

Security of Cloud Computing

Fabrizio Baiardi
f.baiardi@unipi.it



Syllabus

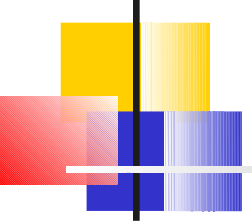
- Introduction to Security
 - Risk Analysis
 - Countermeasures
- Cloud Computing:
 - Supporting Technologies
 - Virtualization
 - Elasticity
 - Properties and Rules
- Security of Cloud Computing
 - New Threat Model
 - New Attacks
 - Countermeasures

Fully general

The course structure is the same with new content due to new attacks and problems

Clouds are the worst case!!!

The beginning of “Cloud Computing”

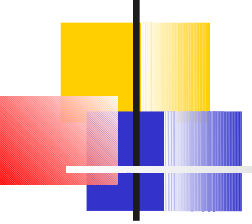
- 
-
- John McCarthy opined that “Computing may someday be organized as a public utility” - John McCarthy, MIT Centennial in 1961
 - “Comes from the early days of the Internet where we drew the network as a cloud... we didn’t care where the messages went... the cloud hid it from us” – Kevin Marks, Google
 - First cloud around networking (TCP/IP abstraction)
 - Second cloud around documents (WWW data abstraction)
 - The emerging cloud hides details to final users by abstracting infrastructure complexities of servers, applications, data, and heterogeneous platforms

Utility vs Cloud Computing

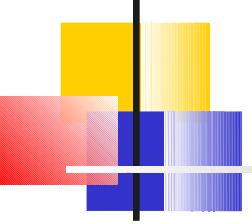


- Utility computing
 - customers receive computing resources from a service provider (hw and/or sw) and “pay by the drink,” as for electric service at home
 - requires a cloud-like infrastructure but in utility computing
 - focused on better economics. Corporate data centers are usually underutilized, with resources often idle 85 percent of the time = overprovisioning = more hardware than needed on average to handle peaks
 - allows companies to only pay for the computing resources they need, when they need them.
- Cloud computing is a broader concept that relates to the underlying architecture where services are designed. It may be applied equally to utility services and internal corporate data center

Fog vs Cloud Computing

- 
-
- Fog Computing extends Cloud computing and services to the edge of the network. Its distinguishing characteristics are its proximity to end-users, its dense geographical distribution, and its support for mobility. Services are hosted at the network edge or even end devices such as set-top-boxes or access points.
 - Fog reduces service latency, and improves QoS, resulting in superior user-experience.
 - Fog Computing supports emerging Internet of Everything (IoE) applications that demand real-time/predictable latency
 - Unlike traditional data centers, fog devices are geographically distributed over heterogeneous platforms, spanning multiple management domains.

A Working Definition of Cloud Computing



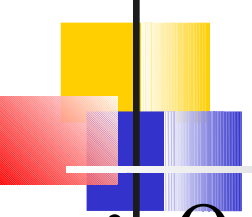
Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable and geographically distributed resources (e.g., networks, servers, storage, applications,) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

This cloud model promotes availability and is composed of

five essential **characteristics**,
three **service models**,
four **deployment models**.

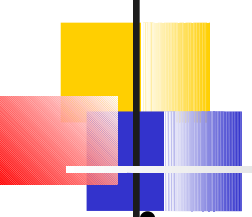
Design space

5 Essential Cloud Characteristics

- 
-
- On-demand self-service
 - Broad network access = web access
 - Resource pooling = Location independence through web / broad band access
 - Rapid elasticity
 - Measured service

Cloud computing is possible only because of web+broadband and it is not available if/when/where internet access is not available

Common Cloud Characteristics

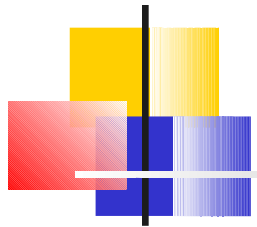
- 
-
- Massive scale
 - Homogeneity
 - Virtualization
 - Resilient computing
 - Low cost software
 - Geographic distribution
 - Service orientation
 - Advanced security technologies



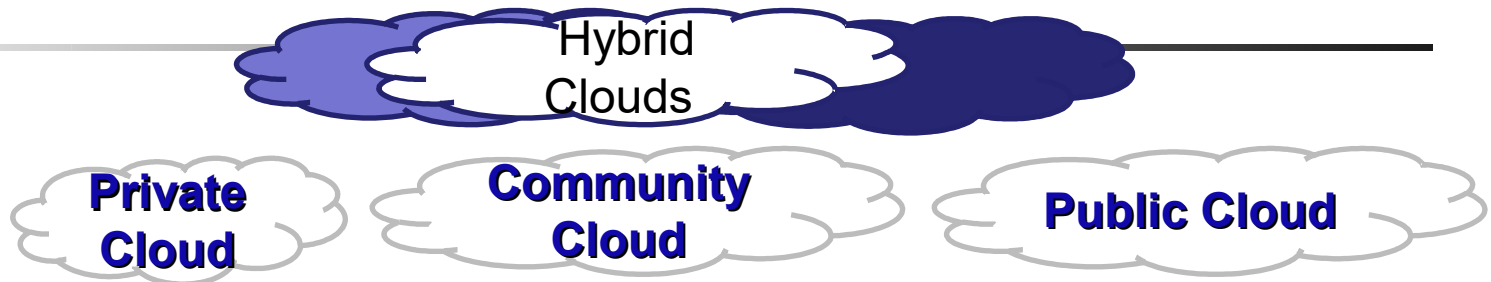
NIST framework and terms

- This course adopts and follows a framework developed by the National Institute of Standard and Technologies
- This framework has been and is used in the USA to drive the adoption of cloud computing in most of federal and state offices
- Focused on
 - the kind of access to the cloud system (service model)
 - the underlying architecture (deployment model)

The NIST Cloud Definition Framework



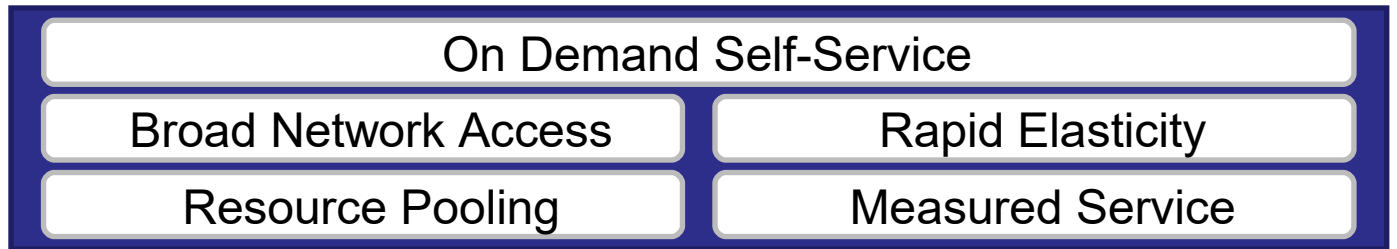
Deployment Models



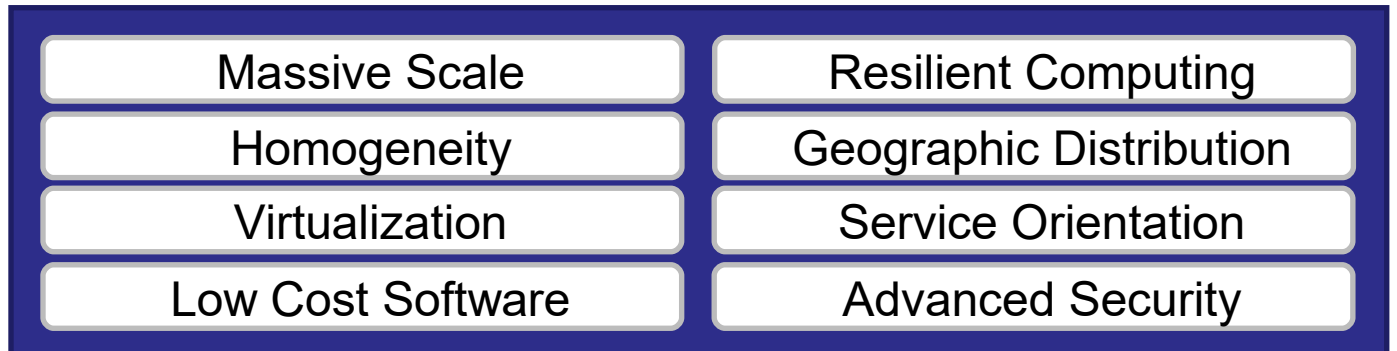
Service Models



Essential Characteristics



Common Characteristics





Cloud and Security - I

Economy and flexibility



**Private
Cloud**

Software as a
Service (SaaS)

**Community
Cloud**

Platform as a
Service (PaaS)

Public Cloud

Infrastructure as a
Service (IaaS)



Economy and flexibility



Cloud and Security - II

Complexity of security problems



**Private
Cloud**

Software as a
Service (SaaS)

**Community
Cloud**

Platform as a
Service (PaaS)

Public Cloud

Infrastructure as a
Service (IaaS)



Complexity of security problems



ICT Security & Risk

- 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 = Risk(probability, damage (or impact))
- 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

