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 (egress filtering for function)
- Hide and protect critical information

Robust programming – check returned results

- Do not leak information before the user is authenticated (banner etc)
- Do not return too much information (yes or no without explaing why)
  - Do not say if the user or the password does not exist but just that the pair (user, password) does not exist
- Information useful to debug should be returned in log files in the node rather than in the user interface
- Avoid dependency on the user to prevent DOS attacks
  - Avoid synchronous communications,
  - If synchronous communications are required, introduce a sacrifical thread with asynchronous communication with the 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 run time support that has a low performance even if it does not enforce the constraints

A distinct perspective

- The CWE/SANS Top 25 Most Dangerous Programming Errors is a list of the most significant programming errors that can lead to serious software vulnerabilities.
- They occur frequently, are often easy to find, and easy to exploit.
- They are dangerous because they will frequently allow attackers to completely take over the software, steal data, or prevent the software from working at all.

The 25 errors

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

Attributes of each error

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

The 2011 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')

The 2011 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 2011 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

2020 List

[1]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	46.82
[2]	CWE-787	Out-of-bounds Write	46.17
[3]	CWE-20	Improper Input Validation	33.47
[4]	CWE-125	Out-of-bounds Read	26.50
[5]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	23.73
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	20.69
[7]	CWE-200	Exposure of Sensitive Information to an Unauthorized Actor	19.1
[8]	CWE-416	Use After Free	18.87
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	17.29
[10]	CWE-78	Improper Neutralization of Special Elements used in an OS Command	16.44

2020 List

[11]	CWE-190	Integer Overflow or Wraparound	15.81
[12]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	13.67
[13]	CWE-476	NULL Pointer Dereference	8.35
[14]	CWE-287	Improper Authentication	8.17
[15]	CWE-434	Unrestricted Upload of File with Dangerous Type	7.38
[16]	CWE-732	Incorrect Permission Assign. for Critical Res.	6.95
[17]	CWE-94	Improper Control of Generation of Code	6.53
[18]	CWE-522	Insufficiently Protected Credentials	5.49
[19]	CWE-611	Improper Restr. of XML External Entity Ref.	5.33

2020

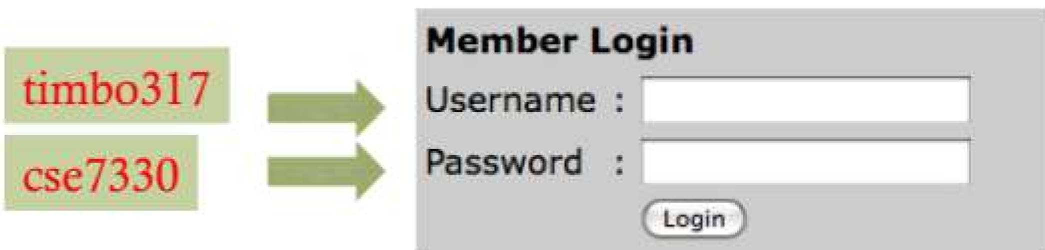
[20] CWE-798	Use of Hard-coded Credentials	5.19
[21] CWE-502	Deserialization of Untrusted Data	4.93
[22] CWE-269	Improper Privilege Management	4.87
[23] CWE-400	Uncontr. Resource Consumption	4.14
[24] CWE-306	Missing Auth. for Critical Function	3.85
[25] CWE-862	Missing Authorization	3.77

