conditions are matched.**
  - ***source***
    - **Number of the network or host from which the packet is being sent. There are two alternative ways to specify the source:**
      - **Use a 32-bit quantity in four-part, dotted-decimal format.**
      - **Use the** any **keyword as an abbreviation for a *source* and *source-wildcard* of 0.0.0.0 255.255.255.255.**
  - ***source-wildcard*** (optional)
    - **Wildcard bits to be applied to the source (bit = 0 means exactly match, bit = 1 means don't care), if omitted it always means 0.0.0.0 = exactly match**
  - log (optional)
    - **Causes an informational logging message about the packet that matches the entry to be sent to the console.**

IP Extended ACL

- **Generic command syntax:**
  - access-list ***access-list-number*** {deny | permit} ***protocol source source-wildcard destination destination-wildcard*** [log] [fragments]
- **Command parameters**:
  - ***protocol***
    - **Name or number of an Internet protocol. It can be one of the keywords** eigrp, gre, icmp, igmp, igrp, ip, ipinip, nos, ospf, pim, tcp, **or** udp**, or an integer in the range from 0 to 255 representing an Internet protocol number. To match any Internet protocol (including ICMP, TCP and UDP) use the** ip **keyword,**
  - ***source* (same for *destination*)**
    - **Number of the network or host from which the packet is being sent or received. There are three alternative ways to specify it:**
      - **Use a 32-bit quantity in four-part, dotted-decimal format.**
      - **Use the keyword** host ***source* as an abbreviation of *source source-wildcard* with wildcard-mask = 0.0.0.0 (means exactly match)**
      - **Use the** any **keyword as an abbreviation for a *source* and *source-wildcard* of 0.0.0.0 255.255.255.255.**
  - fragments (optional)
    - **The access list entry applies to non-initial fragments (L3 information only) of packets; the fragment is either permitted or denied accordingly.**

ACL's and IP Fragments



- **If 2nd packet has manipulated offset, then 2nd packet could overwrite first one (depends on end system)**
- **ACL without fragment keyword will not check non-initial fragments, will check only initial fragment according to L3/L4**
- **If ACL has `fragment` keyword**
  - Then ACL-entry is applied only to non-initial fragments
  - Cannot be specified for ACL entries that check L4 information

IP Extended ACL (cont.)

- **Command syntax TCP/UDP**:
  - access-list ***access-list-number*** **{**deny **|** permit**}** tcp | udp ***source source-wildcard*** **[***operator*** [***port***]] ***destination destination-wildcard*** **[***operator*** [***port***]] [**established**] [**log**] [**fragments**]**
- **Command parameters**:
  - ***operator*** (optional)
    - **Compares source or destination ports. Possible operands include** lt **(less than),** gt **(greater than),** eq **(equal),** neq **(not equal), and** range **(inclusive range). If the operator is positioned after the *source* and *source-wildcard*, it must match the source port. If the operator is positioned after the *destination* and *destination-wildcard*, it must match the destination port. The** range **operator requires two port numbers. All other operators require one port number.**
  - ***port*** (optional)
    - **The decimal number or name of a TCP or UDP port. A port number is a number from 0 to 65535. TCP and UDP port names are listed**
  - established (optional)
    - **For the TCP protocol only: Indicates an established connection. A match occurs if the TCP datagram has the ACK, FIN, PSH, RST, SYN or URG control bits set. The non-matching case is that of the initial TCP datagram to form a connection.**

IP Extended ACL (cont.)

- **Command syntax ICMP**:
  - access-list ***access-list-number*** {deny | permit} icmp ***source source-wildcard destination destination-wildcard*** [***icmp-type*** **[***icmp-code***] | ***icmp-message***] [log] [fragments]
- **Command parameters**:
  - ***icmp-type*** (optional)
    - **ICMP packets can be filtered by ICMP message type. The type**
    - **is a number from 0 to 255**.
  - ***icmp-code*** (optional)
    - **ICMP packets that are filtered by ICMP message type can also be filtered by the ICMP message code. The code is a number from 0 to 255.**
  - ***icmp-message*** (optional)
    - **ICMP packets can be filtered by an ICMP message type name or ICMP message type and code name**.

Applying an ACL on an interface

- **Command syntax**:
  - ip access-group **{*access-list-number* | *access-list-name*}** {in | out}
- **Command parameters**:
  - ***access-list-number***
    - **Number of an access list. This is a decimal number from 1 to 199 or from 1300 to 2699.**
  - ***access-list-name***
    - **Name of an IP access list as specified by an ip access-list command.**
  - in
    - **Filters on inbound packets.**
  - out
    - Filters on outbound packets.