- Authentication = 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 (illegally) 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 (defect) of the target system or of its user
- 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



Our perspective

- Attack focused= a cost effective defense from attacks to an ICT system
- Why/Which/When attacks may be successful
- How the risk due to attacks can be managed (prevented, reduce their frequency or their 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 due to the high cost of the attacks they enable (they increase 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



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



Risk analysis and management

- Not all the attacks are worth preventing
- Economy driven solution
 - Which attacks
 - can be prevented
 - is worth preventing
- A complete and standard methodology is not currently available but several proposals in development
- Several partial solutions to be integrated



The steps of an intrusion

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



Local vs remote attack

- An attack is
 - Local if it can be executed provided that 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



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



Automated Attack and Malware

- Malware is a software designed to attack the system after its installation
- Sometimes this installation requires the user cooperation (phishing)
- A particular kind of exploit because it has to be executed on the system it attacks
- After its installation it may try to replicate itself onto other nodes (computer worm)

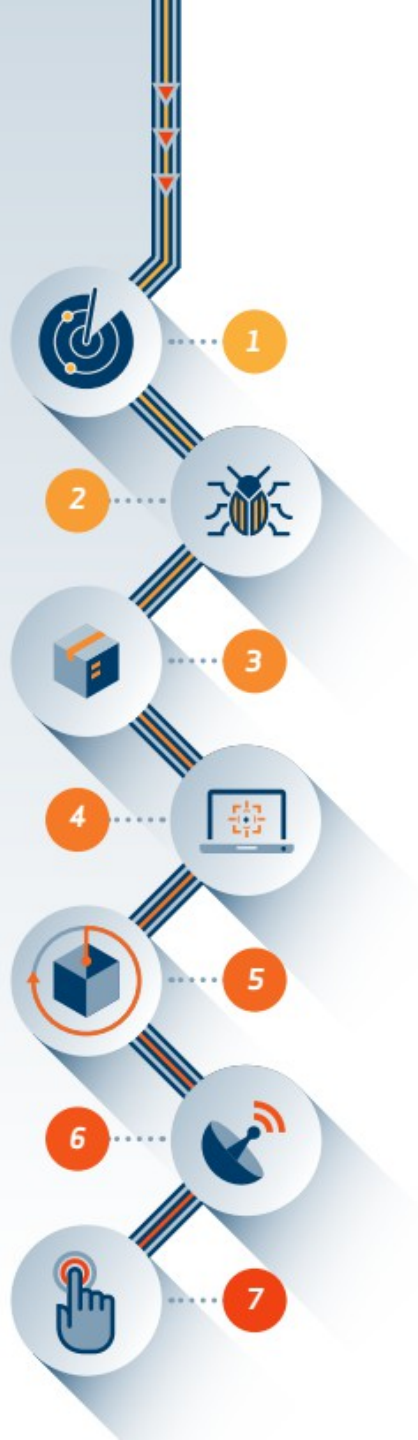
THE LOCKHEED MARTIN CYBER KILL CHAIN®

The Cyber Kill Chain® framework is part of the Intelligence Driven Defense® model for the identification and prevention of cyber intrusions activity. The model identifies what the adversaries must complete in order to achieve their objective.

Stopping adversaries at any stage breaks the chain of attack! Adversaries must completely progress through all phases for success; this puts the odds in our favor as we only need to block them at any given one for success. Every intrusion is a chance to understand more about our adversaries and use their persistence to our advantage.

The kill chain model is designed in seven steps:

- ▶ Defender's goal: understand the aggressor's actions
- ▶ Understanding is Intelligence
- ▶ Intruder succeeds if, and only if, they can proceed through steps 1-6 and reach the final stage of the Cyber Kill Chain®.





Description of the seven steps

Step 1: Reconnaissance. The attacker gathers information on the target

Step 2: Weaponization. The attacker creates a malicious payload .

Step 3: Delivery. The attacker sends the malicious payload to the victim by email or other means.

Step 4: Exploitation. The actual execution of the exploit, which is, again, relevant only when the attacker uses an exploit.

Step 5: Installation. Installing malware on the infected computer is relevant only if the attacker used malware. The installation is a point that takes months to operate.

Step 6: Command and control. The attacker creates a command and control channel in order to operate his internal assets remotely.

Step 7: Action on objectives. The attacker performs the steps to achieve his actual goals inside the victim's network.

RECONNAISSANCE *Identify the Targets*

ADVERSARY

The adversaries are in the planning phase of their operation. They conduct research to understand which targets will enable them to meet their objectives.

- ▶ Harvest email addresses
- ▶ Identify employees on social media networks
- ▶ Collect press releases, contract awards, conference attendee lists
- ▶ Discover internet-facing servers

DEFENDER

Detecting reconnaissance as it happens can be very difficult, but when defenders discover recon – even well after the fact – it can reveal the intent of the adversaries.

- ▶ Collect website visitor logs for alerting and historical searching.
- ▶ Collaborate with web administrators to utilize their existing browser analytics.
- ▶ Build detections for browsing behaviors unique to reconnaissance.
- ▶ Prioritize defenses around particular technologies or people based on recon activity.

1



WEAPONIZATION *Prepare the Operation*

ADVERSARY

The adversaries are in the preparation and staging phase of their operation. Malware generation is likely not done by hand – they use automated tools. A “weaponizer” couples malware and exploit into a deliverable payload.

- ▶ Obtain a weaponizer, either in-house or obtain through public or private channels
- ▶ For file-based exploits, select “decoy” document to present to the victim.
- ▶ Select backdoor implant and appropriate command and control infrastructure for operation
- ▶ Designate a specific “mission id” and embed in the malware
- ▶ Compile the backdoor and weaponize the payload

DEFENDER

This is an essential phase for defenders to understand. Though they cannot detect weaponization as it happens, they can infer by analyzing malware artifacts. Detections against weaponizer artifacts are often the most durable & resilient defenses.

- ▶ Conduct full malware analysis – not just what payload it drops, but how it was made.
- ▶ Build detections for weaponizers – find new campaigns and new payloads only because they re-used a weaponizer toolkit.
- ▶ Analyze timeline of when malware was created relative to when it was used. Old malware is “malware off the shelf” but new malware might mean active, tailored operations.
- ▶ Collect files and metadata for future analysis.
- ▶ Determine which weaponizer artifacts are common to which APT campaigns. Are they widely shared or closely held?

2



DELIVERY *Launch the Operation*

ADVERSARY

The adversaries convey the malware to the target. They have launched their operation.

- ▶ Adversary controlled delivery:
 - ▶ Direct against web servers
- ▶ Adversary released delivery:
 - ▶ Malicious email
 - ▶ Malware on USB stick
 - ▶ Social media interactions
 - ▶ “Watering hole” compromised websites

DEFENDER

This is the first and most important opportunity for defenders to block the operation. A key measure of effectiveness is the fraction of intrusion attempts that are blocked at delivery stage.

- ▶ Analyze delivery medium – understand upstream infrastructure.
- ▶ Understand targeted servers and people, their roles and responsibilities, what information is available.
- ▶ Infer intent of adversary based on targeting.
- ▶ Leverage weaponizer artifacts to detect new malicious payloads at the point of Delivery.
- ▶ Analyze time of day of when operation began.
- ▶ Collect email and web logs for forensic reconstruction. Even if an intrusion is detected late, defenders must be able to determine when and how delivery began.

3



EXPLOITATION *Gain Access to Victim*

ADVERSARY

The adversaries must exploit a vulnerability to gain access. The phrase “zero day” refers to the exploit code used in just this step.

- ▶ Software, hardware, or human vulnerability
- ▶ Acquire or develop zero day exploit
- ▶ Adversary triggered exploits for server-based vulnerabilities
- ▶ Victim triggered exploits
 - ▶ Opening attachment of malicious email
 - ▶ Clicking malicious link

DEFENDER

Here traditional hardening measures add resiliency, but custom capabilities are necessary to stop zero-day exploits at this stage.

- ▶ User awareness training and email testing for employees.
- ▶ Secure coding training for web developers.
- ▶ Regular vulnerability scanning and penetration testing.
- ▶ Endpoint hardening measures:
 - ▶ Restrict admin privileges
 - ▶ Use Microsoft EMET
 - ▶ Custom endpoint rules to block shellcode execution
- ▶ Endpoint process auditing to forensically determine origin of exploit.

4



INSTALLATION *Establish Beachhead at the Victim*

ADVERSARY

Typically, the adversaries install a persistent backdoor or implant in the victim environment to maintain access for an extended period of time.

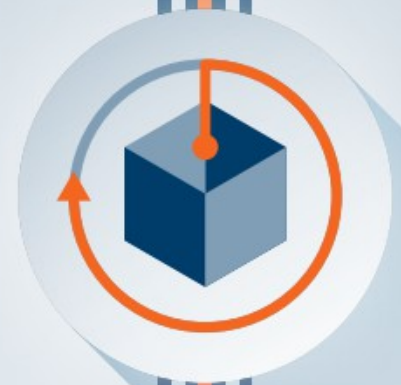
- ▶ Install webshell on web server
- ▶ Install backdoor/implant on client victim
- ▶ Create point of persistence by adding services, AutoRun keys, etc.
- ▶ Some adversaries “time stomp” the file to make malware appear it is part of the standard operating system install.

