Countermeasures Analysis

The goal of this steps is to determine how risk can be reduced

Countermeasures

A first classification
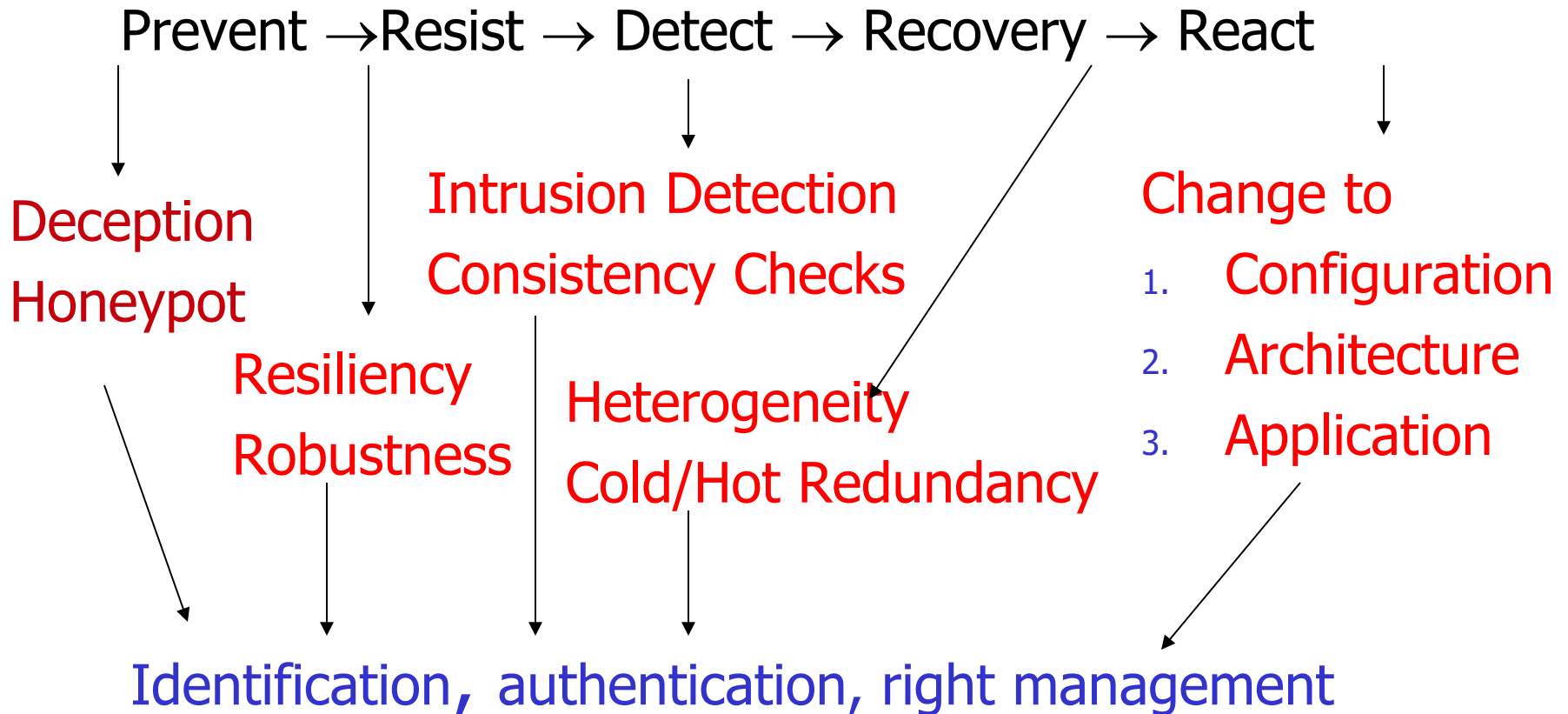
- Proactive
  - They are applied before an attack
    - eg a vulnerability is removed

- Dynamic
  - They are applied as soon as an attack is detected
    - eg a vulnerability is removed
    - eg a connection is killed

Detection?

- Reactive
  - They are applied after a successful attack
    - eg a vulnerability is removed
    - eg a password is changed

A more detailed taxonomy

Prevent →Resist → Detect → Recovery → React

Deception

Honeypot

Resiliency

Robustness

Intrusion Detection

Consistency Checks

Heterogeneity

Cold/Hot Redundancy

Change to

1. **Configuration**
2. **Architecture**
3. **Application**

Identification, authentication, right management

Implementation mechanisms

- Countermeasures are implemented through a set of common mechanisms

- A set of shared mechanisms
  - It can increase the cost effectiveness of countermeaures
  - It should be highly robust because a vuln may affect several countermeasures

Base mechanisms

- The mechanisms are defined on top of a security kernel (= TCB) that manages
  - The user identities
  - User authentication (identity checks)
  - User rights
- This should not be confused with the minimal system that is discussed in the following

Countermeasures Glossary- I

- Deception = no information about the system design is available = S&S, open design
- Honeypot = fake systems are introduced to increase the complexity of discovering nodes to be attacked
- Resiliency/Robustness = prevent a single vulnerability from enabling a successful attack (S&S, least privilege etc)
- Intrusion Detection/ Consistency Check = a set of checks to discover current or previous attacks
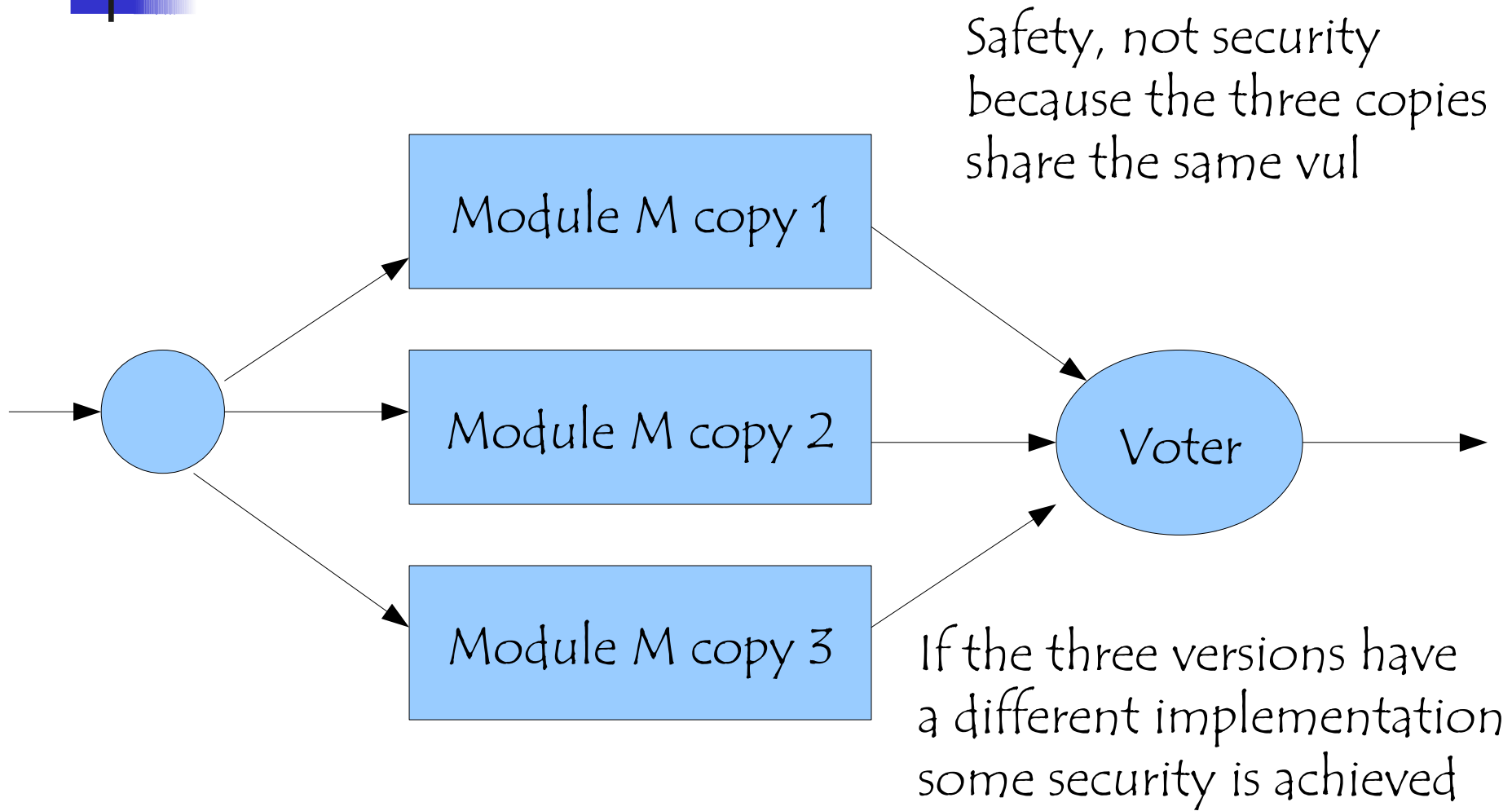
Countermeasures Glossary - II

- Redundancy = spare components to replace the attacked ones. The impact is reduced and control on the system is not lost
  - Cold = Stand by spare components
  - Hot = Spare components are in use (oversize system)

  The underlying problem is a properly evaluation of expected performance
- Heterogeneous components = genetic diversity = the vulns of spare components differs from those of standard components
- A generalization of triple modular redundancy

Triple Modular Redundancy

Safety, not security
because the three copies
share the same vul

Module M copy 1

Module M copy 2

Module M copy 3

Voter

If the three versions have
a different implementation
some security is achieved

Countermeasures Glossary- III

- Minimal system
  - A subset of components
  - More robusted
  - More severe checks
- Control of the minimal system should never be lost
- It is a starting point to gain back control on the whole system
- Strongly related to impact/power law

Countermeasures Glossary- IV

- Reaction = Updates to
    - The configuration of the OS and applications
    - System architecture
    - Enabled application
    - Patch
- The reaction should involve (work on) the target system rather than the attacking one

No action on the attacking sys?

- Stepping stone = a chain of hosts that starts at the one of the attacker and that are, illegally, controlled by the attacker =botnet
- The chain enables the attacker to hide his/her location
- The attack is implemented by the last node of the chain to hide the first one
- Any node connected to the internet has a value as it can be used as a stepping stone
- How can we discover a stepping stone?

Stepping stone - 1

- An analysis of input/output node channel to evaluate their correlation

- If there are an input channel and an output one that are correlated as far as concerns

  - Time = when a communication occurs

  - Data = size of exchanged data

  then the node may act as a stepping stone

- By repeating the analysis for the sender/receiver of the two channeld, the whole chain of stepping stones may be discovered

Stepping stone - 2

- The proposed analysis is a traffic analysis that can be applied even to encrypted flows because it does not consider the information content of the two flows

- It is almost impossible that the flows in a stepping stone chains are in clear

Deception

- Its importance has increased because of the developing of virtualization technologies that minimizes its cost
- It increases the complexity of attacks that use a vulnerability scanner to discover nodes in a network that can be attacked
- For each address generated by the scanner a new fake virtual node is created that has to be analyzed by the attacker
- Useless virtual nodes are introduced that, as far as a scanning is concerned, behave like real nodes
- The fake nodes replies to the fingerprinting are slower and slower to slow down the scanning
- An alarm is raised

Countermeasures - Deception

- Cryptography algorithms
- Information is coded so that only who knows a further info, the key, can access it
- Already discussed

Just a reminder …

- Cryptography does not solve the problems, it only simplify the solution
- It is very difficult to safely store a 2 gb file
- It is rather simpler to encrypt the file through a 256 bit key and safely store the key
- The same problem has to be solved (safely store an info) but now the solution is simpler because the problem size has been reduced

Resist – Robust programming

- Validate program inputs
- Prevent buffer overflow
- Robust implementation
- Check the invocations to other resources
- Check returned results

Robust programming – Input validation

Input validation + default deny (S&S)

- Define the input legal structure
- Check that any input satisfy the defined structure

Example: Strings

- A grammar that defines the structure
- Longest input string
- Define which special characters are legal
- Check that any input satisfies 1-2-3

Robust programming – Input validation

- The ability of defining a set of checks to validate the input should be considered when the program is specified rather than after the design of the application

- In the correct approach, the application is specified and designed to simplify the definition and the implementation of the checks through a simple grammar, eg LR grammar, that means controls implemented by finite state automat

- A complex control may be useless if we are not confident that it has been correctly implemented

Robust programming – Input validation

- Parameters to be validated
  - Environment variables
  - File names (blanks , .., /,)
  - Email addresses
  - URL
  - Html
  - data
- Several languages define built in function to match a string against a predefined pattern (regular expression etc.)

Robust programming – no buffer overflow

- Do not use any library function that does not check it input parameters

- Use only those functions that check the length of their input strings

- Dynamic memory allocation of a data structure rather than static allocation of the largest data structure

# Robust programming – robust implementation - I

- Satisfy S&S
- Rigorous definition of the program interface
- Do not assume that input/output values are related
  - If a function of a library returns a pointer and another function of the same library has a pointer parameter, there is no reason to assume that the one transmitted to the second is the one that has been returned by the first one
  - If an input parameter of a function should be equal to the output of another function, the parameter has to be defined so that this relation can be checked
- Data and instruction should be different
- The data that each function can access should be minimized

Pointer - I

A more robust version

Package that should be robust

Proc$_i$

punt

Proc$_k$

punt

Proc$_p$

i

Pointer array

i

Proc$_k$
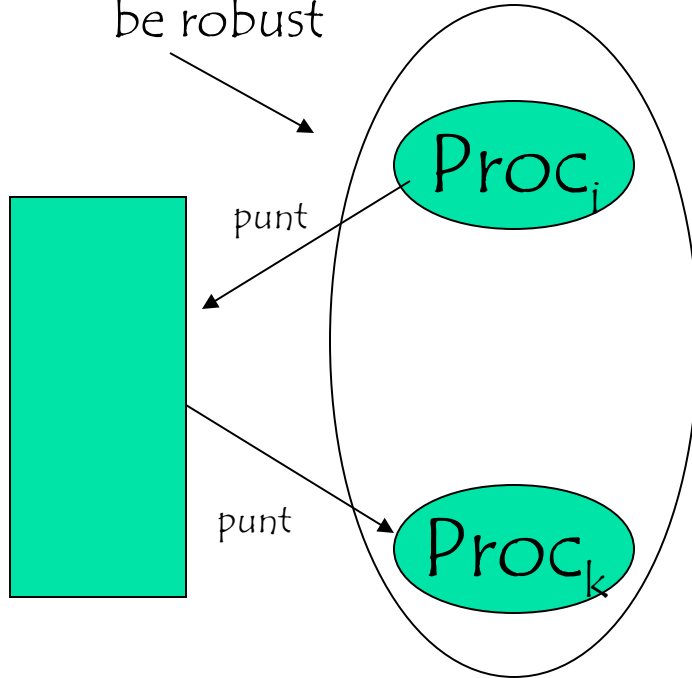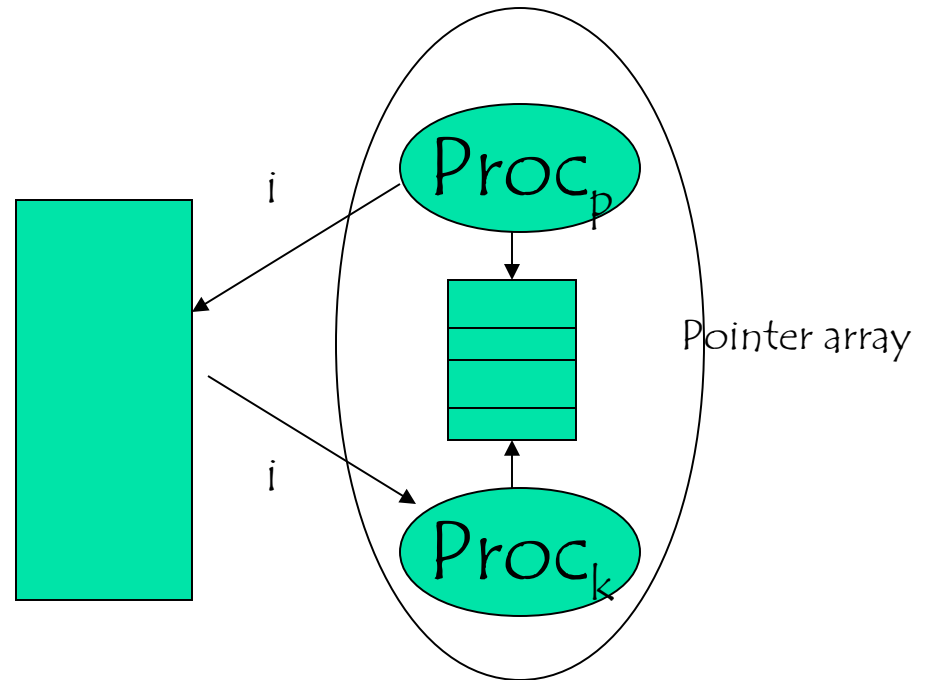
An index is transformed into a pointer by accessing the pointer array

Pointers - II

- By replacing an array of pointers with an array of records we can
  - Introduce fields in the records to discover whether each element is properly initialized
  - Check access to the array
  - Define some check on the input output relation of a pointer
- This is a simplified, redundant version of an access control matrix for the pointers

Pointers - III

- We can also return an encrypted index to the pointer array rather than the real one

  realpositioin= m*returnedpos+cost

- It simplifies the detection of pointer manipulation
- Access control does not change

Robust Programming – robust implementation - II

- Safe variable initialization
- Avoid critical runs by parallelizing operations and consistency checks
  - Time- to-check/time-to-use
  - Open file;checks;close;open;use
- Atomic transaction on the file system
- Lock to guarantee consistency but time out to prevent starvation
- Quota mechanisms for shared resources

Robust programming – check invocations

- Only safe functions should be invoked

  (eg functions that checks their input/output parameters)

- Check
  - the correctness of transmitted parameters
  - of metadata in transmitted parameters
  - the values that are returned

- Hide and protect critical information

Robust programming – check returned results

- Do not leak information before the user is authenticated (banner etc)
- Do not return too much information (yes or no without explaing why)
  - Do not say if the user or the password does not exist but just that the pair (user, password) does not exist
- Information useful for the debugging should be returned in log files in the node rather than in the user interface
- Avoid dependency on the user to prevent DOS attacks
  - Avoid synchronous communications,
  - If synchronous communications are required, introduce a sacrifical thread

Robust programming vs programming language

- Most of the previous constraints can be
  - Enforced by the program run time support (Java)
  - Be satisfied because a discipline is imposed on the programmer (C)
- Both solutions are acceptable, one privileges performance the other security
- The only solution to be avoided is a support that has a low performance even if it does not enforce the constraints

A distinct perspective

- The 2011 CWE/SANS Top 25 Most Dangerous Programming Errors is a list of the most significant programming errors that can lead to serious software vulnerabilities.

- They occur frequently, are often easy to find, and easy to exploit.

- They are dangerous because they will frequently allow attackers to completely take over the software, steal data, or prevent the software from working at all.

The 25 errors

- Aree partitioned into three classes
  - Unsafe interactions among components
  - Risky resource management
  - Porous defenses
- Selected according to
  - Frequency
  - Danger

Attributes of each error

- Weakness Prevalence: diffusion
- Attack Frequency: how often the weakness occurs in vulnerabilities that are exploited by an attacker.
- Ease of Detection: how easy it is for an attacker to find this weakness.
- Remediation Cost: the amount of effort required to fix the weakness.
- Attacker Awareness: the likelihood that an attacker is going to be aware of this particular weakness, and of methods for detection and for exploitation.
- Consequences = Potential impact

The list

- C:\Users\Hp\Dropbox\didattica\corsimiei\c

Countermeasures - Resist

- Correct configuration (hardening) of standard software component (OS, packages)
  - Determine useful functions
  - Remove useless functions
  - Remove any standard account or at least update its password

Countermeasure - Resist

The confinement principle

Running untrusted code

We often need to run buggy/unstrusted code:

- – programs from untrusted Internet sites

- – old or insecure applications: ghostview,   outlook

- – legacy daemons: sendmail, bind
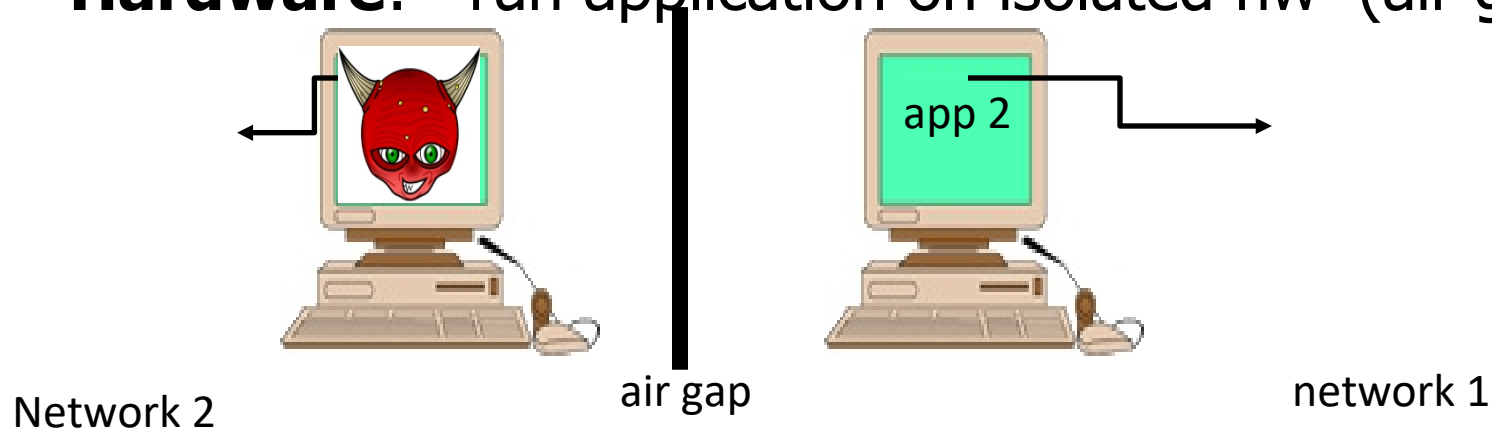
- – Honeypots

    <u>Goal</u>:   if application "misbehaves" ⇒ kill it

Approach: confinement

Confinement: ensure misbehaving app cannot harm rest of system

Can be implemented at many levels:

- **Hardware**:   run application on isolated hw (air gap)

Network 2   air gap   network 1

⇒ difficult to manage

Implementing confinement

- Key component: **reference monitor**

- **Mediates requests** from applications

- Must **<u>always</u>** be invoked:

  - Every application request must be mediate
- **Tamperproof**:
  - Cannot be killed or if killed, then monitored process is killed too
- **Small** enough to be analyzed and validated
- Old implementation : jail

Not all programs can run in a jail

Programs that can run in jail:

· audio player

· web server

Programs that cannot:

· web browser

· mail client

Problems with chroot and jail

Coarse policies:

- All or nothing access to parts of file system

- Inappropriate for apps like a web browser

  · Needs read access to files outside jail
      (e.g. for sending attachments in Gmail)

Does not prevent malicious apps from:

- Accessing network and messing with other machines
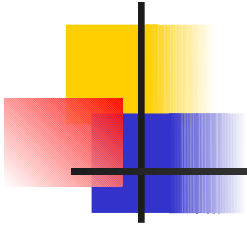
- Trying to crash host OS

Isolation

System Call Interposition

System call interposition

Observation: to damage host system (e.g. persistent changes) app must make system calls:

- To delete/overwrite files: unlink, open, write

- To do network attacks:socket, bind, connect, send

Idea: monitor application system calls to block unauthorized calls

Implementation options:
  - Completely kernel space (e.g. GSWTK)
  - Completely user space (e.g. program shepherding)
  - Hybrid (e.g. Systrace)

Initial implementation (Janus)

Linux **ptrace**:   process tracing

process calls: **ptrace (… , pid_t pid , …)**
and wakes up when **pid** makes sys call.

user space

| monitored application (browser) | | monitor |

open("/etc/passwd", "r")

OS Kernel
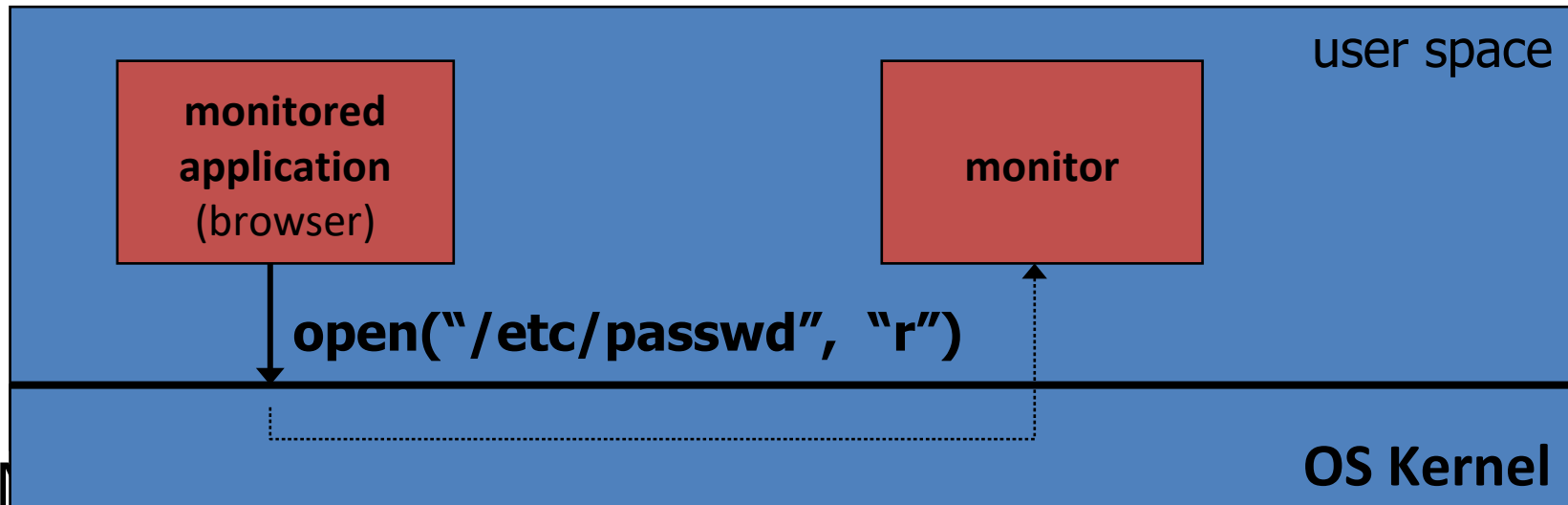
Complications

- If app forks, monitor must also fork

  - forked monitor monitors forked app

- If monitor crashes, app must be killed

- Monitor must maintain all OS state associated with app

  – current-working-dir (**CWD**),   **UID, EUID, GID**

  – When app does "cd path" monitor must update its CWD

    - otherwise: relative path requests interpreted incorrectly

Problems with ptrace

Continued

– Trace all system calls or none

– Monitor cannot abort sys-call without killing app
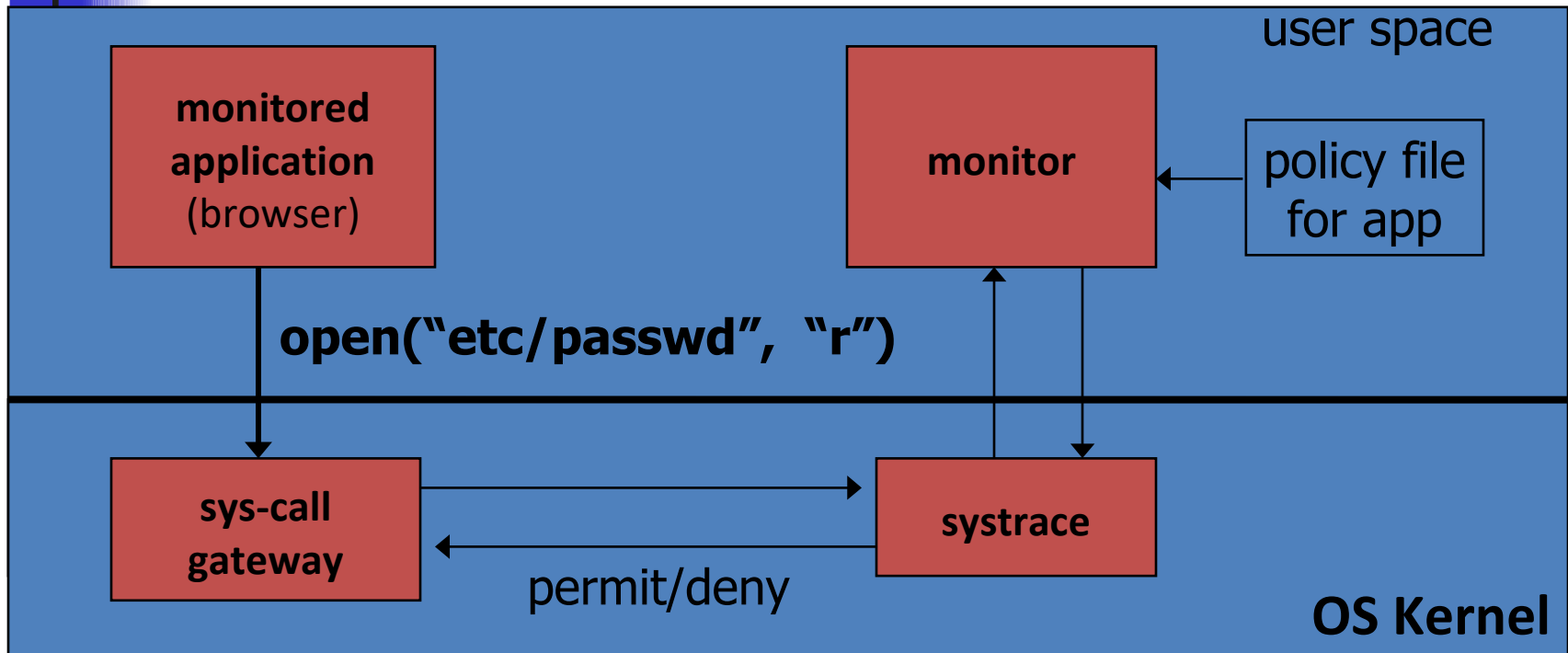
Security problems: **race conditions**

– <u>Example</u>:   symlink: me ⟶ mydata.datproc 1:
    1: open("me")   monitor checks and authorizes

    2:   me ⟶ /etc/passwd

OS executes   open("me")
    Classic **TOCTOU bug**:   time-of-check / time-of-use

Alternate design: systrace

- systrace only forwards monitored sys-calls to monitor (efficiency)
- systrace resolves sym-links and replaces sys-call path arguments by full path to target
- When app calls execve, monitor loads new policy file