Usage Conventions of Access Lists

- **Creation:**
  - When creating numbered ACL's, new lines are appended to the end of the current list
    - Modification requires deleting the list and re-entering the entire list
    - Exception: if you want to add something to the current last entry
  - Common TCP/UDP services and ICMP types are known by name
    - Values such as telnet, ftp/ftp-data and icmp-echo are valid. TCP/UDP ports can be a numeric port range rather than just a single value.
  - There is an implicit "deny any" at the end of the list !!!
- **Processing:**
  - ACL's are processed in top down order
    - When a match is found (either permit or deny), processing ends. Therefore the sequence of ACL entries is important.

Example 1 for an IP Standard ACL

- **Define access-list:**
  - rx(config)# access-list 1 permit 193.212.14.0 0.0.0.255
  - rx(config)# access-list 1 permit 68.23.15.1 0.0.0.0
    - That is the same as saying "access-list 1 permit host 68.23.15.1"
  - rx(config)# access-list 1 deny any log
  - Implicit "deny any" not seen at the end of the configured list but is there

    access-list 1 deny any
- **Be careful**
  - In the moment you apply an access-list to an physical interface the router starts filtering
    - If you have specified only deny statements so far then all traffic will be blocked on that interface. Maybe you are cut off while configuring the router from a remote-site e.g. via telnet
- **ACL configuration verification:**
  - rx# show ip access-list 1

Example 1 for an IP Standard ACL

- **Apply access list to an interface:**
  - rx(config)# interface Ethernet 0
  - rx(config-if)# ip access-group 1 in (in for inbound direction)
  - rx(config)# interface Serial 0
  - rx(config-if)# ip access-group 2 out (out for outbound direction)
- **Inbound ACL's are applied on ALL packets, also those destined for the router**
- **Outbound ACL's are applied on packets that go through the router**

Example 2 for an IP Standard ACL

- **Definition of the access-list:**
  - rx(config)# access-list 10 deny 10.0.0.0 0.255.255.255
  - rx(config)# access-list 10 deny 172.16.0.0 0.15.255.255
  - rx(config)# access-list 10 deny 192.168.0.0 0.0.255.255
  - rx(config)# access-list 10 deny 224.0.0.0 31.255.255.255 log
  - rx(config)# access-list 10 deny 127.0.0.0 0.255.255.255 log
  - rx(config)# access-list 10 remark "IP Spoofing Prevention of own addresses"
  - rx(config)# access-list 10 deny 128.15.0.0 0.255.255 log
  - rx(config)# access-list 10 permit any
  - Implicit "deny any" not seen at the end of the configured list but is there

    access-list 10 deny any
  - In this special case the end of the list will be never reached

Example 3 for an IP Extended ACL

- **Definition of the access-list:**
  - rx(config)# remark “IP Spoofing Prevention of own addresses”
  - rx(config)# access-list 101 deny ip 128.15.0.0 0.255.255 any
  - rx(config)# access-list 101 deny tcp any any range 135 139
  - rx(config)# access-list 101 deny udp any any range 135 139
  - rx(config)# access-list 101 deny tcp any any range 445
  - rx(config)# access-list 101 deny udp any any range 69 log
  - rx(config)# access-list 101 deny udp any any range 161 162 log
  - rx(config)# access-list 101 deny icmp any any host-redirect echo
  - rx(config)# access-list 101 permit any any
  - rx(config)# interface Serial 0
  - rx(config-if)# ip access-group 101 in

Example 4 for an IP Extended ACL

- **How to use the fragment keyword in extended ACL?**
  - rx(config)# access-list 101 deny ip any host 1.1.1.1 fragments
  - rx(config)# access-list 101 permit tcp any host 1.1.1.1 eq 80
  - rx(config)# access-list 101 deny ip any any
  - The first statement will match and deny only non-initial fragments destined for host 1.1.1.1.
  - The second statement will match and permit only the remaining non-fragmented and initial fragments that are destined for host 1.1.1.1 TCP port 80.
  - The third statement will deny all other traffic.
  - Note:
    - If we want to block non-initial fragments for any TCP port 80 of host 1.1.1.1, we must block non-initial fragments for all TCP ports of this host

