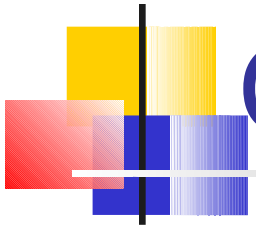
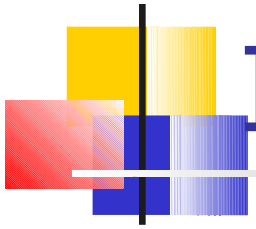


Computer & Network Security

- A topic at the intersection of three areas
  - Computer Science
  - Human Resources and Management
  - Economy
- From ICT security to ICT risk assessment and management
- “Kids speaks about security real women/men about risk assessment and management :-)”
- Risk management = an approach strongly related to probability, impact, cost effectiveness of solutions

Why security is important

- Any organization strongly depends upon
  - Its private ICT resources
  - The ICT resources of its partners
  - The ICT systems that connect its private resources with the partners' resources
- Any organization should be able to prove to other ones that it controls its ICT resources
- Anytime the organization has to prove that it satisfies some standards (not only an ICT one) it needs to prove that it controls its ICT resources



Information Security

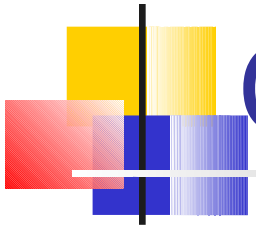
- Confidentiality

  An information can be read only by those that are entitled

- Integrity

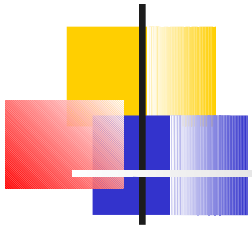
  An information can be updated only by those that are entitled

- Availability
  - An information can be read and updated by those that are entitled when they require the operation
  - An ICT resource should be available to those that are entitled to use it



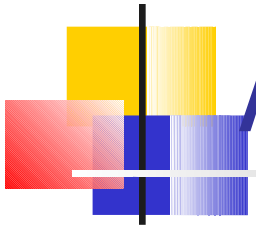
Other properties

- Autenthication = you are who you say you are
- Traced = who has invoked an operation
- Accountability = pay for what you have used
- Auditability = evaluate the effectiveness of security solutions
- Forensics = information to prove that that some laws have been violated (authentication + integrity)
- Privacy = protection of personal information (stronger requirements, no inference)



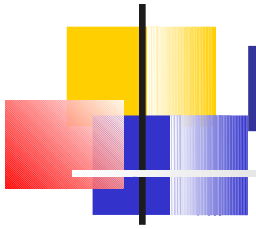
Vulnerability

- A key concept for security
- A vulnerability is a defect (an error, a bug) in a person, a component, a set of rules that makes it possible to violate a security property = it enables an attack
- While all vulnerabilities are bugs (errors...) not all bugs are vulnerabilities



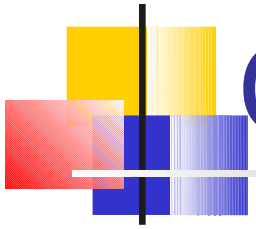
Attack against an ICT system

- An attack is a sequence of actions to gain the control of (a subset of) an ICT system
- The actions can be implemented by a program
- Each attack is possible because of some vulnerabilities of the target system
- Who controls an ICT (sub)system can
  - Collect any information in the (sub)system
  - Update any information in the (sub)system
  - Prevent someone from accessing any resource/information in the (sub)system



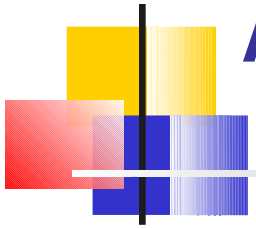
Next lectures

- No lesson on 7th of March
- Lesson on 11th March, 16-18 C1



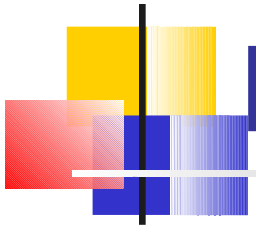
Our perspective

- Attack focused= a cost effective defense from attacks to an ICT system
- Why/Which/When attacks may be successful
- How attacks can be managed (prevented, reduce number, reduce damage …)
- Selection and deployment of cost effective countermeasures (changes to the system)
- Cost, return, investment, ...



Alternative approaches

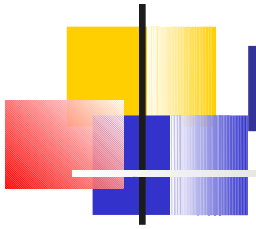
- Unconditional security
  - Any vulnerability in the system can be exploited by an attacker
- Conditional security (risk management)
  - Discover which vulnerabilities are convenient for those interested in attacking the system
  - Some vulnerabilities are not exploited because they are not cost effective due to the high cost of the attacks they enable or the risk for the attacker



Risk analysis

A modern approach to security:

1. Asset analysis (resources to be protected)
2. Vulnerability analysis
3. Attack analysis
4. Threat analysis (sources of attacks)
5. Impact analysis (damages)
6. Risk management =
   - Define acceptable risk
   - Select and implement countermeasures



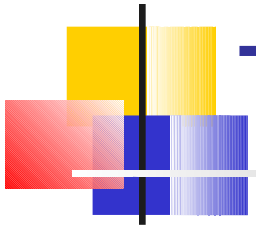
Risk analysis and management

- Not all the attacks are worth preventing
- Economy driven solution
  - Which attacks can be prevented
  - Which of the previous attacks are worth stopping
- A complete and coherent methodology is not currently available
- Several partial solutions to be integrated

Asset Analysis

- Which logical and physical resources of the ICT system that are to be protected
- Who is entitled to access these resources and which operation they are entitled to invoke
  - Who is entitled to read an information
  - Who is entitled to update an information
  - Who is entitled to run a given application

  .....
- The analysis defines the goal of our strategy: which resources are we going to defend

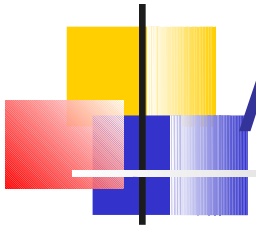


The steps of an attack

1. Collection of information about a system
2. Discovery of system vulnerabilities
3. Search or build of a program (=exploit) to implement the attack (even partially)
4. Implementation of the attack $\Leftrightarrow$

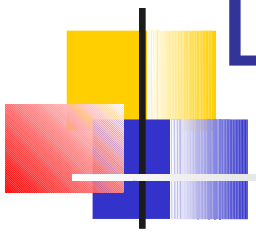
   Execution of the exploit +

   Execution of human action
5. Install tools to control the system
6. Remove any attack trace on the system
7. Access, update, control a subset of the system information



Automated attack

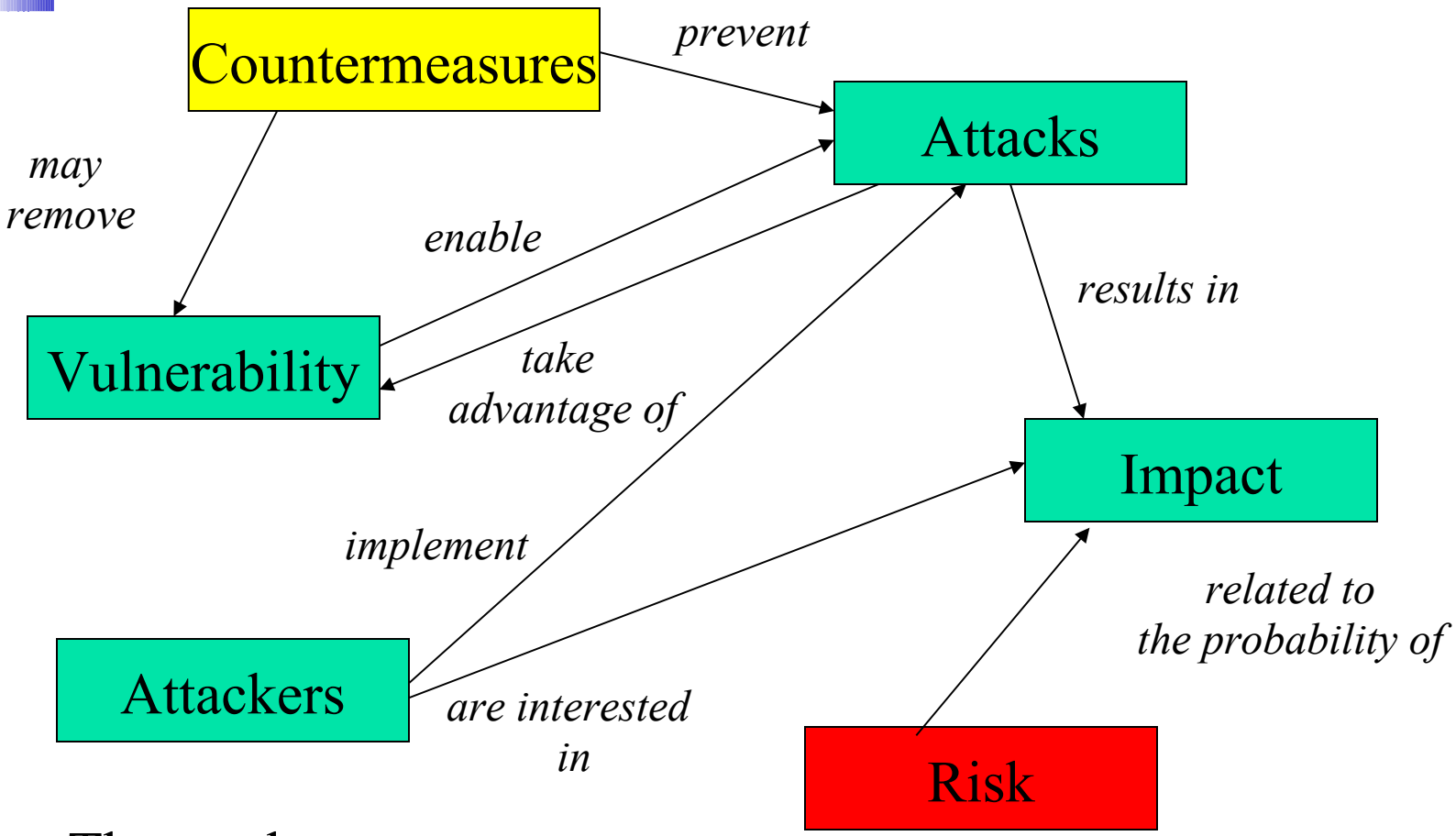
- No human action is required, the implementation of the attack is the execution of the exploit
- This is the most dangerous kind of attacks
- Automated attacks characterize ICT security with respect to security in other fields
  - Execution time of an automated attack is neglectable
  - No know how or abilities are required to the attacker to execute an exploit