Policy

Sample policy file:

```
path allow  /tmp/*
path deny  /etc/passwd
network deny all
```
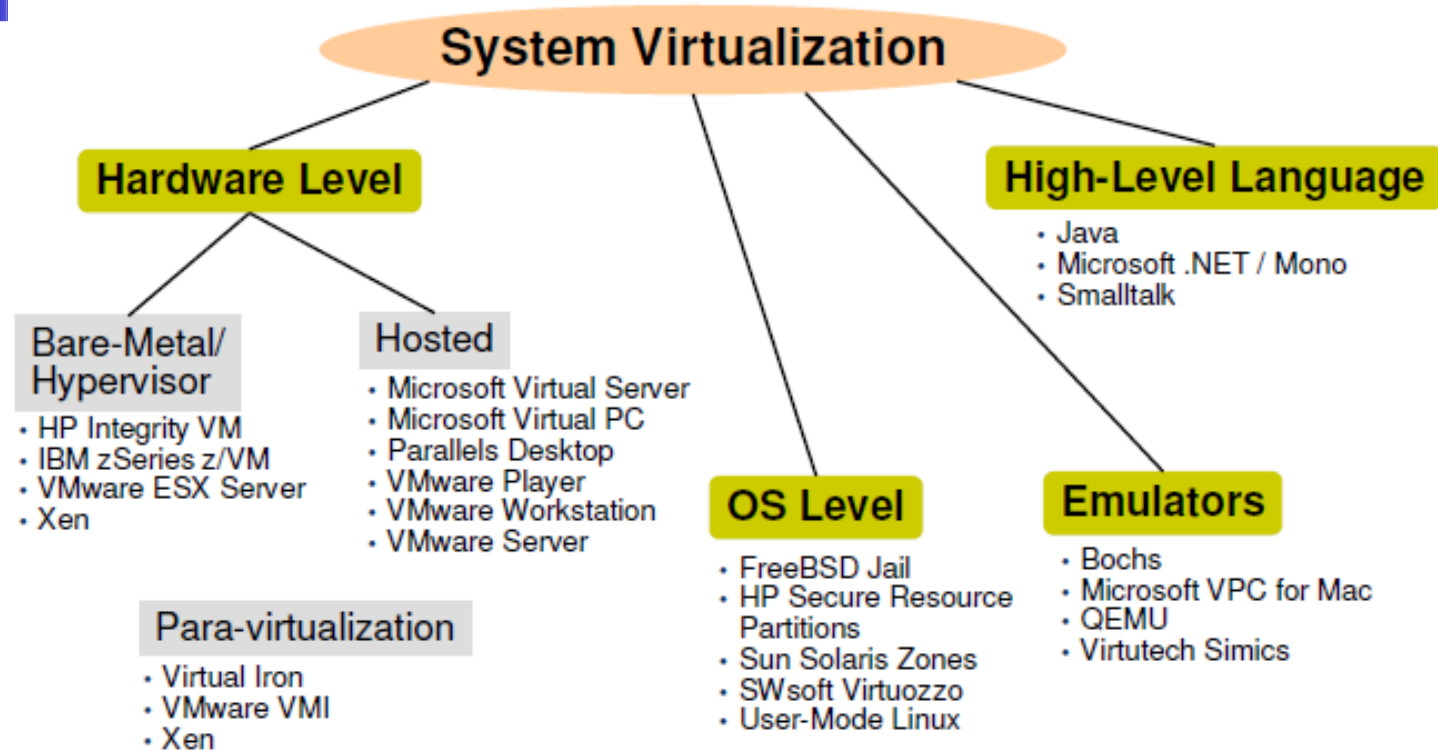
- Manually specifying policy for an app is difficult:

    - Systrace can auto-generate policy by learning how app behaves on "good" inputs
    - If policy does not cover a specific sys-call, ask user … but user has no way to decide

- Difficulty with choosing policy for specific apps (e.g. browser) is the main reason it is not widely used
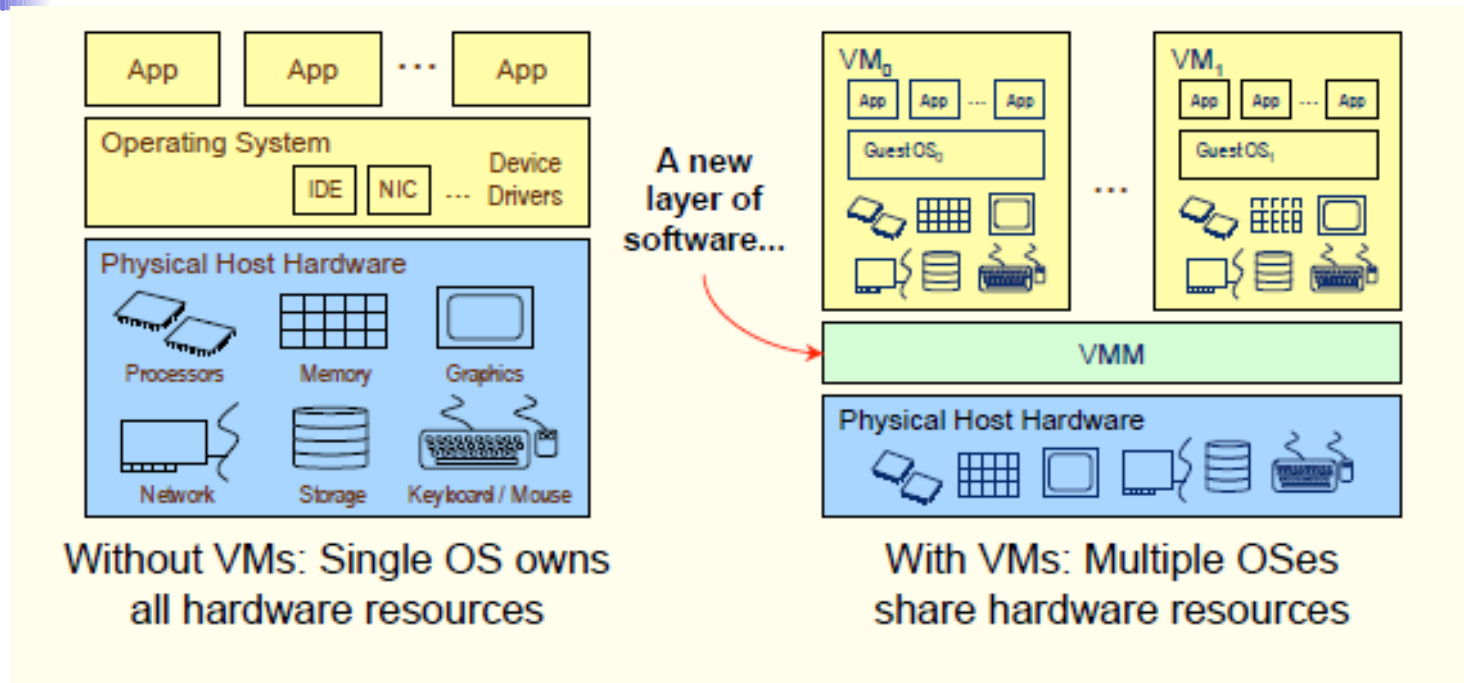
Isolation

Isolation via Virtual Machines

Classification

System Virtualization

Hardware Level

High-Level Language
- Java
- Microsoft .NET / Mono
- Smalltalk

Bare-Metal/ Hypervisor
- HP Integrity VM
- IBM zSeries z/VM
- VMware ESX Server
- Xen

Hosted
- Microsoft Virtual Server
- Microsoft Virtual PC
- Parallels Desktop
- VMware Player
- VMware Workstation
- VMware Server

OS Level
- FreeBSD Jail
- HP Secure Resource Partitions
- Sun Solaris Zones
- SWsoft Virtuozzo
- User-Mode Linux

Emulators
- Bochs
- Microsoft VPC for Mac
- QEMU
- Virtutech Simics

Para-virtualization
- Virtual Iron
- VMware VMI
- Xen

Focus on system VMs

Without VMs: Single OS owns all hardware resources

With VMs: Multiple OSes share hardware resources

A new software layer is introduced that honors the existing ISA to create distinct physical machines

Alternative solutions

Interpretation
- Problem – too inefficient
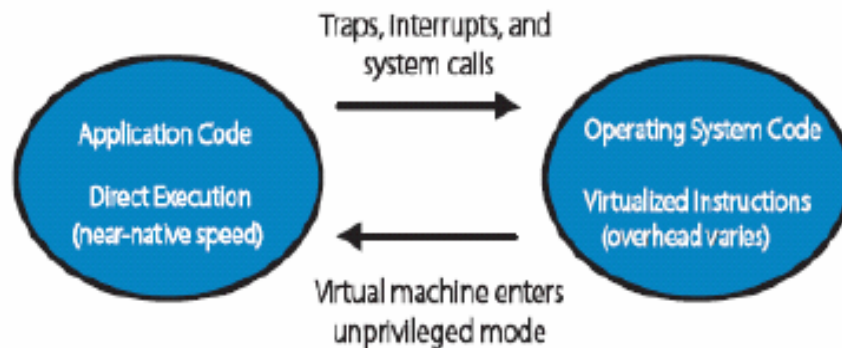- x86 decoding slow

Code Patching
- Problem – not transparent
- Guest can inspect its own code

Binary Translation (BT)
- Approach pioneered by VMware
- Run any unmodified x86 OS in VM

Extend x86 Architecture

Software VMM

Traps, Interrupts, and system calls

Application Code
Direct Execution (near-native speed)

Operating System Code
Virtualized Instructions (overhead varies)

Virtual machine enters unprivileged mode

Direct execute unprivileged guest application code

- Will run at full speed until it traps, we get an interrupt, etc.

"Binary translate" all guest kernel code, run it unprivileged

- Since x86 has non-virtualizable instructions, *proactively* transfer control to the VMM (no need for traps)
- Safe instructions are emitted without change
- For "unsafe" instructions, emit a controlled emulation sequence
- VMM translation cache for good performance

Solution adopted by VMware

Binary – input is x86 "hex", not source

Dynamic – interleave translation and execution
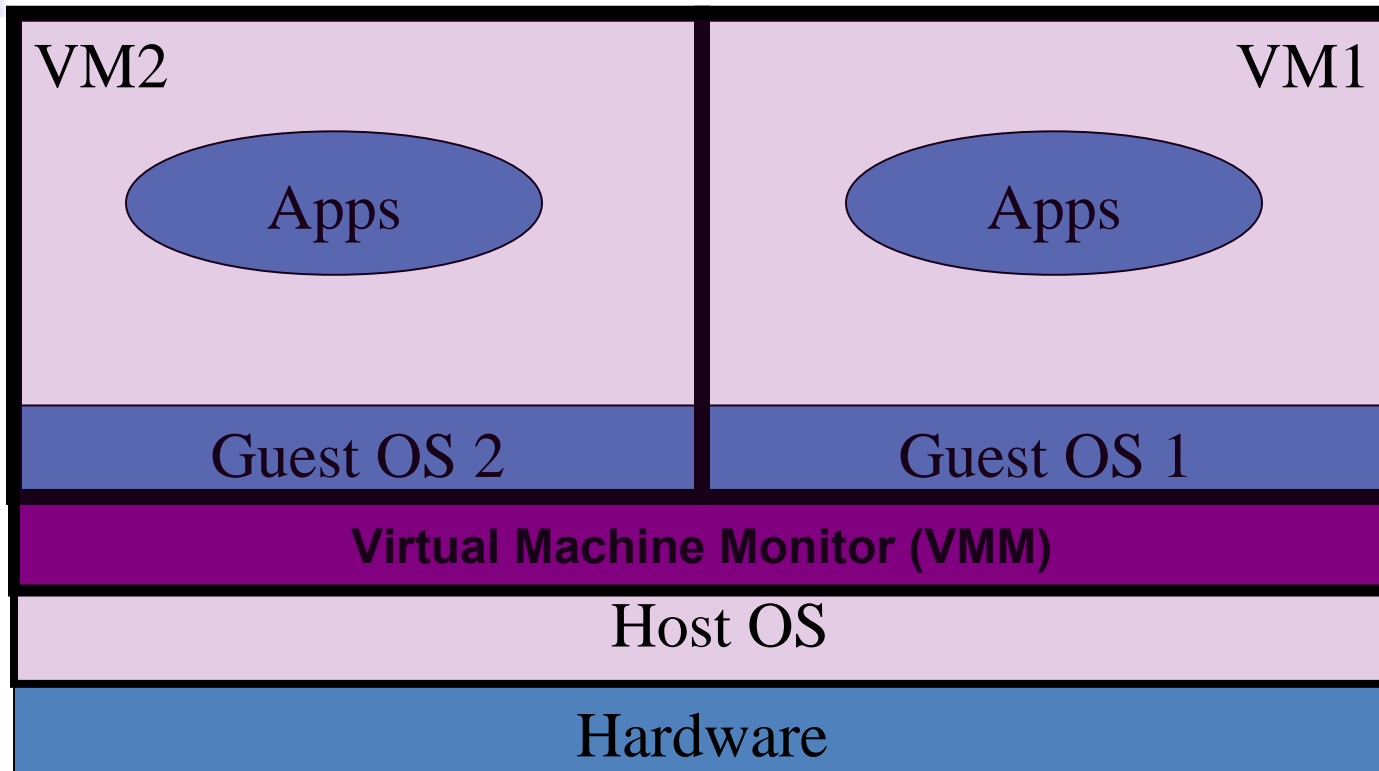
On Demand – translate only what about to execute (lazy)

System Level – makes no assumptions about guest code

Subsetting – full x86 to safe subset

Adaptive – adjust translations based on guest behavior

Virtual Machines

VM2	VM1
Apps	Apps
Guest OS 2	Guest OS 1
Virtual Machine Monitor (VMM)	
Host OS	
Hardware	

Example: single HW platform used for both classified and unclassified data (two levels of a MAC policy)

Why so popular now?

VMs in the 1960's:

- Few computers, lots of users
- VMs allow many users to shares a single computer

VMs 1970's – 2000:   non-existent

VMs since 2000:

- Too many computers, too few users
  - Print, Mail, Web, File server, Database ,
- Wasteful to run each service on different hardware
- More generally: VMs heavily used in cloud computing

VMM security assumption

VMM Security assumption:

– Malware can infect <u>guest</u> OS and guest apps

– But malware

- cannot escape from the infected VM
- Cannot infect <u>host</u> OS or other VMs on the same hardware

Requires that VMM protect itself and is not buggy

– VMM is much simpler than full OS

... but device drivers run in Host OS

Problem: covert channels

- **Covert channel**: unintended communication channel between isolated components

  – Can be used to leak classified data from secure component to public component
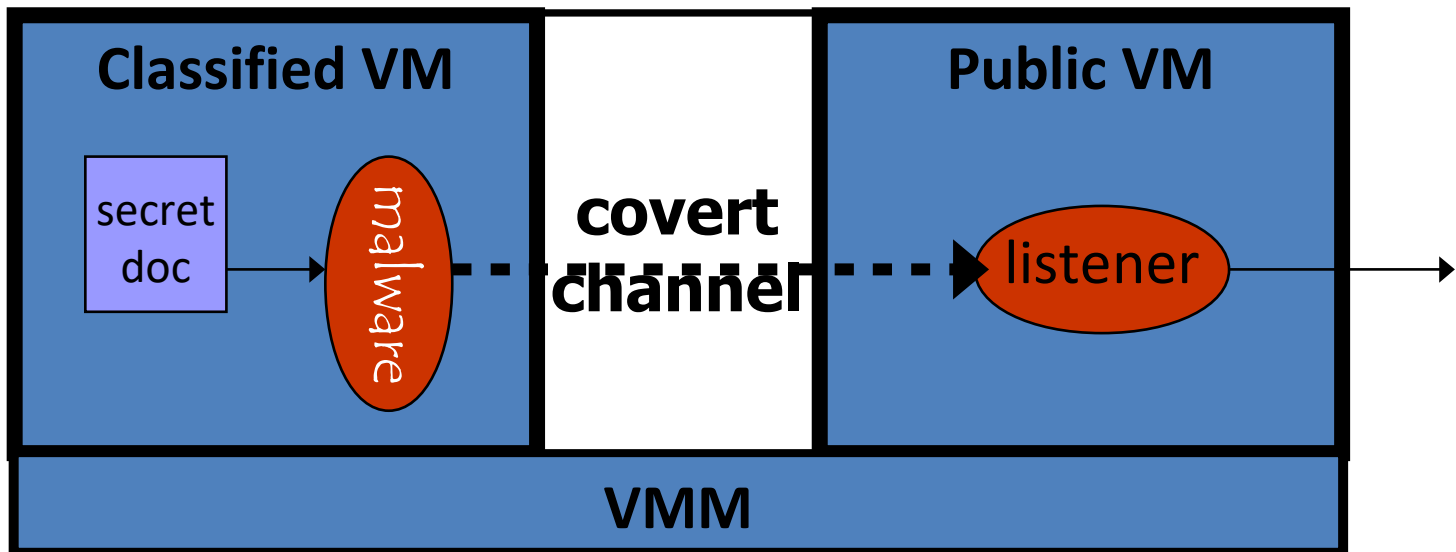
An example covert channel

Both VMs use the same underlying hardware

To send a bit   b $\in$ {0,1}   malware does:

- b= 1:   at 1:00am do CPU intensive calculation
- b= 0:   at 1:00am do nothing

At 1:00am listener does CPU intensive calc. and measures completion time

$$b = 1 \quad \Leftrightarrow \quad \text{completion-time} > \text{threshold}$$

Many covert channels in running system:

- File lock status, cache contents, interrupts,

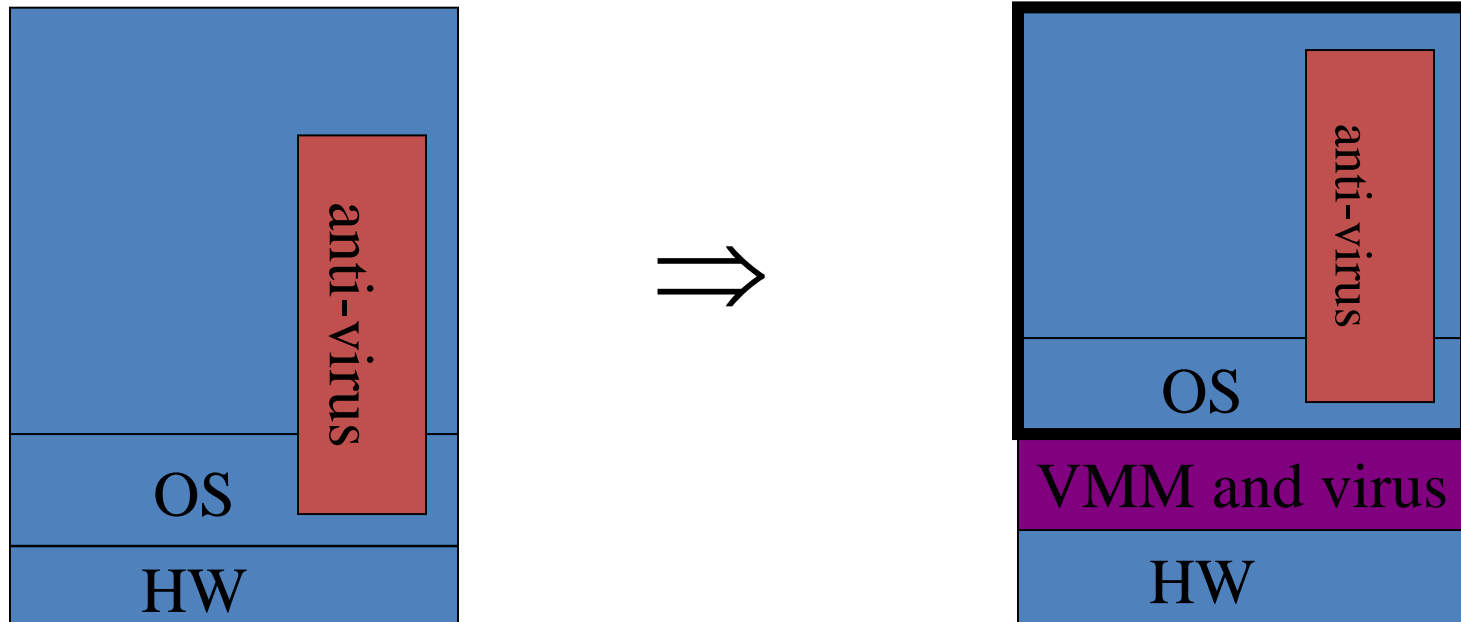- Difficult to eliminate all (reduce bandwidth)

Isolation

Subverting VM Isolation

Subvirt [King et al. 2006]

Virus idea:

- Once on victim machine, install a malicious VMM

- Virus hides in VMM

- Invisible to virus detector running inside VM

The MATRIX

VM Based Malware (blue pill virus)

- **VMBR**:   a virus that installs a malicious VMM (hypervisor)

- **Microsoft Security Bulletin: (Oct, 2006)**

  – Suggests disabling hardware virtualization features
    by default for client-side systems

- **But VMBRs are easy to defeat**

  – A guest OS can detect that it is running on top of VMM

VMM Detection

Can an OS detect it is running on top of a VMM?

Applications:

- Virus detector can detect VMBR

- Normal virus (non-VMBR) can detect VMM

  · refuse to run to avoid reverse engineering

- Software that binds to hardware (e.g. MS Windows) can refuse to run on top of VMM

- DRM systems may refuse to run on top of VMM

VMM detection   (red pill techniques)

- VM platforms often emulate simple hardware

  – VMWare emulates an ancient i440bx chipset

        … but report 8GB RAM, dual CPUs, etc.

- VMM introduces time latency variances
  – Memory cache behavior differs in presence of VMM

  – Results in relative time variations for any two operations

- VMM shares the TLB with GuestOS
  – GuestOS can detect reduced TLB size

- … and many more methods **[GAWF'07]**

Isolation

Software Fault Isolation

Software Fault Isolation [Whabe et al., 1993]

**Goal**:   confine apps running in <u>same address space</u>

– Codec code should not interfere with media player

– Device drivers should not corrupt kernel

Simple solution: runs apps in separate address spaces = distinct processes not threads

– Problem: slow if apps communicate frequently, requires one context switch per message

Software Fault Isolation

Partition process memory into segments

code segment	data segment	code segment	data segment	------
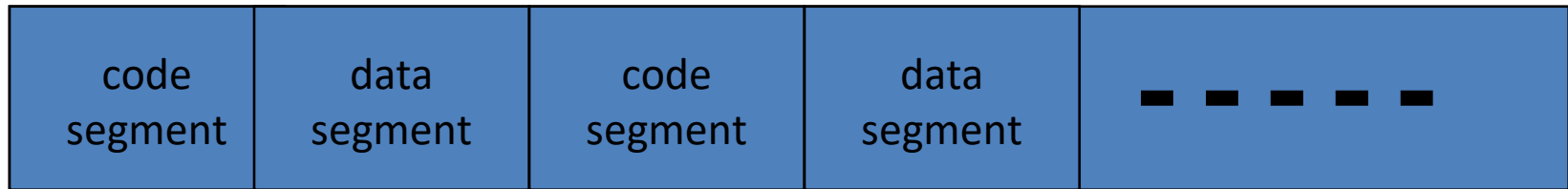
App 1   App 2

- Locate unsafe instructions: **jmp, load, store**

  - At compile time, add guards before unsafe instructions

  - When loading code, ensure all guards are present

Segment matching technique

Designed for MIPS processor.   Many registers available.

- **dr1, dr2**:   dedicated registers not used by binary

  – compiler pretends these registers don't exist

  – **dr2** contains segment ID

- Indirect load instruction   **R12 ← [R34]**   becomes:

```
dr1 ← R34
scratch-reg ← (dr1 >> 20)          : get segment ID
compare scratch-reg  and  dr2      : validate seg. ID
trap if not equal
R12 ← [dr1]                        : do load
```

Address sandboxing technique

- **dr2**:   holds segment ID

- Indirect load instruction   **R12 ← [R34]**   becomes:

    dr1 ← R34 & segment-mask   : zero out seg bits
    dr1 ← dr1 | dr2   : set valid seg ID
    R12 ← [dr1]   : do load

- Fewer instructions than segment matching

… but does not catch offending instructions

- Similar guards places on all unsafe instructions

Address Sandboxing

Problem: what if **jmp [addr]** jumps directly into indirect load?
(bypassing guard)

Solution:

jmp guard must ensure **[addr]** does not bypass load guard

Isolation: summary

- Many sandboxing techniques:

  Physical air gap,   Virtual air gap (VMMs),

  System call interposition, Software Fault isolation

  Application specific (e.g. Javascript in browser)

- Often complete isolation is inappropriate

  - Apps need to communicate through regulated interfaces

- Hardest aspects of sandboxing:
  - Specifying policy: what can apps do and not do

  - Preventing covert channels

Countermeasures - Resist

- Firewall
  - A system that connects two networks with distinct security requirements
  - It filters the information between the two networks and the services each network can access in the other one
  - It hides some components in the most critical networks so that they cannot be accessed from the less critical network
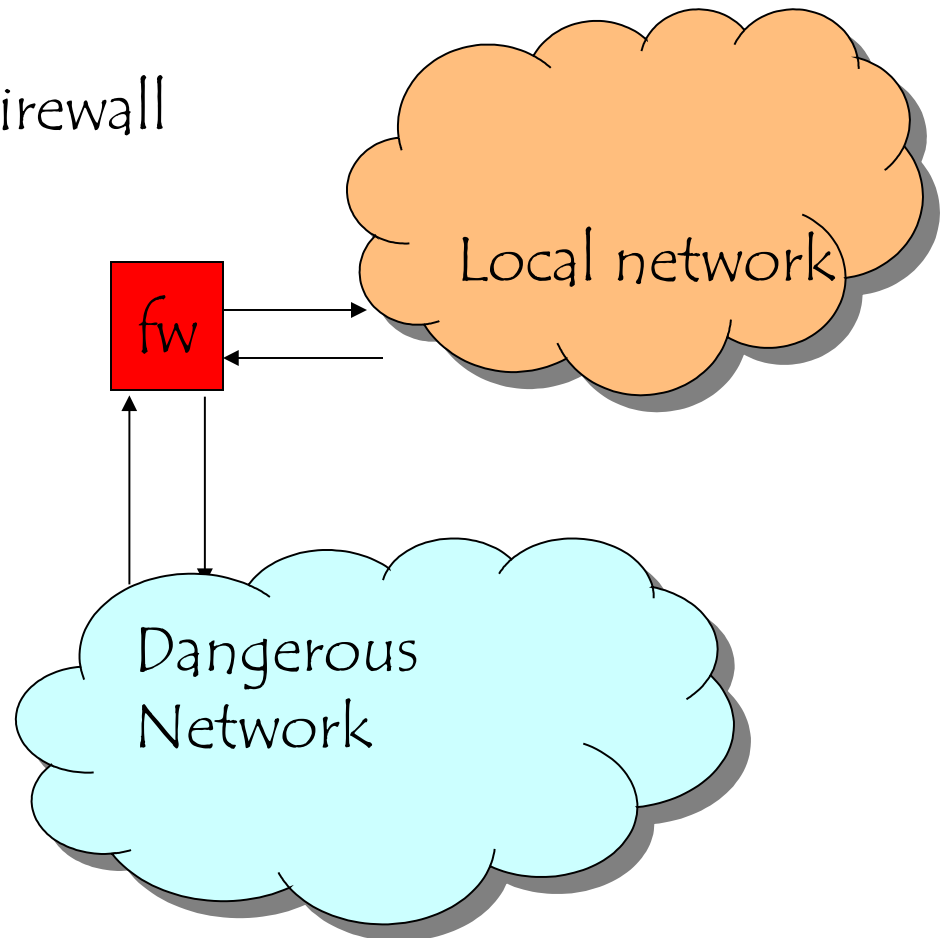 - It defends the most critical network from attacks originating in the less critical and less protected one at the expence of the bandwidth between the two networks

Introducing a Firewall

Introducing a Firewall

- A firewall CAN protect a network from attacks from outside the network
  - It prevents connection to critical nodes of the network it protects
  - It filters information that may be transmitted through legal connections
  - It can force stronger user authentication to connect to enter or to leave the network it protects

Introducing a Firewall

- A firewall CANNOT protect a network from attacks
  - Originating from within the network (insider threat)
  - That exploits lines it cannot control
  - That exploits protocol that it does not know (unless a default deny strategy is adopted)

Introducing a Firewall

- The firewall behaviour fully depends upon the adopted security policy

- The behaviour is based upon the distinction inside/outside

- All the mechanisms are implemented in a single point (controls are fully delegated to the firewall)

