

Countermeasures Analysis



The goal of this steps is to determine some changes to the target system to avoid or at least reduce the risk



Countermeasures

A first classification

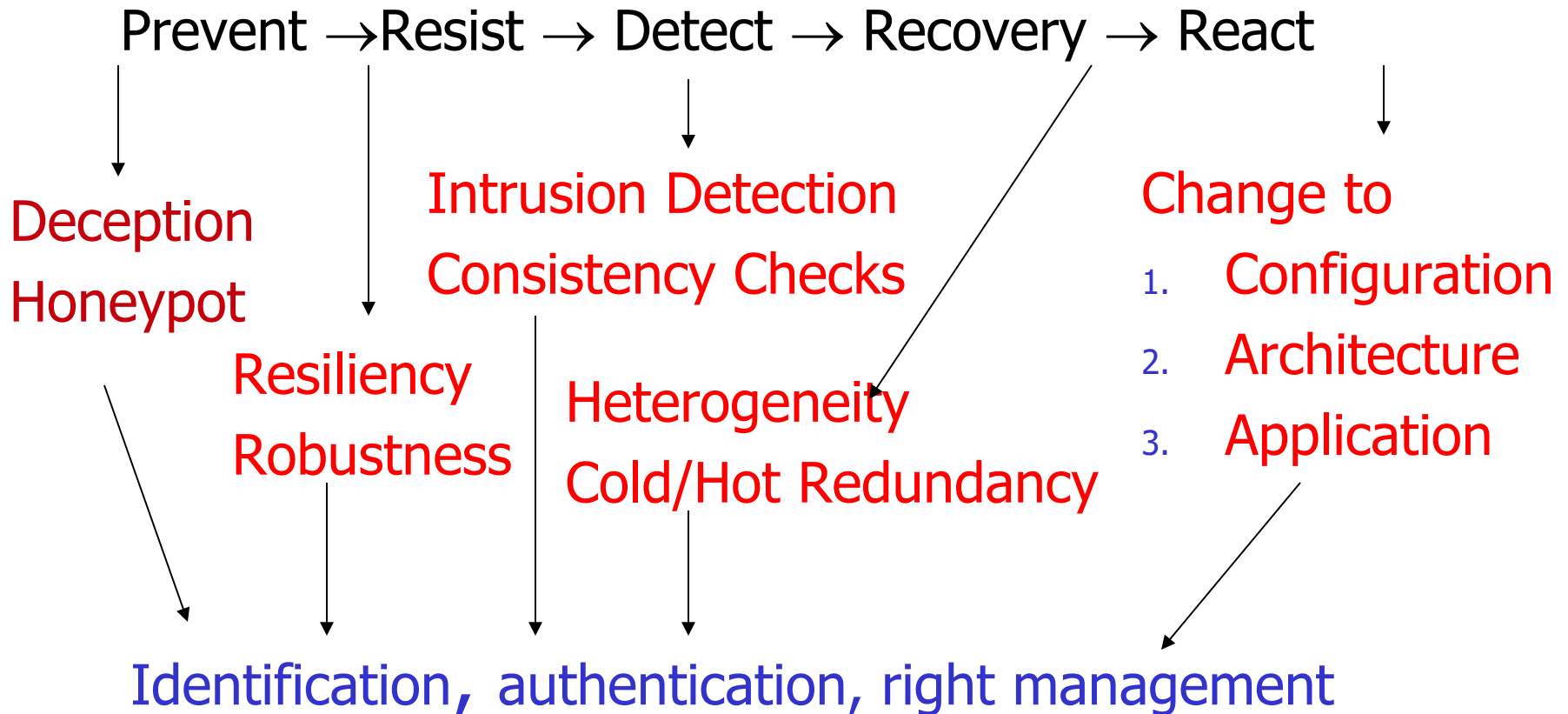
- Proactive
 - They are applied before an attack
eg a vulnerability is removed
- Dynamic
 - They are applied as soon as an attack is detected
eg a vulnerability is removed
eg a connection is killed
- Reactive
 - They are applied after a successful attack
eg a vulnerability is removed
eg a password is changed

Detection?





A more detailed taxonomy





Implementation mechanisms

- Countermeasures are implemented through a set of common mechanisms
- A set of shared mechanisms
 - It can increase the cost effectiveness of countermeasures
 - 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
- Honeytrap = 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



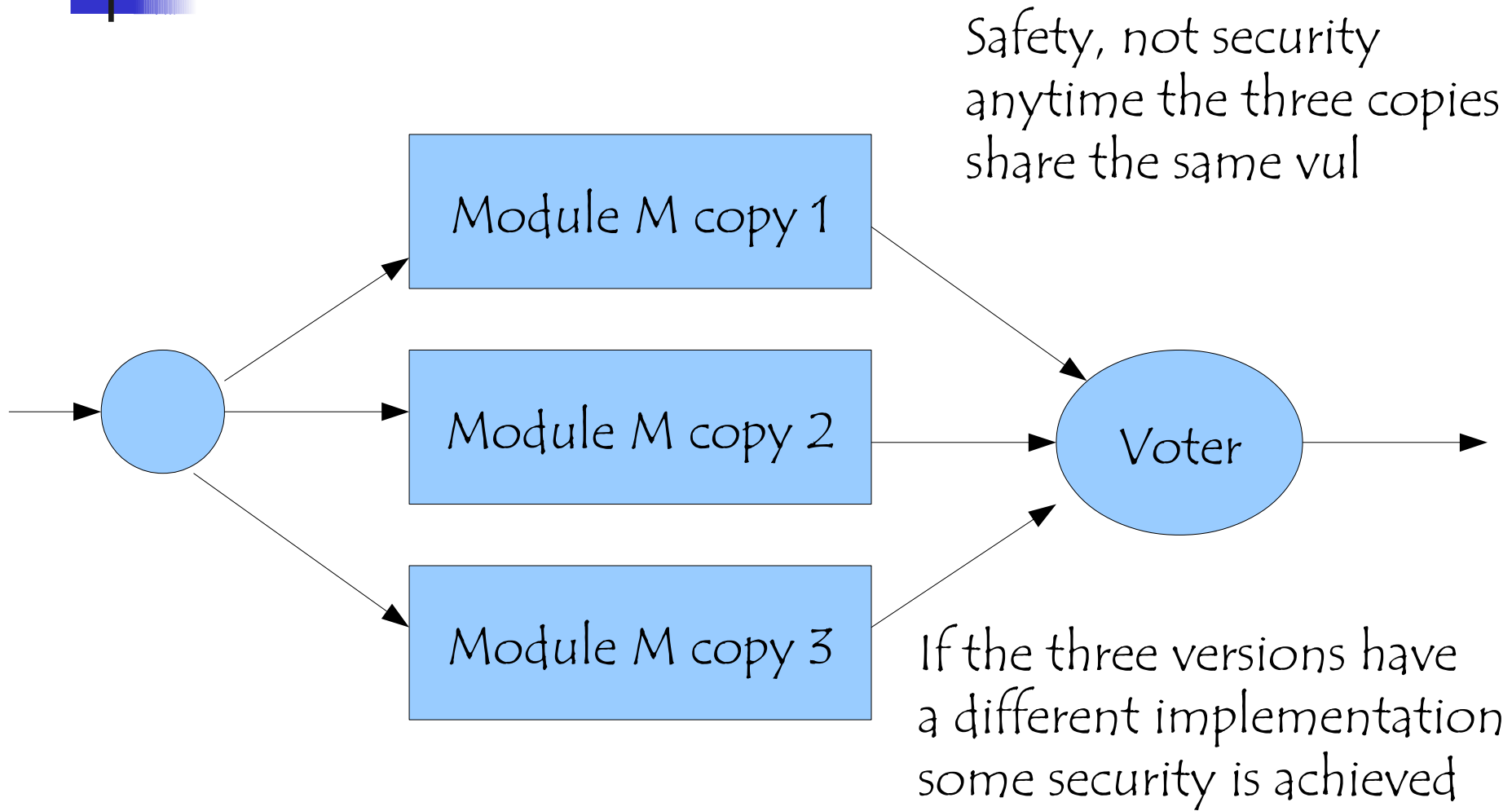
Countermeasures Glossary - II

- 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





Countermeasures Glossary- III

- Minimal system
 - A subset of components
 - More robust
 - Large number of 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 normal vs power law impact



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



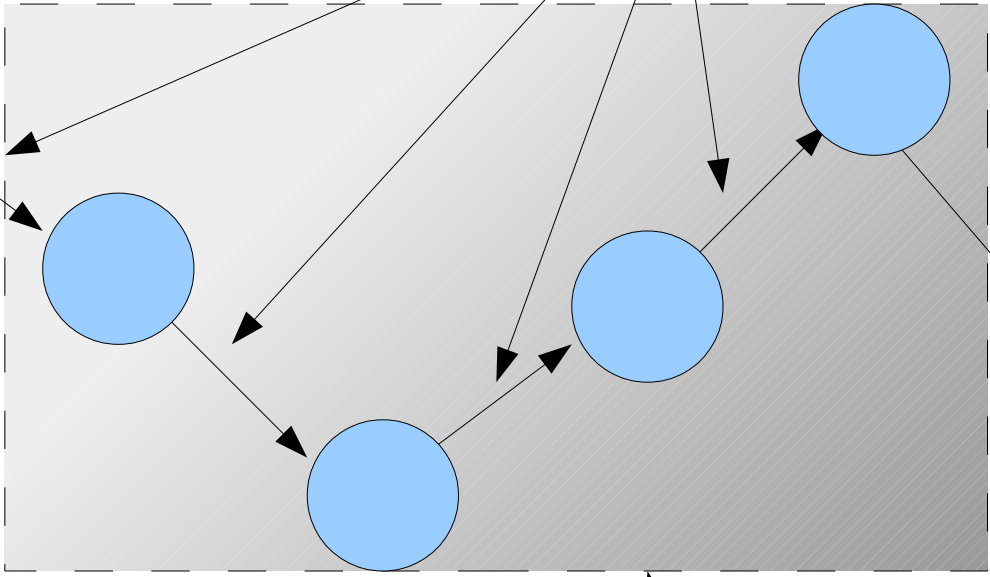
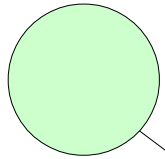
No action on the attacking sys?

- 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?

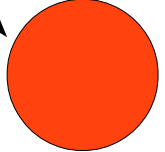
Stepping stone



Attacker node

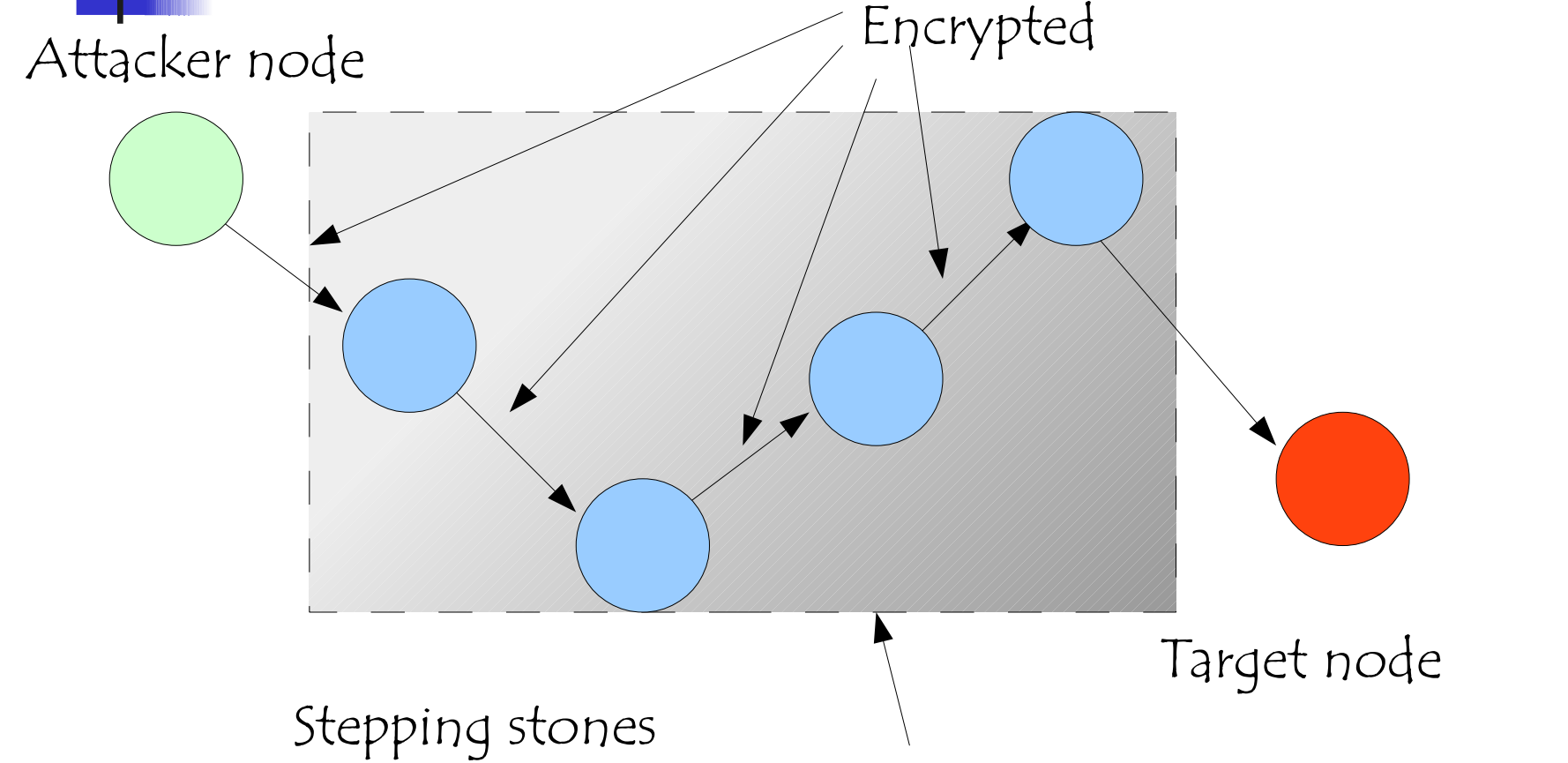


Encrypted



Target node

Stepping stones





Stepping stone - 1

- An analysis of input/output node channel to evaluate their correlation
- If there are an input channel and an output one (i/o port) 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 channels, the whole chain of stepping stones may be discovered

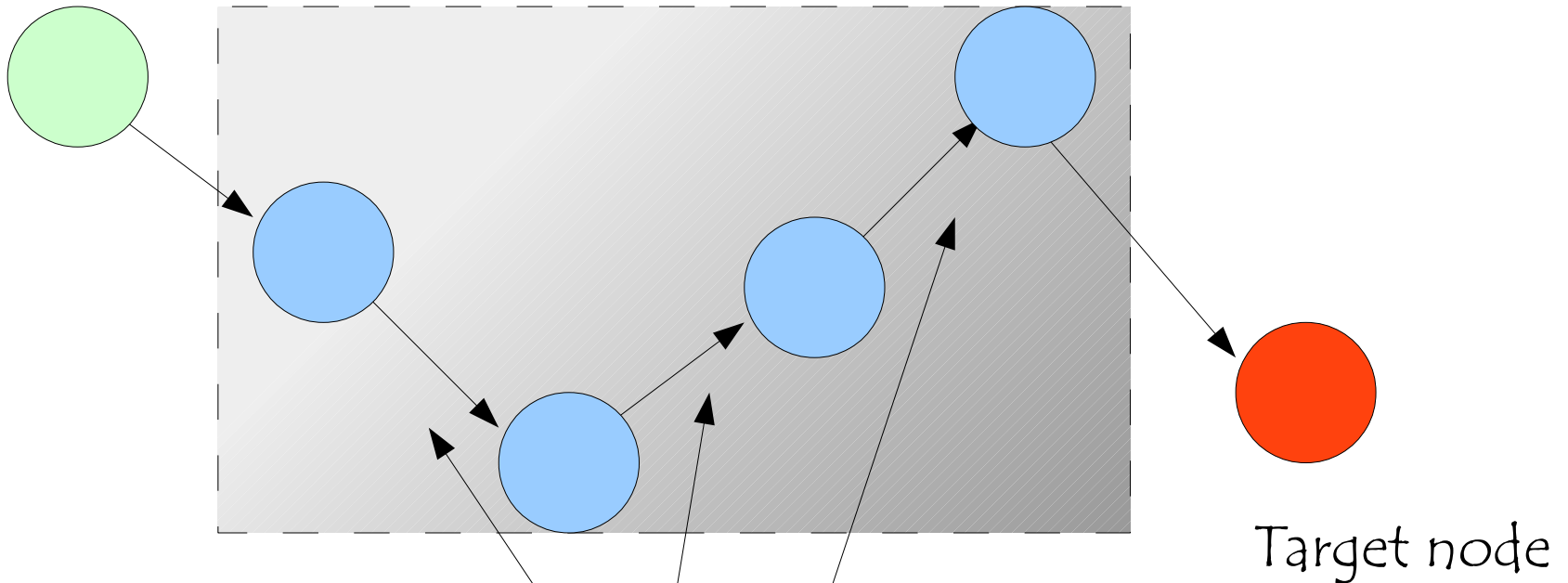


Stepping stone - 2

- 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

Stepping stone

Attacker node



Stepping stones

Correlation among them
can be discovered even
if they are encrypted

Target node



Deception = Honeypot

- 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 it creates a new fake virtual node the attacker has to analyze
- These virtual nodes are useless but how far as the scanning is concerned, they behave like real nodes
- The fake nodes replies to the fingerprinting messages of the scanner are slower and slower to slow down the scanning
- An alarm is raised



Honeytrap - Definition

An ICT resource whose value lies in unauthorized or illicit use of that resource.

- Has no production value; anything going to/from a honeypot is likely a probe, attack or compromise
- Used for monitoring, detecting and analyzing attacks
- Does not solve a specific problem. Instead, they are a highly flexible tool with different applications to security.



Classification

- By level of interaction
 - High
 - Low
 - Middle
- By Implementation
 - Virtual
 - Physical
- By purpose
 - Production
 - Research



Level of Interaction

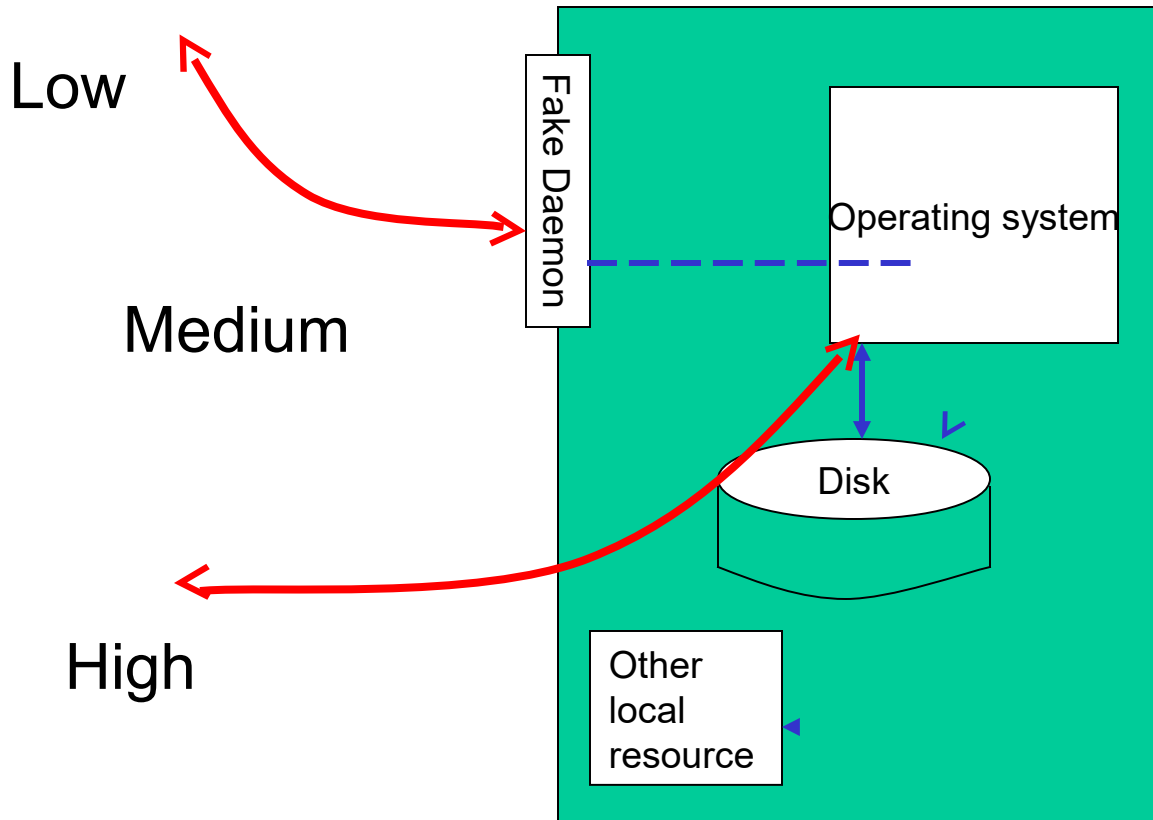
- **Low Interaction**

- Simulates some aspects of the system
- Easy to deploy, minimal risk
- Limited Information
- OS tool - Honeyd

- **High Interaction**

- Simulates all aspects of the OS: real systems
- Can be compromised completely, higher risk
- More Information
- OS tool - Honeydnet

Level of Interaction





Physical V.S. Virtual Honeypots

- Two types
 - Physical
 - Real machines
 - Own IP Addresses
 - Often high-interactive
 - Virtual
 - Simulated by other machines that:
 - Respond to the traffic sent to the honeypots
 - May simulate distinct virtual honeypots at the same time



Production HPs: Protect the systems

- Prevention

- Keeping the bad guys out
- not effective prevention mechanisms.
- Deception, Deterrence, Decoys do NOT work against automated attacks: worms, auto-rooters, mass-rooters

- Detection

- Detecting the burglar when he breaks in.
- Great work

- Response

- Can easily be pulled offline
- Little to no data pollution



Research HPs: gathering information

- Collect compact amounts of high value information
- Discover new Tools and Tactics
- Understand Motives, Behavior, and Organization
- Develop Analysis and Forensic Skills
- Used to discover new worms/viruses and their signatures



Building your HoneyPots

- Specifying Goals
- Selecting the implementation strategies
 - Types, Number, Locations and Deployment
- Implementing Data Capture
- Logging and managing data
- Mitigating Risk
- Mitigating Fingerprint



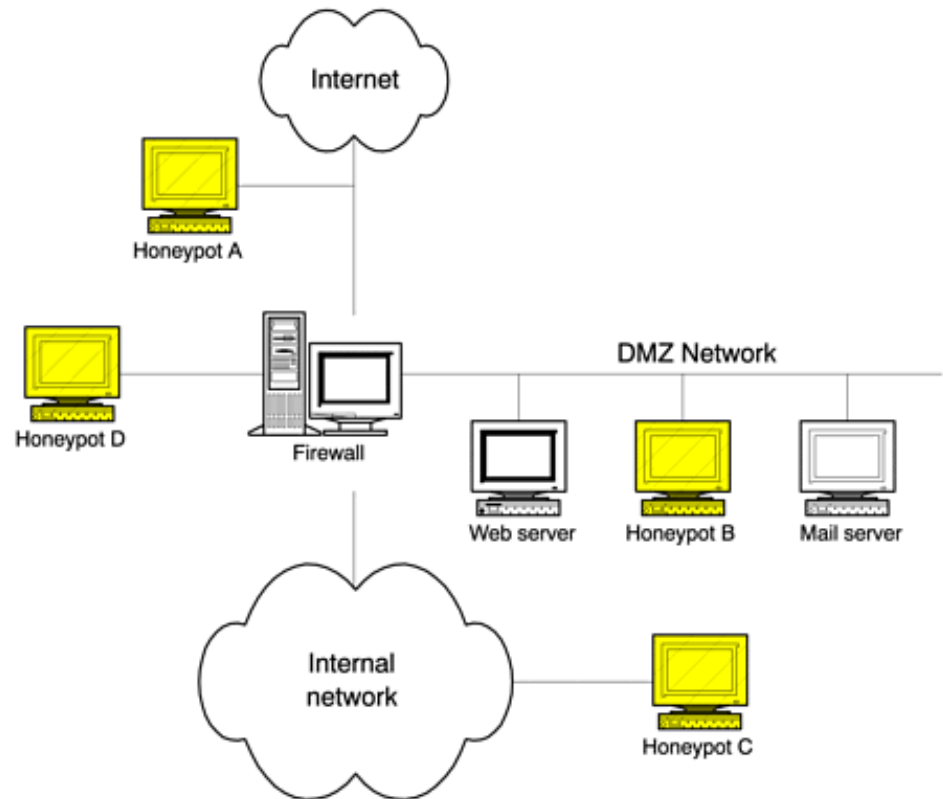
Just an anticipation ...

■ Firewall

- A system that connects two networks with distinct security requirements
- It filters the information flowing across 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 expense of the bandwidth between the two networks

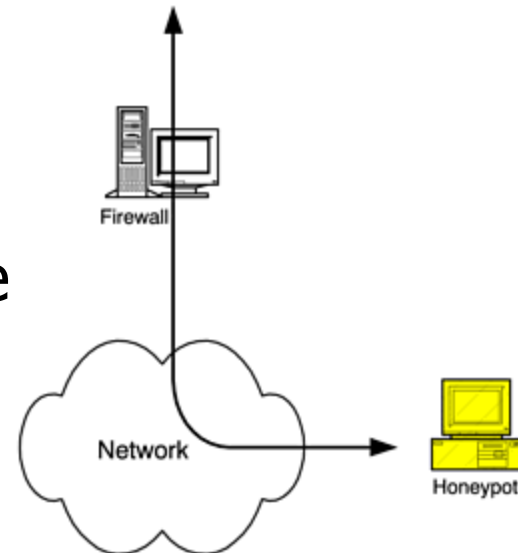
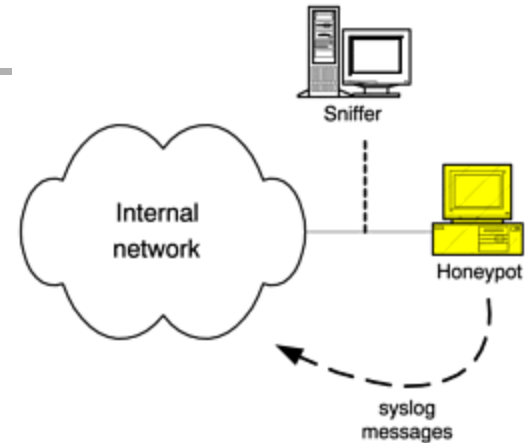
Location of Honeypots

- In front of the firewall
- Demilitarized Zone
- Behind the firewall (Intranet)

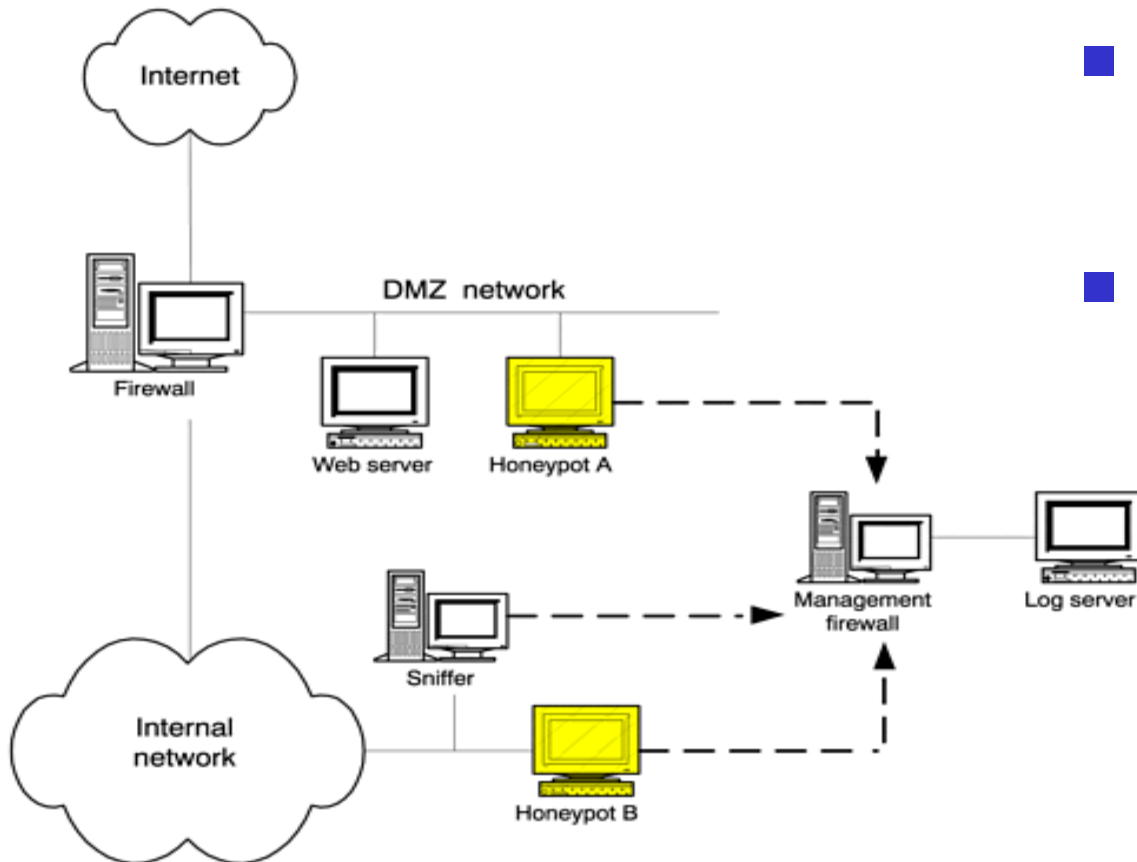


Capturing Information

- Host based:
 - Keystrokes
 - Syslog
- Network based:
 - Firewall
 - Sniffer
 - IP not resolved name



Logging and Managing Data



- Logging architecture
- Managing data



What is Honeyd?

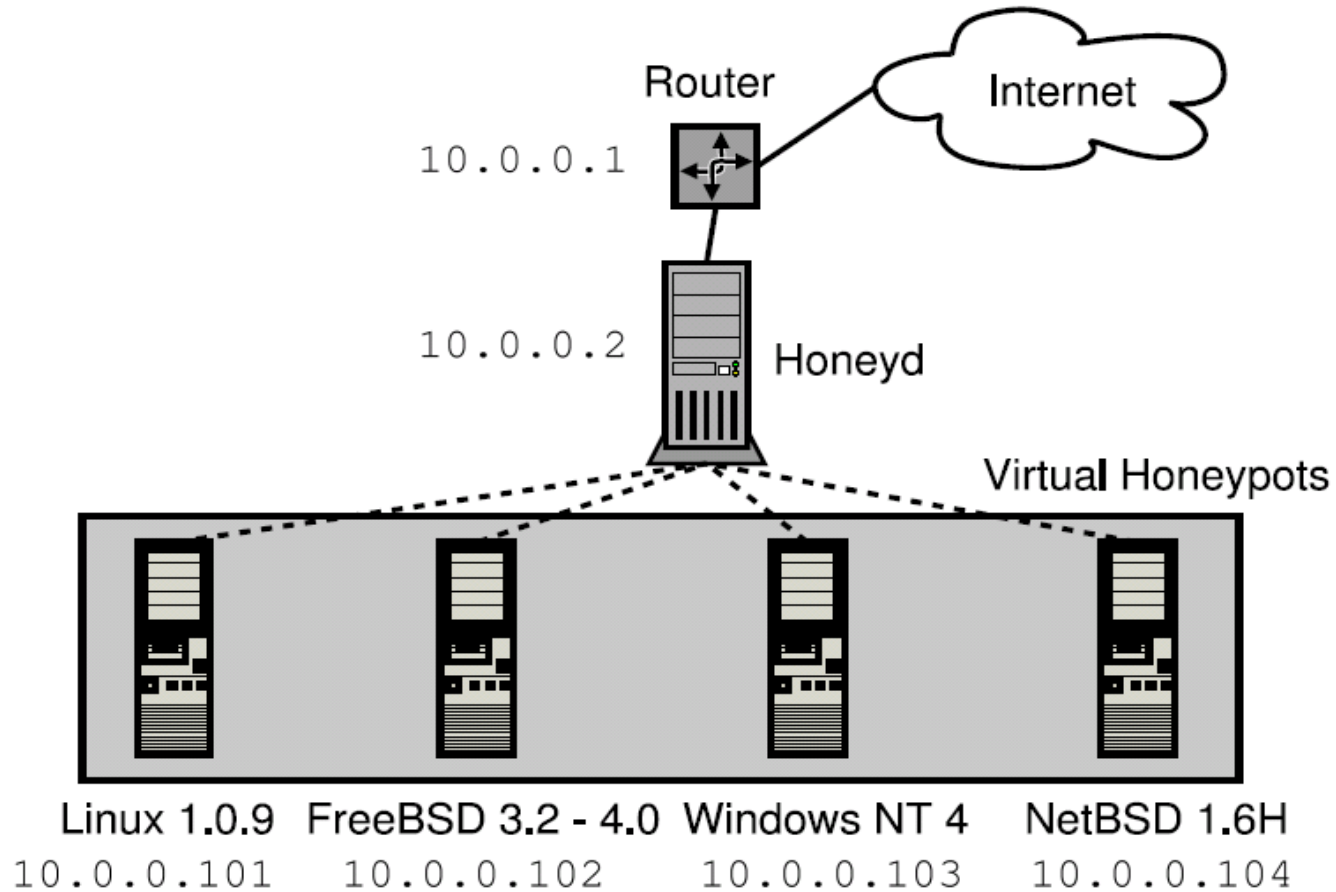
- **Honeyd**: A virtual honeypot application, which allows us to create thousands of IP addresses with virtual machines and corresponding network services.
- Written by Neil Provos available at <http://www.honeyd.org/>



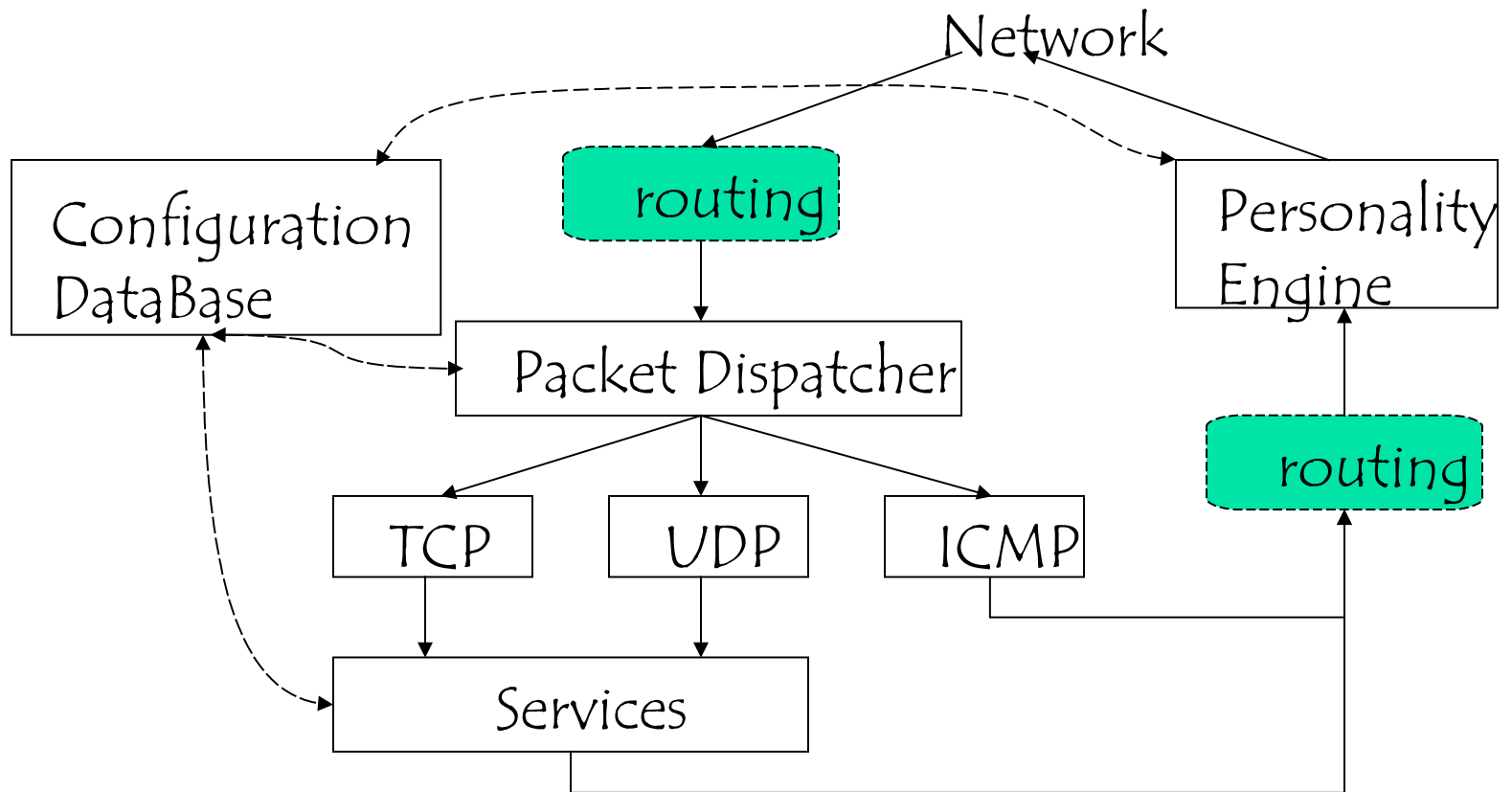
What can honeyd do?

- Simulates operating systems at TCP/IP stack level, supporting TCP/UDP/ICMP;
- Support arbitrary services;
- Simulate arbitrary network topologies;
- Support tunneling and redirecting net traffic;

Illustration Simple



How it works?





Why Personality Engine?

- To fool fingerprinting tools
- Uses fingerprint databases by
 - Nmap, for TCP, UDP
 - Xprobe, for ICMP
- Introduces changes to the headers of every outgoing packet before it is sent to the network



Why Routing topology?