DEFENDER

Endpoint instrumentation to detect and log installation activity. Analyze installation phase during malware analysis to create new endpoint mitigations.

- ▶ HIPS to alert or block on common installation paths, e.g. RECYCLER.
- ▶ Understand if malware requires administrator privileges or only user.
- ▶ Endpoint process auditing to discover abnormal file creations.
- ▶ Extract certificates of any signed executables.
- ▶ Understand compile time of malware to determine if it is old or new.

5



COMMAND & CONTROL (C2) *Remotely Control the Implants*

ADVERSARY

Malware opens a command channel to enable the adversary to remotely manipulate the victim.

- ▶ Open two way communications channel to C2 infrastructure
- ▶ Most common C2 channels are over web, DNS, and email protocols
- ▶ C2 infrastructure may be adversary owned or another victim network itself

DEFENDER

The defender's last best chance to block the operation: by blocking the C2 channel. If adversaries can't issue commands, defenders can prevent impact.

- ▶ Discover C2 infrastructure thorough malware analysis.
- ▶ Harden network:
 - ▶ Consolidate number of internet points of presence
 - ▶ Require proxies for all types of traffic (HTTP, DNS)
- ▶ Customize blocks of C2 protocols on web proxies.
- ▶ Proxy category blocks, including "none" or "uncategorized" domains.
- ▶ DNS sink holing and name server poisoning.
- ▶ Conduct open source research to discover new adversary C2 infrastructure.

6



ACTIONS ON OBJECTIVES *Achieve the Mission's Goal*

ADVERSARY

With hands-on keyboard access, intruders accomplish the mission's goal. What happens next depends on who is on the keyboard.

- ▶ Collect user credentials
- ▶ Privilege escalation
- ▶ Internal reconnaissance
- ▶ Lateral movement through environment
- ▶ Collect and exfiltrate data
- ▶ Destroy systems
- ▶ Overwrite or corrupt data
- ▶ Surreptitiously modify data

DEFENDER

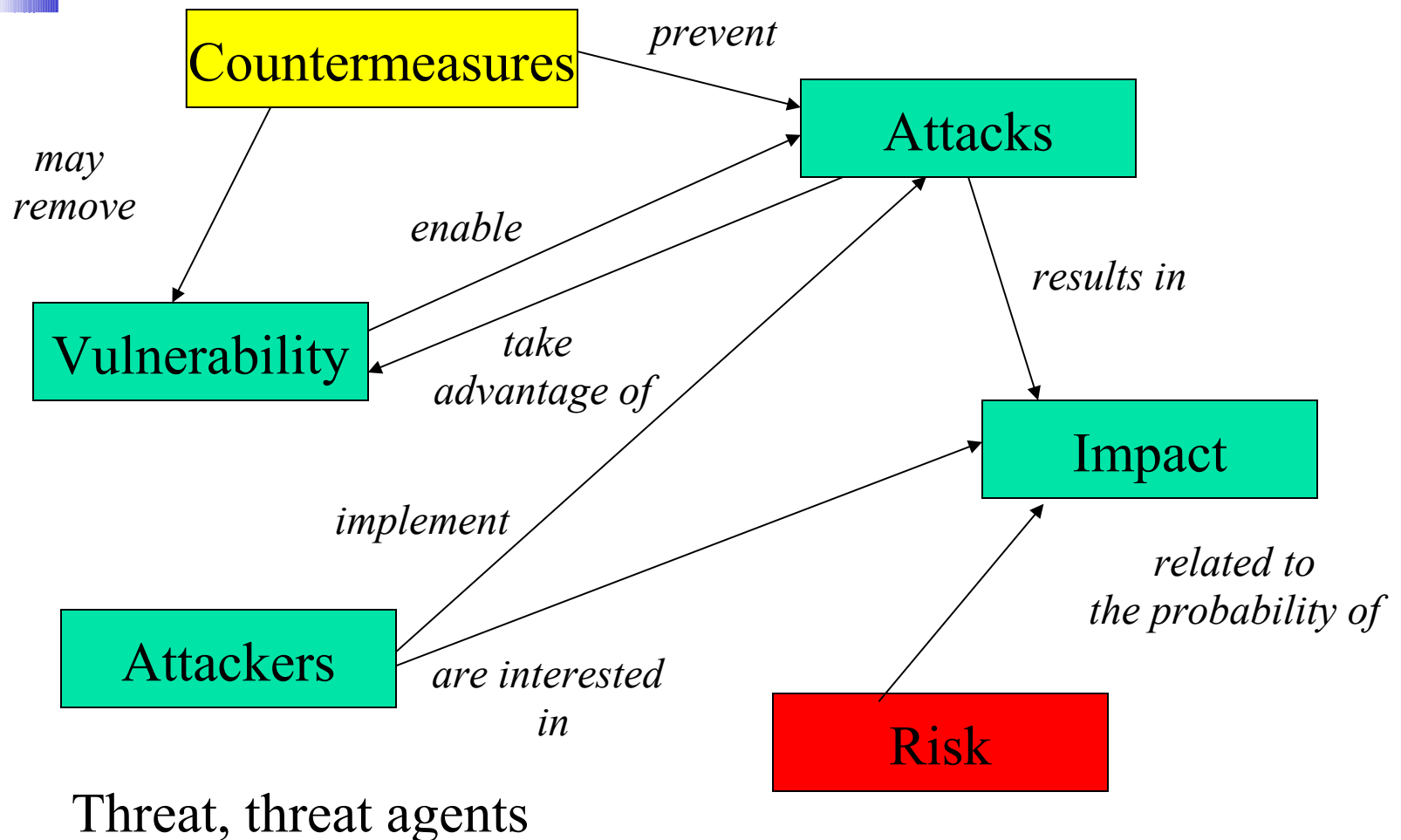
The longer an adversary has CKC7 access, the greater the impact. Defenders must detect this stage as quickly as possible by using forensic evidence – including network packet captures, for damage assessment.

- ▶ Establish incident response playbook, including executive engagement and communications plan.
- ▶ Detect data exfiltration, lateral movement, unauthorized credential usage.
- ▶ Immediate analyst response to all CKC7 alerts.
- ▶ Forensic agents pre-deployed to endpoints for rapid triage.
- ▶ Network package capture to recreate activity.
- ▶ Conduct damage assessment with subject matter experts.

7



Terminology and relations ...





Partial points view on sec– I

- Security = Confidentiality
⇔ Cryptography
- A set of algorithms 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 $\langle \text{user, resources, rights=operations on the res} \rangle$ 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 state, the probability of an unsafe behavior = $1/10^n$ the system safety increase with n
- If the same system has one state out 10^n that is not secure, the attacker can force the system to enter such a not secure state
- The system security depends upon the success probability of the attacker rather than on the overall number of states
- Attackers are intelligent rather than random