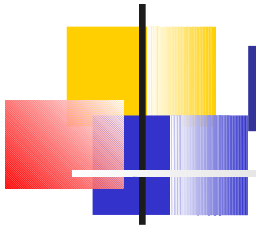
Local vs remote attack

- An attack is
  - Local if it can be executed if the attacker can access an account on the system
  - Remote if it can be executed even if the attacker cannot access an account
- A remote attack is obviously more dangerous
- Remote + automated = one of the worst cases

Terminology and relations ...



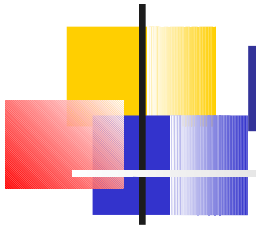
Threat, threat agents



Partial points view on sec– I

- Security = Confidentialy

  $\Leftrightarrow$ Cryptography
- A set of algorithm to hide information so that only those who know another information (the key) can read it
- A fundamental but partial information because it cannot guarantee availability
- It simplifies but not solves a problem



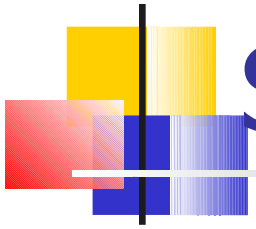
Partial points of view – II

- Several security problems are related to the triple <user, resources, rights=operations on the res>

  that determines who can execute what
- Several security mechanisms are related to the solution of these problems
  1. Identifying the user
  2. Identifying the resource
  3. Discover the user rights on the resources
- Sophisticated identification system (biometrics etc.) can solve 1 but neither of the other ones

Partial point of view - III

- Security is not safety
- In a system with 10^n -1 safe states and 1 unsafe one, the probability of an unsafe behavior $= 1/10^n$, system safety increase with n
- If one state out 10^n is not secure, the attacker works so that the system enters the not secure state
- The system security depends upon the attacker success probability rather than on the overall number of states
- Attackers have an intelligent, not random behavior



Some examples

- Vulnerability
- Attack
- Some countermeasures

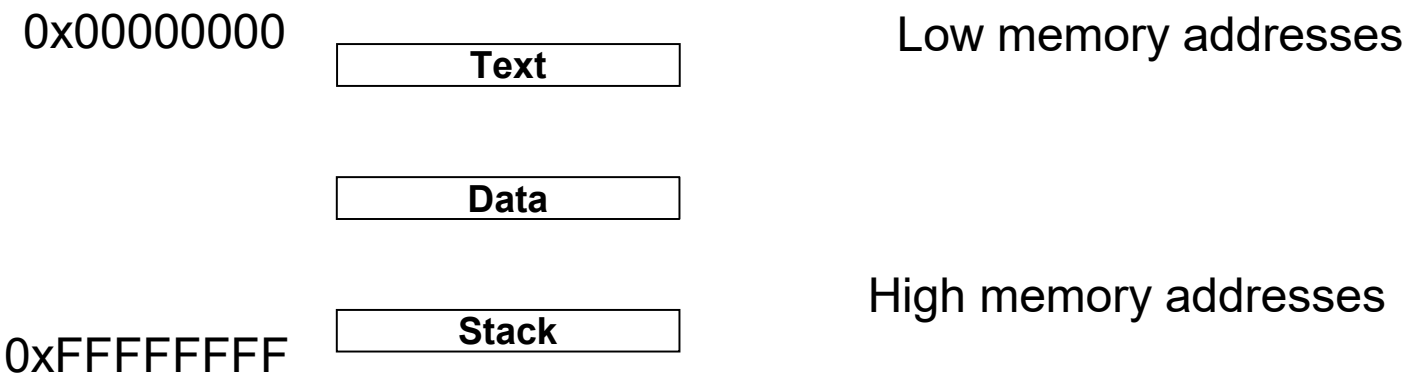
We describe a stack overflow, a popular attack that is an instance of buffer overrun

Buffer overflow

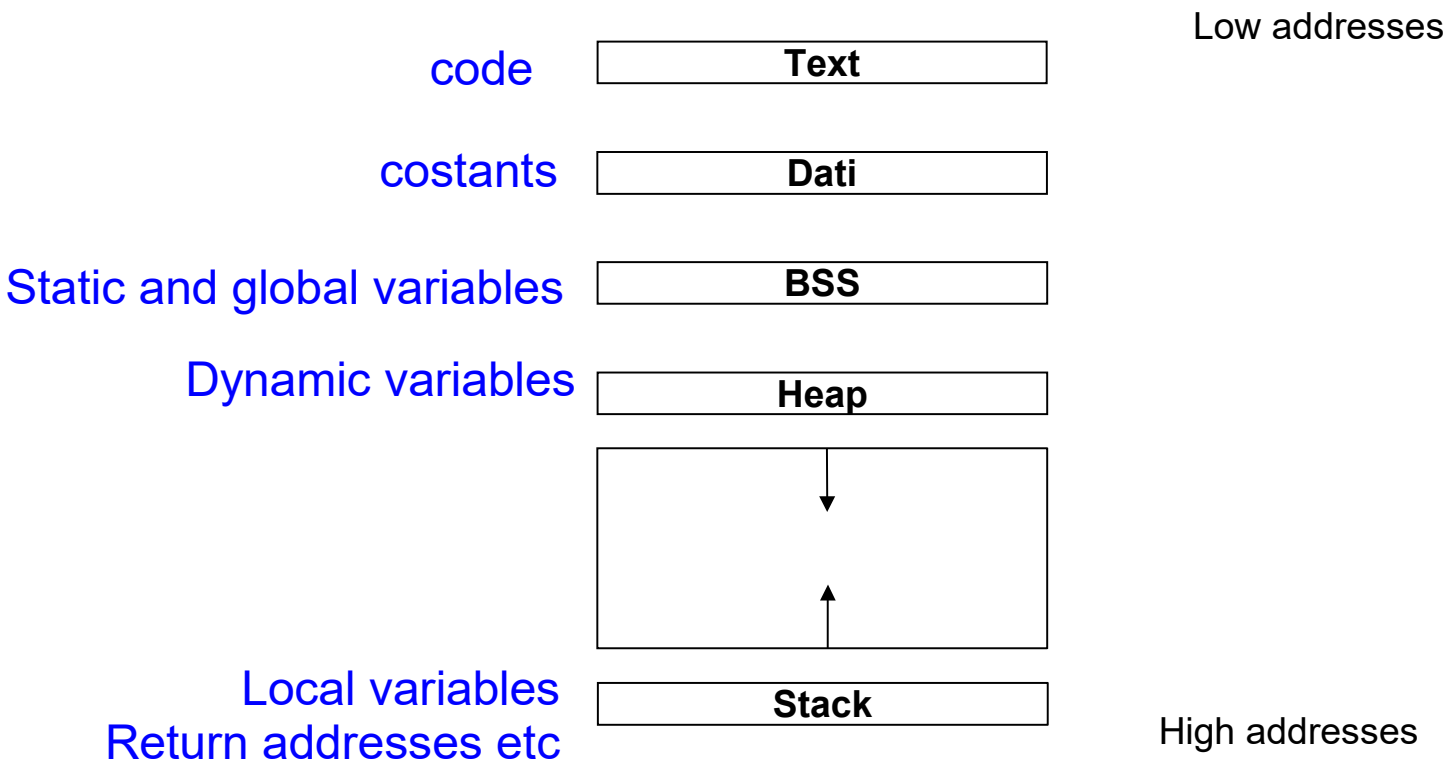
- The *buffer overflow* problem
  - the most common problem among all the vulnerability of C code
  - it does not arise in high level languages where the programmer is not involved in memory management or with strong data types
  - The most important security issue in the last 10 years (not replaced by web vulnerabilities)
  - based on a forced write of some data with a size larger than expected. If the program type system does not discover this inconsistency, then some data is replaced in memory.
- In this way, a program can be inserted (code injection) into a system and it can, among other execute some shell commands. If the program is executed at root level, then it fully control any system function.
- A buffer overflow can exploit any of the following areas allocated by the compiler: stack, heap e bss (block started by symbol) static variables.