Named Access Lists 1

- **Avoids numbering approach**
  - Deletion is possible
  - Using indexes, arbitrary changes are possible
- **Example:**
  - rx(config)# ip access-list standard ***MyFirstRule***
  - rx(config-std-nacl)# permit 193.212.14.0 0.0.0.255
  - rx(config-std-nacl)# deny any log
  - rx(config-std-nacl)# exit
  - rx(config)# interface Ethernet 0
  - rx(config-if)# ip access-group ***MyFirstRule*** in
- **ACL configuration verification:**
  - rx# show ip access-list ***MyFirstRule***
  - You will see line numbers (10, 20, 30, ….) which are automatically be inserted by IOS

Named Access Lists 2

- **To insert new entries**
  - Simply specify a new sequence number first and enter a normal entry (in the config-std-nacl mode or in the config-ext-nacl mode)
  - rx(config-std-nacl)# 25 permit 195.210.23.0 0.0.0.255
- **To delete entry number 20**
  - rx(config-std-nacl)# no 20
- **To resequence**
  - rx(config)# ip access-list resequence ***MyFirstRule*** 20 10
    - Starting by line 20 and proceeding with numbering by 10
  - Issue this command after inserting several new entries in order to resequence each line for further insertions

Other Enhanced Access Lists

- **Dynamic ("Lock and Key")**
  - Create specific temporary openings (dynamic ACL's) in response to a successful user authentication
  - Superseded by FW-Feature-Set "Proxy Authentication"
- **Time-based**
  - Conventional numbered or named ACL activated at a certain data/time and released at another specified date/time
- **Reflexive** (since IOS 11.3)
  - Create dynamic entries
  - Replacement for "established" keyword of conventional numbered or named ACL
  - Superseded by FW-Feature-Set "CABC"
- **Context Based Access List (CBAC)**

Reflexive Access Lists

- **Reflexive**
  - Creates state information of sessions and controls them
  - Stateful inspection for TCP sessions
  - This allows you to control TCP connections arriving on the un-trusted side of your router when the TCP connection was initiated from the trusted side of your router
  - Works basically with two extended access lists for outgoing and incoming traffic
    - New keyword reflect (for outgoing)
    - Specifying of a TCP state-table by a name (for outgoing)
    - Specifying evaluation (=monitoring) of appropriate state table (for incoming
  - Applying of the two filters at the corresponding interface

Reflexive Access Lists

- **Example**
  - rx(config)# ip access-list extended ***Filterout***
  - rx(config-ext-nacl)# permit tcp 193.212.14.0 0.0.0.255 any eq 23 reflect Telnet-Connection
  - rx(config-ext-nacl)# deny any log
  - rx(config-etx-nacl)# exit
  - rx(config)# ip access-list extended ***Filterin***
  - rx(config-etx-nacl)# evaluate Telnet-Connections
  - rx(config-etx-nacl)# exit
  - rx(config)# interface serial 0
  - rx(config-if)# ip access-group ***Filterout*** out
  - rx(config-if)# ip access-group ***Filterin*** in

CBAC (Context Based Access Control)

- Intelligently filters TCP and UDP packets based on protocol session information with dynamic ACL's
- In principle can inspect traffic for sessions that originate on any interface of the router but most done for sessions which originate inside
- Inspects traffic that travels through the firewall to discover and manage **state information** for TCP and UDP sessions
  - This state information is used to create temporary openings in the firewall's ACL's to allow return traffic and additional data connections for permissible sessions
- Detection and prevention of certain types of network attacks are possible
  - such as SYN flooding.
  - CBAC also inspects packet sequence numbers in TCP connections to see if they are within expected ranges – and drops any suspicious packets.
  - Additionally, CBAC can detect unusually high rates of new connections and issue alert messages
  - CBAC inspection can help protect against certain denial of service (DoS) attacks involving fragmented IP packets

CBAC Configuration Concept

- **CABC does basically stateful inspection for a series of protocols**
- **NOTE: Router with "Firewall Feature Set" still remains a router !!!**
  - No default protections are enabled
- **Recommended configuration:**
  - 1) `deny ip any any` at outside interface for inbound traffic
  - 2) `ip inspect NAME` at inside interface for inbound traffic
    - Outside ACL will be modified dynamically to allow return traffic
- **`no ip inspect`**
  - Removes entire CBAC configuration
  - And also existing ACLs (!!!)