SQL Iniection

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();
  }
```

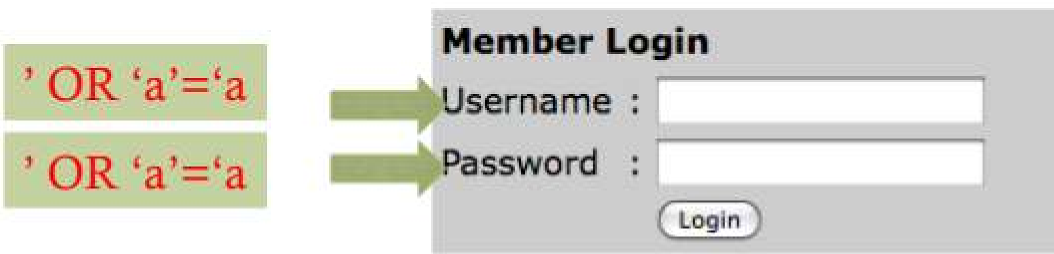
Hack Me! SQL Injection



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

SQL Injection

Hack Me! SQL Injection

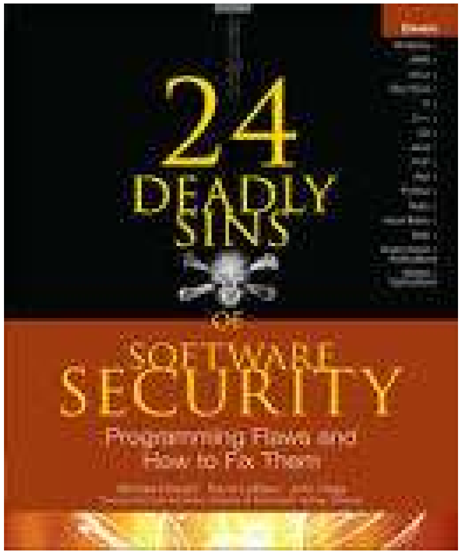


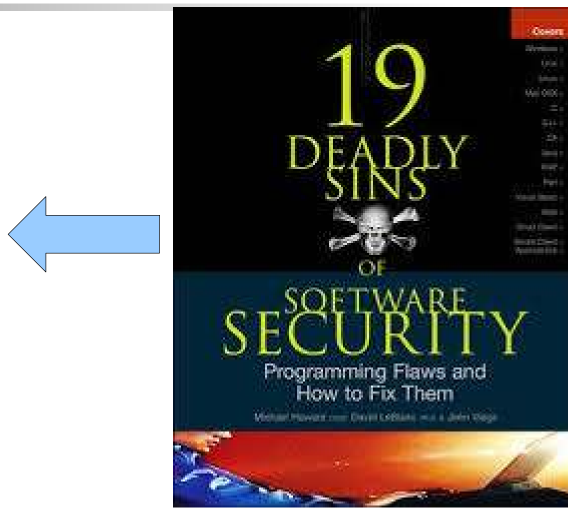
SELECT * FROM `login` WHERE `user`=‘’ OR ‘a’=‘a’ AND `pass`=‘’ OR ‘a’=‘a’

SQL Injection

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...

- 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;

Magic urls

- Do test all web input, including forms, with malicious input.
- Do not embed confidential data in any HTTP or HTML construct (the URL, cookie, or form) if the channel is not secured using SSL, TLS, or IPSec, or it uses application-level cryptographic defenses.
- Do not trust any data, confidential or not, in a web form, because malicious users can easily change the data to any value they like, regardless of SSL use or not.
- Do not think the application is safe just because you plan to use cryptography; attackers will attack the system in other ways = attackers ll not attempt to guess cryptographically random numbers they'll try to view them.

24 sins...

- Cryptographic Sins
  - Not Using Least Priveleges;
  - 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
  - P1 S&S economy of mechanism

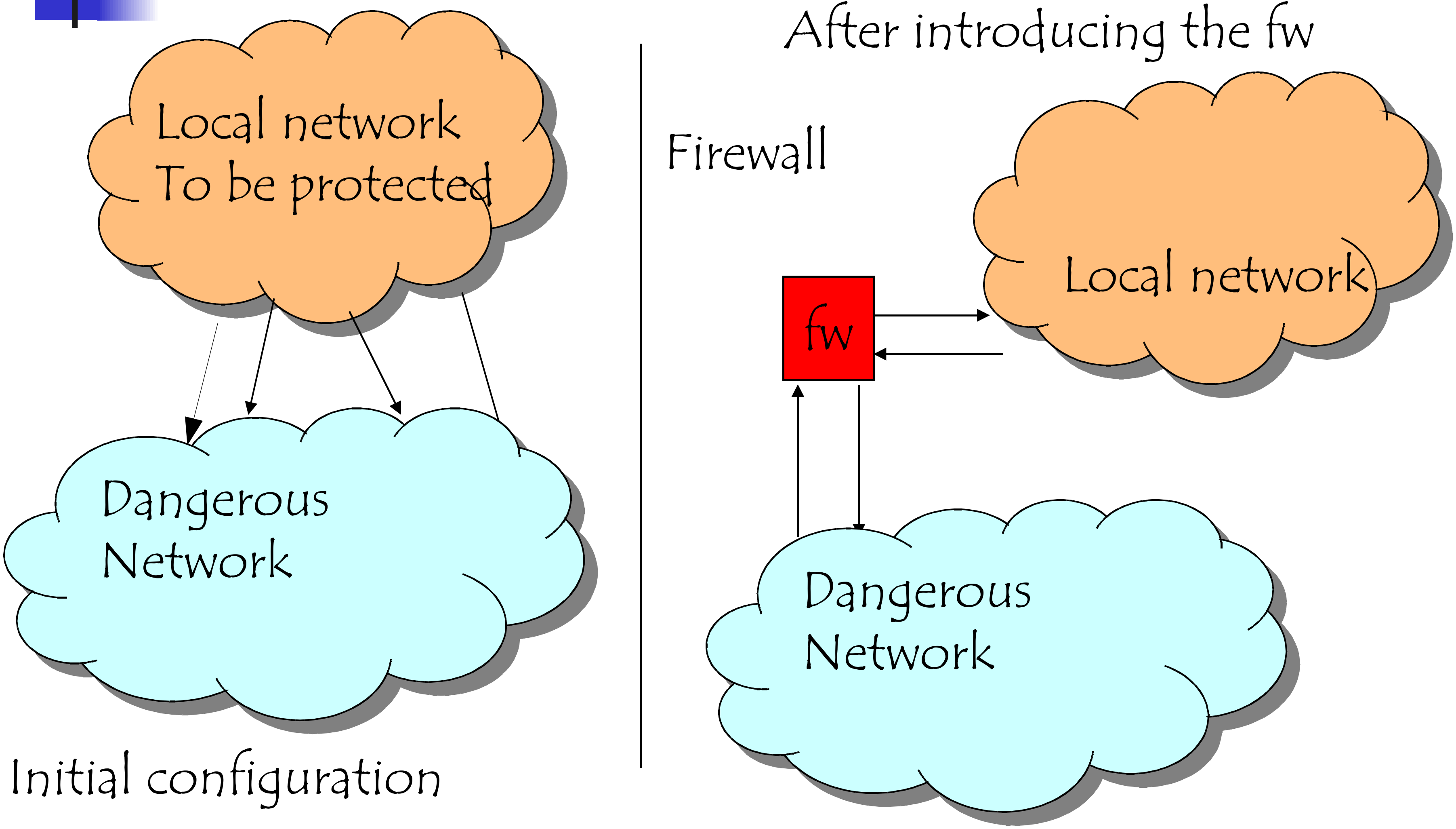
Countermeasures – Resist – Second Step

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 directly accessed from the less critical network
- It defends the most critical network from attacks originating in the less critical and less protected one at the expence of the bandwidth between the two networks

Introducing a Firewall

After introducing the fw

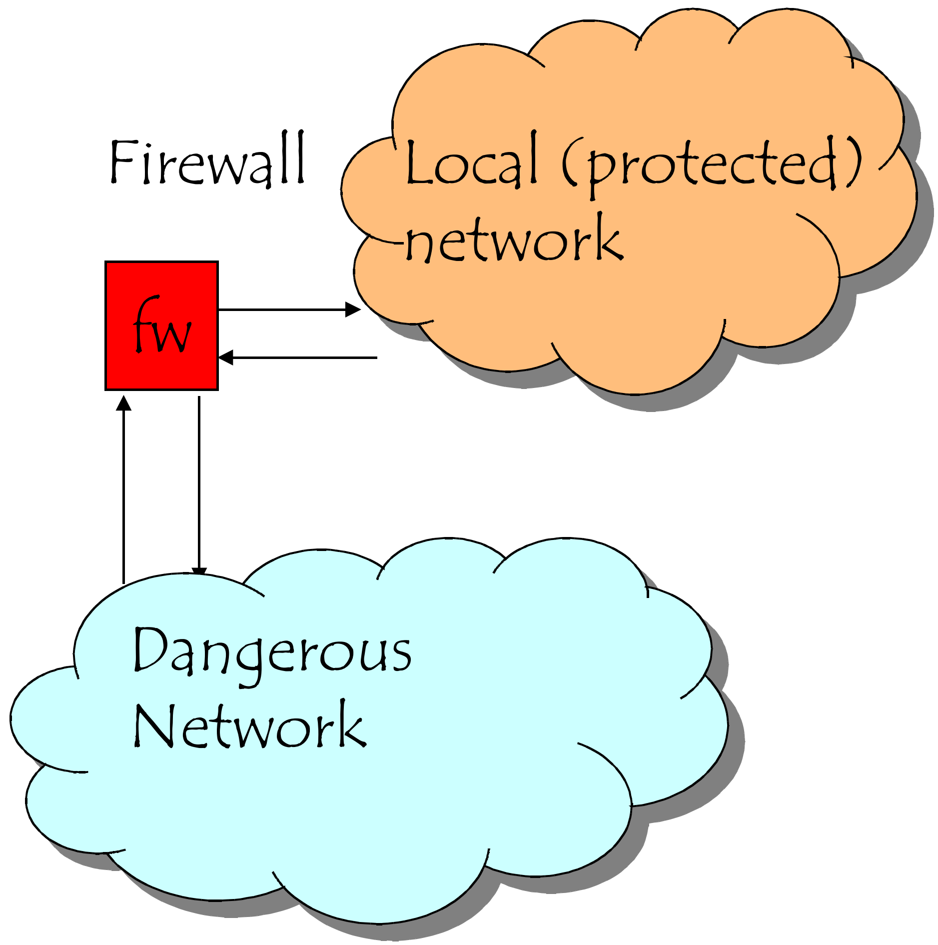


Initial configuration

Introducing a Firewall

Fw

- can filter all the traffic and attachments
- determines the protocols to interact with the protected network
- Determines the nodes in the protected network that can
  - receive messages from outside
  - send messages to the outside



A very simple firewall

- We have already seen a very simple firewall when discussing the acl for a router
- This is a simple form of firewalling where the protocols are TCP/IP/UDP and the security policy is focused on nodes rather than on the users running application on nodes

A very simple firewall

- ACL of input 1
  - 131.114.*.* → route
  - 131.4.5.6 → route
  - 131.4.*.* → drop

  Traffic from 131.114.*.* is routed and that from 131.4.*.* is dropped but that from 131.4.5.6

- ACL of output 1
  - 131.114.*.* → drop
  - 131.4.*.* → drop

  No address in 131.4.*.* and in 131.4.*.* can send traffic to the network connected to output 1

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 that enter or 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
    - protocols it does not know (unless a default deny strategy is adopted)