- Simulates virtual network topologies;
 - Some honeypots are also configured as routers
 - Latency and loss rate for each edge is configured;
- Support network tunneling and traffic redirection;



What is a Honeynet <> Honeytrap

- High-interaction honeypot designed to:
 - capture in-depth *information*
 - learn who would like to use your system without your permission for their own ends
- Its an architecture, not a product or software.
- Populate with live systems.
- Can look like an actual production system



What is a Honeynet

- Once compromised, data is collected to learn the tools, tactics, and motives of the blackhat community.
- Information has different value to different organizations.
 - Learn vulnerabilities
 - Develop response plans

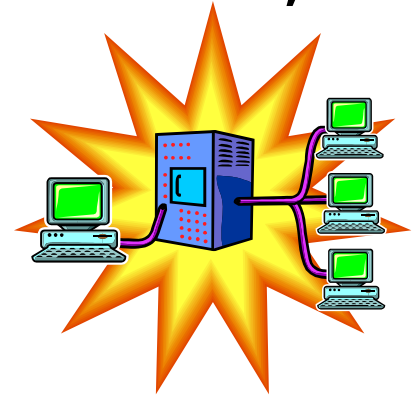


What's The Difference?

- Honeypots use known vulnerabilities to lure attack.
 - Configure a single system with special software or system emulations
 - Want to find out actively who is attacking the system
- Honeynets are networks open to attack
 - often use default installations of system software
 - behind a firewall
 - hope attackers mess up the Honeynet instead than your production system

How it works

- A highly controlled network where every packet entering or leaving is monitored, captured, and analyzed.



- Any traffic entering or leaving the Honeynet is suspect by nature.



Countermeasures - Deception

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



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



Just a reminder ...

- Hiding and protecting
 - Information at rest
 - Exchanged information
- Authentication (digital signature)
- Integrity (hash function)
- Coprocessor (smartcard)
- Symmetric and Asymmetric



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 automaton
- 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 its 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

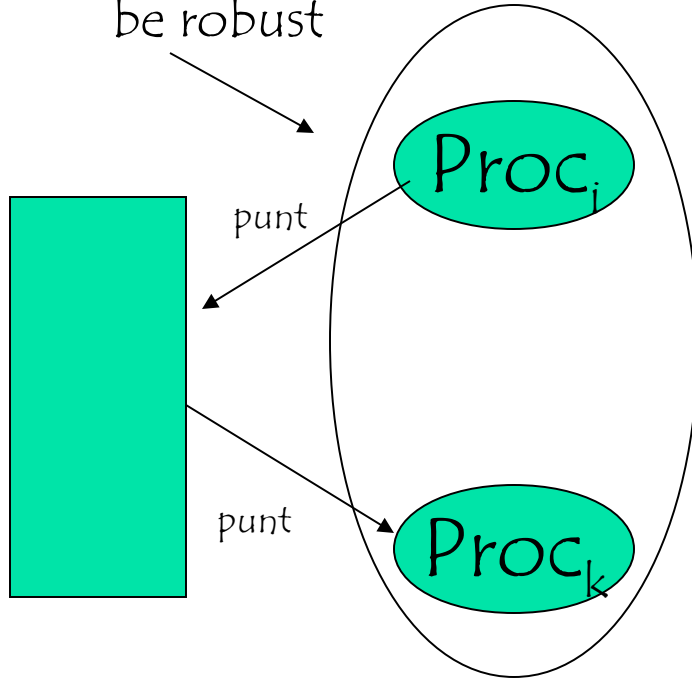


Robust programming – robust implementation - I

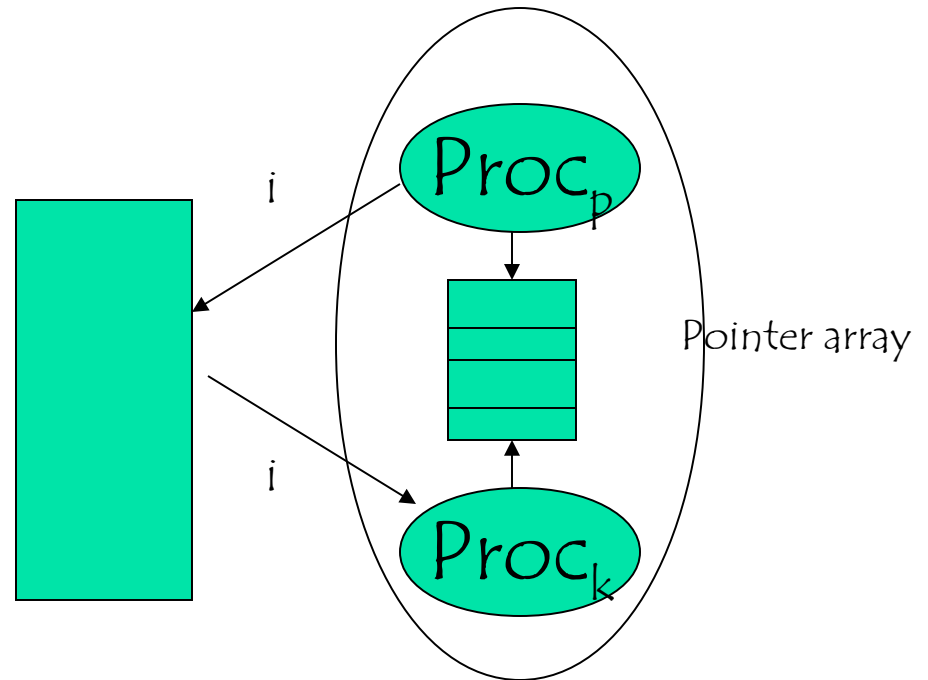
- Satisfy S&S
- Rigorous definition of the program interface
- Do not assume that input/output values are related
 - If a function of a library returns a pointer and another function of the same library has a pointer parameter, there is no reason to assume that the one transmitted to the second is the 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

Package that should
be robust



A more robust version



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
$$\text{realposition} = m * \text{returnedpos} + \text{cost}$$
- 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 explaining 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 sacrificial 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

- Are 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 - 1

Insecure Interaction Between Components

These weaknesses are related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threads, or systems.

For each weakness, its ranking in the general list is provided in square brackets.

Rank	CWE ID	Name
[1]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[4]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[9]	CWE-434	Unrestricted Upload of File with Dangerous Type
[12]	CWE-352	Cross-Site Request Forgery (CSRF)
[22]	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')



SQL Iniection

Hack Me! SQL Injection

Member Login

Username :

Password :

Login

SQL Iniection

- ▣ `function connect_to_db() { ... }`
- ▣ `function display_form() { ... }`
- ▣ `function grant_access() { ... }`
- ▣ `function deny_access() { ... }`

```
connect_to_db();

if (!isset($_POST['submit'])) {
    display_form();
}
else{
    // Get Form Data
    $user = stripslashes($_POST["username"]);
    $pass = stripslashes($_POST["password"]);

    // Run Query
    $query = "SELECT * FROM `login` WHERE `user`='$_user' AND `pass`='$_pass'";
    echo $query . "<br><br>";
    $SQL = mysql_query($query);

    // If user / pass combo found, grant access
    if(mysql_num_rows($SQL) > 0)
        grant_access();

    // Otherwise deny access
    else
        deny_access();
}
```



Hack Me! SQL Injection

timbo317

cse7330



Member Login

Username :

Password :

Login

```
SELECT * FROM `login` WHERE `user`='timbo317' AND `pass`='cse7330'
```

SQL Injection

Hack Me! SQL Injection

' OR 'a'='a

' OR 'a'='a

Member Login

Username :

Password :

Login

```
SELECT * FROM `login` WHERE `user`=' ' OR 'a'='a' AND  
`pass`=' ' OR 'a'='a'
```



SQL Injection

Hack Me! SQL Injection

```
' ; DROP TABLE `login` ; --
```



Member Login

Username :

Password :

Login



The list - 2

Risky Resource Management

The weaknesses in this category are related to ways in which software does not properly manage the creation, usage, transfer, or destruction of important system resources.

Rank	CWE ID	Name
[3]	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[13]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[14]	CWE-494	Download of Code Without Integrity Check
[16]	CWE-829	Inclusion of Functionality from Untrusted Control Sphere
[18]	CWE-676	Use of Potentially Dangerous Function
[20]	CWE-131	Incorrect Calculation of Buffer Size
[23]	CWE-134	Uncontrolled Format String
[24]	CWE-190	Integer Overflow or Wraparound



The list - 3

Porous Defenses

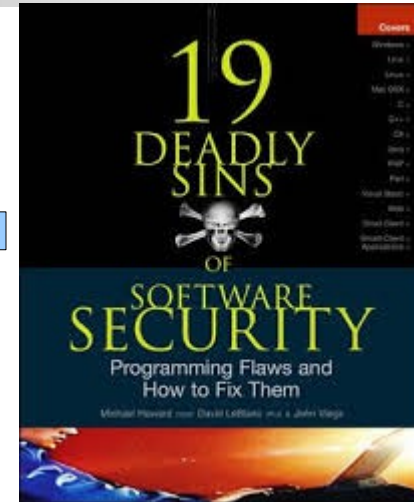
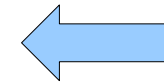
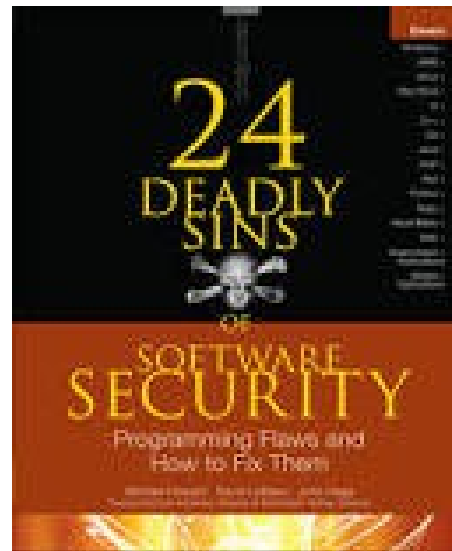
The weaknesses in this category are related to defensive techniques that are often misused, abused, or just plain ignored.

Rank	CWE ID	Name
[5]	CWE-306	Missing Authentication for Critical Function
[6]	CWE-862	Missing Authorization
[7]	CWE-798	Use of Hard-coded Credentials
[8]	CWE-311	Missing Encryption of Sensitive Data
[10]	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	CWE-250	Execution with Unnecessary Privileges
[15]	CWE-863	Incorrect Authorization
[17]	CWE-732	Incorrect Permission Assignment for Critical Resource
[19]	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
[21]	CWE-307	Improper Restriction of Excessive Authentication Attempts
[25]	CWE-759	Use of a One-Way Hash without a Salt

24 sins...

5 kinds of sins

- Web
- Implementation
- Cryptographic
- Network
- Stored Data



■ Web Application Sins;

- SQL Injection;
- Server Side Cross-Site Scripting
- Web-Client Related Vulnerabilities;



24 sins...

- 4 kinds of sins
 - Web
 - Implementation
 - Cryptographic
 - Network
- Web Application Sins;
 - SQL Injection;
 - Server Side Cross-Site Scripting
 - Web-Client Related Vulnerabilities;



24 sins...

- Implementation Sins
 - Use of Magic URLs
 - Buffer Overruns;
 - Format String Problems;
 - Integer Overflows;
 - C++ Catastrophes;
 - Catching All Exceptions;
 - Command Injection;
 - Failure to Handle Errors;
 - Information Leakage;
 - Race Conditions;
 - Poor Usability; Chapter
 - Not Updating Easily;



24 sins...

- Cryptographic Sins
 - Not Using Least Privileges;
 - Weak Password Systems;
 - Unauthenticated Key Exchange;
 - Random Numbers;
- Networking Sins;
 - Wrong Algorithm;
 - Failure to Protect Network Traffic;
 - Trusting Name Resolution;
- Stored Data Sins;
 - Improper Use of SSL/TLS;
 - Failure to Protect Stored Data



Countermeasures – Resist – First Step

- 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



Countermeasures – Resist – Second Step

■ Firewall

- A system that connects two networks with distinct security requirements
- It filters the information flowing across 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 expense of the bandwidth between the two networks

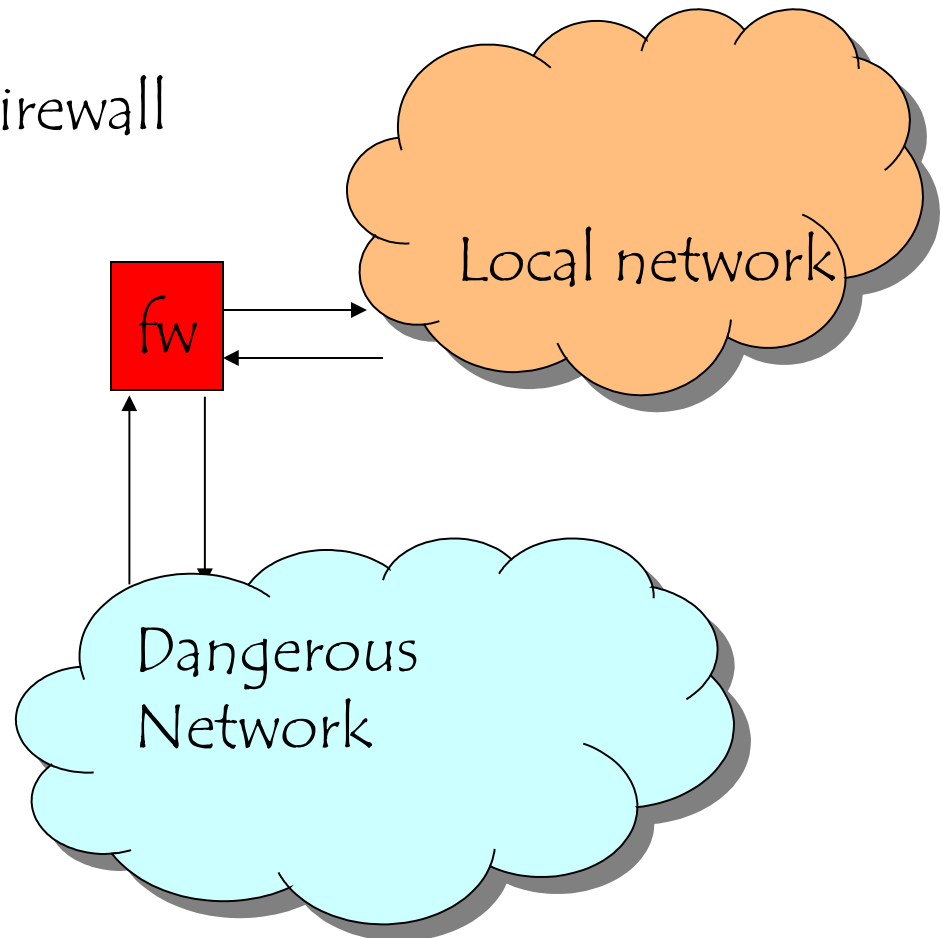
Introducing a Firewall



Initial configuration

After introducing the fw

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 transmitted through legal connections
 - It can force stronger user authentication when it generates connections 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

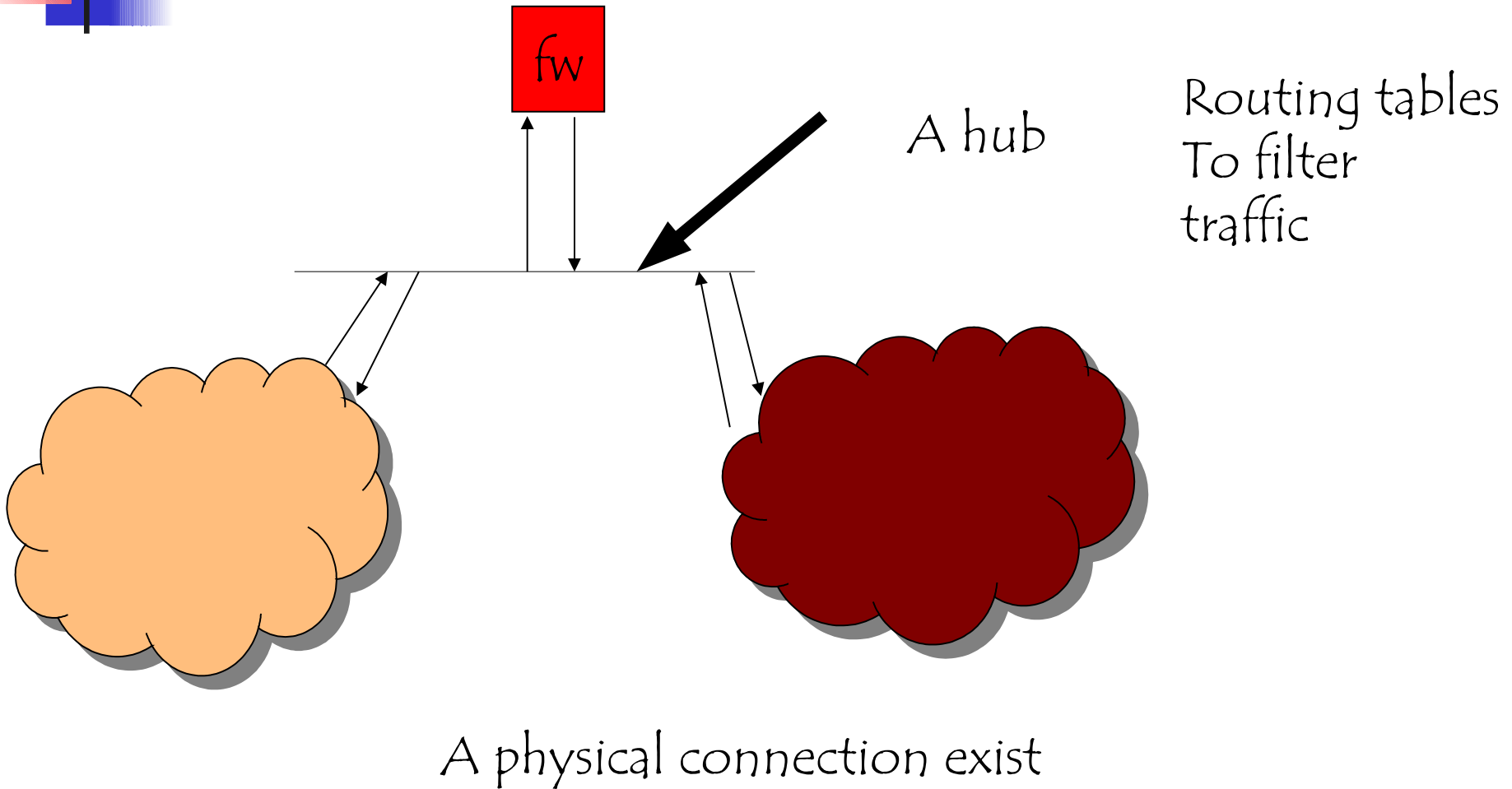
- A firewall is characterized by
 - The protocols it knows and can analyze (communication stack layers it can analyze to protect a network)
 - Its architecture (router, dedicated node, router+ dedicated node)
 - The two properties are distinct and fully orthogonal and they determine the overall robustness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



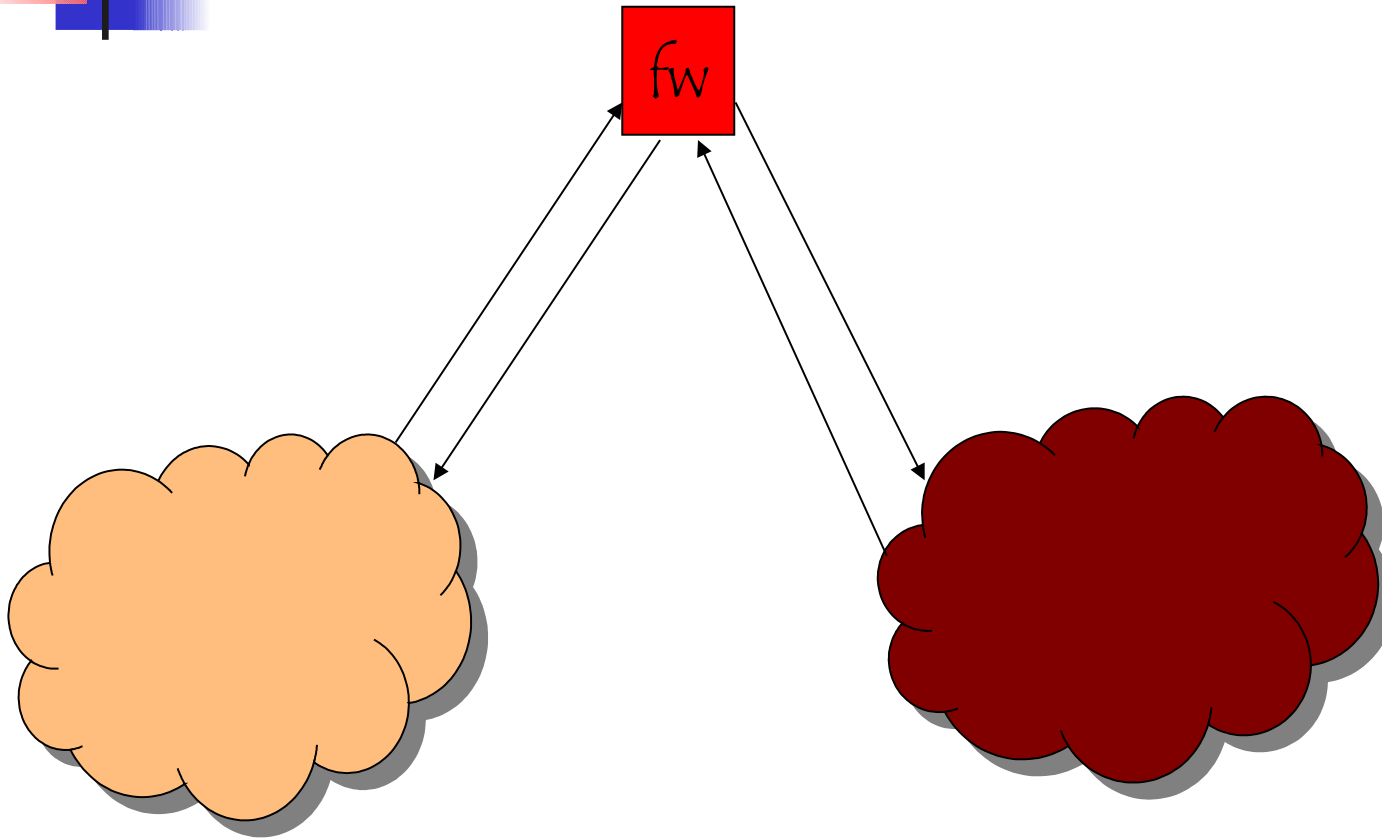
An example - I

- 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

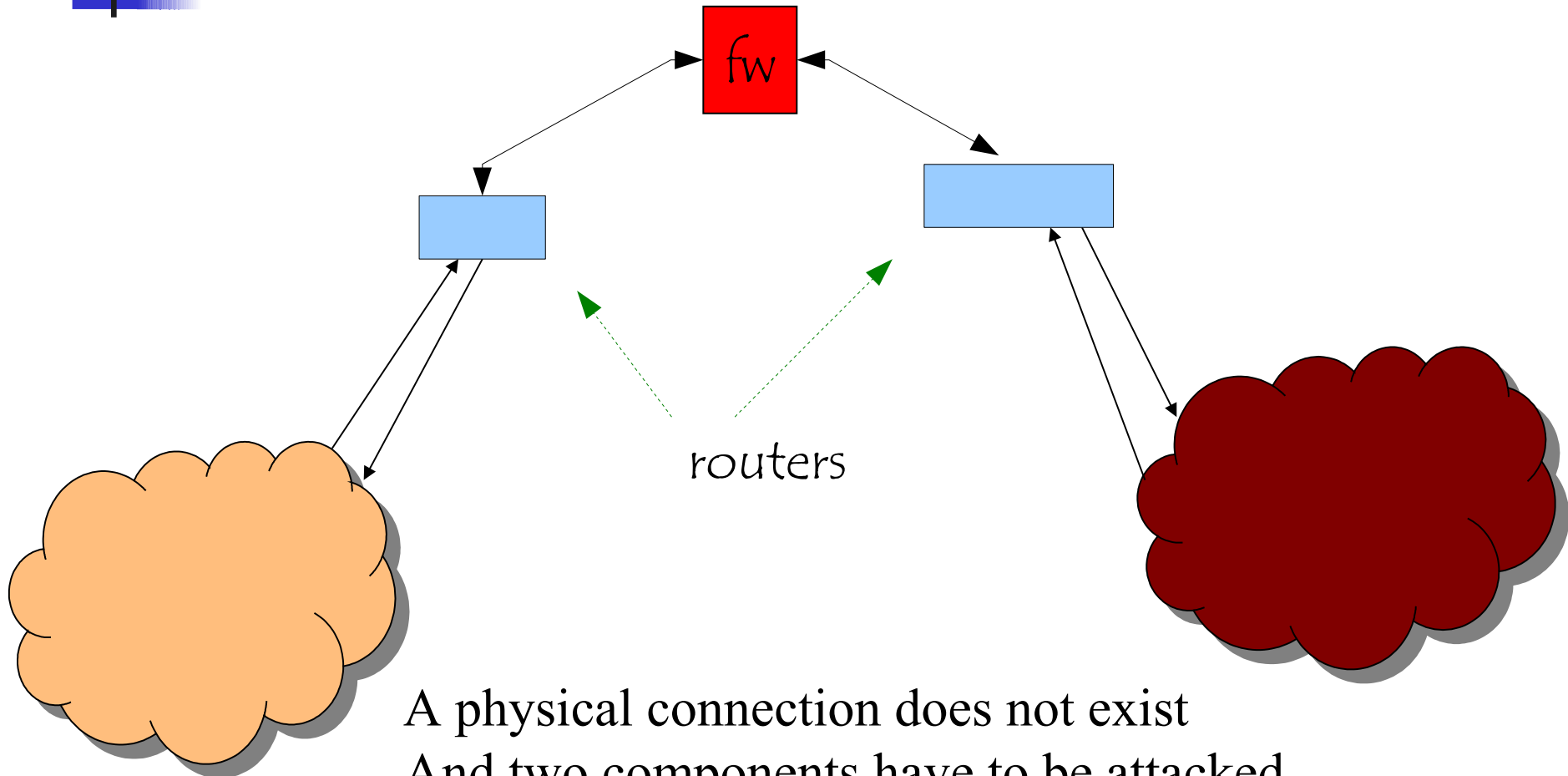


Some architectures - 2



A physical connection does not exist

Some architectures - 3



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



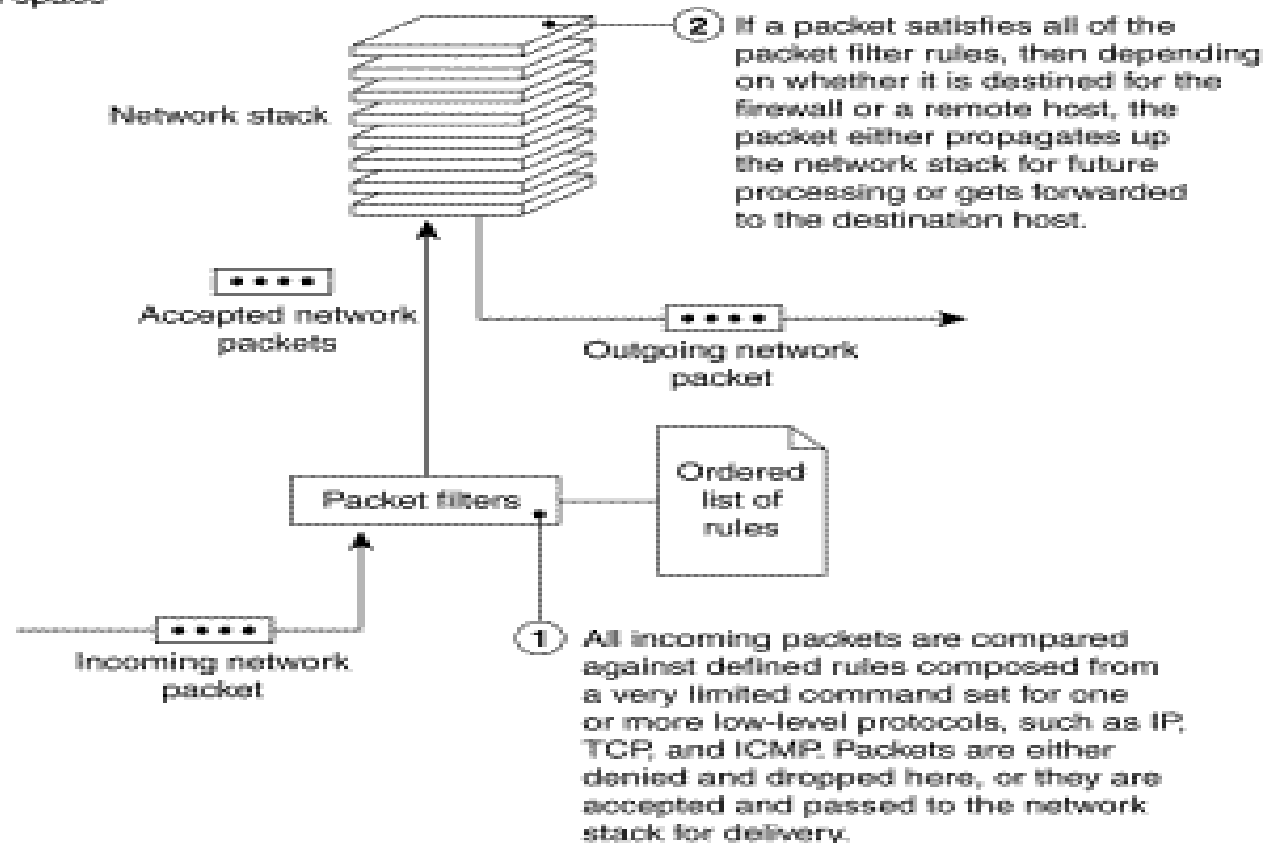
Controls

- Controls implemented through rules route/drop according to some conditions
- The conditions are related to the protocol
- The simplest case:
 - ACL in a router (see in S&S) rather than a distinct node = a layer 3 firewall conditions on ports and hosts
 - it can prevent the opening of an outbound connection by checking the bits in an IP packet (three way handshake)
- 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



Firewalls – Packet Filters

Simplest of components

Uses transport-layer information only

- IP Source Address, Destination Address
- Protocol/Next Header (TCP, UDP, ICMP, etc)
- TCP or UDP source & destination ports
- TCP Flags (SYN, ACK, FIN, RST, PSH, etc)
- ICMP message type

Examples

- DNS uses port 53
 - No incoming port 53 packets except known trusted servers

Usage of Packet Filters

Filtering with incoming or outgoing interfaces

- E.g., Ingress filtering of spoofed IP addresses
- Egress filtering

Permits or denies certain services

- Requires intimate knowledge of TCP and UDP port utilization on a number of operating systems

How to Configure a Packet Filter

Start with a security policy

Specify allowable packets in terms of logical expressions on packet fields

Rewrite expressions in syntax supported by your vendor

General rules - least privilege

- All that is not expressly permitted is prohibited
- If you do not need it, eliminate it

Every ruleset is followed by an implicit rule reading like this.

action	ourhost	port	theirhost	port	comment
block	*	*	*	*	<i>default</i>

Example 1:

Suppose we want to allow inbound mail (SMTP, port 25) but only to our gateway machine. Also suppose that traffic from some particular site SPIGOT is to be blocked.

Solution 1:

action	ourhost	port	theirhost	port	comment
block	*	*	SPIGOT	*	<i>we don't trust these people</i>
allow	OUR-GW	25	*	*	<i>connection to our SMTP port</i>

Example 2:

Now suppose that we want to implement the policy "any inside host can send mail to the outside".

Solution 2:

action	ourhost	port	theirhost	port	comment
allow	*	*	*	25	<i>connection to their SMTP port</i>

This solution allows calls to come from any port on an inside machine, and will direct them to port 25 on the outside. Simple enough...

So why is it wrong?

Our defined restriction is based solely on the outside host's port number, which we have no way of controlling.

Now an enemy can access any internal machines and port by originating his call from port 25 on the outside machine.

What can be a better solution ?

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	25		<i>our packets to their SMTP port</i>
allow	*	25	*	*	ACK	<i>their replies</i>

- The ACK signifies that the packet is part of an ongoing conversation
- Packets without the ACK are connection establishment messages, which we are only permitting from internal hosts

Security & Performance of Packet Filters

Tiny fragment attacks

- Split TCP header info over several tiny packets
- Either discard or reassemble before check

Degradation depends on number of rules applied at any point

Order rules so that most common traffic is dealt with first

Correctness is more important than speed

Port Numbering

TCP connection

- Server port is number less than 1024
- Client port is number between 1024 and 16383

Permanent assignment

- Ports <1024 assigned permanently
 - 20,21 for FTP 23 for Telnet
 - 25 for server SMTP 80 for HTTP

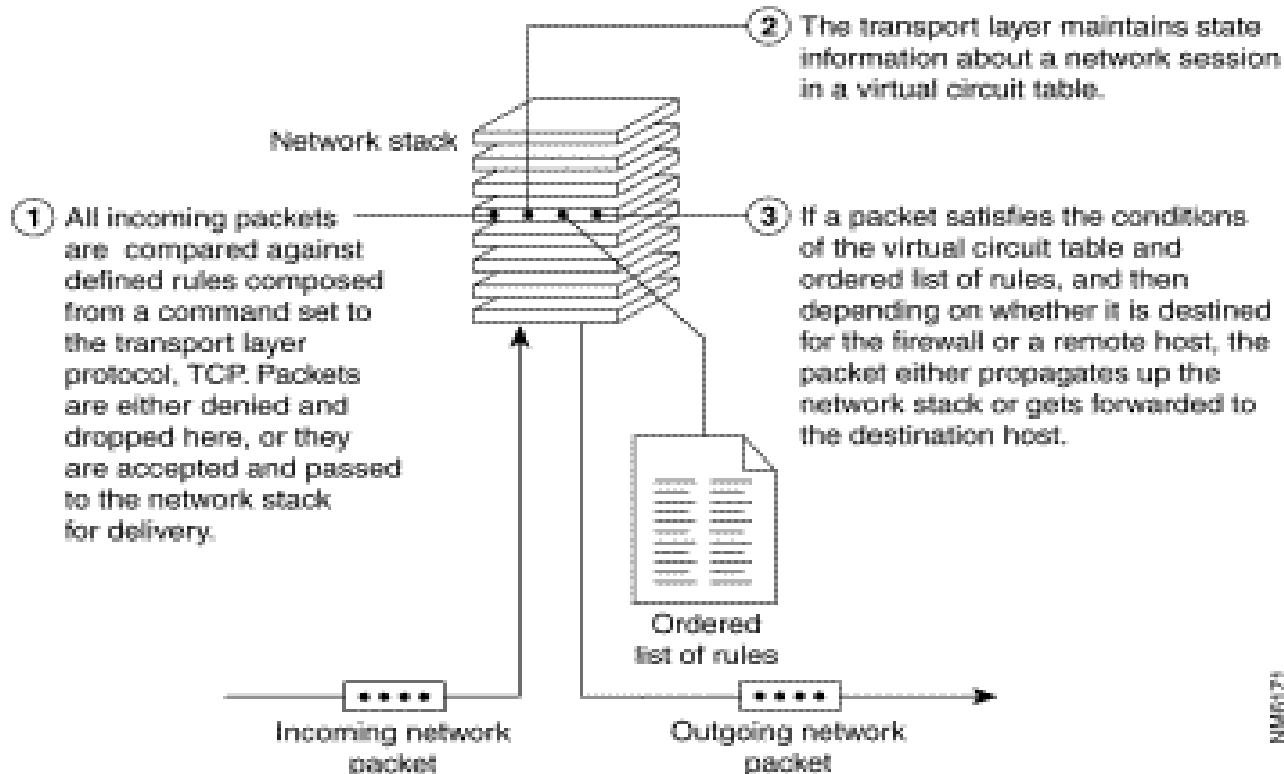
Variable use

- Ports >1024 must be available for client to make any connection
- This presents a limitation for stateless packet filtering
 - If client wants to use port 2048, firewall must allow *incoming* traffic on this port

Circuit level

Application space

Kernel space



Firewalls – Stateful Packet Filters

Traditional packet filters do not examine transport layer context

- ie matching return packets with outgoing flow

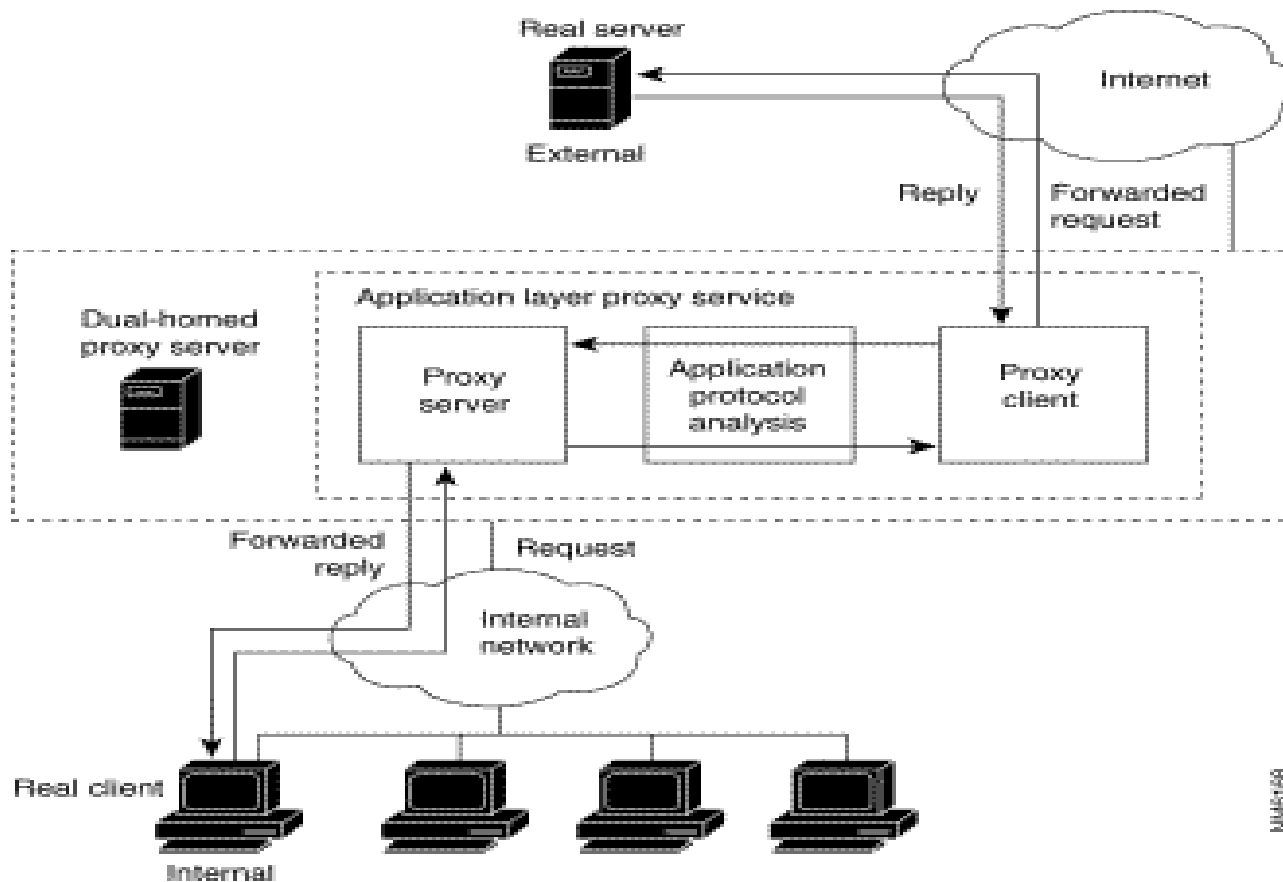
Stateful packet filters address this need