A process memory

- To understand buffer overflow, we have to recall the structure of a process memory.
- A process memory is partitioned into three segments: *text*, *data* and *stack*.
- The *text segment* is fixed, stores the program code and it is read only. Any write attempts results in a segmentation error (segmentation fault – core dump)
- The *data segment* stores the process static and dynamic variables
- The *stack segment* stores the data to manage function calls and returns

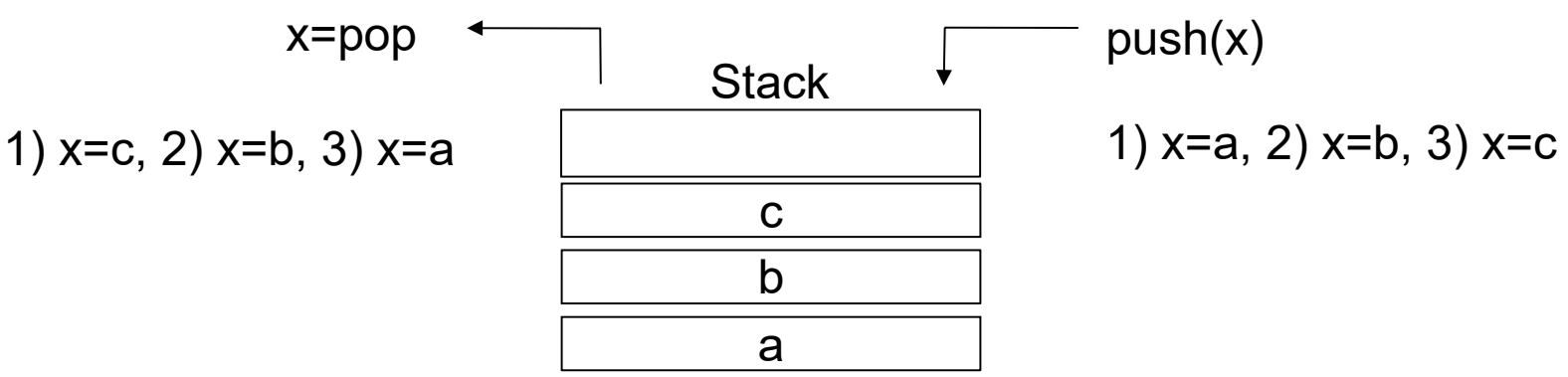


A process memory



Stack

- A Lifo (Last In First Out) data structure that stores a dynamic amount of information
- It is used to manage function calls and returns (call assembly instruction).
- The stack memory area is logically partioned into records (stack frame) one for each call



Stack and system registers

- The memory address of the instruction to be executed is stored in the EIP (Extended Instruction Pointer) register
- EBP (Extended Base Pointer) points to the beginning of a *stack frame* while ESP (Extended Stack Pointer) points to the end of the stack frame
- When a function is called, the system
  - pushes onto the stack
    - the return address = EIP+4,
    - the base address of the current frame = EBP
  - copies ESP into EBP to initialize the new *stack frame*.

Stack and system registers

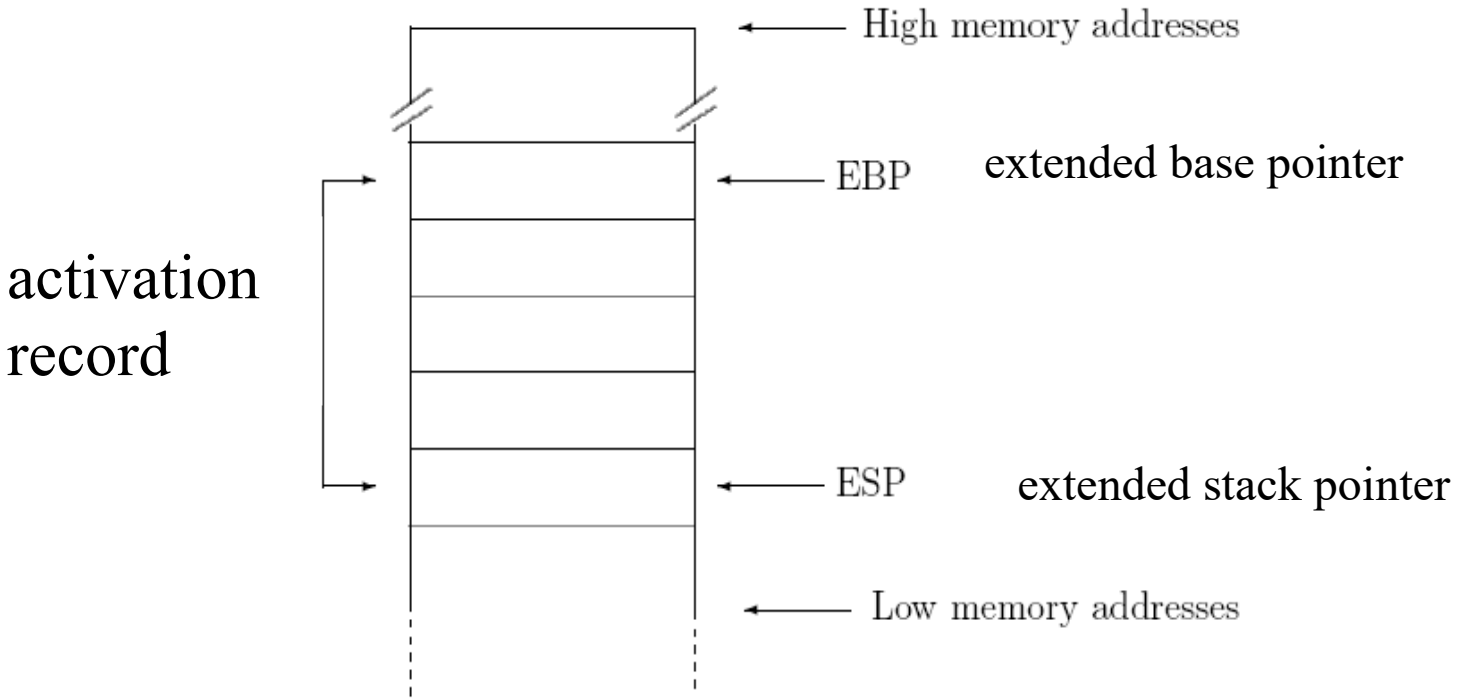
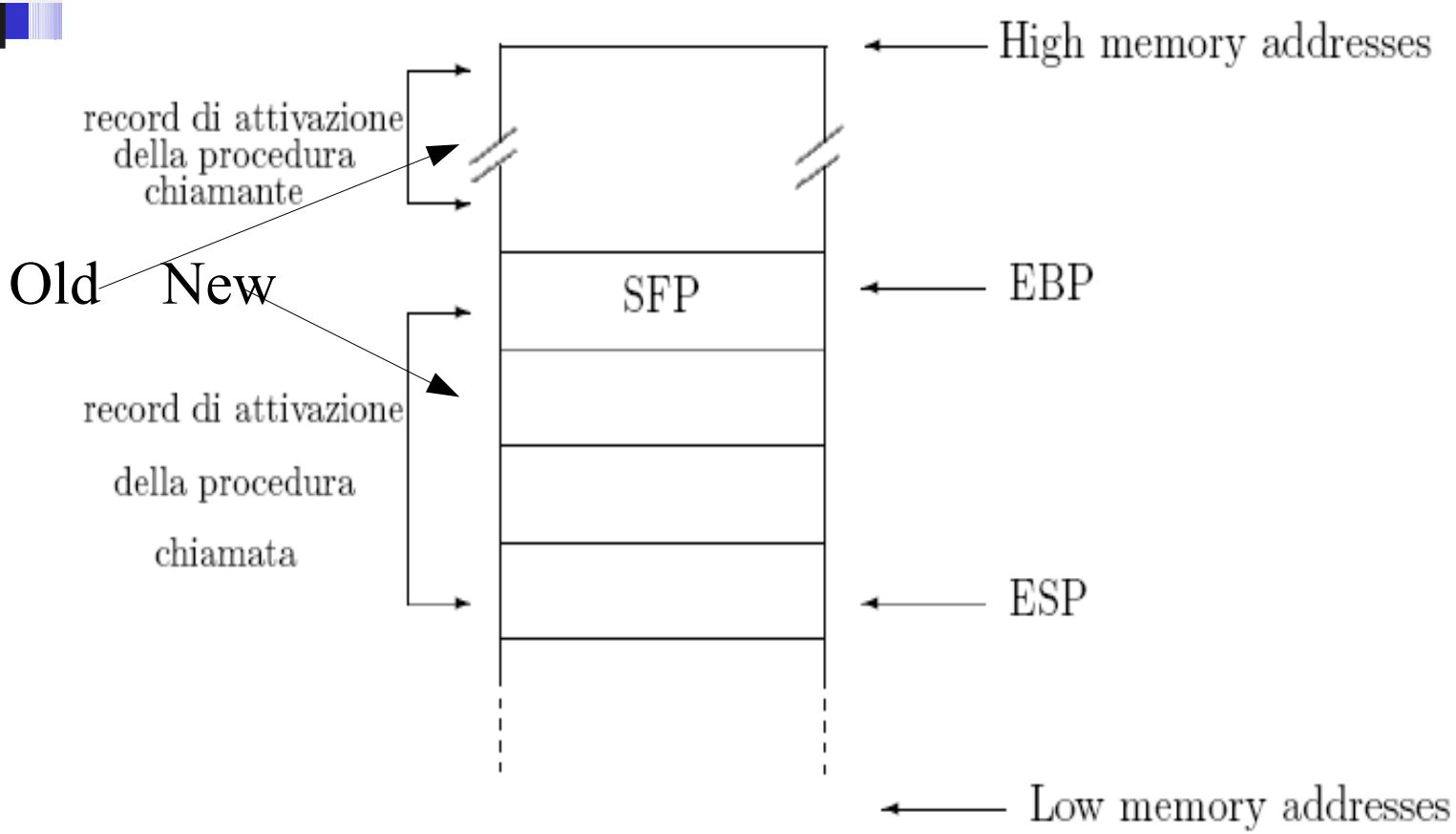