Introducing a Firewall

- The firewall behaviour fully depends upon the adopted security policy = A firewall is useless without a security policy that drive the filtering of the information flowing across the two network
- 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 two properties

- 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 = firewall robustness =

robustness enabled by the controls

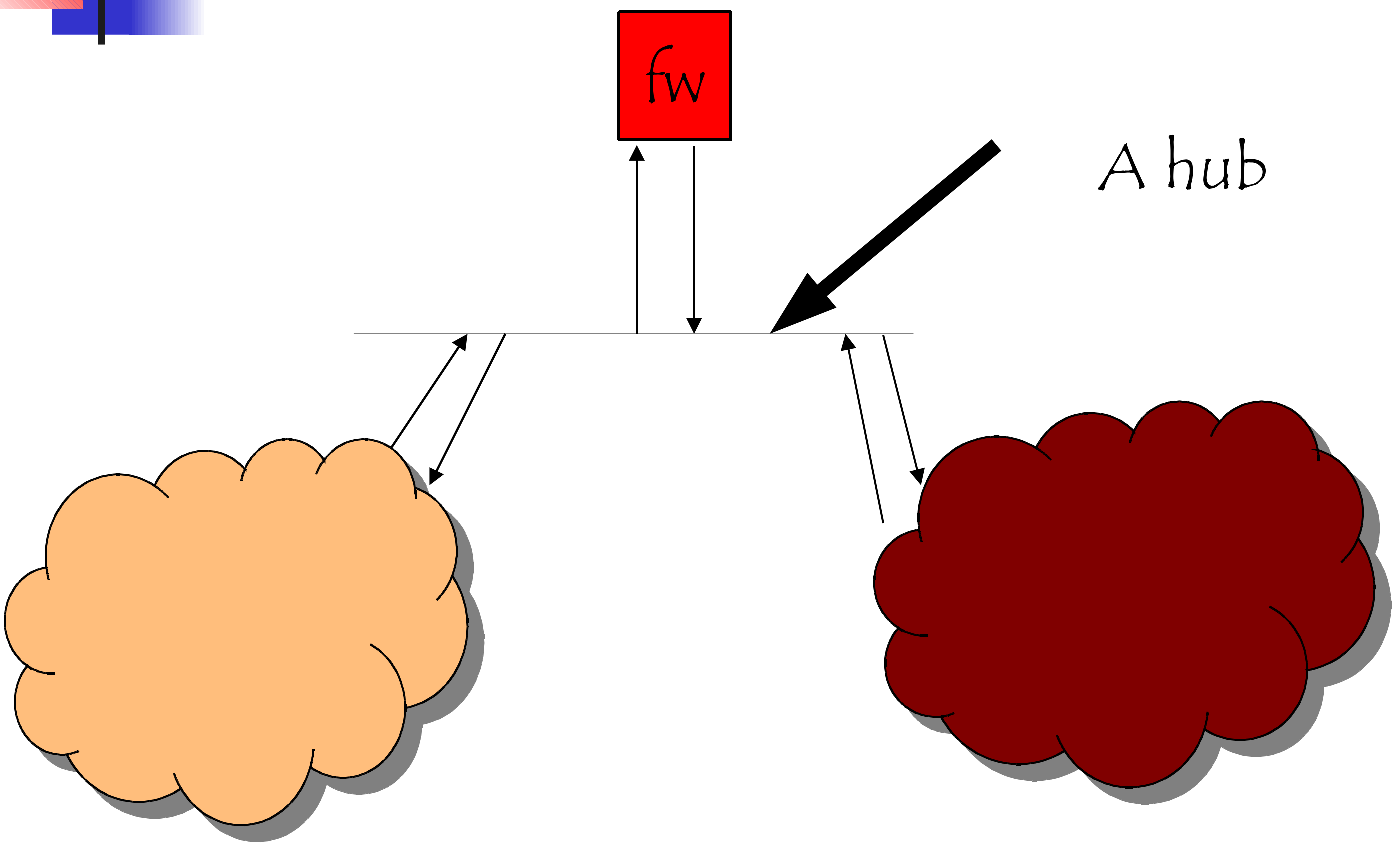
\+

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

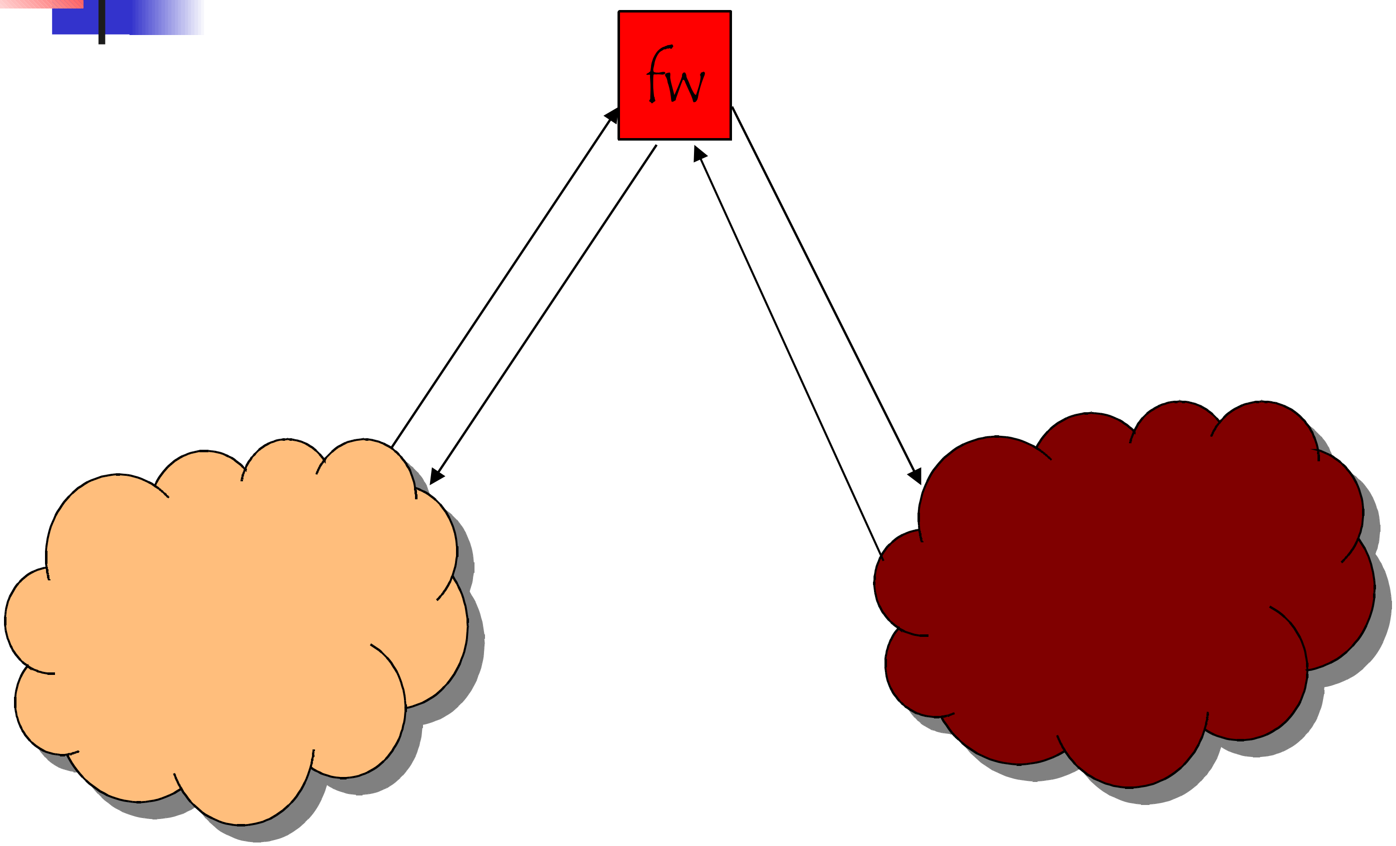
Some architectures - 1



Routing tables
To filter
traffic

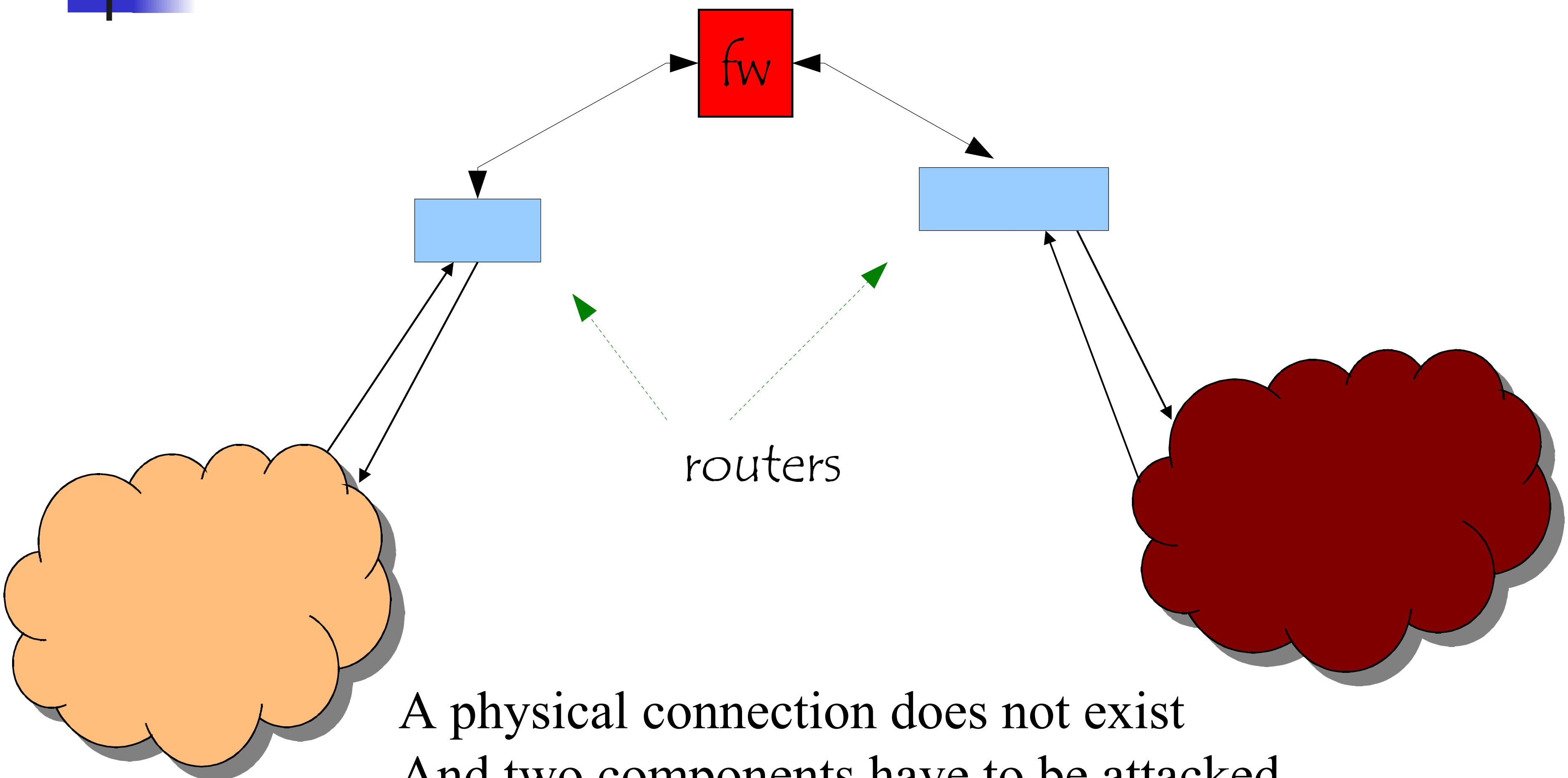
A physical connection exist

Some architectures - 2



A physical connection does not exist

Some architectures - 3

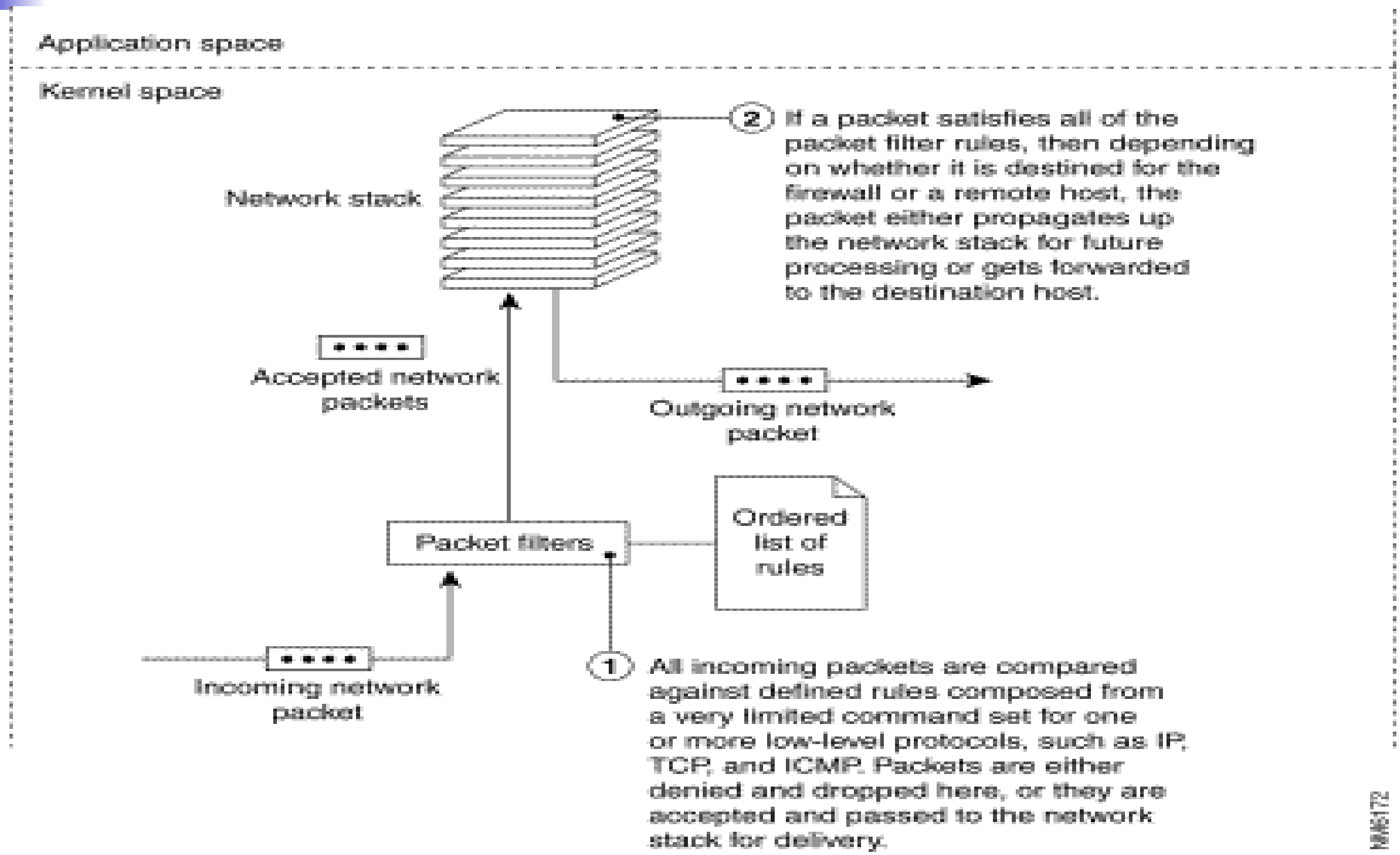


A physical connection does not exist
And two components have to be attacked

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



Firewalls – Packet Filters

- Simplest firewall using transport-layer info
  - 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

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
  - Better: stateful filtering knows outgoing requests

Usage of Packet Filters

- Filtering with incoming or outgoing interfaces
  - Ingress filtering of spoofed IP addresses
  - Egress filtering (undetected attack)
- 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 legal packets in terms of logical expressions on packet fields
- Rewrite pattern matching expressions in syntax supported by your vendor
- General rules – default deny
  - 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	*	*	*	*	*default*

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	*	*we don't trust these people*
allow	OUR-GW	25	*	*	*connection to our SMTP port*

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	*connection to their SMTP port*

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		*our packets to their SMTP port*
allow	*	25	*	*	ACK	*their replies*

- We include the ACK bit in our checks
- 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

Packet filtering firewall at 192.168.1. protecting 192.168.1.0

	Source Address	Source Port	Destination Address	Destination Port	Action	Description
1	Any	Any	192.168.1.0	> 1023	Allow	Rule to allow return TCP Connections to internal subnet
2	192.168.1.1	Any	Any	Any	Deny	Prevent Firewall system itself from directly connecting to anything
3	Any	Any	192.168.1.1	Any	Deny	Prevent External users from directly accessing the Firewall system.
4	192.168.1.0	Any	Any	Any	Allow	Internal Users can access External servers
5	Any	Any	192.168.1.2	SMTP	Allow	Allow External Users to send email in
6	Any	Any	192.168.1.3	HTTP	Allow	Allow External Users to access WWW server
7	Any	Any	Any	Any	Deny	"Catch-All" Rule - Everything not previously allowed is explicitly denied