They examine each IP packet in context

- Keep track of client-server sessions
- Check each packet validly belongs to one

Hence are better able to detect bogus packets out of context

Proxy service



Firewall Gateways

Firewall runs set of proxy programs

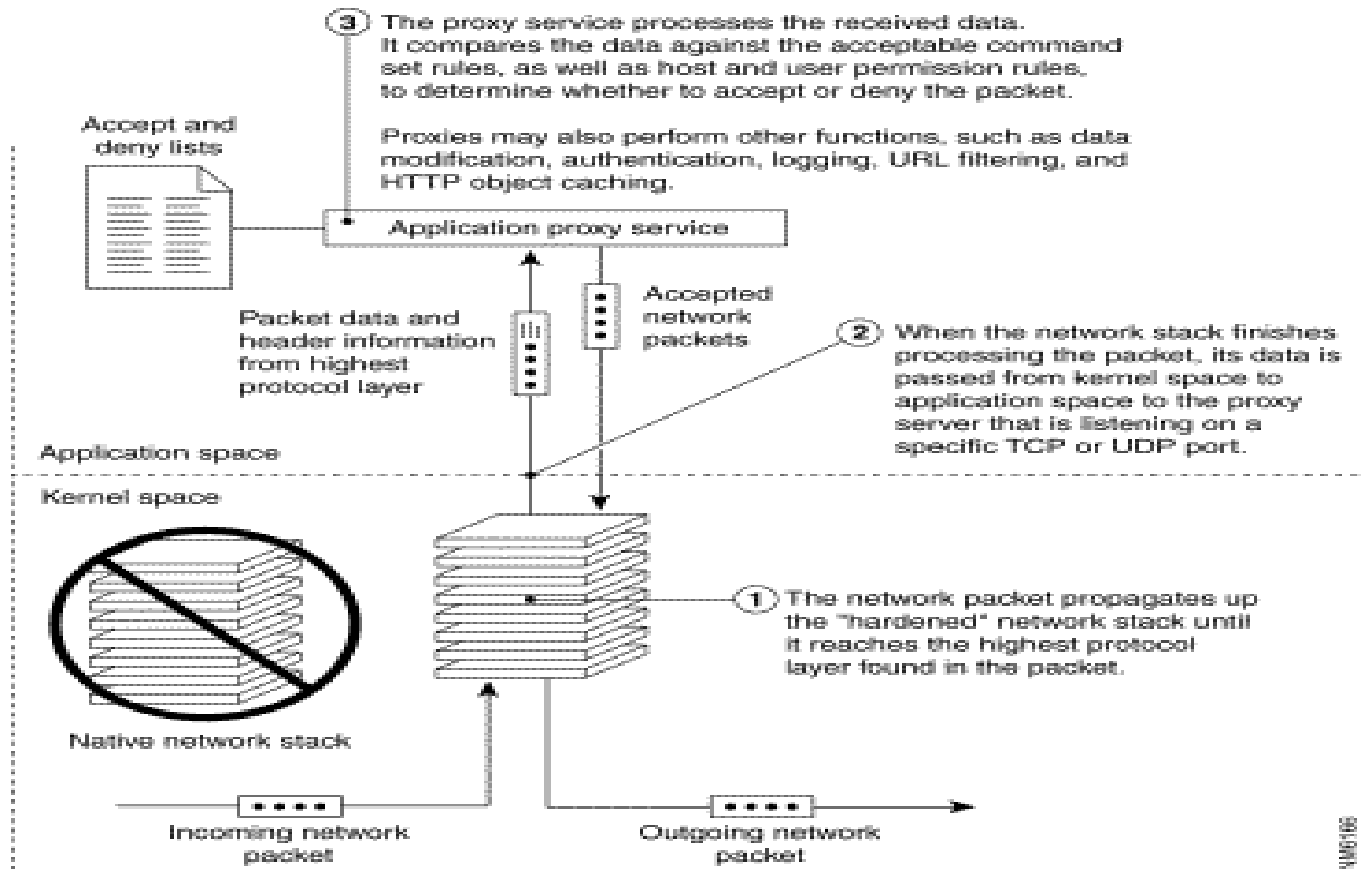
- Proxies filter incoming, outgoing packets
- All incoming traffic directed to firewall
- All outgoing traffic appears to come from firewall

Policy embedded in proxy programs

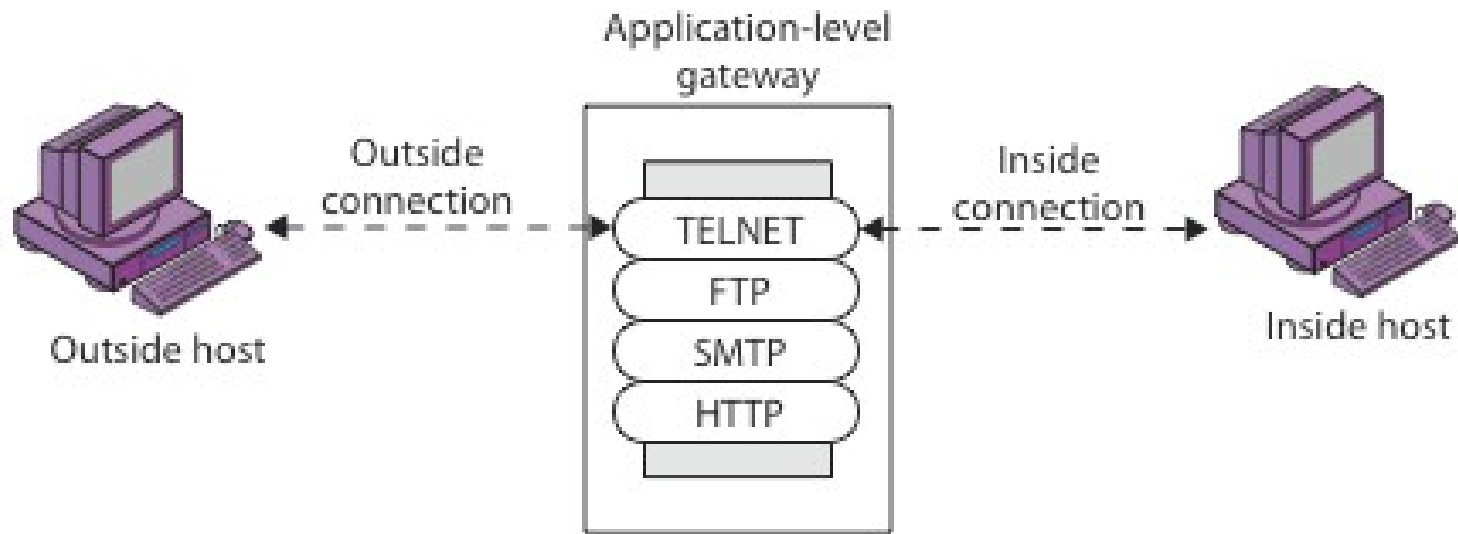
Two kinds of proxies

- Application-level gateways/proxies
 - Tailored to http, ftp, smtp, etc.
- Circuit-level gateways/proxies
 - Working on TCP level

Application Level



Firewalls - Application Level Gateway (or Proxy)



(b) Application-level gateway

Application-Level Filtering

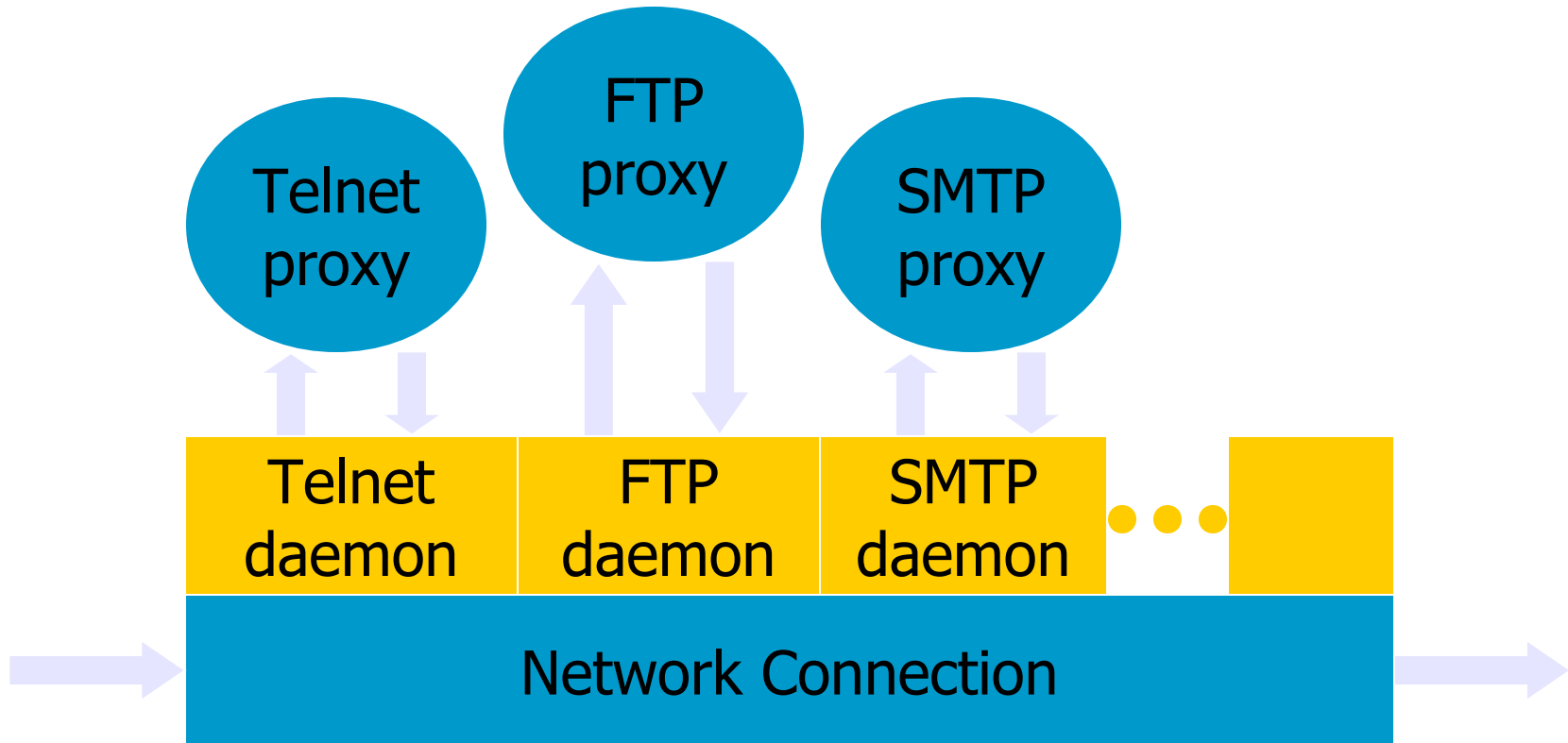
Has full access to protocol

- user requests service from proxy
- proxy validates request as legal
- then actions request and returns result to user

Need separate proxies for each service

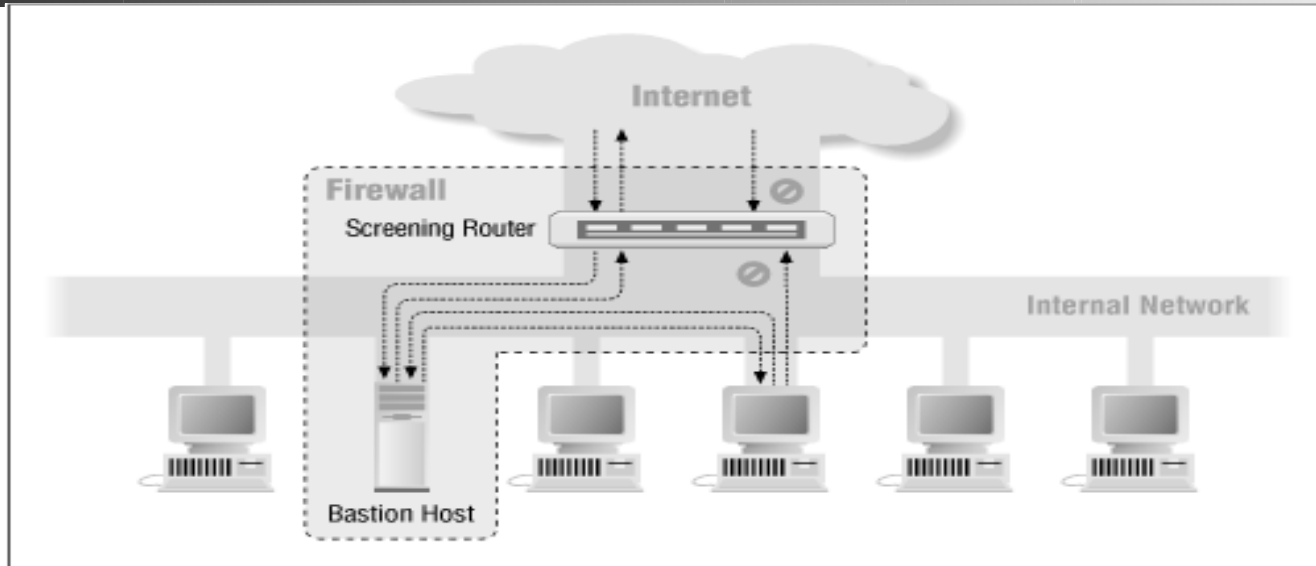
- E.g., SMTP (E-Mail)
- NNTP (Net news)
- DNS (Domain Name System)
- NTP (Network Time Protocol)
- custom services generally not supported

App-level Firewall Architecture



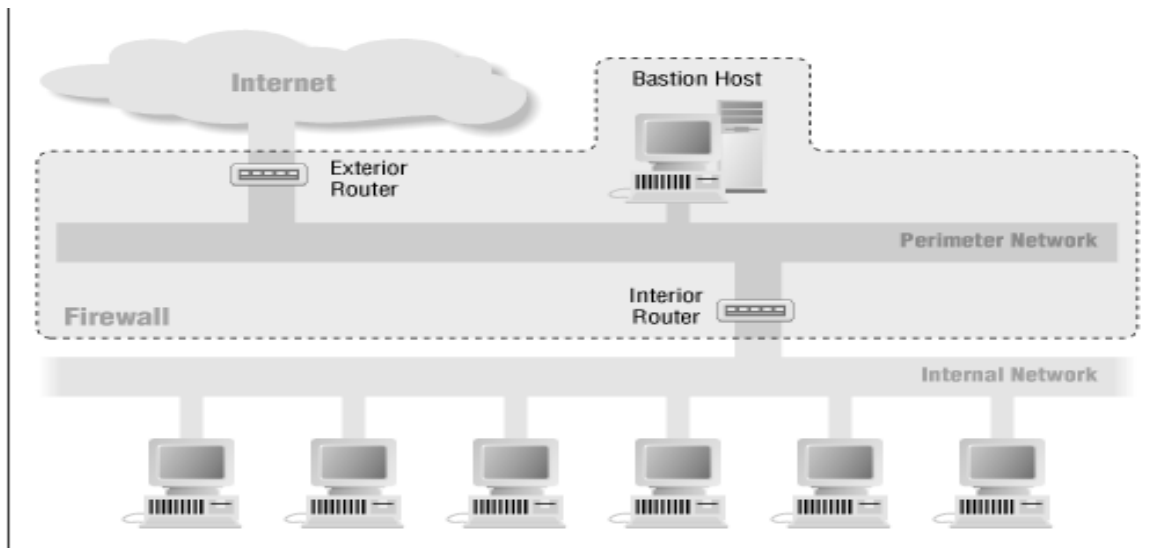
Daemon spawns proxy when communication detected

Screening router + bastion host



The bastion host is the only system on the internal network that hosts on the Internet can open connections to (for example, to deliver email). Only certain types of connections are allowed. Any external system trying to access internal systems or services will have to connect to this host. The bastion host thus needs to maintain a high level of host security.

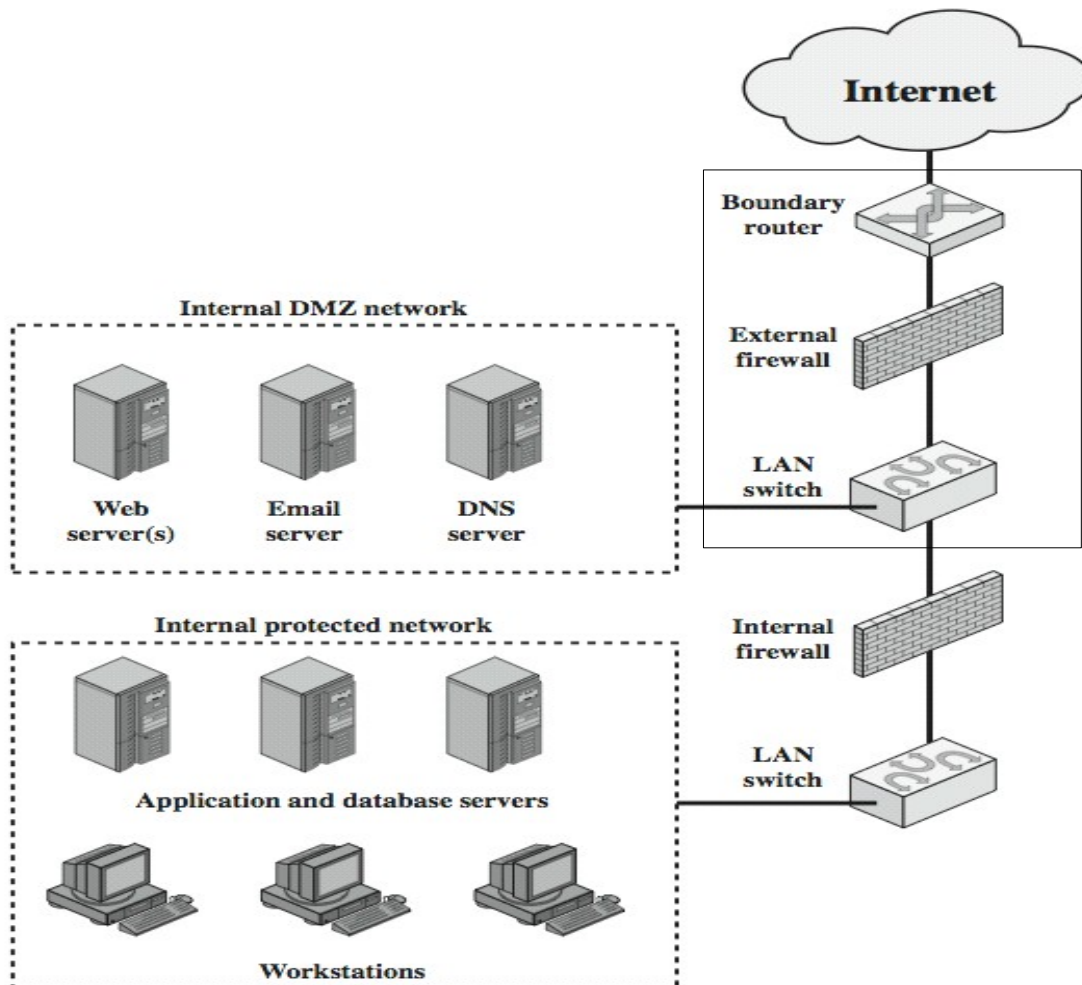
Screened subnet architecture



An extra layer of security by adding a perimeter network that further isolates the internal network from the Internet.

Bastion hosts are the most vulnerable machines they are the machines most likely to be attacked, because they're the machines that can be attacked.

DMZ – Layered protection = defence in depth



Even a simple router with packet filtering



DMZ – Advantages

- The creation of three layers of protection that segregate the protected network. To penetrate the protected network, the intruder must crack three separate routers:
 - the outside firewall router,
 - the bastion firewall
 - the inside firewall router devices.
- The outside router advertises the DMZ network only to the Internet systems on the Internet do not have routes to the protected private network. This allows the private network to be "invisible," and only selected systems on the DMZ are known to the Internet
- The inside router advertises the DMZ network only to the private network, systems on the private network do not have direct routes to the Internet.
- Since the DMZ network is a different network from the private one, a Network Address Translator (NAT) can be installed on the bastion host to eliminate the need to renumber or re-subnet the private network.

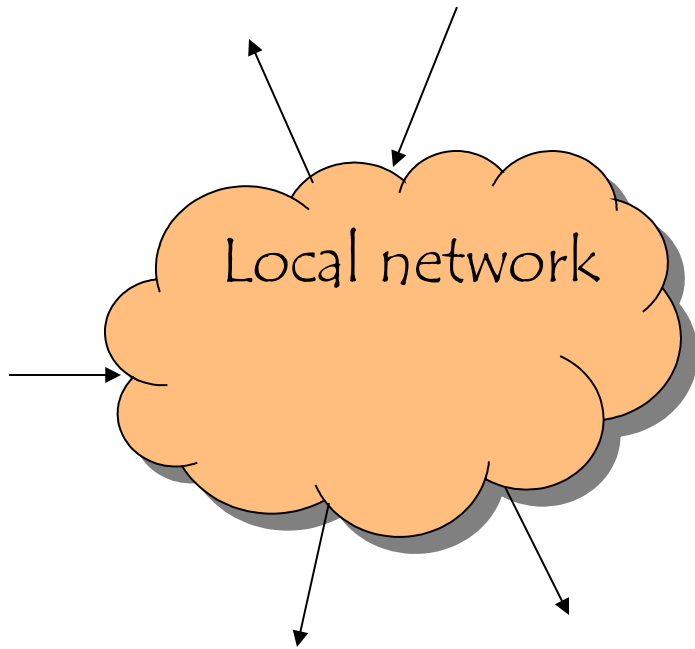


Countermeasures – Resist & Recovery

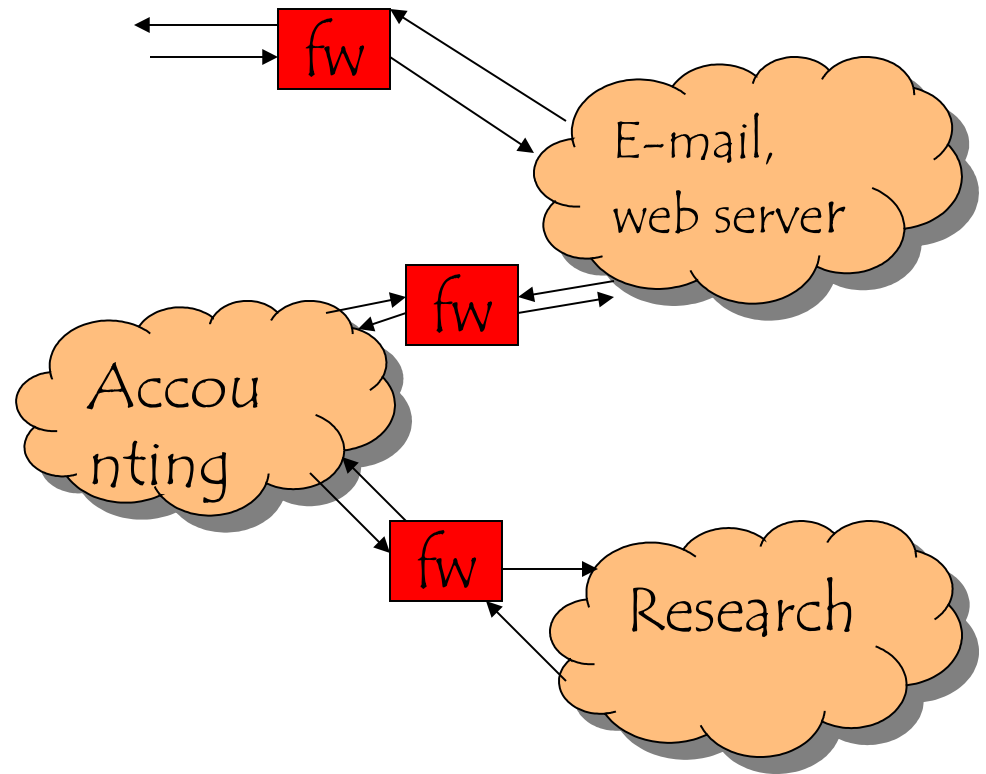
- Defence-in-depth
 - A network is segmented into several subnetworks, each with a security level
 - Networks with consecutive security levels only are connected
 - Any connection from a network to another one is protected by a firewall
 - Physical node connections may have to be updated



Defence-in-depth



Initial configuration



Defence in depth



Firewall & Virtual Machine

- Virtualization technology supports the definition of virtual network (overlay network)
- This makes it possible to spread information across a large number of nodes and of networks
- Virtual networks are protected by (virtual) firewall
- Some applications can be protected by mapping the corresponding virtual nodes onto distinct physical nodes
- 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

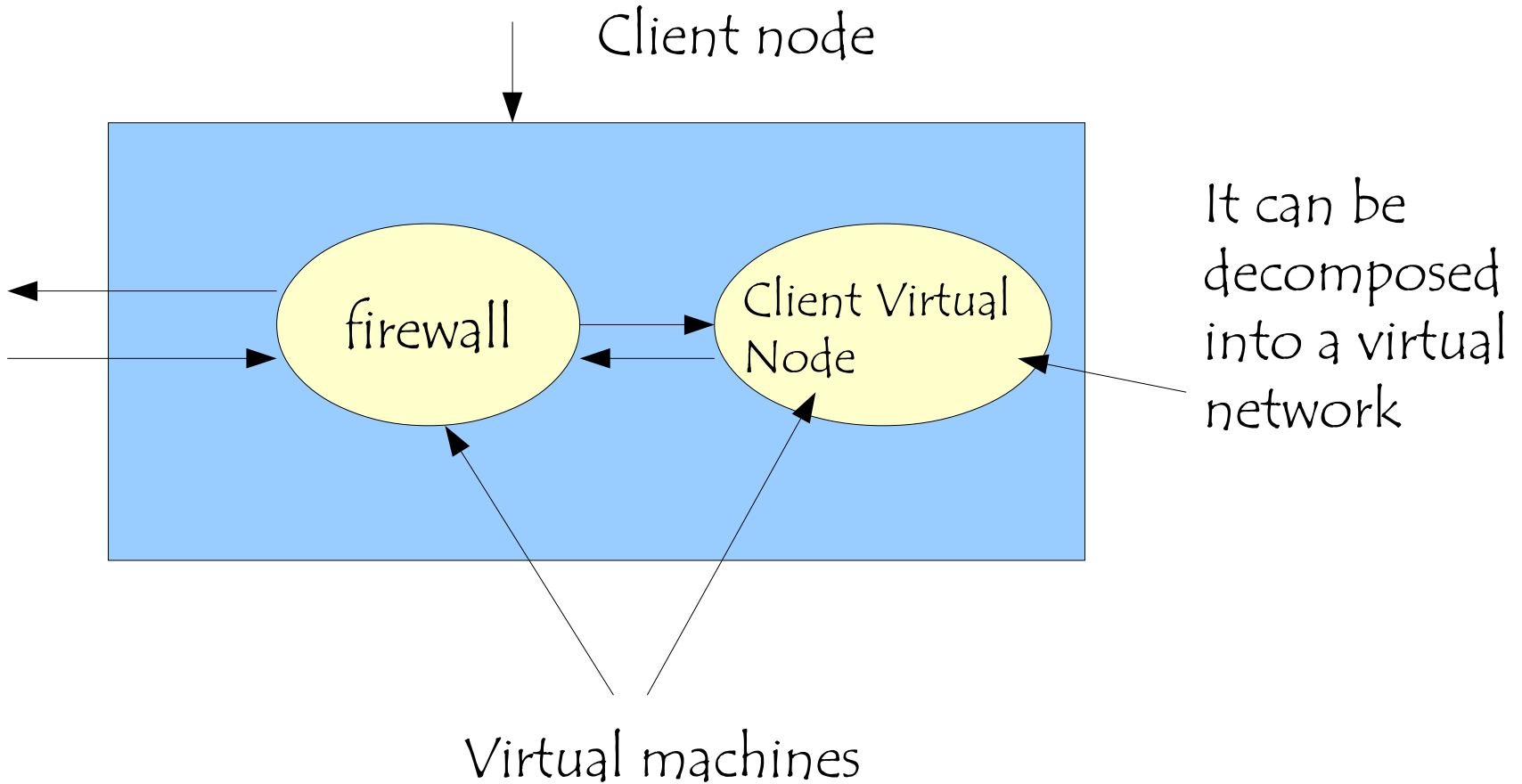
Checks are more rigorous as sharing may be minimized



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?

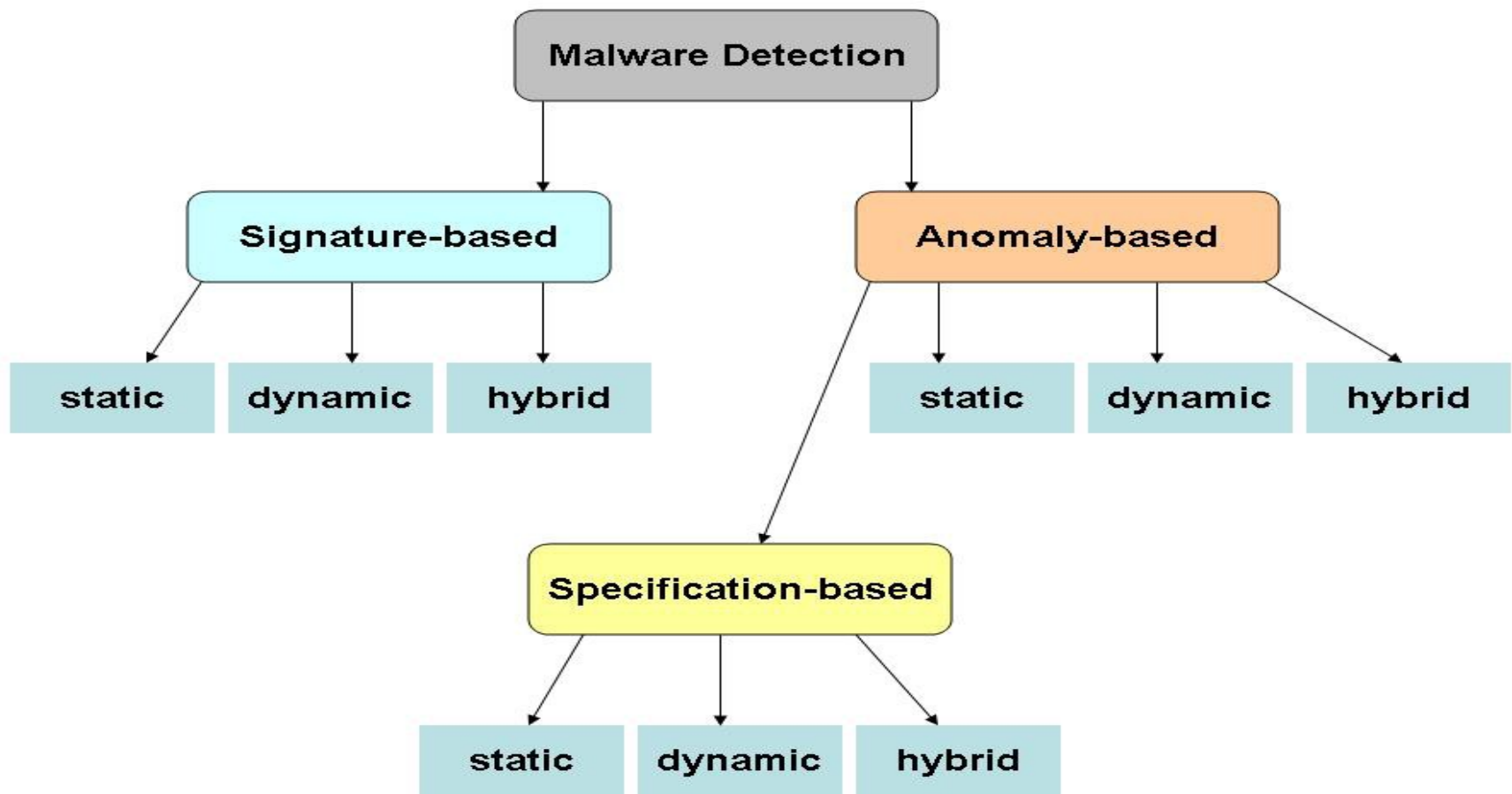




Countermeasure - Detect

- 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

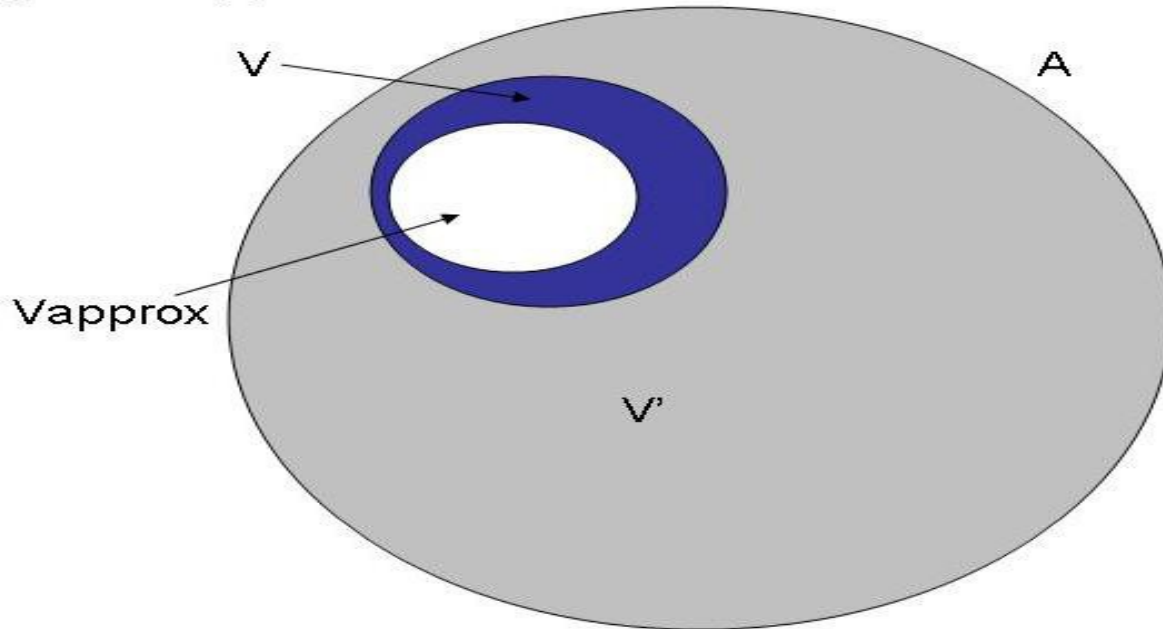


Detection – Anomaly Based

- 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
V_{approx} = approximation to V



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 discovered, 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 from 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 – Signature/Anomaly Based

Known-Known
Detect the exactly known infection, as seen before

Known-Unknown
Detect previously unseen variations of known threats, subfamilies or related new threats

Unknown-Unknown
Detect zero-days, unrelated to any known malware

Threat Type vs. Suitable Detection Technique

	Static Signatures	Dynamic Signatures	Behavioral Signatures	High-Level Patterns	Unsupervised Anomalies
Examples	Concrete malicious domain name associated to trojan server1.39slxu3bw.ru	Houdini RAT telemetry pattern regex: .*i[a-z]-(ready ri-noy gnfoh)	Two illustrative found instances hxxp://crazyerror.su/bl opt/8681BAE3DB3A2F9D446 CD5E3 hxxp://50.63.147.69:8080 /b/req/3D111E6B21F373015 C646CA4	Generic characteristics of suspicious traffic 	Expected vs. unexplained and unexpected 
What It Does	Manual definition, possibly tooling-assisted Exact matching of predefined character or numeric sequences Definitions human-readable	Manual definition, possibly tooling-assisted Matching of predefined rules (for example, regex) Definitions human-readable	Applicable through supervised machine-learning Matching of machine-learned rules or recognition of machine-learned behavioral patterns in transformed feature space	Task for semi-supervised machine learning Very high-level patterns, machine-learned to distinguish generic behavior	Unsupervised machine learning Cases significantly distant to all known normal behavior, where the model of known behavior is machine-learned Distance measures can be highly abstract
Technique Properties	Very high precision No generalization Recall limited to the exact same cases Good explainability Does not scale Requires manual definition <i>Not applicable to encrypted data without MITM</i>	Very high precision Generalization limited Recall limited to predefined pattern; finds variations explicitly covered by the pattern Good explainability Does not scale well Requires manual definition <i>Not applicable to encrypted data without MITM</i>	High precision Generalization based on similarity to known malware Ideal for finding previously unseen variations/subfamilies of known infections Good explainability but more complex Scales somewhat well Learned (semi)auto from data <i>Applicable to encrypted data without decryption</i>	Good precision Generalization based on common suspicious behaviors High recall, good chance to find true zero-days, at the cost of more false alarms Explainability limited Findings may be difficult to attribute to known infections Scales well Learned (semi)auto from data <i>Applicable to encrypted data without decryption</i>	Low precision Generalization based on unusual behaviors Best chance to find true zero-days; highest risk of false alarms Explainability difficult Findings may be difficult to attribute to known infections Scales well Learned auto from data <i>Applicable to encrypted data without decryption</i>

Better Precision and Explainability, Simplicity of Proof

Better Recall, Scalability, Applicability to Encrypted Data, Ability to Detect Zero-Days

Technique Trade-Off

Please note: scaling statements refer to human time required to maintain detection system

Please note: this diagram represents a simplified illustration of machine learning capabilities in security



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...