Supported protocols:

- **FTP**
- **TFTP**
- **H.323**
- **HTTP**
- **ICMP**
- **Java**
- **SIP**
- **RTSP**
- **SQL*Net**
- **Unix r-commands**
- **RPC**
- **SMTP**
- **etc**

CBAC Configuration


```
access-list 100 permit ospf any any  !!! If any RT-protocol needed
access-list 100 deny ip any any log  !!! Important !!!

interface eth0                       !!! Outside interface
   ip access-group 100 in

ip inspect name FW tcp  !!! Important
ip inspect name FW udp  !!! Important
                  icmp
                  ftp

interface eth1               !!! Inside interface
   ip inspect FW in
```

- **Note: inspection can only modify an existing ACL**
  - Therefore, if no ACL configured, then inspection does nothing !!!
- **`# show ip inspect sessions`**
  - `Session 837C7AB0 (10.0.3.12:0)=>(0.0.0.0:0) icmp SIS_OPEN`

Important Timeouts (1)

- **Synwait-time**
  - Time between SYN, SYN-ACK and ACK
  - Default: 30 seconds
  - `ip inspect tcp synwait-time <sec>`
- **Finwait-time**
  - Max completion time of final handshake
  - Default: 5 seconds
  - `ip inspect tcp finwait-time <sec>`

Important Timeouts (2)

- **Idle-time**
  - Max inactivity time
  - Default TCP: 3600 seconds (1 hour)
  - Default UDP: 30 seconds
  - `ip inspect tcp | udp idle-time <sec>`
- ***DNS-timeout***
  - Max inactivity time for DNS session
  - Default: 5 seconds

Other Inspection Parameters (1)

- **`ip inspect max-incomplete high <number>`**
  - Max number of existing half-opened sessions for aggressive mode
  - If reached, IOS deletes half-open sessions until low-level reached (see next command)
  - Default: 500
- **`ip inspect max-incomplete low <number>`**
  - Max number of existing half-opened sessions that cause IOS to stop deleting half-opened sessions
  - Default: 400 (then next 100 are accepted again)
- **`ip inspect one-minute high <number>`**
  - Number of new half-opened sessions per minute (=rate!) at which IOS starts to delete
  - Default: 500
- **`ip inspect one-minute low <number>`**
  - Number of new half-opened sessions per minute (=rate!) at which they stop being deleted
  - Default: 400

Other Inspection Parameters (3)

- **`ip inspect tcp max-incomplete host <number> block-time <minutes>`**
  - Allows only <number> simultaneous sessions to specified DA

Port-to-Application Mapping (PAM)

- **Port-to-Application Mapping (PAM) enables you to customize TCP or UDP port numbers for network services or applications**
  - In order to support network environments with services using ports that are different from the registered or well-known ports
- **Now, admin can add new ports**
  - `ip port-map <appl-name> port [tcp|udp] <port-num> [list <ACL>] [description <desc>]`
    - Appl-name is either system or user-defined (latter must use prefix "user-")
    - Up to five port numbers per application
    - Specification of TCP or UDP is only needed for user-defined applications
  - `access-list permit <acl-num> <ip-addr>`

Audit Trail

- **`rx(config)# ip inspect audit-trail`**
  - Generates Syslog messages upon every CBAC event
- **`rx(config)# no ip inspect alert-off`**
  - Enables real-time alerts

```
Router(config)# logging on
Router(config)# logging 10.0.0.3
Router(config)# ip inspect audit-trail
Router(config)# no ip inspect alert-off
```

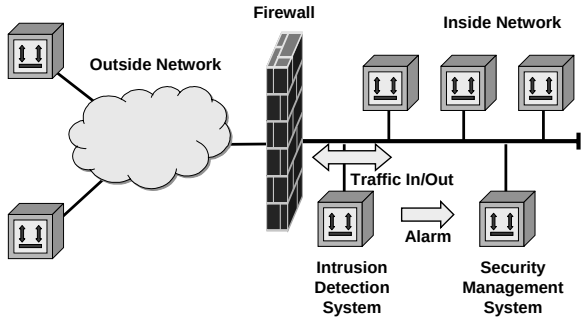
Agenda

- **Firewall Techniques**
  - Introduction
  - Packet-Level FW
  - Stateful Inspection FW
  - Application Level (Proxy) FW
  - Circuit Level FW
  - DMZ
  - Cisco ACL's
- **Intrusion Detection**
- **Security Testing**