Security & Performance of Packet Filters

Tiny fragment attacks

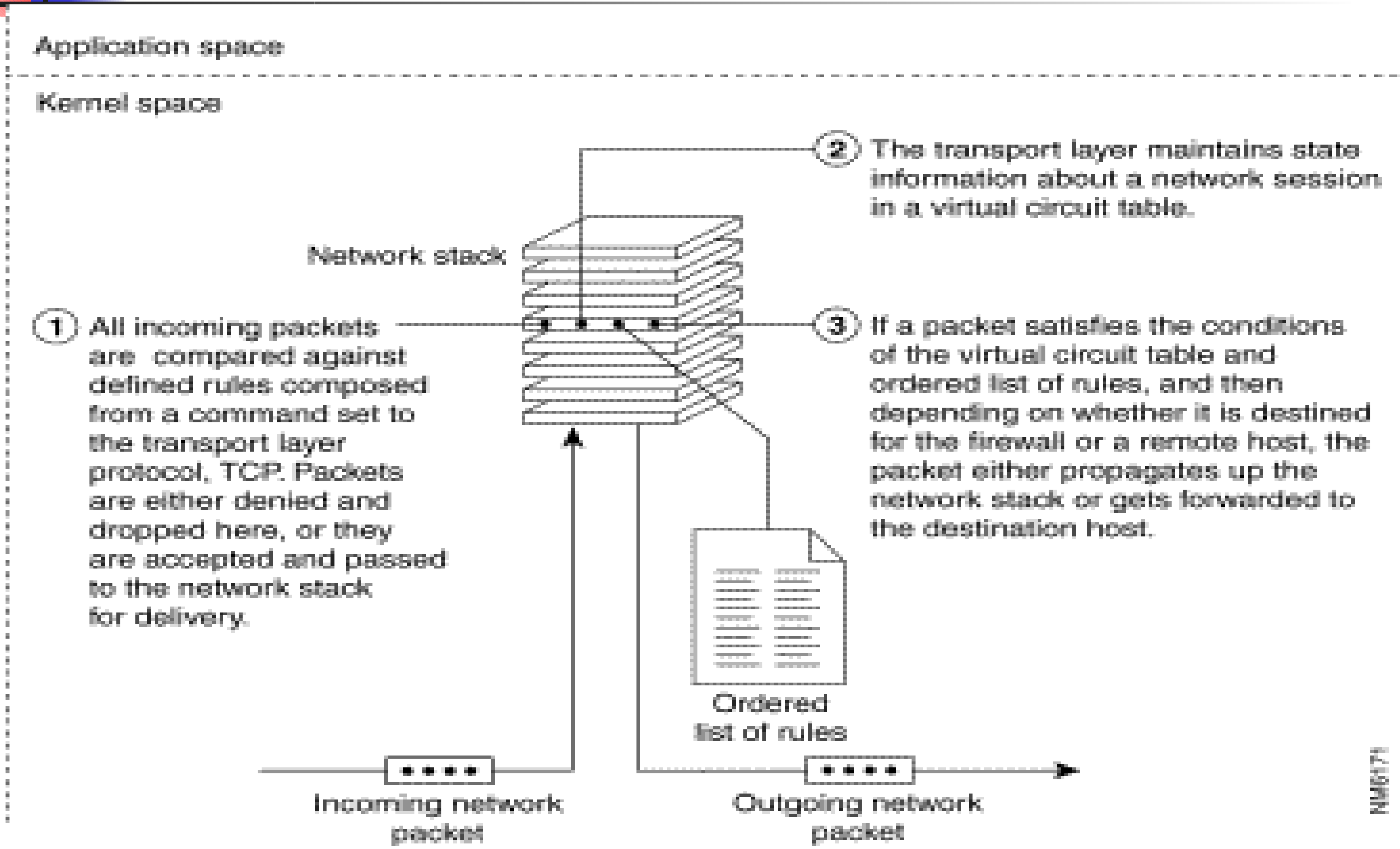
- Split TCP header info over several tiny packets
- Either discard or reassemble before check

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

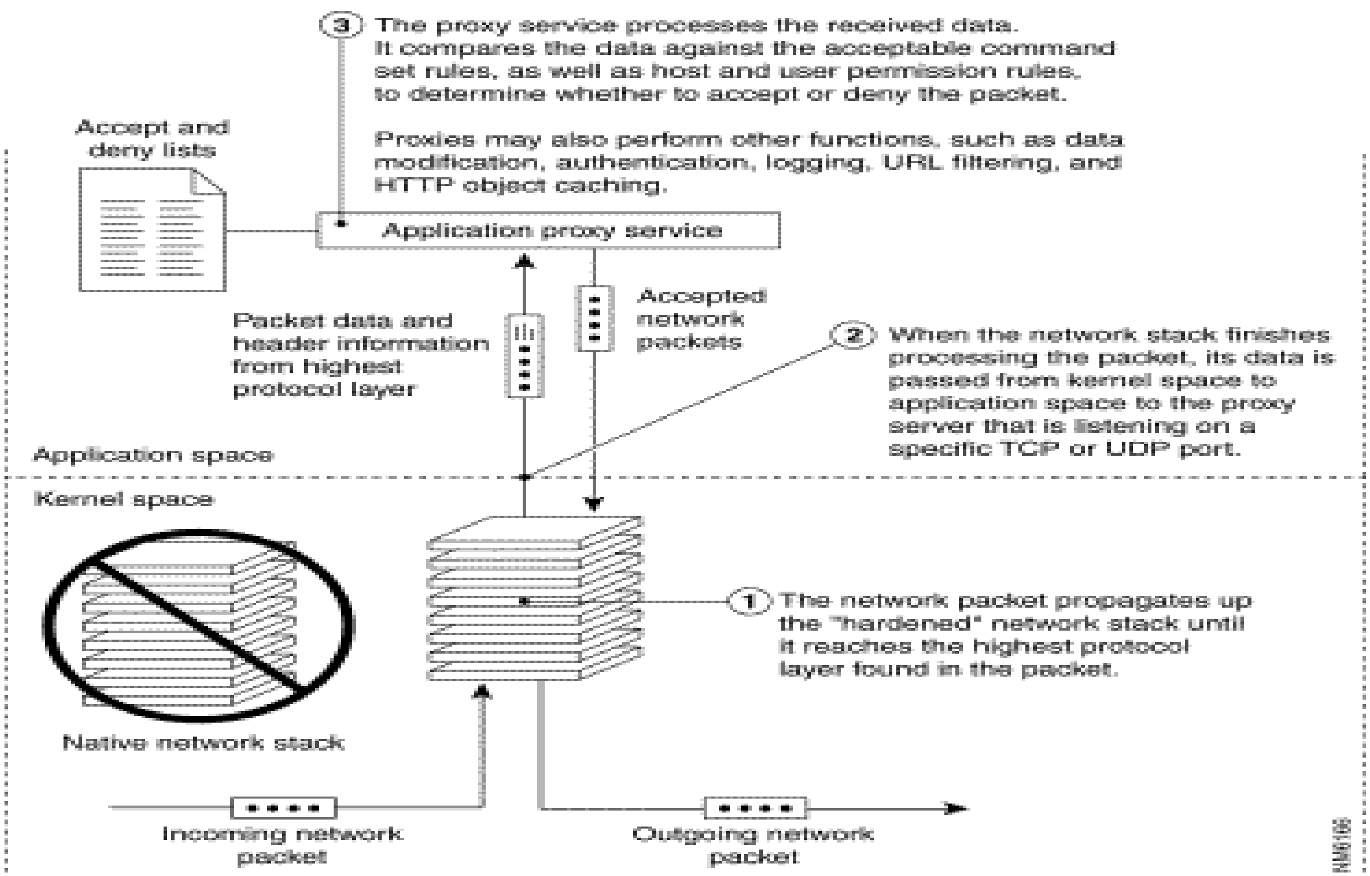
Circuit level – virtual circuit table



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 can detect bogus packets out of context

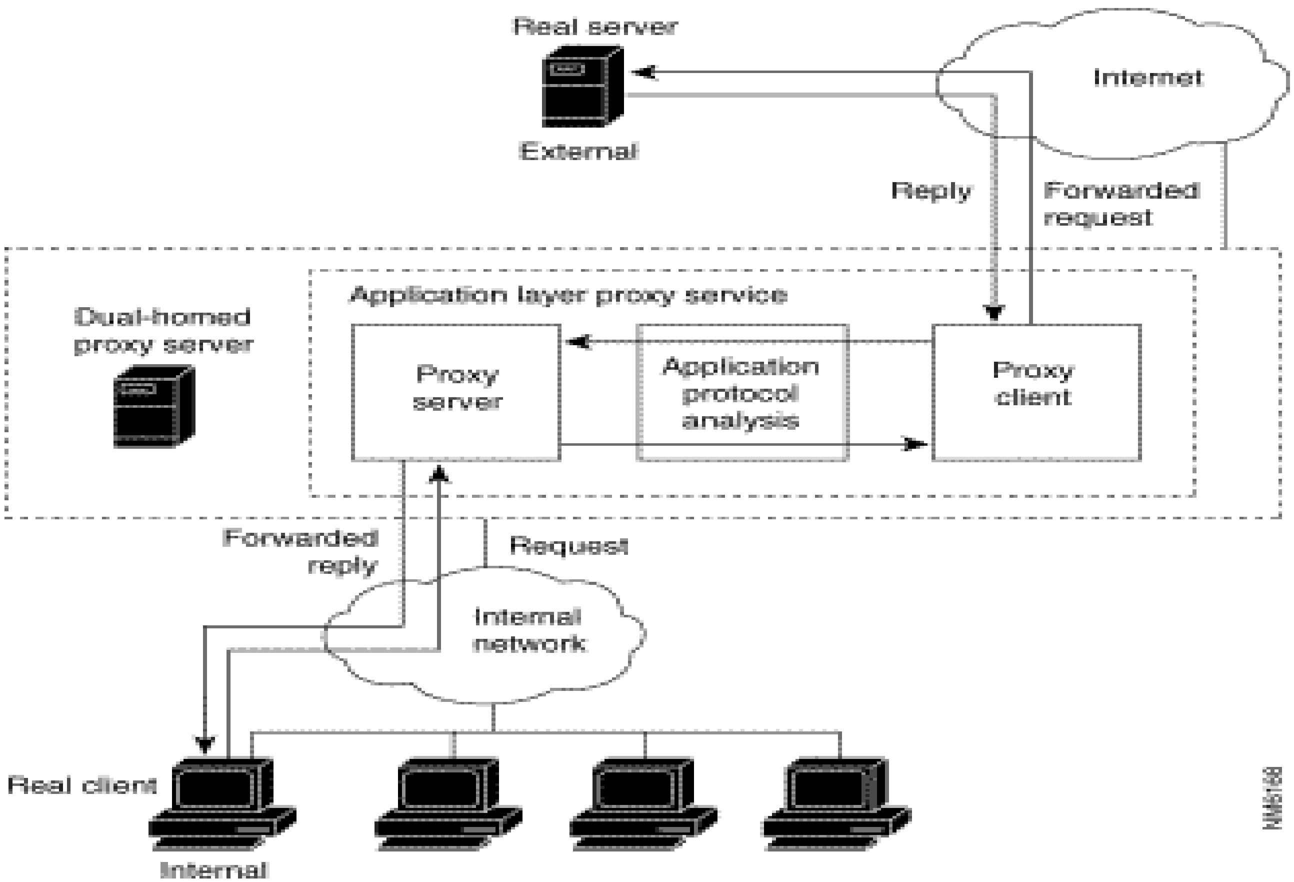
Application Level -Proxy



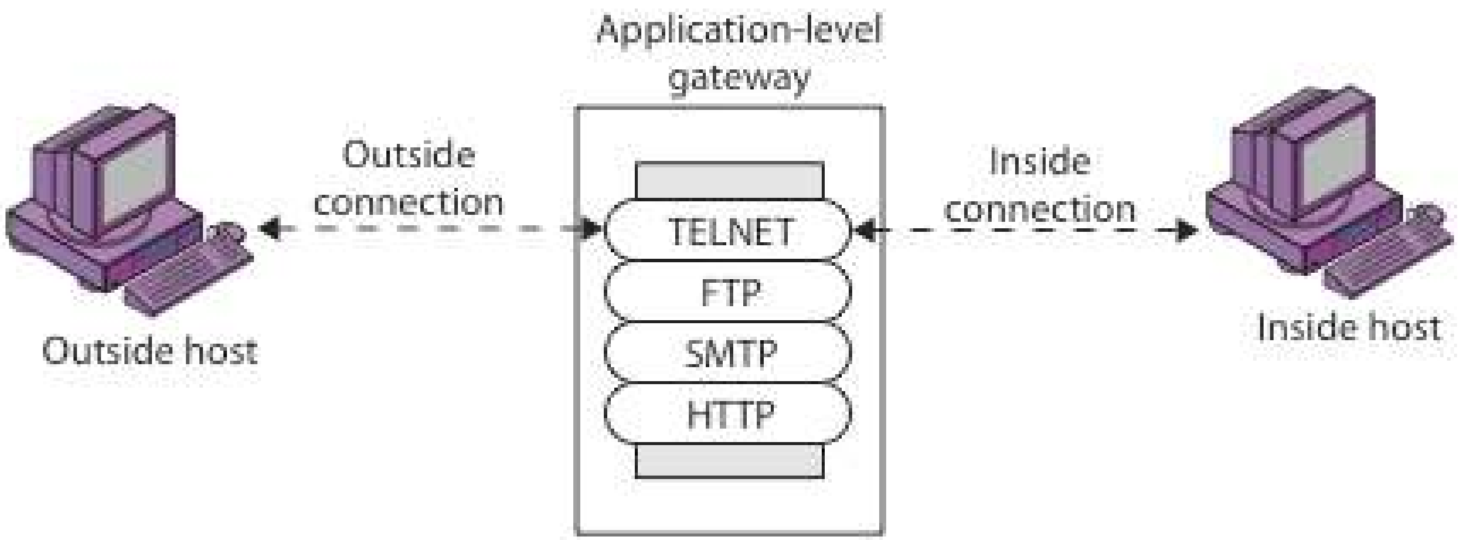
Application-Level Filtering

- 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

Proxy service



Firewalls - Application Level Gateway (or Proxy)

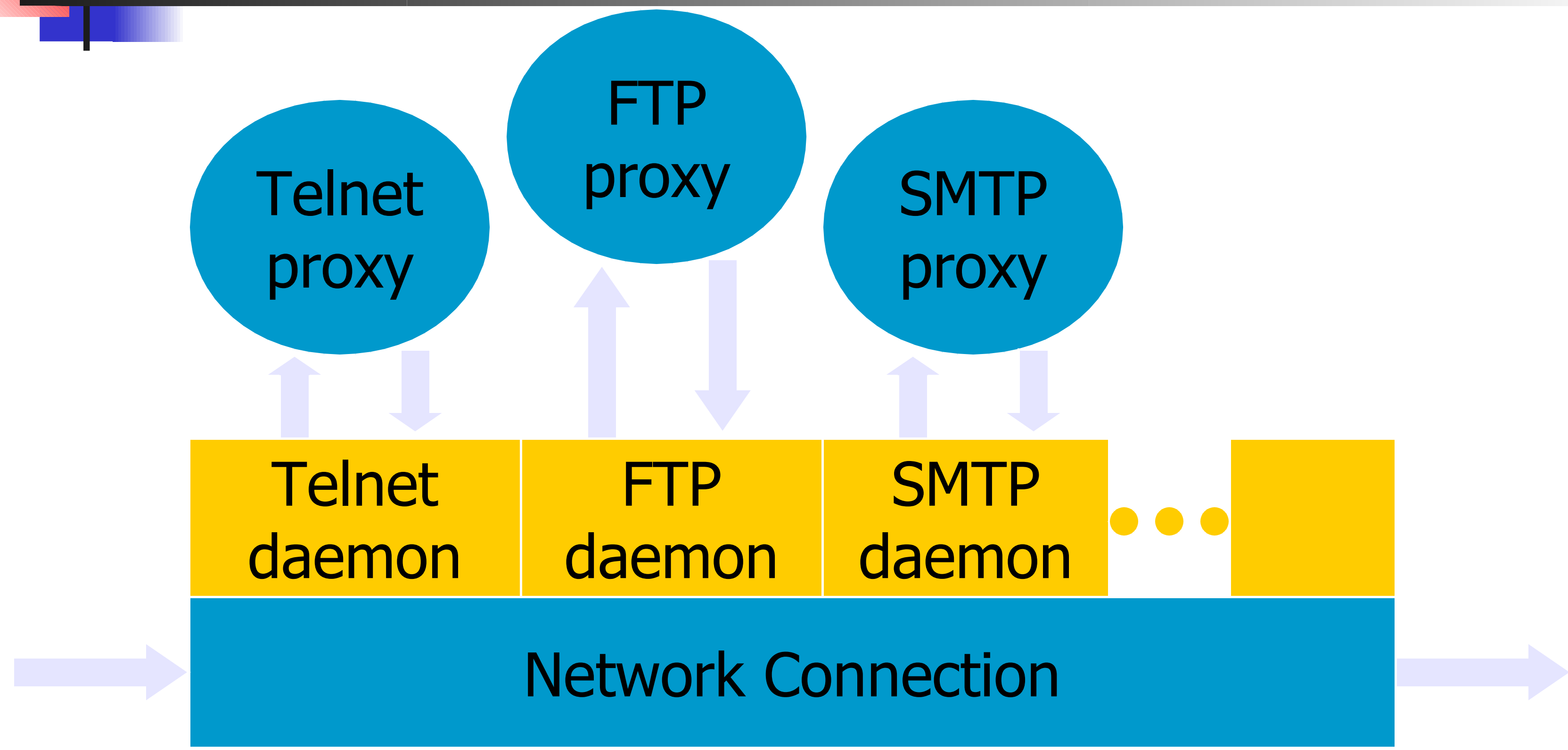


(b) Application-level gateway

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

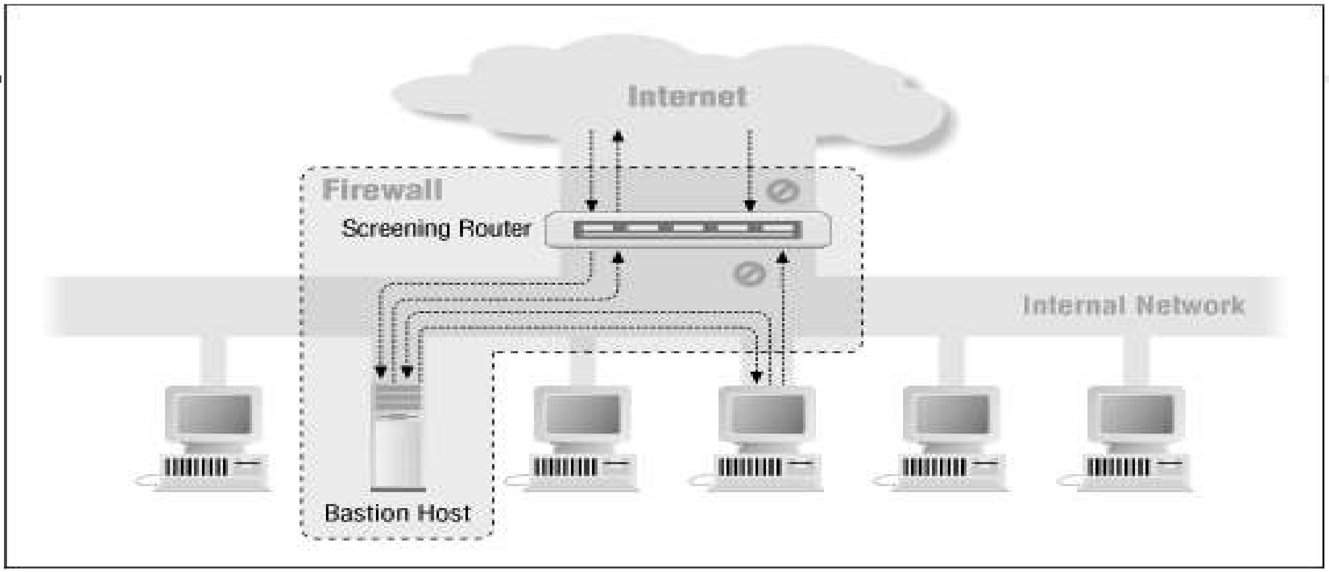
App-level Firewall Architecture



Daemon spawns proxy only when communication is detected to minimize load

Further Firewall Architectures

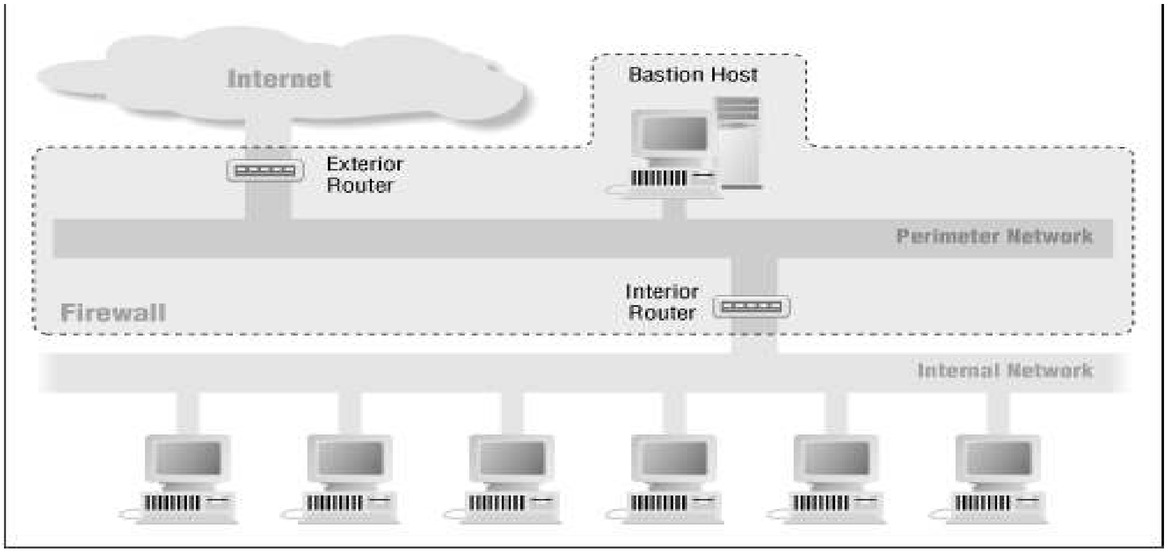
Screening router + bastion host



- The bastion host is the only system on the internal network the Internet can open connections to (ie to deliver email).
- Only certain types of connections are allowed. Any external system trying to access internal systems or services has to connect to this host.

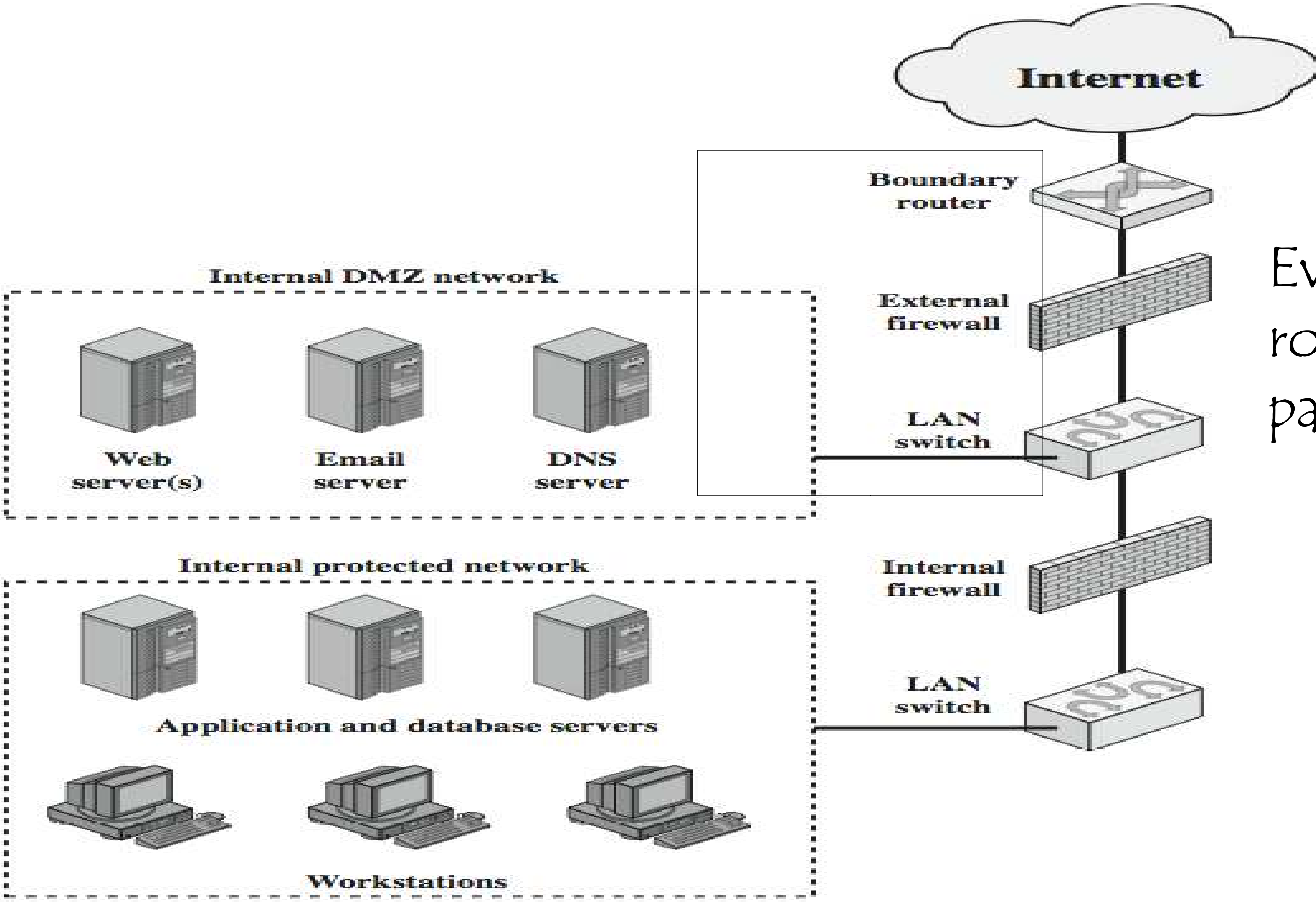
•The bastion host thus needs to maintain a high level of host security.

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.

De Militarized Zone – Layered protection = defence in depth



Even a simple router with packer filtering

DMZ – Advantages

- Three layers of protection that segregate the protected network. To penetrate the protected network, the intruder must crack :
  - the outside firewall router,
  - the bastion firewall
  - the inside firewall router devices.
- The outside router advertises the DMZ network only to the Internet so that systems on the Internet do not have routes to the protected private network. Hence the private network is "invisible," and the Internet knows only selected DMZ systems
- The inside router advertises the DMZ network only to the private network, private network systems have no direct routes to the Internet.
- Since the DMZ is a different network from the private one, a Network Address Translator (NAT) can be installed on the bastion host to avoid renumbering or re-subneting the private network.

Firewall in short - 1

Packet Filter Firewall

- controls the network access by analyzing the outgoing and incoming packets.
- It lets a packet pass or block its way by comparing it with pre-established criteria like IP addresses, packet type, port number, etc.
- This technique is suitable for small networks but gets complex for larger networks.
- This firewall can neither tackle the attacks that use application layers vulnerabilities nor can fight against spoofing attacks.