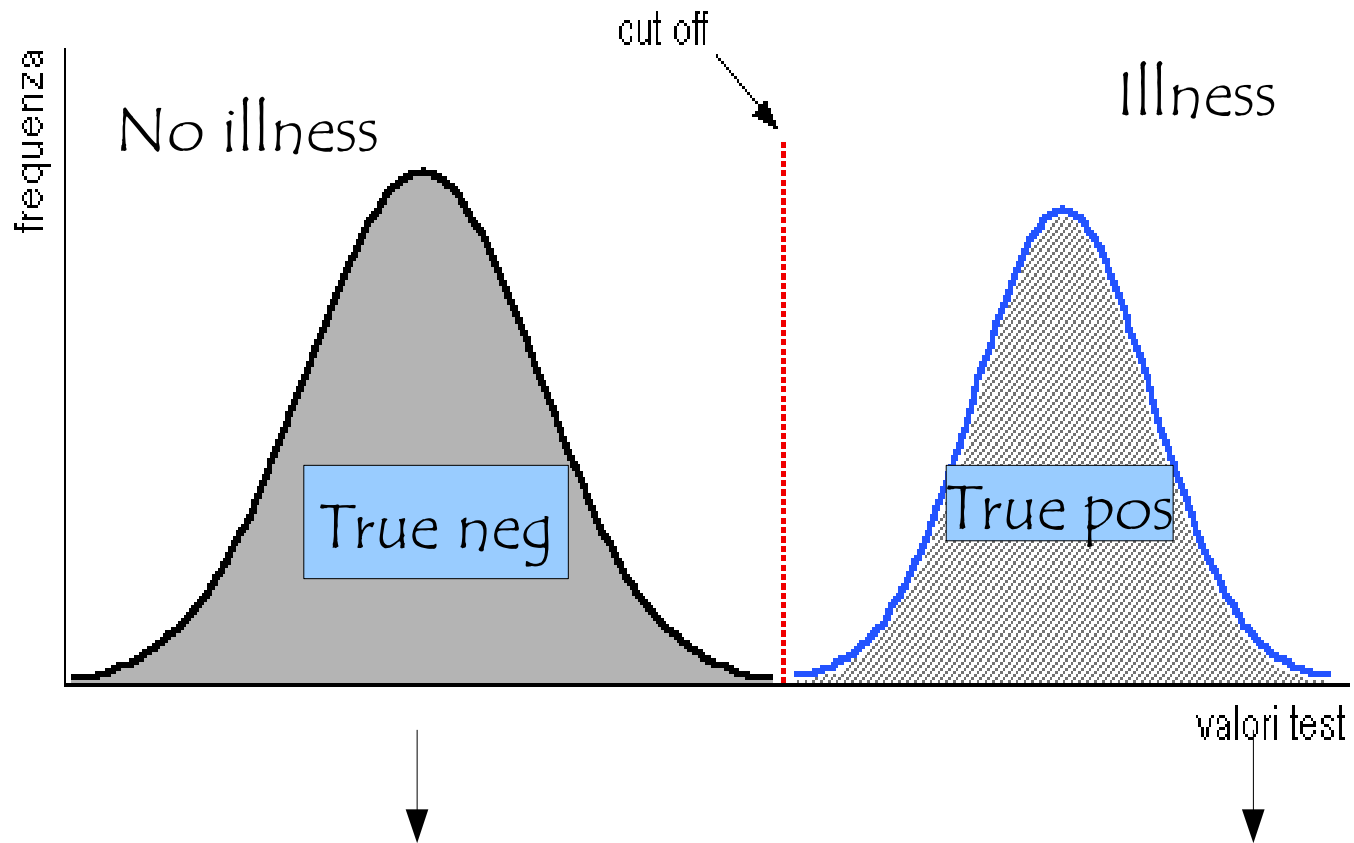
- The behavior that the tool detects are an approximation of those of interest. This implies that some statistic notion may be very useful
- The problem arises because we do not have a perfect test to discover if a system is being or has been attacked
- There is a set of symptoms (behavior) that suggest 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

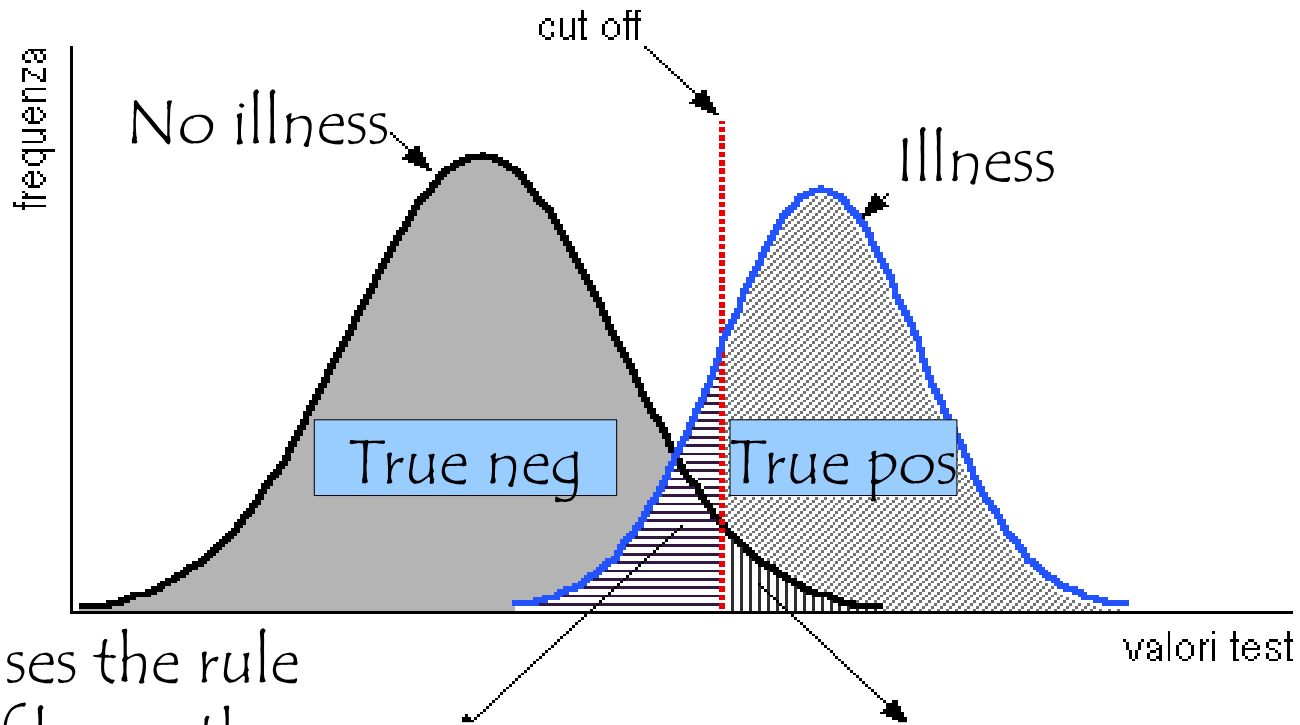
The ideal test



Probability distribution
of the parameter, no illness

Probability distribution
of the parameter, illness

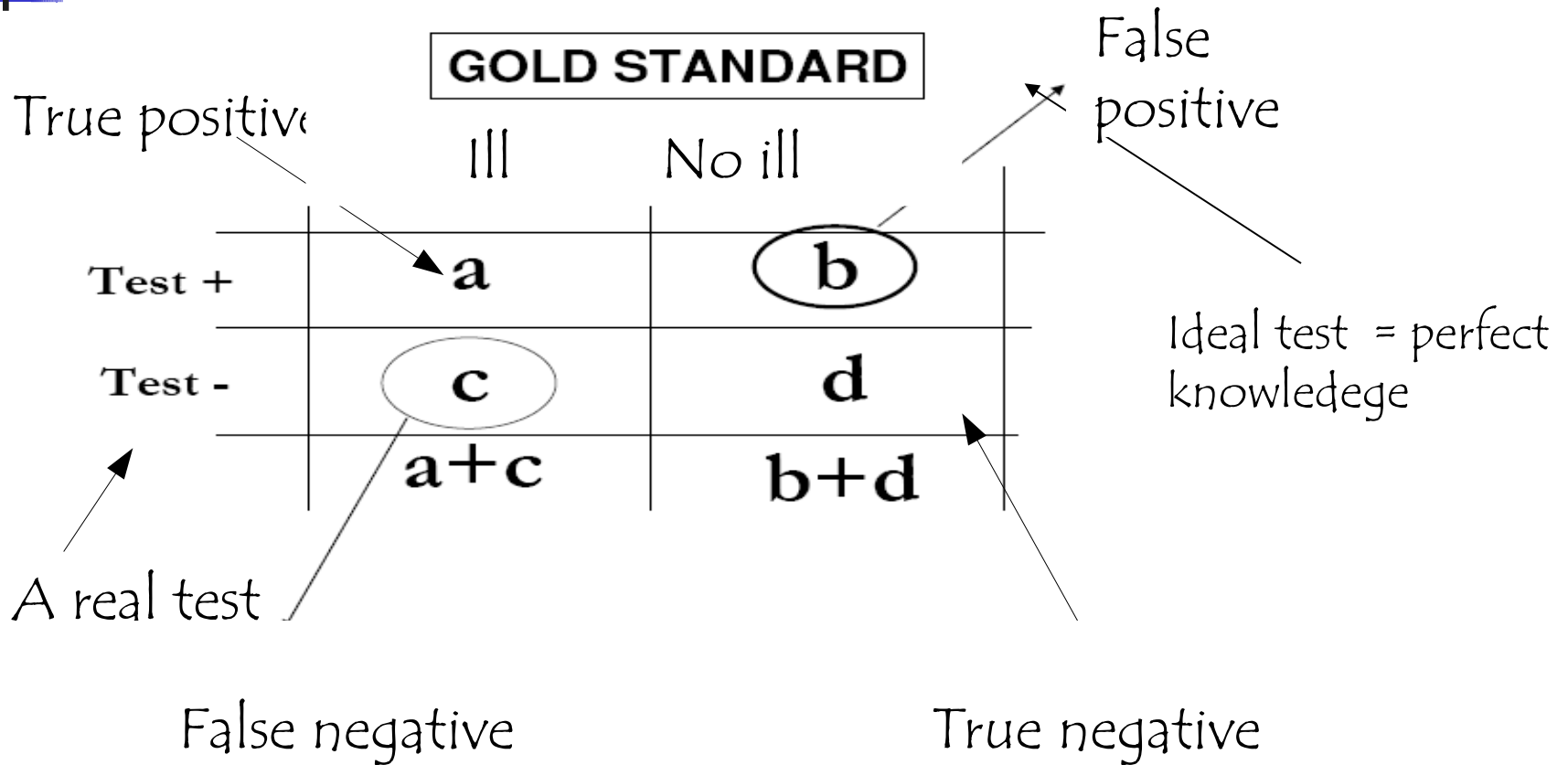
Any real test



Detection uses the rule
"no illness if lower than
threshold"

False negative false positive

A real test



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

	ill
Test +	a
Test -	c
	a+c

$$\text{Sen} = \text{pr}(T^+ | M^+)$$

$$\text{Sen} = a / (a + c)$$



Specificity

Specificity = probability of a false answer if no illness

	No ill
Test +	b
Test -	d
	b+d

$$Spe = pr(T^- | M^-)$$

$$Spe = d / (b + d)$$



Likelihood

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

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

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

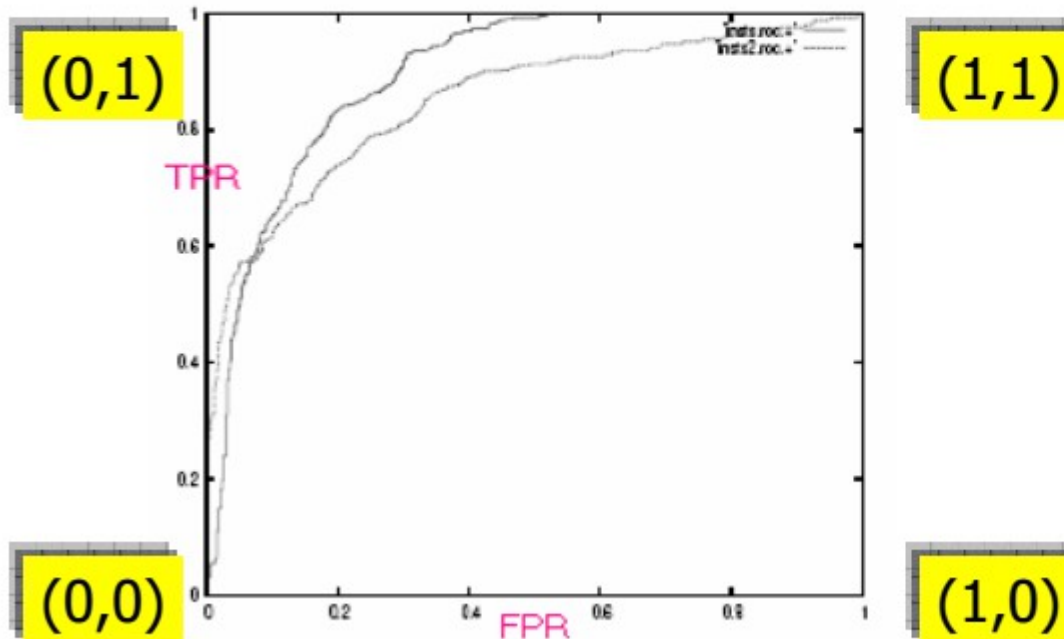
ROC curve

receiver operating characteristics

- Y axis: TPR = sensitivity
- X axis: FPR = 1-specificity

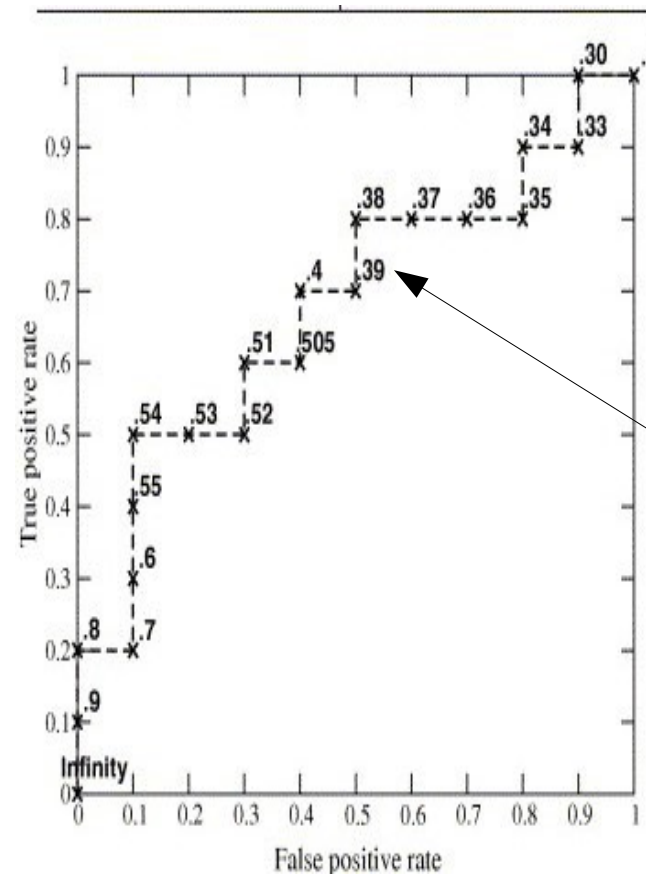
Benefits (TP)

Costs (FP)



Drawing a curve

Inst#	Class	Score	Inst#	Class	Score
1	p	.9	11	p	.4
2	p	.8	12	n	.39
3	n	.7	13	p	.38
4	p	.6	14	n	.37
5	p	.55	15	n	.36
6	p	.54	16	n	.35
7	n	.53	17	p	.34
8	n	.52	18	n	.33
9	p	.51	19	p	.30
10	n	.505	20	n	.1



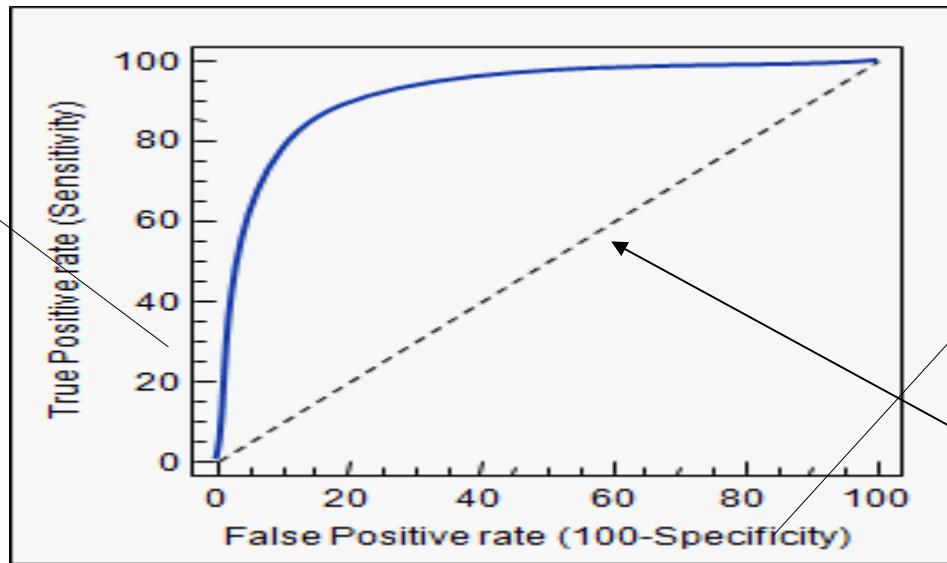
th chosen

if score < th then n else p

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 x for distinct values of the parameter (it opens x connections in a second)

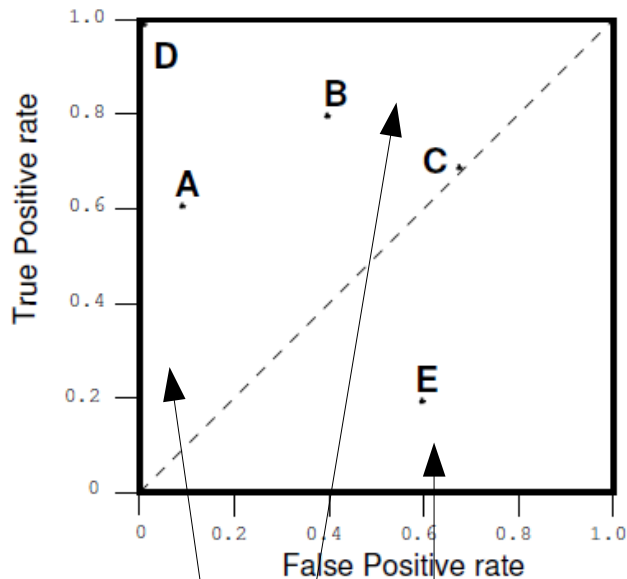
Each value of x results in a percentage of false and true positives

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

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 positives and few true positive

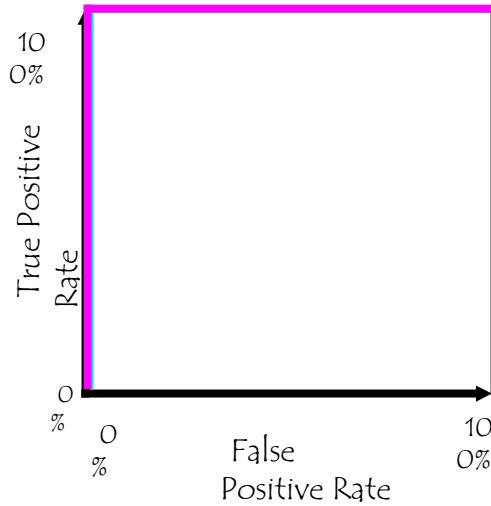
A rule under the bisector = worse than random (= the bisector) it can be improved by negating it

Area under ROC curve (AUC)

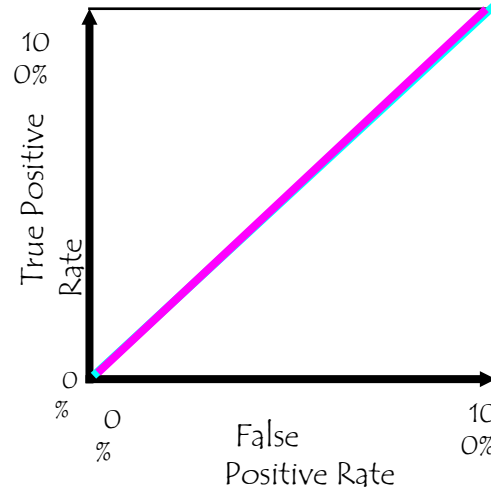
- *Overall measure* of test performance
- *Comparisons* between two tests based on differences between (estimated) AUC
- AUC can be interpreted as the probability that the test result from a randomly chosen diseased individual is more indicative of disease than that from a randomly chosen nondiseased individual:
$$P(X_i \geq X_j \mid D_i = 1, D_j = 0)$$
- AUC evaluates the features we have chosen to define our test. Distinct features result in distinct curves

AUC for ROC curves

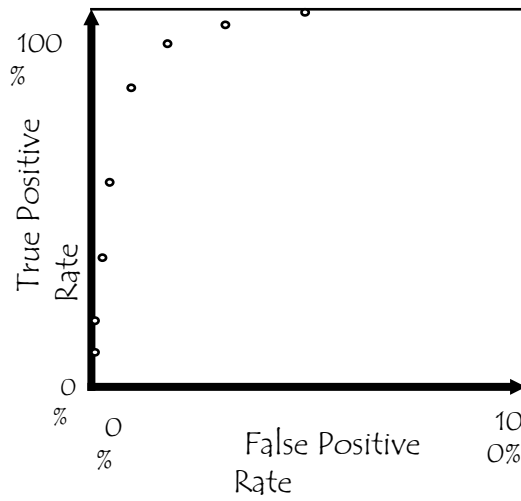
AUC = 100%



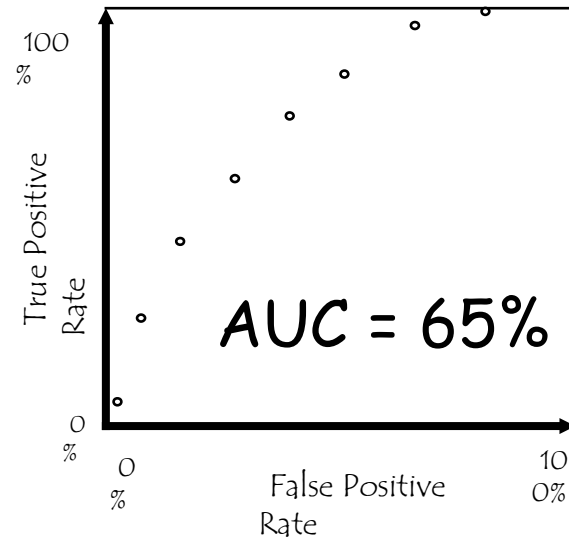
AUC = 50%



AUC = 90%

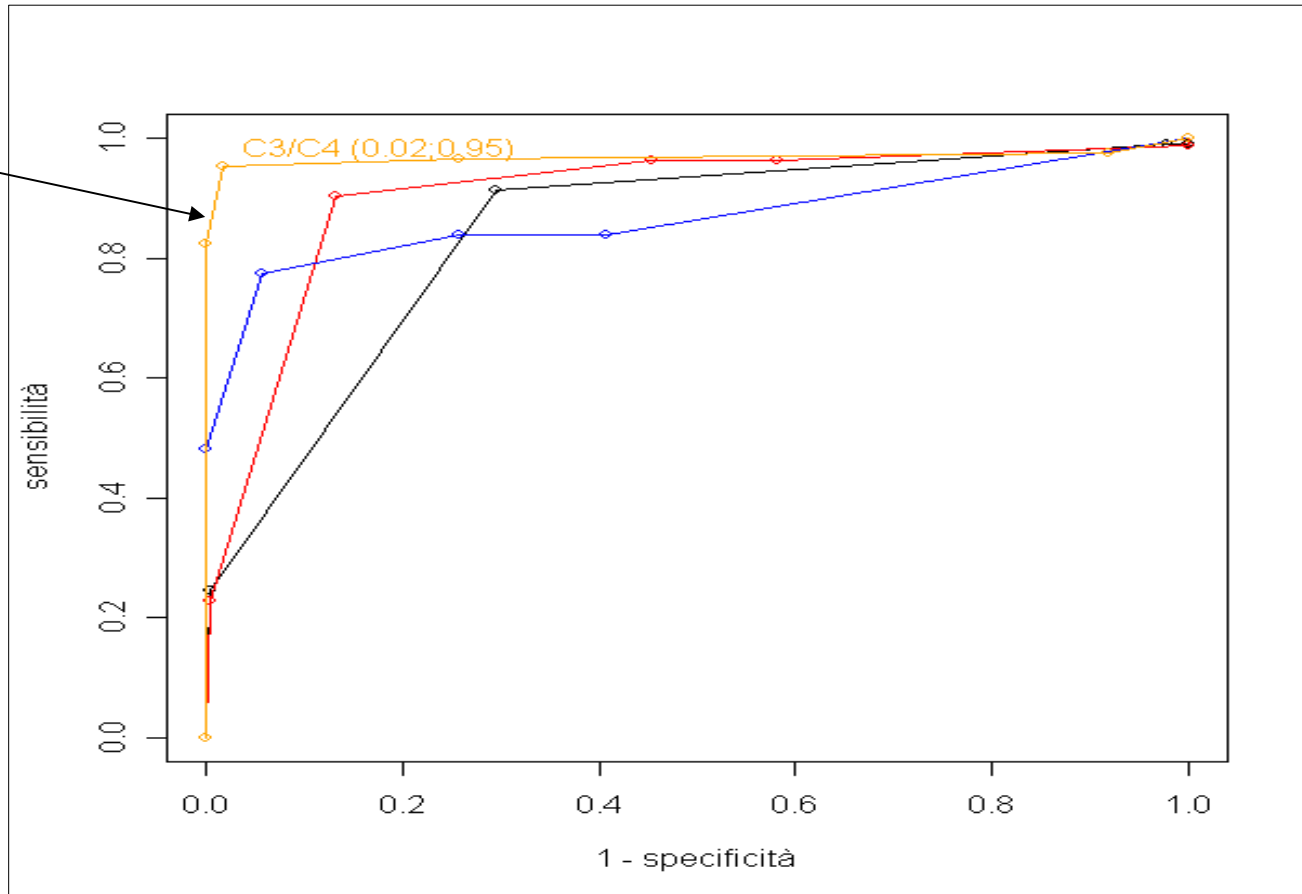


AUC = 65%



Applying ROC (AUC) to select a strategy

Best solution
Always higher
Than the others





Problems with AUC

- *No clear and rigorous semantic interpretation*
- A lot of the area is coming from the range of *large false positive* values, no one cares what's going on in that region (need to examine restricted regions)
- The curves might *cross*, so that there might be a meaningful difference in performance that is not picked up by AUC



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

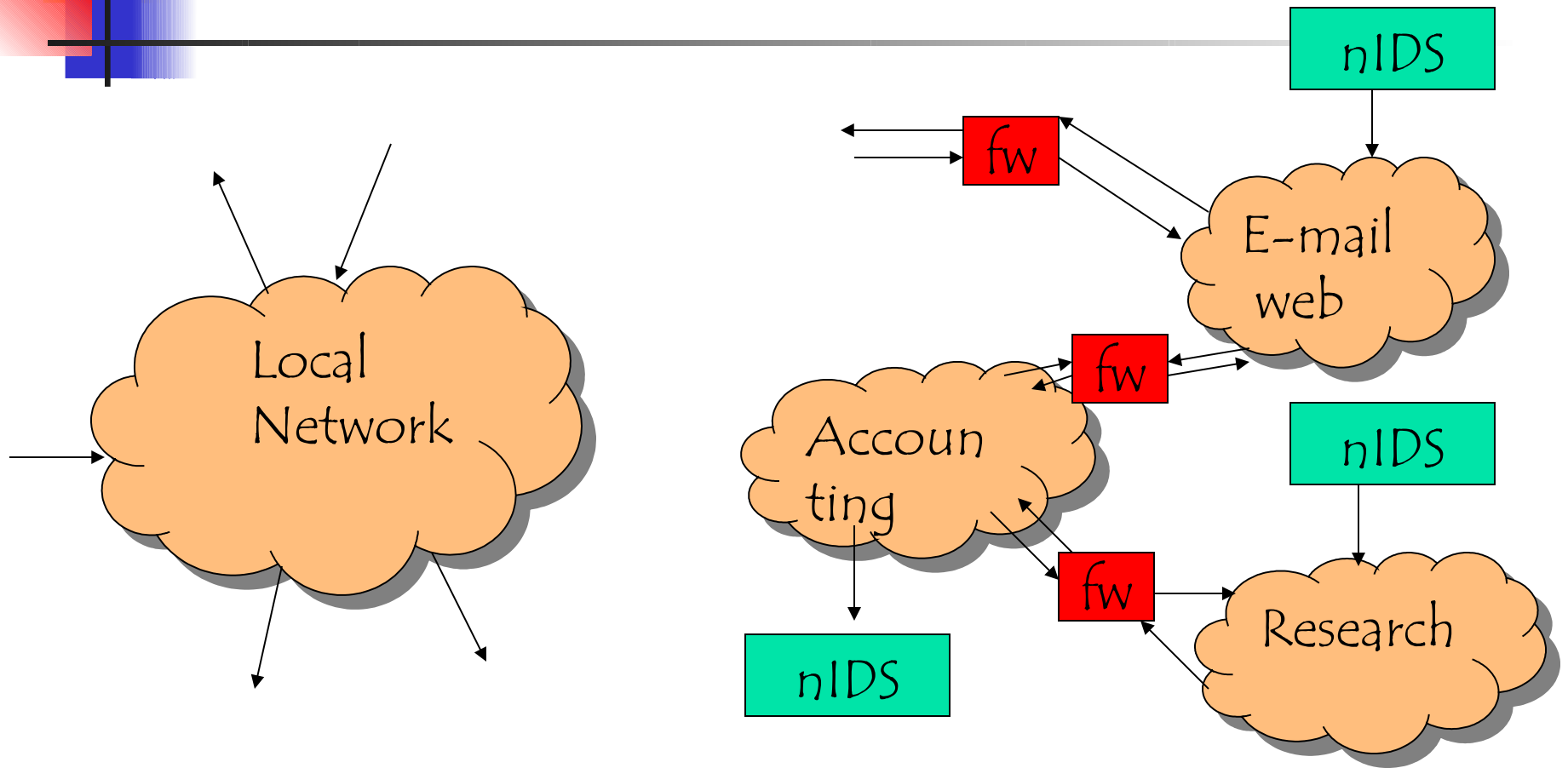
- In any case, the adoption of an IDS has **to be transparent** for the user
- In several cases, the users should not to be aware that an IDS has been adopted (it can discover insider threats)
- Legal problems
 - According to the italian law the adoption of any tool that can be used to monitor a worker 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

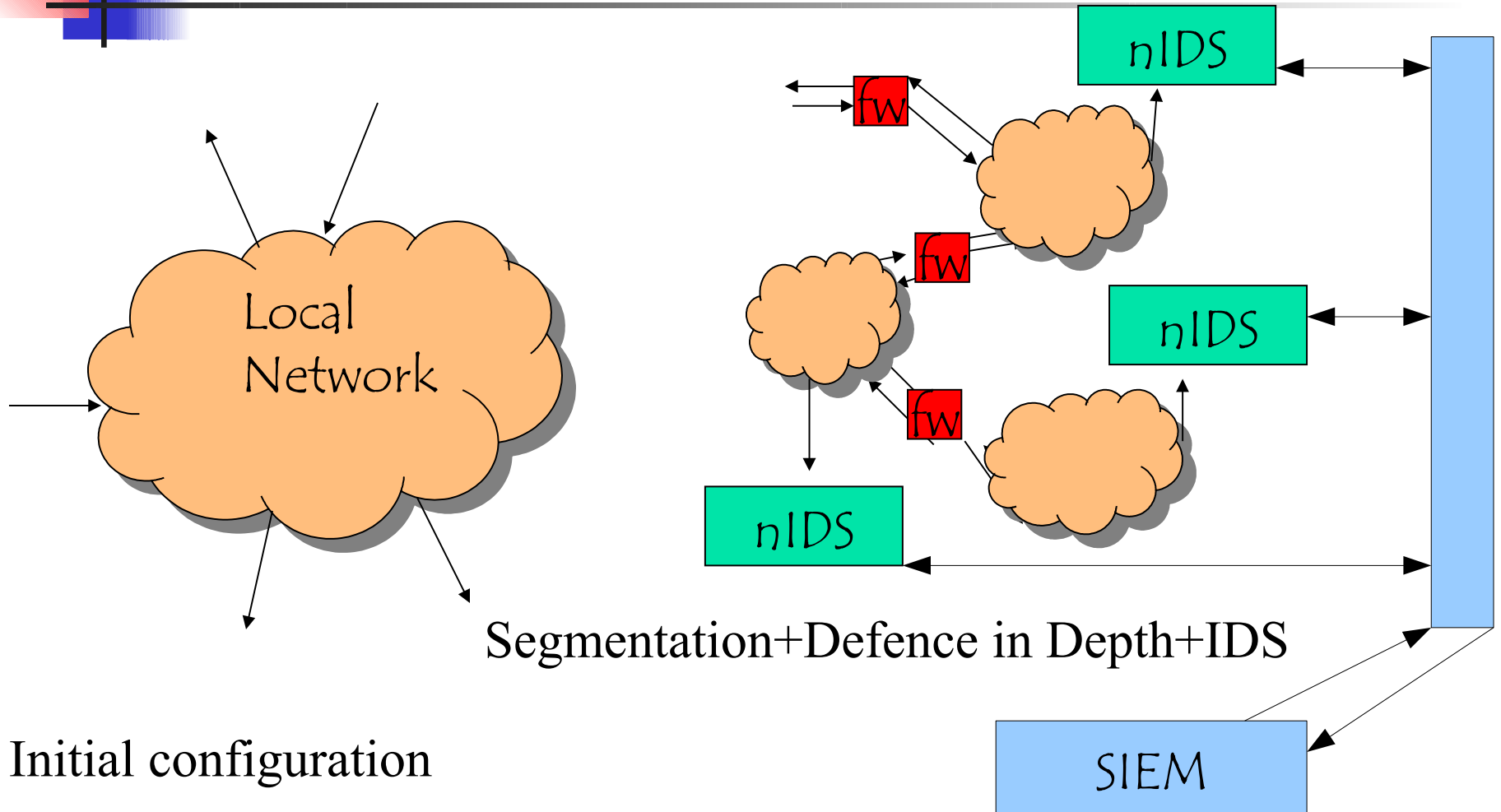
Intrusion Detection System



Initial configuration

Segmentation+Defence in Depth+IDS

Security information and event management = SIEM



Initial configuration

Segmentation+Defence in Depth+IDS

SIEM



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 successful
- 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
- ⇒ NIDS has to be executed on a dedicated host to analyze all the information flows



hIDS and nIDS technologies

- Base element that is analyzed
 - IP packets 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 analysis
- Normalization of a protocol to handle all those cases that are not defined by a standard (overlapping IP packets)



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



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

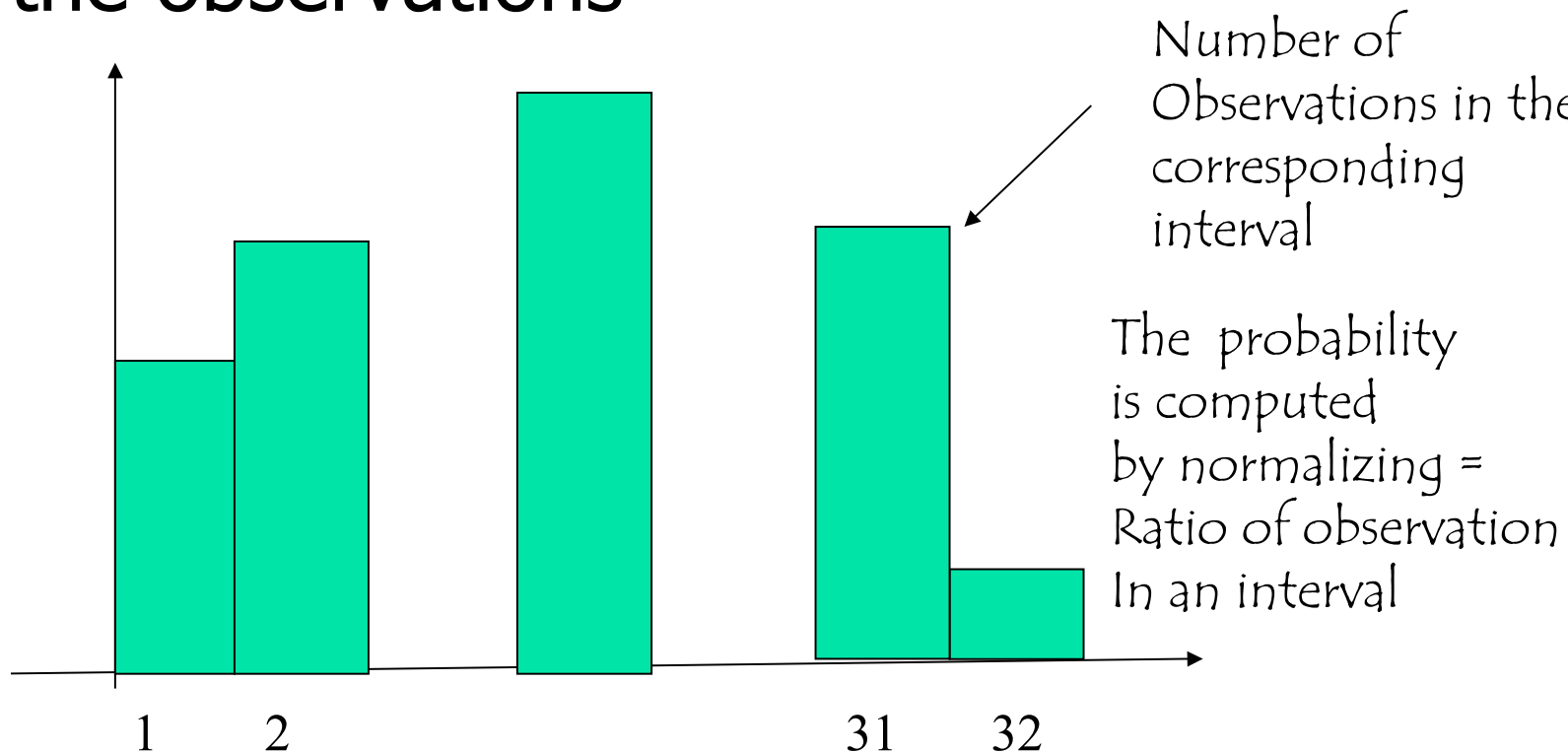


N&H-IDS: anomaly detection

- 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





N&H-IDS: anomaly detection

- The difference between two discrete distributions is the sum of the absolute differences between two corresponding intervals
- $\text{Dist} = \sum |long_i - short_i|$
- Several distributions of the same measures can be generated by distinct observation frequency or for distinct cases
 - Open files
 - Read the number at each minute or at each hour
 - Read the number for each user or group of users



N&H-IDS: anomaly detection

- The IDS raises an alarm 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 weighed 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 measures correspond to distinct ROC curves