- Fail safe or fault tolerance (redundancy) of the firewall

Firewall: properties

- A firewall is characterized by
    - The protocols it can analyze (stack layer it can analyze to protect a network)
    - Its implementation (router, dedicated node, router+ dedicated node)
    - The two properties are distinct and fully orthogonal and they determine the overall robustness of the firewall = robustness enabled by the controls +

   robustness in the control implementation

An example - I

- The same set of controls can be implemented in
  - A firewall that receives and transmits through the same network interface
  - A firewall that receives and transmits through two distinct network interfaces
  - A firewall with two interfaces that are the only connections between the two networks
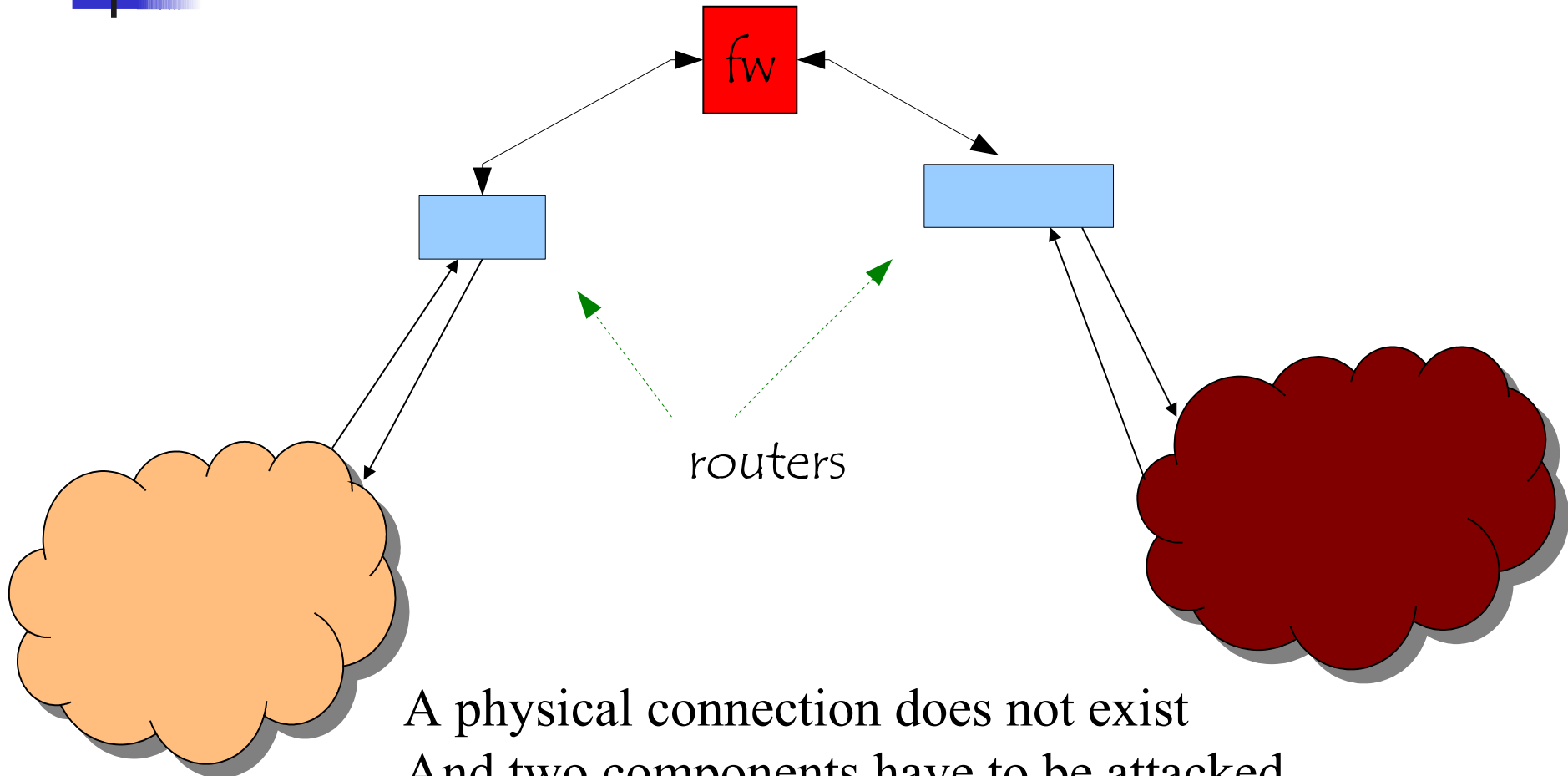
Some architectures - 1

fw

A hub

Routing tables
To filter
traffic

A physical connection exist
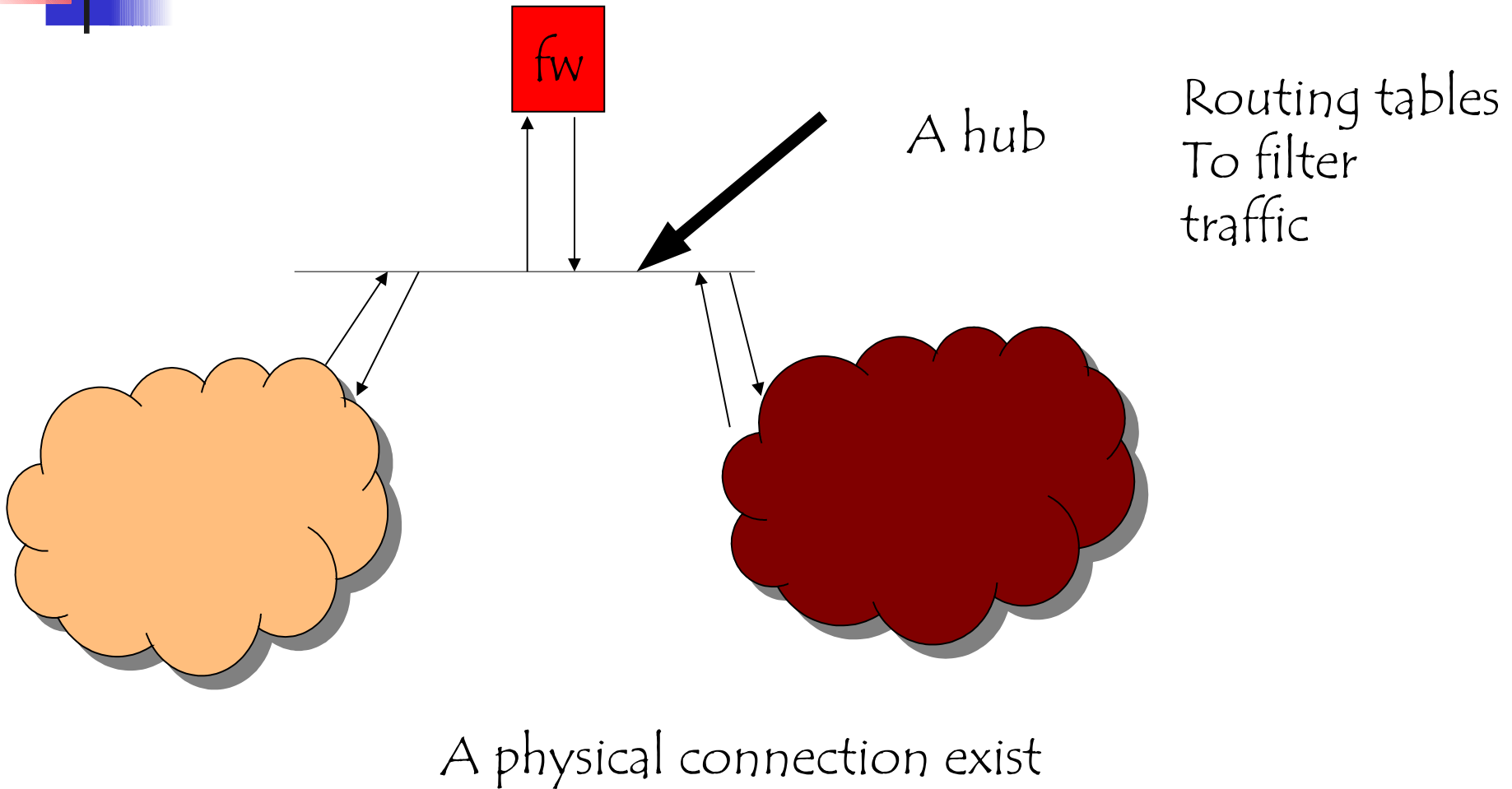
Some architectures - 2

fw
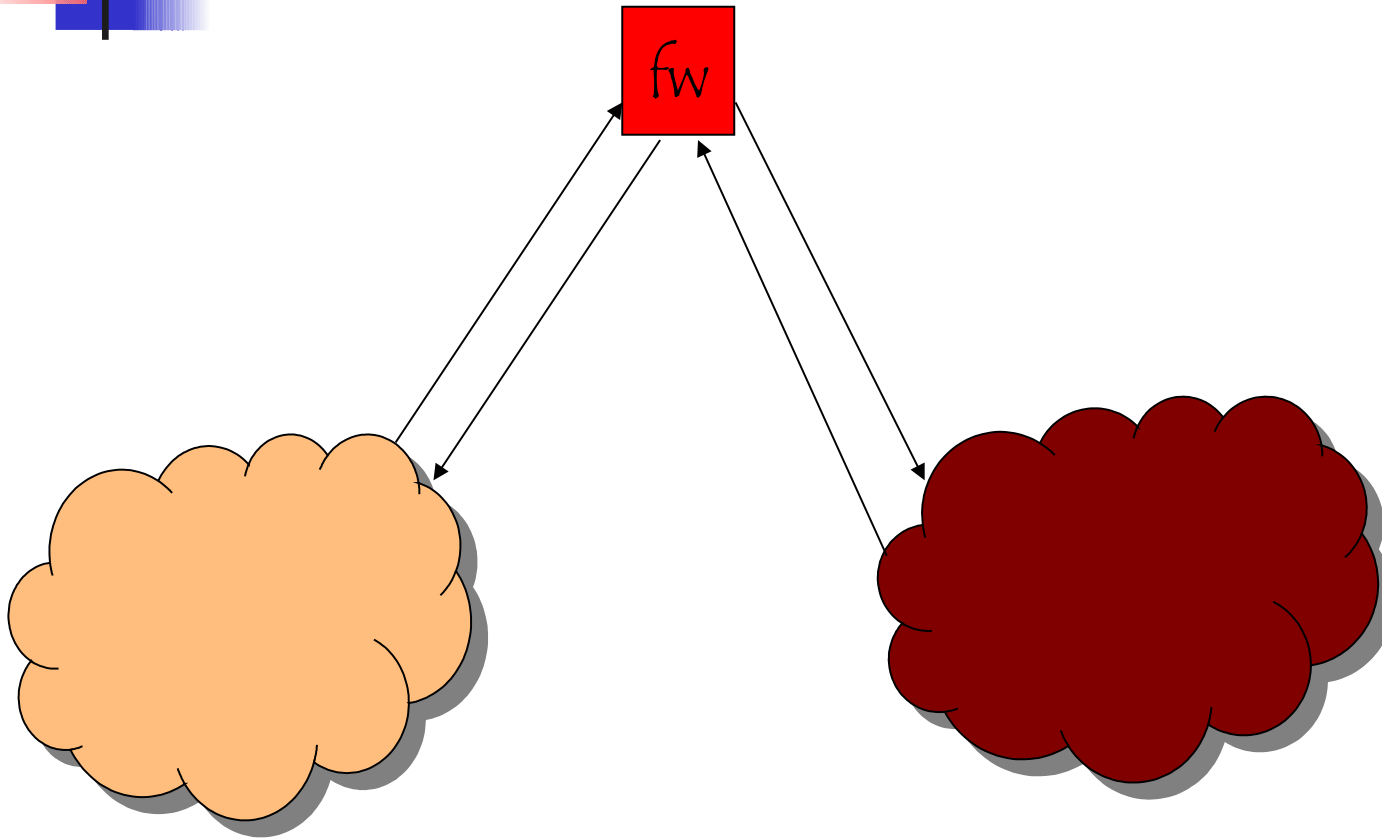
A physical connection does not exist

Some architectures - 3

fw

routers

A physical connection does not exist
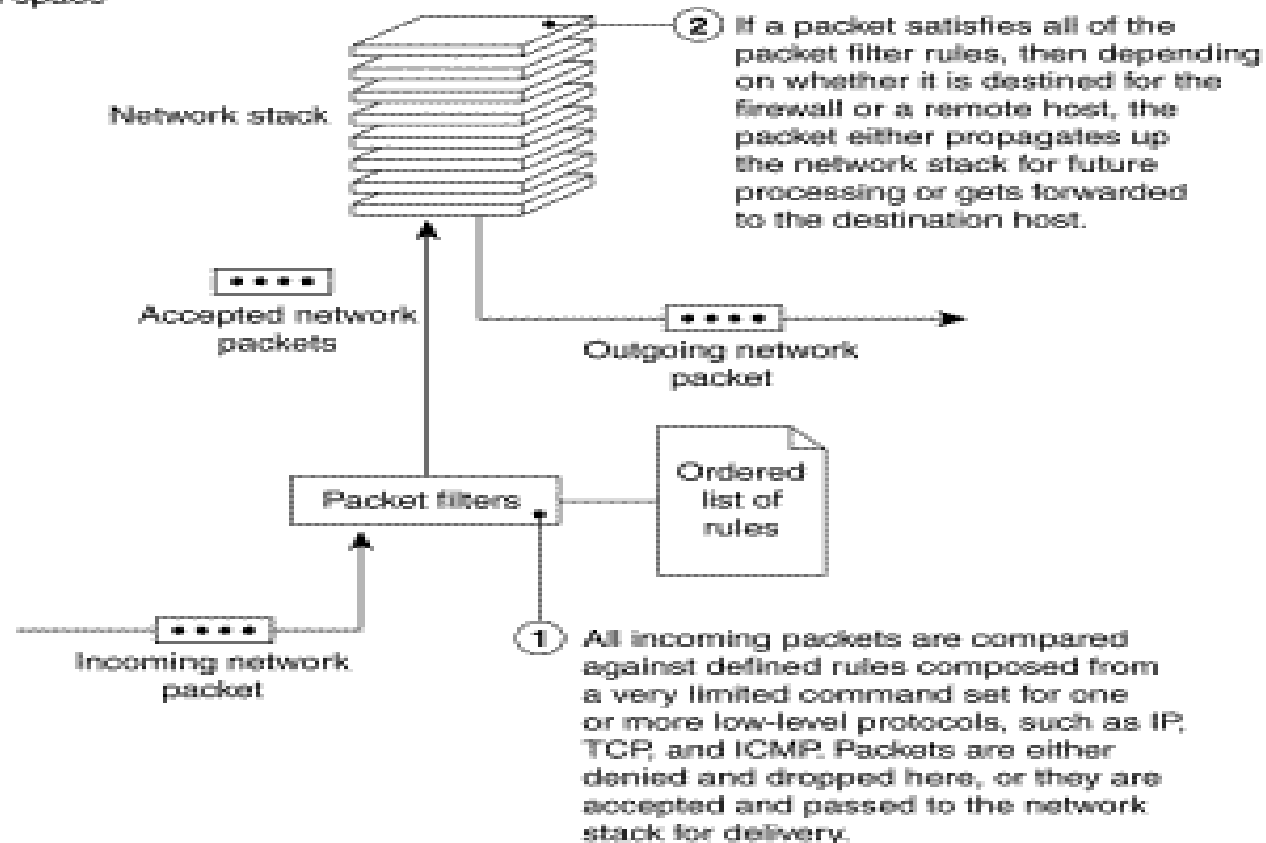And two components have to be attacked

Some architectures - 4

- In the simplest case we have a router with some ACL (already seen in S&S) rather than a node = a layer 3 firewall that can examine ports and hosts

- A layer 3 firewall does not remember the history of a connection but can prevent that an outbound connection is opened by checking the bits in an IP packet

- It can be also implemented by a dedicated system or a system with other functions, eg a Linux node plus netchain and/or iptable
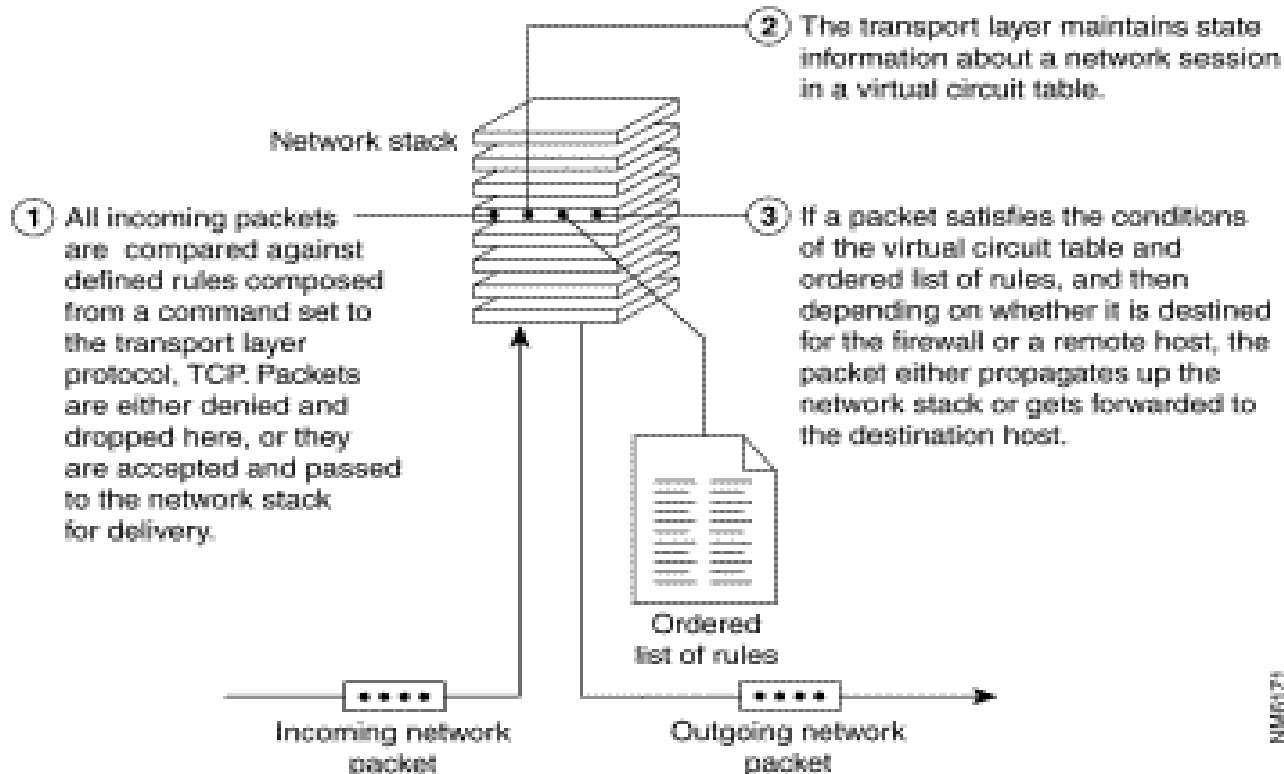
Packet filtering

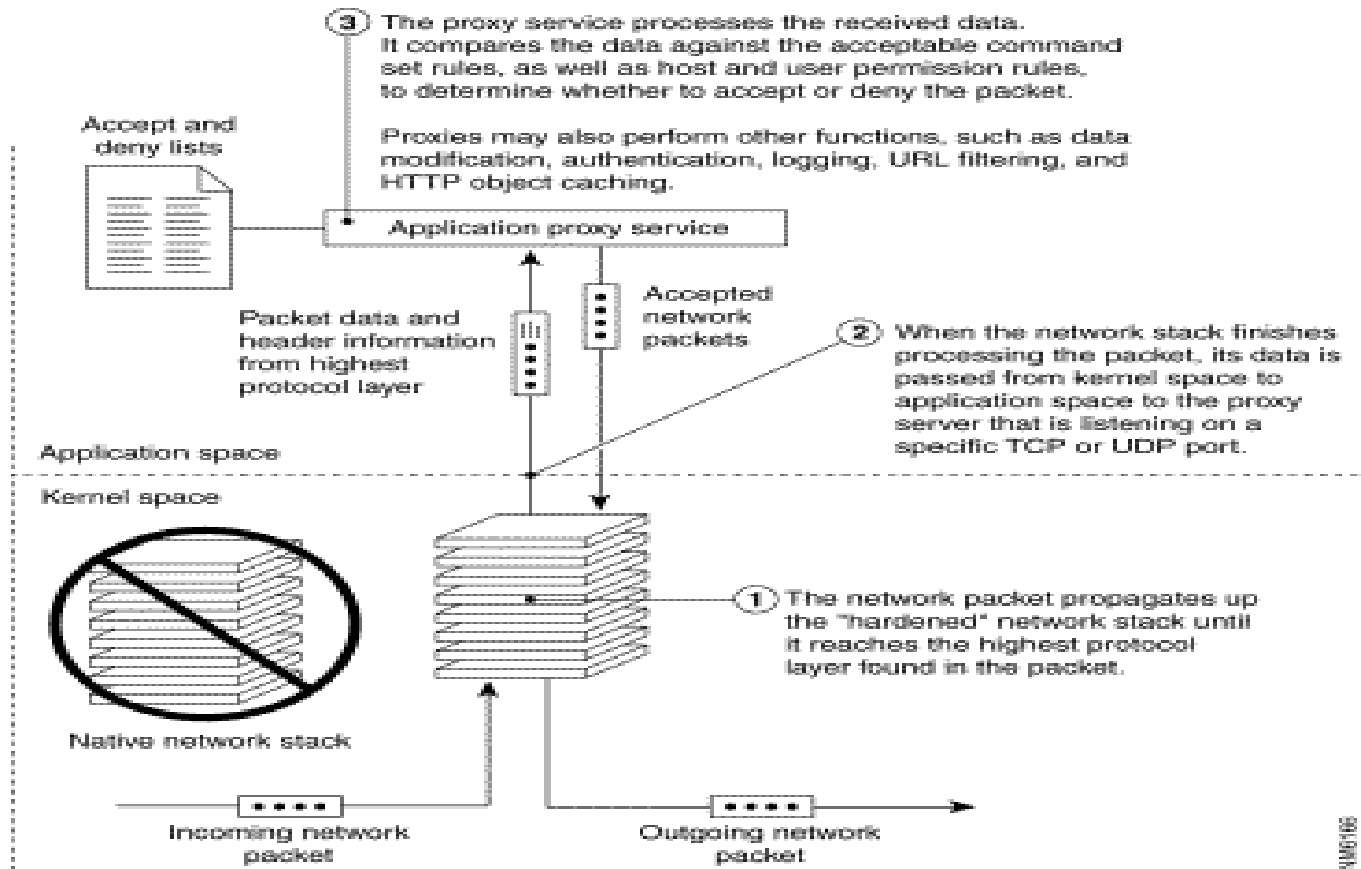
Application space

Kernel space

Network stack

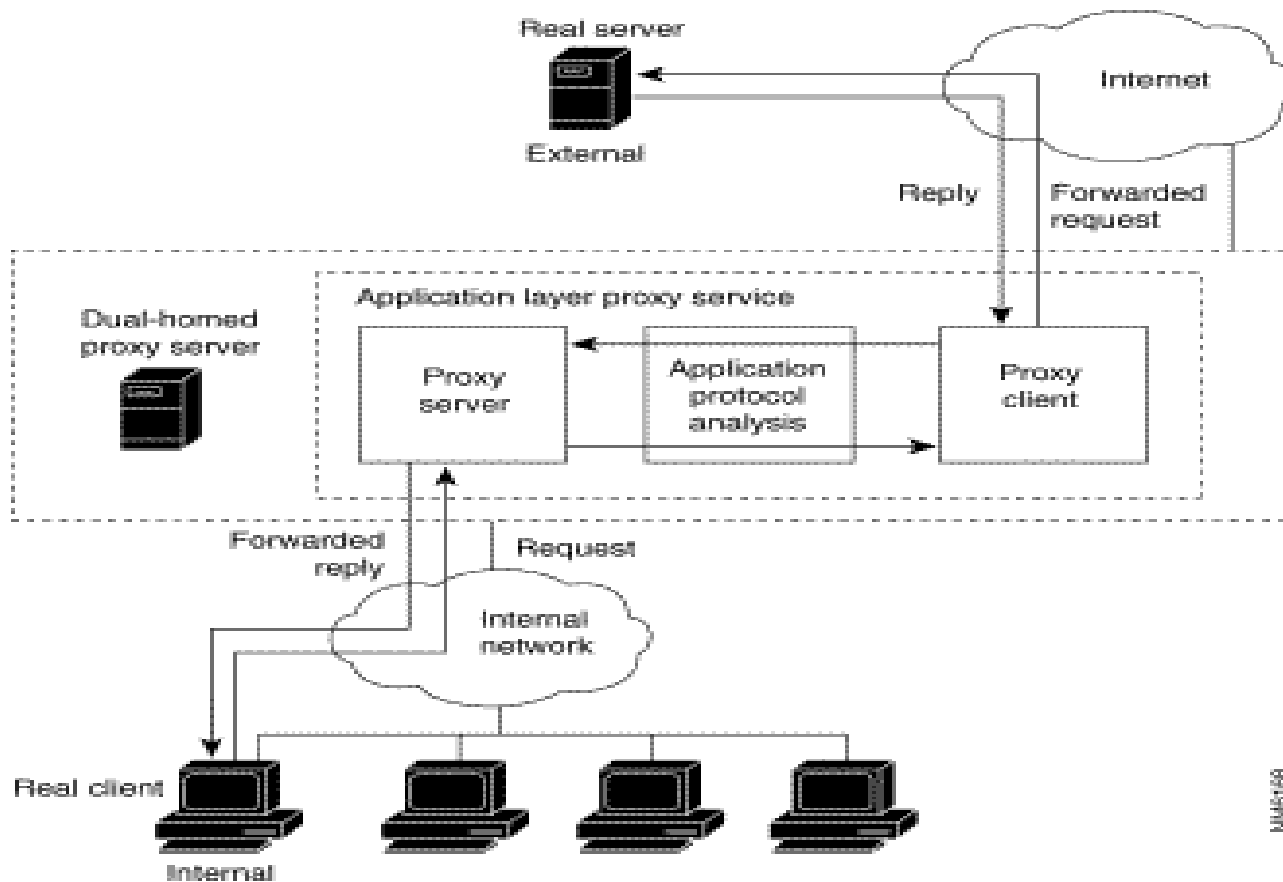② If a packet satisfies all of the packet filter rules, then depending on whether it is destined for the firewall or a remote host, the packet either propagates up the network stack for future processing or gets forwarded to the destination host.

Accepted network packets

Outgoing network packet

Packet filters

Ordered list of rules

Incoming network packet

① All incoming packets are compared against defined rules composed from a very limited command set for one or more low-level protocols, such as IP, TCP, and ICMP. Packets are either denied and dropped here, or they are accepted and passed to the network stack for delivery.

MM6172

Circuit level

Application space

Kernel space

Network stack

① All incoming packets are compared against defined rules composed from a command set to the transport layer protocol, TCP. Packets are either denied and dropped here, or they are accepted and passed to the network stack for delivery.

② The transport layer maintains state information about a network session in a virtual circuit table.

③ If a packet satisfies the conditions of the virtual circuit table and ordered list of rules, and then depending on whether it is destined for the firewall or a remote host, the packet either propagates up the network stack or gets forwarded to the destination host.

Ordered list of rules

Incoming network packet

Outgoing network packet

NM6171

Application Level

Proxy service

Proxy Server

- A server that acts as a "proxy" for an application by making a request on the Internet in its stead.

- whenever a user uses a client application configured to use a proxy server, the application will first connect to the proxy server and give it its request.

- The proxy server then connects to the server and sends that server the request.

- The proxy server can act as a cache for frequent requests

- Next, the server gives its reply to the proxy, which then finally sends it to the application client

Proxy Server

- Using a proxy server, connections can be tracked by creating *logs* for systematically recording user queries when they request connections to the Internet

- Internet connections can be filtered, by analysing both client requests and server replies.

- comparing a client's request to a list
  - of authorised requests= *whitelisting*,
  - of forbidden sites= *blacklisting*.

- Analysing server replies that comply with a list of criteria (such as keywords) is called *content filtering*

Reverse Proxy Server

- A proxy server that, rather than allowing internal users to access the Internet, lets Internet users indirectly access certain internal servers

- It protects the web server from direct outside attacks, which increases the internal network's strength.

- A reverse-proxy's cache function lowers the workload of the server it is assigned to, and for this reason is sometimes called a *server accelerator*.

- The reverse-proxy can distribute the workload by redirecting requests to other, similar servers (*load balancer*)

Dynamic filtering

(5) All network packets associated with an authenticated session are processed by an application running on the firewall host.

Authentication daemons

Application space

Kernel space

Network stack

(4) If a packet satisfies all of the packet filter rules, then depending on whether it is destined for the firewall or a remote host, the packet either propagates up the network stack for future processing or gets forwarded to the network host.

Accepted network packets

Outgoing network packet

(2) Based on information contained within each packet, each packet is associated with additional state information.

Packet filters

Dynamic rules

Ordered list of rules

(3) Dynamic rules are added and removed based on a combination of the data contained within the network packet and the state information.

Incoming network packet

(1) All incoming packets are compared against defined rules composed from a very limited command set for one or more low-level protocols, such as IP, TCP, and dropped here, or they are accepted and passed to the network stack for delivery or they are accepted.

ACL & Router

- Lista costruita a partire da due casi fondamentali

  Range$_1$ IP $\rightarrow$ route

  Range$_2$ IP $\rightarrow$ drop

- Posso avere una lista per ogni interfaccia di ingresso/uscita in modo da controllare il traffico che attraversa il router anche se non conosco gli indirizzi di chi genera il traffico che attraversa il router

ACL & Router

- ACL di ingresso 1
  - 131.114.*.* $\rightarrow$ route
  - 131.4.5.6 $\rightarrow$ route
  - 131.4.*.* $\rightarrow$ drop

  dalla rete 131.4.*.* solo il traffico da 131.4.5.6 può passare il resto viene eliminato

- ACL di uscita 1
  - 131.114.*.* $\rightarrow$ drop
  - 131.4.*.* $\rightarrow$ drop

  nessun nodo 131.4.*.* può comunicare con la rete collegata all'interfaccia 1

ACL & Router & Linux

- Si possono determinare le regole di filtraggio dei pacchetti in Linux mediante
  - Netfilter nel kernel
  - Iptables per configurazione a livello superiore
- Filter table: determina quali pacchetti passano mediante le catene nel seguito
- Nat table: riscrittura di alcuni campi
- Mangle table: opzioni su QOS

Azioni possibili

- Drop
- Route
- Return – ritorna a catena chiamante
- Queue – passaggio in spazio utente
- Log
- Reject
- Dnat/Snat/Masquerade

Countermeasures – Resist & Recovery

- Defence-in-depth
  - A network is segmented into several subnetworks, each with a security level
  - Networks with consecutive security levels only are connected
  - Any connection from a network to another one is protected by a firewall
  - Physical node connections may have to be updated

Defence-in-depth

Local network

E-mail,
web server

Accou
nting

Resear
ch

fw

fw

fw
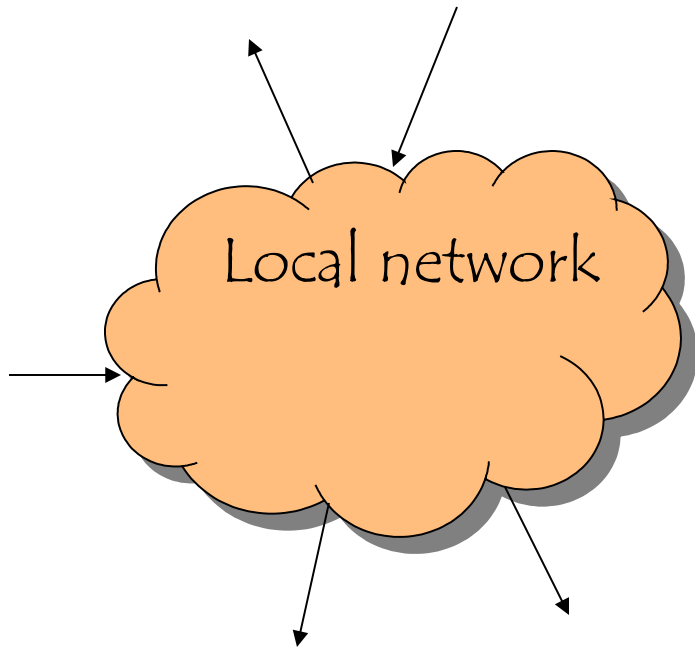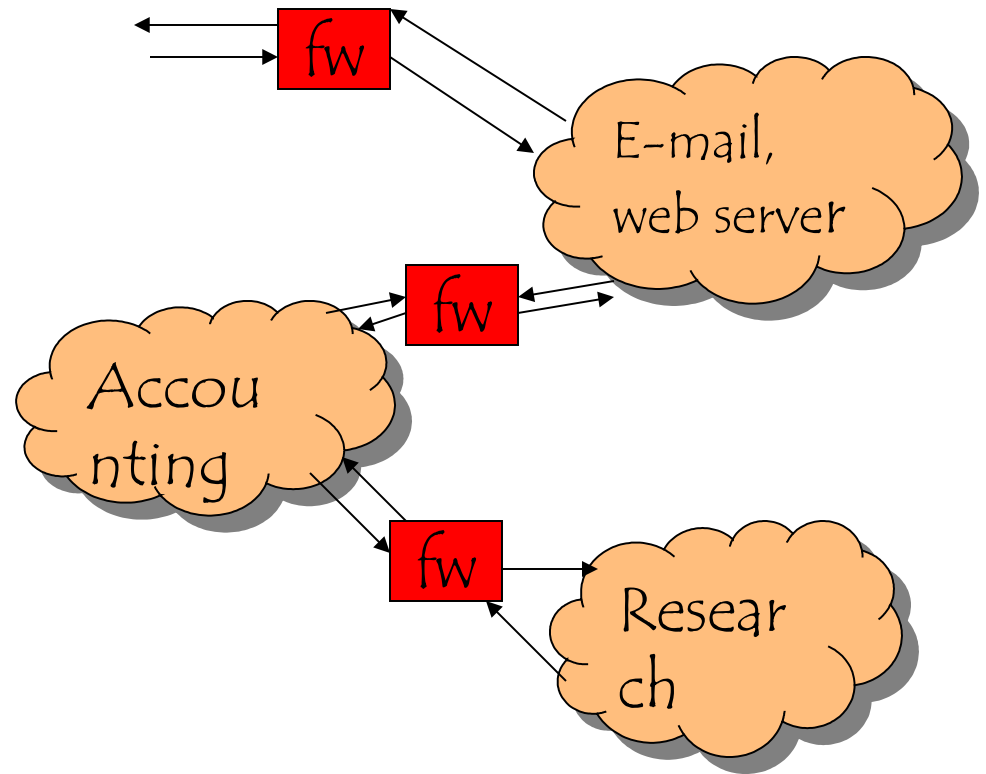
Initial configuration   Defence in depth

Firewall & Virtual Machine

- Virtualization technology supports the definition of virtual network (overlay network)
- Information is spread across a large number of nodes and of networks and by mapping the corresponding virtual nodes onto distinct physical nodes
- Checks are more rigorous as sharing may be minimized
- Virtual networks are protected by (virtual) firewall
- The ability of introducing several nodes and distinct networks simplify information management as each network can manage a low amount of homogeneous information from a security perspective

=micro segmentation, micro defence in depth

Countermeasures – Personal Firewall

- Initially , the target of the attack where the server systems

- Currently attacks are complex (eg sequences of attacks) and one of the target of an intermediate step may be a client system, eg to steal information used to authenticate users

- A personal firewall is a software component to protect the client and the information exchange between the client and the server

- A special purpose application may be useless because the ability of defining a virtual network makes it possible to protect the applications running on a client system through standard components

Personal or real firewall?