Firewall in short - 2

Stateful Packet Inspection (SPI) aka dynamic packet filtering

- A powerful firewall solution which examines traffic streams from end to end.
- These smart and fast firewalls use an intelligent way to ward off the unauthorized traffic by analyzing the packet headers and inspecting the state of the packets
- This firewall works at the network layer in the OSI model and are more secured than packet filtering firewalls.

Firewall in short - 3

Proxy Server Firewalls aka application level gateways, Proxy Server

- The most secured type of firewall that protect the network by filtering messages at the application layer.
- It masks your IP address and limit traffic types.
- It provides a complete and protocol-aware security analysis for the protocols they support.
- Proxy Servers can result in network performance improvements.

Countermeasures – Resist & Recovery

Defense in Depth (DiD)

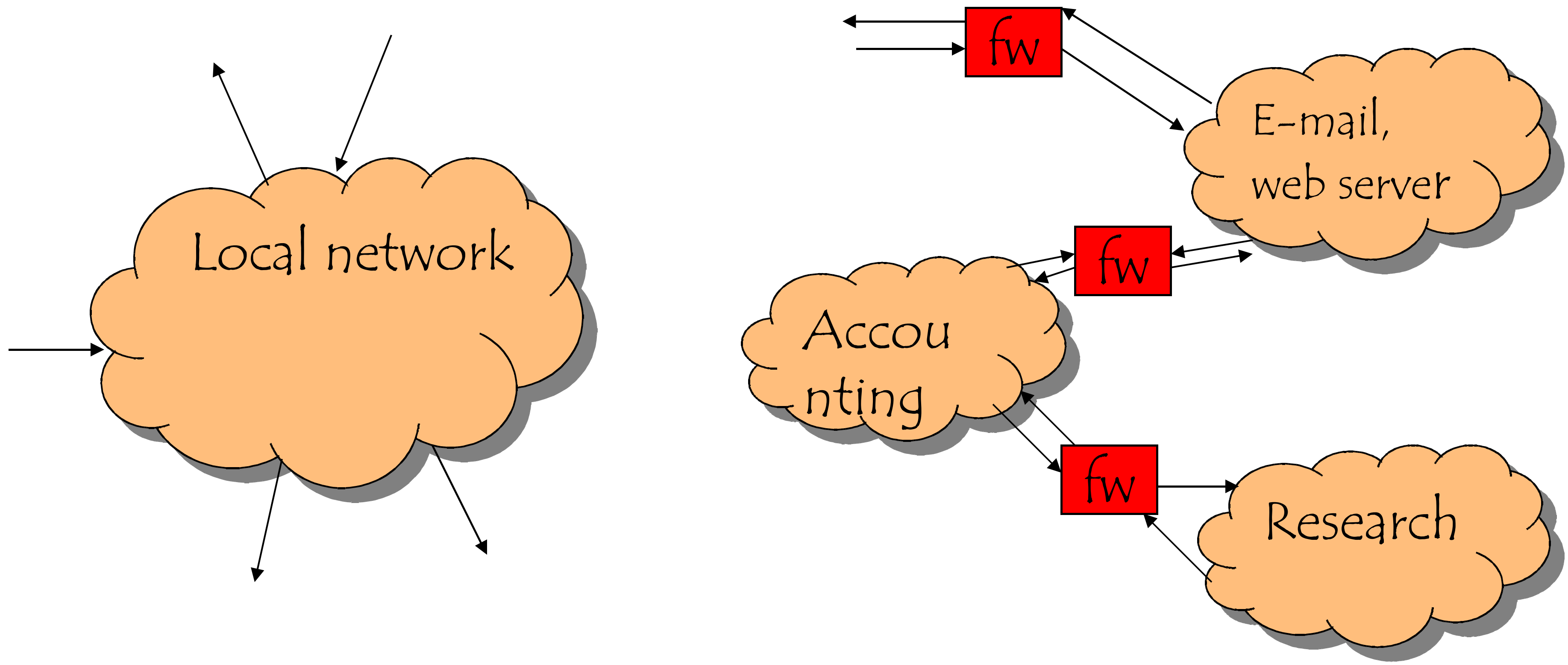
- A cybersecurity strategy that layers defensive mechanisms to protect valuable data and information. If one mechanism fails, another steps up immediately to thwart an attack.
- This multi-layered approach with intentional redundancies increases the security of a system as a whole and addresses many different attack vectors

Countermeasures – Resist & Recovery

Defence-in-depth

- A flat network is segmented into distinct subnetworks, each with a security level
- Only networks with consecutive security levels are connected, ie at most two connections for each network
- A firewall protects any connection between two networks
- Physical node connections may have to be updated

Defence-in-depth



Initial configuration

Defence in depth

Firewall & Virtual Machine

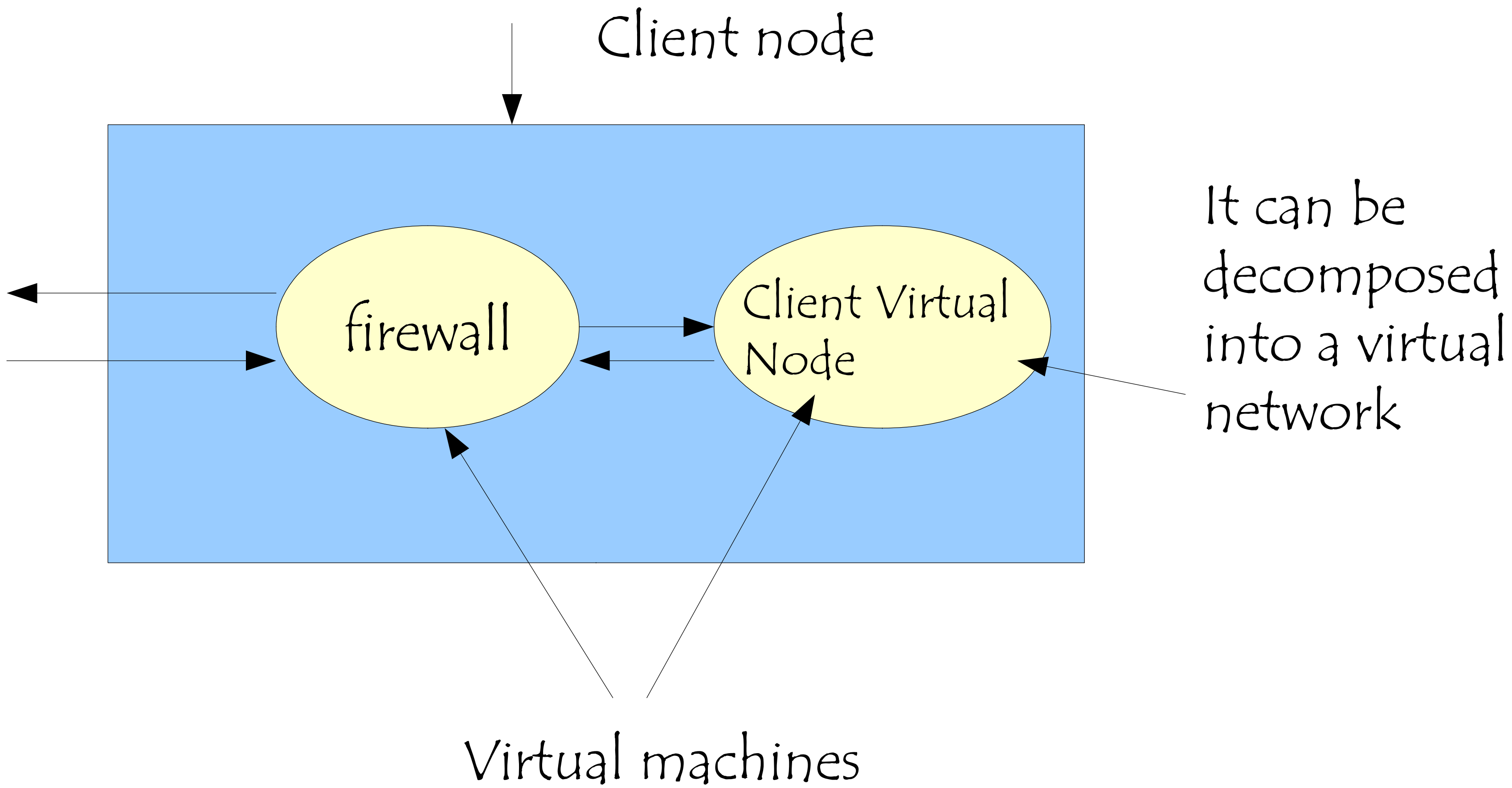
- Virtualization technology supports the definition of virtual network (overlay network of virtual machine) to spread information across a large number of nodes and of networks
- Virtual networks may be protected by (virtual) firewall and by mapping some virtual nodes onto distinct physical nodes
- Distinct virtual nodes + distinct networks simplify information management as each network stores and handles a low amount of homogeneous information from a security perspective = microsegmentation

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

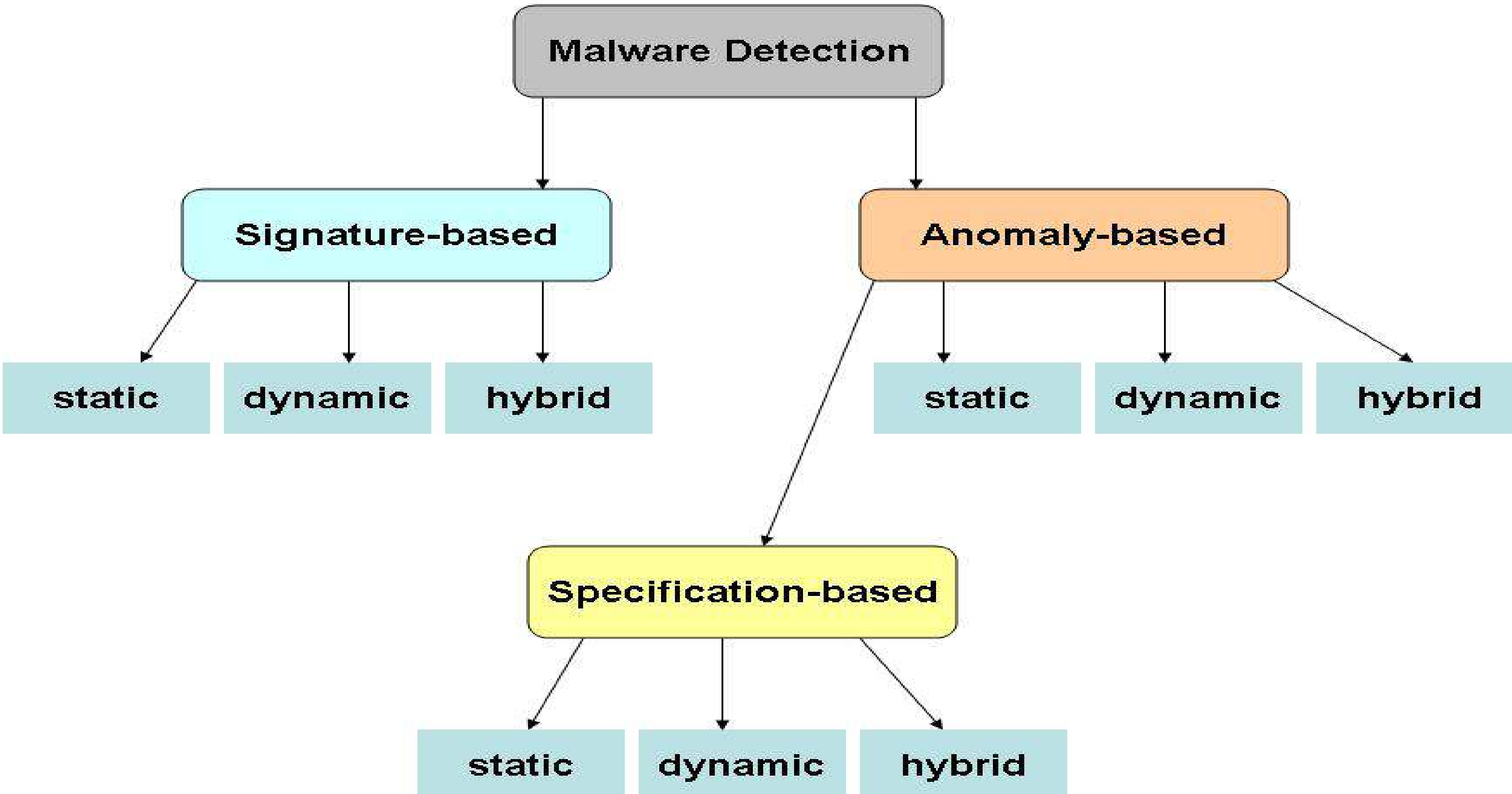
Personal or real firewall?



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
- Alternative strategies to discover events of interest

Countermeasures - Detect



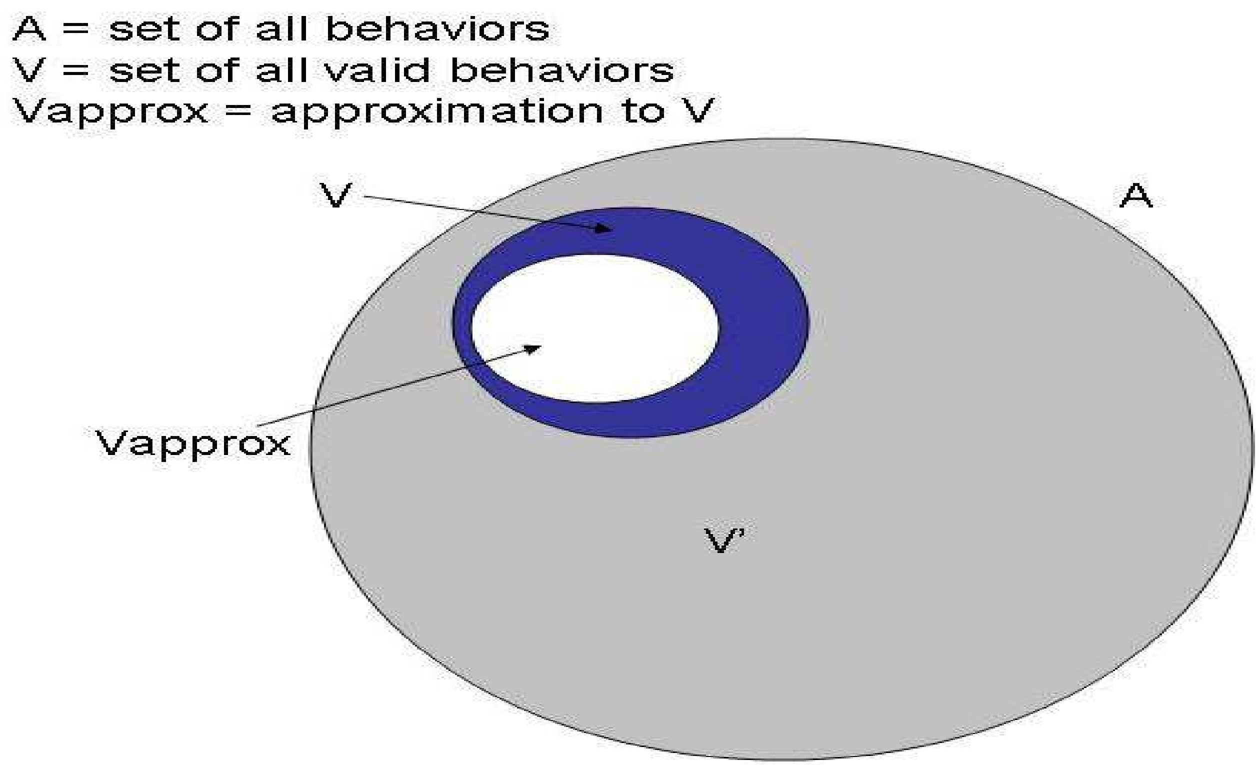
Detection – Anomaly Based

- The behaviour of the system to be protected is observed for an interval of time (learning the normal behavior)
- After the learning, any behavior too “distant” from those that have been observed, and learnt 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 by executions and then compared against the behavior