Anomaly detection: an example



- Nides = next generation intrusion detection system
- To protect military systems
- First rigorous definitions of 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

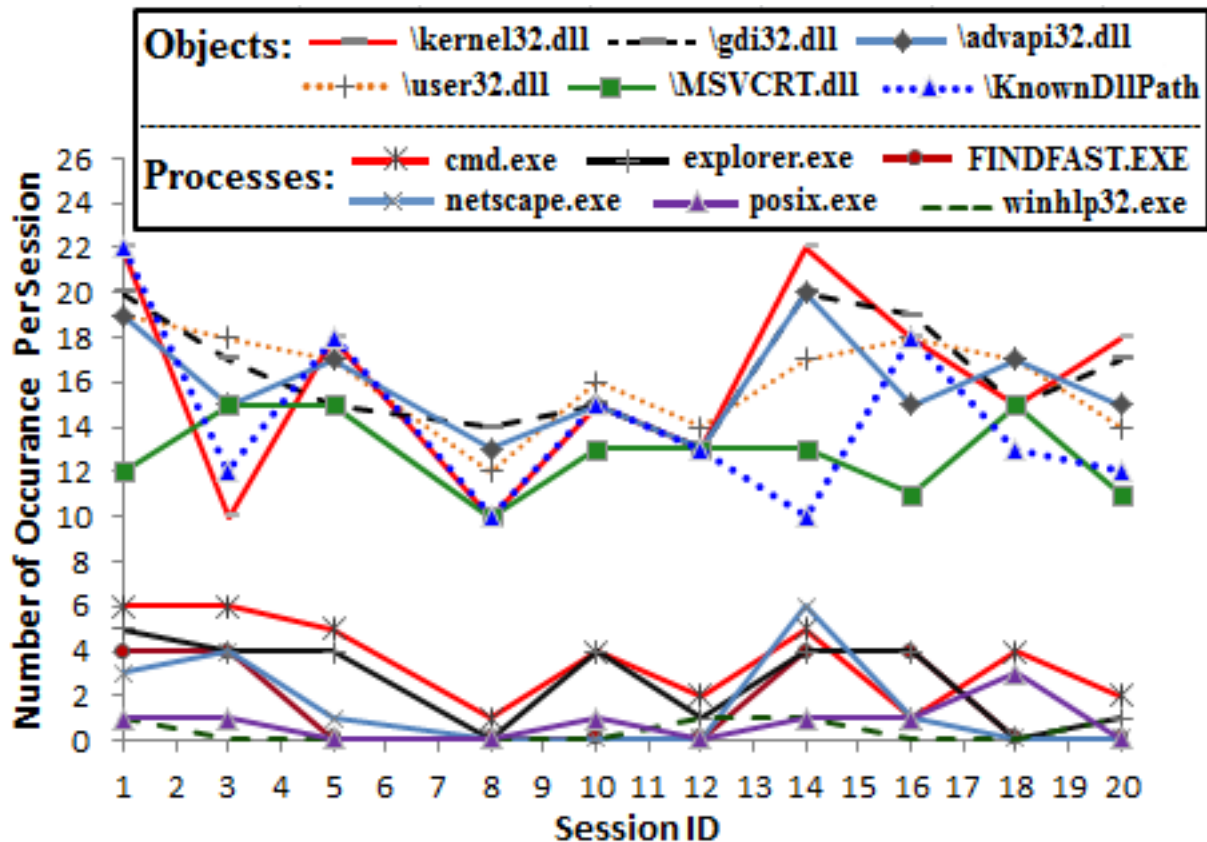


NIDES – Learning time - II

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

Detecting Masqueraders in Clouds based on Security Events and NetFlow Data Analysis

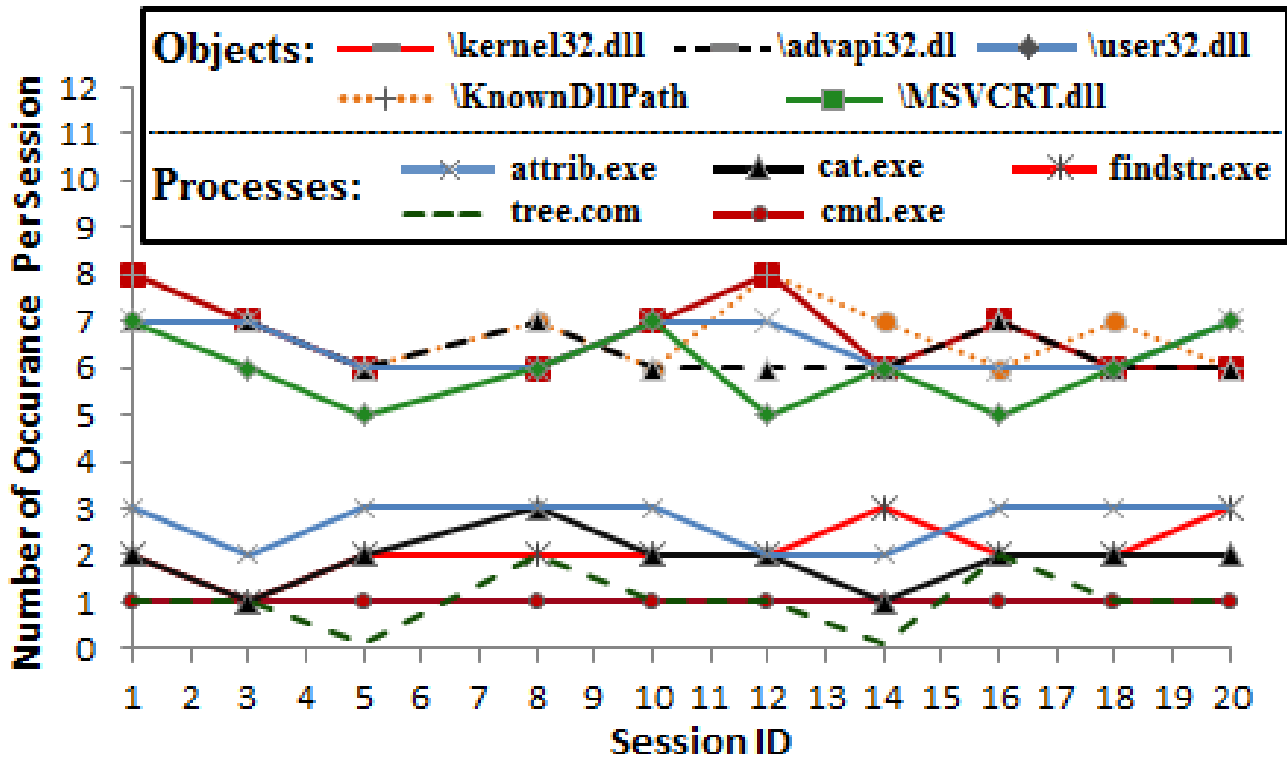
Hisham A. Kholidy, Fabrizio Baiardi, and Salim Hariri



A real user

Detecting Masqueraders in Clouds based on Security Events and NetFlow Data Analysis

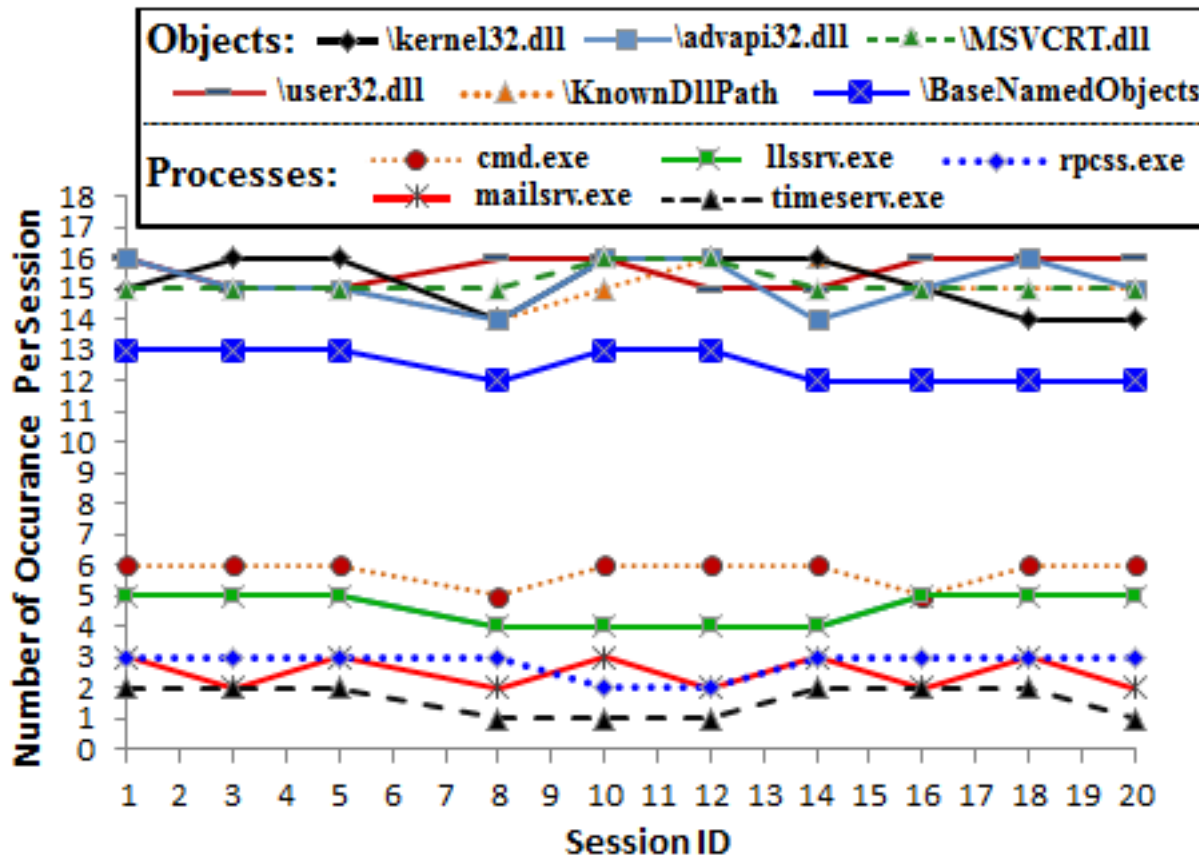
Hisham A. Kholidy, Fabrizio Baiardi, and Salim Hariri



A server

Detecting Masqueraders in Clouds based on Security Events and NetFlow Data Analysis

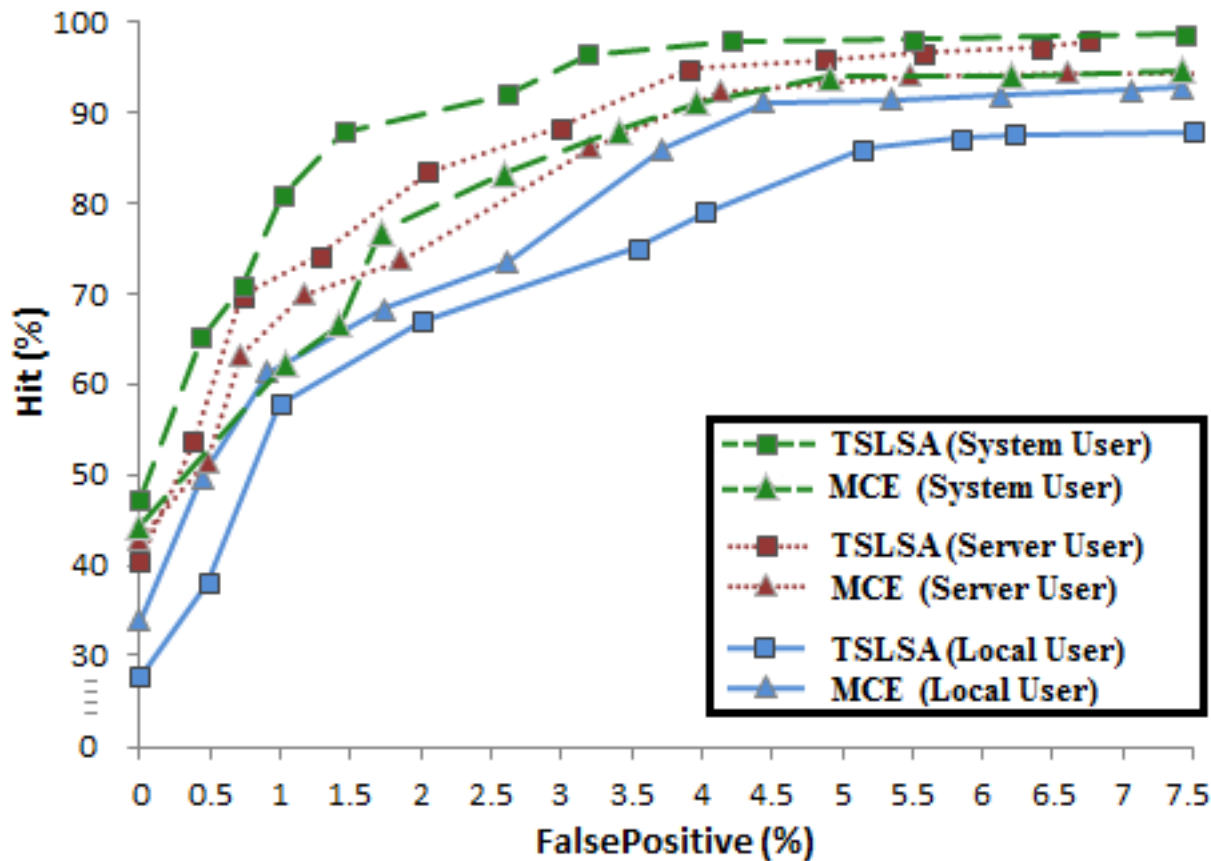
Hisham A. Kholidy, Fabrizio Baiardi, and Salim Hariri



An OS process

Detecting Masqueraders in Clouds based on Security Events and NetFlow Data Analysis

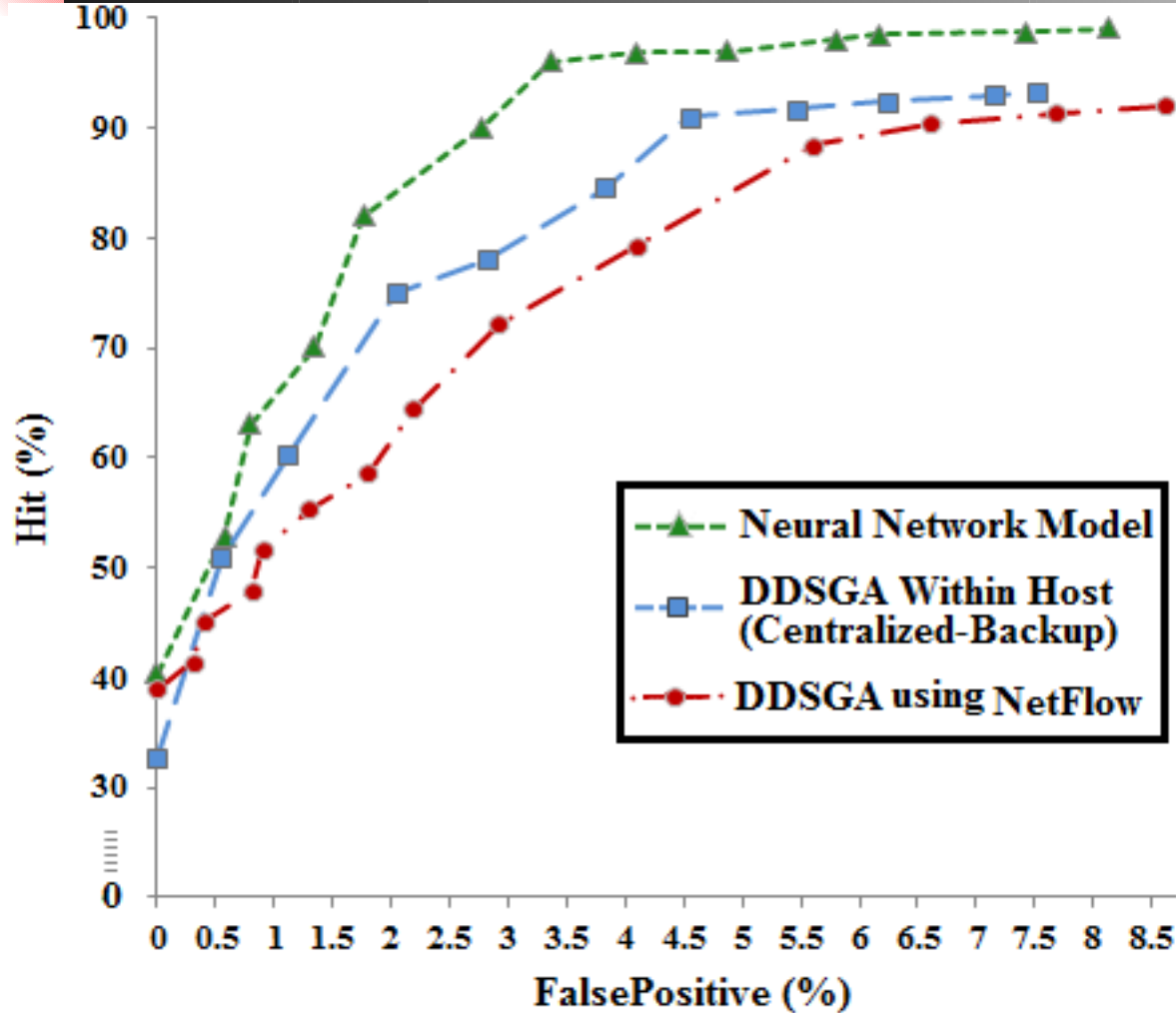
Hisham A. Kholidy, Fabrizio Baiardi, and Salim Hariri



ROC curves

Detecting Masqueraders in Clouds based on Security Events and NetFlow Data Analysis

Hisham A. Kholidy, Fabrizio Baiardi, and Salim Hariri



ROC curves
For local + networks
events



N&H-IDS: signature detection

- The overall behavior strongly resembles an antivirus tool
- A pattern database (signature) for known attacks, each action is matched against each pattern
- Currently the pattern may be stored in a server in a cloud actions are checked there
- Any matching is recorded
- Anytime a pattern has been fully matched, an alarm is fired



N&H-IDS: signature detection

- Describe an attack against a system where the IDS stores its signature database in a cloud
- List some countermeasures

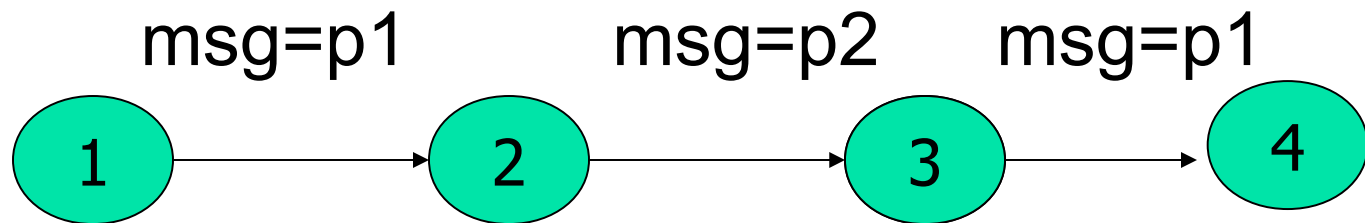


N&H-IDS: signature detection

- Wrt to Antivirus some differences:
 - Dynamic generation of the elements to be matched
 - The time inbetween two consecutive generations is unknown
 - An element can match several patterns
- The complexity is much larger 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



- If the recognizer is currently in state 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



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
```



N&H-IDS:

signature detection & evasion

- When sniffing a packet P the NIDS has no means to anticipate
 - Whether P will be received
 - How P will be handled
- An attacker can inject 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

Bypassing NIDS - Fragmentation



- NIDS must reconstruct fragments
 - Maintain state = drain on resources
 - Must overwrite correctly = more drain on resources
- Target server correctly de-frags
- Attack #1 - just fragment
- Attack #2 - frag with overwrite
- Attack #3 - start an attack, follow with many false attacks, finish the first attack



Bypassing NIDS - TCP un-sync

- Inject a packet with a bad TCP checksum
 - fake 'FIN' packet
- Inject a packet with a weird TCP sequence number
 - step up
 - wrapping numbers



Bypassing HIDS - Stack Protection

- Stackguard
 - A 'canary' is placed next to return address
 - Program halts and logs if canary is altered
 - Canary can be random or terminating
 - Bypass: overwrite return address without touching canary
 - Fix: XOR the return address and the canary
 - Yet another example of an arms race



NIDS - Overwhelming

- Send as many false attacks as possible while still doing the real attack
 - May overload console
 - May drop packets
 - Admins may not believe there is a threat
- Send packets that “cost” the NIDS CPU cycles to process
 - Fragmented, overlapping, de-synchronized web attacks with the occasional bad checksum



NIDS - 'Slow Roll'

- Port scans and sweeps
 - Obvious: incremental destination ports
 - Trivial: randomized ports
 - Sweep: one port and many addresses
 - Stealthy: random ports and addresses over time

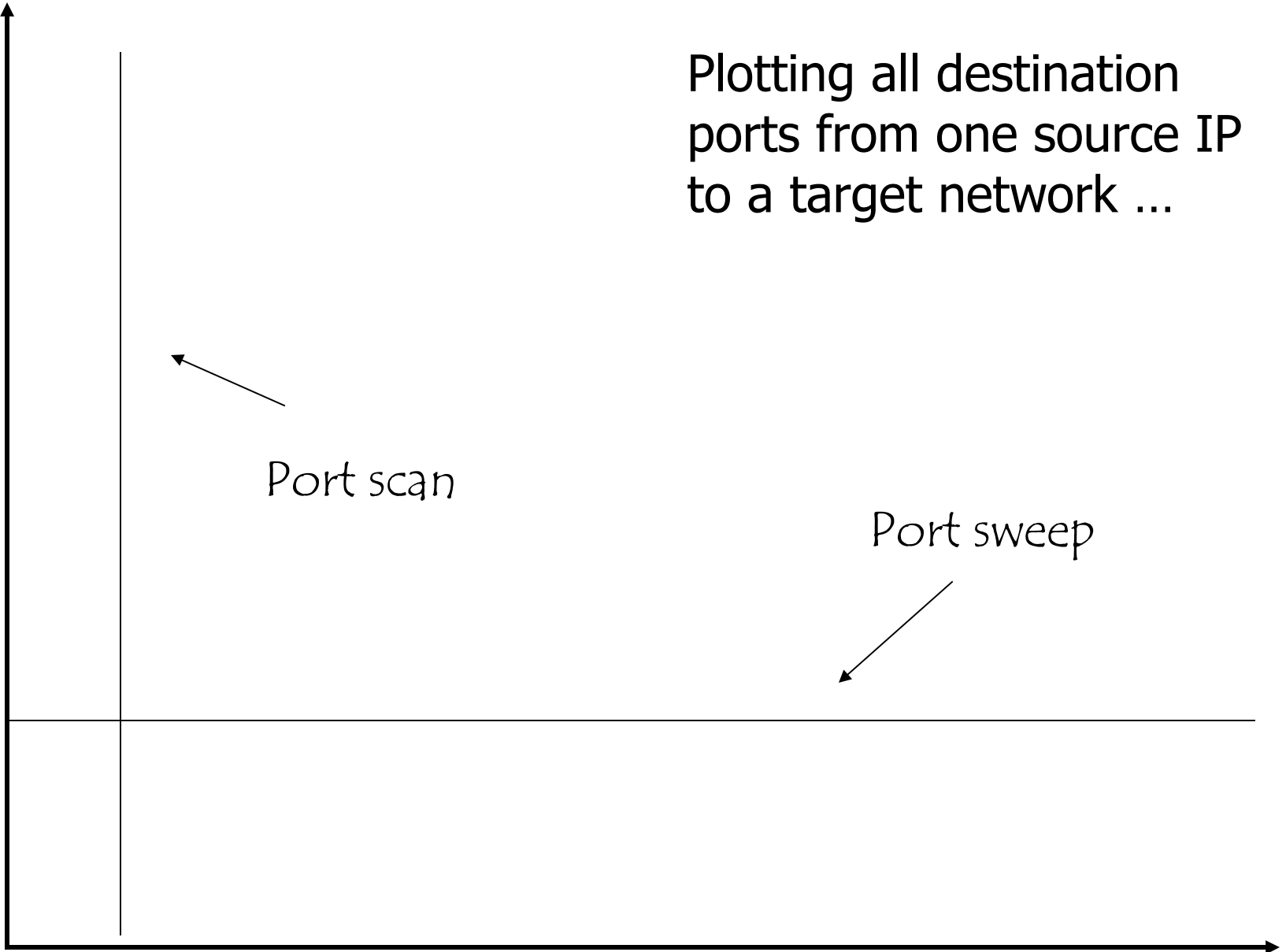
Plotting all destination ports from one source IP to a target network ...

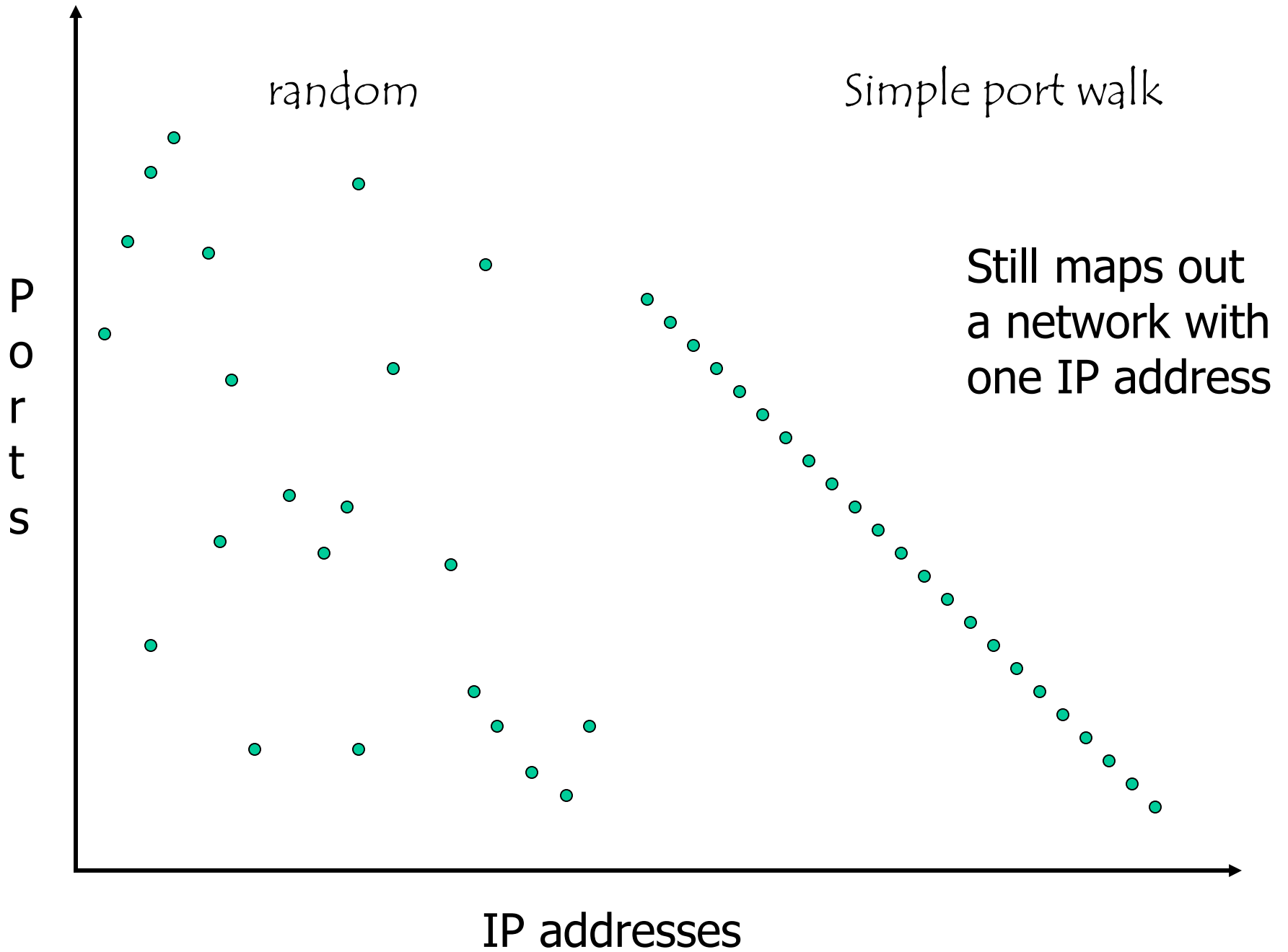
P
O
R
T
S

Port scan

Port sweep

IP addresses







N&H-IDS: signature detection

- New attacks can be detected 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



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.

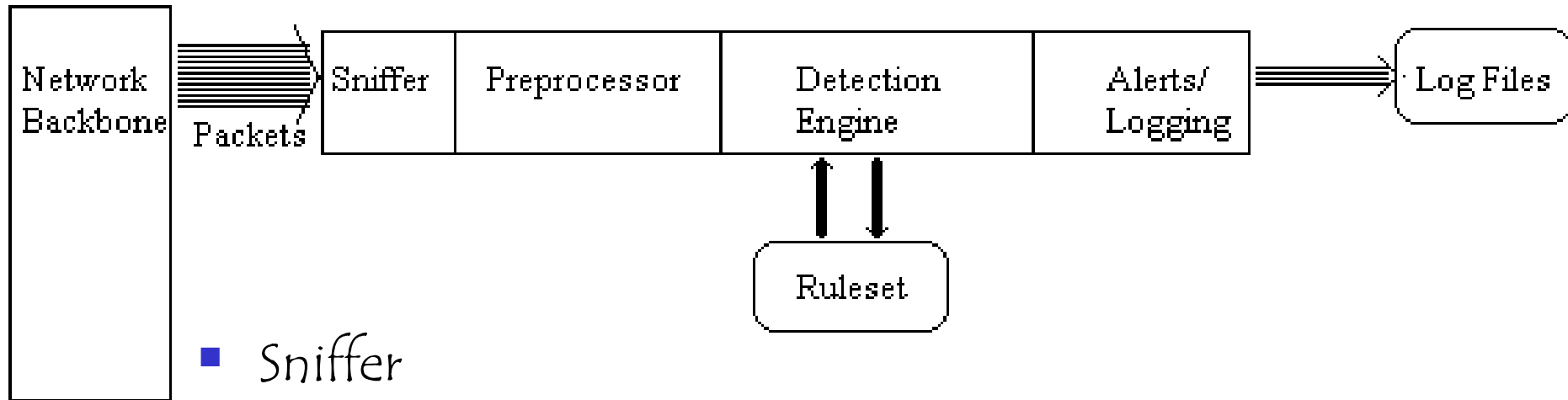


Snort Basics

- Intrusions have “signatures”
- Examples
 - Directory Traversal Vulnerability
 - Solaris Sadmin/IIS worm (2001)
 - Allowed HTTP GET requests to change to root directory with “../..”.
 - Allowed to copy cmd.exe into the Scripts directory.
 - Gained control usually at admin level

```
GET/ scripts/../../winnt/system32/cmd.exe /c+  
copy+ \wint\system32\CMD.exe+root.exe
```


Snort Architecture



- Sniffer
- Preprocessor
- Detection Engine
- Alert Logging



Using Snort

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



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

- Rules contains the rule header and the rule option.

```
alert tcp !10.1.1.0/24 any -> 10.1.1.0/24 any (flags: SF; msg: "SYN-FIN scan")
```

Alerts to traffic from outside the 10.1.1.x subnet to the 10.1.1.x subnet with the Syn and the Fin flags set.



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 Header Fields
 - Protocol Field
 - TCP
 - UDP
 - ICMP
 - IP
 - Others (ARP, RARP, GRE, ...) to come



Snort Rules

- Rule Header Fields
 - Source and Destination IP Address Field
 - Format: Address/netmask or any or
 - Address x.x.x.x
 - Netmask = bits of network mask
 - For example
 - 24.0.0.0/8 Class A
 - 24.3.0.0/16 Class b
 - 192.185.67.0/24 Class C
 - 192.185.67.188 host address
 - Special keywords:
 - any
 - ! (negation)
 - \$HOME_NET (variable defined elsewhere)



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

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

- Rule Options
 - Msg Option = message to print

Rule: alert udp any any -> 129.210.18.0 / 24 31337 \
(msg: "Back Orifice");

Log: [**] Back Orifice [**]
05/10-08:44:26.398345 192.120.81.5:60256 -> 129.210.18.34:31337
UDP TTL:41 TOS:0x0 ID:49951
Len: 8



Snort Rules Options

- The heart of intrusion detection engine,
- Four major categories of rule options.

General : provide information about the rule but
do not affect detection

Payload: look for data inside the packet
payload and can be inter-related

Non-payload: look for non-payload data

Post-detection: rule specific triggers that happen after
a rule has ``fired.''



Snort Rules

- Rule Options
 - Separated by parentheses

```
alert tcp !$HOME_NET any -> $HOME_NET any (flags: SF; \
msg: "Syn-Fin" scan";)
```



Snort Rules

- Rule Options

- Logto Option

- Specifies filename to which to log the activity.
 - Allows to separate the annoyances from the truly dangerous.

```
alert udp any any -> 129.210.18.0 / 24 31335 \  
(msg: "trinoo port"; logto "DDoS")
```



Snort Rules

- Rule Options, not payload
 - TTL option
 - Allows to use the time to live field in packet
 - Format: ttl: number

```
alert udp any any -> 129.210.18.0 / 24 33000;34000 \  
(msg: "Unix traceroute"; ttl: 1;)
```



Snort Rules

- Rule Options
 - ID option
 - 16-bit value found in the IP header of each datagram.