Client node

It can be decomposed into a virtual network

firewall

Client Virtual Node

Virtual machines
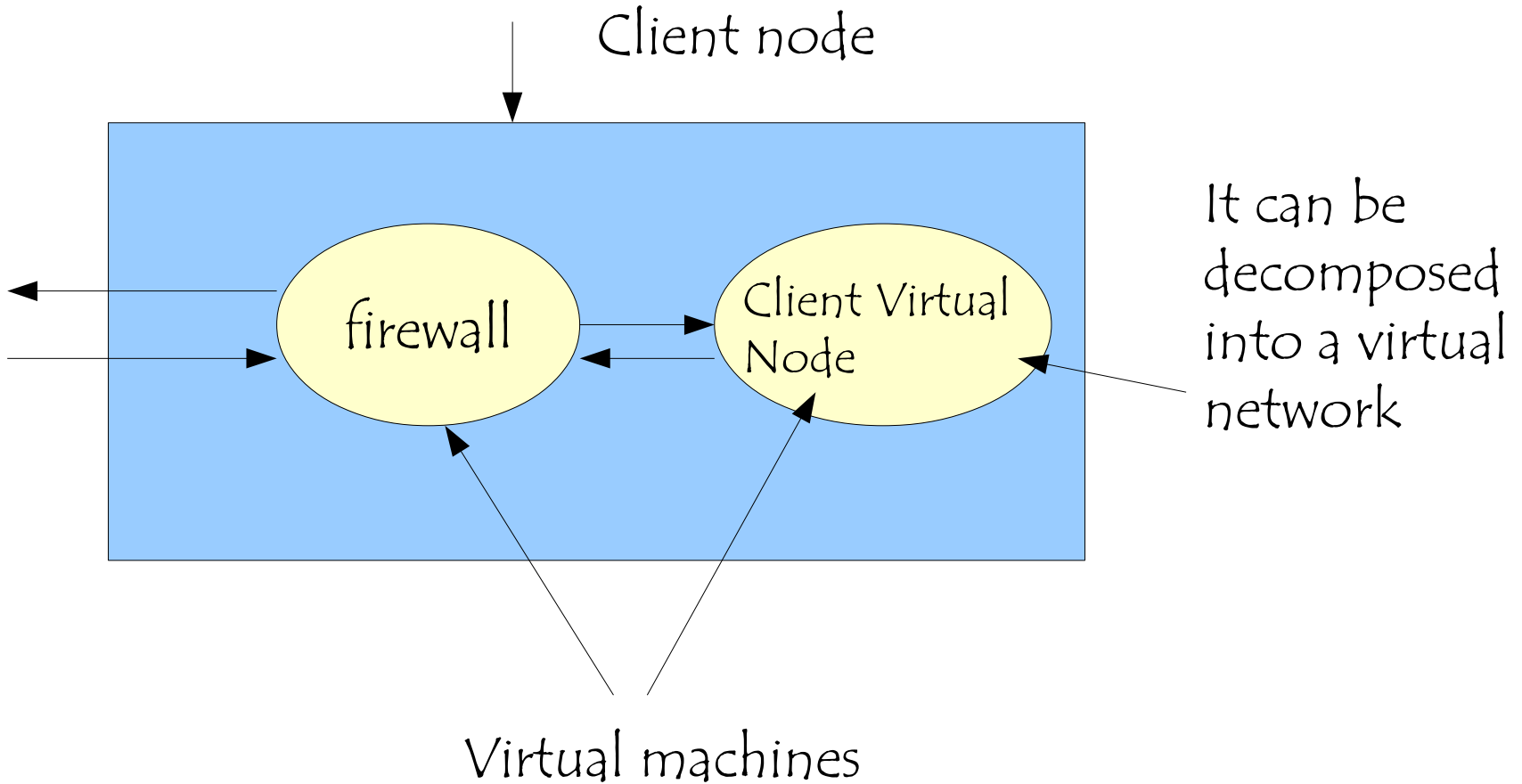
Countermeasure - Detect

- Discover attacks against a node
- There are two cases of interest
  - Discover ongoing attacks = discover a malware trying to attack a node
  - Discover malware that has been installed on a node after a successful attack
- There are alternative strategies to discover events of interest

Countermeasures - Detect
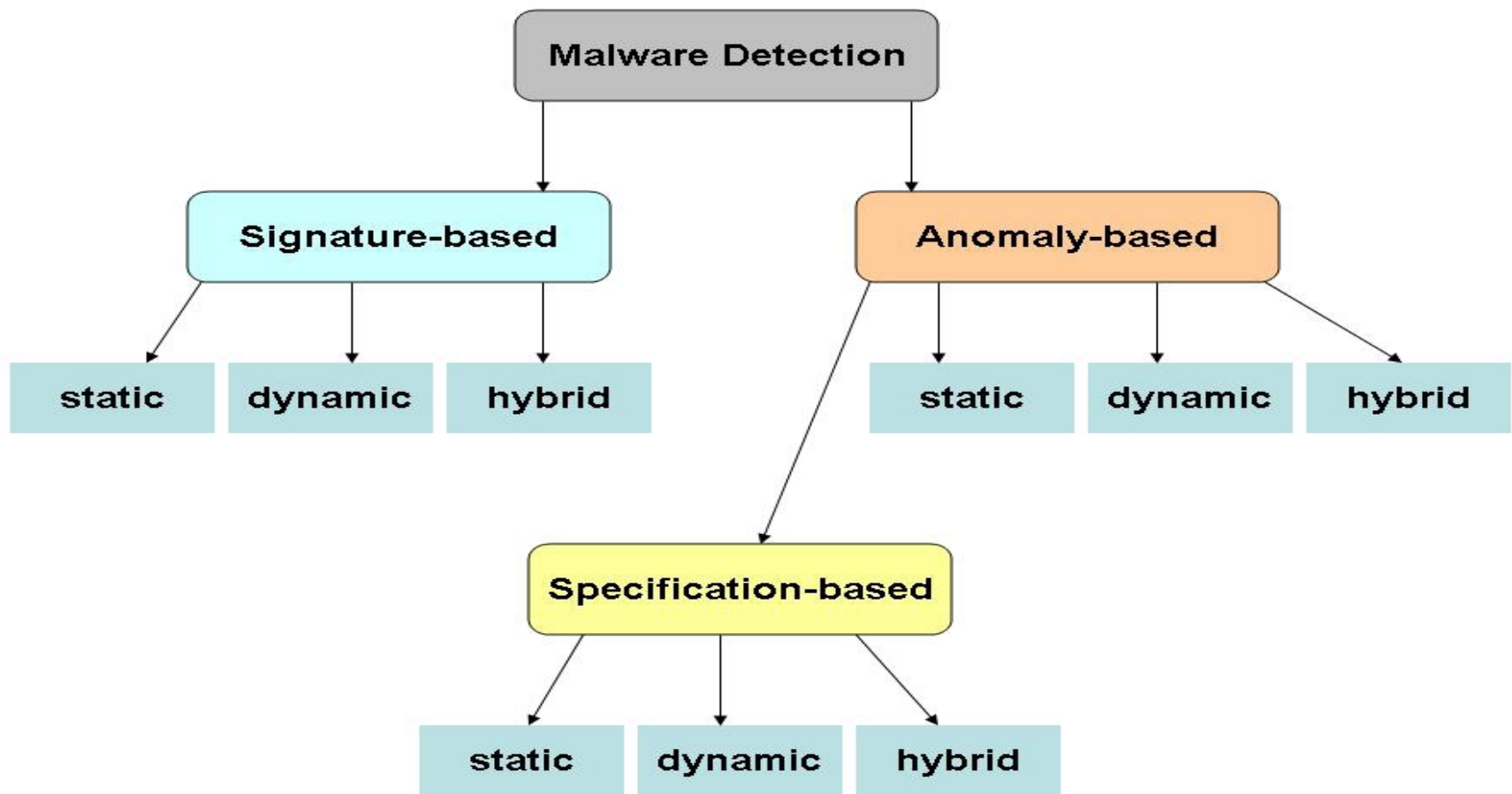
Detection – Anomaly Based

- The behaviour of the system to be protected is observed for an interval of time (learning the normal behavior)

- After the learning, any behavior that is too "distant" from those that have been observed is signalled as an anomaly

- The critical element is the amount of information on the system acquired in the learning phase

Detection – Anomaly Based

- Dynamic
  - Information on a program behavior is collected to discover attacks against it
- Static
  - Information on the structure of a program or of file record are collected
- Hybrid
  - The expected behavior of the program is compared against the actual one

Detection – Anomaly Based

A = set of all behaviors
V = set of all valid behaviors
Vapprox = approximation to V

V

Vapprox

V'

A
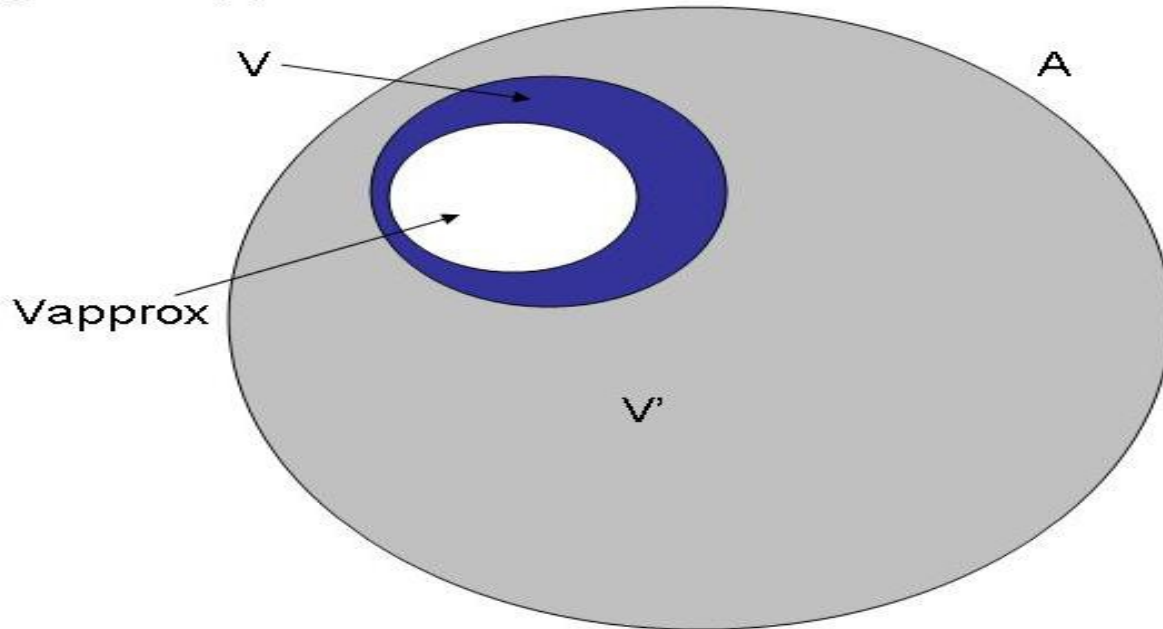
In general the information that is collected makes it possible to approximate the behavior of interest

Detection – Specification Based

- Normal behaviors are not learned, instead they are specified by the security policy

- Dynamic
    - Information on the program behavior are collected and compared against the program specification

- Static
    - A program is statically analysed and the results are compared against the specification

- Hybrid
    - The program compilation returns some specification to be compared against the program behavior

Detection – Signature Based

- Main idea: there are some behavior that fully characterize and identify a malware, they are a signature of the malware

- All the signatures are collected in a database that drives the detection. This poses two problems
    - The discovery of a signature
    - The update of the database

- A malware can be discovered only if its signature is known = a 0-day exploit cannot be detected = new attacks can be discoverd, only if an anomaly detection approach is being implemented

- Alternative strategies can be adopted to define the signature

Detection – Signature Based

- A default allow strategy, anything that is different fro a signature is allowed

- Dynamic
  - Information on the program behaviour are collected and compared against the signature

- Static
  - The program code is analyzed and compared against the signature
  - Used by antivirus tool

- Hybrid
  - The two approaches are merged: a subset of the programs is selected by a static analysis and the behaviour of these programs is monitored

Detection

- Which events are used to define a signature
- Events local to a node
  - OS calls
  - File operations
- Global network events
  - Messagges
  - Protocol events

Detection

- Intrusion Detection System
  - It monitors either a host (host IDS) or a subnet (network IDS) to detect attacks
  - It integrates with a firewall to detect
    - Attacks from the outside that escape the firewall
    - Insider attacks that the firewall cannot prevent
  - Unstable technology

IDS, false positive, negative...

- The behavior that the tool detects are an approximation of those of interest. This implies that some statistic notion may be very useful

- The problem arises because we do not have a perfect test to discover if a system is being or has been attacked

- There is a set of symptoms (behavior) that suggest that the system has been or is being attacked

- However, we are not sure of the attack

False, true positive etc

- We define a test to discover whether some one is ill

- 4 cases are possible
  - Test positive, illness   = true positive
  - Test positive, no illness = false positive
  - Test negative, illness   = false negative
  - Test negative, no illness = true negative

The ideal test
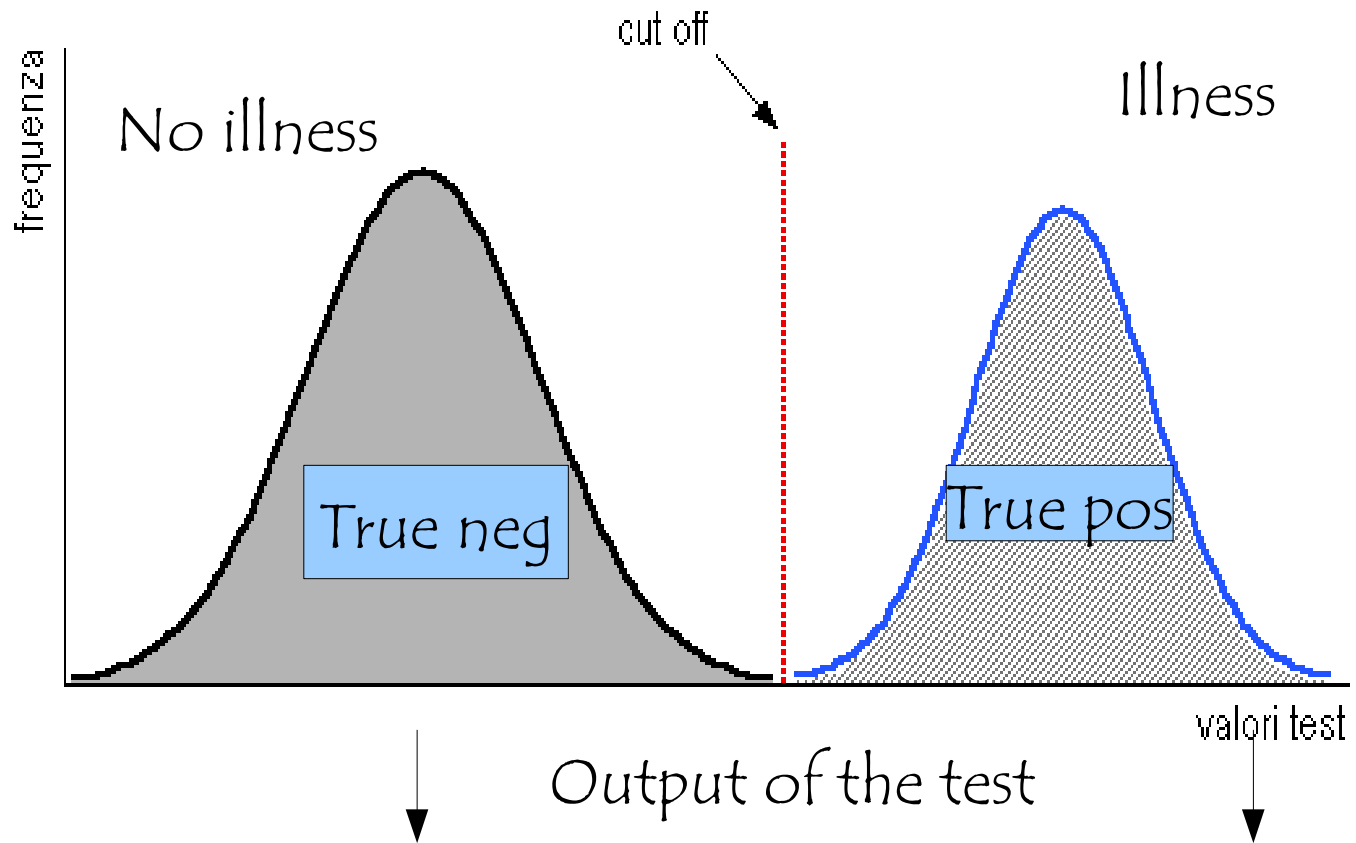
Probability distribution
of the parameter, no illness

Probability distribution
of the parameter, illness

Any real test
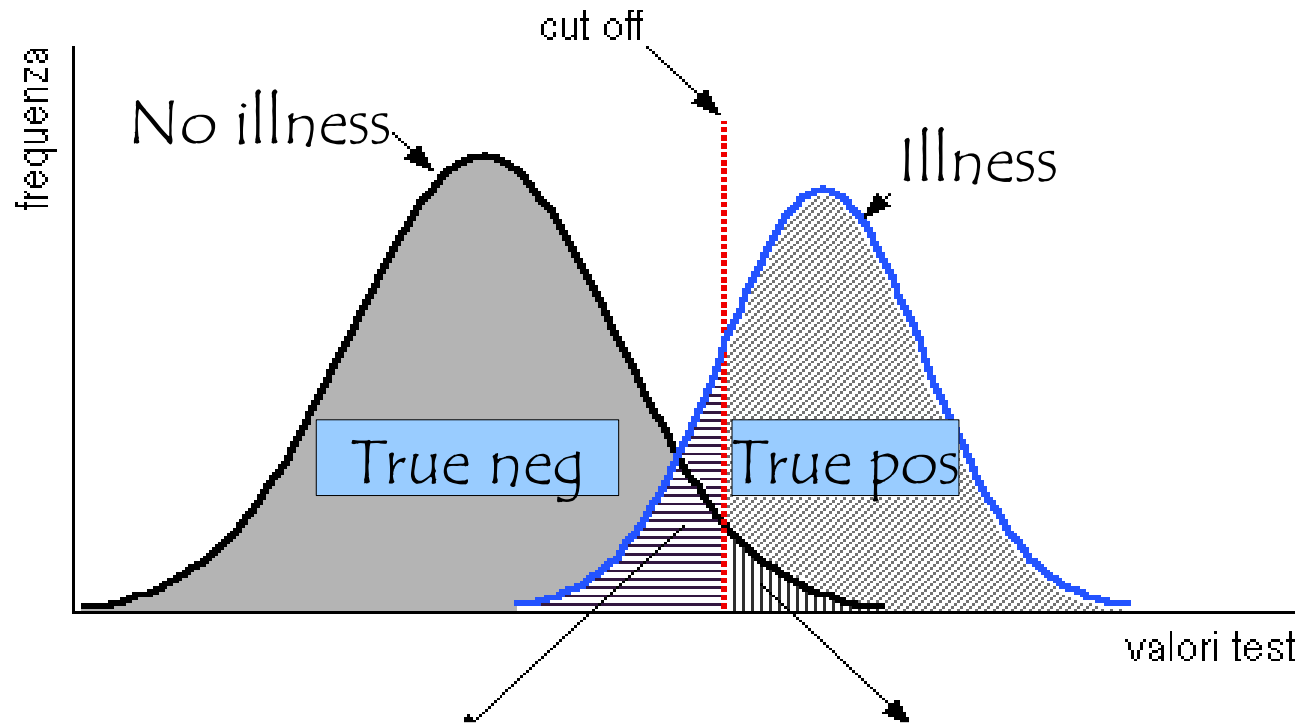
False negative false positive

Any real test

False positive = false alarms

False negative = missed intrusion

True neg

True pos
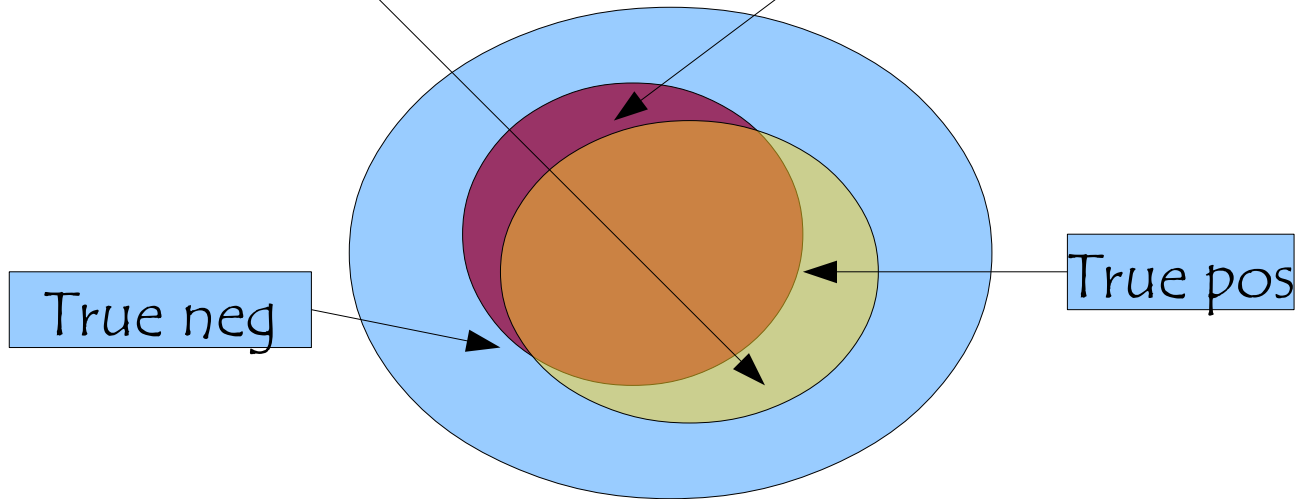
Possible behaviors   The test   Intrusions

A real test

True positive

GOLD STANDARD

False positive

Ill | No ill

	a	b
Test +		
Test -	c	d
	a+c	b+d

Ideal test = perfect knowledege

A real test

False negative

True negative
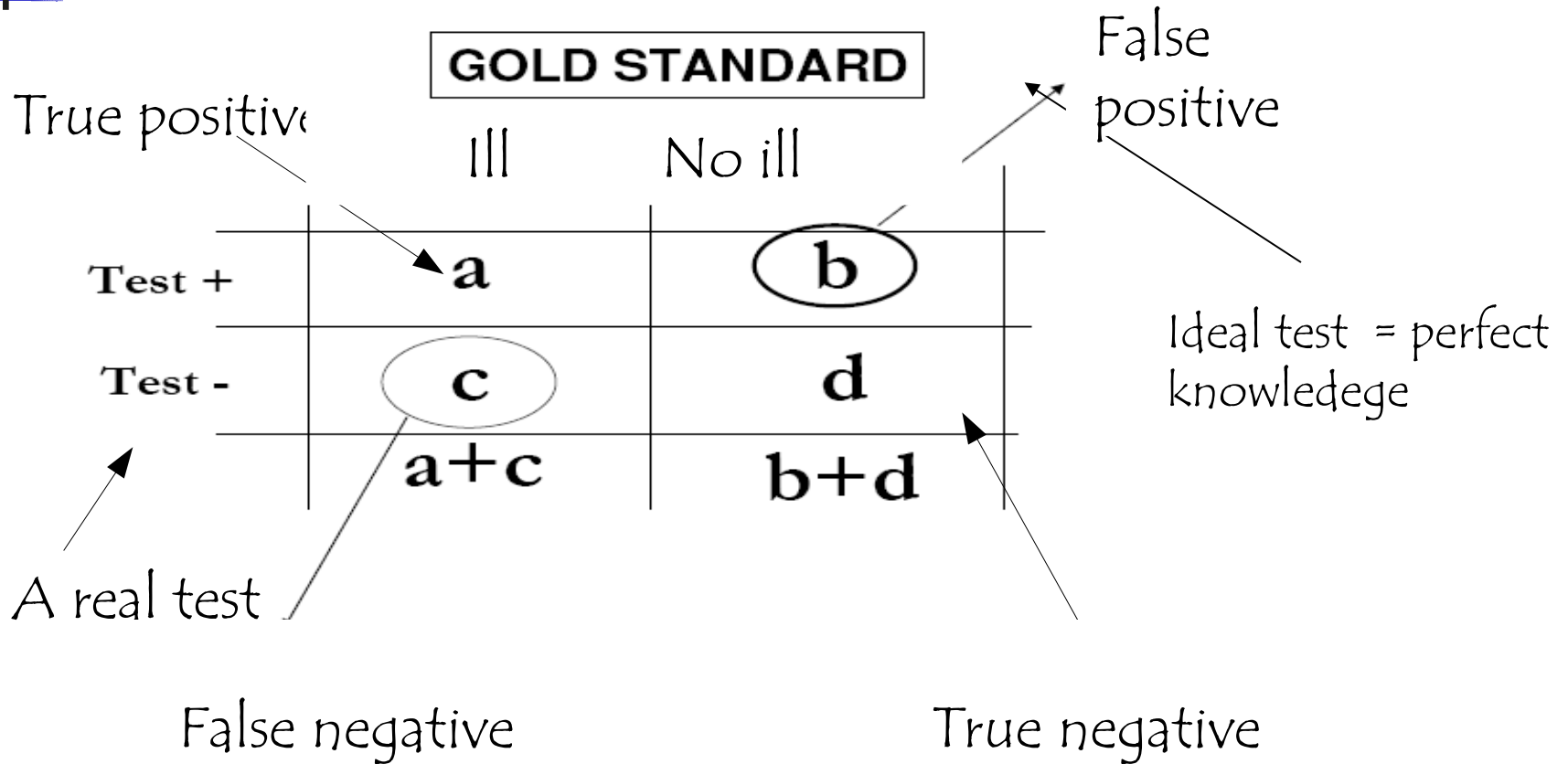
Another case: biometrics

BIOMETRICS COMPARISON CHART

Biometric	Verify	ID	Accuracy	Reliability	Error Rate	Errors	False Pos.	False Neg.
Fingerprint	✔	✔	☺☺☺☺	►►►	1 in 500+	dryness, dirt, age	Ext. Diff.	Ext. Diff.
Facial Recognition	✔	✘	☺☺☺	►►	no data	lighting, age, glasses, hair	Difficult	Easy
Hand Geometry	✔	✘	☺☺☺	►►	1 in 500	hand injury, age	Very Diff.	Medium
Speaker Recognition	✔	✘	☺☺	►	1 in 50	noise, weather, colds	Medium	Easy
Iris Scan	✔	✔	☺☺☺☺	►►►	1 in 131,000	poor lighting	Very Diff.	Very Diff.
Retinal Scan	✔	✔	☺☺☺☺	►►►	1 in 10,000,000	glasses	Ext. Diff.	Ext. Diff.
Signature Recognition	✔	✘	☺☺	►	1 in 50	changing signatures	Medium	Easy
Keystroke Recognition	✔	✘	☺	►	no data	hand injury, tiredness	Difficult	Easy
DNA	✔	✔	☺☺☺☺	►►►	no data	none	Ext. Diff.	Ext. Diff.