- Static
  - Information on the expected behavior are produced by a static analysis of the structure of a program
- Hybrid
  - The expected behavior of the program is collected to fill the gap due to a static analysis and the output is compared against the behavior

Detection – Anomaly Based



In general the collected information enables
to approximate the behavior of interest

Detection – Specification Based

Normal behaviors are not learned by execution, 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 to compute the spec and the behavior is compared against the specification
- Hybrid
  - The compiler builds a specification and observations are collected to be compared against the program behavior

Detection – Signature Based

- Main idea: there are some behaviors and data (constant) that fully characterize and identify a malware = the malware signature
- 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 detected only if its signature is known = a 0-day exploit cannot be detected = to discover new attacks an anomaly detection approach should be implemented or information returned from threat intelligence should be exploited
- Alternative strategies can be adopted to define the signature

Detection – Signature Based

Determines whether a program is malicious through default allow = anything that differs from a signature is allowed.

- Dynamic
  - Information on the program behaviour are collected (even by an protectd execution) and compared against the signature
- Static
  - The program code is analyzed and compared against the signature
  - Used by antivirus tool
- Hybrid
  - Mergest the two approaches: a static analysis selects a subset of the programs and the behaviour of these programs is monitored

Detection – Signature/Anomaly Based

Known-Known	Known-Unknown	Unknown-Unknown
Detect the exactly known infection, as seen before	Detect previously unseen variations of known threats, subfamilies or related new threats	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/b/opt/8681BAE3DB3A2F9D446CD5E3** **hxxp://50.63.147.69:8080/b/req/3D111E6B21F373015C646CA4**	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	**Very high precision** **Generalization limited** Recall limited to predefined pattern; finds variations explicitly covered by the pattern	**High precision** **Generalization based on similarity to known malware** Ideal for finding previously unseen variations/subfamilies of known infections	**Good precision** **Generalization based on common suspicious behaviors** High recall, good chance to find true zero-days, at the cost of more false alarms	**Low precision** **Generalization based on unusual behaviors** Best chance to find true zero-days; highest risk of false alarms