Some examples

- Vulnerability
- Attack
- Some countermeasures

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

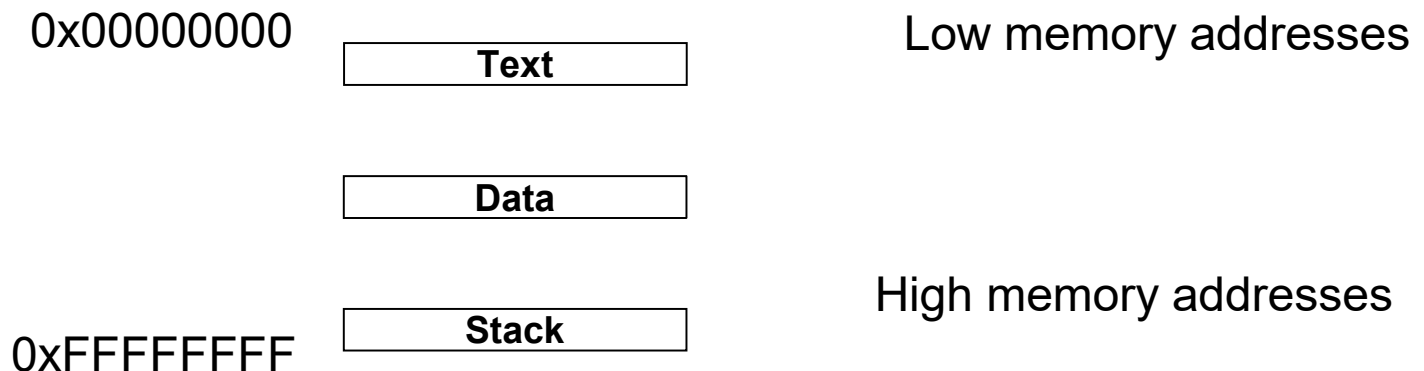


Buffer overflow

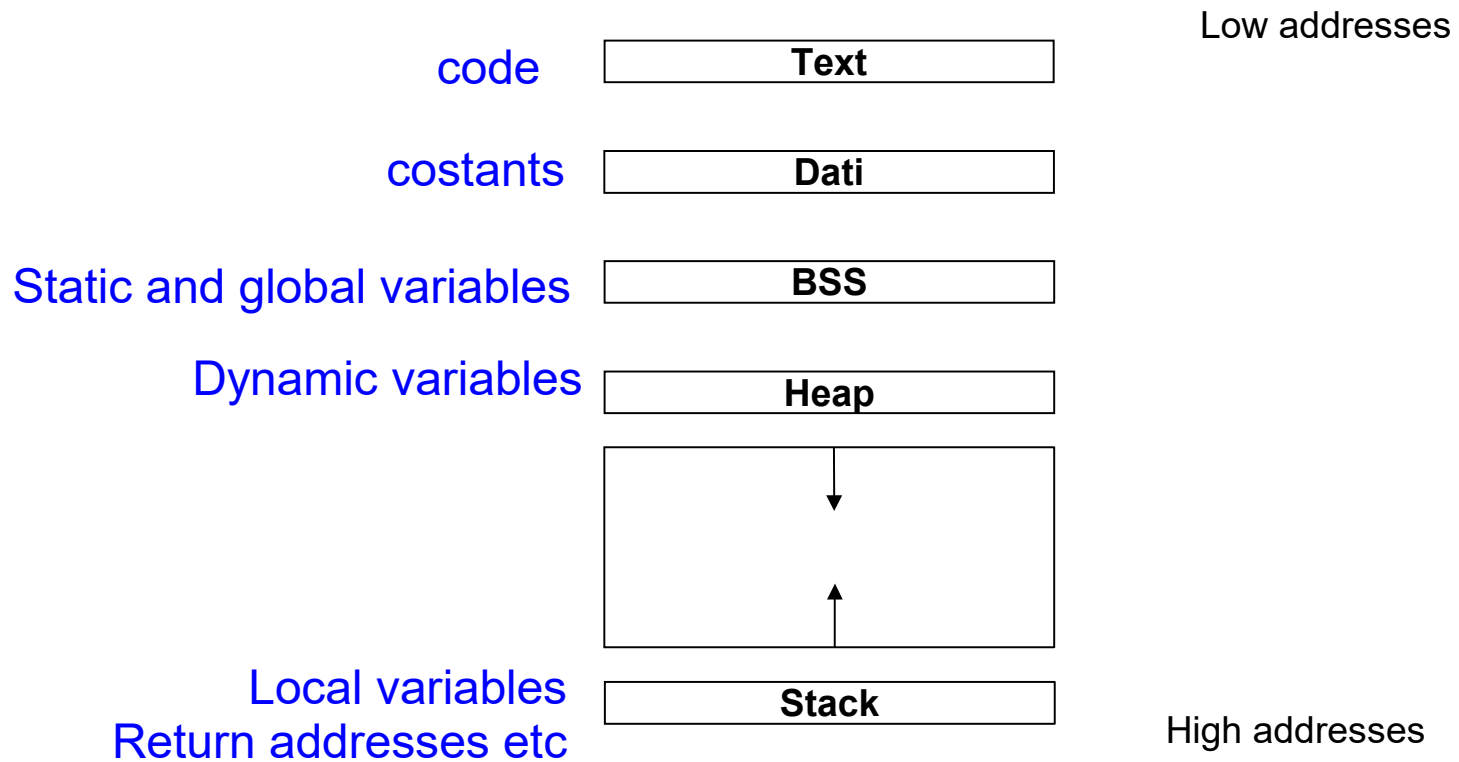
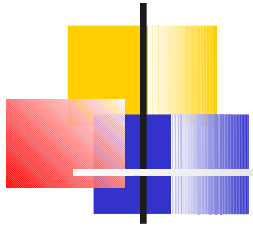
- The *buffer overflow* problem
 - the most common problem among all the vulnerability of C code
 - it does not arise in high level languages where the programmer is not involved in memory management or with strong data types
 - The most important security issue in the last 10 years (not replaced by 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, some program can be inserted (code injection) into a system that can, among other execute some shell command. 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 **stack, heap e bss (block started by symbol) static variables that are allocated by the compiler.**

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

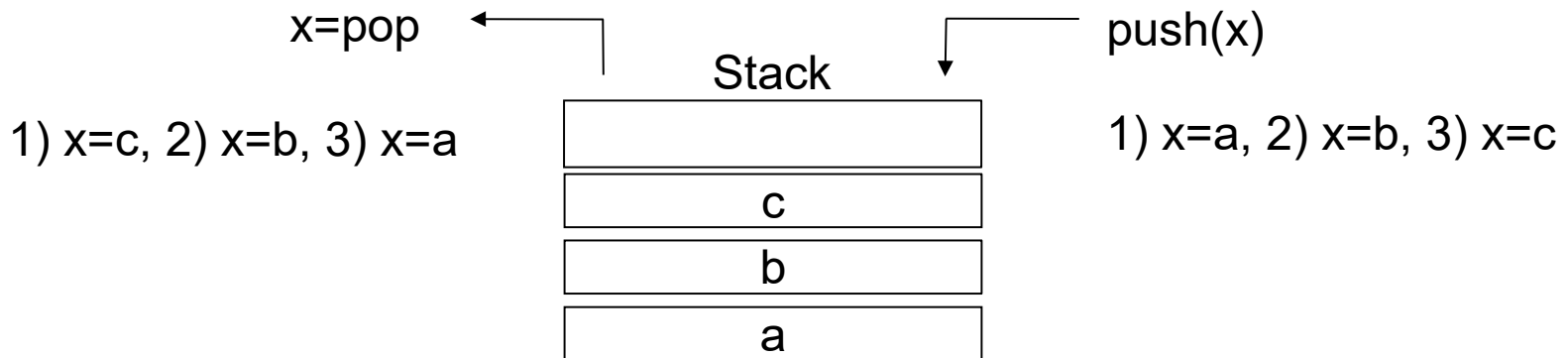


A process memory



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 partitioned into records (stack frame) one for each call



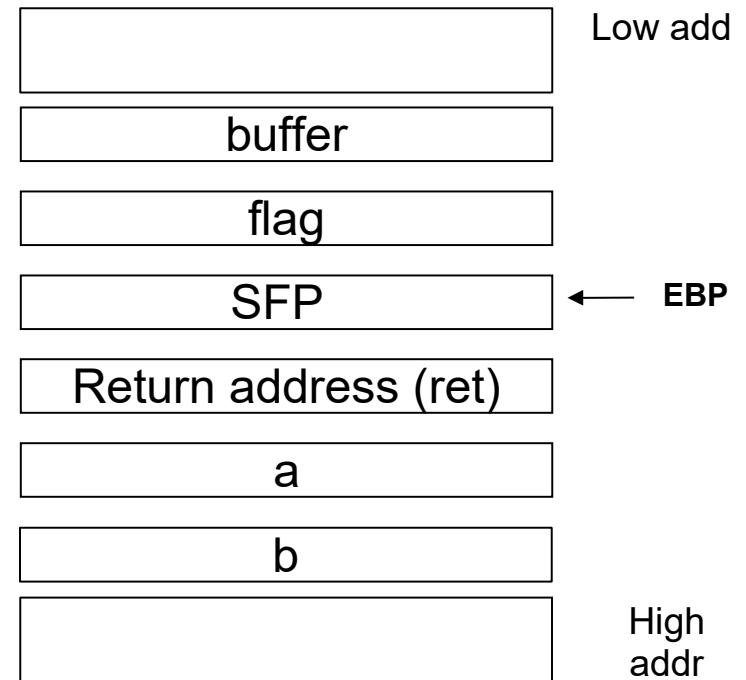


Stack and system registers

- 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*.

The stack frame

- Local variable of *test_function* are addressed by subtracting a displacement from **EBP** while the function parameters 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**).