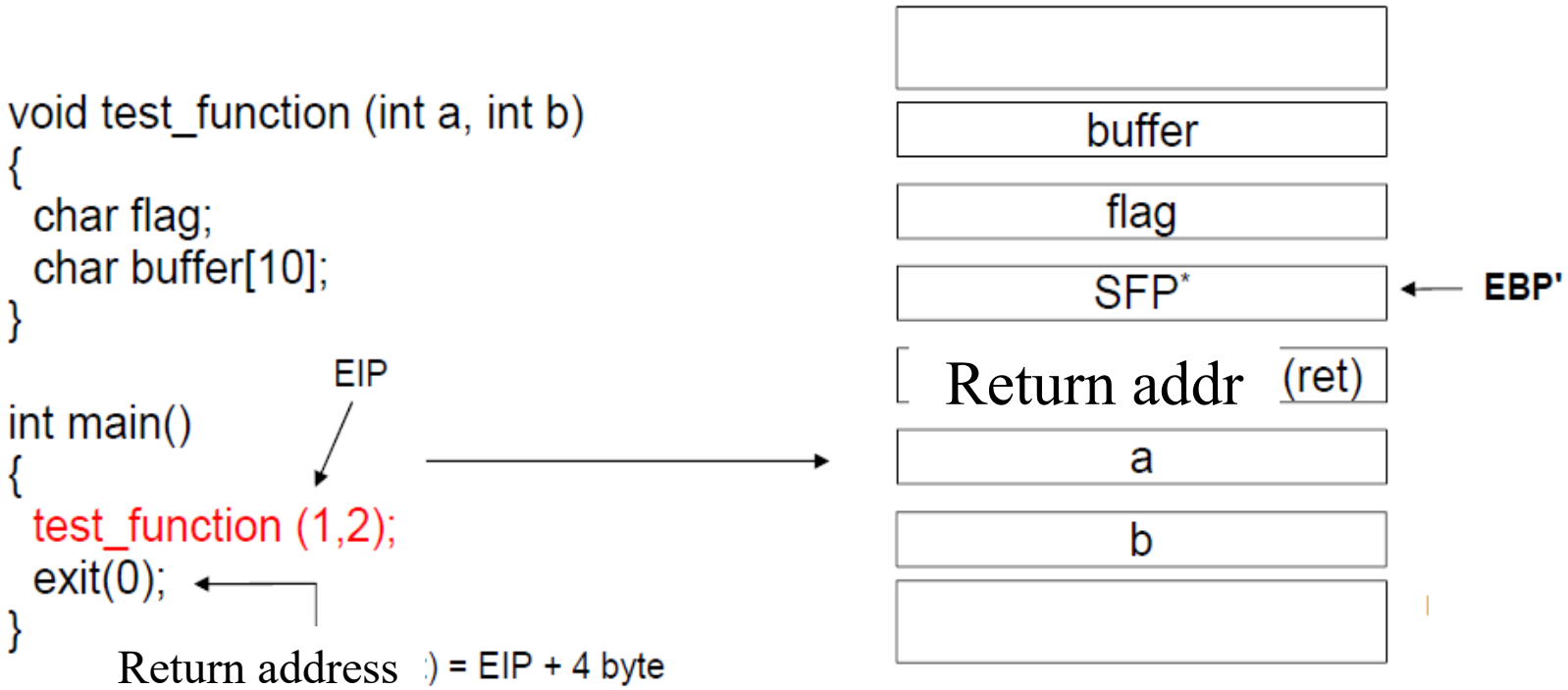
Figura 4: Stack pointer e frame pointer

Stack and system registers



C: an example

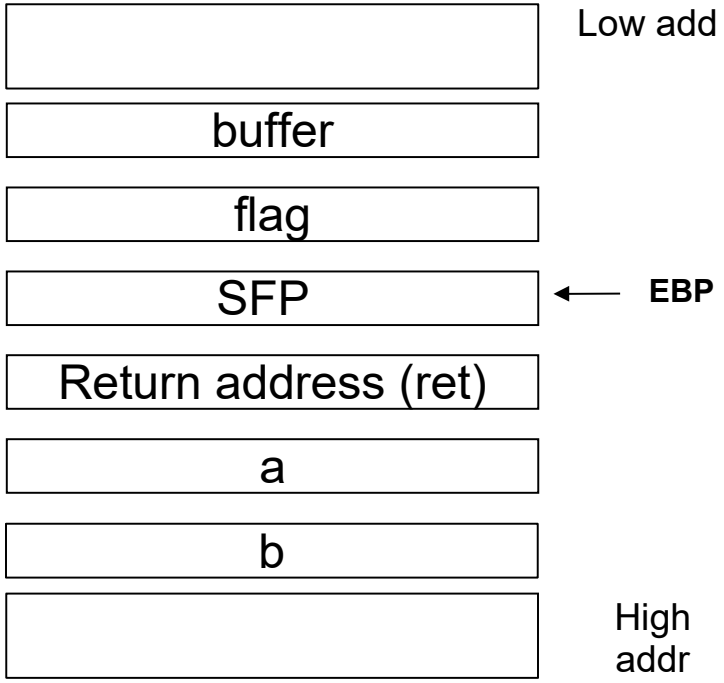
This is a simple example to see how all the stuff works



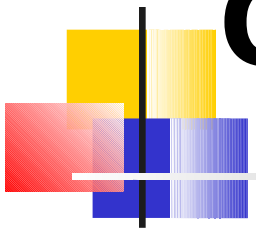
SFP = saved frame pointer = it is used to restore the original value of EBP on a return

The stack frame

- Local variable of *test_function* are addressed by subtracting a displacement from EBP while the function parametes are addressed by a positive displacement
- When a function is called EIP points to the function code.
- The stack stores both local variables and parameters of a function. When the function ends, the whole stack frame is removed before returning (ret).



Overflow: an example



This C code results in a stack overflow:

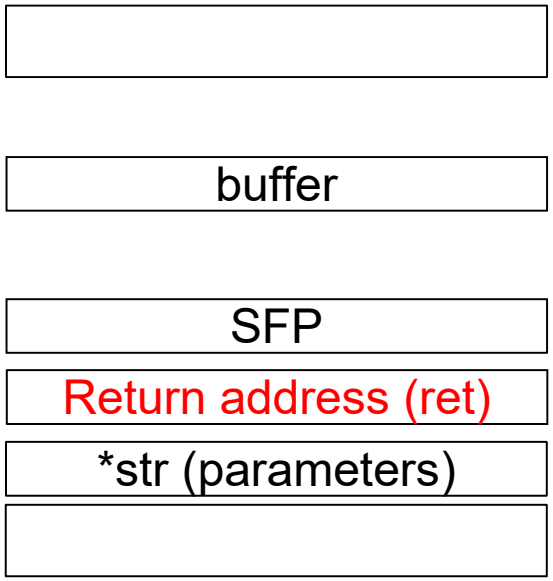
```
void overflow_function (char *str) {
  char buffer[20];

  strcpy(buffer, str);  // This function copies str into buffer
}

int main() {
  char big_string[128];
  int i;

  for(i=0; i < 128; i++)
  {
    big_string[i] = 'A';
  }
  overflow_function(big_string);
  exit(0);
}
```

This results in an overflow!