	Good explainability	**Good explainability**	**Good explainability but more complex**	**Explainability limited** Findings may be difficult to attribute to known infections	**Explainability difficult** Findings may be difficult to attribute to known infections
	Does not scale **Requires manual definition**	**Does not scale well** **Requires manual definition**	**Scales somewhat well** **Learned (semi)auto from data**	**Scales well** **Learned (semi)auto from data**	**Scales well** **Learned auto from data**
	Not applicable to encrypted data without MiTM	*Not applicable to encrypted data without MiTM*	*Applicable to encrypted data without decryption*	*Applicable to encrypted data without decryption*	*Applicable to encrypted data without decryption*

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 signatures

- Node local events
  - OS calls
  - File operations
- 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
    - Intrusions from the outside that have violated the firewall
    - Insider intrusions that the firewall cannot prevent
  - Unstable technology

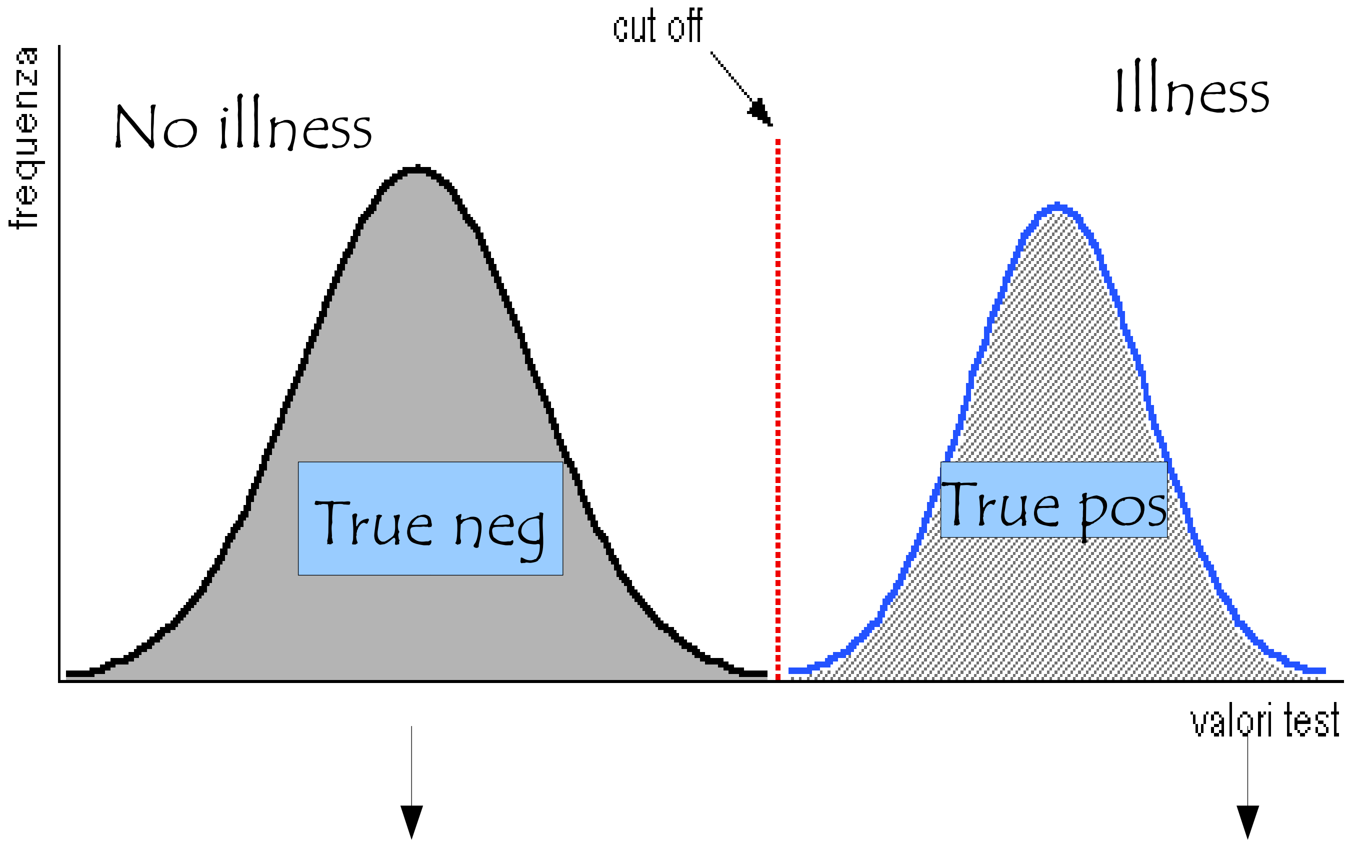
IDS, false positive, negative…

- The tool can detect behaviors that approximate 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 (behaviors) that suggest that the system has been or is being attacked
- However, we are not sure that the attack is going on

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 – one parameter

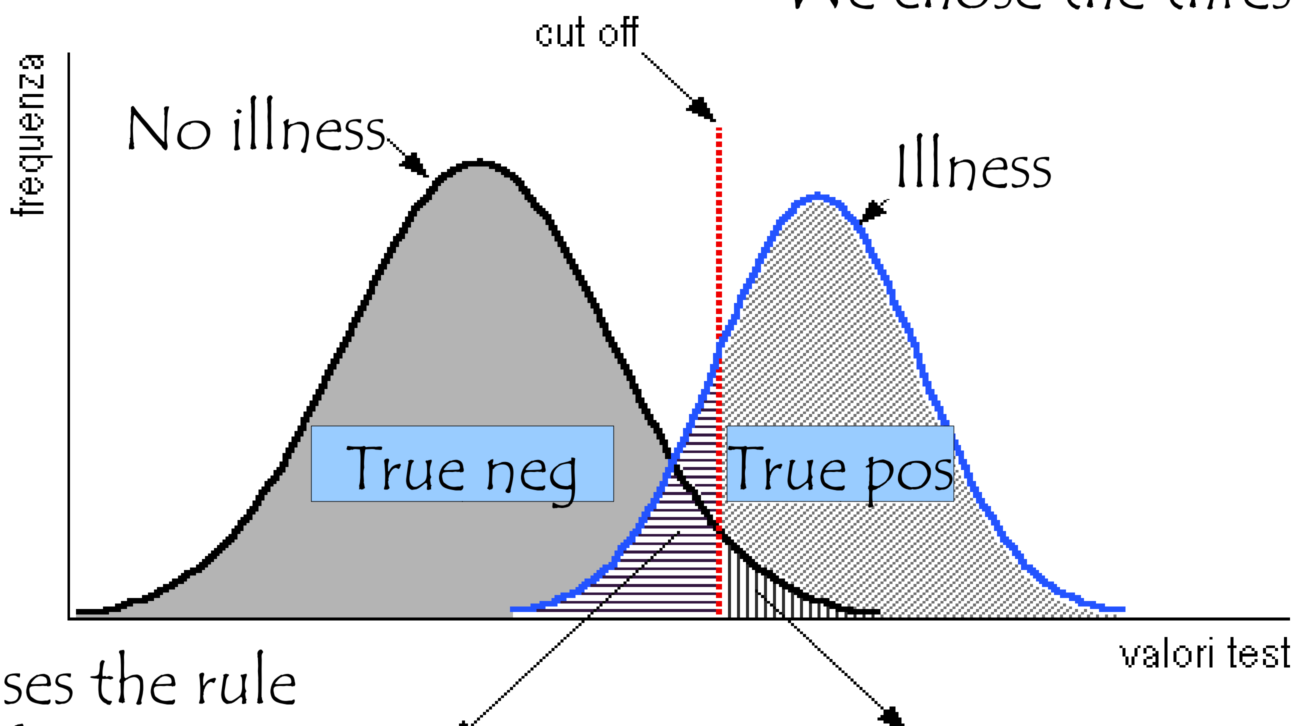


Probability distribution of the parameter, no illness

Probability distribution of the parameter, illness

Any real test

We chose the threshold



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

False negative false positive

A real test

False positive

GOLD STANDARD

True positiv

	Ill	No ill
Test +	a	b
Test -	c	d
	a+c	b+d

Ideal test = perfect knowledege

A real test

False negative

True negative

Another case: biometrics

BIOMETRICS COMPARISON CHART

Biometric	Verify	ID	Accuracy	Reliability	Error Rate	Errors	False Pos.	False Neg.
Fingerprint	✓	✓	◎◎◎◎	▶▶▶	1 in 500+	dryness, dirt, age	Ext. Diff.	Ext. Diff.
Facial Recognition	✓	X	◎◎◎	▶▶	no data	lighting, age, glasses, hair	Difficult	Easy
Hand Geometry	✓	X	◎◎◎	▶▶	1 in 500	hand injury, age	Very Diff.	Medium
Speaker Recognition	✓	X	◎◎	▶	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	✓	X	◎◎	▶	1 in 50	changing signatures	Medium	Easy
Keystroke Recognition	✓	X	◎	▶	no data	hand injury, tiredness	Difficult	Easy
DNA	✓	✓	◎◎◎◎	▶▶▶	no data	none	Ext. Diff.	Ext. Diff.

Sensitivity

Sensitivity = probability of a positive answer in an ill person

	Ill
Test +	a
Test -	c
	a+c

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

$$Sen = a/(a+c)$$

Specificity

Specifity= probability of a false answer if no illness

		No ill
Test +		b
Test -		d
		b+d

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

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

Likelyhood

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