Stack and system registers

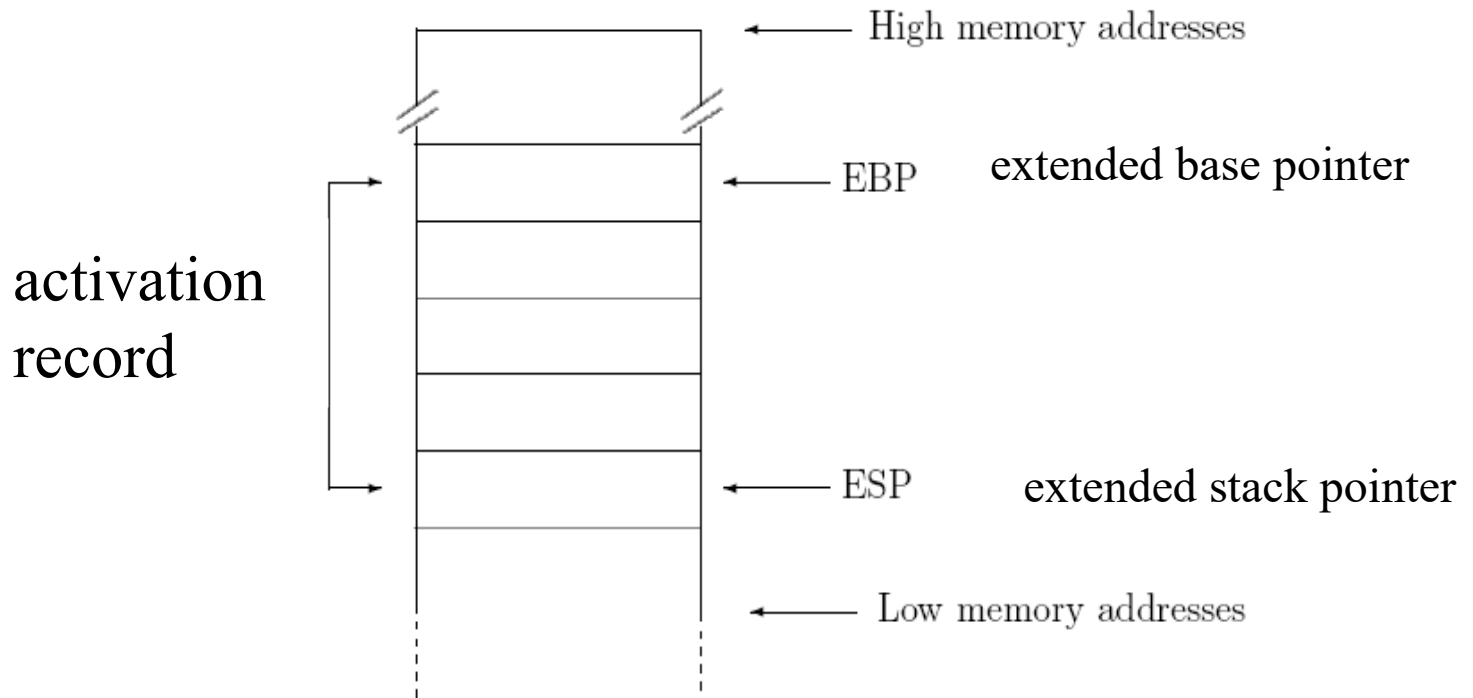
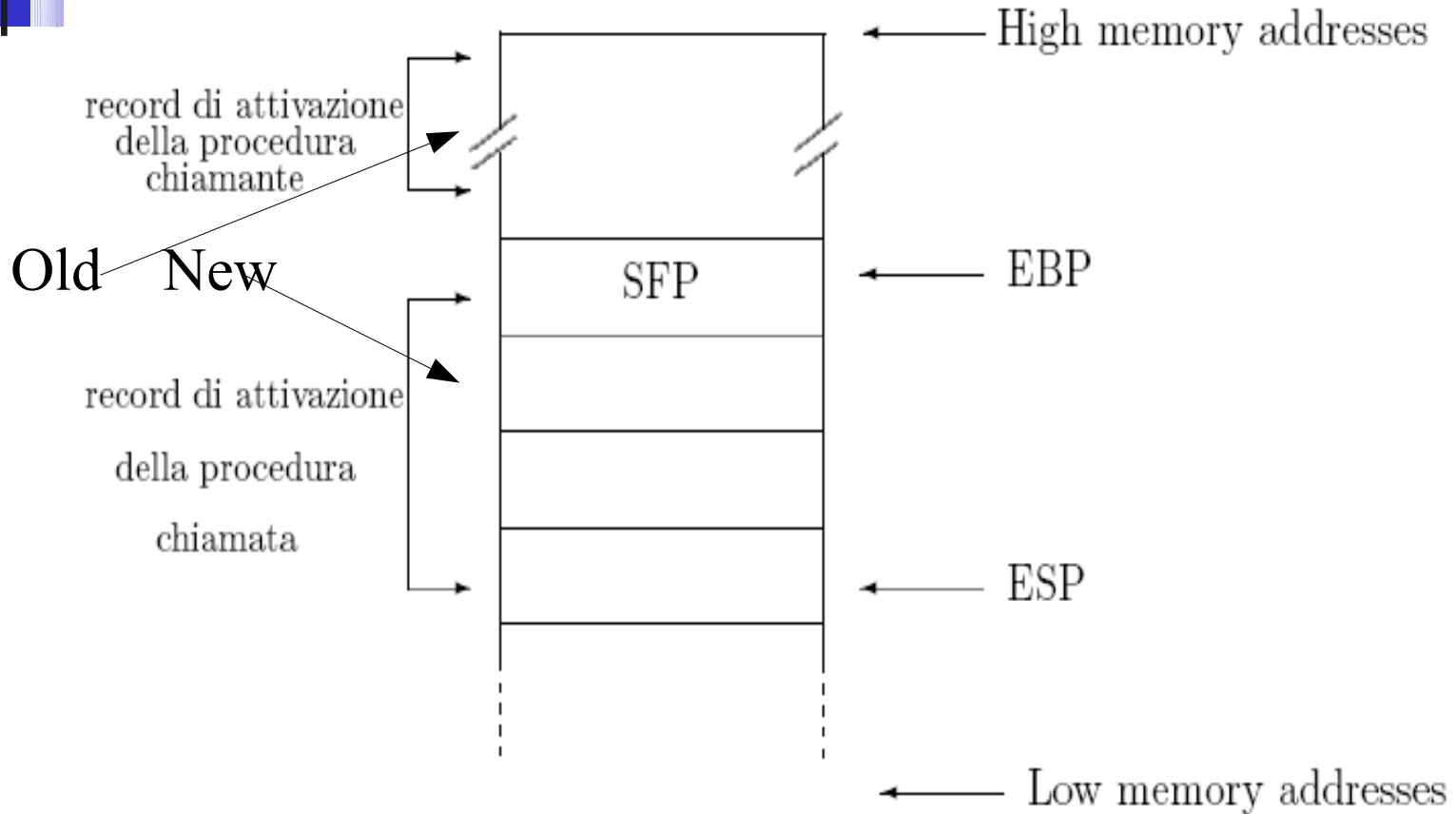


Figura 4: Stack pointer e frame pointer

Stack and system registers



C: an example

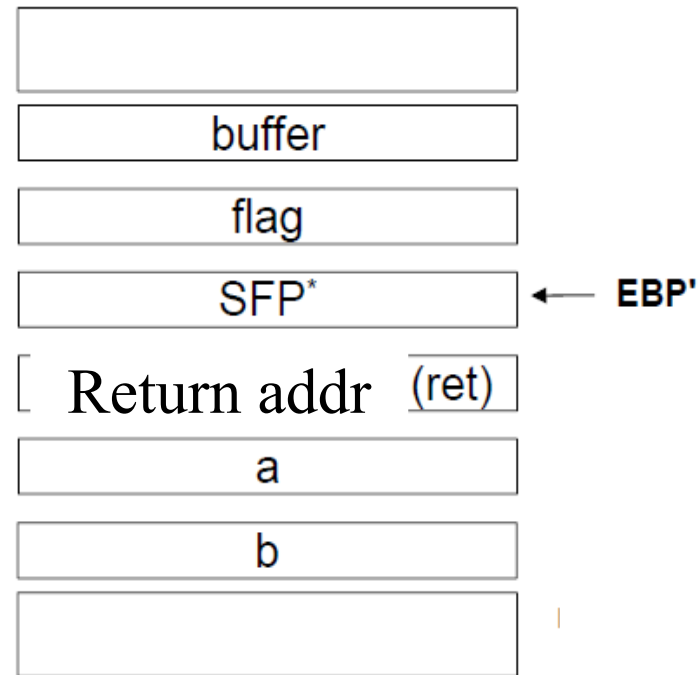
This is a simple example to see how all the stuff works

```
void test_function (int a, int b)
{
  char flag;
  char buffer[10];
}
```

```
int main()
{
  test_function (1,2);
  exit(0);
}
```

EIP →

Return address = EIP + 4 byte ←



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



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  
}
```

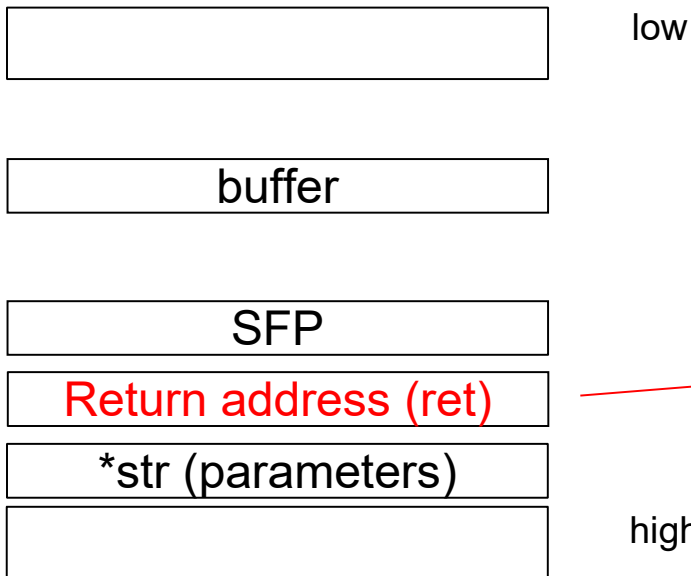
```
int main() {  
    char big_string[128];  
    int i;  
  