Sensitivity

Sensitive = probability of a positive answer in an ill person

Test +	a
Test -	c
	a+c

$$Sen = pr(T^+|M^+)$$
$$Sen = a/(a+c)$$

Specificity

Specifity= probability of a false answer if no illness

	No ill
Test +	b
Test -	d
	b+d

$$Spe = pr(T^- \mid M^-)$$
$$Spe = d/(b+d)$$

Likelyhood

$$LR^+ \equiv \frac{P(T^+|M^+)}{P(T^+|M^-)} = \frac{Se}{1 - Sp} \qquad LR^- \equiv \frac{P(T^-|M^+)}{P(T^-|M^-)} = \frac{1 - Se}{Sp}$$

LR+= ratio between the probabilities of a positive test in one ill and one healthy person

LR-= ratio between the probabilities of a negative test in one ill and one healthy person

Evaluating rules to detect intrusion
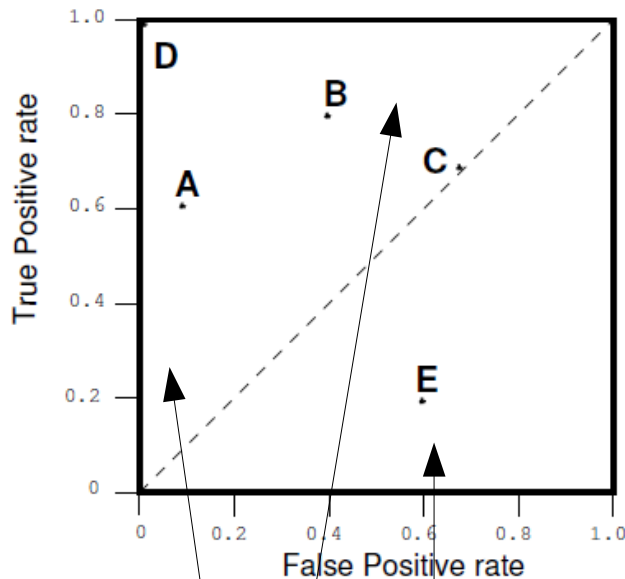
To each rule to detect an intrusion
• it sends at least x Mb/sec
• It open at least x connection in a sec
we can pair a point in this space according the probability of false and true positive for each value of x.
As x changes, we have a curve in ROC space

A rule low and left =   conservative low number of false positives but also a low detection capability

A rule high and right =   good detection capability at the expense of a lot of false positves and few true positive

A rule under the bisector = worse than random (= the bisector) it can be improved by negating it

ROC curve
receiver operating characteristics

$Se \equiv P(T^+|M^+)$
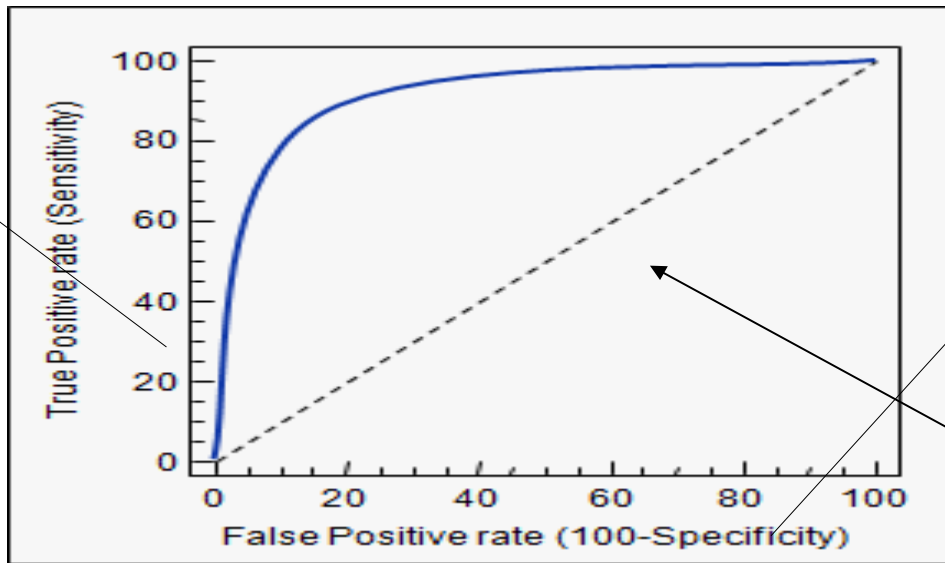
$Sp \equiv P(T^-|M^-)$

Sensitivity vs 1-specificity

Random answer

The curve is drawn by considering a rule that depends upon a parameter for distinct values of the parameter (it opens x connections in a second)
Each value results in a percentage of false and true positives
The bisector corresponds to a rule that chooses at random
Rule can be evaluated according to the surface they define, the larger, the better
No curve can be worse than the bisector because we can define a curve better than the bisector by negating the rule

Applying ROC to select a strategy

Best solution
Always higher
Than the others

C3/C4 (0.02;0.95)

sensibilità

1 - specificità
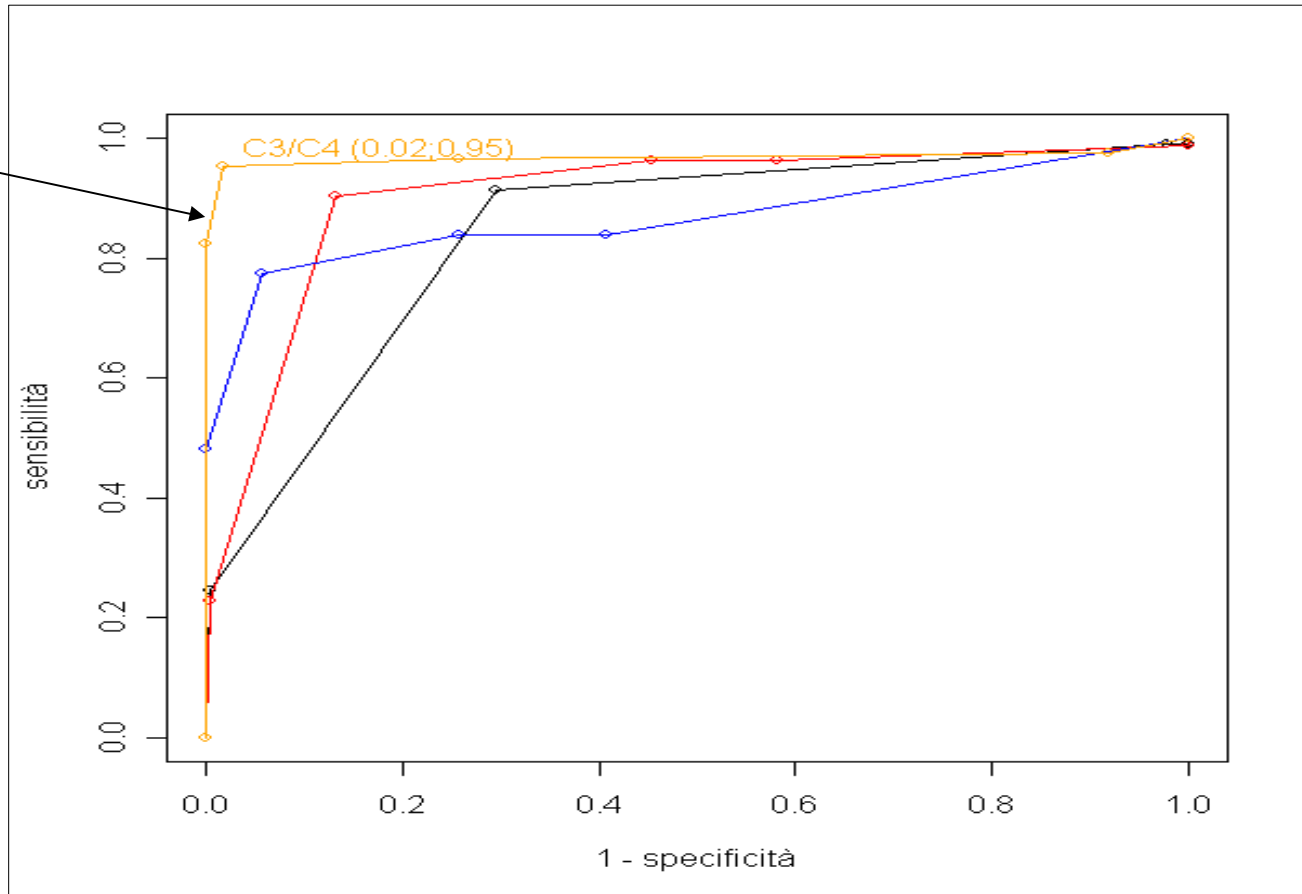
Pay attention to the population size

- When considering an IDS the number of "people" to be tested is fairly larger than in the case of a medical test

- A test that produce a false positive with a probability equal to 10^{-6} is almost ideal in the medical field

- The same test, if applied to a network that transmits 10^9 IP packet in one day, returns about 100 false positive a day, about 5 false alarms for each our = the test is useless

Host IDS

- It monitors a single host
- It checks system and user process to discover
  - OS commands that have been changed
  - Attackers that impersonate legal users
  - Attacks against the host
- Base mechanisms to define a monitor:
  - Interception of OS calls

    then either
  - Analyze the call

    or
  - Produce a log with the calls and analyze it

Network IDS

- It monitors the network segment inbetween two switches (a collision domain)

- The monitoring has to detect anomalous or dangerous traffic

- The basic mechanism is sniffing, the same one used by an attacker

- A dedicated host should be used for both performance and security

NIDS + HIDS

- The two tools can cooperate through a distinct interconnection network

- The real problem is how much one tool can trust the other (mutual trust)

- The host running a tool may be attacked and controlled by the attacker

NIDS+ HIDS = IDS = sensors+ engine

- The most coherent perspective consider a set of sensors and an inference engine
- Each sensor monitors some components and transmits information to the engine
- The engine applies a set of rules to the input from the sensors to detect intrusions
- The communication among the engine and the sensors exploits a segregated connection network
- It is important to determine whether two events are independent because if several independent events signal an intrusion, then the probability of a true positive increases
- Danger model = inspired by biology, rules that produces a larger number of false positive may be applied as the probability of an intrusion increases

IDS

- In any case, the adoption of an IDS has to be trasparent for the user

- In several cases, the users should not to be aware that an IDS has been adopted (it can discover insider threats)

- Legal problems
  - According to the italian law the adoption of any tool that can be used to monitor a worker has to be authorized by trade unions

IDS

- Which actions can be automatically taken as soon as an IDS discover an attack?
  - It is correct to take action on the target system: kill an internet connection increase the amount of data that are recorded in a log, ends some user sections
  - No action should be taken against other systems, eg the attacker one, for two reasons:
    - Stepping stones
    - False positives

Intrusion Detection System

nIDS

fw

E-mail web

fw

nIDS

Accoun ting

fw

Research
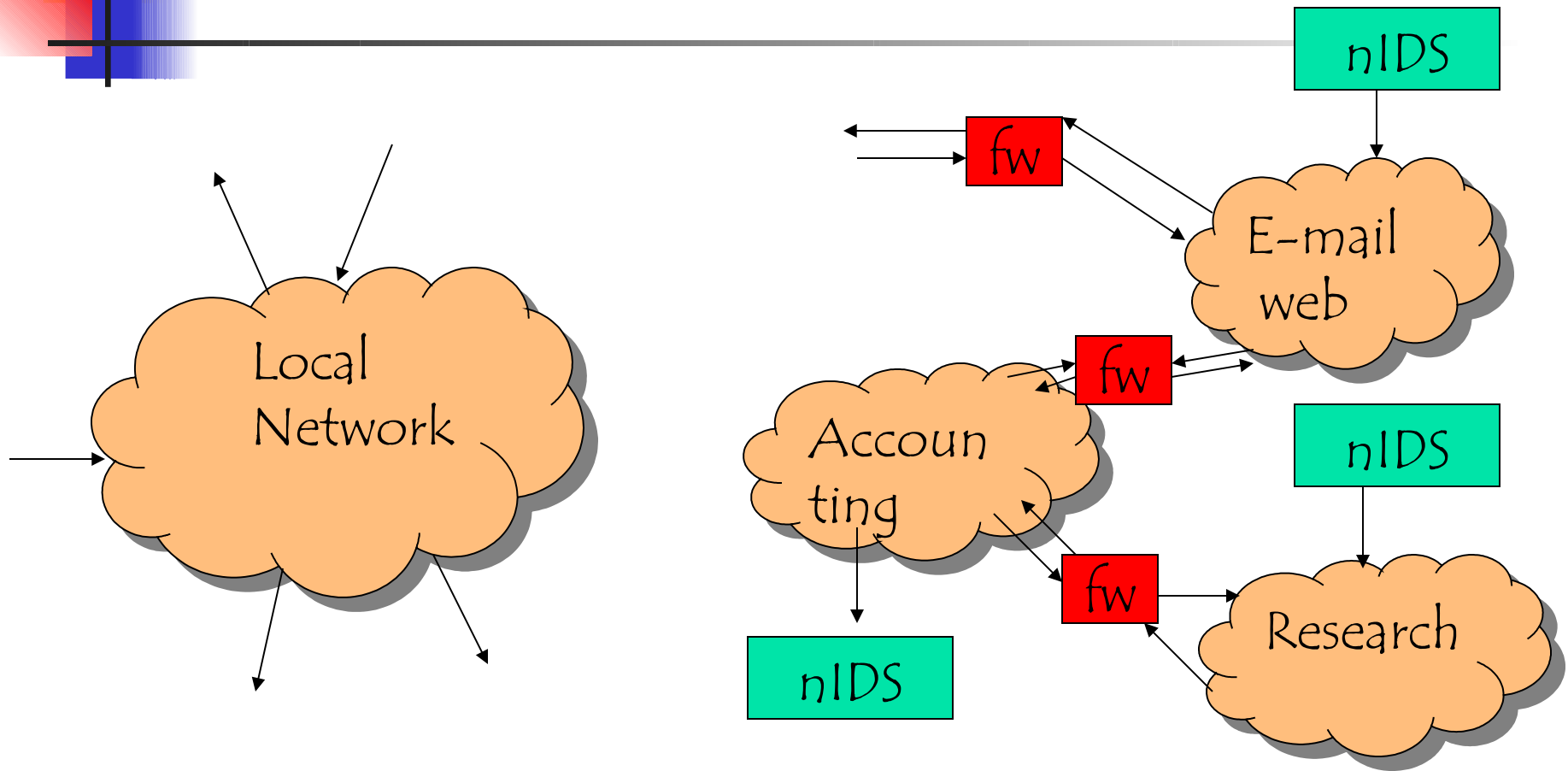
nIDS

Initial configuration

Segmentation+Defence in Depth+IDS

Local Network

Intrusion Detection System

nIDS

fw

nIDS

fw

fw

nIDS

Segmentation+Defence in Depth+IDS
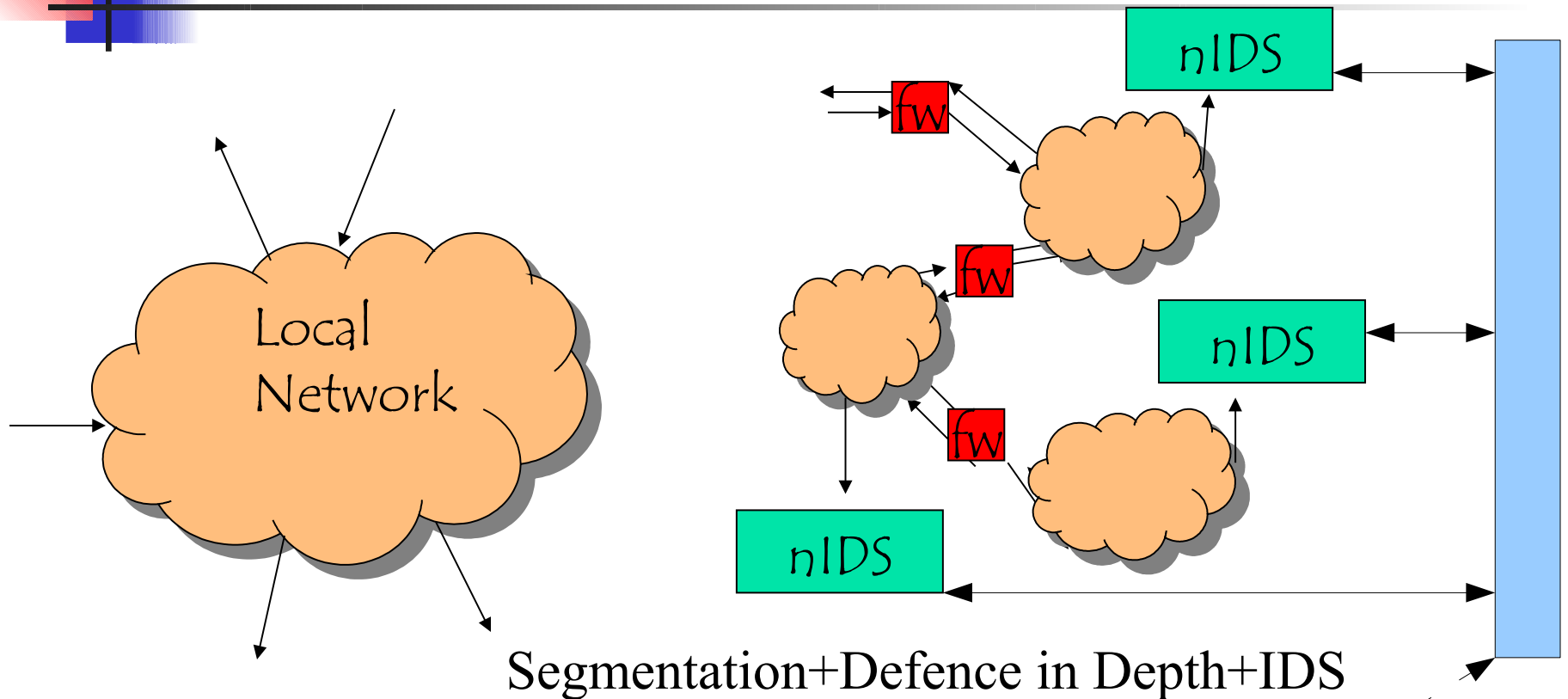
Initial configuration

Cooperation among nIDS

IDS+Virtualization

We can insert IDSs in virtual networks

- Increase the number of controls
  - When crossing a network
  - Within a, virtual, network
- By reducing the number of nodes within a virtual network we can control in a more rigorous way the traffic and the protocol in the network

Sensors

- Two kind of sensors
  - off-line: they analyze the system and user logs to discover attacks that have been implemented and their impact
  - real-time: they analyze the current system behavior to discover ongoing attacks and stop them before they are successfull
- real time
  - Some compromises have to be accepted = minimize the number of control to avoid a loss of performance
  - Hardware supports, eg similar to the routing one for NIDS
- Off line = CIDF, common intrusion detection framework standard for logs

NIDS vs HIDS sensors

- hIDS
  - It filter the requests from a user process to the OS, the OS executes only requests that have not been rejected
  - It may slow down a host but any request is controlled
- nIDS is not involved in the service that manages a given packet, there is no way to slow down the receiving host

  $\Rightarrow$ NIDS has to be executed on a dedicated host to analyze all the information flows

hIDS and nIDS technologies

- Anomaly detection
  - By observing a system, a database is built that stores the normal system behavior
  - Behaviors that differ more than a predefined threshold are signalled
  - Zero day exploit
- Signature specification based
  - Default allow (attack signatures have to be specified)
    - A database storing attack signatures
    - At run time any behavior matching one in the database is signalled
    - The update of the database is critical
  - Default deny = legal behavior has to be specified

hIDS and nIDS technologies

- Base element that is analyzed
  - IP packects and protocol events for a nIDS
  - OS call for a hIDS
  - They can be generalized if the hierarchy of virtual machines is considered
    - String of vm invocations for a hIDS
    - A stream of information for a nIDS

nIDS: some problems

- Fragmentation of IP packets
- Analysis of a TCP stream (reordering ..)
- Protocol analyis
- Normalization of a protocol to handle all those cases that are not defined by a standard (overlapping IP packets)

N&H-IDS: anomaly detection

First step: interesting measures

- Number of open file
  - global & for each user
- Number of open port
  - global & for each user
- Frequency of commands
- Number of connected user
- Time when a user connects
- Usage of system resources

N&H-IDS: anomaly detection

- An histogram is built by observing the system and by using a number of intervals (eg 32)

- The intervals are chosen so that the last one include less than 1% observations

- We monitor the system for a time interval (we observe the value of interest at each minute, for 30 days) and build the distribution that pairs each interval with a probability = long term distribution

- We monitor the system for a shorter interval (eg. at each minute for two hours) and build a short term distribution

- An anomaly arises if the two distributions differs

Generating a distribution

- Defined starting from an histogram of the observations

Number of Observations in the corresponding interval

The probability is computed by normalizing

1 2   31   32
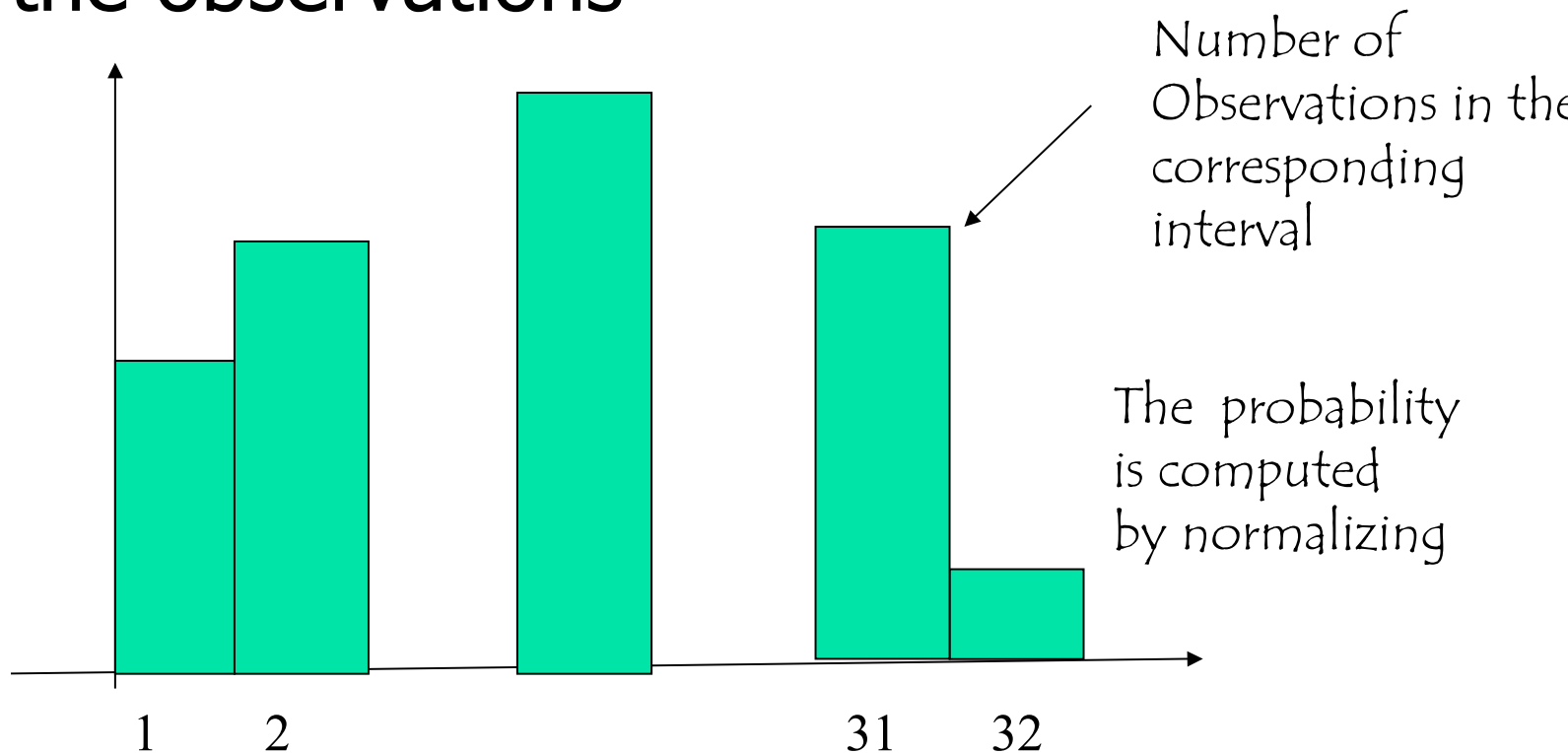
N&H-IDS: anomaly detection

- The difference between two distributions is defined as the sum of the absolute differences between two correspoding intervals

- Several distributions of the same measures can be generated by distinct observation frequency or for distinct cases

  - Open files
    - The number can be read at each minute or at each hour
    - The number can be read for each user or for each group of users

N&H-IDS: anomaly detection

- The IDS raises an allarm anytime the absolute difference is larger than a user defined threshold

- The observations collected to build the short term distribution are used to

  - Discover anomalies and signal attacks

  - Update the long term distribution to mimic the system evolution (a weigthed sum is used)

  - The long term distribution is updated at predefined times (eg at the end of the day) rather than in real time

N&H-IDS: anomaly detection

- The overall system behavior may be seen as a learning system

- Initially, the system learns its normal behavior

- The learning and the discover of anomalous behavior are a life long property of the system

N&H-IDS: anomaly detection

- The definition of anomaly is related to a user defined threshold
- A large threshold corresponds to a large difference among behaviors $\Rightarrow$

  A few false positives, several false negatives
- A small threshold corresponds to a small difference among behaviors $\Rightarrow$

  A few false negatives, several false positives
- Different measures, different set of meausures correspond to distinct ROC curve

The threshold ...
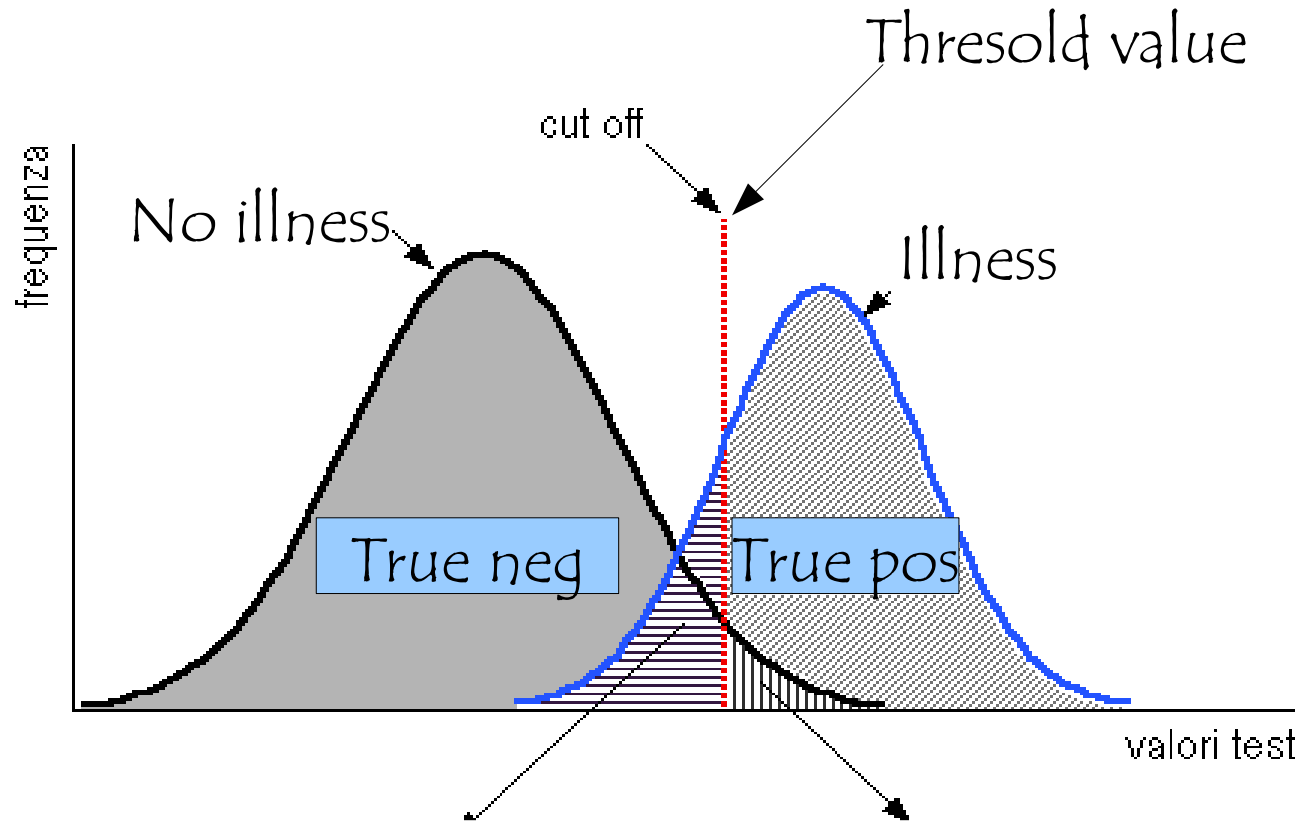
Anomaly detection: an example

- Nides = next generation intrusion detection system
- To protect military systems
- First attempt to define in a rigorous way long and short term distributions
- Measure
  - Continuous = any value
  - Categorical = one value in a predefined range
  - Binary
  - IDS related = The IDS activity is measured as well

NIDES - SRI - Continuous - I

- UCPU   User CPU time
- SCPU   System CPU time
- IO   Number of character exchanged in an application execution
- MEMCMB   Largest amount of memory to execute the application
- MEMUSE   Sum of the amount of memory used multiplied by the time it has been used = KByte*seconds.

NIDES - Continuous -II

- TEXTSZ   Size of a segment
- OPENF   Number of open file
- PGFLT   Number of memory faults
- PGIN   Number of disk pages read
- PRCTIME   Elapsed time
- SIGNAL   Number of received signals

NIDES - SRI - Categorical

- UID   New user name if changed
- HOUR   Hour when the application began
- RNETHOST   Name of the remote host that has invoked the program
- LNETHOST   Name of the local host that has invoked the program
- RNETTYPE   Name of the application invoked by the remote host

NIDES – SRI - Binary

- RNET   Application executed on a remote host
- LNET   Application executed on a local host

NIDES – IDS related

- INTARR   continuous Seconds from the last record
- I60   continuous Number of audit records produced in 1 min
- I600   continuous Number of audit records produced in 10 min
- I3600   continuous Number of audit records produced in 1 hour

NIDES – Learning time - I

Subject (Application)	Total Records	Training Records	Testing Records	Unique Days
as	1688	1539	149	39
cat	1195	1058	137	68
ccom	886	736	150	36
compile	1010	838	172	43
cp	378	273	105	60
cpp	2625	2470	155	44
csh	909	709	200	57
diff *	690	596	94	46
discuss	1328	1040	288	60
emacs	7929	6227	1702	84
finger *	619	537	82	78
fmt	1819	1522	297	64
gawk *	613	530	83	56
getfullnm *	353	269	84	52
ghostview *	320	225	95	50
grep	5685	3474	2211	60

NIDES – Learning time - II

latex	928	758	170	52
less	5409	4709	700	80
ls	9020	7368	1652	78
mail *	613	527	86	60
make	1251	1095	156	52
man	938	708	230	60
more	8015	6497	1518	68
mymoreproc	3901	3406	495	77
pwd	1405	1181	224	62
rm	2539	2184	355	82
sed	1801	1464	337	64
sort	891	702	189	61
stty	1003	871	132	68
vi	5452	4663	789	77

N&H-IDS: signature detection

- The overall behavior strongly resembles an antivirus tool

- A pattern database (signature) for known attacks, each action is compared against the components of each pattern

- Any matching is recorded

- Anytime a pattern has been fully matched, an alarm is fired

N&H-IDS: signature detection

- Wrt to Antivirus some differences:
  - The elements to be matched against the patterns are dynamically generated
  - The time inbetween the geneation of two consecutive elements is unknown
  - An element can match several patterns
- The complexity is much higher for IDSes than for antivirus where we match a sequence of characters in a file against a set of patterns

N&H-IDS: signature detection

msg=p1   msg=p2   msg=p1

1 → 2 → 3 → 4
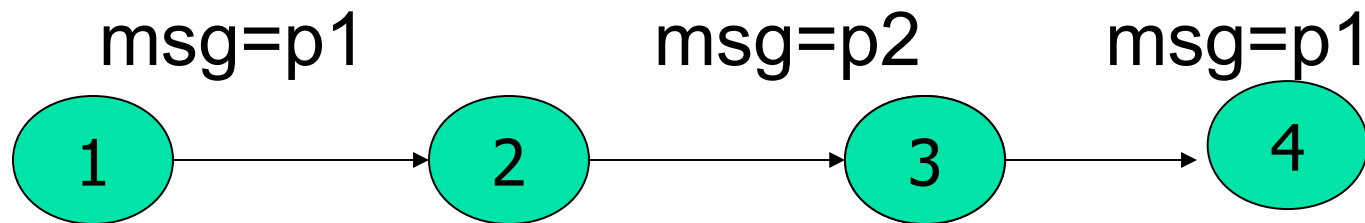
- If the current status of the recognizer is 3 and a packet = p1 is sniffed then the next state may be
  - The one following 3=4
  - The one following 1= 2
- A nondeterministic behavior is required = the status of the automata is both 2 and 4

N&H-IDS: signature detection & evasion

- When sniffing a packet P the NIDS has no mean to discover
  - Whether P will be received
  - How P will be handled
- An attacker can iniject packets to hide other ones or to confuse the IDS (eg packet with a wrong checksum that will be discarded by the receiver)
- Encrypted traffic is a further problem

N&H-IDS: signature detection

- New attacks can be discovered only if the database is continuously updated and after the update

- The detection of unknown attacks is fully delegated to anomaly detection only

- Anomaly detection can discover a new attack provided that it results in some anomaly for some time

NIDS e HIDS: new attacks??

- An alternative approach considers the IDS as a rule base expert system
  - A rule database rather than a pattern database
  - Rules describe attacks and anomaly
- A generalization (abstraction) procedure can be applied to rules to discover, at least, variants of attacks that are already known

Nimbda Signature (log)

GET /scripts/root.exe?/c+dir

GET /MSADC/root.exe?/c+dir

GET /c/winnt/system32/cmd.exe?/c+dir

GET /d/winnt/system32/cmd.exe?/c+dir

GET /scripts/..%5c../winnt/system32/cmd.exe?/c+dir

GET /_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe?/c+dir

GET /_mem_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe?/c+dir

GET /msadc/..%5c../..%5c../..%5c/..\xc1\x1c../..\xc1\x1c../..\xc1\x1c../winnt/system32/cmd.exe?/c+dir

GET /scripts/..\xc1\x1c../winnt/system32/cmd.exe?/c+dir

GET /scripts/..\xc0/../winnt/system32/cmd.exe?/c+dir

GET /scripts/..\xc0\xaf../winnt/system32/cmd.exe?/c+dir

GET /scripts/..\xc1\x9c../winnt/system32/cmd.exe?/c+dir

GET /scripts/..%35c../winnt/system32/cmd.exe?/c+dir

GET /scripts/..%35c../winnt/system32/cmd.exe?/c+dir

GET /scripts/..%5c../winnt/system32/cmd.exe?/c+dir

GET /scripts/..%2f../winnt/system32/cmd.exe?/c+dir

HTTP-WHISKER-SPLICING-ATTACK-SPACE

Signature Snort compatible (snort,prelude,etc)

alert TCP $EXTERNAL any -> $INTERNAL 80 (msg: "IDS296/web-misc_http-whisker-splicing-attack-space"; dsize: <5; flags: A+; content: "|20|"; classtype: suspicious; reference: arachnids,296;)

Signature Dragon Sensor

T D T B 10 0 W IDS296:web-misc_http-whisker-splicing-attack-space /20

Defenseworx Signature

1 B 6 T 0 80 [IDS296/web-misc_http-whisker-splicing-attack-space] "\20"

Pakemon Signature   IDS296/web-misc_http-whisker-splicing-attack-space tcp * 80 "|20|"

Shoki Signature

tcp and (dst port 80) and (ip[2:2] > ((ip[0:1] & 0x0f) + (tcp[12:1] & 0xf0) + 5)) and (tcp[13]&16!=0) 65536
SEARCH IDS296 web-misc_http-whisker-splicing-attack-space '0x20' ALL 1 NULL

Snort

- Freeware.
- Designed as a network sniffer.
- Useful for
  - traffic analysis.
  - intrusion detection.
    - Warning: Has become a target of attackers!
      - What's more fun for them than to find a vulnerability in security software.