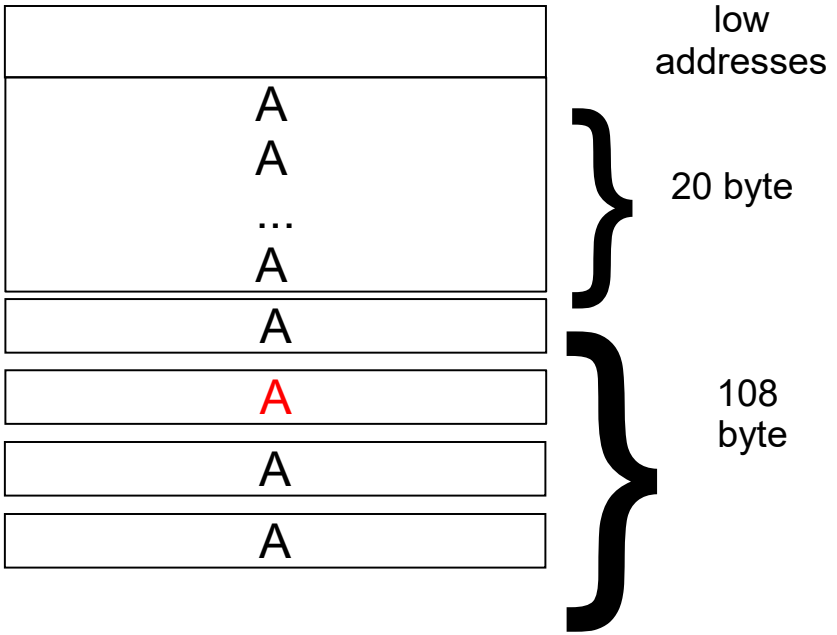
Segmentation fault

The previous code results in a segmentation fault

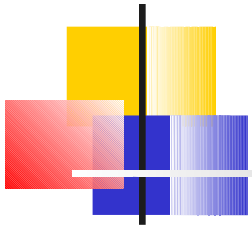
1)The first call to *overflow_function* correctly initializes the stack frame



2) When *overflow_function* ends, the return address has been overwritten by the character A (segmentation fault!)



Buffer (stack) overflow

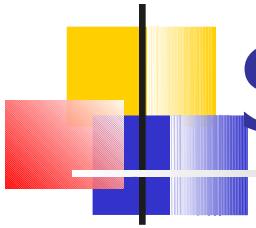


What happens if the return address (ret) stores a valid memory address?

- In this case no exception is signalled and the process continues by executing the instruction pointed by ret.
- A *stack based buffer overflow* exploits this opportunity by replacing ret with a pointer to some code injected by the attacker maybe into the stack itself
- How can we update the return address and inject some code in the system?

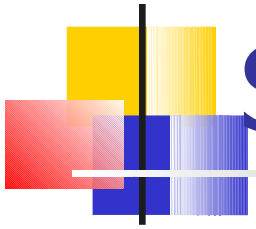
A Buffer Overrun

- It occurs when some variable is larger than expected and it overwrite other variables
- It may be implemented if the language lacks a typing system
- Four kinds:
  - Stack based buffer overrun
  - Heap based buffer overrun
  - V-table and function pointer overrun
  - Exception handler overrun
- Rather popular among computer worm



Stack Overflow

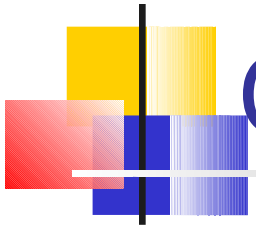
- By copying x into the stack we destroy (update ??)
  - The return address
  - Other values on the stack
- The values that are copied codify a program
- The new return address points to the program we have copied onto the stack
- Overall result: an administrative shell
- This is possible only if the procedure that is attacked is executed in root mode



Stack overflow

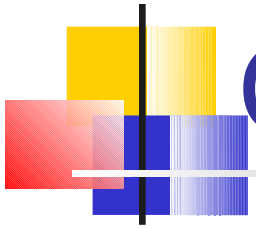
Vulnerability = distinct perspective

1. Lack of control on the size in the program
2. Bad type system
3. Incorrect memory operation
4. Growth direction of the stack
5. ...



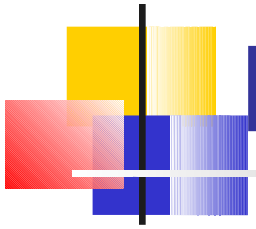
Overflow: countermeasures

- Strong typing
- Controls on string lengths
- Insert a “canary” into the stack
- Not executable stack
- Ad hoc checks in the compiler



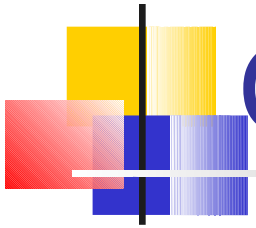
Canary

- A value that is updated at each invocation
- Inserted into the stack before any parameters
- Before returning we check that the canary has not been updated
- Updated at each invocation so that the attacker does not know its value



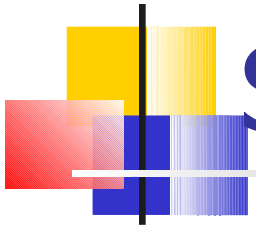
Not executable stack

- Controls when fetching an instructions, they can be supported by the MMU
- No data structure can store instructions
- It does not work with Linux that stores some drivers in the stack to manage i/o devices



Cost of the countermeasures

- Each countermeasure has a distinct cost
  - Strong typing = 10-30% run time overhead
  - Checks on string length = large cost but lower than the previous one
  - Canary = specialized control, low cost
  - Not executable stack = lowest cost because it exploits an hardware/firmware support

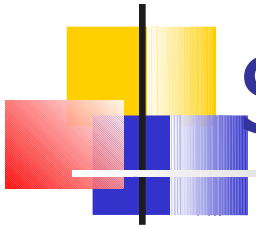


Structural vulnerability TCP/IP

- When the TCP/IP stack has been defined, the main goals was resilience against physical attack against the network (attack = bombing)
- Main goal = availability

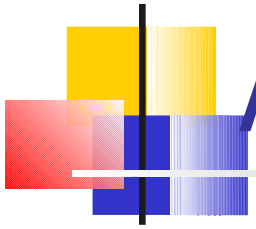
  $\Rightarrow$ Some mechanism defined to discover which nodes are alive and reachable

  $\Rightarrow$ No mechanism is available to guarantee \ (authenticate) the source of a message



Structural vuln: an Example

1. To check whether a node is alive and reachable, another node can send an ECHO message. The receiver reply with the same message
2. The sender can specify a partial IP address to broadcast a message to some other nodes
3. There is no control on the fields of an IP packet



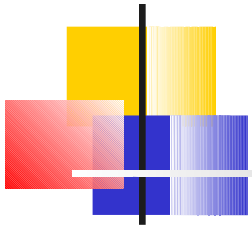
All toghether now ..

- R is a network with 1000 node, X is a partial IP address that matches the addresses of all nodes of R
- A sends a ECHO message to the address X but it specifies the address of B as the sender address of the packet
- Any node in R replies to B
- B cannot interact with other nodes because its communication lines are overflown by the ECHO messages

Distributed Denial of Service

Security as an holistic property

- The security of a composition is not related the one of its component
- Even if each of the component is secure the overall composition is not be secure
- In a virtual machine hierarchy the security of a machine may be undermined by a lower one



Impact and countermeasures

- The impact
  - depends upon the numbers of nodes, zombies, whose address matches that in the message
  - may be amplified by further messages
- Very few effective countermeasures exist and B is not aware that the attack is going on till it starts to receive messages
- A structural vulnerability, it depends not upon the pieces but upon the composition

Design approaches

When designing and building a system two approaches may be adopted

a) pretend there are no vulnerabilities in the components (penetrate and patch)

b) be aware that there are vulnerabilities and try to anticipate them even if we still do not know which vulnerabilities (proactive approach)

Penetrate and patch

- Vulnerabilities have not been anticipated
- Since we have assumed that components are free from vulnerabilities, a vulnerability should be removed as soon as it is discovered.
- There is a competition between
  - discovering and exploiting vulnerabilities
  - patching the system to remove them

Penetrate and patch

- Vulnerabilities have not been anticipated
- Since we have assumed that components are free from vulnerabilities, a vulnerability should be removed as soon as it is discovered.
- There is a competition between
  - discovering and exploiting vulnerabilities
  - patching the system to remove them

Security Patch (wikipedia)

- A security patch is a change applied to an asset (OS, application, ...) t correct the weakness described by a vulnerability.
- This corrective action will prevent successful exploitation and remove or mitigate a threat's capability to exploit the vulnerability to attack an asset.
- Security patches are the primary method of fixing security vulnerabilities in software. Currently Microsoft releases its security patches once a month, and other operating systems and software projects have security teams dedicated to releasing the most reliable software patches as soon after a vulnerability announcement as possible.
- Security patches are closely tied to responsible disclosure.