```
alert udp any any -> 129.210.18.0 / 24 33000;34000 \  
(msg: "Suspicious IP Identification"; ID: 0;)
```



Snort Rules

- Rule Options
 - Dsize option
 - Size of payload

```
alert icmp any any -> 129.210.18.0 / 24 any \  
(msg: "Large ICMP payload"; dsize: >1024;)
```



Snort Rules

- Rule Options
 - Sequence Option
 - Value of tcp sequence number
 - Ack option
 - Value of ack number in tcp

```
alert tcp any any -> any any \  
(msg: "Possible Shaft DDoS"; seq: 0x28374839;)
```

```
alert tcp any any -> any any \  
(msg: "nmap tcp ping"; flags: A; ack: 0;)
```



Snort Rules

- Rule Options
 - Itype and Icode Options
 - Select ICMP message type and operations code

```
alert icmp 1.1.1.0/24 any -> 129.210.18.0 / 24 any \  
(msg: "port unreachable"; itype: 3; icode: 3;)
```



Snort Rules

- Rule Options
 - Flags option

```
alert tcp any any -> any any \  
(msg: "null scan"; flags: 0;)
```




Snort Rules

- Rule Options
 - Content Option

```
alert udp $EXTERNAL_NET any -> $HOME_NET 53 \  
(msg: "Exploit bind tsig Overflow attempt"; \  
content: "|00 FA 00 FF|"; content: "/bin/sh");)
```



Snort Rules

- Rule Options
 - Offset option
 - Specifies offset of content
 - Depth option
 - Specifies how far into packet to search for content
 - Nocase option
 - Makes content searches case insensitive
 - Regex Option
 - Allows wildcards in content searches



Snort Rules

- Rule Options
 - Session Options
 - Allows to capture TCP session.
 - Rest Option
 - Allows an automatic active response
 - Tag Option
 - Allows to dynamically capture additional packages after a rule triggers.

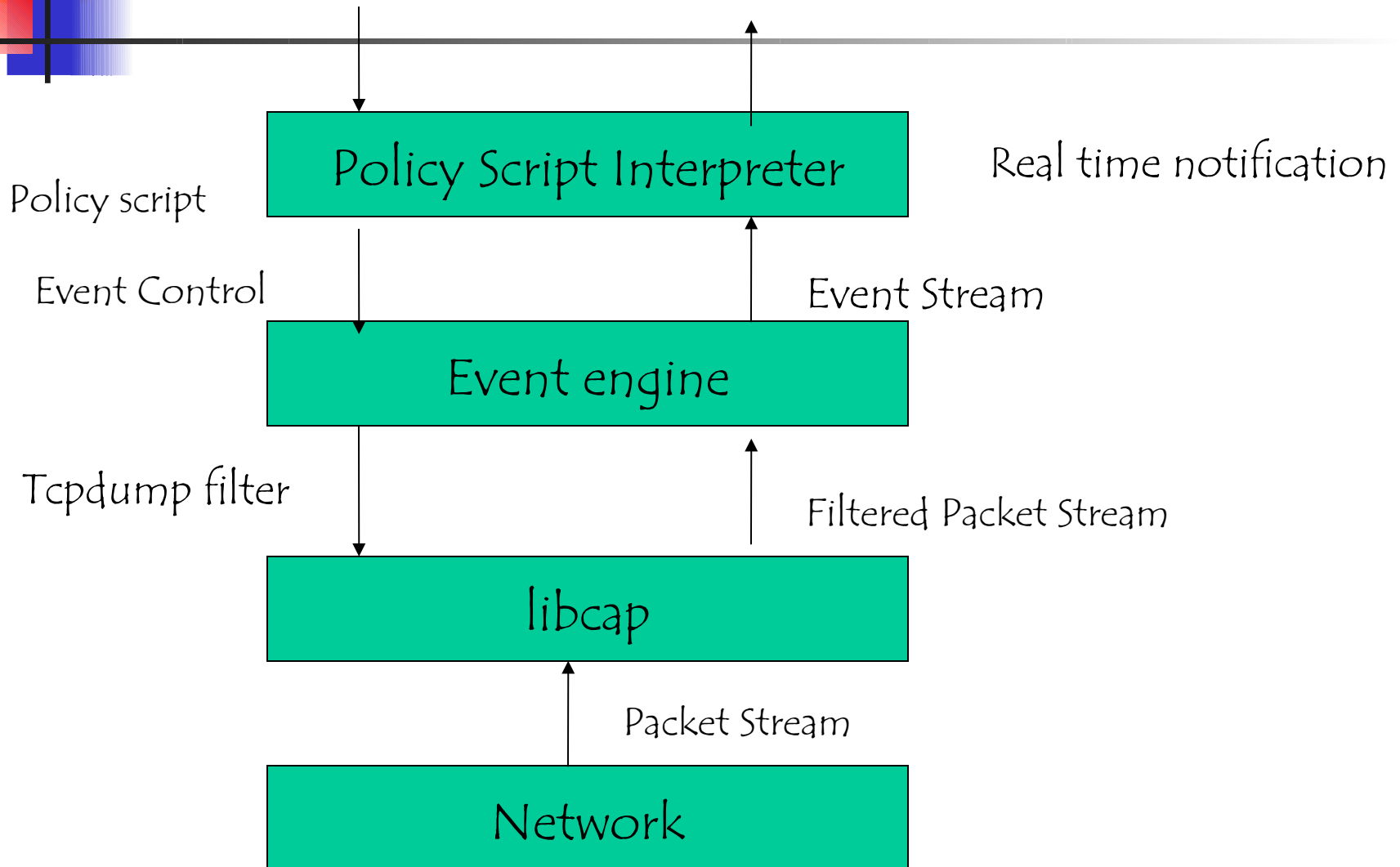


Rule Order

- A packet is checked should be checked in the order
drop > pass > alert > log this order.
- This scheme is the most secure since no packet passes through without being checked against all drop rules
- However most of the packets are normal traffic and do not show any intruder activity. Testing all of the packets against all alert rules requires a lot of processing power. You can change this order to a more efficient, but more dangerous.

Pass > Drop > Alert > Log

Structure of the Bro System





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.

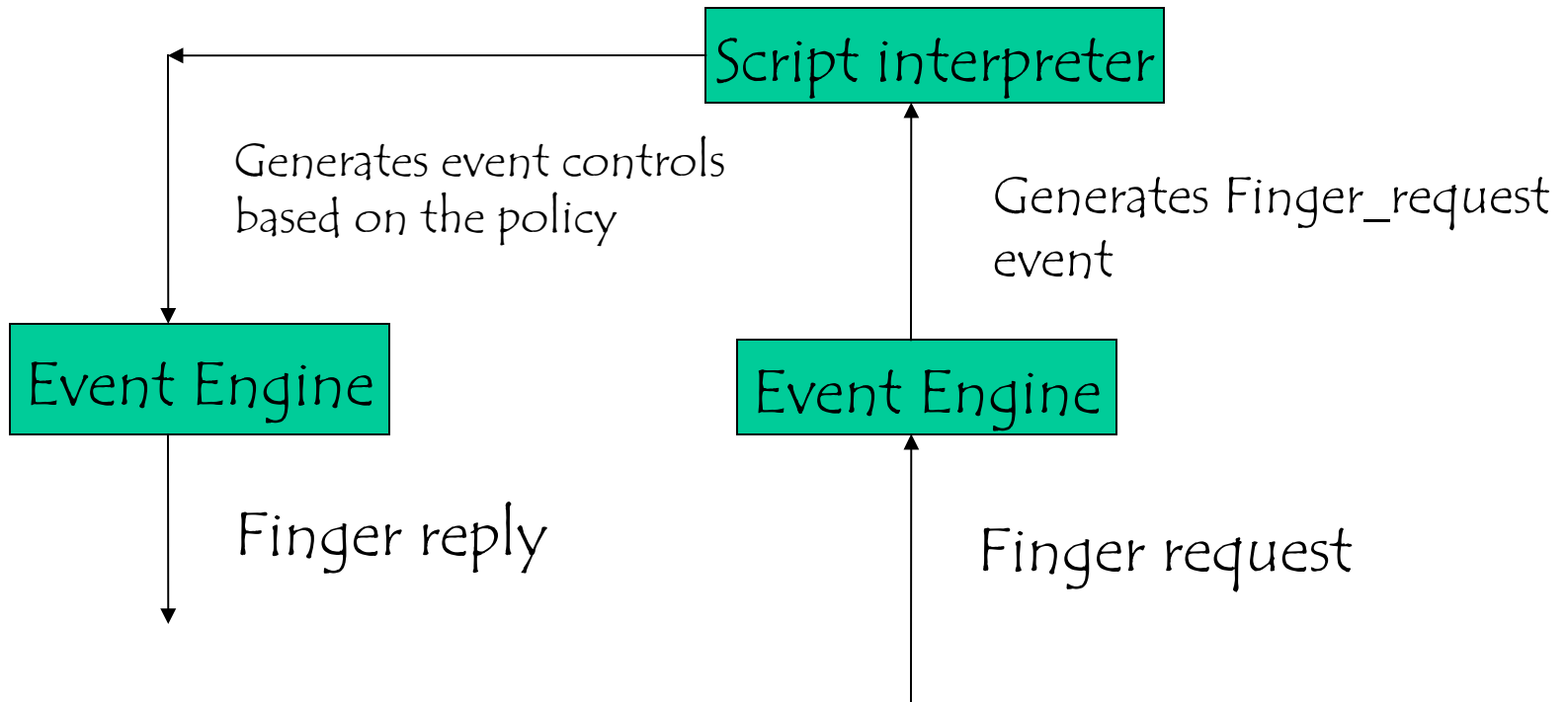


Policy Script Interpreter

- 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





Using a public network

- Several institutions have to connect remote, local networks
- Leased lines are too expensive
- The most convenient connection exploits a public network, eg the internet
- The connection security is very low because information flows on a public network
- This is a particular case of a problem we will meet again in clouds



Countermeasures - Robustness

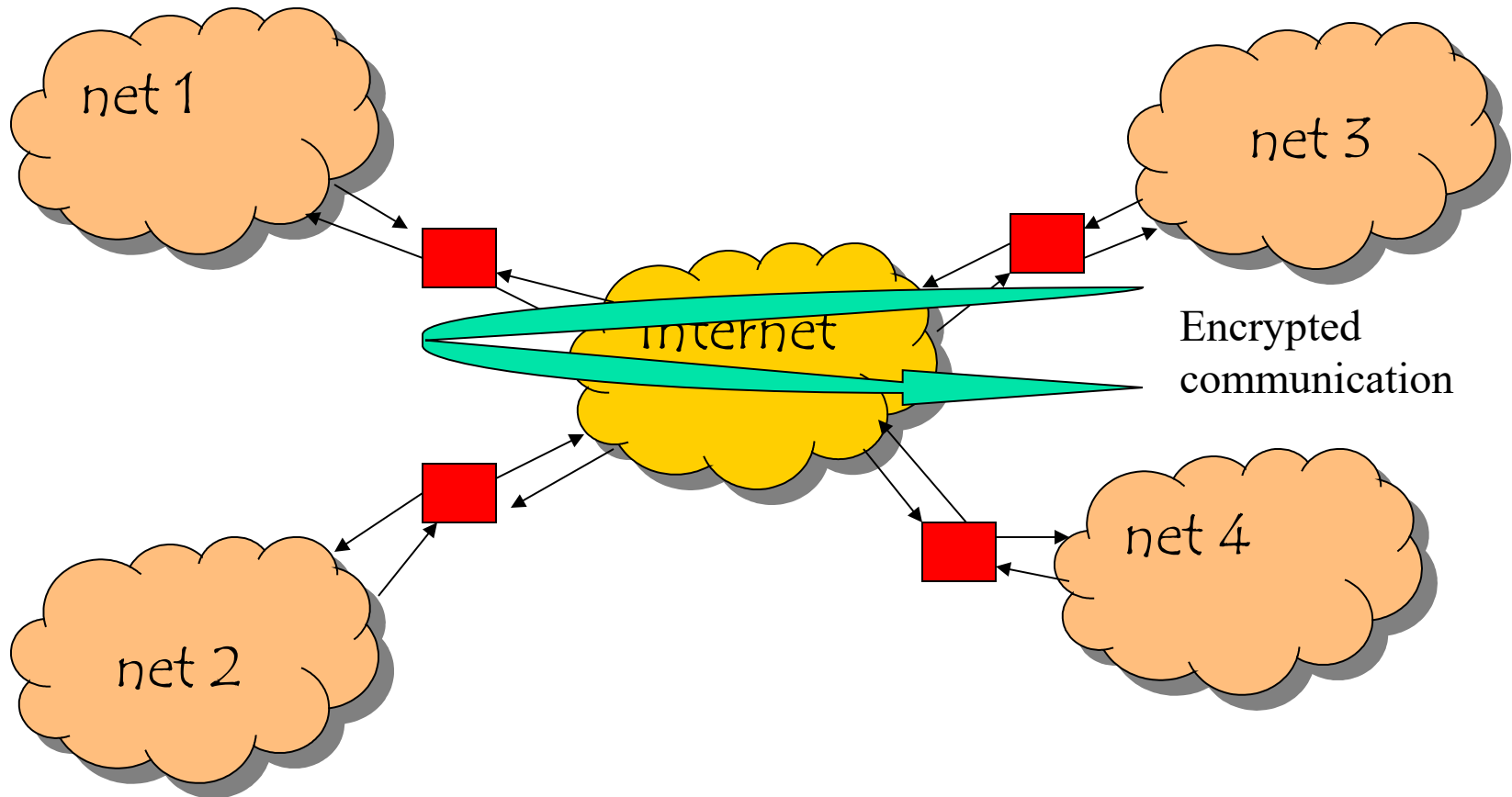
- Virtual Private Network
 - It emulates a secure connection on top of an unsafe connection
 - Assuming that each local network is protected by a firewall, secure connections are established among the firewalls
 - Secure = integrity and confidentiality are achieved by encrypting the traffic between any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



VPN ≠ VLAN

- VLAN denotes a logical network that is set up to minimise the number of conflicts
- A vlan is built by pairing
 - Transmission frequency
 - Tagswith a subset of the nodes
- No security property

Virtual Private Network





Virtual Private Network

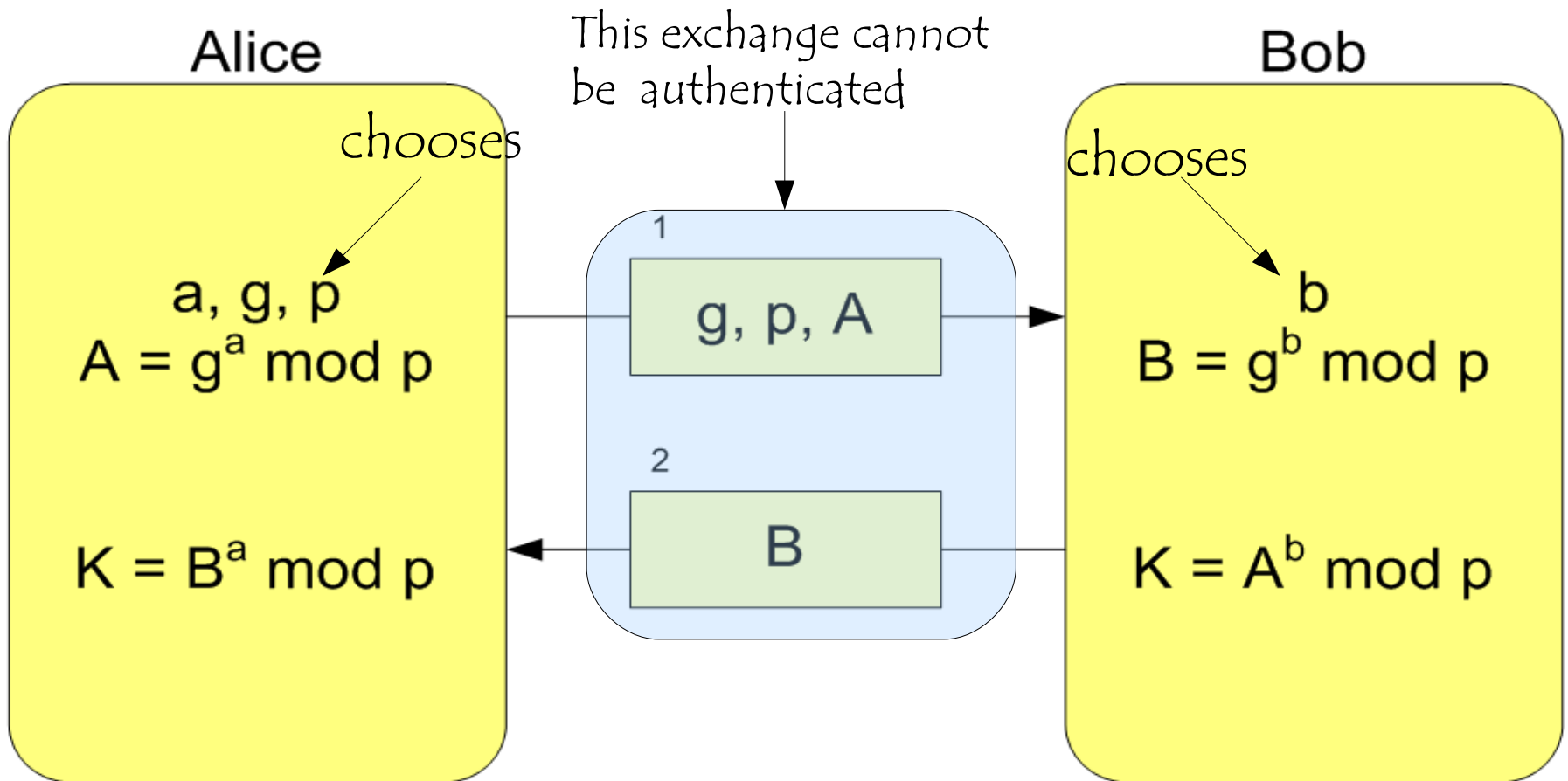
- Symmetric Encryption due to the large amount of transmitted data
- A distinct key for each pair of firewalls
- The key is updated according to the amount of exchanged data
- The key is chosen in a preamble and changed according to the amount of information that is exchanged



VPN and symmetric encryption - I

- The simplest strategy to share a key without transmitting it is the Diffie_Hellmann 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

Diffie-Hellman



$$K = A^b \text{ mod } p = (g^a \text{ mod } p)^b \text{ mod } p = g^{ab} \text{ mod } p = (g^b \text{ mod } p)^a \text{ mod } p = B^a \text{ mod } p$$



VPN and symmetric encryption -II

- Each firewall publishes a public key and knows the corresponding secret key
- The two keys make 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 the cleartext otherwise it discards the message
- On receiving a flood of fake messages, the receiver will be busy to discard them and cannot run legal applications or receive legal messages
- 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
 - 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

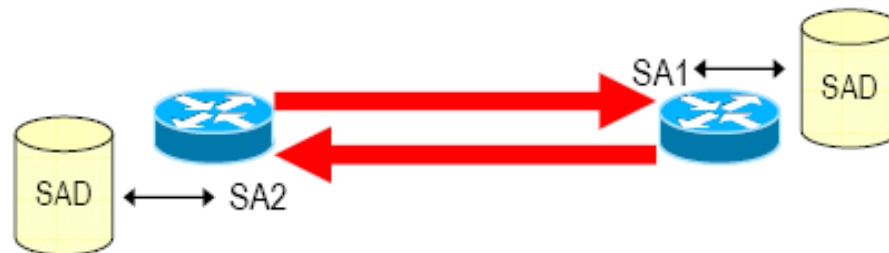
- **AH** (Authentication Header) it protect the integrity of and authenticate the data
- **ESP** (Encapsulating Security Payload) it offers confidentiality because of encryption.
- **IKE** (Internet Key Exchange) two partners can agree on the key to be used and on how long it should be used
- **ISAKMP** (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

SA unidirectional



→ SPI = Security Parameters Index

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

→ Destination Address based

⇒ Security Association managed at the receiver side

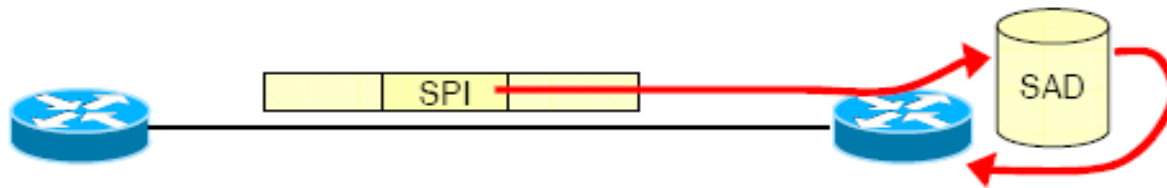
→ SAD = Security Associations Database

⇒ 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

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**



IPSEC

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

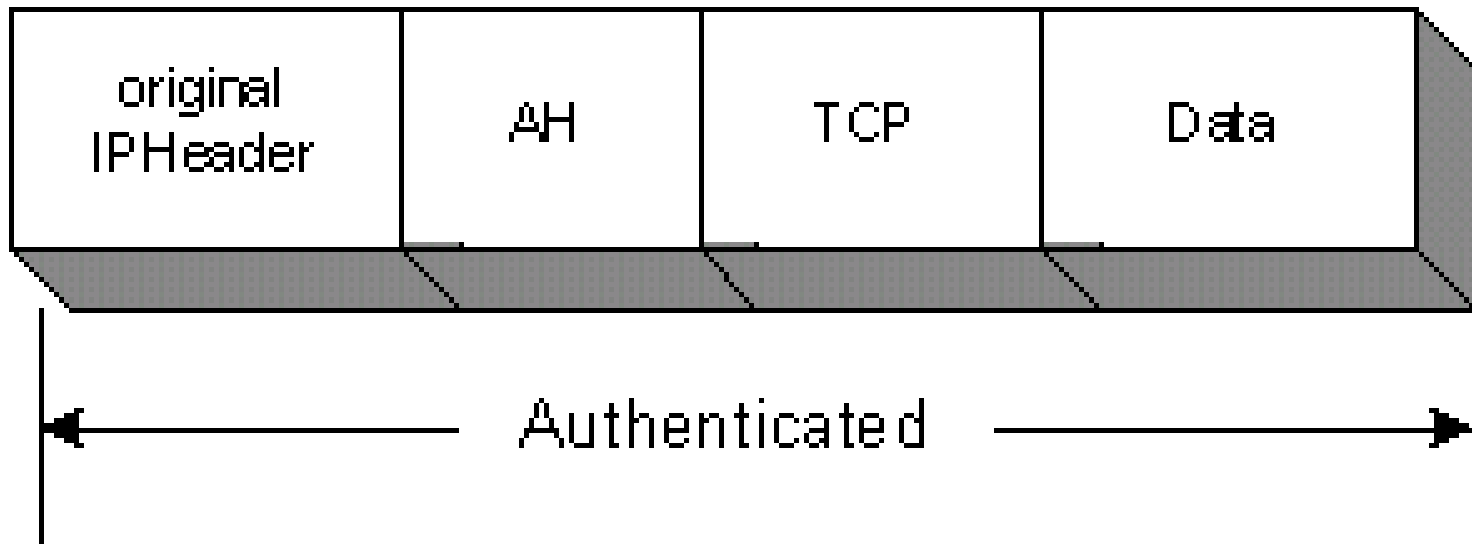
Transport mode (SA between two hosts) the security header immediately follows the IP header.

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

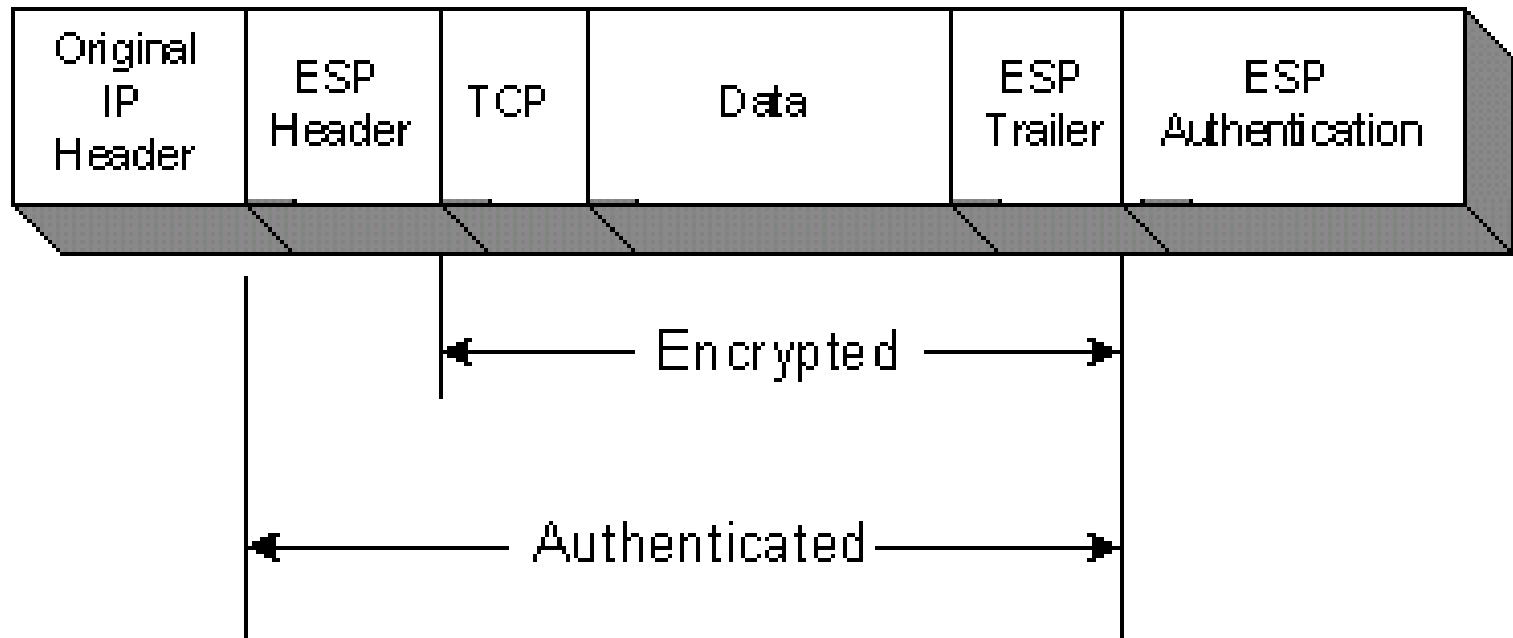
IPSEC

Authentication Header (AH)

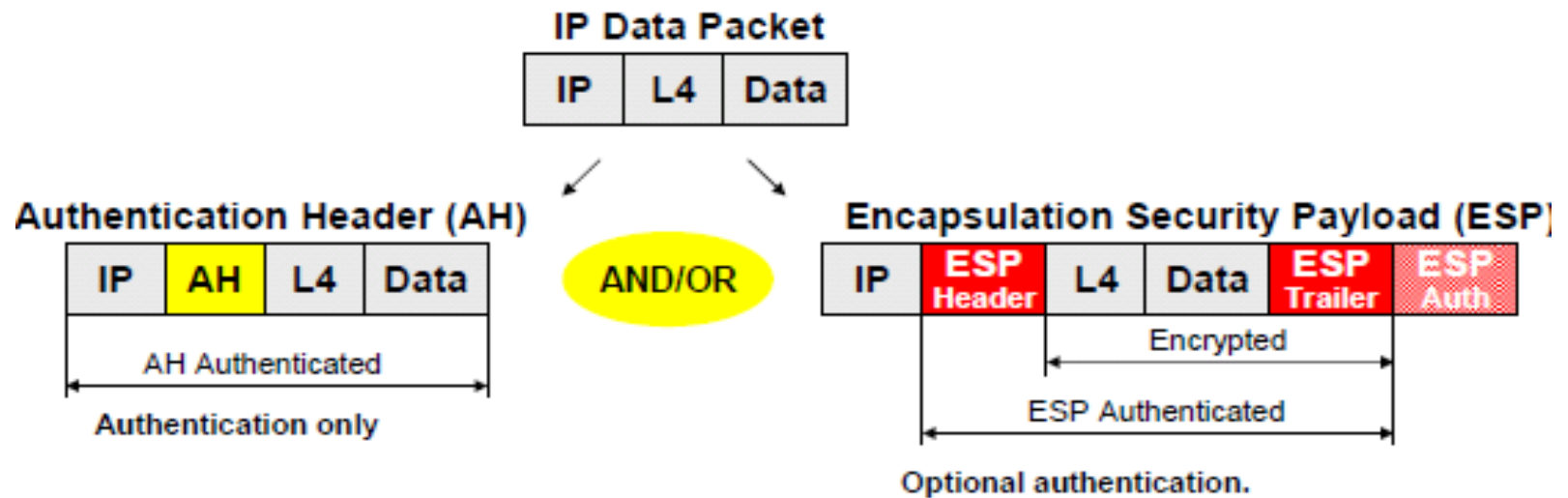


IPSEC

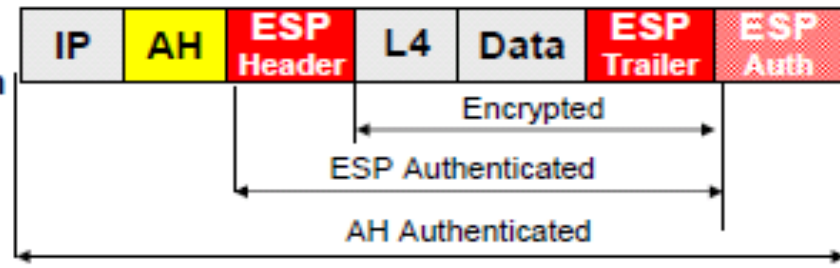
Encapsulating Payload Protocol (ESP)



IPSEC

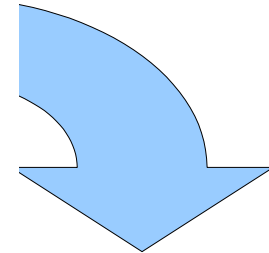


AH + ESP together:
first perform ESP then
AH computation



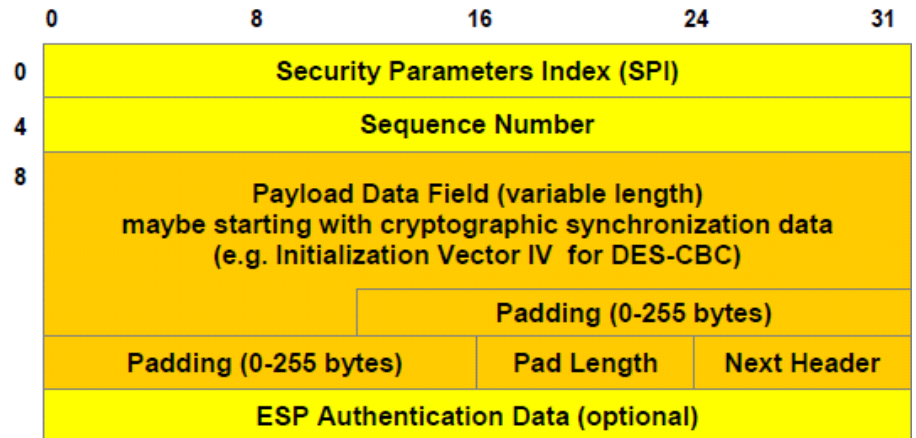
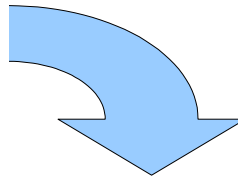
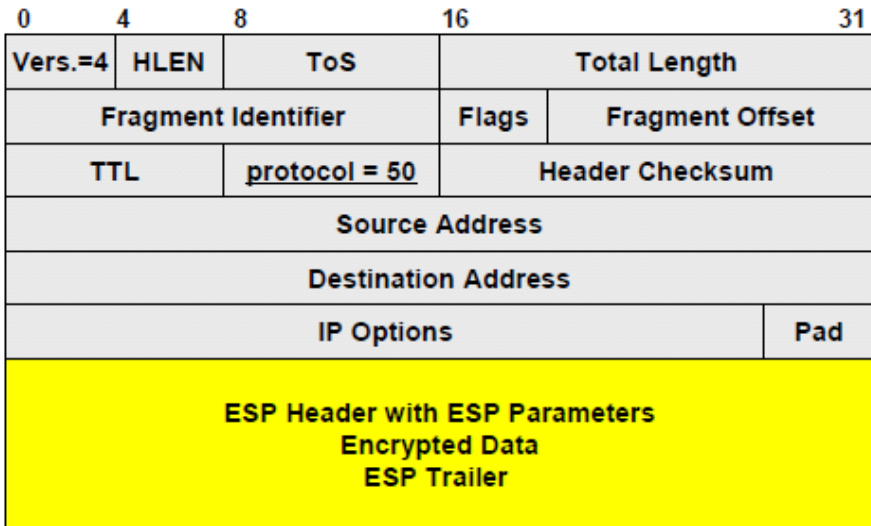
Authentication Mode

0	4	8	16	31
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)				

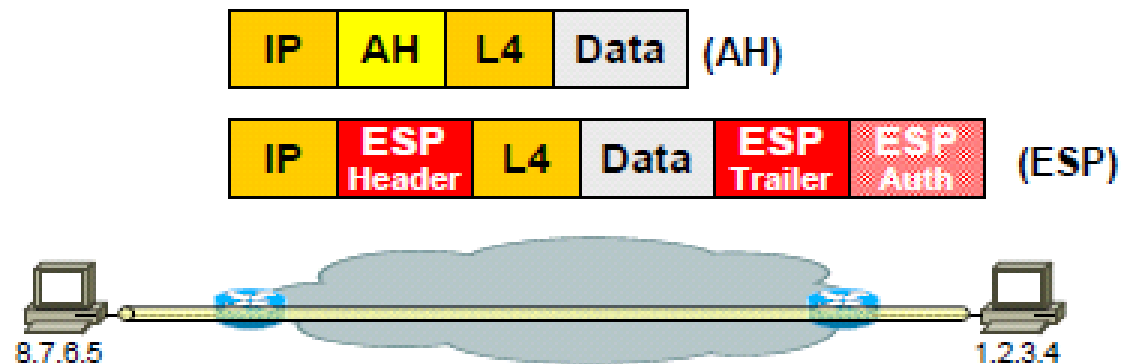
ESP



IPSEC

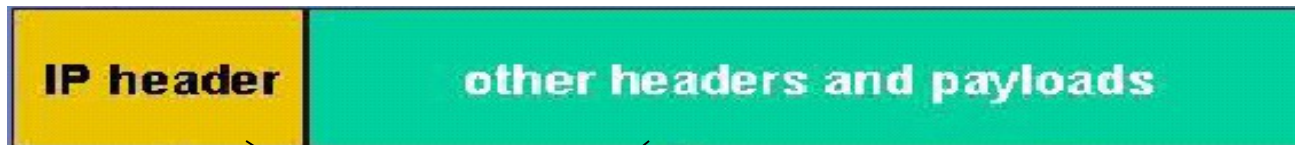
Transport Mode

- 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 packet



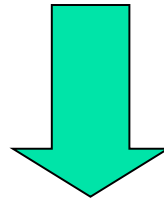
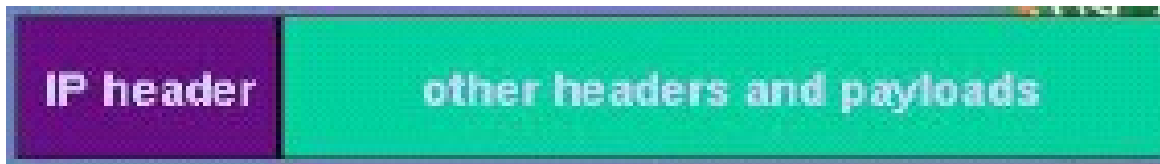
MD5/SHA-1



Authenticated packet

IPSEC: **ESP** in Transport Mode

Original IP packet

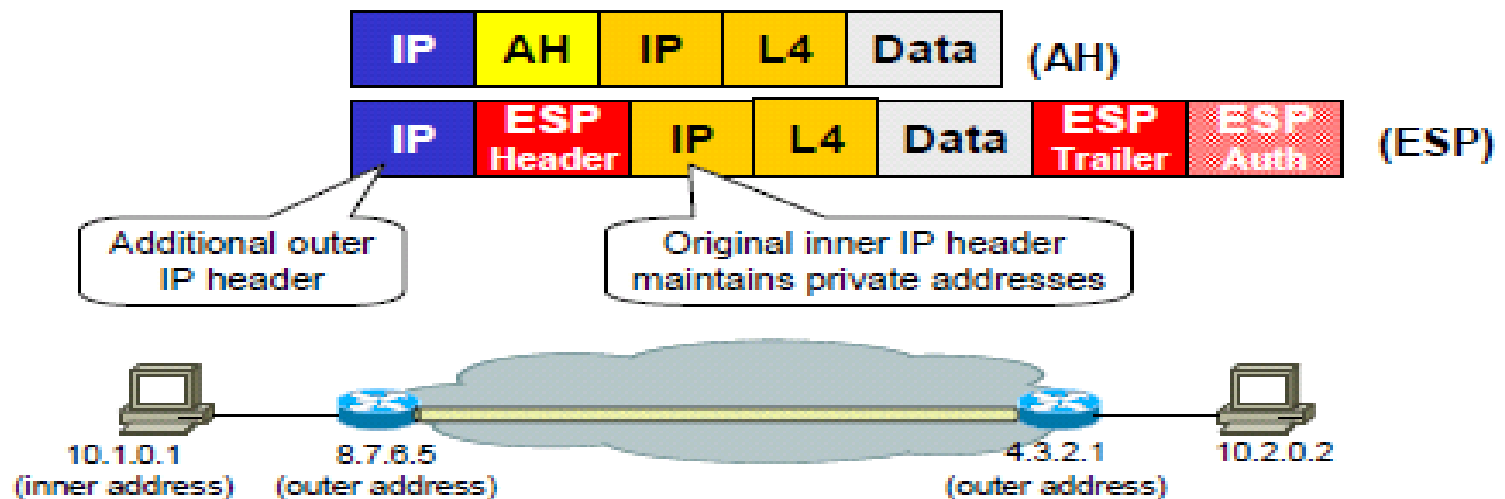


IP packet with ESP in Transport mode

IPSEC

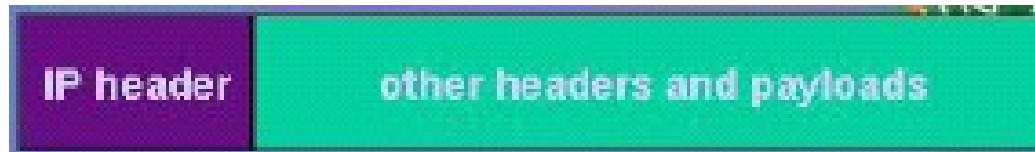
Tunnel Mode for Site-to-Site VPN

- Whole original IP packet is IPsec-encapsulated.
- Used for VPNs.
- Does hide traffic patterns!

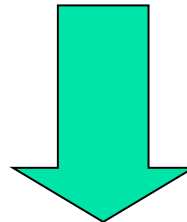


IPSEC: ESP in Tunnel Mode

Original IP packet

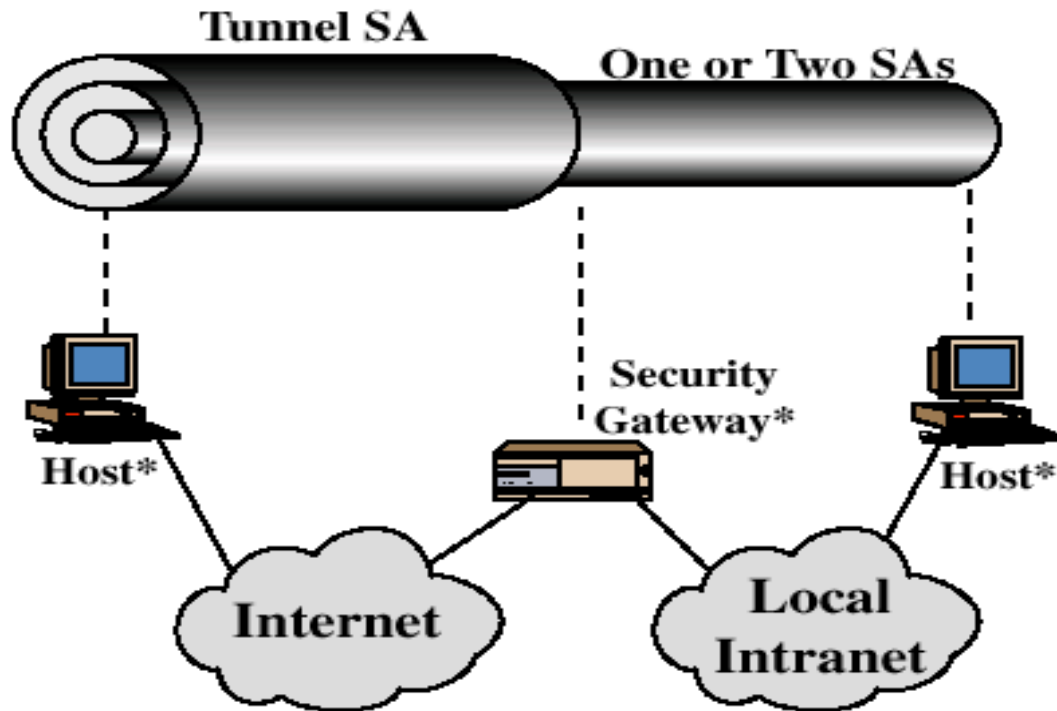


new IP header



IP packet ESP + Tunnel mode

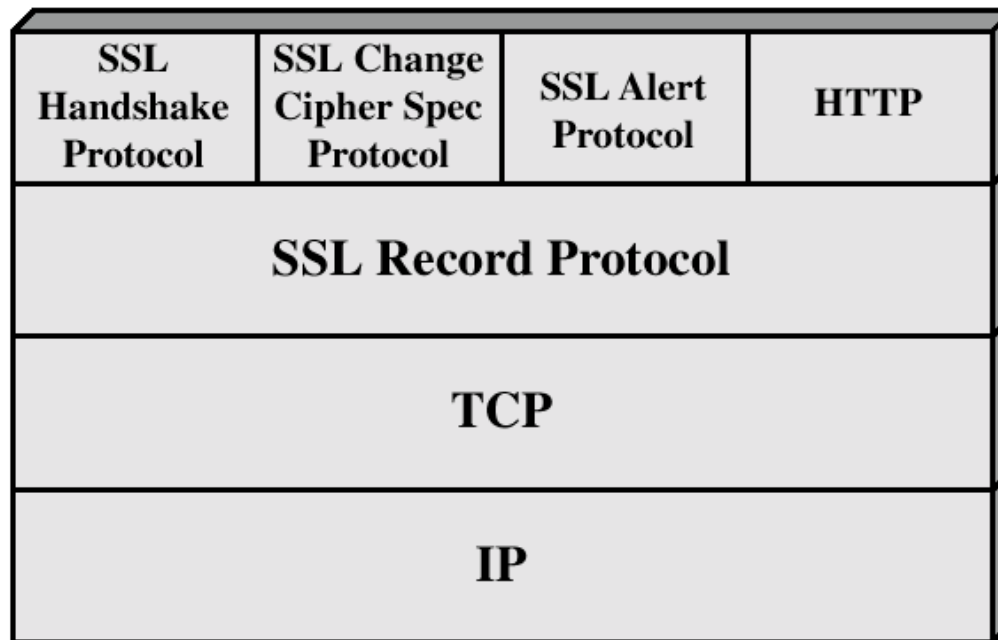
Applying several SAs



(d) Case 4



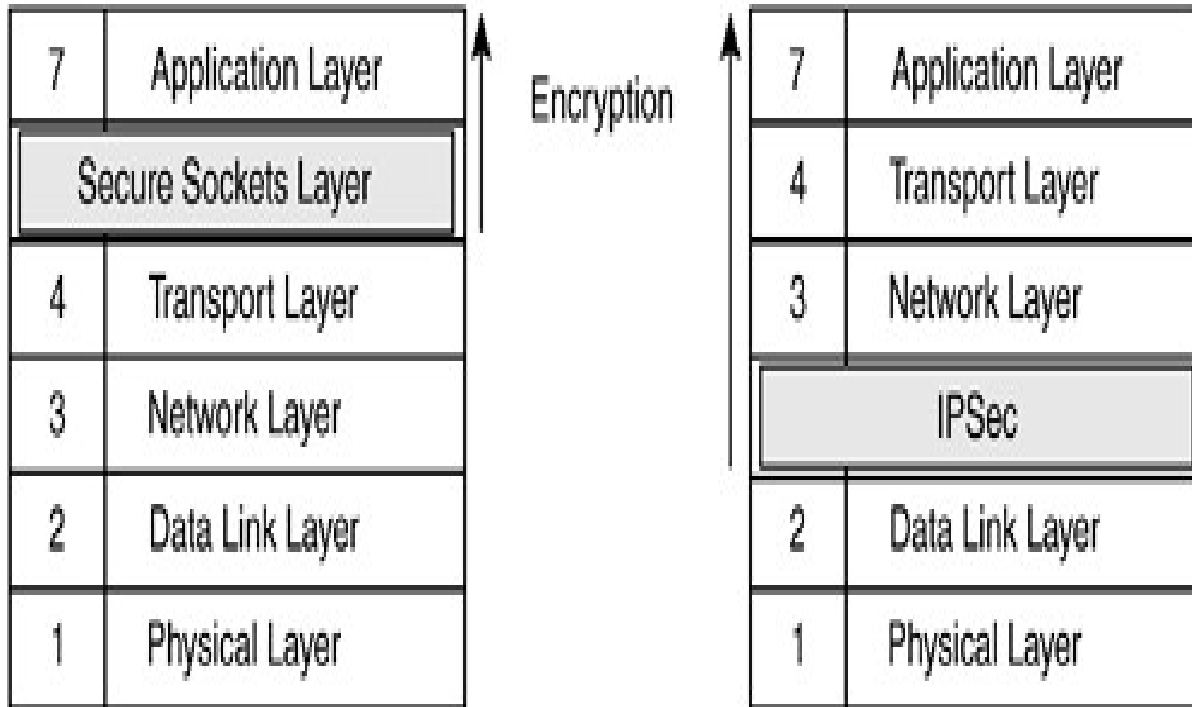
SSL = applicative VPN



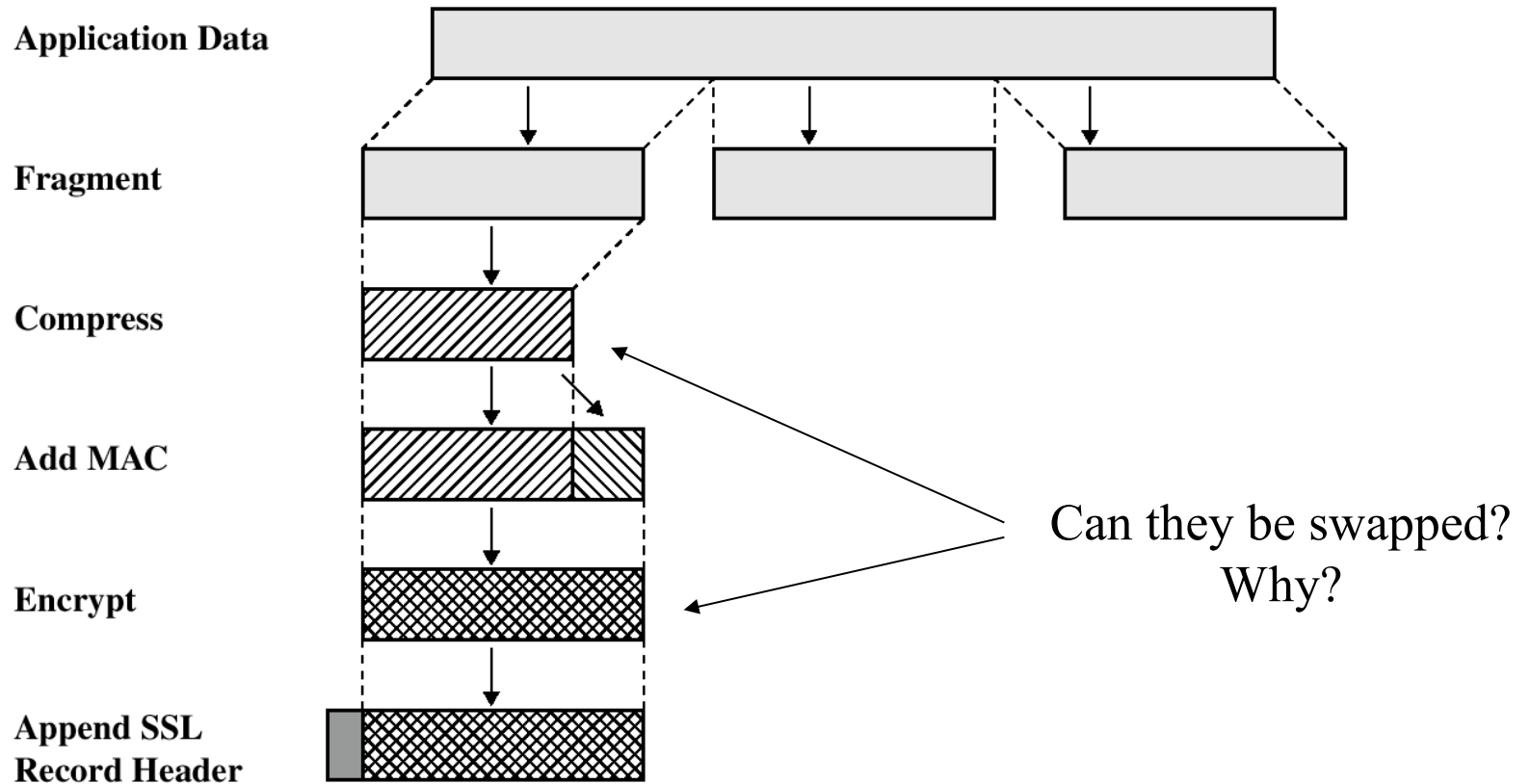
Four protocols

Figure 7.2 SSL Protocol Stack

SSL vs IPSEC



SSL





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: to compute the MAC on the server data ;
- Client write MAC secret: to compute the MAC on the client data;
- Server write key: to encrypt the data server → client ;
- Client write key: to encrypt the data client → server ;
- Initialization vectors: a data for Cipher Block Chaining encryption. IS is shared by both partners because it is needed both to encrypt and to decrypt.
- Sequence numbers $0.. 2^{64}-1$: each partner stores and manages the sequence numbers to send and receive messages on each connection. A number is zeroed when one partner sends a change cipher spec.



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 Protocols

- Handshaking Protocol
 - Establish communication variables
- ChangeCipherSpec Protocol
 - Alert to a change in communication variables
- Alert Protocol
 - Messages important to SSL connections
- Application Protocol: the one that is encrypted



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))
```