Snort

- Snort is a good sniffer.
- Snort uses a detection engine, based on rules.
- Packets that do not match any rule are discarded.
- Otherwise, they are logged.
- Rule matching packets can also trigger an alert.

Snort Architecture

| Network Backbone | Sniffer | Preprocessor | Detection Engine | Alerts/ Logging | Log Files |

Packets

Ruleset
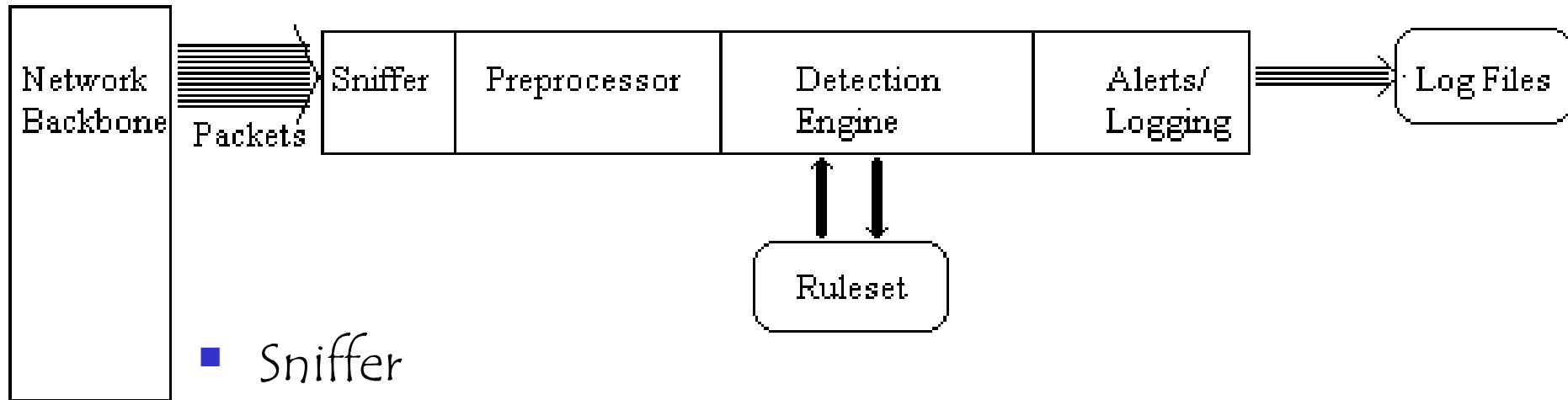
- Sniffer
- Preprocessor
- Detection Engine
- Alert Logging

SNORT Architecture

- Packet Sniffer
  - Taps into network

- Preprocessor
  - Checks against plug-ins
    - RPC plug-in
    - Port scanner plug-in
    - …

SNORT Architecture

- Detection Engine
  - Signature-based implemented via rule-sets
  - Rules
    - Consists of rule header
      - Action to take
      - Type of packet
      - Source, destination IP address
      - …
    - And rule option
      - Content of package that should make the packet match the rule

SNORT Architecture

- Snort Alerting
- Incoming "interesting packets" are sent to log files.
- Also sent to various Add-ons
  - SnortSnarf (diagnostics with html output)
  - SnortPlot (Perl script that plots attacks)
  - Swatch (provides email alerts).
  - ...

Snort: Architecture

- Packet Decode Engine
  - Uses the libpcap package
  - Packages are decoded for link-level protocols, then for higher protocols.
- Preprocessor Plug-ins
  - Each preprocessors examines and manipulates packages, e.g. for alerts.
- Detection Engine
  - Checks packages against the various options in the snort rules files.
- Detection Plug-Ins
  - Allow additional examinations
- Output Plug-Ins

Snort: Architecture

Package View:

- NIC in promiscuous mode.
- Grab packages from the network card.
- Decode packages
- Run through various rule sets.
- Output logs and alerts.

Snort Rules: Example

- ## Rule Header
  - alert tcp $External_NET any -> $Home_Net21
- ## Rule Options
  - (msg: "ftp Exploit"; flow_to_server, established; content: "|31c031db 41c9b046 cd80 31c031db|"; reference: bugtraq,1387; classtype:attempted-admin; sid 344; rev4;)

Snort Rules

- Rule Header
  - Action
  - tcp: Protocol being used. UDP / IP / ICMP
  - $External_NET: This is the source IP, default is any.
  - any: This is the source port set to "any"
  - ->: Direction of conversation.
  - $Home_Net: This is a variable that Snort will replace with
  - 21: Port to be monitored.
- The header concerns all tcp packages coming from any port from the outside to port 21 on the inside.

Snort Rules: Action

alert: generate an alert using the selected method and log

log: log the packet

pass: ignore the packet

activate: alert and then turn on another dynamic rule

dynamic: idle until activated by a rule, then act as a log rule

drop: block and log the packet

reject: block the packet, log it, and then send a TCP reset if TCP or an ICMP port unreachable if UDP

sdrop: block the packet but do not log it.

Snort Rules

Rule Options

- ( ): Rule option is placed in parentheses.
- msg: "ftp Exploit";
- flow_to_server, established;
- content: "|31c031db 41c9b046 cd80 31c031db|"; Snort will look whether the package contains this string, the dangerous payload.
- reference: bugtraq,1387; Snorts allow links to third-party warnings.
- classtype:attempted-admin; Class Types allow users to quickly scan for attack types
- sid 344; Snort rule unique identifier. Can be checked against www.snort.org/snort-db.
- rev4; All rules are part of a revision process to limit false positives and detect new attacks.

Snort Rules

- TCP: TCP protocol, for example SMTP, HTTP, FTP
- UDP: For example DNS traffic
- ICMP: For example ping, traceroute.
- IP: For example IPSec, IGMP

Snort Rules

- Content: Content checked by the Boyer Moore pattern matching algorithm.
- Flow: Link to the detection plug-ins.

Using Snort

- Binary log files are in tcpdump format
- Can be read by snort with the –r switch
- Readback can be used to dump, log, or perform detection

Using Snort

Full Text Logging

- Packets are logged in plain ascii format
- One file created per protocol port pair
- A port scan creates too many files.

Using Snort

NIDS Mode

- Load snort with a set of rules, configure packet analysis plug-ins, and let it monitor hostile network activity

Using Snort

NIDS mode:

- Specify an alternative logging directory with –l

- Specify an alternate alert mode
  - -AL fast, full, none, console
  - -M <wrkstn> Send SMB (popup) alerts

Snort analysis example

- Snort rule in rule file "rules":

  alert tcp any any -> any 12345

- snort –r cap.wdp –b –l snortlog –c rules
- This captures all traffic destined to port 12345, usually used for BackOrifice traffic.

Structure of the Bro System

Policy Script Interpreter

Real time notification

Policy script

Event Control

Event Stream

Event engine

Tcpdump filter

Filtered Packet Stream

libcap

Packet Stream
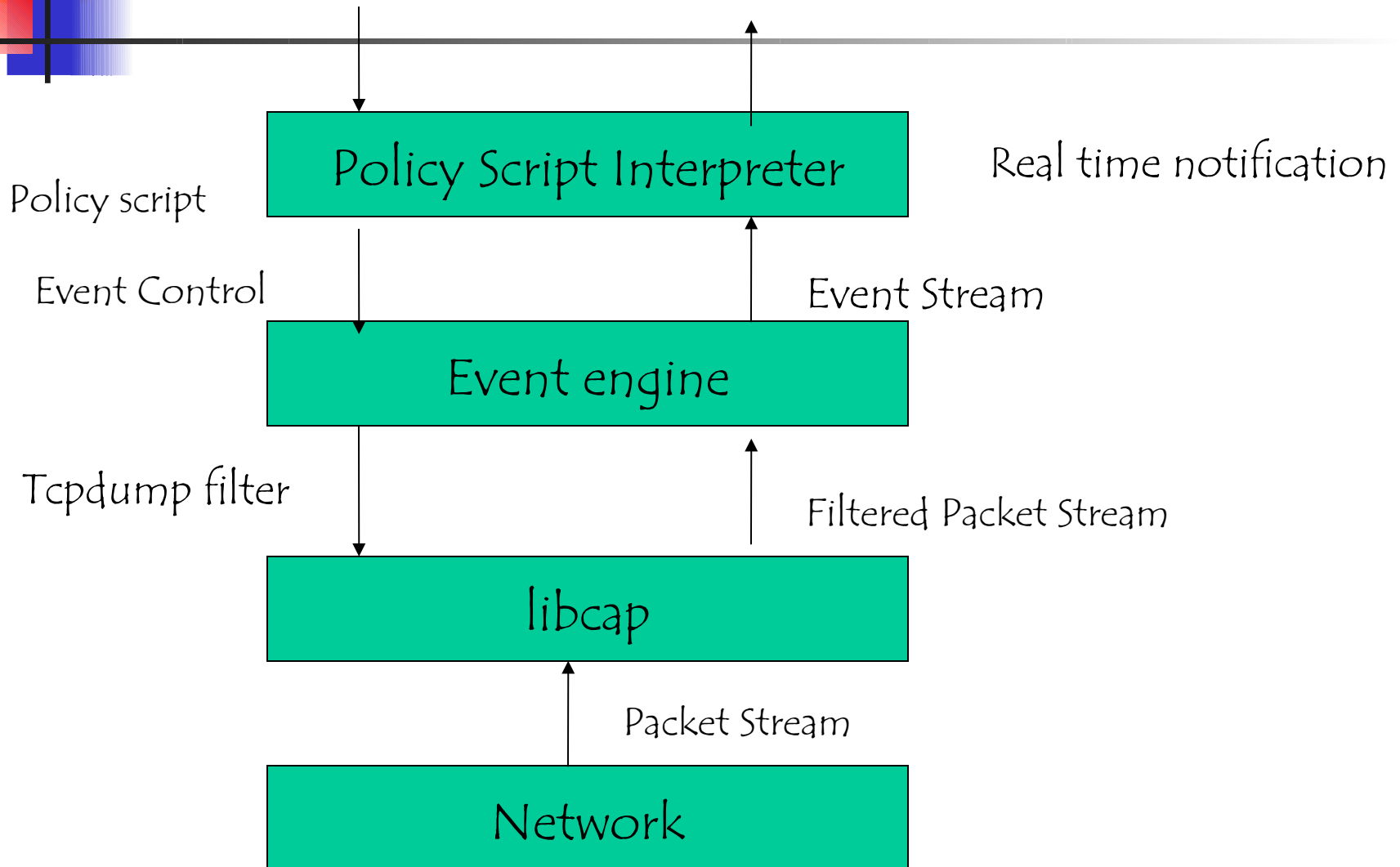
Network

Bro - libcap

- It's the packet capture library used by tcpdump.

- Isolates Bro from details of the network link technology.

- Filters the incoming packet stream from the network to extract the required packets.

- E.g port finger, port ftp, tcp port 113 (Ident), port telnet, port login, port 111 (Portmapper).

- Can also capture packets with the SYN, FIN, or RST Control bits set.

Bro – Event Engine

- The filtered packet stream from the libcap is handed over to the Event Engine.

- Performs several integrity checks to assure that the packet headers are well formed.

- It looks up the connection state associated with the tuple of the two IP addresses and the two TCP or UDP port numbers.

- It then dispatches the packet to a handler for the corresponding connection.

Bro – TCP Handler

- For each TCP packet, the connection handler verifies that the entire TCP Header is present and validates the TCP checksum.

- If successful, it then tests whether the TCP header includes any of the SYN/FIN/RST control flags and adjusts the connection's state accordingly.

- Different changes in the connection's state generate different events.

Policy Script Interpreter

- The policy script interpreter receives the events generated by the Event Engine.

- It then executes scripts written in the Bro language which generates events like logging real-time notifications, recording data to disk or modifying internal state.

- Adding new functionality to Bro consists of adding a new protocol analyzer to the event engine and then writing new events handlers in the interpreter.

Application Specific Processing - Finger

Tests for buffer overflow, checks the user against sensitive ids, etc

Script interpreter

Generates event controls based on the policy

Generates Finger_request event

Event Engine

Event Engine

Finger reply

Finger request
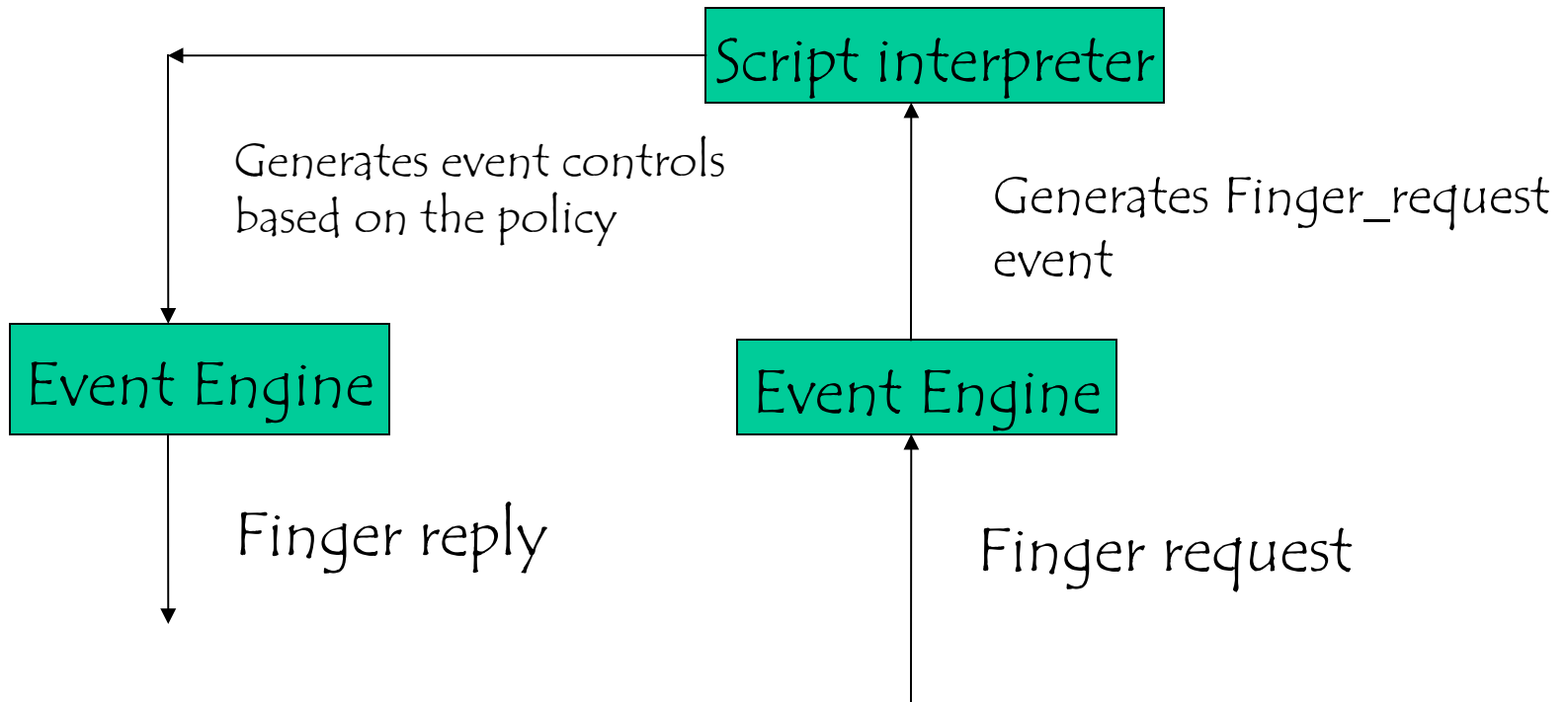
VMM Introspection: [GR'03]

protecting the anti-virus system

Intrusion Detection / Anti-virus

Runs as part of OS kernel and user space process

– Kernel root kit can shutdown protection system
– Common practice for modern malware

Standard solution: **run IDS system in the network**

– Problem: insufficient visibility into user's machine

Better: **run IDS as part of VMM (protected from malware)**

– VMM can monitor virtual hardware for anomalies
– VMI: Virtual Machine Introspection
  · Allows VMM to check Guest OS internals
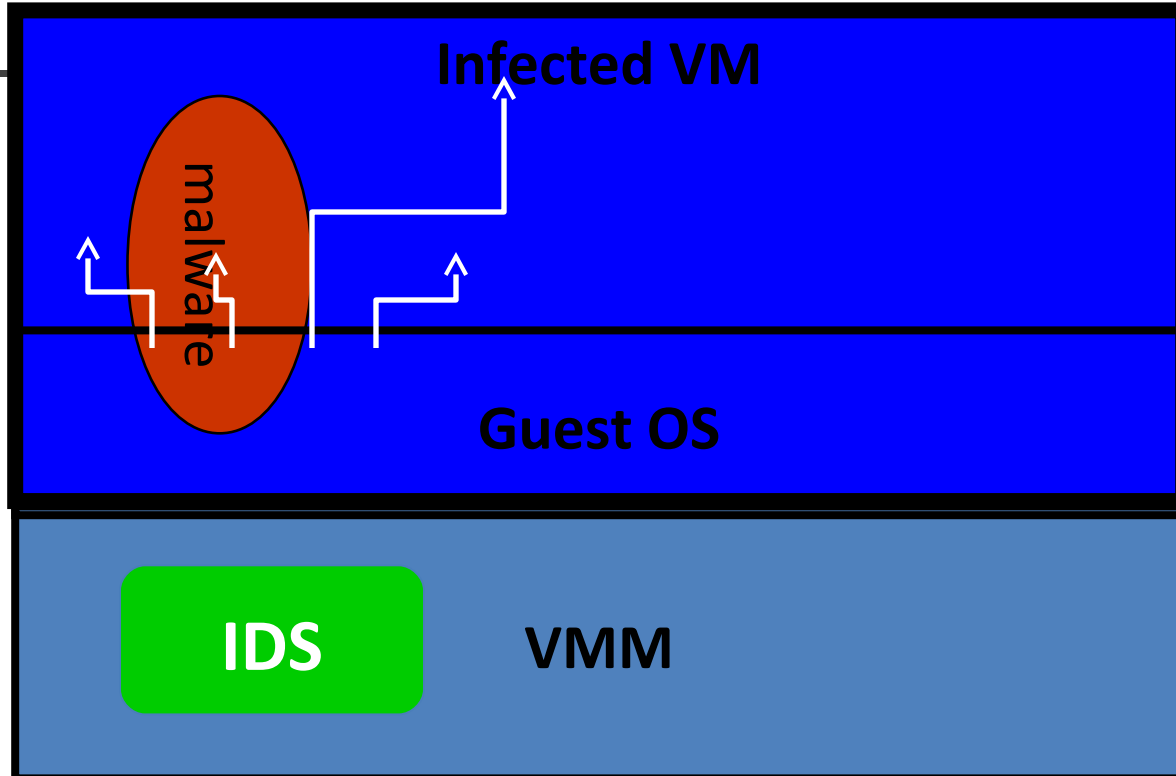
Sample checks

Stealth root-kit malware:

- Creates processes that are invisible to "ps"
- Opens sockets that are invisible to "netstat"

1. **Lie detector check**

   - Goal: detect stealth malware that hides processes and network activity

   - Method:

     · VMM lists processes running in GuestOS
     · VMM requests GuestOS to list processes (e.g. ps)
     · If mismatch: kill VM

Using a pubblic network

- Several institution have to connect remote, local networks
- Leased lines are too expensive
- The most convenient connection is the one that exploits a pubblic network, eg the internet
- The security of the connection is very low since information flows on a pubblic network

Countermeasures - Robustness

- Virtual Private Network
  - It emulates a secure connection on top of an unsafe connection
  - Assuming that each local network is protected by a firewall, secure connections are established among the firewall
  - Secure = integrity and confidentiality are achieved by encrypting the traffic between any pair of firewalls

VPN≠VLAN

- VLAN denotes a logical network that is set up to minimise the number of conflicts
- A vlan can exploit
  - Transmission frequency
  - Tags

  That are paired with some nodes
- No security property

Virtual Private Network

net 1

net 3

Internet

Encrypted
communication

net 2

net 4
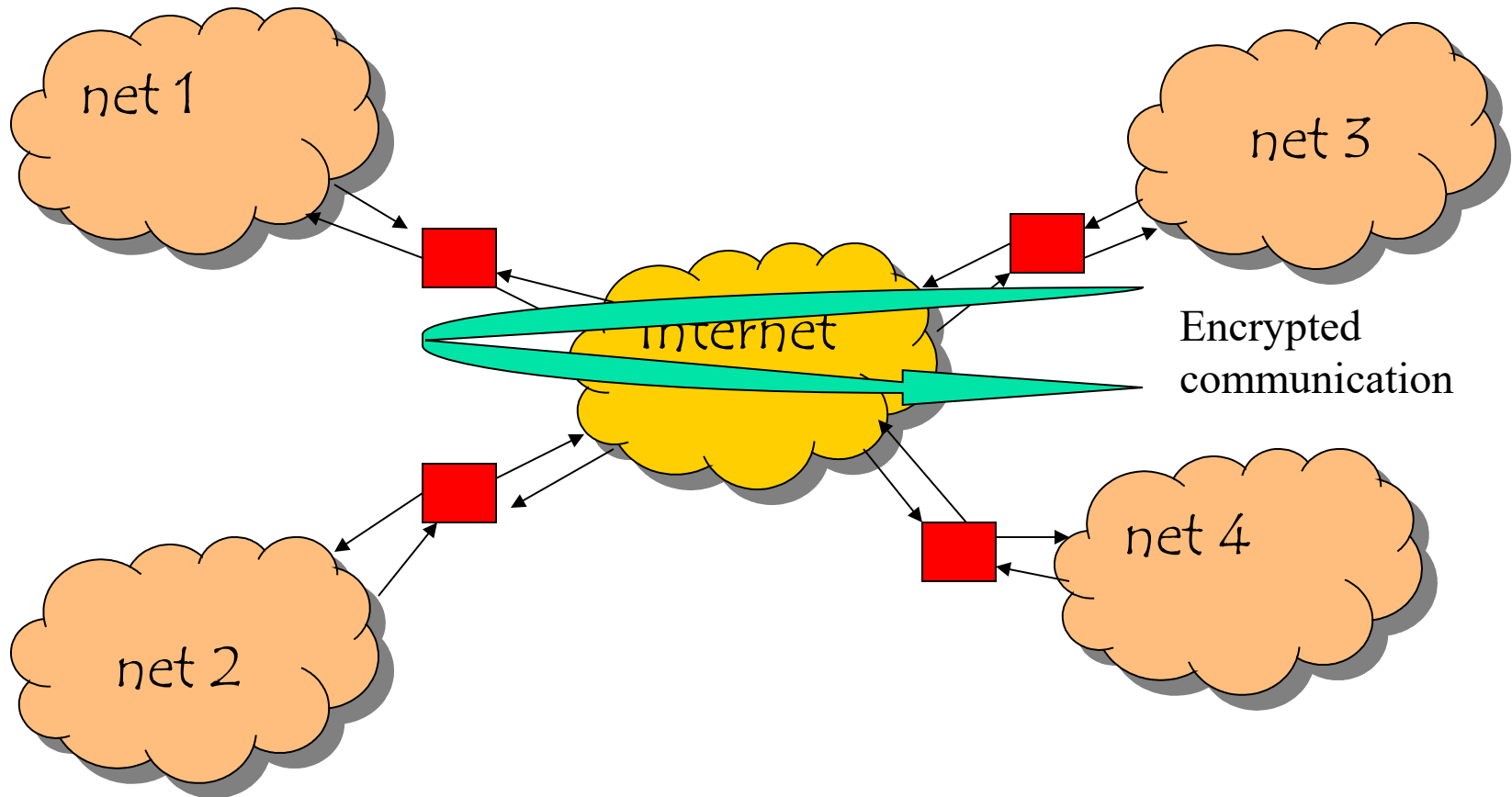
Virtual Private Network

- Symmetric Encryption due to the large amount of transmitted data
- A distinct key for each pair of firewalls
- The key is updated according to the amount of exchanged data

VPN and symmetric encryption - I

- The simpliest strategy to share a key without transmitting it is the Diffie_Helmann protocol
  - each firewall produces a number
  - All-to-all exchange
  - After the exchange, each firewall produce a key for each partner
  - Man-in-the-middle attack

VPN and symmetric encryption -II

- Each firewall pubblish a pubblic key and know the corresponding secret key
- The two keys makes it possible to compute a symmetric key
- Data to be exchanged is protected with the symmetric key
- IP v6

VPN a shared problem

- Any implementation of any VPN may be the target of a Denial of Service attack
- A VPN decrypts any message it receives. If the output satisfies the protocol, it forwards otherwise it discards the message
- If a flood of fake messages is produced, the receiver will be busy to discard them and cannot run legal applications
- This shows that any security solutions that only applies encryption cannot guarantee resource availability

IPSEC

- An IPv4 extension to authenticate and encrypt information flows, to be used till IPv4 will be replaced by IPv6 ☺ ☺

- There are further solutions that offer security service at distinct level of the OSI stacks (PGP, HTTPS, SSL, etc).

- Two IPSEC behaviours (protocols)
  - Authentication Mode = authentication header
  - Encapsulated Security Payload = the information is encrypted
  - Both protocols can be used in one of two modes
    - Transport Mode = the original packet is updated
    - Tunnel Mode = the old IP is protected and becomes the information of a new packet

IPSEC

- IPSEC can also be used

  - Between two hosts (even clients),

  - a gateway and an host

  - Between two gateways.

- By replacing IP with IPSEC, we increase communication security in a more transparent way for the involved hosts

- No update to the software or hardware network components to adopt IPSEC.

IPSEC

IPSEC defines the following, further protocols

- **AH** (Authentication Header) it protect the integrity of and authenticate the data

- **ESP** (Encapsulating Security Payload) it offers confidentiality because of encryption.

- **IKE** (Internet Key Exchange) two partners can agree on the key to be used and on how long it should be used

- **ISAKMP** (Internet Security Association and Key Management Protocol) it is used to set up and update " **Security Association** (**SA**)" and their attributes

IPSEC

- A Security Association (SA) is a directed connection that also defines the security services paired with the traffic it transmits

- To secure a bidirectional connection, two SAs are required, one in each direction

- An SA also includes any information to execute the security services

- The security services of an SA are implemented either through AH or through ESP. In general the protocols are never applied simultaneously ….but ...

IPSEC

There are two types of SA that introduce some updates to an IP packet:

Transport mode (SA between two hosts) the security header immediately follows the IP header.

Tunnel mode (at least one end point is a gateway) there are two IP headers
• The first one is the more external one and it shows where the tunnel ends
• The inner one defines the packet final destination

AH+ESP vs ESP+AH

- A VPN requires both authentication and encryption.
- Wrapping ESP inside of AH is technically possible, but is not commonly used because of AH limitations with respect to NAT. By using AH+ESP, this tunnel could never traverse a NAT device.
- ESP+AH is used in Tunnel mode to fully encapsulate the traffic across an untrusted network, protected by both encryption and authentication in the same thing.
- This traffic yields nearly no useful information save for the fact that a VPN connects two sites. This information might help to understand trust relationships, but it reveals nothing about the actual traffic, even the encapsulated protocol is hidden from outsiders.

AH+ESP according to Ms

- Using both AH and ESP is the only way to both protect the IP header and encrypt the data. However, this protection is rarely used because of the increased overhead that AH would incur for packets that are already adequately protected by ESP. ESP protects everything but the IP header, and modifying the IP header does not provide a valuable target for attackers. Generally, the only valuable information in the header is the addresses, and these cannot be spoofed effectively because ESP guarantees data origin authentication for the packets.

- In addition, some IPSec hardware offload network adapters do not support the use of AH and ESP on the same packet. If you are using such offload adapters, determine the protocol support that they provide before selecting an IPSec protocol to use.

IPSEC

Authentication Header (AH)
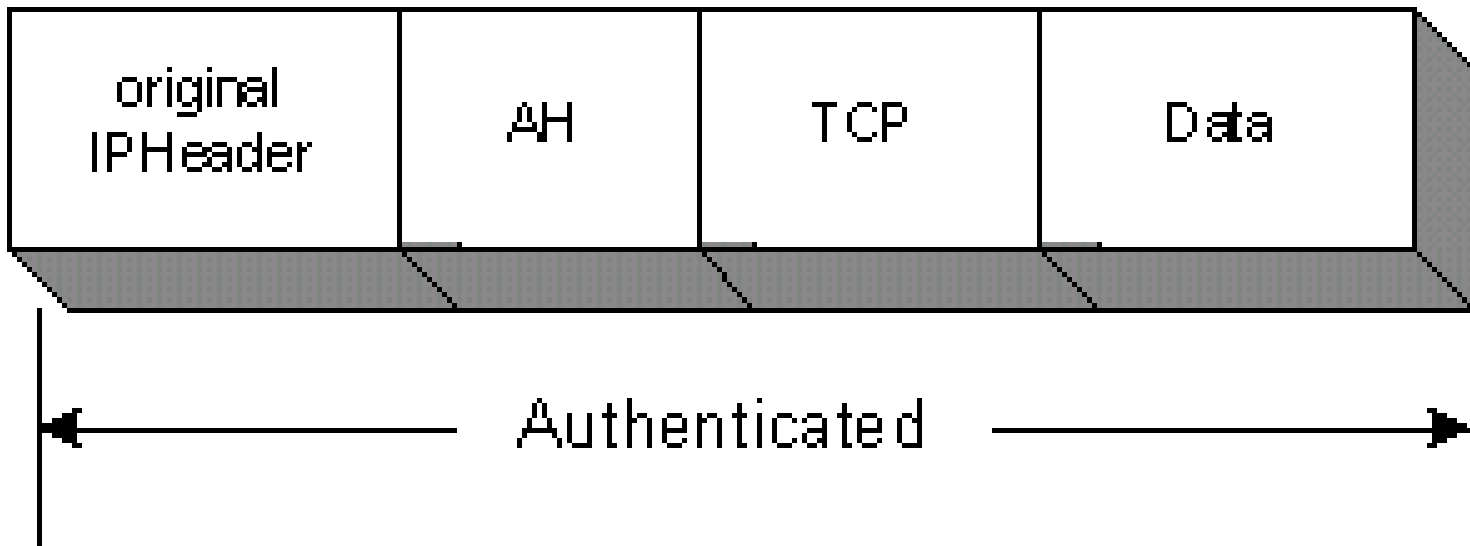
IPSEC

Encapsulating Payload Protocol (ESP)

IPSEC

IP Data Packet

IP	L4	Data

Authentication Header (AH)

IP	AH	L4	Data

AH Authenticated

Authentication only

AND/OR

Encapsulation Security Payload (ESP)

IP	ESP Header	L4	Data	ESP Trailer	ESP Auth

Encrypted

ESP Authenticated

Optional authentication.