Patches: problem

- Any patching updates a software component and changes its behaviour
- The change may influence the users
- A patch can be applied only after checking that the changes can be accepted
- Sometime a patch cannot be applied, eg certification of a system where the software is just one component

Number of vulnerabilities discovered

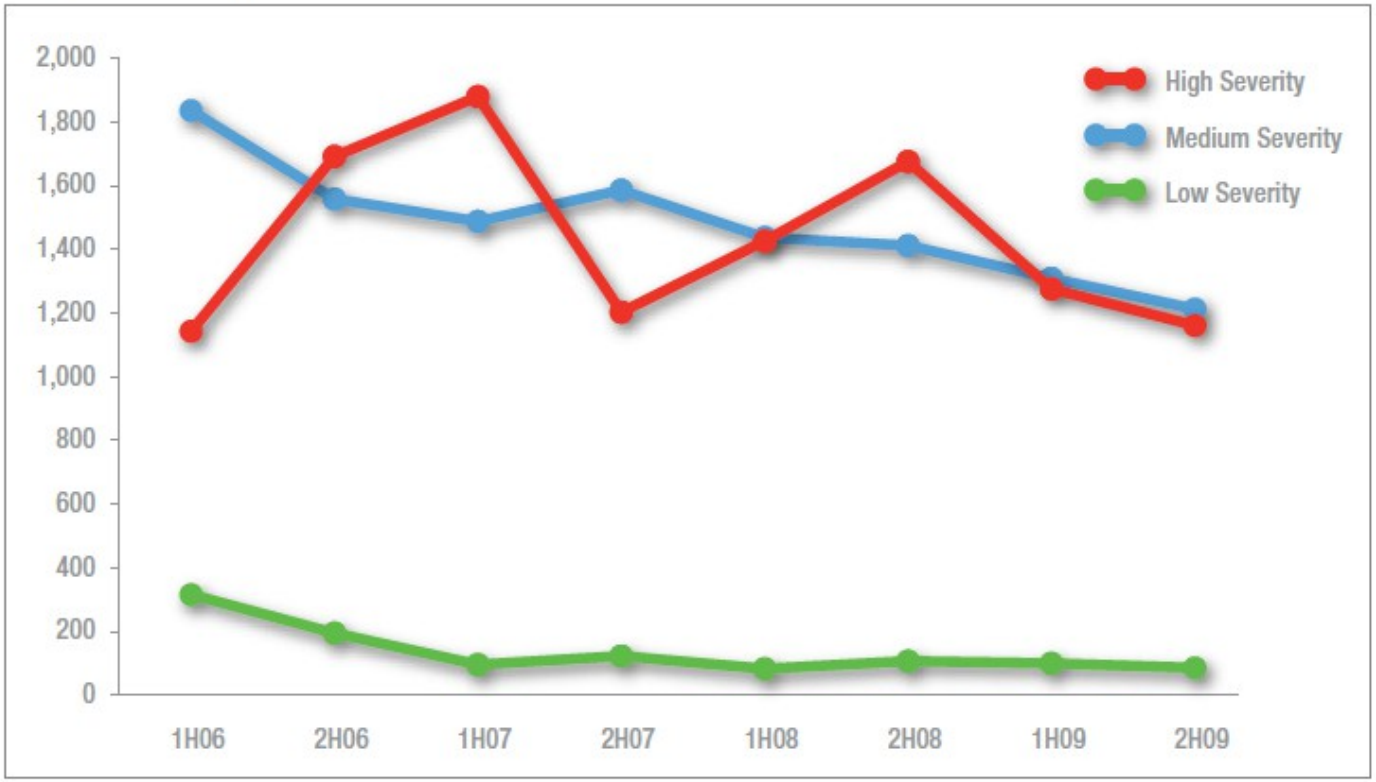


Figure 1: Industry-wide vulnerability disclosures by severity, by half-year from the first half of 2006 through the second half of 2009

Browser vulnerability

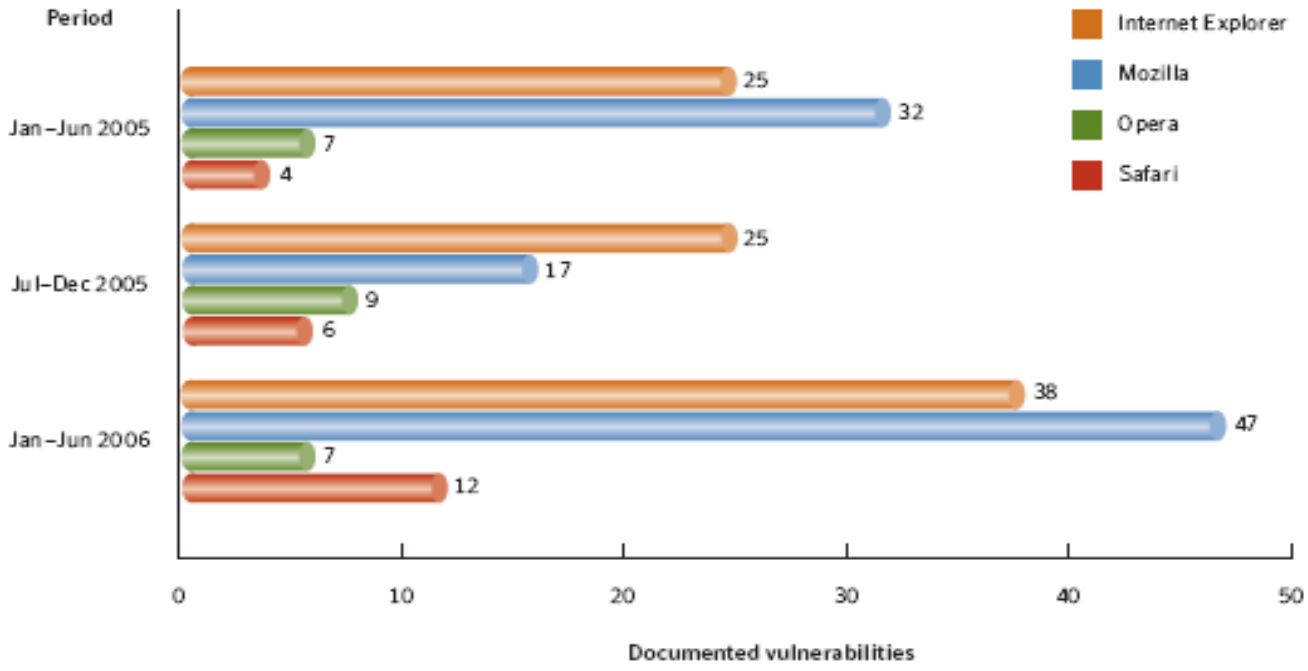
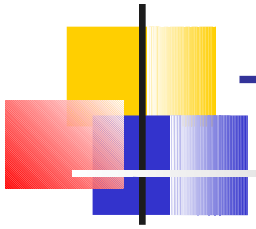
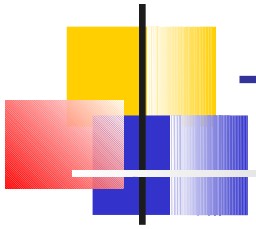


Figure 5. Web browser vulnerabilities
Source: Symantec Corporation



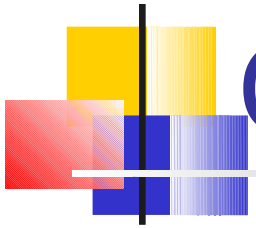
Top 10 Vulnerabilities - Windows Systems

1. Internet Information Services
2. Microsoft SQL Server
3. Windows Authentication
4. Internet Explorer
5. Windows Remote Access Services
6. Data Access Components(MDAC
7. Windows Scripting Host
8. Outlook and Outlook Express
9. Peer to Peer File Sharing
10. Simple Network Management



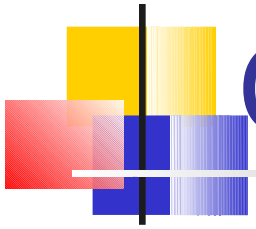
Top 10 Vulnerabilities - Unix Systems

1. BIND Domain Name System
2. Remote Procedure Calls (RPC)
3. Apache Web Server
4. Accounts with No Passwords or Weak Passwords
5. Clear Text Services
6. Sendmail
7. Simple Network Management Protocol
8. Secure Shell (SSH)
9. Misconfiguration of NIS/NFS
10. Open Secure Sockets Layer (SSL)