    for(i=0; i < 128; i++)  
    {  
        big_string[i] = 'A';  
    }  
    overflow_function(big_string);  
    exit(0);  
}
```

This results in an overflow!

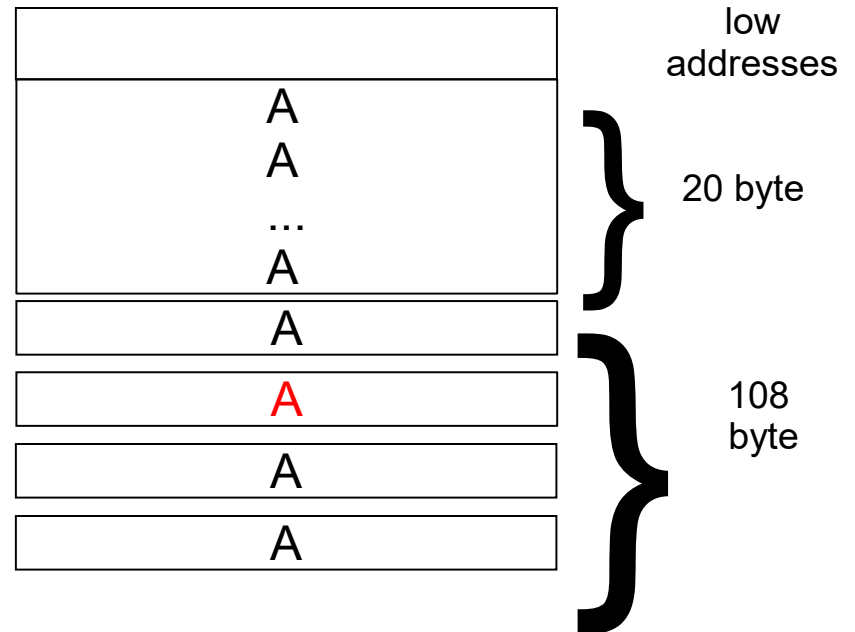
Segmentation fault

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 (malware)



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 memory
- Ad hoc checks in the compiler
- ASLR: address space layout randomization



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
- NX bit (the last one) introduced in AMD processors
- It does not work with Linux that stores some drivers in the stack to manage i/o devices



ASLR

- The starting point of the various segment is selected randomly
- The attacker cannot know in advance the starting address of data structures of interest
- The first step of the attack has to compute the starting address
- Attack more complex and slower



ASLR – entropy

Type	Description	Protection	Granularity of Rebasing
Free	Free space	Inaccessible	Not rebased
Code	Executable or DLL code	Read-only	15 bits
Static data	Within executable or DLL	Read-Write	15 bits
Stack	Process and thread stacks	Read-Write	29 bits
Heap	Main and other heaps	Read-Write	20 bits
TEB	Thread Environment Block	Read-Write	19 bits
PEB	Process Environment Block	Read-Write	19 bits
Parameters	Command-line and Environment variables	Read-Write	19 bits
VAD	Returned by virtual memory allocation routines	Read-Write	15 bits
VAD	Shared Info for kernel and user mode	Unwritable	Not rebased



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
 - ASLR supported by MMU translation 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 goal was resilience against physical attack against the network (attack = bombing)
- Main goal = availability
 - ⇒ Some mechanism defined to discover which nodes are alive and reachable
 - ⇒ No mechanism is available to guarantee \ (authenticate) the source of a message



Structural vuln: an Example

1. A node can send an ECHO message to check whether another node is alive and reachable, The receiver replies by returning the same message.
2. The sender can specify a partial IP address to broadcast a message to check a set of other nodes