AH + ESP together:
first perform ESP then AH computation

IP	AH	ESP Header	L4	Data	ESP Trailer	ESP Auth

Encrypted

ESP Authenticated

AH Authenticated

IPSEC

IP Data Packet

IP	L4	Data

Authentication Header (AH)

IP	AH	L4	Data

AH Authenticated

Authentication only

AND/OR

Encapsulation Security Payload (ESP)

IP	ESP Header	L4	Data	ESP Trailer	ESP Auth

Encrypted

ESP Authenticated

Optional authentication.

AH + ESP together:
first perform ESP then
AH computation

IP	AH	ESP Header	L4	Data	ESP Trailer	ESP Auth

Encrypted

ESP Authenticated

AH Authenticated

Authentication Mode

Vers.=4	HLEN	ToS or DSCP	Total Length	
Fragment Identifier			Flags	Fragment Offset
TTL		protocol = 51	Header Checksum	
Source Address				
Destination Address				
IP Options				Pad
First 32 bits of AH				
....				
Last 32 bits of AH				
Payload				
....				
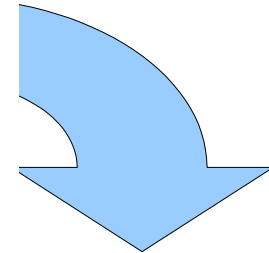
0	8	16	24	31
Next Header	Length	Reserved		
Security Parameters Index (SPI)				
Sequence Number				
Authentication Data (variable number of 32-bit words)				

ESP

0	4	8	16		31
Vers.=4	HLEN	ToS	Total Length		
Fragment Identifier			Flags	Fragment Offset	
TTL		protocol = 50	Header Checksum		
Source Address					
Destination Address					
IP Options					Pad
ESP Header with ESP Parameters Encrypted Data ESP Trailer					

	0	8	16	24	31
0	Security Parameters Index (SPI)				
4	Sequence Number				
8	Payload Data Field (variable length) maybe starting with cryptographic synchronization data (e.g. Initialization Vector IV for DES-CBC)				
			Padding (0-255 bytes)		
	Padding (0-255 bytes)			Pad Length	Next Header
	ESP Authentication Data (optional)				

IPSEC

Transport Mode

- Only one IP header.
- Used for end-to-end sessions
- Does not hide communication statistics because of network header (IP addresses of the end systems) is sent in clear text
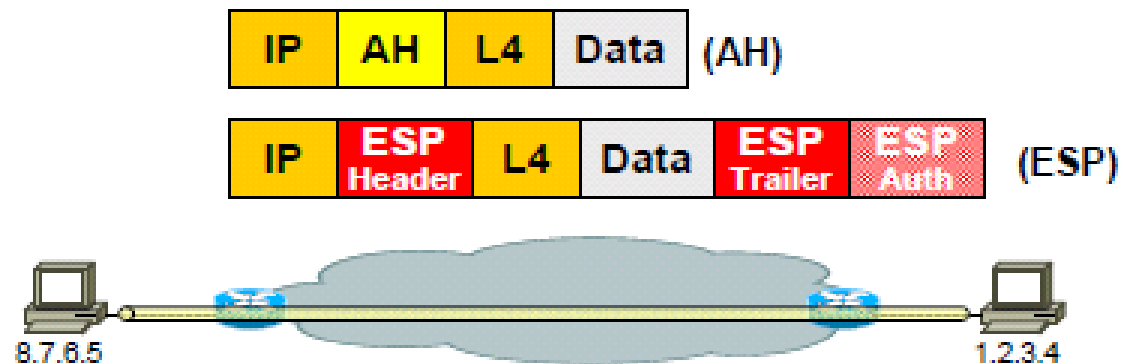
IPSEC **Authentication Header** (AH)

Original IP packed

IP header | other headers and payloads

MD5/SHA-1

IP header | Auth. header | other headers and payloads
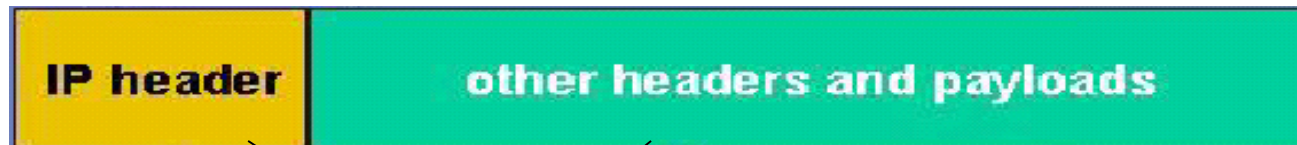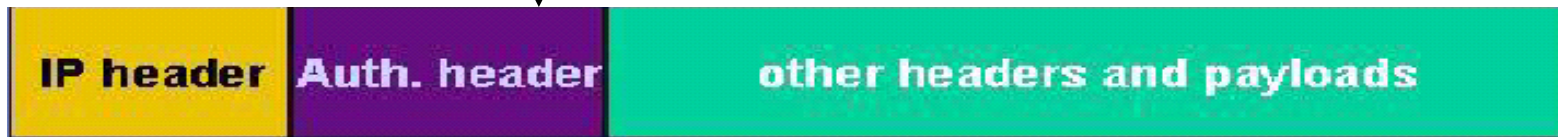
Authenticated packet

IPSEC: **ESP in Transport Mode**

Original IP packet

IP header	other headers and payloads

IP header	ESP header	other headers and payloads	ESP trailer
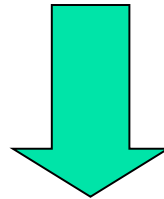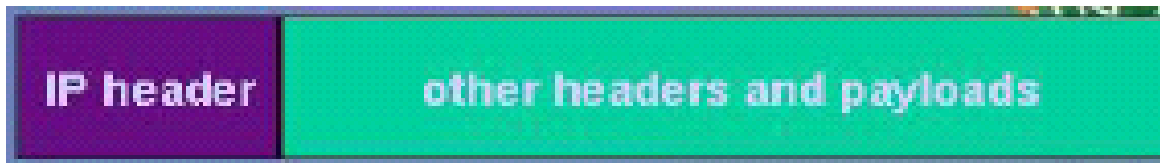
IP packet with ESP in Transport mode

IPSEC

Tunnel Mode
for Site-to-Site VPN

- Whole original IP packet is IPsec-encapsulated.
- Used for VPNs.
- Does hide traffic patterns!
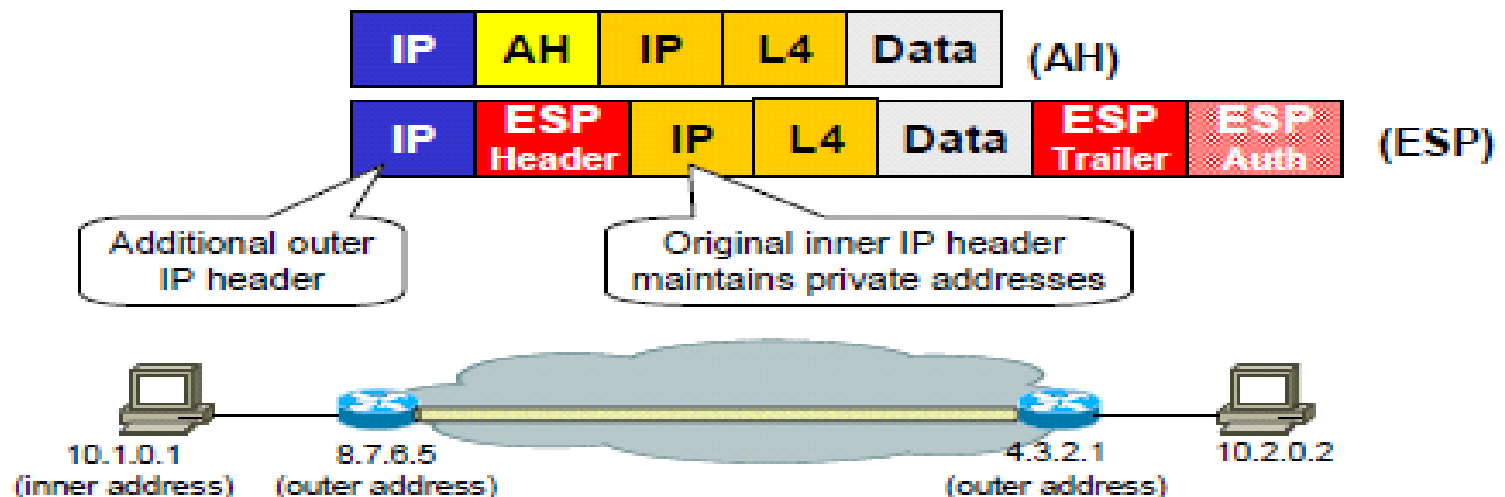
IPSEC: **ESP in Tunnel Mode**

Original IP packet
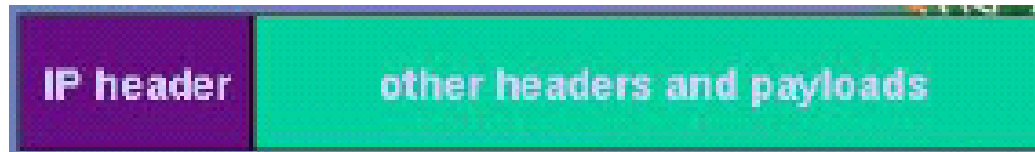
new IP
header
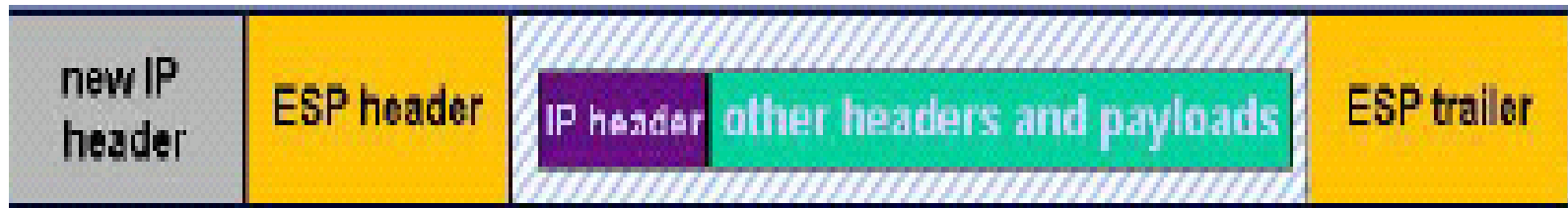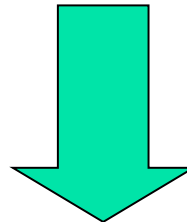
IP packet ESP + Tunnel mode

SA unidirectional
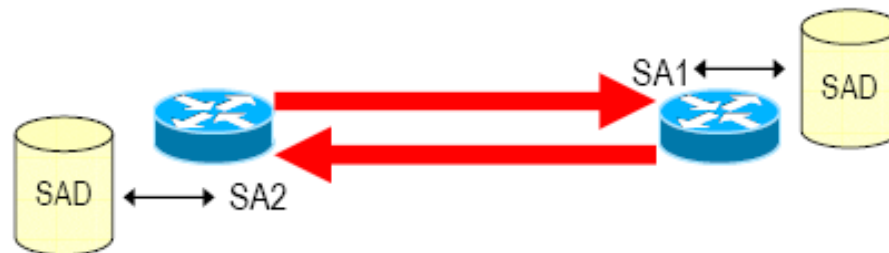
→ **SPI = Security Parameters Index**

⇨ The (somewhat) unique "name" of an SA

→ **Destination Address based**

⇨ Security Association managed at the receiver side

→ **SAD = Security Associations Database**

⇨ SPI = search key (at least)

⇨ Stores set of security services per each SA, and related parameters

→ E.g. which encryption algorithm; shared key for encryption, SA lifetime, Sequence number counter, etc

Applying several SAs
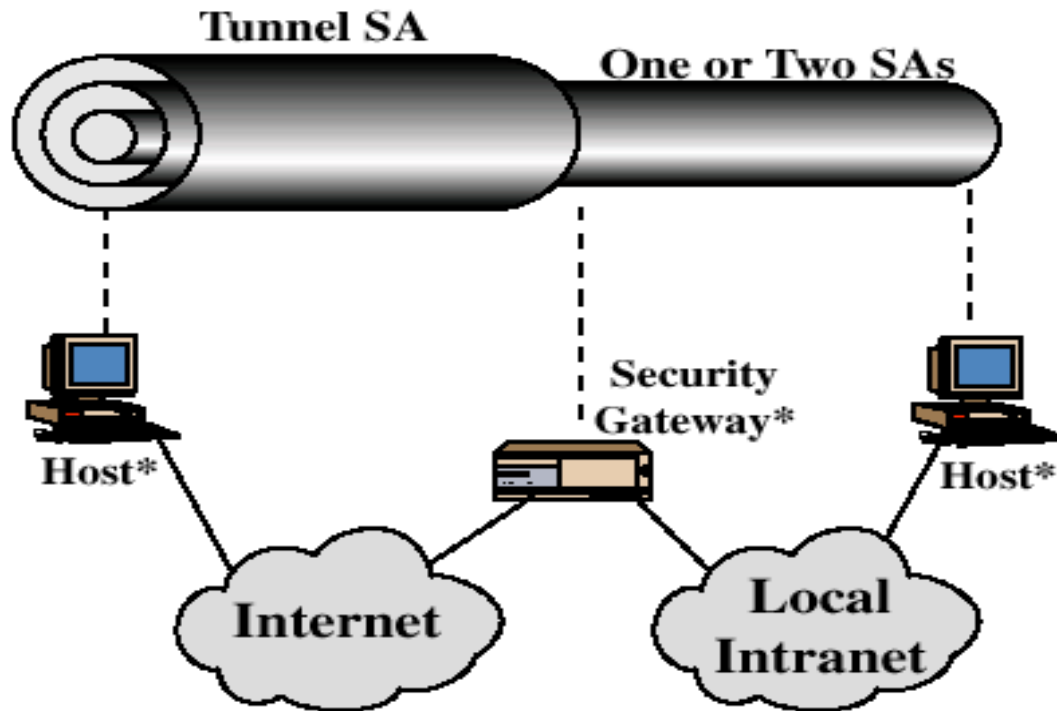
(d) Case 4

SPI – Header field
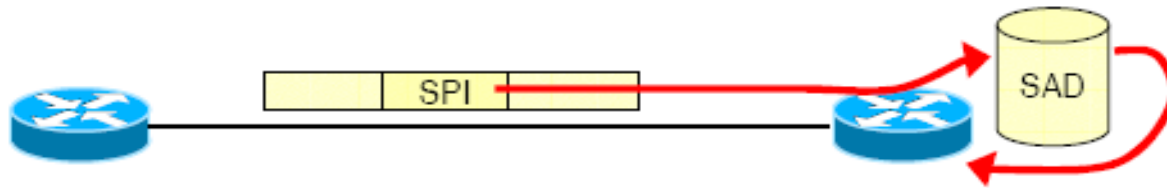
→ **32 bit index**

→ **Used to lookup the SAD at destination**

   ⇨ Lookup also uses

      →destination address

      →source address

      →security protocol (AH/ESP)

→ **Retrieves algorithms and parameters that allow to process received packet**
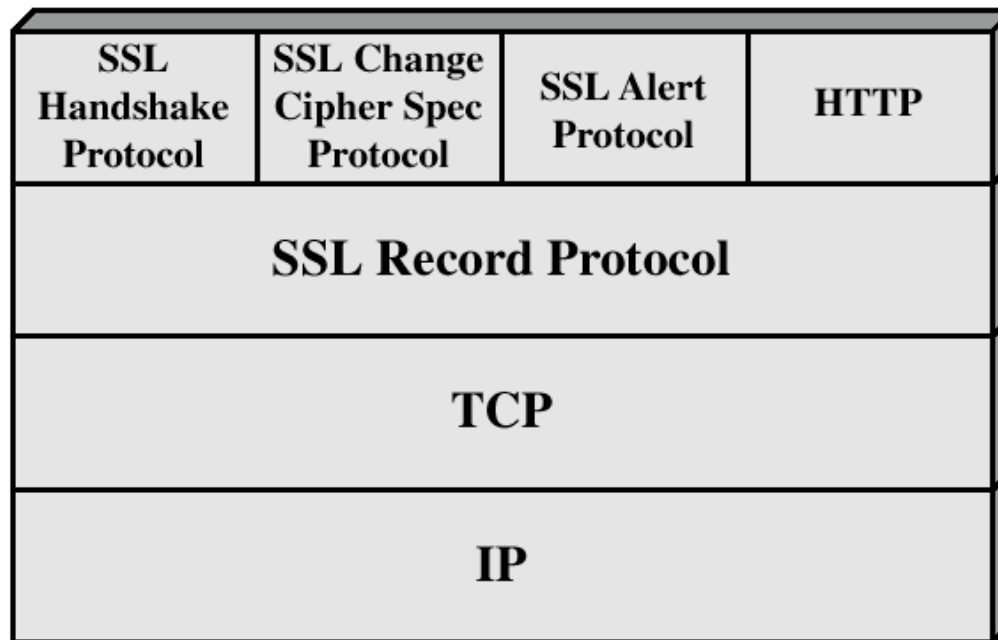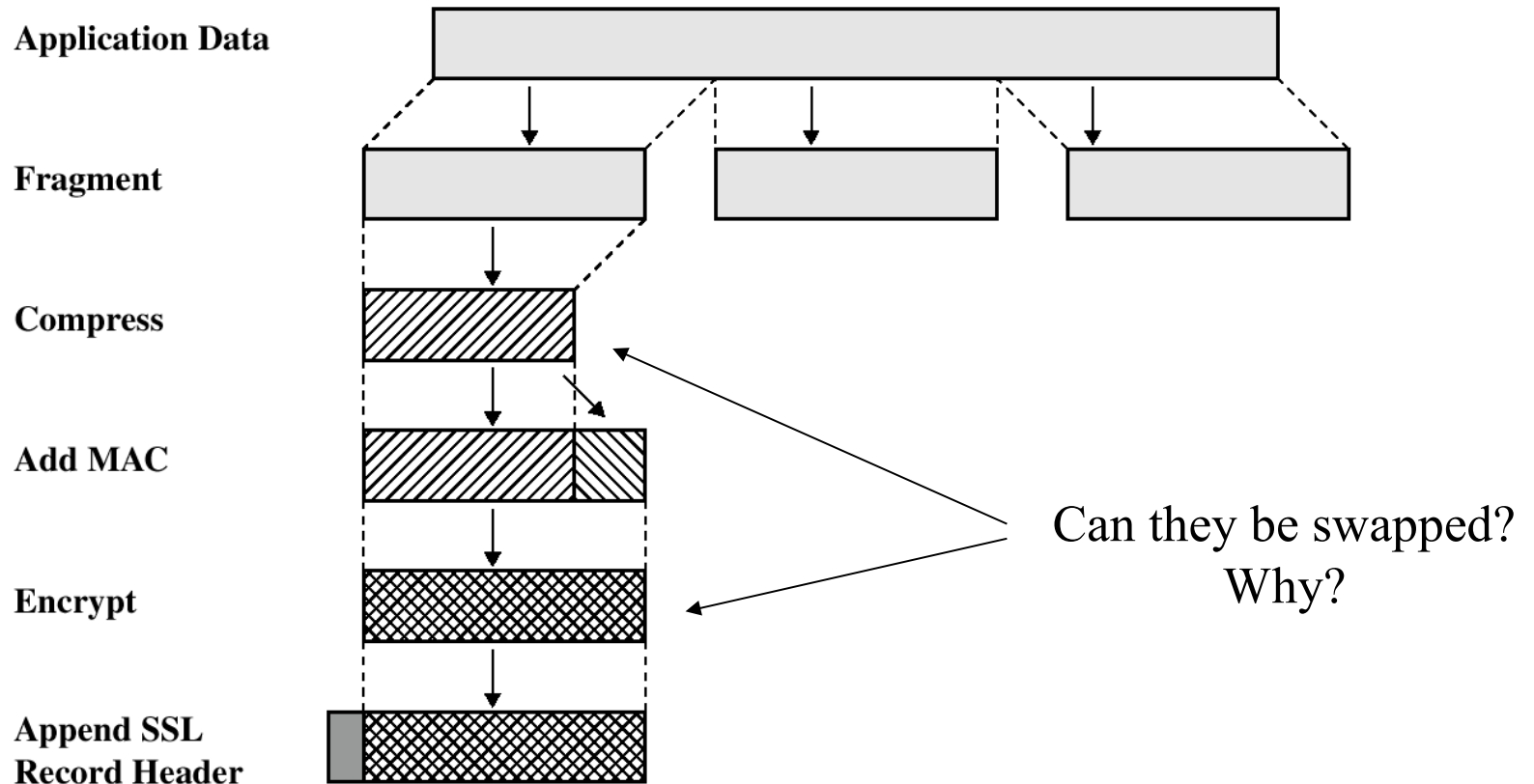
SSL = applicative VPN

SSL Handshake Protocol	SSL Change Cipher Spec Protocol	SSL Alert Protocol	HTTP
SSL Record Protocol			
TCP			
IP			

Figure 7.2   SSL Protocol Stack

SSL

Application Data

Fragment

Compress

Add MAC

Encrypt

Append SSL Record Header

Can they be swapped? Why?

SSL

- Fragment, at most 16384 bytes (2**14)
- SSLv3 does not specify a compression method
  - No information loss, and length increase should be lower than 1024
  - Default = no compression
- Encryption methods
  - Idea (128) des (56) triple des (168)
  - Stream cipher: rc4-40, rc4-128

Some definitions

- session:
    - association between a client and a server that defines a set of parameters such as algorithms used, session number etc.
    - a session is created by the Handshake Protocol that allows parameters to be shared among the connections made between the server and the client, and sessions are used to avoid negotiation of new parameters for each connection.
- connection: logical client/server link, associated with the provision of a suitable type of service. In SSL terms, it is a peer-to-peer connection with two network nodes.
- A single session is shared among multiple SSL connections between the client and the server. Multiple sessions may be shared by a single connection, but this is not used in practice.

Session state

Session identifier: an arbitrary byte sequence, chosen by the server to identify the state of an active section and can be reused to continue the session ;

- Peer certificate: the node certificate that may not exist;
- Compression method: the algorithm to compress the data;
- Cipher spec: the encryption algorithm and the one use to compute the MAC. It also defines cryptographic attributes as the hash_size;
- Master secret: a 48 byte secret information shared by the client and the server that will be used to compute the encryption keys;
- Is resumable: a flag that shows if the section can be reused

Connection State

The connection state is defined by the following parameters:
- Server and client random: a random byte sequence chosen by the client and by the server for each connection ;
- Server write MAC secret:the secret key to compute the MAC on the server data ;
- Client write MAC secret:the secret key to compute the MAC on the client data;
- Server write key: the key to encrypt the data from the server to the client ;
- Client write key: the key to encrypt the data to the server from the client ;
- Initialization vectors: a data for CBC encryption (Cipher Block Chaining). It is shared by both partners because it is need both to encrypt and to decrypt.
- Sequence numbers: each partner stores and manages the sequence numbers to send and receive messages on each connection. When one of the partners send a change cipher spec, the corresponding sequence number is zeroed. Sequence numbers are $2^{64}-1$ at most .

Record Protocol

- Frames and encrypts upper level data into one protocol for transport through TCP (reliable communications)

- 5 byte frame
  - 1st byte protocol indicator
  - 2nd byte is major version of SSL
  - 3rd byte is minor version of SSL
  - Last two bytes indicate length of data inside frame, up to 2^{14}

- Message Authentication Code (MAC)

The Four Upper Layer Protocols

- Handshaking Protocol
  - Establish communication variables
- ChangeCipherSpec Protocol
  - Alert to a change in communication variables
- Alert Protocol
  - Messages important to SSL connections
- Application Encryption Protocol
  - Encrypt/Decrypt application data

Message Authentication Code

- MAC secures connection in two ways
  - Ensure Client and Server are using same encryption and compression methods
  - Ensure messages sent were received without error or interference
- Both sides compute MACs to match them
- No match = error or attack

MAC

- hash(MAC_write_secret || pad_2 || hash(MAC_write_secret || pad_1 || seq_num || SSLCompressed.type || SSLCompressed.length || SSLCompressed.fragment))

  dove:

- ||= concatenation;
- MAC_write_secret: secret shared key;
- hash: hash algorithm (MD5 o SHA-1);
- pad_1: byte 0x36 (00110110) repeated 48 times (384 bit) for MD5 and 40 (320 bit) for SHA-1;
- pad_2: byte 0x5C (01011100) repeated 48 times for MD5 and 40 for SHA-1;
- seq_num: sequential number of the message;
- SSLCompressed.type: higher level protocol to be applied;
- SSLCompressed.length: length of the compressed packet;
- SSLCompressed.fragment: compressed fragment (the clear text fragment if no compression is applied).

Handshaking Messages

- ClientHello
- ServerHello
- *Certificate
- ServerKeyExchange
- *CertificateRequest
- ServerHelloDone
- *Certificate
- *CertificateVerify
- ClientKeyExchange
- ChangeCipherSpec
- Finished

*=optional

Client ... Server

client hello →
← server hello
← Certificate
← server key exchange
← Request for Client's Certificate
← server done
Certificate →
Certificate Verification →
client key exchange →
change cipher spec →
finished →
← change cipher spec
← finished
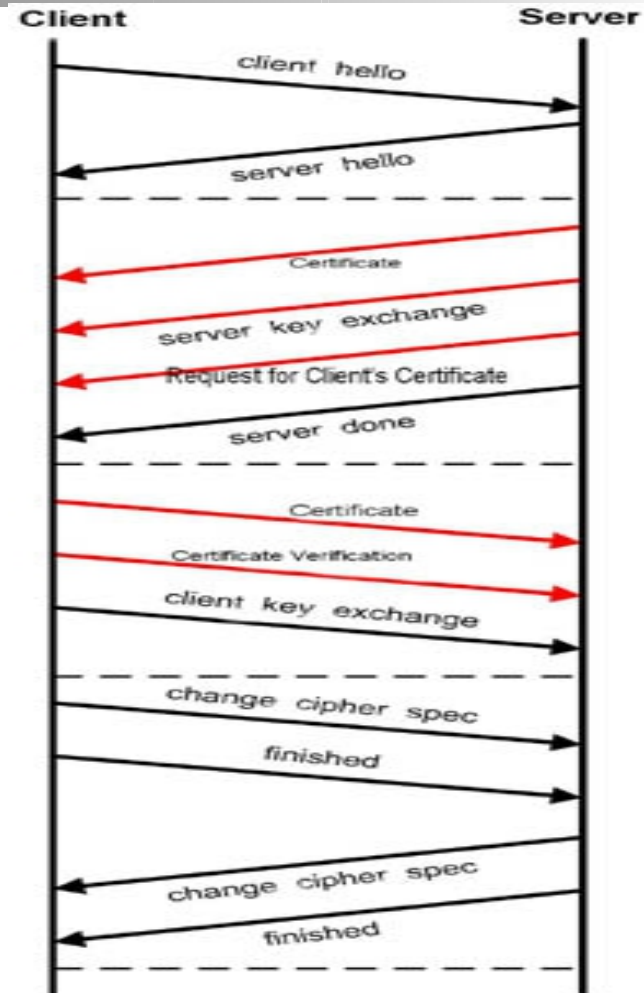
In brief …

1. The client sends the server the client's SSL version number, cipher settings, a nonce, and possibly a request for the server's certificate.
2. The server sends the client the server's SSL version number, cipher settings, a nonce, its own certificate, and requests the client's certificate if it is needed.
3. Client authenticates the server (warning box if it fails).
4. Client creates the **premaster secret** for the session, encrypts it with the server's public key and sends it to the server. Client also sends its own certificate, if requested.
5. Server authenticates the client (terminates session if authentication fails).
6. Server uses its private key to decrypt the premaster secret, then performs a series of steps (which the client also performs, starting from the same premaster secret) to generate the shared **master secret** (shared session key). Client simultaneously computes session key.
7. Client and server inform each other that they have computed a session key, and both signal termination of the handshake protocol.

Premaster secret vs secret
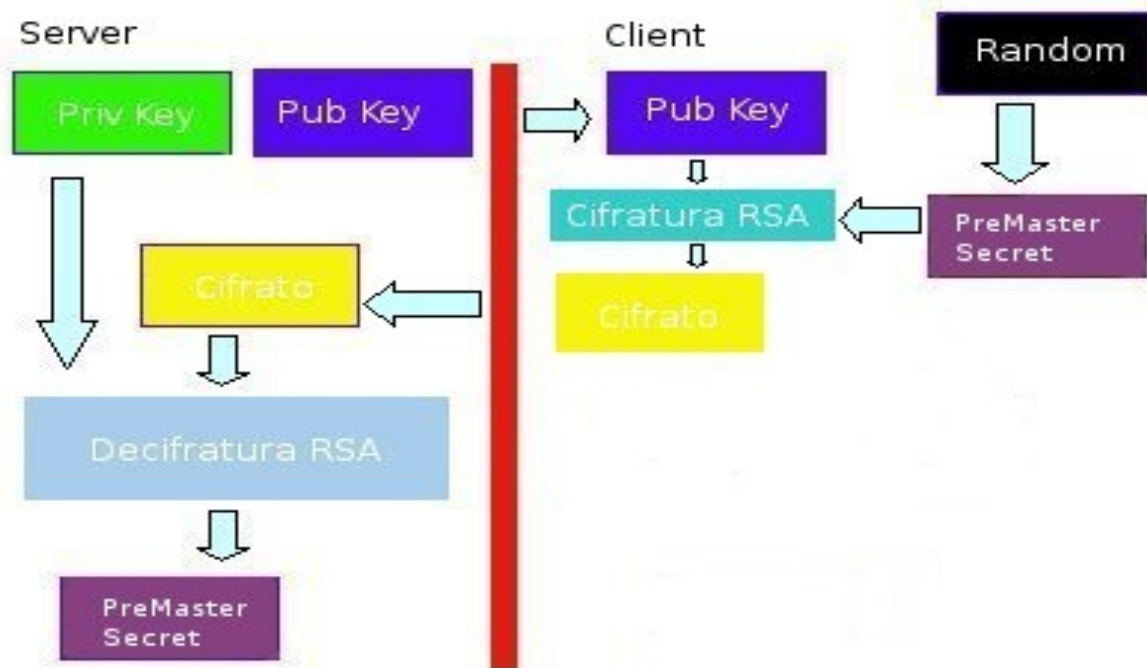
master_secret = MD5(pre_master_secret || SHA('A' || pre_master_secret || ClientHello.random || ServerHello.random) || MD5(pre_master_secret || SHA('BB' || pre_master_secret || ClientHello.random || ServerHello.random)) || MD5(pre_master_secret || SHA('CCC' || pre_master_secret || ClientHello.random || ServerHello.random));

X.509 certificates

- Version: Which version of the X.509 standards is applied (v1, v2 or v3)

- Serial number: This number is assigned by the CA to identify the certificate;

- Signature algorithm: the algorithm used by the CA to sign the certificate.

- Issuer: thel X.500 Distinguished Name of the signing CA ;

- Validity period: The lifetime of the certificate;

- Subject: the DN of the entity that is identified by the certificate;

- Subject Public key information: information on the subject pubblic key
  - Public key algorithm: algorithm used to generate the pubblic and private keys .
  - RSA Public key:key length;
  - Modulus: the modulo N used to sign ;
  - Exponent: the exponent e used to sign.