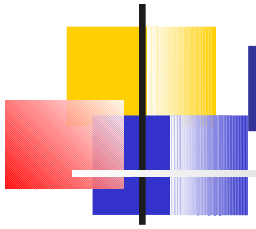
Other lists - I

- Top Vulnerabilities in Windows Systems
  - W1. Windows Services
  - W2. Internet Explorer
  - W3. Windows Libraries
  - W4. Microsoft Office and Outlook Express
  - W5. Windows Configuration Weaknesses
- Top Vulnerabilities in Cross-Platform Applications
  - C1. Backup Software
  - C2. Anti-virus Software
  - C3. PHP-based Applications
  - C4. Database Software
  - C5. File Sharing Applications
  - C6. DNS Software
  - C7. Media Players
  - C8. Instant Messaging Applications
  - C9. Mozilla and Firefox Browsers
  - C10. Other Cross-platform Applications



Other lists - II

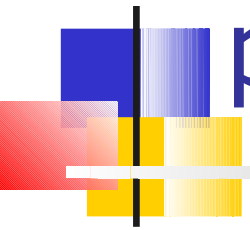
- Top Vulnerabilities in UNIX Systems
  - U1. UNIX Configuration Weaknesses
  - U2. Mac OS X
- Top Vulnerabilities in Networking Products
  - N1. Cisco IOS and non-IOS Products
  - N2. Juniper, CheckPoint and Symantec Products
  - N3. Cisco Devices Configuration Weaknesses

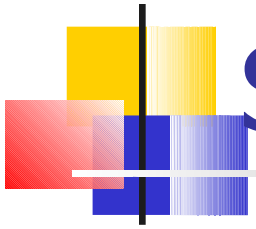


Hippa vulnerabilities

- Firewall and System Probing
- Network File Systems (NFS) Application
- Electronic Mail Attacks
- Vendor Default Password Attacks
- Spoofing, Sniffing, Fragmentation and Splicing
- Social Engineering Attacks
- Easy-To-Guess Password
- Destructive Computer Viruses
- Prefix Scanning (Illegal Modem)
- Trojan Horses

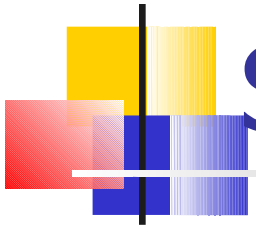
Life cycle of a vulnerability in a penetrate and patch world





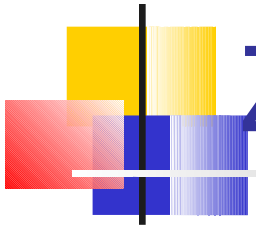
State of a vulnerability - 1

1. The vulnerability has been discovered
2. Both the vulnerability and an exploit that takes advantage of the vulnerability have been discovered
3. Both the vulnerability and a patch that removes the vulnerability have been discovered (a race with 2)
4. The vulnerability, the exploit and the patch have been discovered



State of a vulnerability - 2

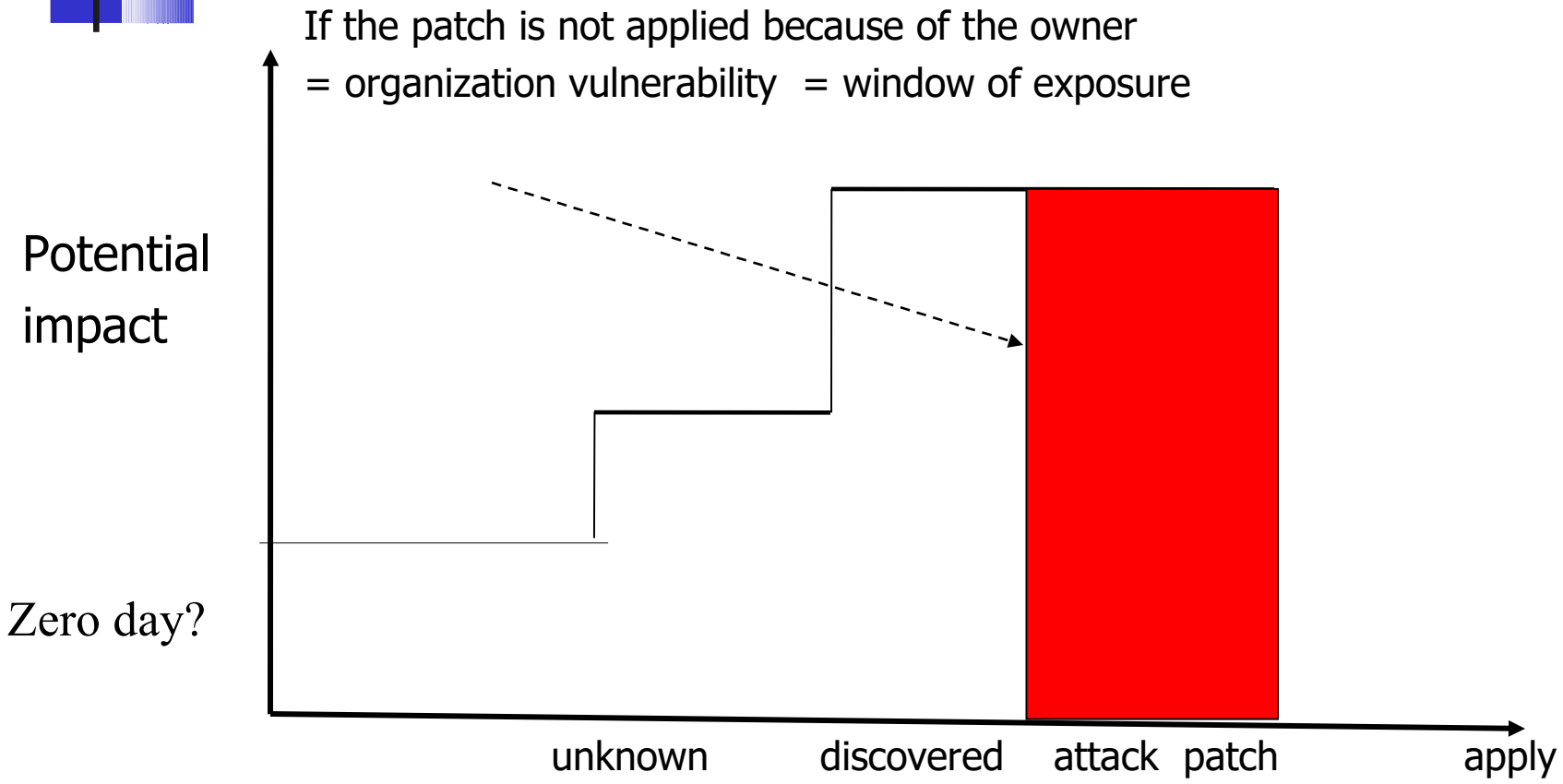
- Sometimes a system is attacked even if a vulnerability is in the last status
- It is well known that sometimes the owner of a system does not apply a patch even if it is available
- Asymmetry between the owner and the supplier (applying the patch is the owner responsibility rather than the supplier one)

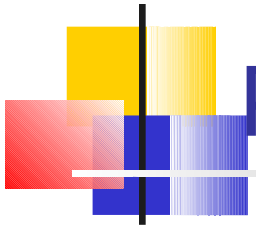


Zero day exploit

- An exploit for a vulnerability that has been discovered but not disclosed to all the users
- Sometimes those who discover a vulnerability sell it to those interested in attacking the system (black market of vulnerabilities)
- Can we design a system that resists attacks even when a vulnerability is discovered?

Potential impact of a vulnerability





Potential impact

- In the best case, a patch is available before an attack is known
- If the owner does not apply the patch, then any benefit of discovering the patch before the attack is lost
- It is the application of the patch not its definition that reduces the danger