Intrusion Detection

- **Process of identifying and responding to malicious activities targeted against networks and its resources**
- **System that performs intrusion detection is called**
  - Intrusion Detection System (IDS)
- **Provides a level of protection beyond the normal firewall service**
  - By securing the network not only against external but also against internal attacks
    - Normally a defense mechanism behind outer barrier
- **Complements defense techniques like firewalls**

Usual Topology with IDS



Intrusion Detection System

- **Base idea**
  - Sniffing the network traffic in real-time
  - Comparing the current network activities with known attack forms (so called signatures)
    - E.g. several TCP SYN segments from the same source IP address to the same destination IP address to several ports within a certain time interval (maybe a DoS attack)
    - E.g. several TCP SYN segments from the different source IP addresses to the same destination IP address to several ports within a certain time interval (maybe a DDoS attack)
  - Create an alarm when an attack is recognized
  - Signatures need to be updated
    - Compare it with normal virus scanner on host machines

Intrusion Detection System

- **Different types of signatures**
  - Atomic
    - A single packet is sufficient
    - No history memory is necessary
  - Compound
    - Several packets within certain period o time needs inspection to identify an attack
    - Needs history memory
    - Such a machine needs more performance and RAM to maintain state information of ongoing traffic

Intrusion Detection / Prevention Systems (IDS / IPS)

- **Network based**
  - Part of the network infrastructure
    - E.g. Dedicated machine
    - E.g. Part of a router / switch
- **Host based**
  - Part of the OS of a computer
- **IDS informs network (security) administrator about attacks**
- **IPS additionally filters malicious packets in case of an attack**
  - Optionally can sent TCP RST packets to end-points of TCP connections to terminate half-open sessions

Intrusion Detection Techniques 1

- **Misuse-based (signature-based)**
  - Observed behavior is compared against description of known, undesirable behavior (signatures)
  - Intrusion is assumed when signature appears in the captured network activity
  - Most commercial systems follow this approach
  - Advantages
    - Accurate reports (low false-positive rate)
  - Disadvantages
    - Needs continuous update of signatures (like a virus scanner)
    - Unable of detecting novel attacks

Intrusion Detection Techniques 2

- **Anomaly-based (or profile-based)**
  - Network behavior is compared against description of anticipated or recorded legal behavior (profile / baseline)
  - Intrusion is assumed when deviation between current network activity and profile is significant
  - Uses statistical methods and AI techniques
  - Advantages
    - Capable of detecting novel attacks
  - Disadvantages
    - Difficult to configure / train
      - E.g. What is the normal behavior?
      - E.g. People work not like machines, so deviation may vary strongly
    - Therefore often a high number of false alarms will be seen

Intrusion Detection Techniques 3

- **Protocol analysis**
  - The different protocol headers of a packet together we the corresponding data payload are analyzed in detail to detect any abnormal or suspicious usage of header and data fields
    - E.g. data payload is not zero in a packet which should not have any data payload
  - In principle like the already mentioned signature technique
    - But most signatures work on the protocol header fields only and do not include the data payload

Signature Types

- **Built-in signature**
  - Contained in the base SW of the IDS
- **Tuned signature**
  - Allows changing of certain parameters of built-in signatures
- **Custom signature**
  - Allows adding of new signatures to the pool

Some Signature Examples

- **Attempt to connect from a reserved address**
  - IP source address not allowed
- **DoS attack on a server**
  - To many packets within a certain time period
- **Illegal TCP flag combination**
  - Bad versus known good flag combinations
- **DNS buffer overflow attempt contained in the payload of an DNS query**

Agenda

- **Firewall Techniques**
  - Introduction
  - Packet-Level F
  - Stateful Inspection FW
  - Application Level (Proxy) FW
  - Circuit Level FW
  - DMZ
  - Cisco ACL's
- **Intrusion Detection**
- **Security Testing**

Tools 1

- **Penetration Tool**
  - Nessus (Vulnerability Scanner)
    - Grandson/daughter of SATAN
    - www.nessus.org
- **Sniffer**
  - tcpdump (Unix Sniffer)
    - www.tcpdump.org
  - WinDump (tcpdump for Windows)
    - netgroup-serv.polito.it
  - WinPcap (Windows Packet Capture Library, libcap for Windows)
    - netgroup-serv.polito.it
  - Ethereal
    - www.ethereal.com