- Signature algorithm: the signature of the certificate, encrypted by the CA private key

Detail: The process begins

- Client Sends ClientHello
  - Highest SSL version supported
  - 32-byte random number
  - SessionID
  - List of supported encryption methods
  - List of supported compression methods

The Server Responds

- Server Sends ServerHello
  - SSL version that will be used
  - 32-byte random number
  - SessionID
  - Encryption method that will be used
  - Compression method that will be used

Server Authentication

- To authenticate Server, Server sends Certificate
  - Server's public key certificate
  - Issuing authority's root certificate
- When Client receives Certificate, it decides whether or not to trust Server
  - This is the only step that might involve User if User never specified whether or not to trust the issuing authority before

Still Shaking Hands

- Server Sends ServerKeyExchange
  - Any information necessary for public key encryption system
- If Server wishes Client to be authenticated, Server sends CertificateRequest message
  - The client would respond to this with a Certificate message encrypted with Server's public key
- Server sends ServerHelloDone

Client Responds

- Client sends ClientKeyExchange
  - Information necessary for public key encryption system
  - Encrypted with Server's public key
- Compute secret keys using Key Derivation Function such as Diffie-Hellman
- If Client is being authenticated, Client sends CertificateVerify
  - Digest of previous messages encrypted with Client's private key

ChangeCipherSpec Protocol

- Special protocol with only one message
- When Client processes encryption information, it sends ChangeCipherSpec message
  - Signals all following messages will be encrypted
- ChangeCipherSpec is always followed by Finished message

The End of the Beginning

- Upon receipt of ChangeCipherSpec, Server sends its own ChangeCipherSpec and Finished messages

- After both Client and Server receive Finish messages, Handshaking phase is over

- All following communication is encrypted

- Encryption and compression methods can be changed with new ChangeCipherSpec messages

Alert and Application Protocols

- Alert protocol always two byte message
  - First byte indicates severity of message
    - Warning or Fatal
    - A Fatal alert will terminate the connection
  - Second byte indicate preset error code
  - Secure connection end alert not always used
- Application Protocol is HTTP, POP3, SMTP, or whatever application is being used
  - Simply give a datagram to the Record Layer

Alert

- unexpected_message;

- bad_record_mac;

- decompression_failure;

- handshake_failure: the sender cannot negotiate an acceptable set of parameters

- illegal_parameter: an uncorrect handshake parameter.

- close_notify: sent by each side before closing its side of the connection

- no_certificate: reply if no certificate can be used ;

- bad_certificate: the received certificate has been manipulated

- unsupported_certificate: the receiver certificate is not supported ;

- certificate_revoked, _expired, _unknown: the certificate has been revoked, or is out of date or it cannot be elaborated

Benefits

- Ease of implementation
  - For network application developers
    - As easy as implementing unsecured Sockets
  - For network implementation developers
    - Simply add layer to established network protocol stack
  - For Users
    - Only need to authorize certificates

Drawbacks

- More bandwidth needed

- Slower

- Needs a dedicated port – 443 for HTTPS

- Assumes reliable transport for underlying transport protocol
  - No UDP
  - Implications for streaming media, VoIP

Countermeasures - OS

- An OS that can implement a large set of security policy rather than a predefined one
- Implemented by the OS rather than on top the OS
- Large set = MAC + DAC + RBAC …
- It increases the security of the applications it supports

Security Enhanced Linux

- A set of mechanisms to implement MAC e DAC security policies
- A set of tools that support
  - A simple description of the security policy of interest
  - Check the consistency of the description
  - Force the adoption of the policy
- Evolution of two OSs: Flask e Fluke
- Both are microkernel OS
- NSA + NAI + MITRE

SELinux - NSA

The increased awareness of the need for security has resulted in an increase of efforts to add security to computing environments. However, these efforts suffer from the flawed assumption that security can adequately be provided in application space without certain security features in the operating system. In reality, operating system security mechanisms play a critical role in supporting security at higher levels. This has been well understood for at least twenty five years and continues to be reaffirmed in the literature. Yet today, debate in the research community as to what role operating systems should play in secure systems persists. The computer industry has not accepted the critical role of the operating system to security, as evidenced by the inadequacies of the basic protection mechanisms provided by current mainstream operating systems. The necessity of operating system security to overall system security is undeniable; the underlying operating system is responsible for protecting application-space mechanisms against tampering, bypassing, and spoofing attacks. If it fails to meet this responsibility, system-wide vulnerabilities will result.

An interesting comment...

Let me assure you that this action by the NSA was the crypto-equivalent of the Pope coming down off the balcony in Rome, working the crowd with a few loaves of bread and some fish, and then inviting everyone to come over to his place to watch the soccer game and have a few beers. There are some things that one just never expects to see, and the NSA handing out source code along with details of the security mechanism behind it was right up there on that list.

Why do we need a SE Linux and not only Linux?

Definition of the security policy
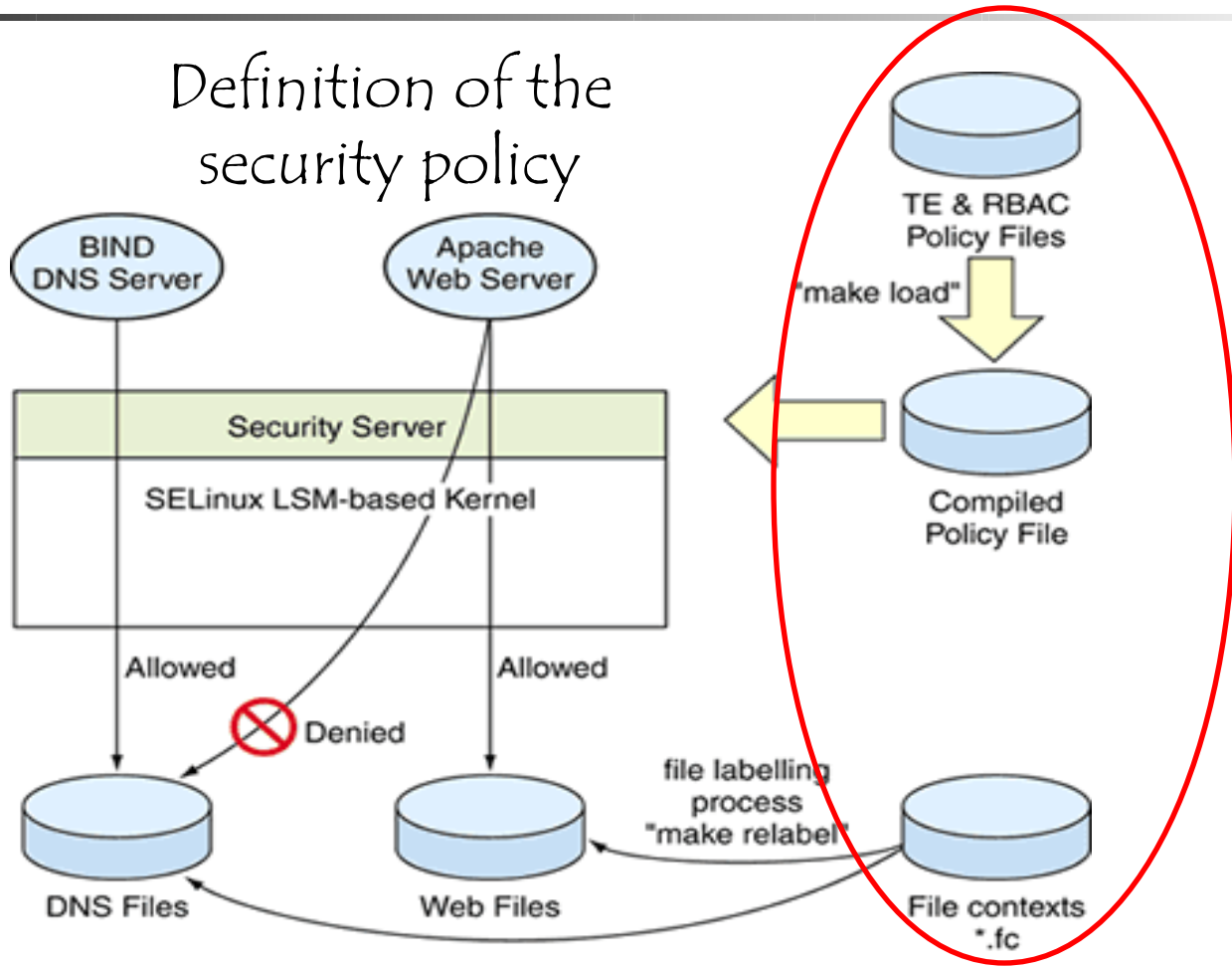
SeLinux vs Linux

- Linux defines the user rights
- Selinux defines
  - The rights of each program
  - The programs that each user can run
- Rights are defined in terms of types, of roles and of levels
  - Type1 can do this op on type2
  - This role can run program with these types

SE - Linux

- Final goal: the security policy is a configuration parameter
- Both MAC and DAC security policy can be defined
- No notion of root user
- Model to define security policies is based upon Flask and Fluke

In brief

- DAC = Discretionary Access Control = user rights are defined by the owner

- MAC = Mandatory Access Control = system wide constrains that the owner has to respect

- RBAC = Role Based Access control = rigths defined according to the user role

- Role= set of users = distinct rights of the same user at distinct times

- MLS = multilevel security = MAC constrain defined in terms of levels of subjects and objects

General Model - SID

- Each subject and each object is paired with a security context, the one used to solve access control decisions

- Context = type, level, role

- This information is stored in a security server that is invoked before executing an operation

- Each process can only access a logical pointer to this context that it transmits to the server

General Model - PSID

- PSID = SID for persisten object
- Each file system includes a file to map each inode into a PSID and then into a context
- This file is used when the file system is mounted

General model - Interactions
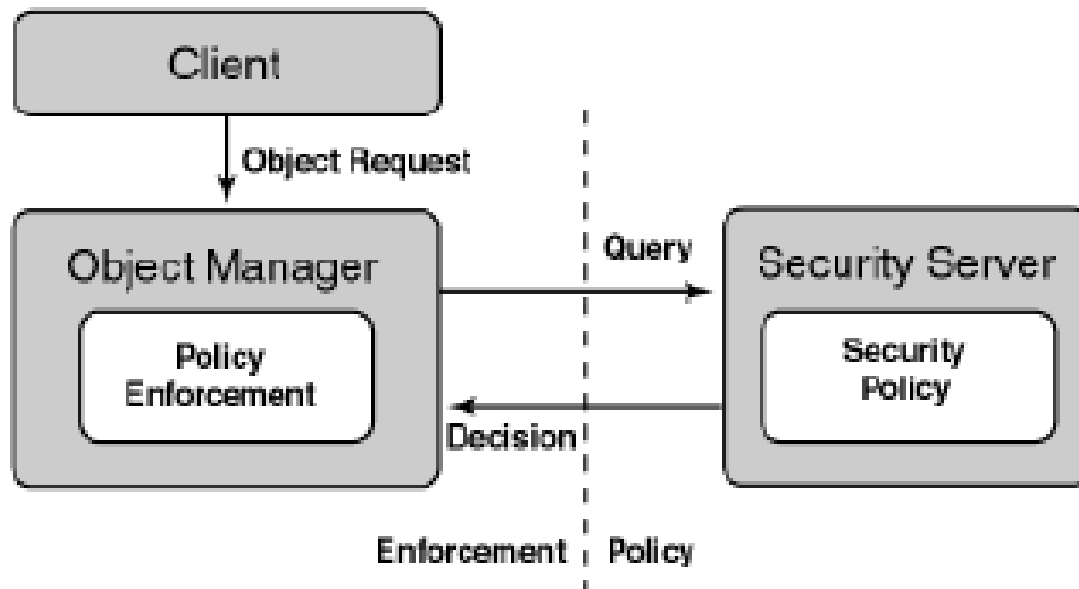
Enforcement with no informatio about the security policy
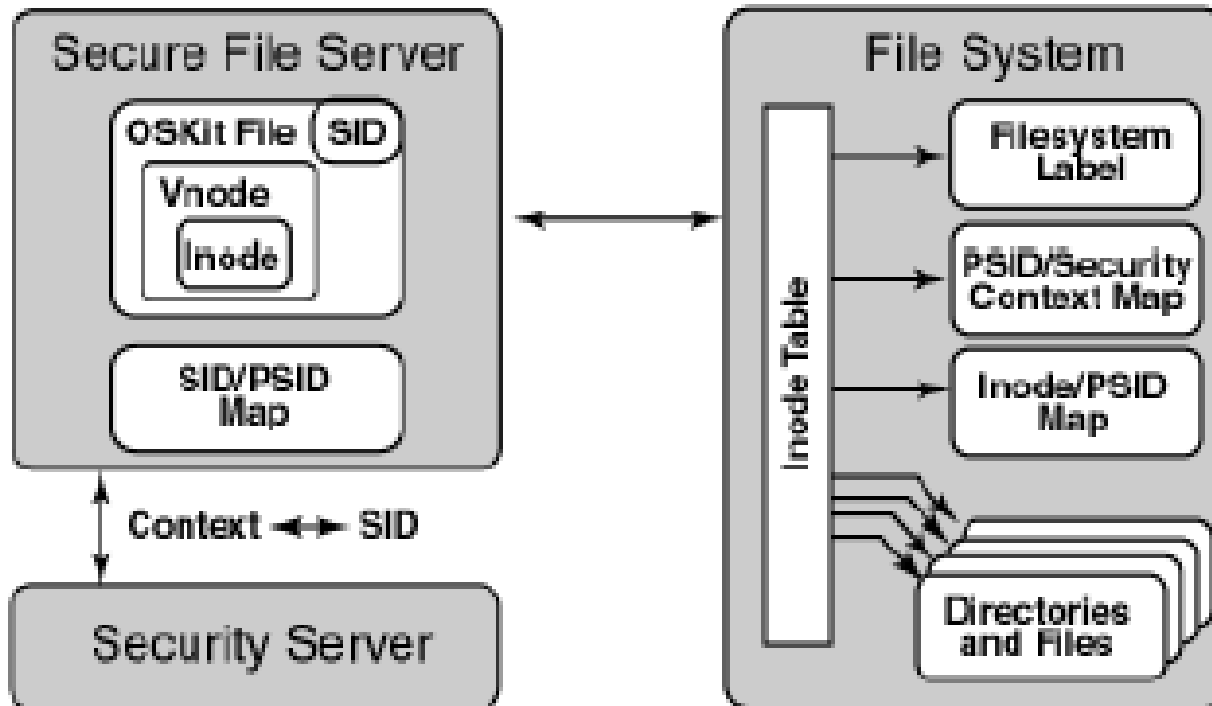
Security policy with no enforcement

SID and Context
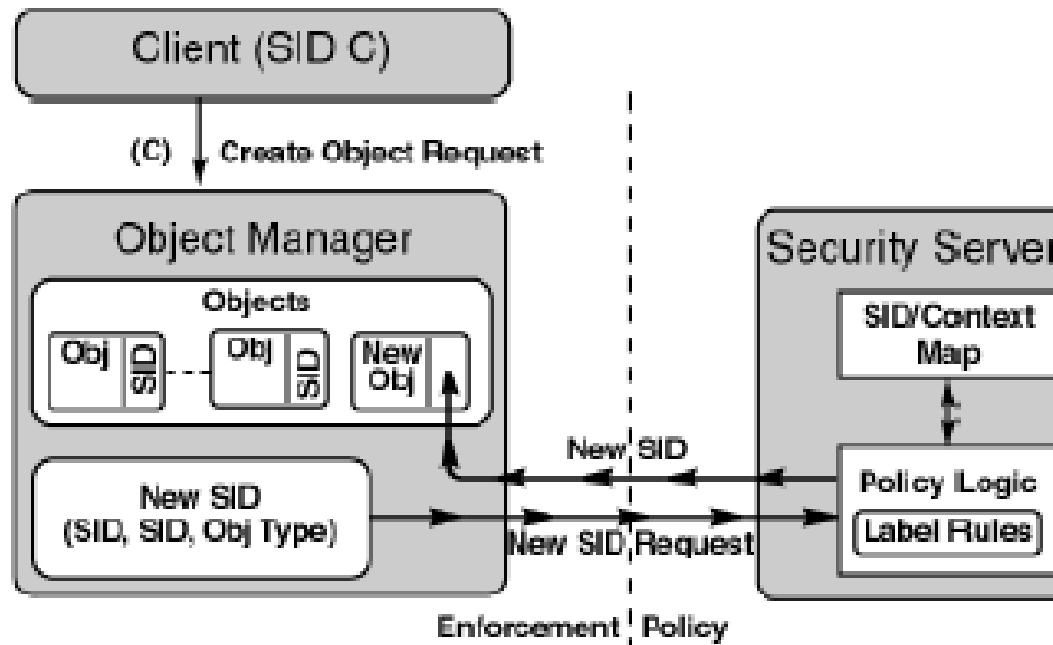
Caching
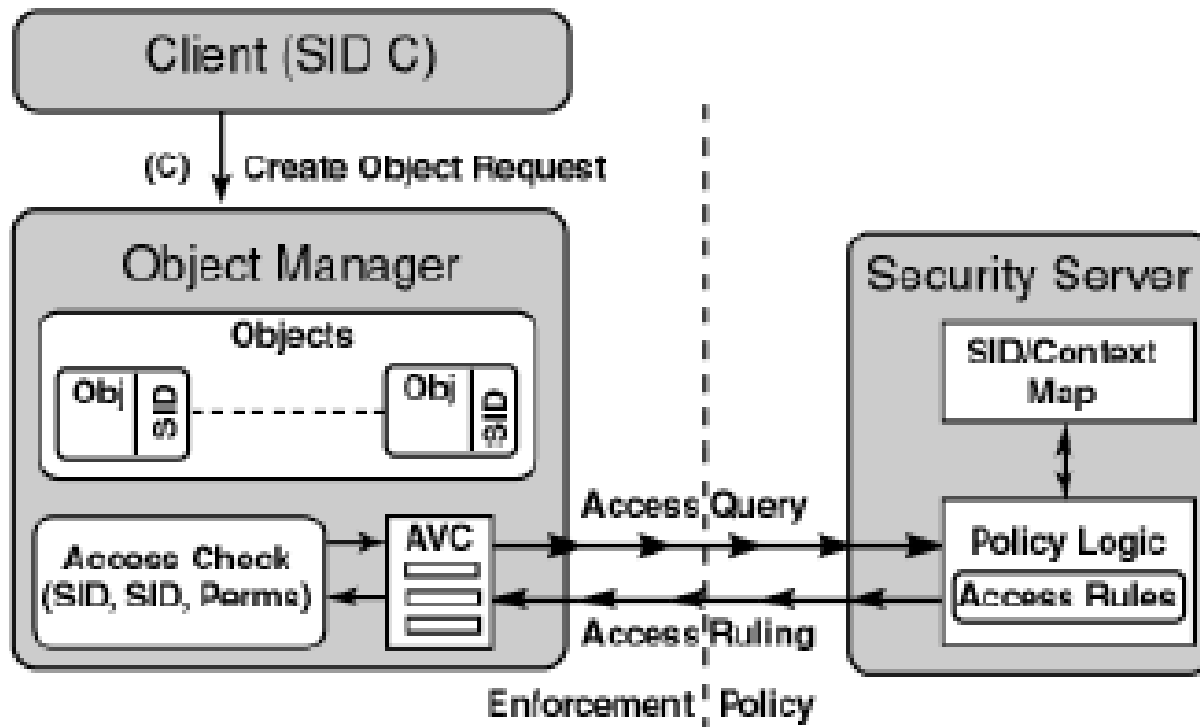
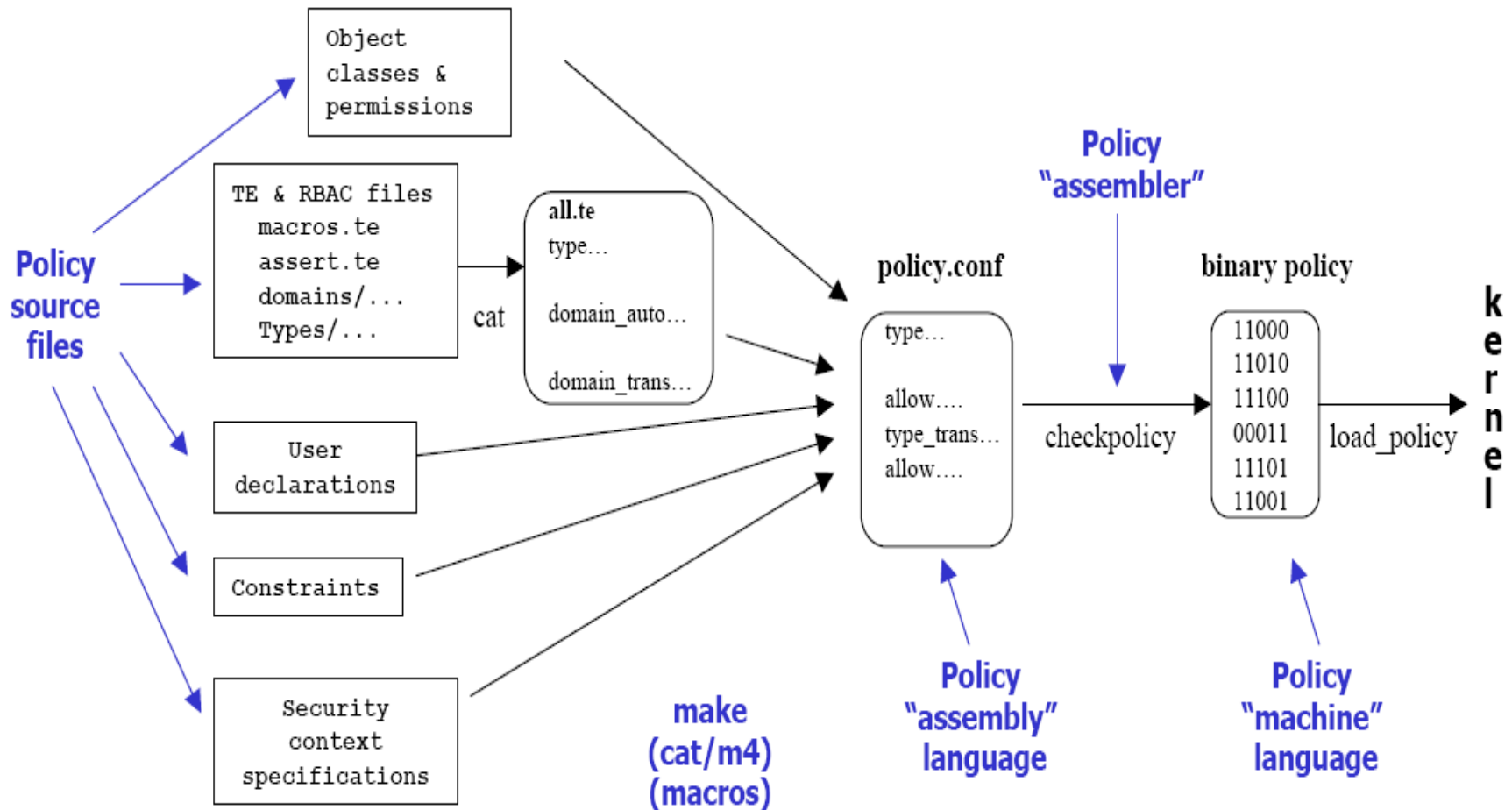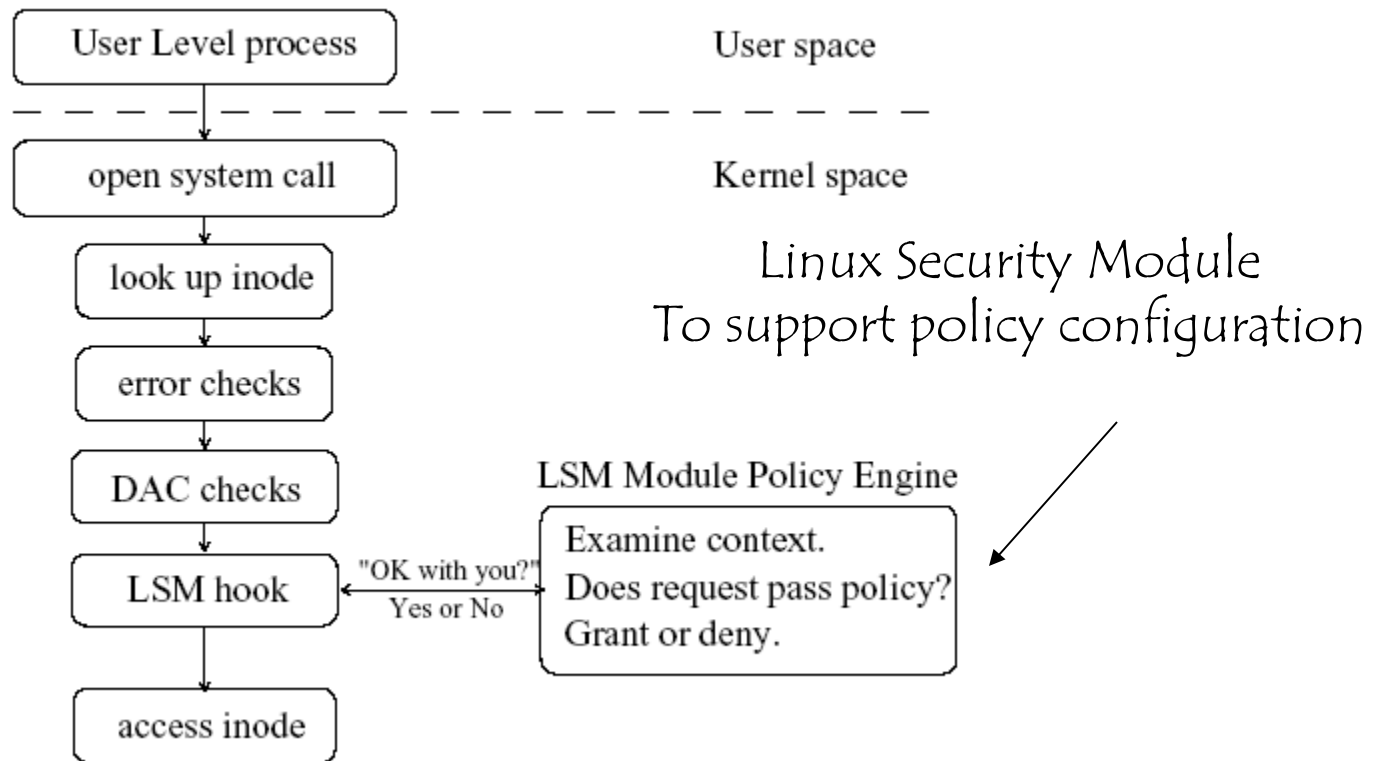We reduce security to reduce the overhead

PSID

SELinux – Policy

- The description of a policy is rather complex even in the case of simple policies
- As an example, to specify the Linux policy
  - 29 types
  - 121 operations
  - 27.000 rules
- Little support for an high level description and to check the consistency of a policy

SELinux – Policy - Tools

SELinux - Implementation

SELinux – Implementation

Implementation of Linux standard
Security policy

Test Type	2.5.15	2.5.15-lsm	% Overhead with LSM
null call	0.49	0.48	-2.0%
null I/O	0.89	0.91	-2.2%
stat	5.39	5.49	1.9%
open/close	6.94	7.13	2.7%
select TCP	39	41	5.1%
sig inst	1.18	1.19	0.8%
sig handl	4.10	4.09	-0.2%
fork proc	187	187	0%
exec proc	705	706	0.1%
sh proc	3608	3611	0.1%

Overhead due to SE

Connection rate measured in connections per second.

Number of clients	Server connection rate 2.5.7	Server connection rate 2.5.7-SEL	% Overhead
8	916.56	766.58	16.4%
16	917.64	766.48	15.5%
24	917.44	765.56	16.6%
32	918.91	764.80	16.8%

Table 7: UP Webstone results comparing SELinux to standard kernel.

This points out that the cost is
- Acceptable if we consider the execution overhead
- Fairly large if we consider the complexity of the description

Example - NSA NetTop
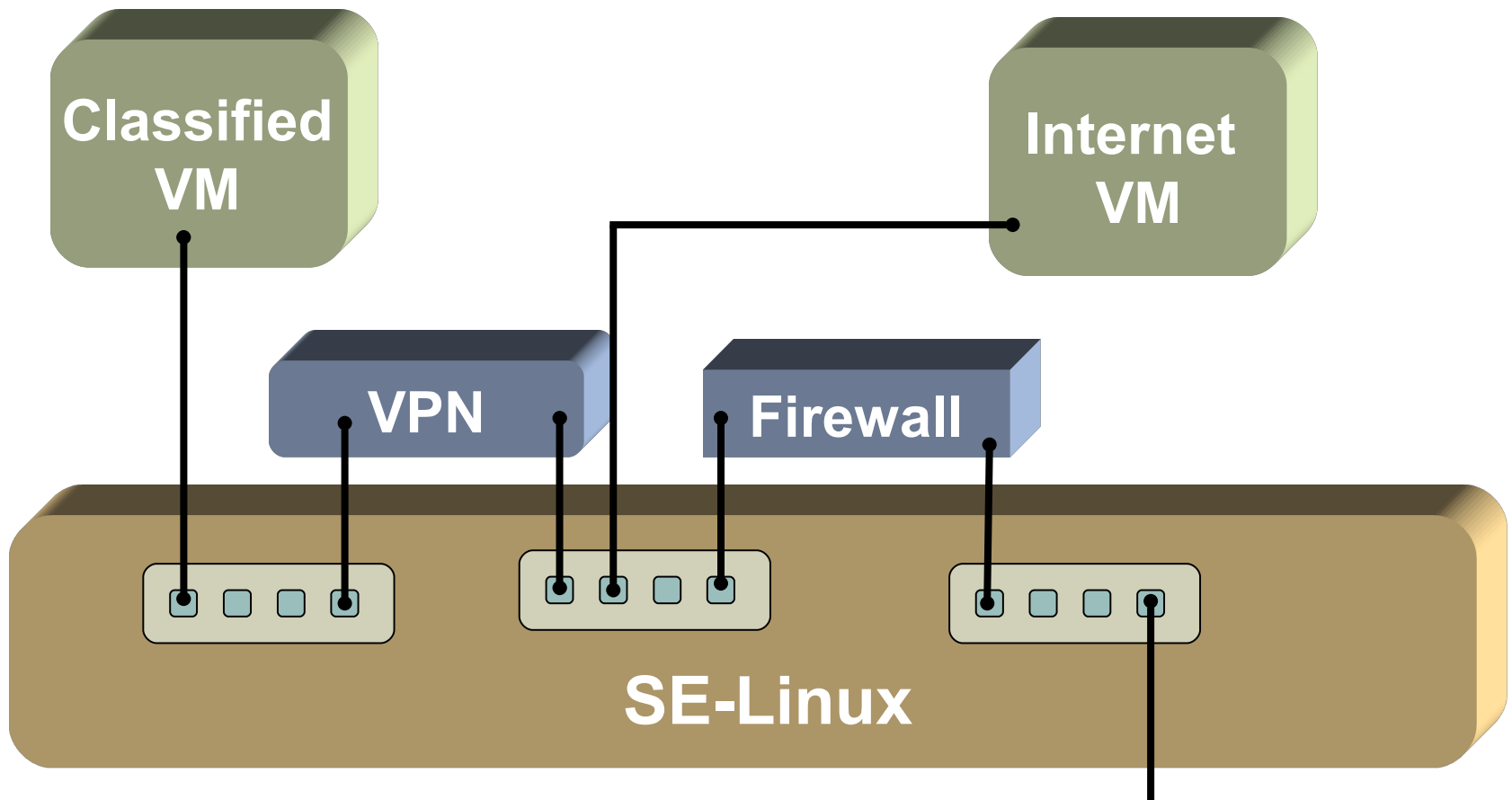
NetTop = SE-Linux + VMware

- SE-Linux:
  - Security-Enhanced Linux
  - Mandatory Access Control with flexible security policy
- VMware Workstation:
  - VMs configuration limited by security policy
- NetTop:
  - Locked-down SE-Linux policy
  - No networking on the host itself

Flexible Networking: VMnets

Virtual networ devices

Physical LAN