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

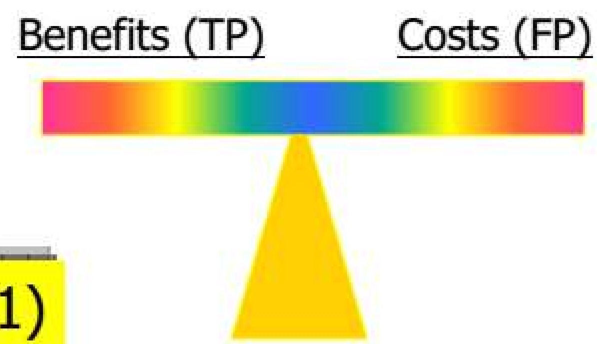
LR-= ratio between the probabilities of a negative test in one ill and one healthy person

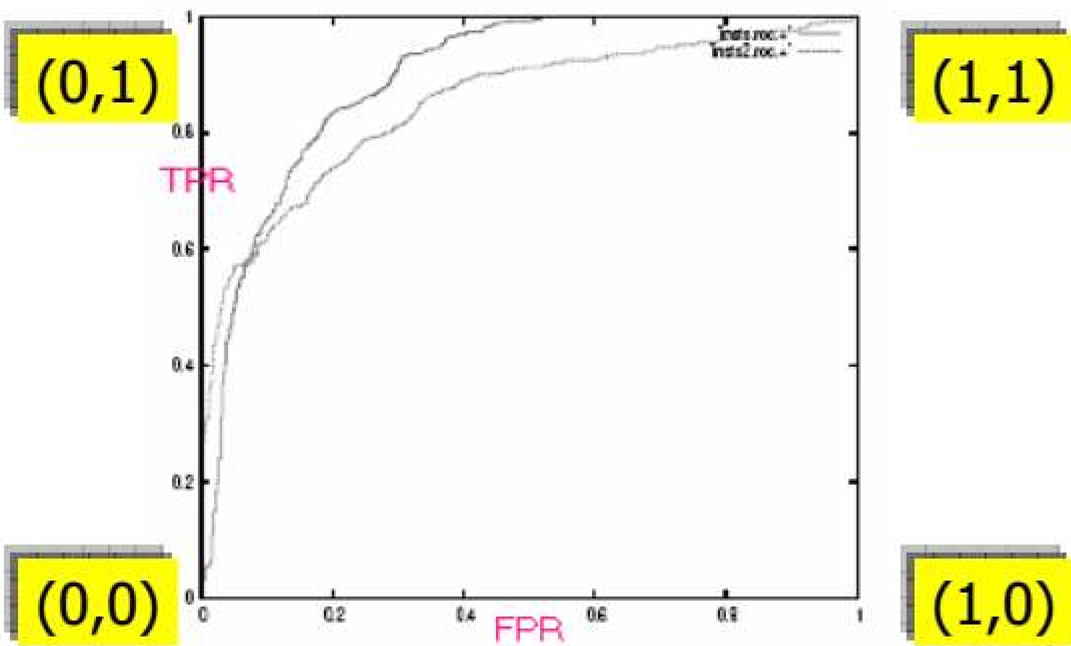
ROC curve

- This curve describe a classifier and plots the true positive rate (TPR) against the false positive rate (FPR) out of all positive observations (TP/(TP + FN)) vs proportion of observations incorrectly predicted to be positive out of all negative observations (FP/(TN + FP)).
- A classifier that uses a treshold and returns only a class gives a single point on the space we create a curve by varing the threshold. This evaluates the features the classifier considers rather than the threshold
- It shows the trade-off between sensitivity (or TPR) and specificity (1 – FPR). The closer to the 45-degree diagonal the less accurate the test.
- ROC is useful to evaluate classifiers predicting rare events such as disasters. In contrast, evaluating performance using accuracy (TP +TN)/(TP + TN + FN + FP) favors classifiers that always predict a negative outcome for rare events.

ROC curve
receiver operating characteristics

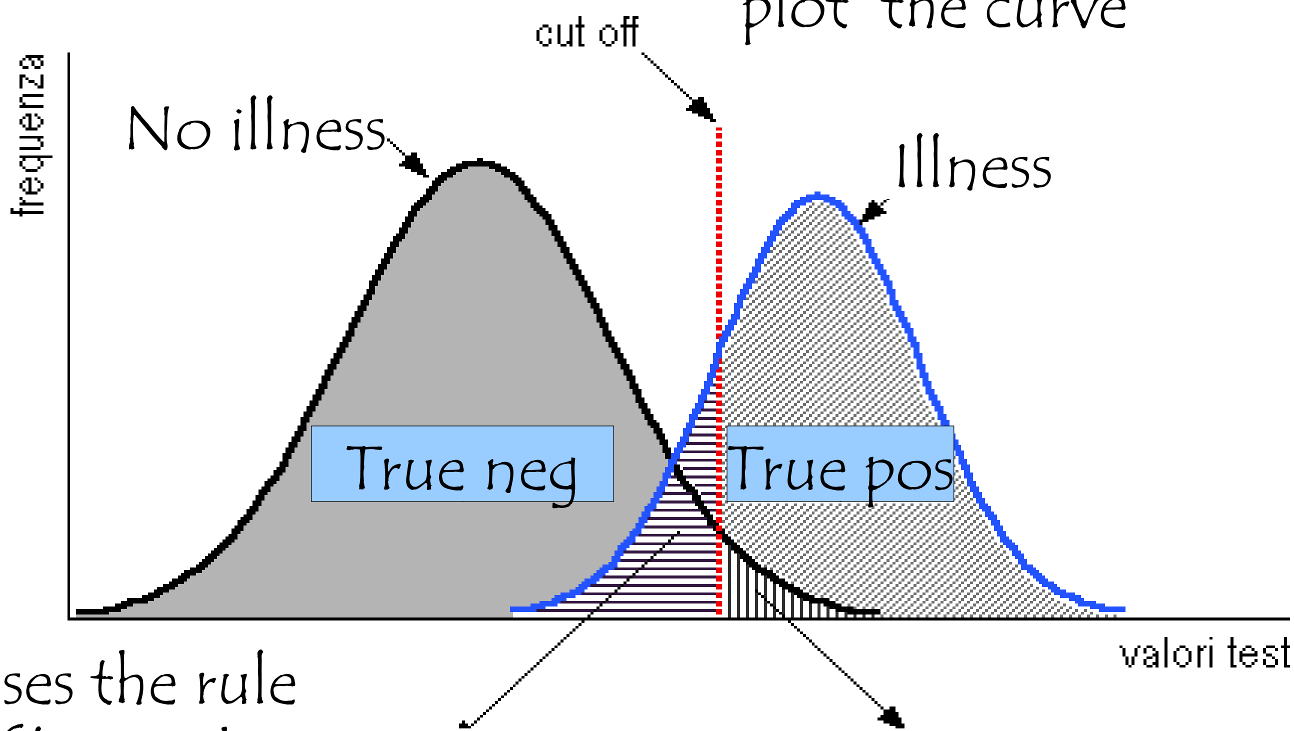
- Y axis: TPR = sensitivity
  X axis: FPR = 1-specifity





Drawing the curve

We move the threshold and plot the curve

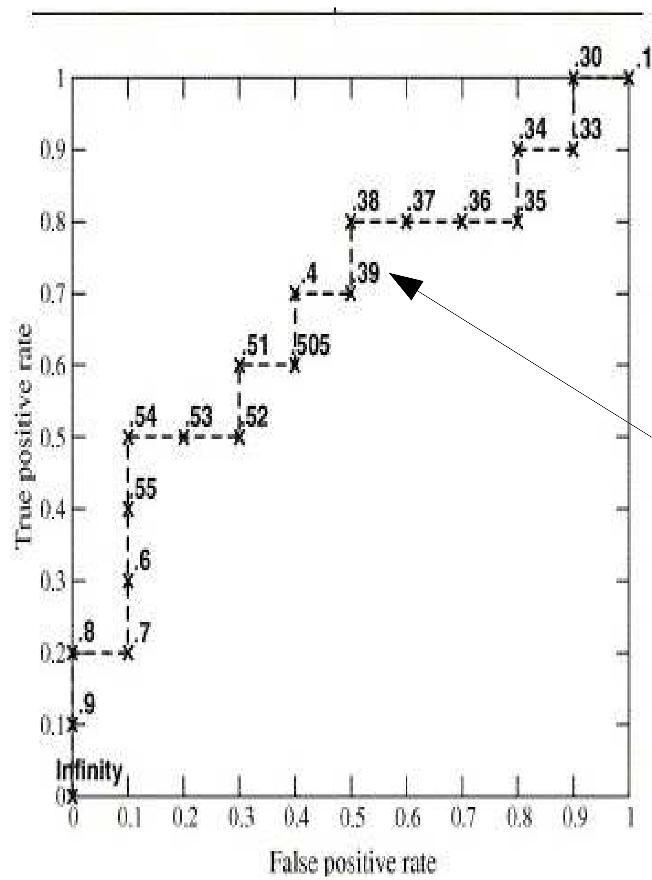


Detection uses the rule "no illness if lower than threshold"

False negative false positive

Drawing a curve

nst#	Class	Score	Inst#	Class	Scor
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



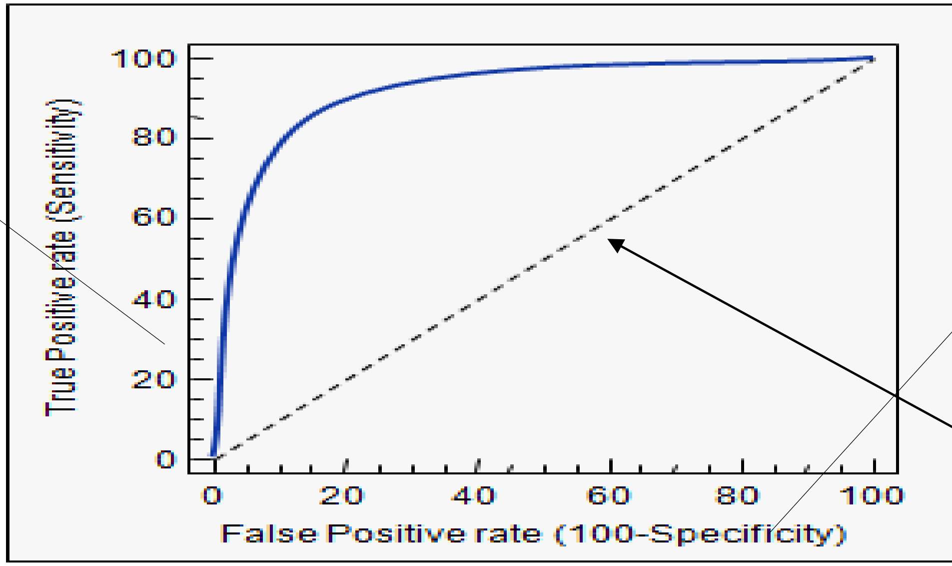
th chosen

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)

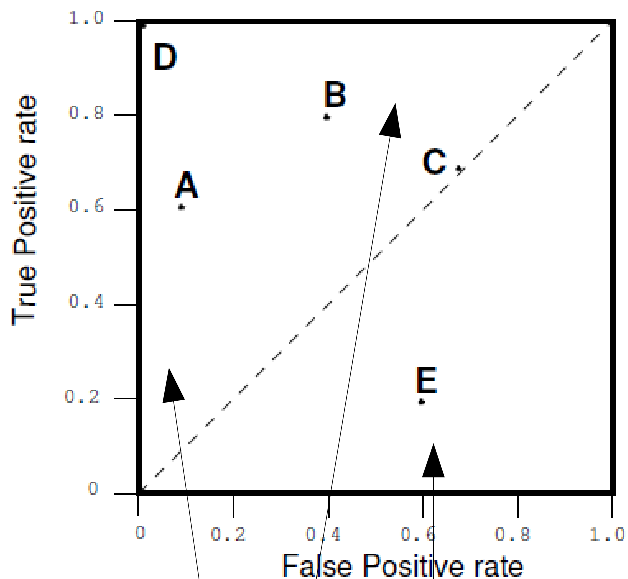
Each value of x results in a percentage of false and true positives

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 positves and few true positive

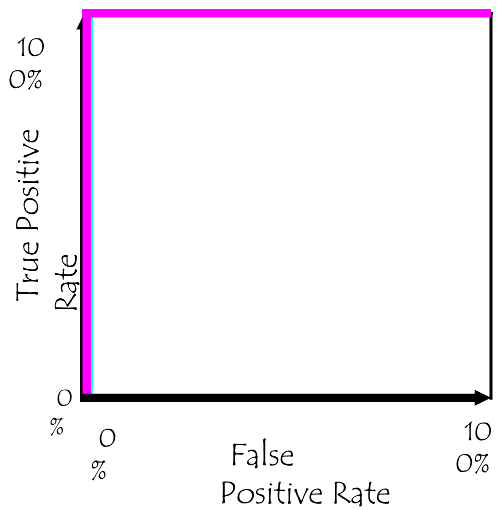
A rule under the bisector = worse than random (= the bisector) it can be improved by negating it

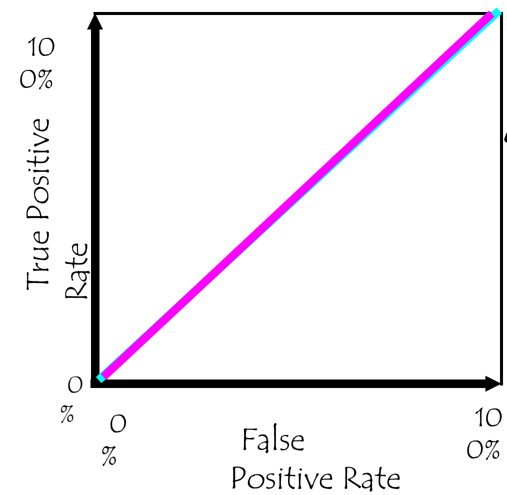
Area under ROC curve (AUC)

- *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

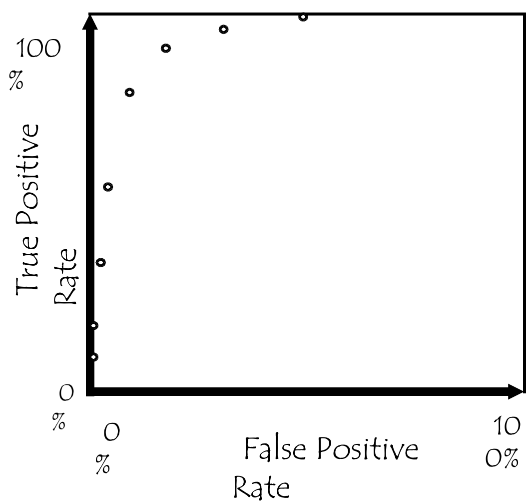
AUC = 100%

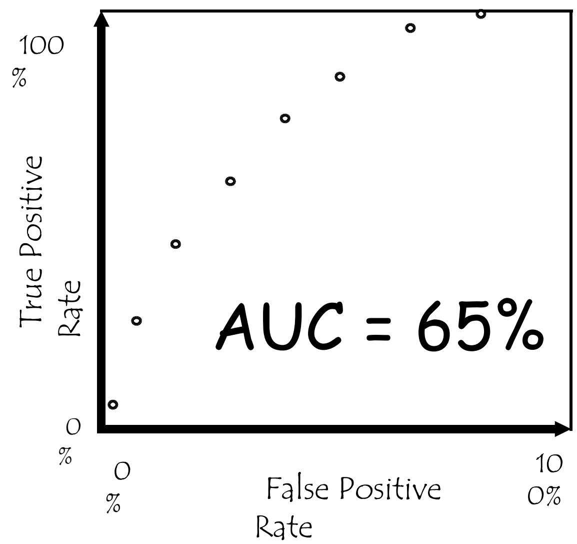




AUC = 50%

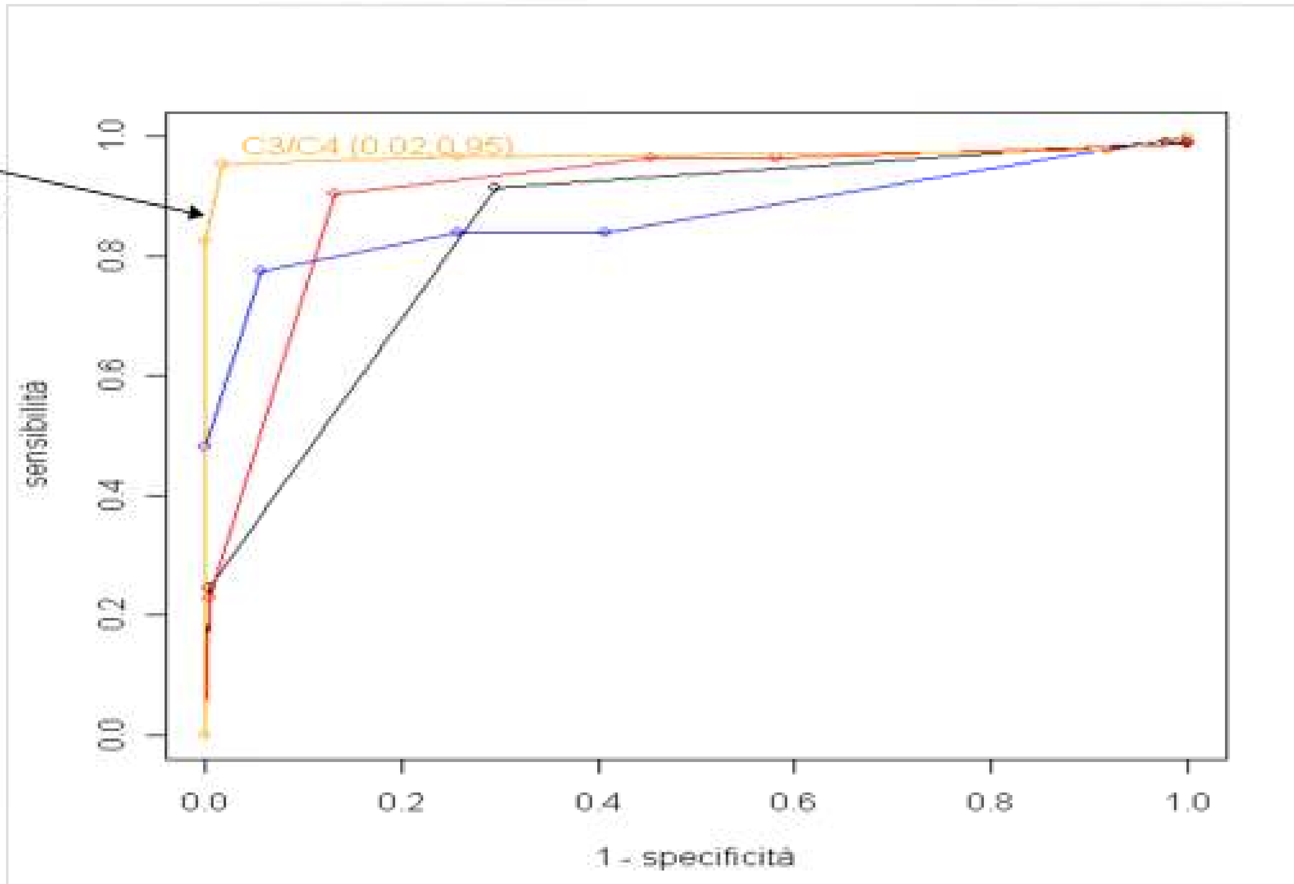
AUC = 90%





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 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 is the 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 because of both performance and security

NIDS + HIDS

- The two tools can cooperate through a distinct interconnection network (a control network)
- The real problem is how much one tool can trust the other (mutual trust)
- The host running a tool may be attacked and controlled by the attacker

NIDS+ HIDS = IDS = sensors+ engine

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

IDS

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