3. There is no control on the fields of an IP packet a node sends



All together now ..

- R is a network with 1000 nodes, 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 overflowed by the ECHO messages

Distributed Denial of Service



Security as an holistic property

- The security of a composition is not implied by the one of its component
- The overall composition may be unsecure even when each component is secure
- In a virtual machine hierarchy the security of a machine may be destroyed by a vulnerability in an underlying machine



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



Security Patch (wikipedia)

- A security patch is a change applied to an asset (OS, application, ...) to 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

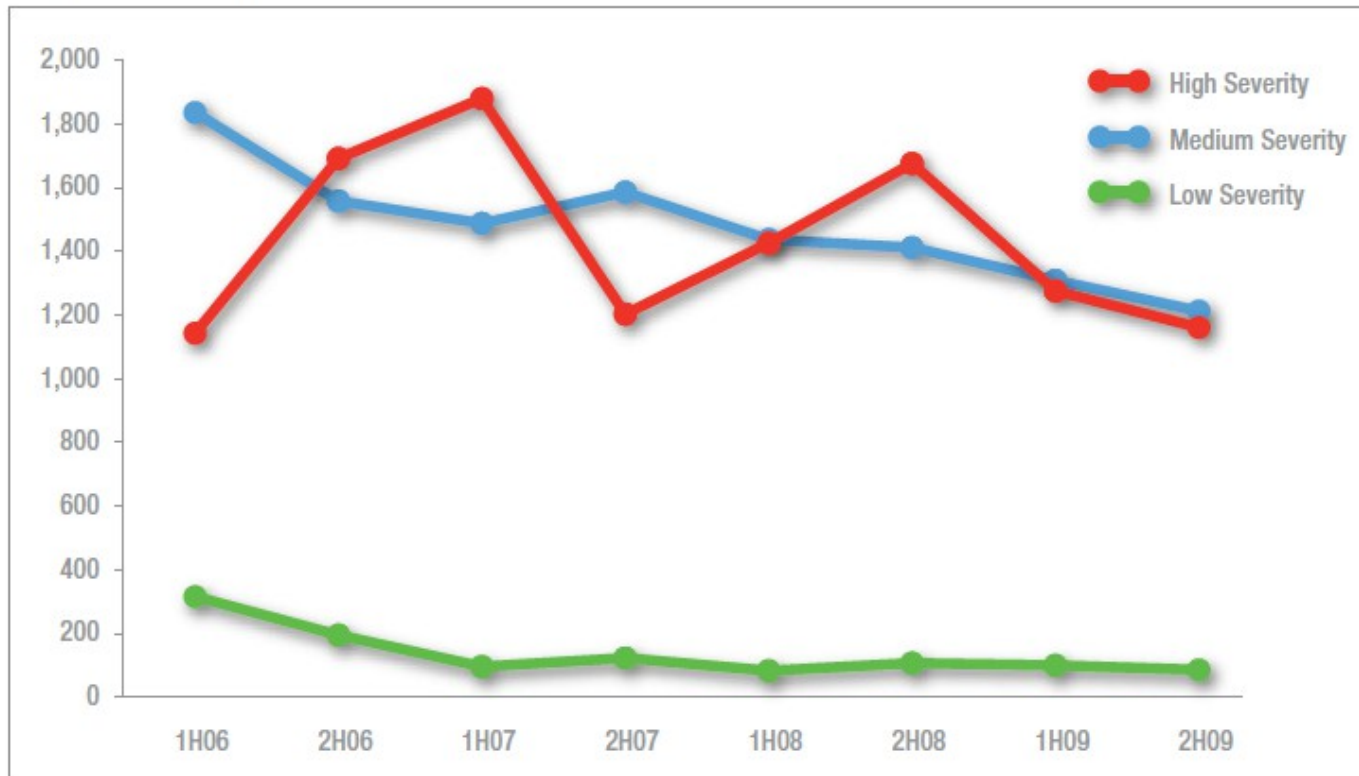


Figure 1: Industry-wide vulnerability disclosures by severity, by half-year from the first half of 2006 through the second half of 2009

Browser vulnerability

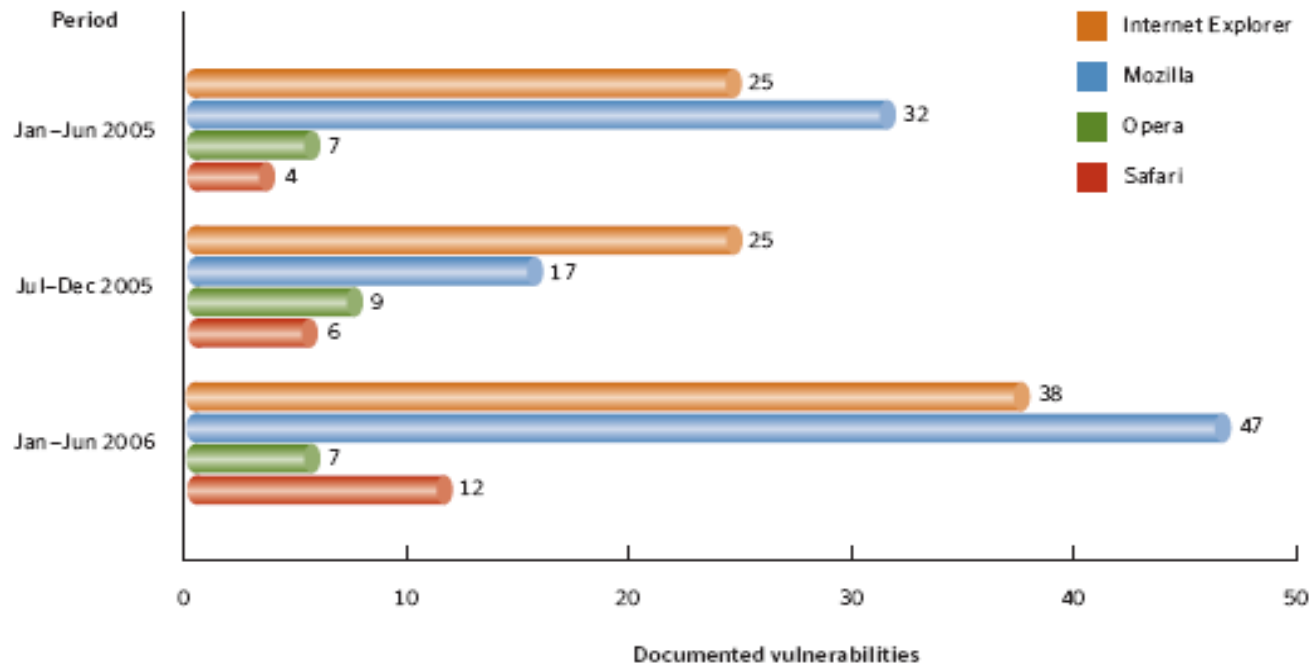
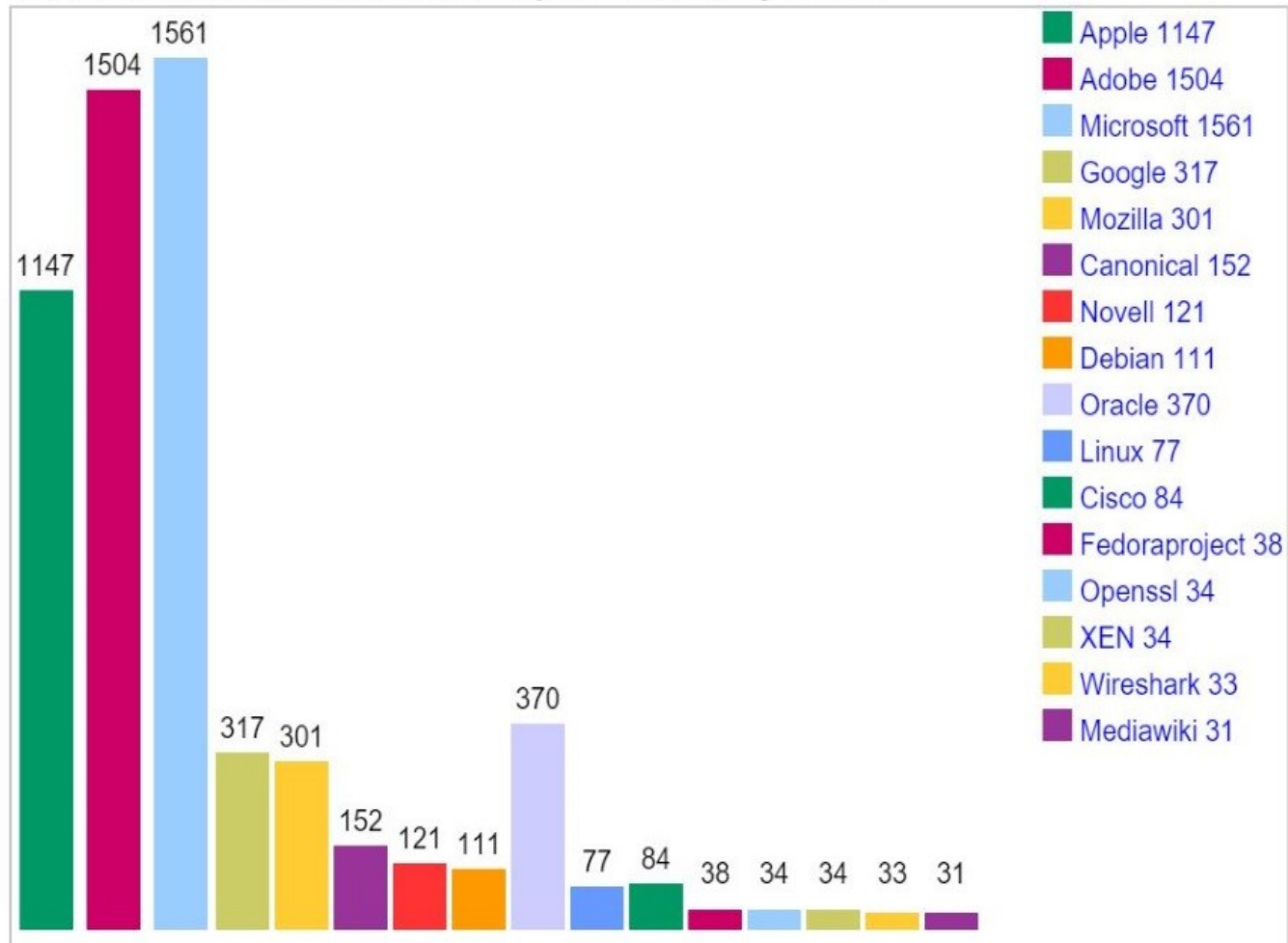


Figure 5. Web browser vulnerabilities

Source: Symantec Corporation

Vulnerability per vendor 2015

Total Number Of Vulnerabilities Of Top 50 Products By Vendor



Vulnerabilities per year

Navigation: Home, SCAP, Checklists, 800-53/800-53A, Product Dictionary, Impact Metrics, Data Feeds, Statistics, FAQs

Statistics Results Page (Refine Search)

[New Query](#)

Below are a table and graphs with data matching the characteristics you specified on the Statistics Query Page.

Search Parameters:

- Contains Software Flaws (CVE)
- Publication Start Date: January 2000
- Publication End Date: December 2016
- CVSS Version: 3

Total Matches By Year

Year	Matches
2000	1000
2001	1500
2002	2000
2003	1800
2004	2500
2005	5000
2006	7000
2007	6800
2008	5500
2009	5800
2010	4800
2011	4200
2012	5200
2013	5100
2014	8500
2015	6800
2016	6500

Percent Matches By Year



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

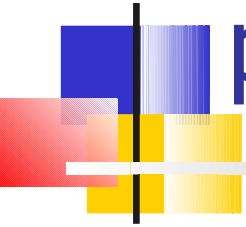
- Top Vulnerabilities in UNIX Systems
 - U1. UNIX Configuration Weaknesses
 - U2. Mac OS X
- Top Vulnerabilities in Networking Products
 - N1. Cisco IOS and non-IOS Products
 - N2. Juniper, CheckPoint and Symantec Products