Window of exposure

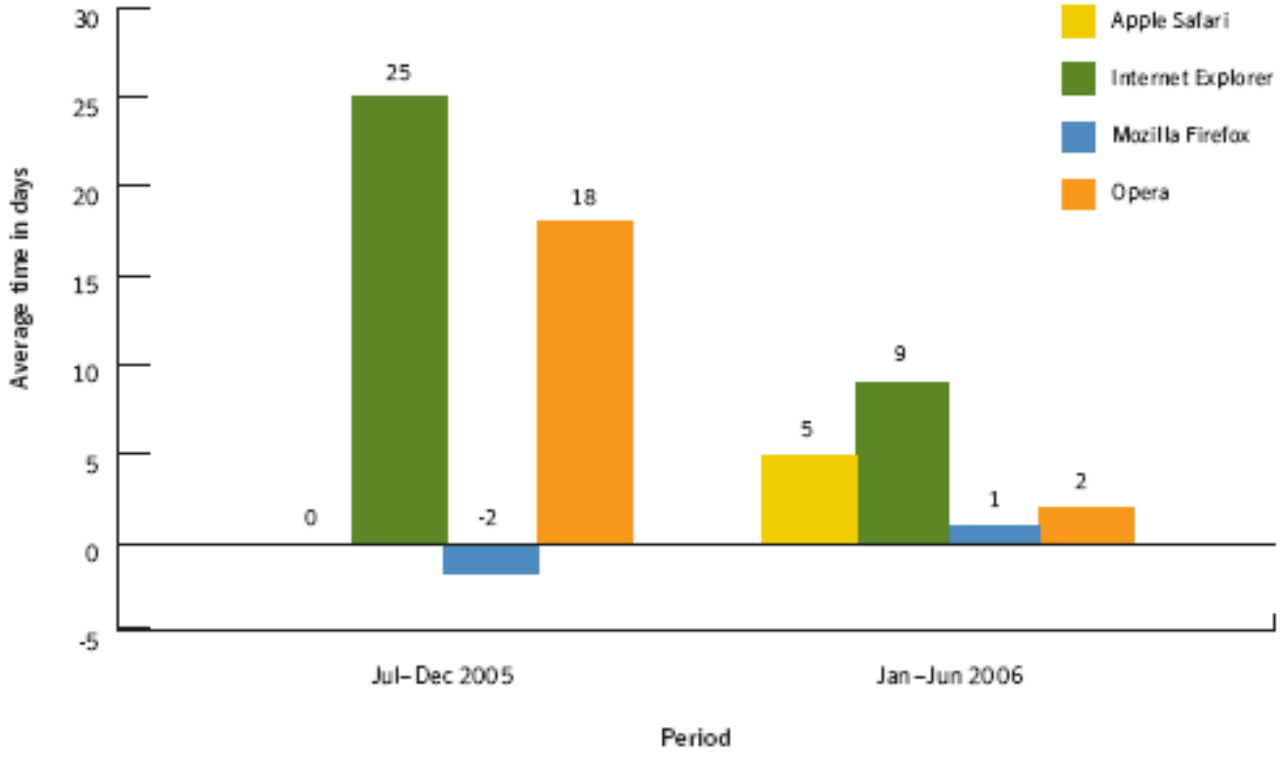
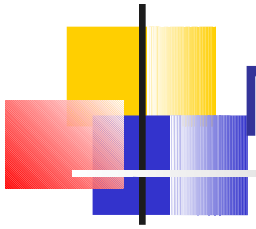
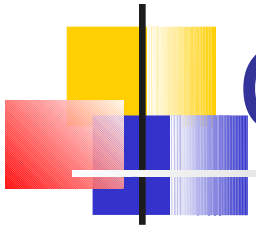


Figure 4. Web browsers window of exposure
Source: Symantec Corporation



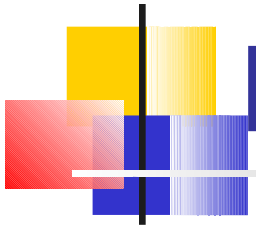
Number of vulnerability vs quality

- The number of vulnerabilities discovered = known in a component is always lower than existent ones
- This number depends upon
  - the availability of the source code
  - the number of applications and of people using the tool
  - the expected benefit of an attack against the tool
- If a tools is scarcely used, very few vulnerabilities are known but this does not imply they do not exist
- The number of disclosed vulnerability cannot be used as a quality index



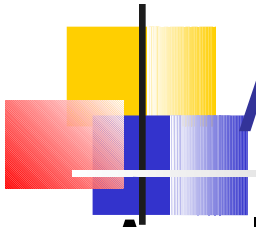
Genetic difference

- A system is more robust if it composes components from distinct suppliers
- The joint existence of vulnerabilities and a monopoly in the supplying of components can results in several problems because all the instances of a component are affected by the same vulnerabilities
- How much configuration influences vulnerabilities (??!!)



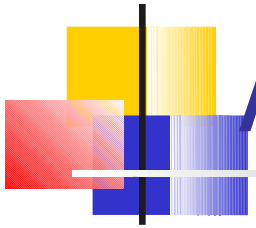
Defence in depth

- Any system component can be affected by a vulnerability
- A security expert
  - Does not need to know any vulnerability
  - Can design a system so that the discovery of a vulnerability in a component does not make the whole system useless
  - Layered defence or defence in depth = redundancies and diversities in the controls
- Alternative approach from the application of a patch



Adopted Approach -I

- A solution that tries to anticipate any vulnerability in any component has an huge cost
- Hence some vulnerabilities cannot be anticipated
- According to their potential impact we want to understand which vulnerabilities
  - should be accepted
  - should be anticipated
  - Should be patched asap
- Problem: how to classify each vulnerability

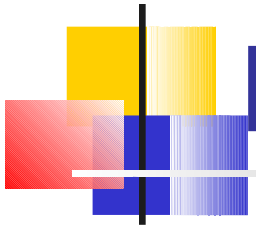


Adopted Approach - II

- The classification (handling) of a vulnerability depends upon the corresponding risk
- Risk
  1) Average impact if the vulnerability is successfully exploited
  2) Risk of a vulnerability = F($P_{attsucc,}$ Imp)
     - $P_{attsucc}$ = probability of a successful attack
     - Imp = impact due to a successful attack

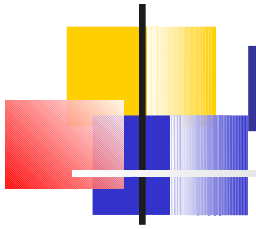
Adopted Approach - III

- $P_{attsucc}$ is a function of several parameters
  - Threat agents that
    - are interested in implementing the attack
    - Have the know how and the resources to implement the attack
  - Complexity of the implementation (automated or not?)
  - Are there other vulnerabilities that can be exploited to reach the same goal?
  - Are these attacks more or less complex?



Probability and impact

- A detailed evaluation of the success probability of an attack is extremely complex
  - No historical information available
  - Quick evolution of hardware and software
  - Human factor
- Similar problems are to be faced for the impact because of factors such has loss of new clients, damage to the reputation etc



Probability - II

- Sometimes both the success probability and the impact are evaluated in an approximated way

  {low, medium, high} oppure

  {low, medium-low, medium …}
- We also need a risk matrix that approximates the risk given the input approximated values

Risk Matrix

Prob Impact	VL	L	M	H	VH
VH	H	H	H	VH	VH
H	M	H	H	H	H
M	L	L	M	M	M
L	L	L	L	M	M
VL	VL	L	M	H	VH

A critical problem

- Any probability assumes some information about the past behavior of a system and of the attackers
- From this information we can estrapolate the future behavior under a continuity assumption
- A breakthrough in the technology for the attacker or the owner can invalidate the continuity assumption and results in distinct probabilities

Summing Up

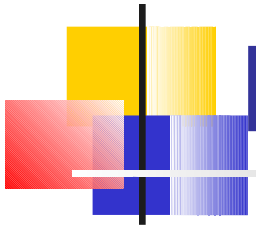
- A risk attitude is defined by two of four parameters
  - Penetrate and patch/Proactive (choose one)
  - Conditional/Unconditional (choose one)
- If a vulnerability is discovered

  a) conditional security = assess the risk and remove only
    - there is a non zero risk (Probsucc, Impact)
    - if it is cost effective

  b) unconditional security: remove
- Penetrate and patch: the number of critical vulnerabilities (there is a risk) is much higher than in proactive

Evaluating risk with no data

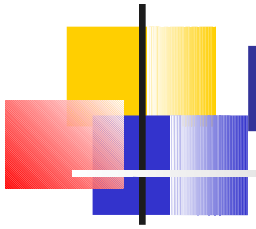
- The current research of our group is focused on the evaluation of risk when no data is available
- We have shown how to produce accurate and realistic data to replace historical one that, in general, is not available or is not pubblic



Risk Assessment

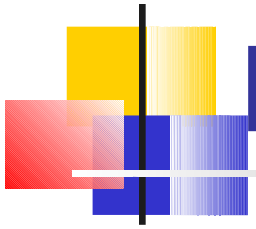
The formalization of the approach we have described, it includes:

1. Asset analysis
2. Vulnerability Analysis
3. Attack Analysis
4. Threat Analysis
5. Impact Analysis
6. Risk Evaluation and Management = which countermeasures are to be adopted



Risk Assessment

- The most modern approach to ICT security
- It consider the overall risk for an organization and it frames the risk due to ICT system with other risks
- A larger context has to be considered because ICT security should not be seen as a technological problem only



Return on investement ROI

- The security analyst should be able to justify the cost of the countermeasures that are selected to be implemented (deployed)
- A countermeasure should be adopted only for those vulnerabilities that enable attacks that have xxx
  - A large success probability
  - A large impact

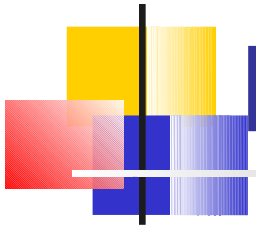
    = they have a large risk
- An interesting debate about (xxx= both) or (xxx=one of)

Return of investment

- It is the difference between
  - The overall risk before the countermeasures
  - The overall risk after the adoption of countermeasures
- The difference is due to countermeasures because they decrease the success probability or the impact of an attack
- The case where a vulnerability is removed (success probability$\rightarrow$0) is a particular one

Return of investment=Earning

- It is the difference between the ROI and the cost of countermeasures
- The difference should be larger than or equal to zero
- An alternative definition consider the ratio between the ROI and the countermeasure cost
- The ratio should be larger than 1



Next steps

- Asset analysis
- Security policy
- Vulnerability Analysis
- Possible countermeasures
- Attack Analysis
- Risk Management = countermeasure selection