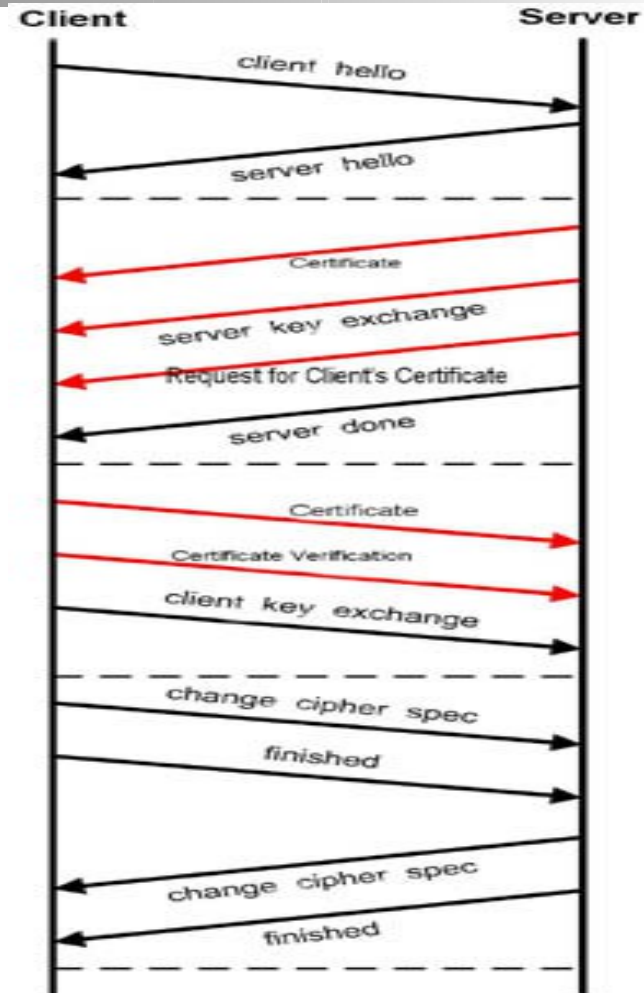
where :

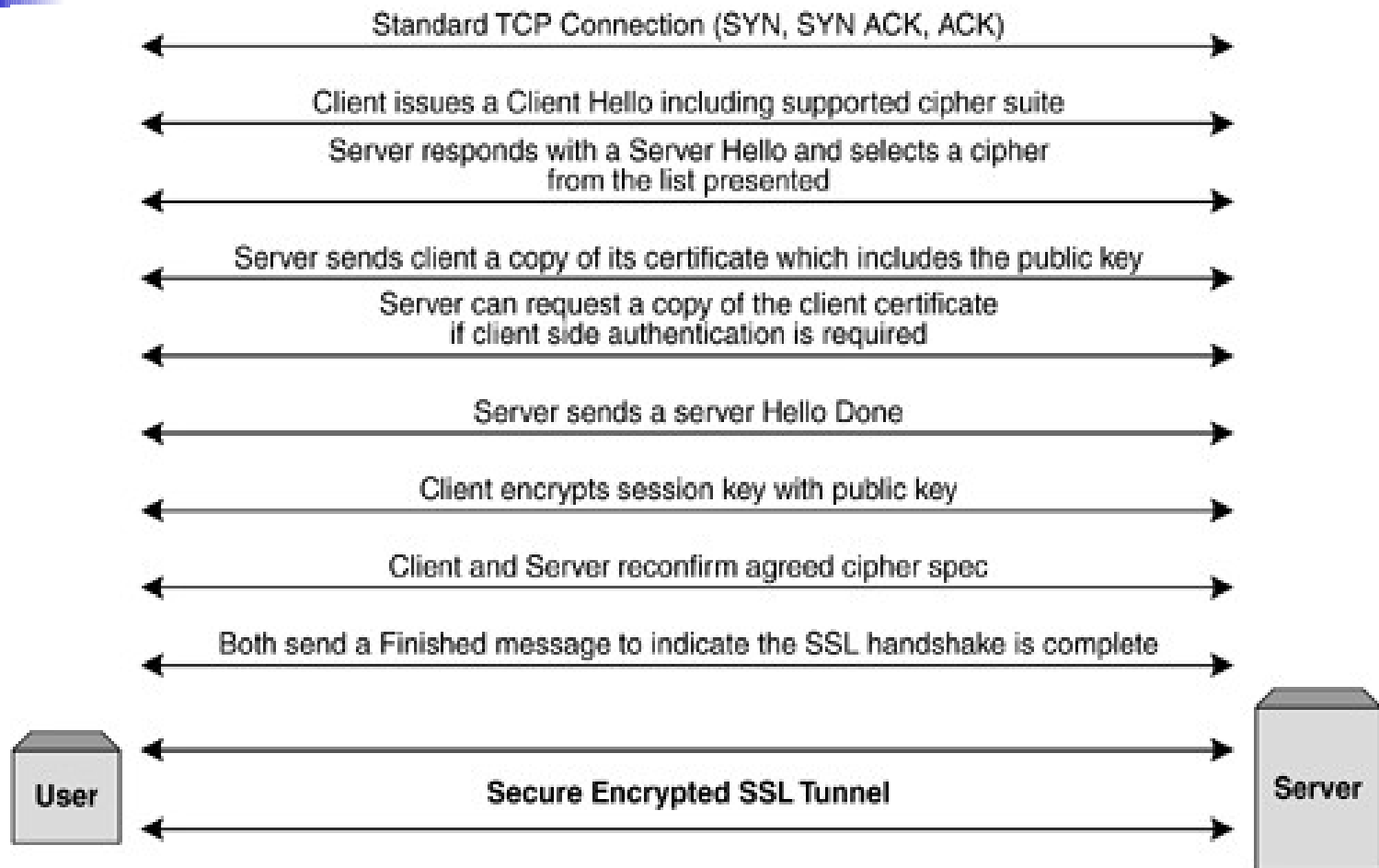
- `||` = concatenation;
- `MAC_write_secret`: secret shared key;
- `hash`: hash algorithm (MD5 or SHA-1);
- `pad_1`: byte 0x36 (00110110) repeated 48 times (384 bit) for MD5 and 40 (320 bit) for SHA-1;
- `pad_2`: byte 0x5C (01011100) repeated 48 times for MD5 and 40 for SHA-1;
- `seq_num`: sequential 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



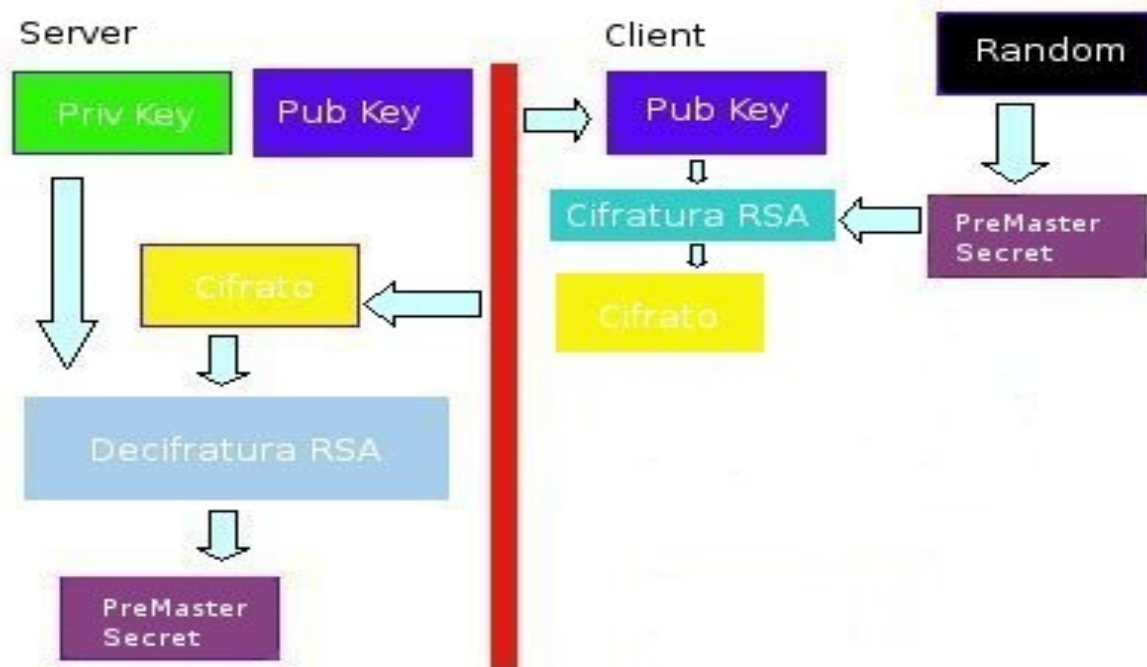




In brief ...

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: Assigned by the CA to identify the certificate;
- Signature algorithm: the algorithm the CA uses to sign the certificate.
- Issuer: the 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 public key
 - Public key algorithm: algorithm used to generate the public 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 certificate signature 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



Detail: 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



Detail 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



Detail: 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



Detail: 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

Detail 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

Detail: 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 = Exception

- 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 emulated on top the OS using the OS one
- Large set of choices = MAC + DAC + RBAC ...
- It increases the security of the applications it supports



Security Enhanced Linux

- A set of mechanisms to implement MAC + 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.**

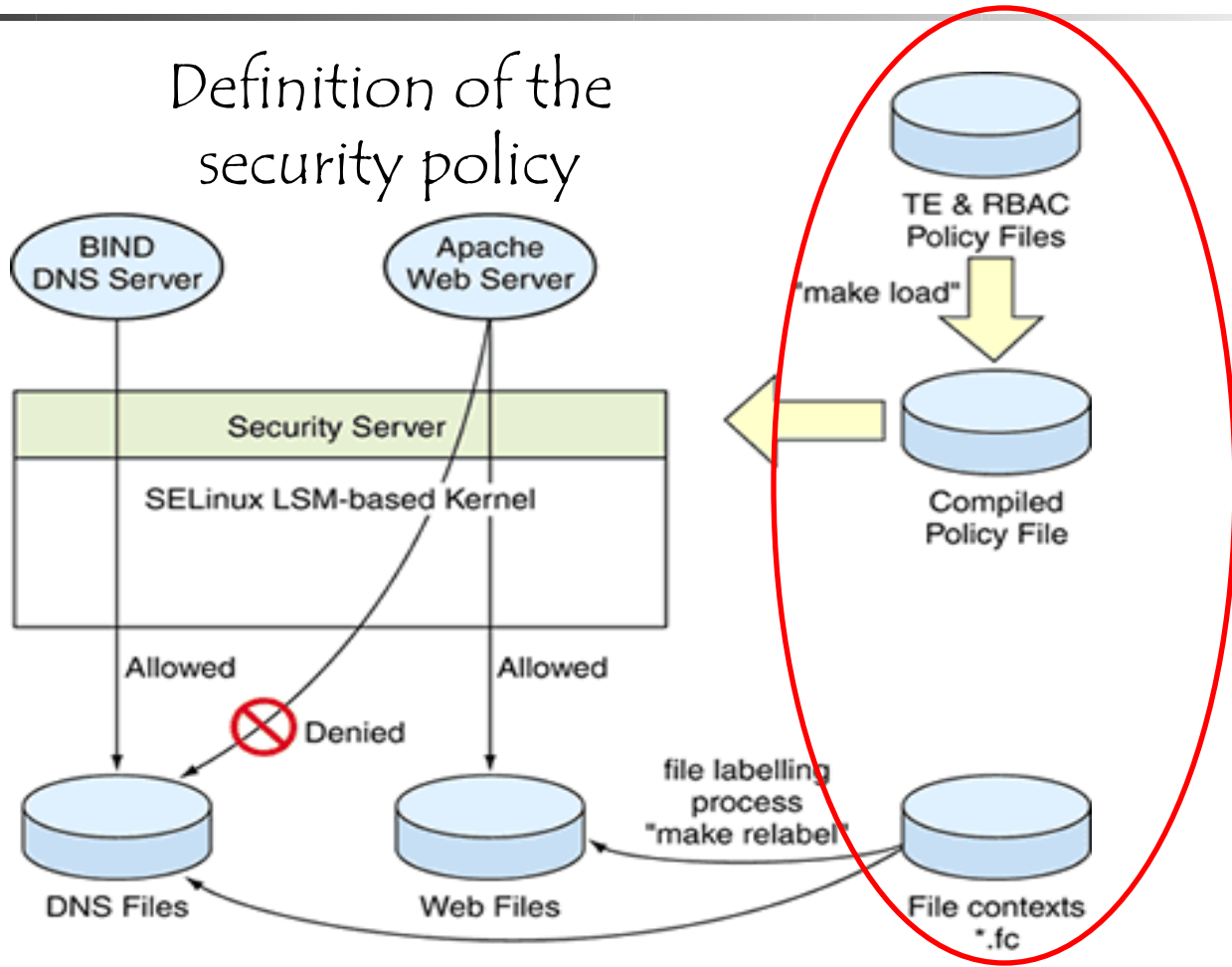


An interesting comment...

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 = rights 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



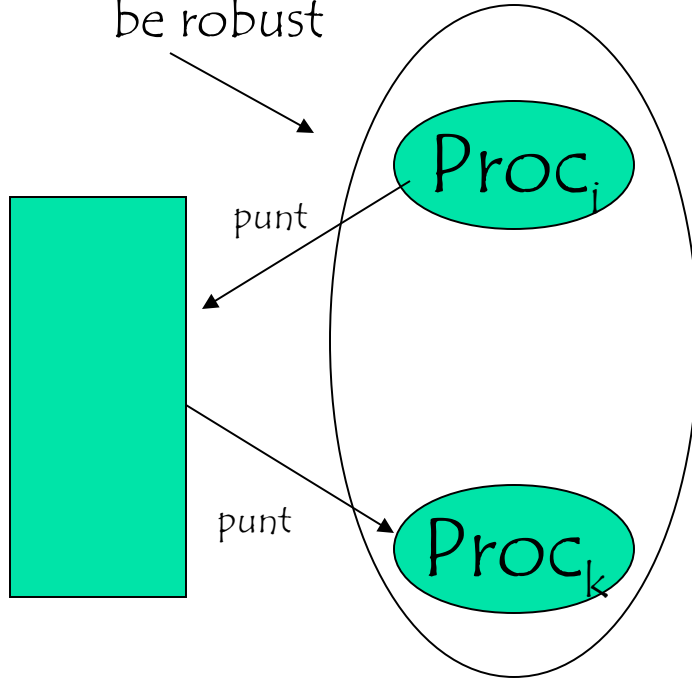
General Model - SID

- 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

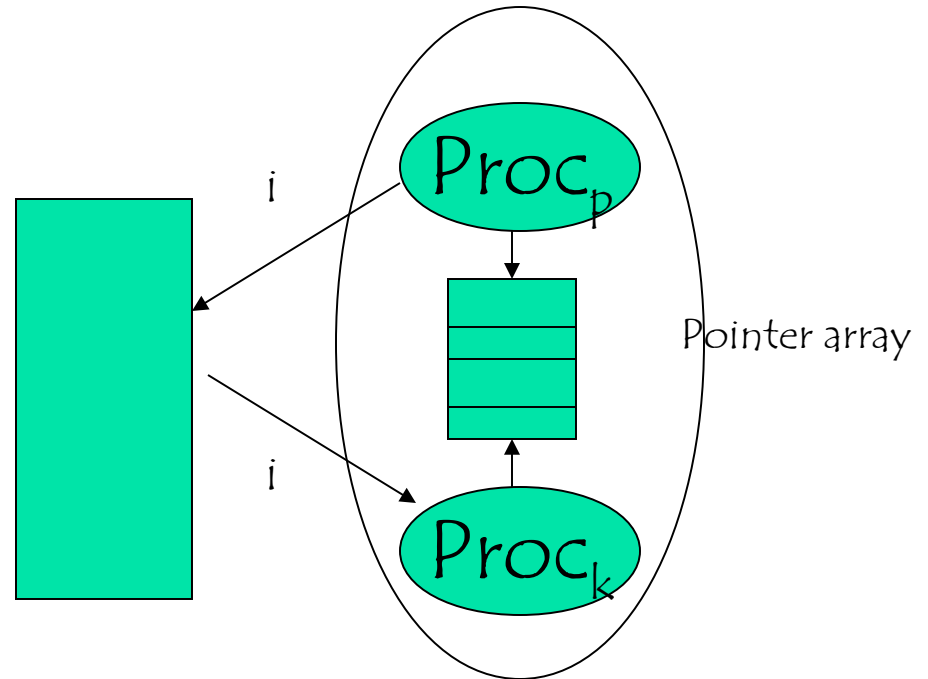
We have already seen this

Pointer - I

Package that should
be robust

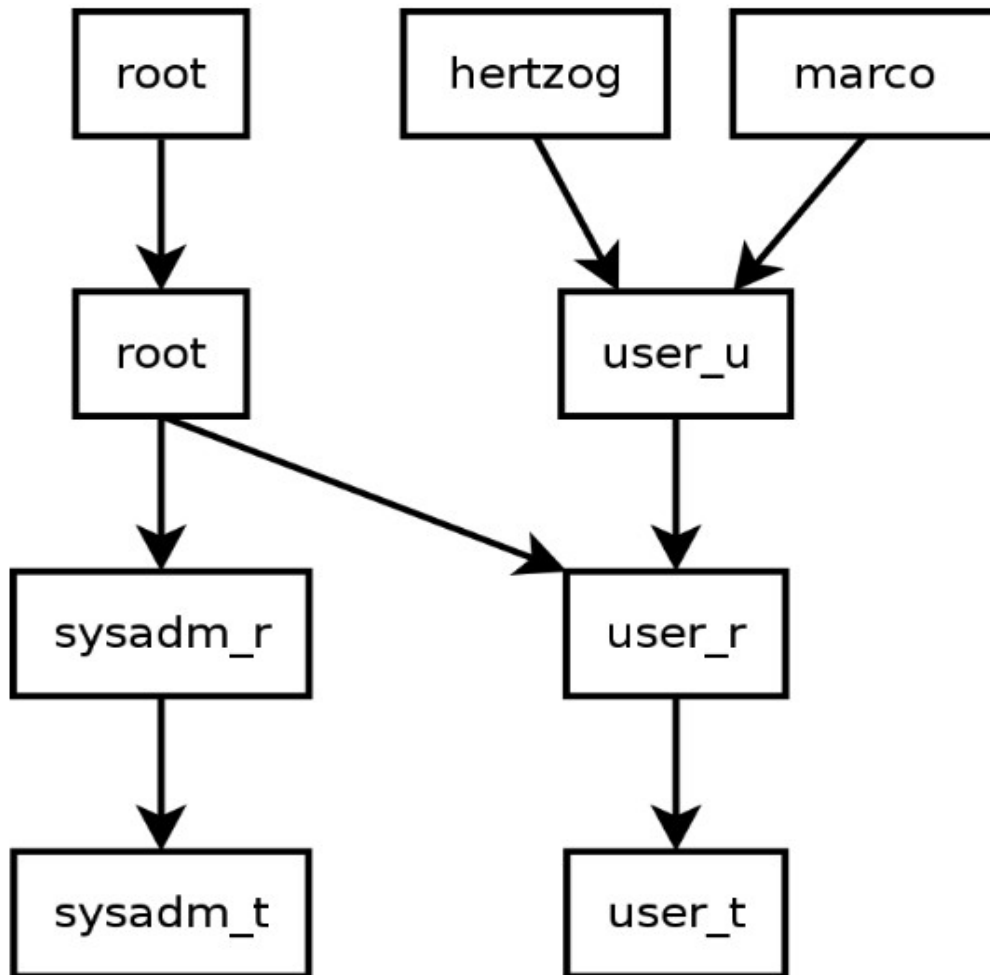


A more robust version



An index is transformed into a
pointer by accessing the
pointer array

Relazioni tra nomi, ruoli etc



**Utenti
Unix**

↓ Uno a uno

**Identità
SELinux**

↓ Uno a N

Ruoli

↓ Uno a uno

Dominio



Type Enforcement

- *Object(s)*: items in a system that are acted upon (files, IPC, sockets, etc....)
- *Subject(s)*: process that are requesting access to an object
- All Objects and Subjects contain a security context
- *Security Context(s)* are composed of four parts
- All Security Context components are checked against the policy to see if access is allowed.
- Type is the base component while role and user are used to further restrict type enforcement



TE Access Control

allow user_t bin_t : file {read execute write getattr setattr}

- *Source type(s)*: The domain type of the process accessing the object
- *Target type(s)*: The type of the object being accessed by the process
- *Object class(es)*: The class of object to permit access to
- *Permission(s)*: The kind of access permitted for the indicated object class



Type Enforcement

- Several major keywords
 - `type`
 - `attribute`
 - `typeattribute`
 - `typealias`
 - `allow`
 - `dontaudit`
 - `auditallow`
 - `neverallow`

Type Enforcement

```
rule na    src_type_set target_type_set : class_set perm_set;
allow user_t bin_t : file { read getattr } ;
allow user_t bin_t : dir { read getattr search } ;

#invalid since file does not have a search permission
allow user_t bin_t { file dir } {read getattr search } ;
#valid

#dontaudit when this access is denied
dontaudit httpd_t etc_t : dir search ;

#audit when this access is allowed
#by default allowed access is not audited
auditallow domain shadow_t : file write ;

#This statement may never be allowed by any rule
neverallow user_t shadow_t : file write

allow user_t bin_t : { file dir } * ;
allow user_t bin_t : file ~{ write setattr ioctl } ;
```



Domain Transitions

- Analogous to SetUID programs
- Joe running as user_t (untrusted user) needs to change his password. How does Joe change his password?
- allow user_t passwd_exec_t : file {getattr execute}
- allow passwd_t passwd_exec_t : file entrypoint
- allow user_t passwd_t : process transition
- What does this solve? Restricts trusted domain passwd_t and allows user_t to transition to it.
- Implicit domain transitions provided via type_transition.



Users & Roles

- First and second component of a security context
- SELinux usernames and DAC usernames are not synonymous
- Roles are collections of types geared towards a purpose
- Roles can be used to further restrict actions on the system
- SELinux usernames are granted roles in the system



MLS

- MLS portion of Security Context is composed of 4 parts
 - Low/High
 - Sensitivity/Category
- Includes syntax to define dominance of security levels
- Subjects with range of levels considered *trusted subjects*
- Implements a variation of Bell-La Padula



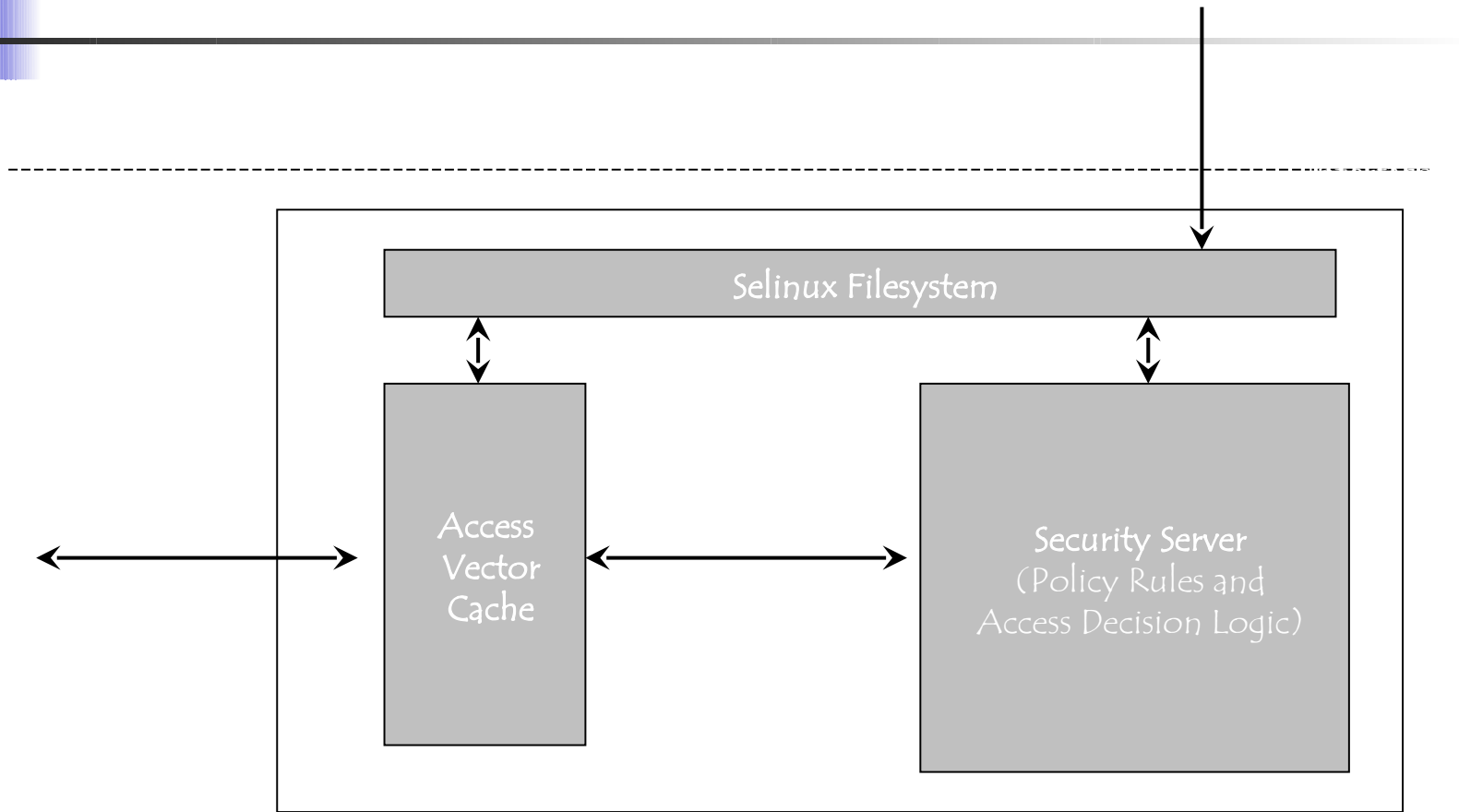
Architecture



LSM

- Kernel framework for security modules
- Provides a set of hooks to implement further security checks
- Usually placed after existing DAC checks and before resource access
- Implications? SELinux check is not called if the DAC fails
- Makes auditing difficult at times.