 Firewall, IDS, v4.6 81

Tools 2

- **Network Mapper / Scanner**
  - Nmap
    - www.insecure.org/nmap
- **Sniffer / Packet Replayer**
  - Packetyzer
    - www.networkchemistry.com/products/packetyzer/
    - www.packetyzer.com/forum/
- **Firewall Mapper / Scanner**
    - www.packetfactory.net/

 Firewall, IDS, v4.6 82

Tools 3

- **Password Cracker**
  - WinSniffer
    - www.winsniffer.com/
  - L0phtCrack
    - For Windows NT
  - LC5 (latest commercial version of L0phtCrack)
    - www.atstake.com/products/lc/
- **Man-in-the-middle / Password Cracker**
  - Ettercap
    - ettercap.sourceforge.net
  - Cain and Able
    - www.oxid.it/projects.html

 Firewall, IDS, v4.6 83

Infos

- **www.heise.de/security**
- **isc.sans.org**
- **www.cert.org**
- **www.packetstormsecurity.org/**
- **www.informationweek.com**
  - www.informationweek.com/techcenters/security/
- **www.searchsecurity.com**
- **www.securityfocus.com**
- **www.icsalabs.com (Cybertrust)**
- **…. and many others of course**

 Firewall, IDS, v4.6 84

Virus and Hoax Infos

- **Viruses**
  - www.cert.org
  - www.symantec.com (Norton Antivirus)
  - www.mcafee.com
  - www.icsalab.com
  - www.virusbtn.com
- **Hoaxes**
  - kumite.com/myths
  - www.hoaxkill.com

Diverses Ideen

TCP Wrappers

- **allow host based access control on connections**
- **tcpd replaces daemons from inetd**
- **listens at ports, accepts connections**
- **checks hosts. allow and hosts. deny files**
  - log connection
  - perform double reverse lookups (prevent DNS/ spoofing attacks)

Firewall

- **Not the ultimate solution**
  - cannot deal satisfactorily with content
  - vulnerable to inside attacks and covert channels
  - potential performance bottlenecks
  - when compromised, network is unprotected
- **Security Strategies**
  - least privilege
- **only permissions that are necessary should be granted**
  - defense in depth
- **additional security installations should be present**
  - controlled access
  - fail
  - safe
- **• a failing firewall may not reduce security**

Filtering Routers

- **Filtering Routers route packets between internal and external hosts**
  - do it selectively
  - perform filtering
    - allow or block certain types of packets
- **Screening procedure is based on**
  - Protocol (whether the packet is a TCP, UDP, or ICMP packet)
  - IP source/ destination address
  - TCP or UDP source/ destination port
  - TCP flags
  - ICMP message type
  - interfaces where packets are arriving and leaving

Packet Filter

- **Old ones might be vulnerable to spoofing**
- **Fragmented Datagrams**
  - discarded when not enough information to apply filter
  - when first fragment contains enough information, remaining ones are passed unchecked
  - potential vulnerability
    - first fragment with innocent values
    - other fragments with non- zero offset rewrite these value with malicious ones
    - reassembled fragment is delivered to protected service

Packet Filter

- **Advantages**
  - easy to implement (relies on existing hardware)
  - good performance
- **Limits**
  - limited auditing
  - difficult to configure
  - not very flexible, extensible
- **Linux**
  - iptables , ipchains

Stateful Inspection

- **acts as a packet filter**
- **but accesses higher- level protocol information**
  - check also content of packet / deny on match (e. g. virus)
  - allows to track sessions (e. g. ftp, http)
  - virtual sessions for connection- less protocols (e. g. UDP)
- **firewall stores ports used in a particular UDP transaction**
- **temporarily creates an exception to let the answer pass through**
- **Checkpoint firewall**

Circuit Level Gateway

- **Not only checks packets, but sessions / connections**
  - based on user / password (e. g. first telnet to gateway, then telnet to the outside)
  - time of day
- **all traffic is disallowed, only validated sessions may transfer data**
- **do not need to be aware of the protocol**
- **cannot perform application- level filtering**

IDS and firewalls

- **Firewalls and IDS will eventually be combined into a single capability**
  - Many firewalls can trigger alerts when traffic to “bad destination” is seen
  - This capability can be used to build “burglar alarms”
- **A burglar alarm is a misuse detection system that is carefully targeted**
  - You may not care about people port- scanning your firewall from the outside, but this information is important if it is happening from the inside.
  - Trivial burglar alarms can be built using tcpdump and perl