 - N3. Cisco Devices Configuration Weaknesses



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)

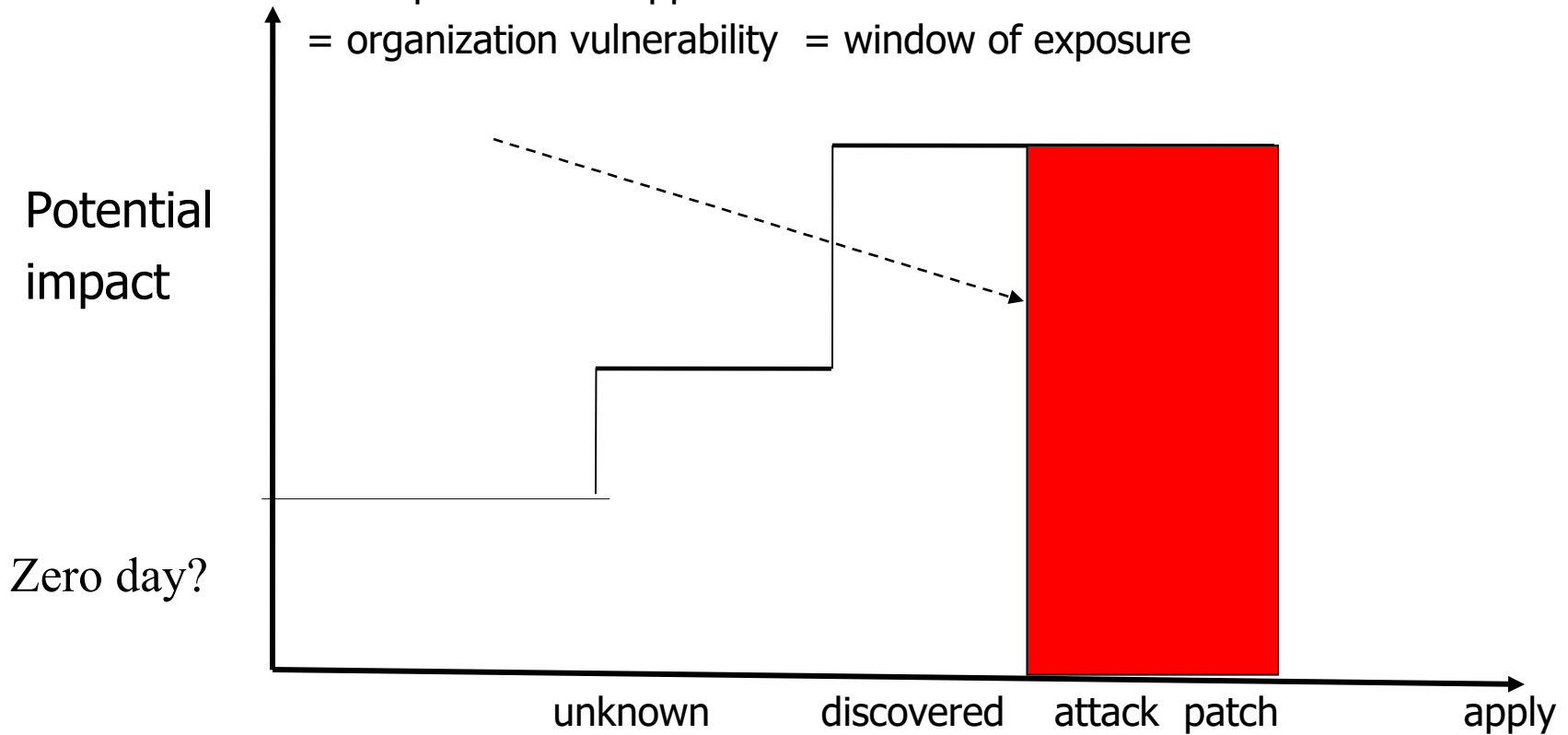


Zero day exploit

- 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

If the patch is not applied because of the owner
= organization vulnerability = window of exposure





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

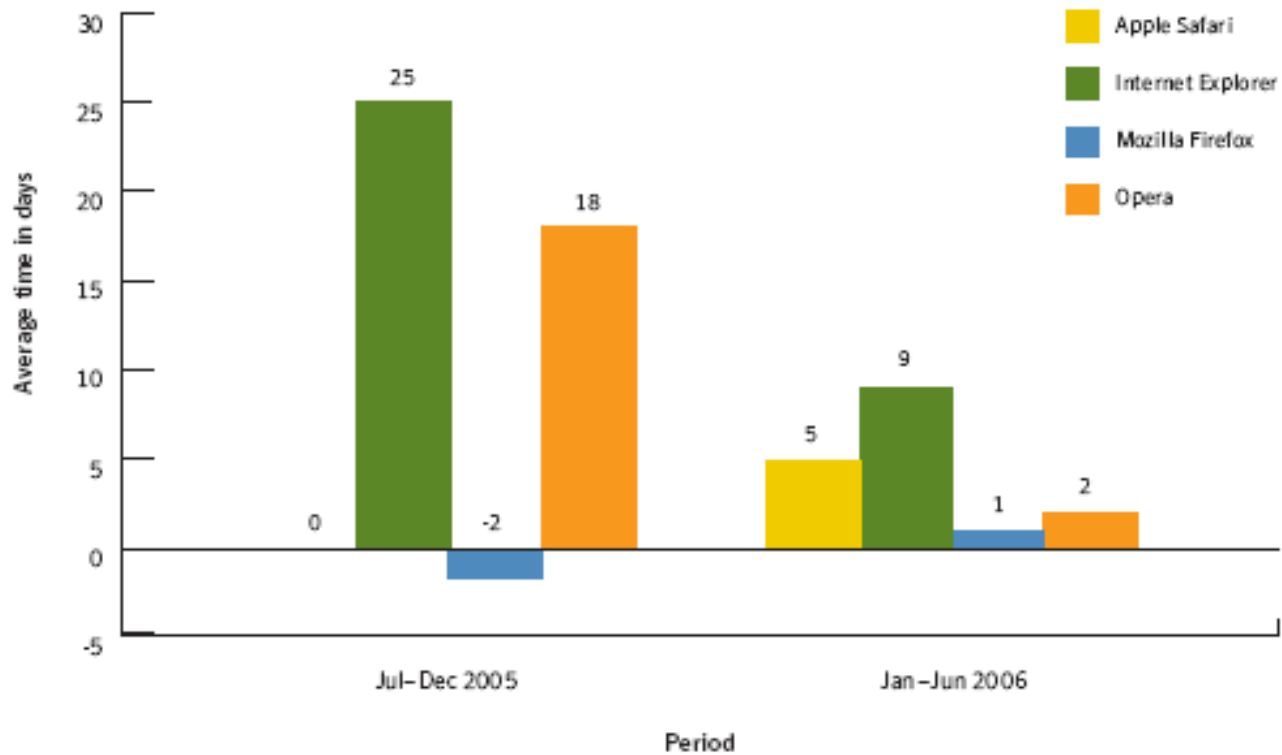


Figure 4. Web browsers window of exposure

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 tool 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 result 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 = $F(P_{\text{attsucc}}, \text{Imp})$
 - 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 as 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 extrapolate 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 public



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 investment 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 (0 success probability) is a particular case

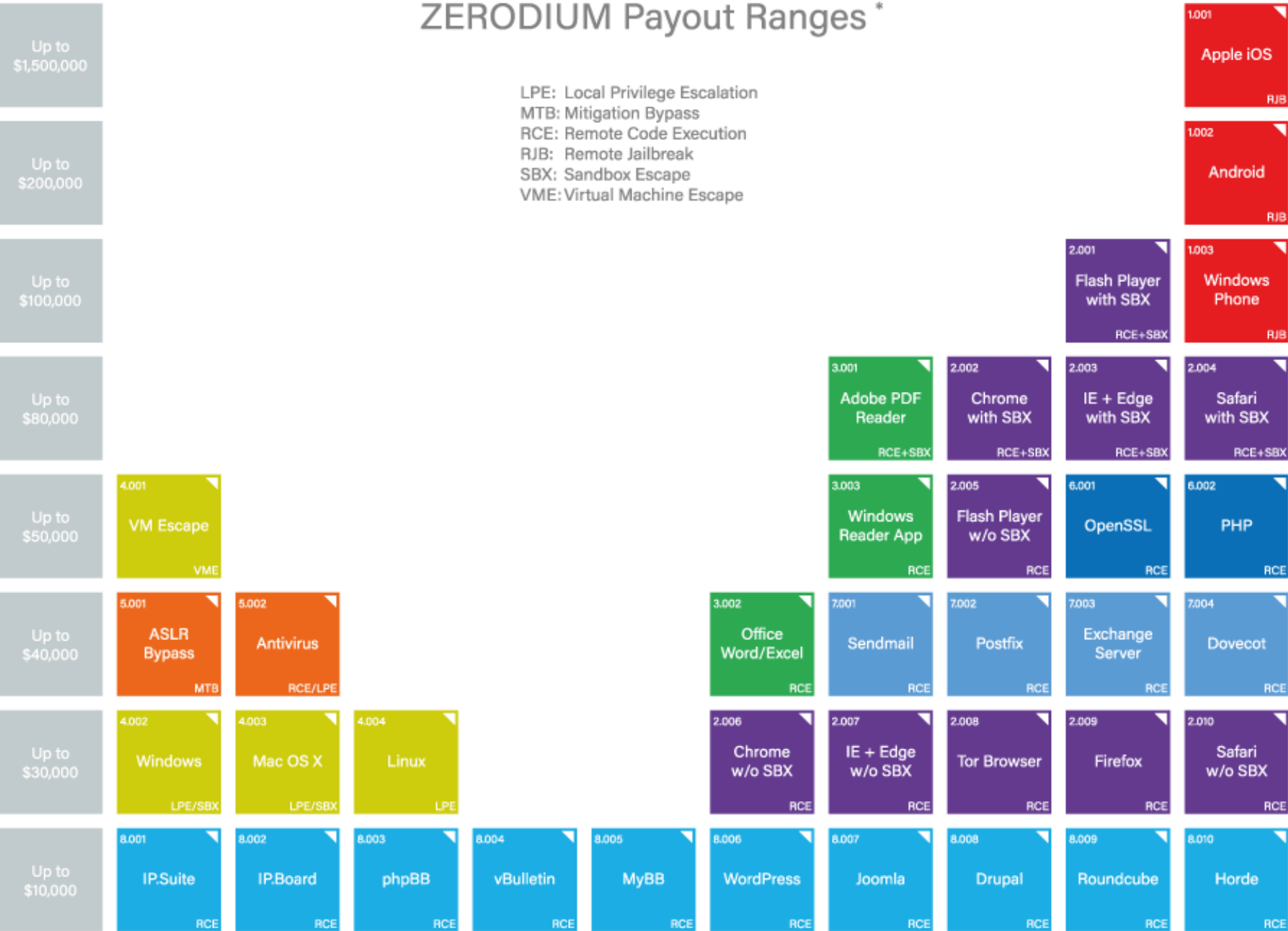


Return of investment=Earning

- 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

ZERODIUM Payout Ranges *

LPE: Local Privilege Escalation
MTB: Mitigation Bypass
RCE: Remote Code Execution
RJB: Remote Jailbreak
SBX: Sandbox Escape
VME: Virtual Machine Escape



* All payout amounts are chosen at the discretion of ZERODIUM and are subject to change or cancellation without notice.

2016/09 © zerodium.com



Other buyer ...

ADOBE READER	\$5,000-\$30,000
MAC OSX	\$20,000-\$50,000
ANDROID	\$30,000-\$60,000
FLASH OR JAVA BROWSER PLUG-INS	\$40,000-\$100,000
MICROSOFT WORD	\$50,000-\$100,000
WINDOWS	\$60,000-\$120,000
FIREFOX OR SAFARI	\$60,000-\$150,000
CHROME OR INTERNET EXPLORER	\$80,000-\$200,000
IOS	\$100,000-\$250,000



Next steps

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



Next Steps - II

- In principle, the security policy is a countermeasure
- In practice it is defined independently of, and before, the risk assessment because it also defines the goals of a system (what it is expected to do) and the rules to use it
- Its satisfaction is one of the goal of the assessment