SELinux LSM Module

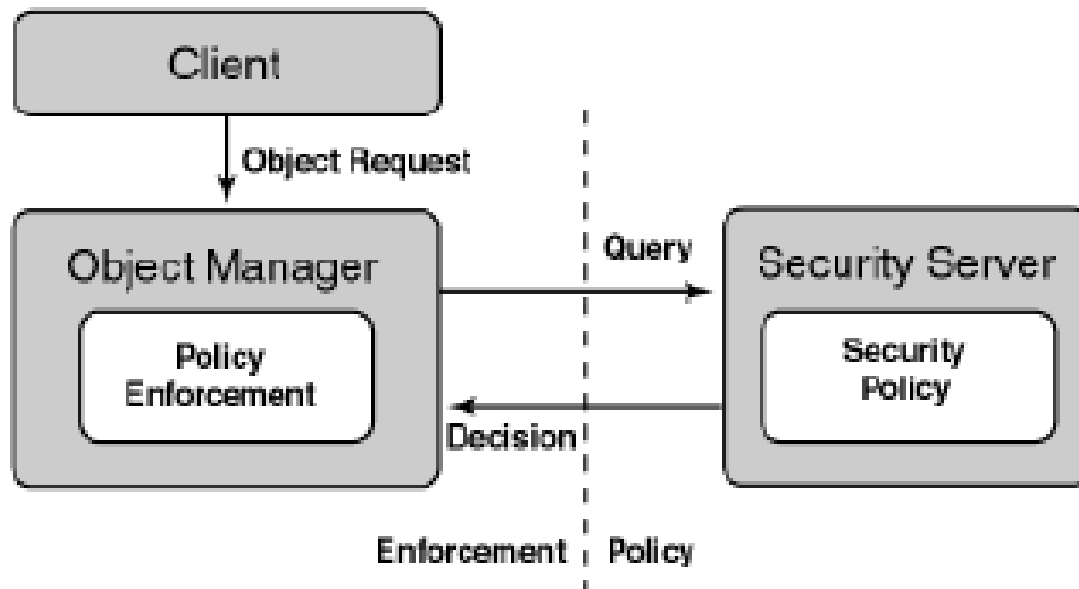




General Model - PSID

- PSID = SID for persistent 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

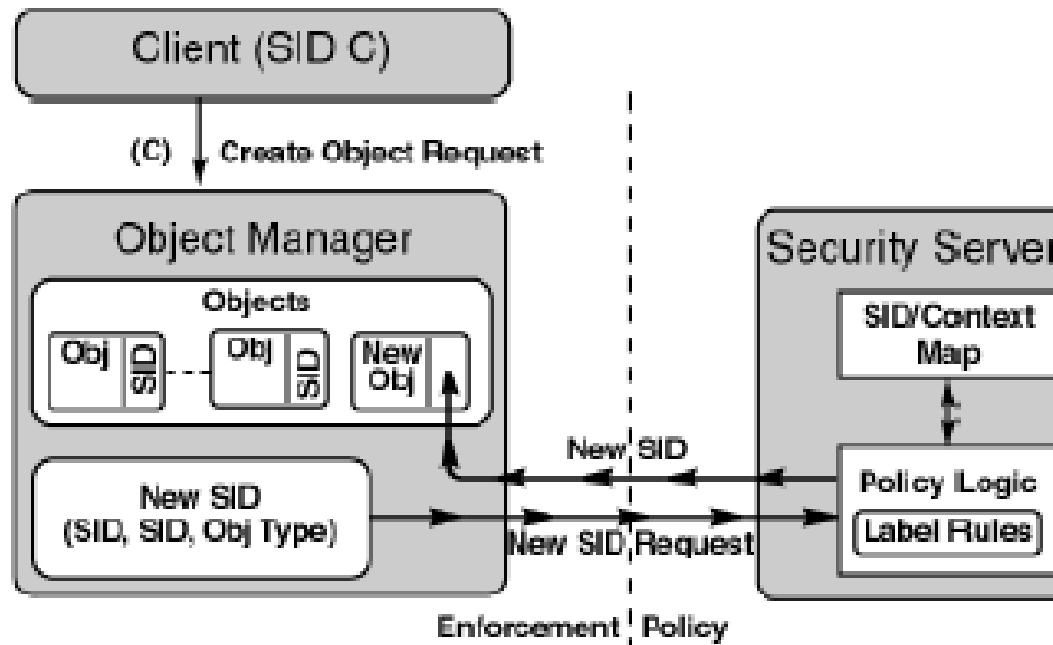
General model - Interactions



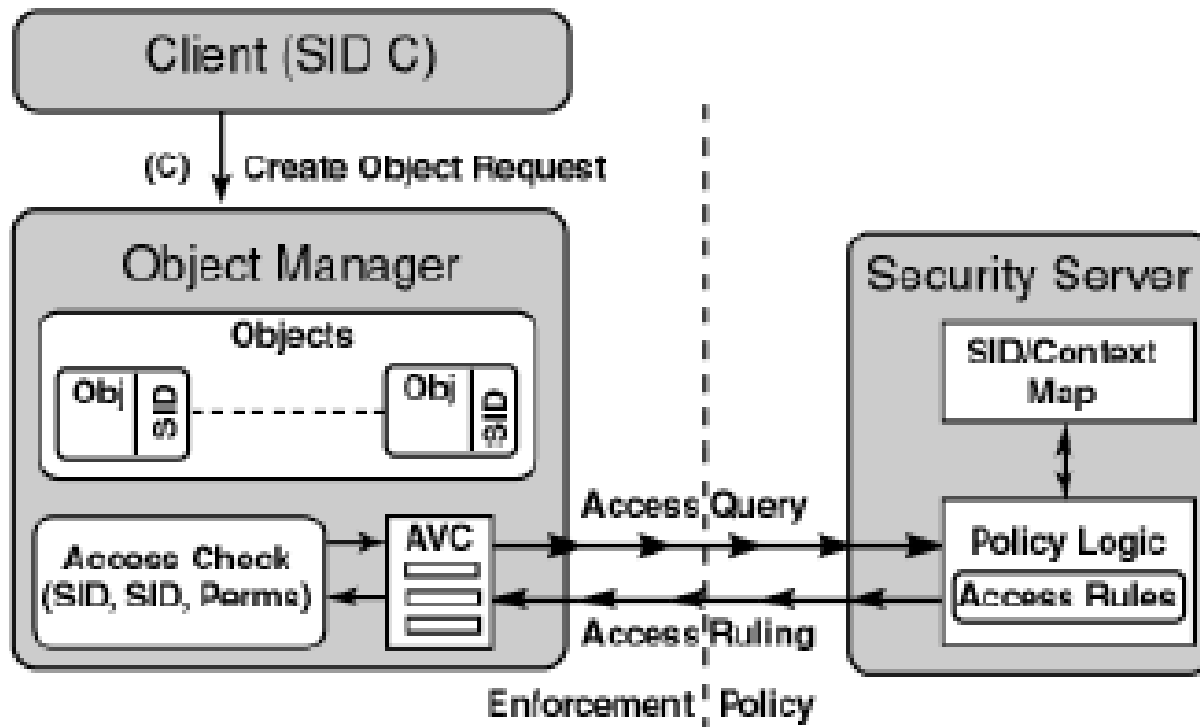
Enforcement with no information about the security policy

Security policy with no enforcement

SID and Context

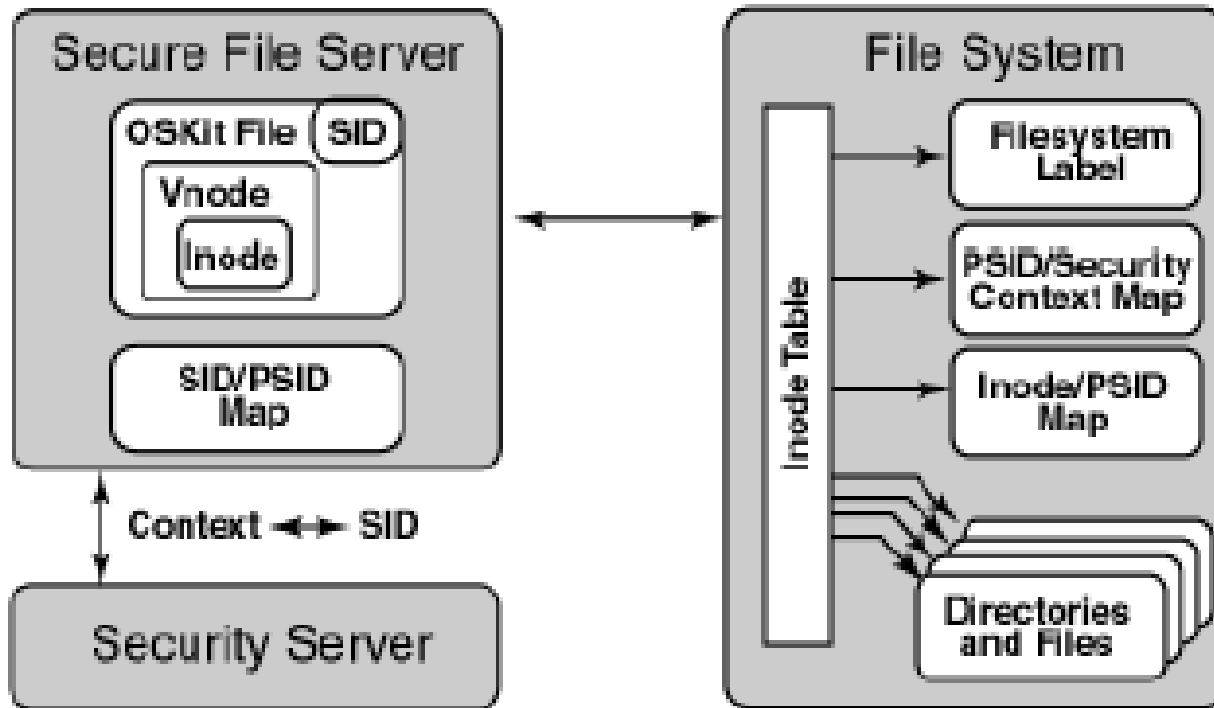


Caching

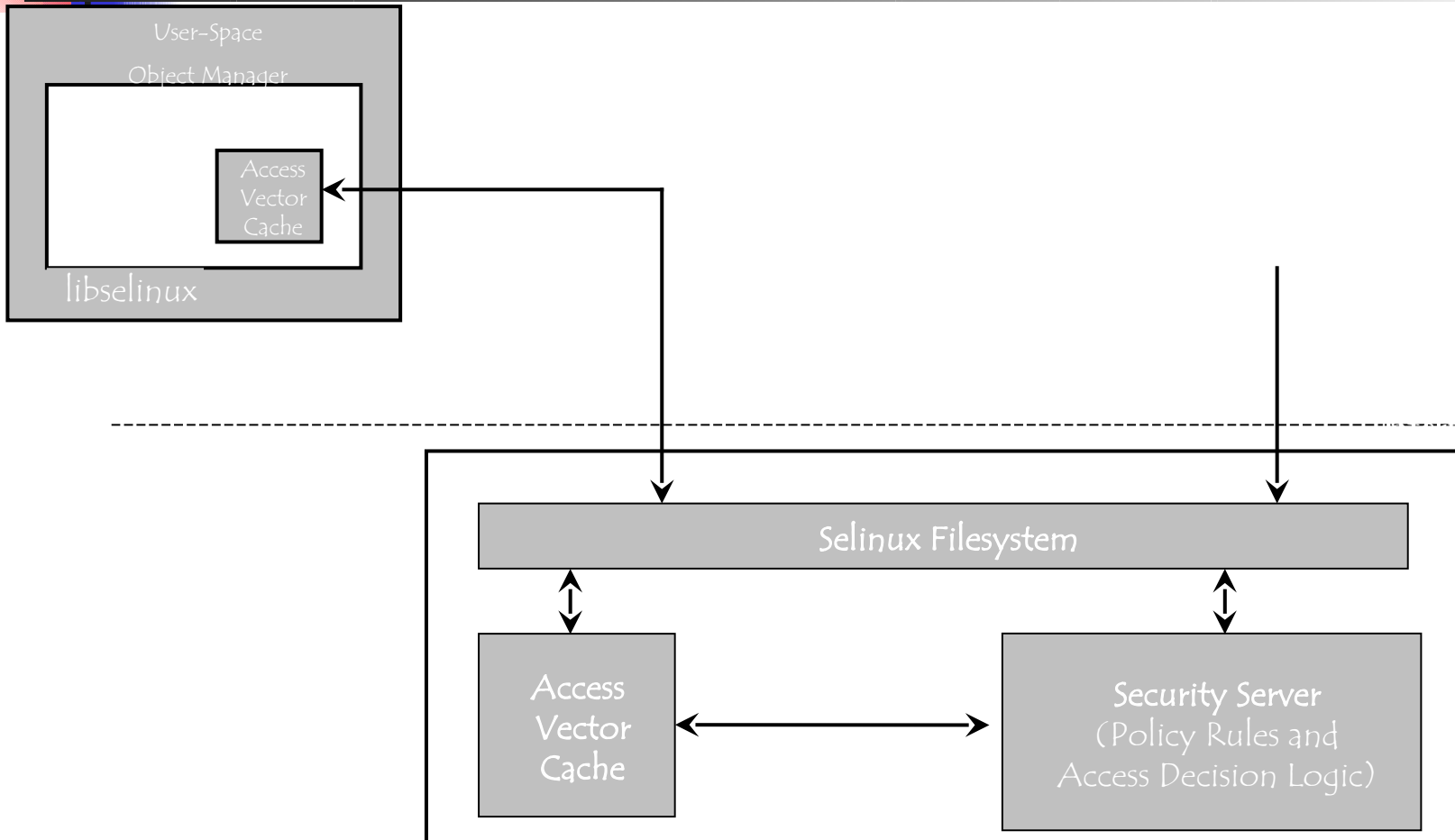


We reduce security to reduce the overhead

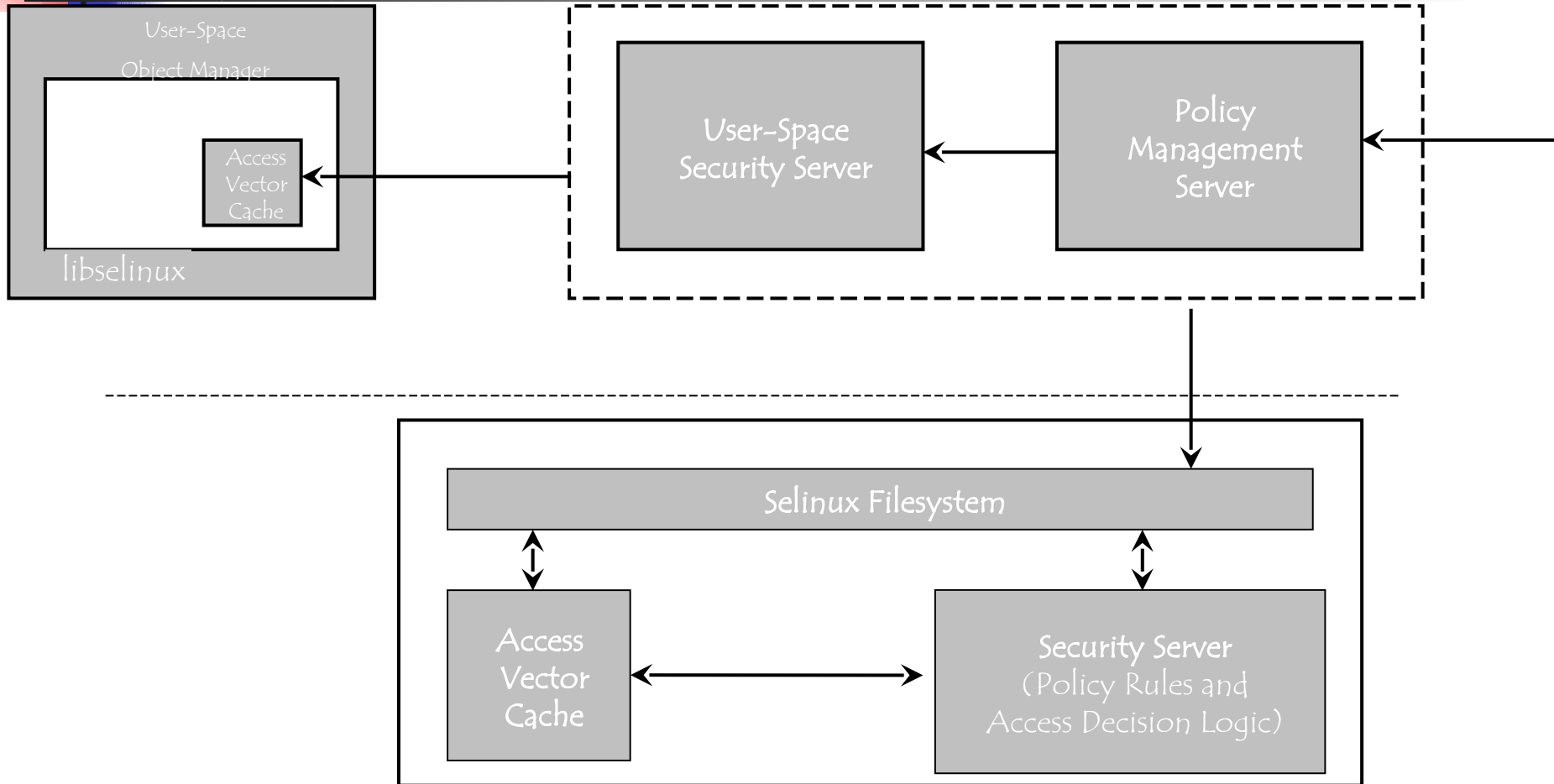
PSID



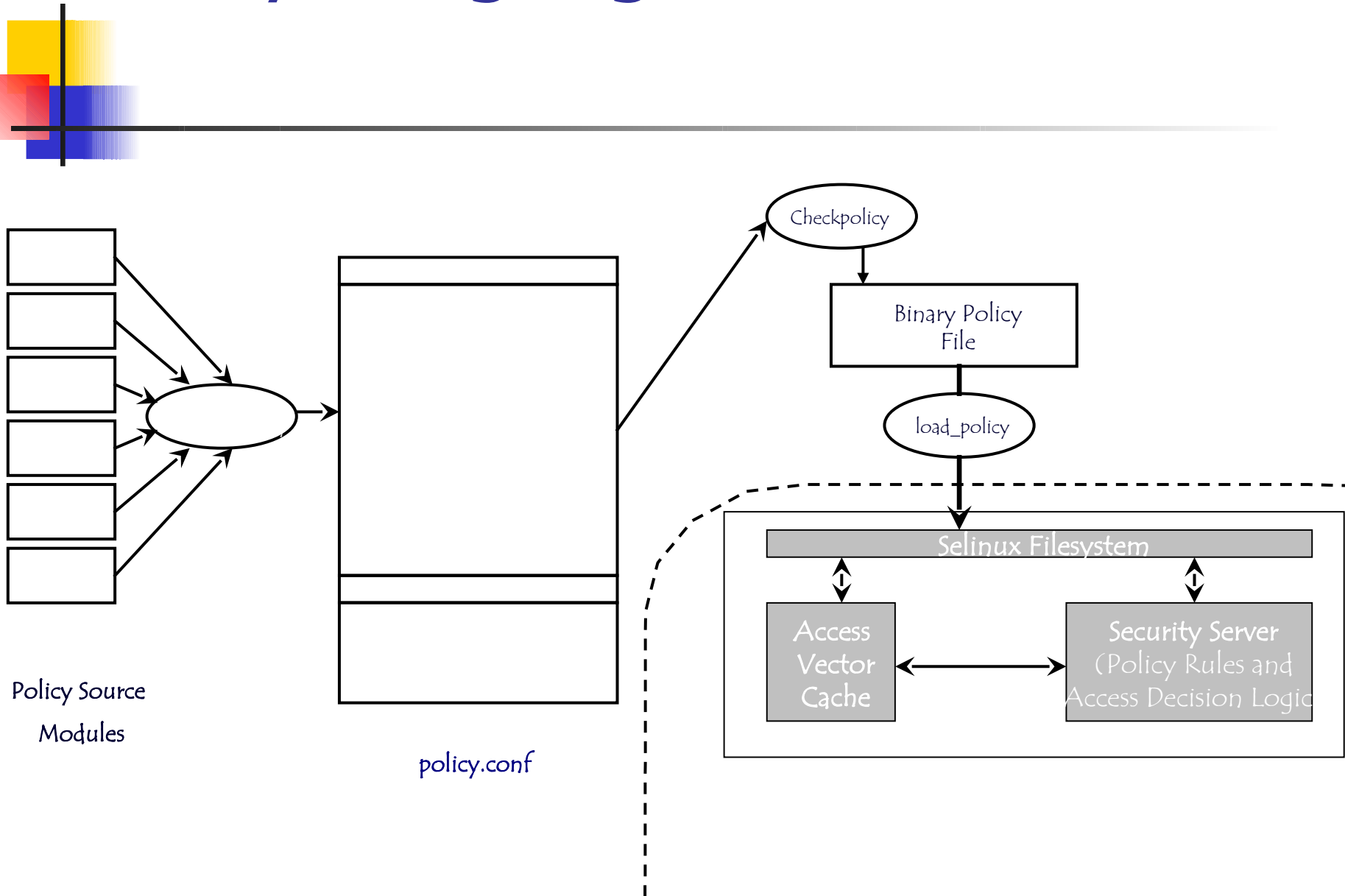
Userspace Object Managers



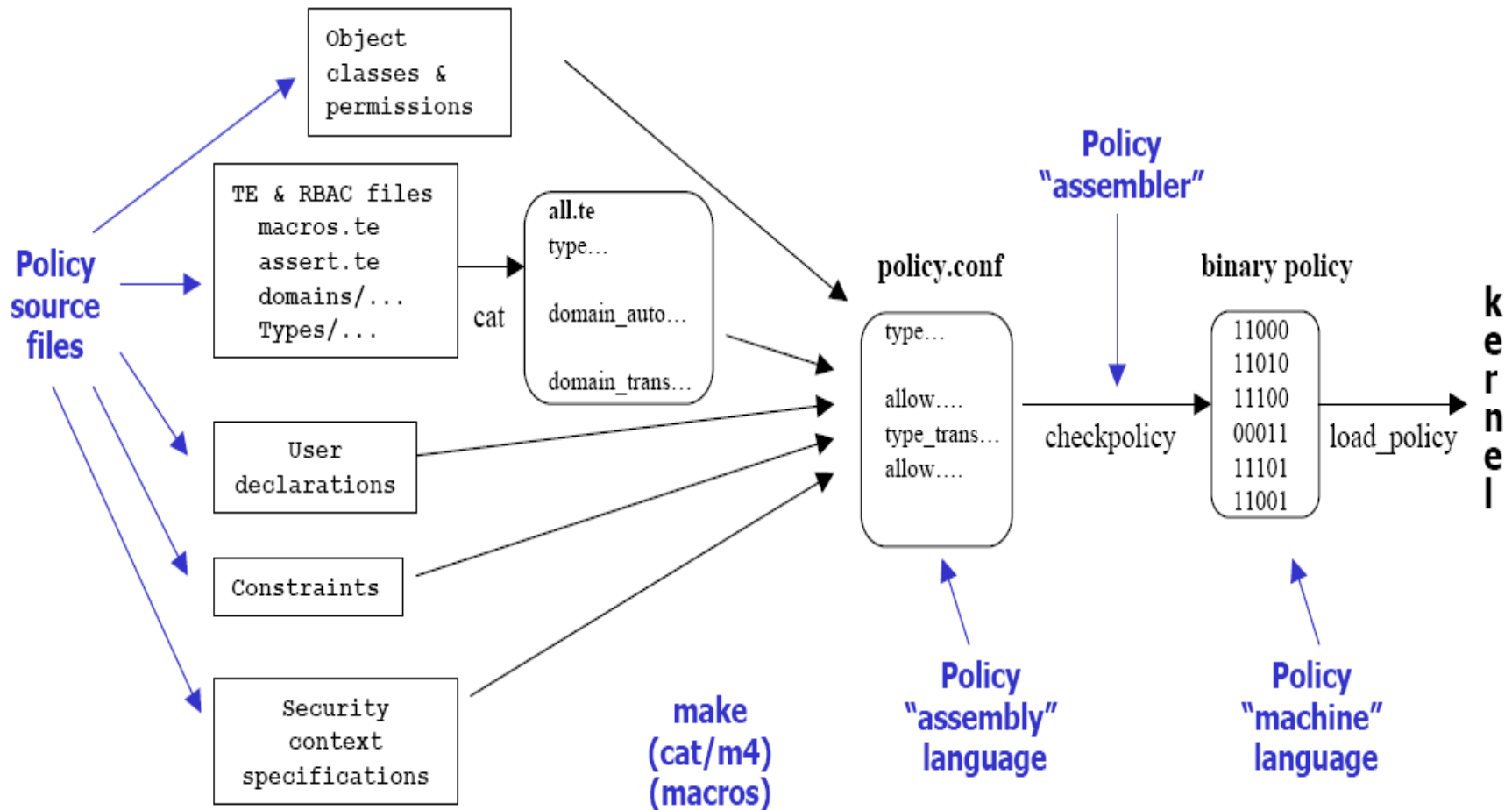
Policy Server




Policy Language



SELinux – Policy - Tools

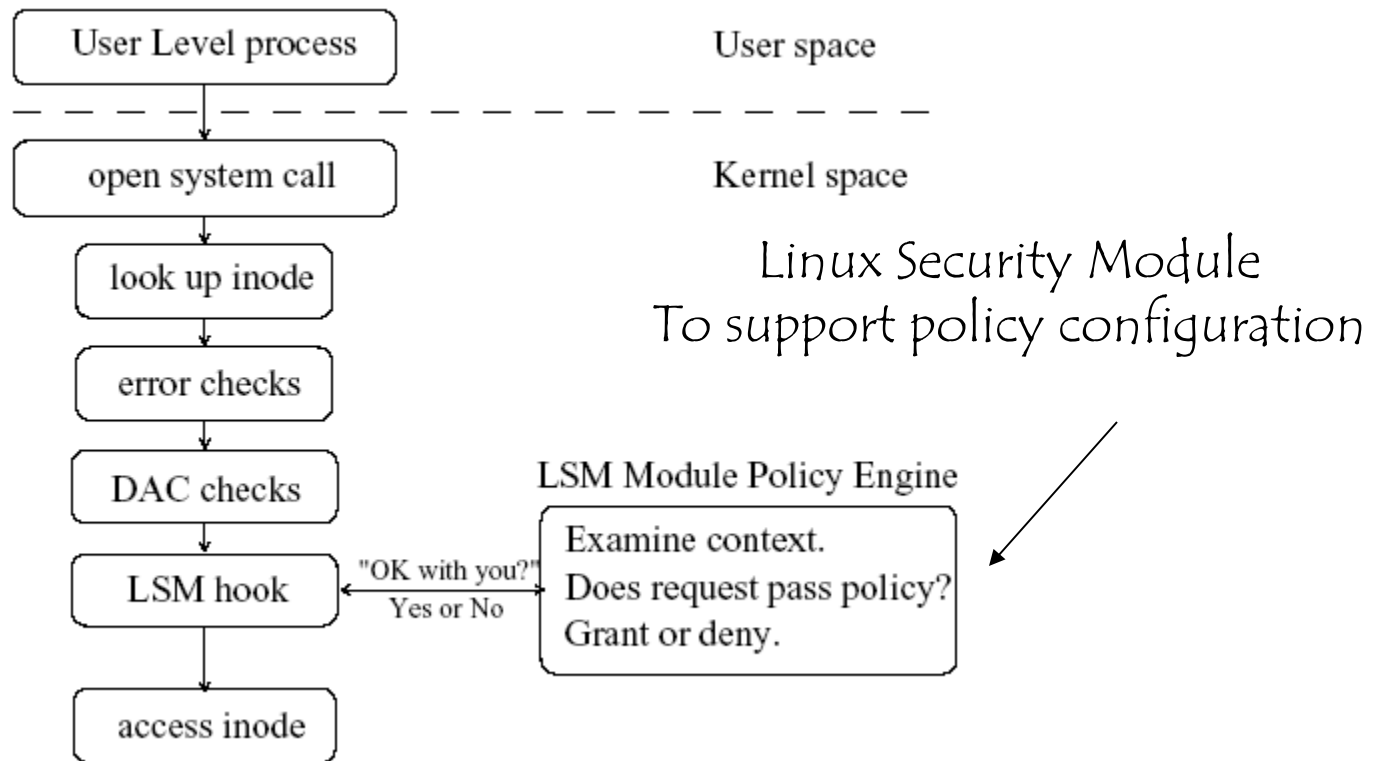




SELinux – Policy

- 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 - 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

The logo consists of a yellow square in the top-left, a red square in the bottom-left, and a blue square in the bottom-right, all overlapping. A black crosshair is centered over the intersection of the squares.

Webstone

Creates a load on a Web server by simulating multiple clients which can be thought of as users, Web browsers that retrieves files from a Web server. This simulation is carried out using multiple Web clients running on one or more computers. It is possible to run in excess of 100 simulated Web clients on a single computer.

In order to create large loads on a Web server, WebStone is able to distribute Web clients among client computers. The Webmaster is the program that controls all of the testing done by WebStone. It can be run on one of the client computers or on a separate computer. The Webmaster distributes the Web client software and test configuration files to the client computers. The Webmaster combines the performance results from all the clients into a single summary report.



AppArmor

AppArmor, supplements rather than replaces the default Discretionary Access Control (DAC). As such it's impossible to grant a process more privileges than it had in the first place.

SELinux attaches labels to all files, processes and objects and is therefore very flexible. However configuring SELinux is very complicated and requires a supported filesystem.

AppArmor on the other hand works using file paths and its configuration can be easily adapted.

The logo consists of a vertical black line on the left, a horizontal black line below it, and a grid of colored squares (yellow, red, blue) to the left of the vertical line.

AppArmor

AppArmor proactively protects the operating system and applications from external or internal threats and even zero-day attacks by enforcing a specific rule set on a per application basis.

Security policies completely define what system resources individual applications can access, and with what privileges. Access is denied by default if no profile says otherwise.

A few default policies are included with AppArmor and using a combination of advanced static analysis and learning-based tools, AppArmor policies for even very complex applications can be deployed successfully in a matter of hours.

Every breach of policy triggers a message in the system log and with real-time violation warnings popping up on the desktop.



Profile Modes

AppArmor operates in the following two types of profile modes:

Enforce

In the enforce mode, system begins enforcing the rules and report the violation attempts in syslog or auditd and operation will not be permitted.

Complain

In the complain mode, system doesn't enforce any rules. It will only log the violation attempts.



Profile

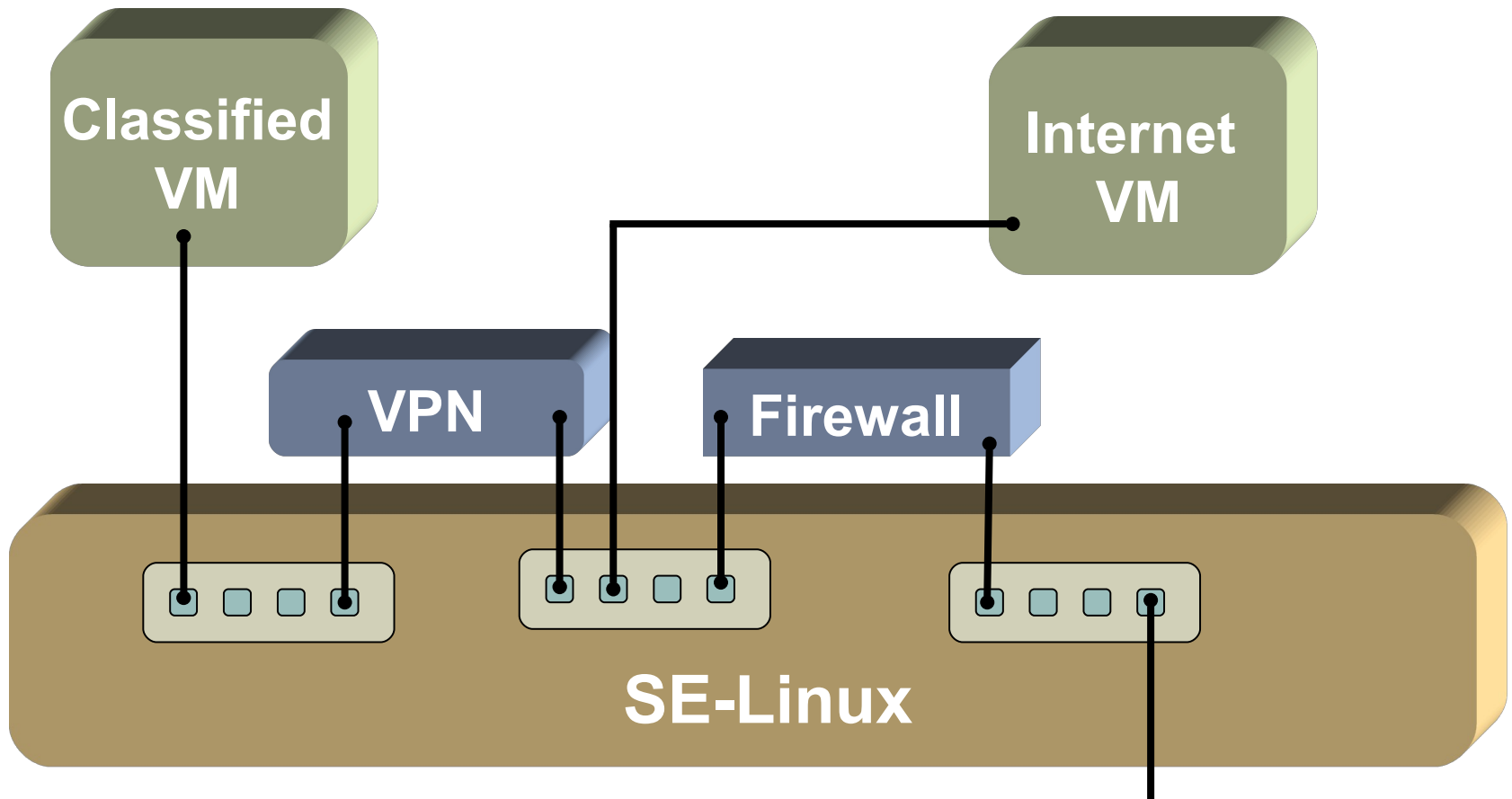
```
/usr/sbin/mysqld {  
  #include <abstractions/base>  
  ...  
  capability dac_override,  
  capability sys_resource,  
  capability setgid,  
  capability setuid,  
  network tcp,  
  /etc/hosts.allow r,  
  /etc/hosts.deny r,  
  /etc/mysql/*.pem r,  
  /etc/mysql/conf.d/ r,  
  /etc/mysql/conf.d/* rw,  
  /etc/mysql/*.cnf r,  
}
```

Path entries: This has information on which files the application is allowed to access.

Capability entries: determines the privileges a confined process is allowed to use.

Network entries: determines the connection-type. For example: tcp. For a packet-analyzer network can be raw or packet etc.

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



Attributes of VMware Virtual Machines

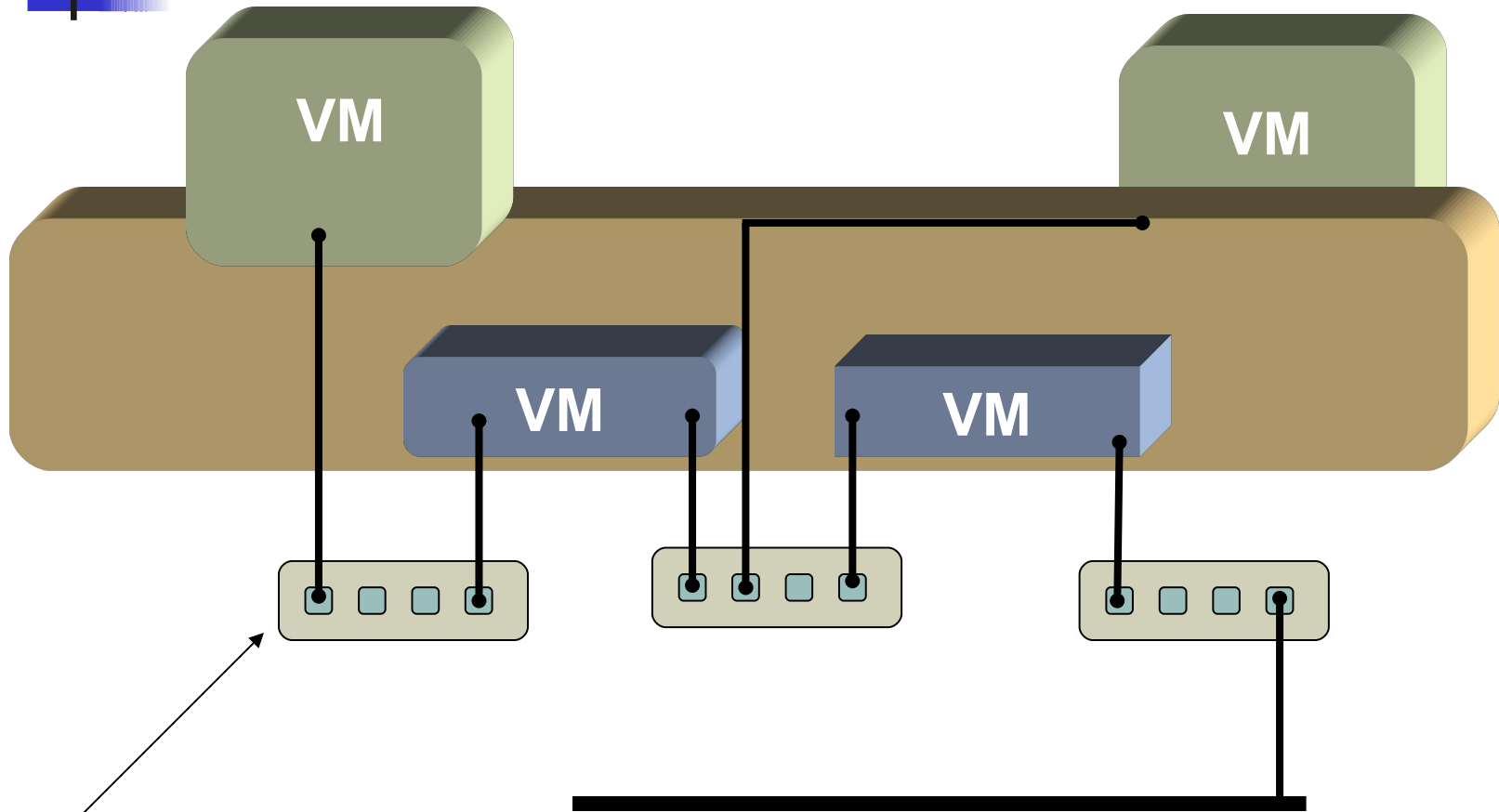
- Software compatibility
 - Runs pretty much all software
 - BIOS, OS, Apps, viruses, ...
- Near-native performance
- Encapsulation
 - Virtual machines are not tied to physical machines
- Consolidation
 - Run multiple VMs on a single desktop or server
- Isolation



Isolation at multiple levels

- Data :
 - Each VM is managed independently
 - Different OS, disks (→ files, registry), MAC address (→ IP address)
 - Data sharing is not possible = Each file system is a SE Linux file
- Faults:
 - Crashes are contained within a VM
- Performance
 - Guaranteed performance levels for individual VMs
- Security
 - No assumptions on the software running inside a VM.

Flexible Networking: VMnets



Virtual network devices

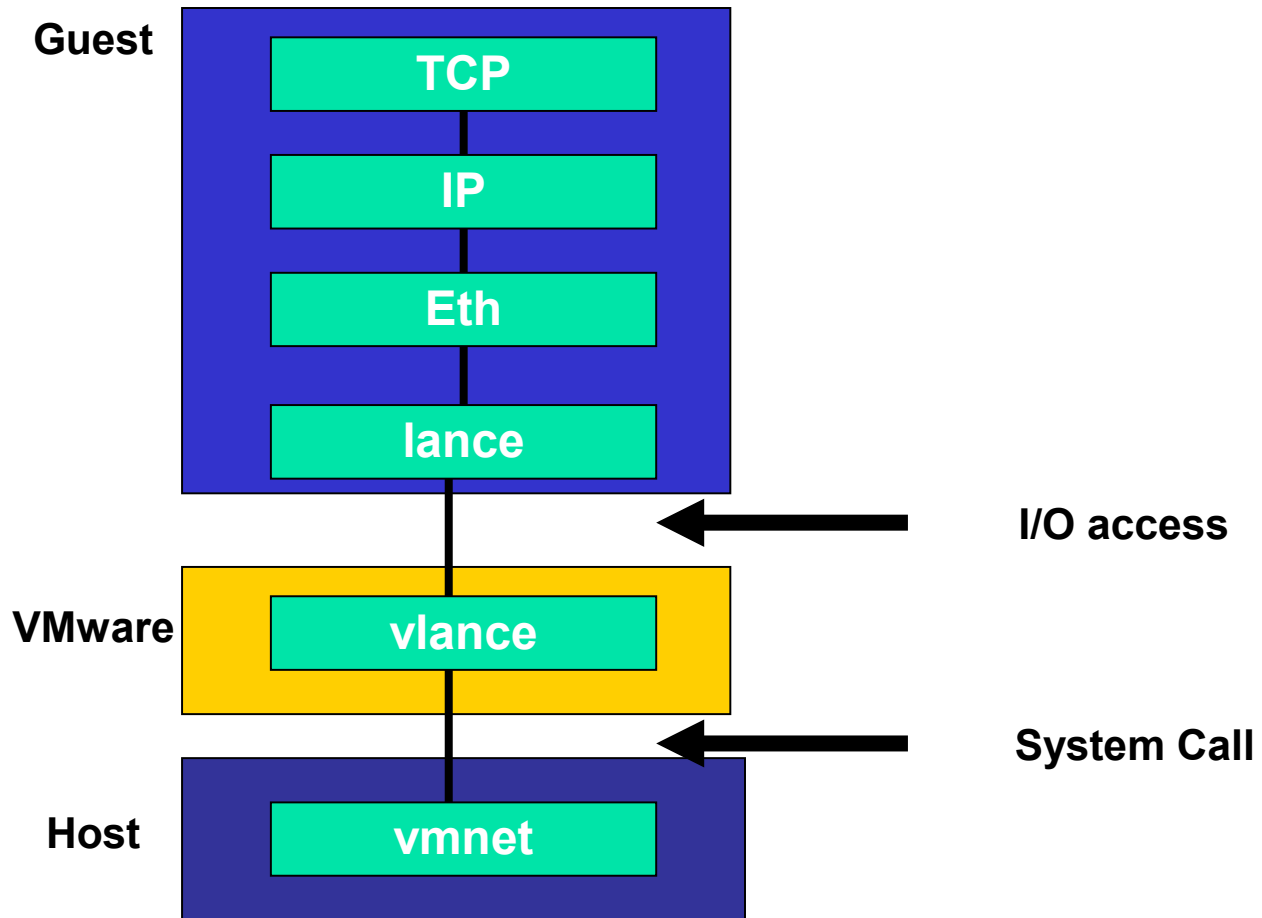
Physical LAN



Mandatory Interposition on all I/O

- 2 levels of mandatory I/O interposition (VM level and OS level)
- Guest cannot directly initiate I/O
 - → All guest I/O operations mediated by VMware
- VMware relies on the host for I/O access
 - → VMware process uses system calls to execute all I/O requests.
- Example: networking, disk I/O

I/O Interposition example: Networking





Processes running on the Host system

- “See without being seen” advantage
 - Very difficult within a computer
 - Possible on the host
- Observation points:
 - Networking (through vmnet)
 - Disk I/O (read and write)
 - Any other I/O
 - Physical Memory of the VM

Why NetTop?





Example: Access to classified networks

- Traditional tension : Security vs. Usability
 - Secure systems are not that usable
 - E.g: require some particular OS setups
 - Flexible systems are not that secure
 - Many documented examples
- Additional requirement:
 - Data cannot flow between networks of different classification
- Conventional solution:
 - Dedicate distinct computer for access to each network



Security of Isolation

- Q: How securely isolated are the virtual machines?
- A: Pretty well ...

