Secure Applications

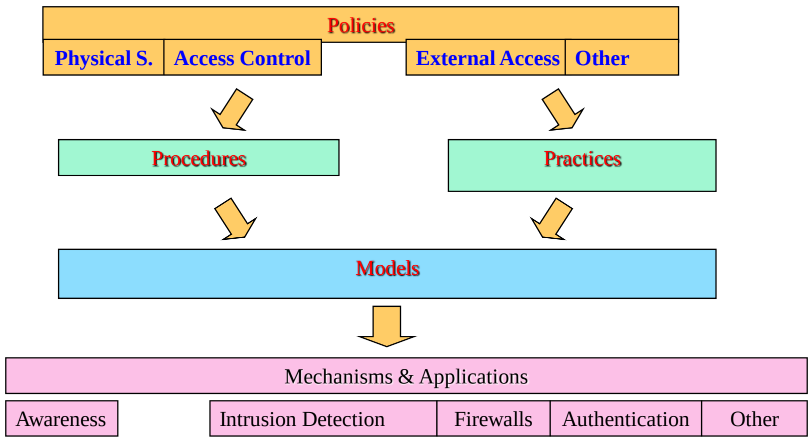
Security Architecture and the creation of secure applications

1

What is a Security Architecture?

- Security infrastructure
  - All entities (people, tools, objects, etc...) contributing to security
- Specifications
  - Policies
  - Procedures
  - Security practices
- Policy implementation mechanisms
  - Software/hardware mechanisms
  - People-oriented mechanisms

Security Architecture



Safety Specifications

- Policies
  - Define acceptable behavior
- Procedures
  - Plans, process, or methods on how to do them/enforce them
- Standards
  - Agreed rules in order to facilitate intercommunication

Policies: Where to start?

- Policies to be defined
- Resources that need to be protected
  - –Inventory:
    - Hardware, software, people and other resources
- Responsibilities
  - –Ownership of resources
  - –External elements

Policies

- Define the power/privileges of the agents
  - –Principle of least privilege
  - –Hardening
- Define the security requirements of a domain
  - –Security levels
  - –Required authorization
    - and respective minimum requirements for a satisfactory authentication
- Define the defense strategies and counter-attack tactics
  - –Defensive architecture
  - –Monitoring of critical activities or evidence of attacks
  - –Reaction to attacks or abnormal situations
- Define the universe of lawful and unlawful activities
  - –Everything that is not denied is permitted
  - –Everything that is not permitted is denied

Policies - Basic Properties

- Should be **supported by specific procedures**
- Realistic and enforceable
- Should define responsibilities
- Set for the long term
  - The procedures and practices must be constantly updated but policies should be kept for long periods
- Simple and Short
  - Many policies *vs* a single policy
  - To facilitate reading and understanding
  - Facilitate dissemination

Policies - Guidelines

- Security is a social problem not a computer problem
  - If the problem does not exist, it is not necessary to solve it
- Social solutions capable of being computerized should be devised
  - Separation of duties
  - Passwords selection
  - Keeping passwords secret
  - People awareness

Procedures

- Written and approved plans
- Explain how to apply/enforce the policy
- Dependent on the physical infrastructure
- Limited scope of action
- Consistency with the other procedures is facilitated by the consistency of policy
- The required update effort is smaller than that for the policies

Standards

- Establishes rules that facilitate the work
- Do not add degrees of security
- Do not confuse them with policies:
  - Example policy:
    - All users have a single unique *userid*
  - Example of standard:
    - The composition of the identifier of each user (*userid*) is formed by the initials of his name

Security: implementation of the specifications

- Generic security mechanisms
  - –Confinement
  - –Authentication
  - –Access Control
  - –Privileged execution
  - –Filtering
  - –Registration
  - –Inspection
  - –Auditing
  - –Cryptographic algorithms
  - –Cryptographic protocols

Documentation

- The documentation of the entire infrastructure is of extreme importance:
  - –Raises awareness
  - –Clarifies responsibilities of each stakeholder
  - –Reduces delays in the application of measures
  - –Broadens the base of support to the security project (administration)

Security Infrastructure

- All security components created (or existing)
- Inventory of resources
  - –Databases, networks, applications, operating systems
  - –Identification of the owner of each resource
- Definition of roles / responsibilities
  - –Policy approval authority
  - –Security Team:
    - •Group responsible for the continuous security tasks
      - –Risk analysis, people awareness, enforcement and implementation of security products
    - •Local security responsibility agents (distribution)
    - •Roles of the other users

Security Infrastructure (cont.)

- Classification of information (by the owner)
  - –Confidentiality
  - –Integrity
  - –Availability

  } Cost / Benefit

- Identification of threats to each resource, and the probability of occurrence

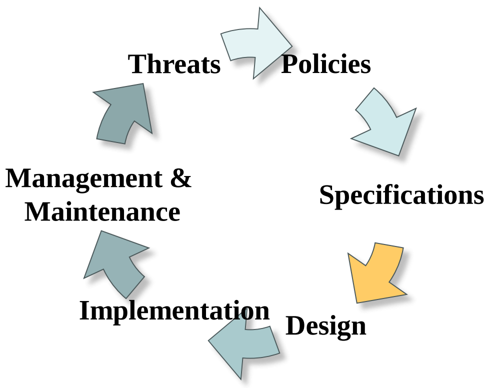
Risk Analysis

Risk Analysis

- Quantitative vs. Qualitative
- Results
  - –Value of information
  - –Costs per threat
  - –Probability of the threat
  - –Recommendations and safeguards
- Solutions
  - –Risk Reduction
  - –Risk transfer (insurance)
  - –Living with risk

Development of secure applications

Security life cycle



Development of secure applications

- Secure applications do not exist!
- Measure of Trust in the security:
  - A system is said to be trustworthy if there is sufficient **evidence** to satisfy a set of security requirements
- Trust is obtained through Assurance techniques:
  - Development methodologies
  - Formal methods
- Certification is the acceptance by assurance experts and the assignment of an assurance level

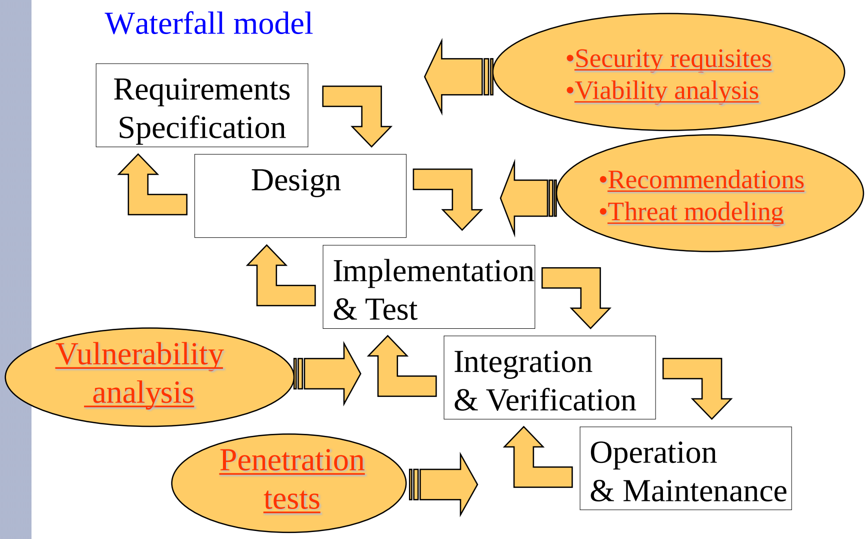
Policies ⇨ Implementation

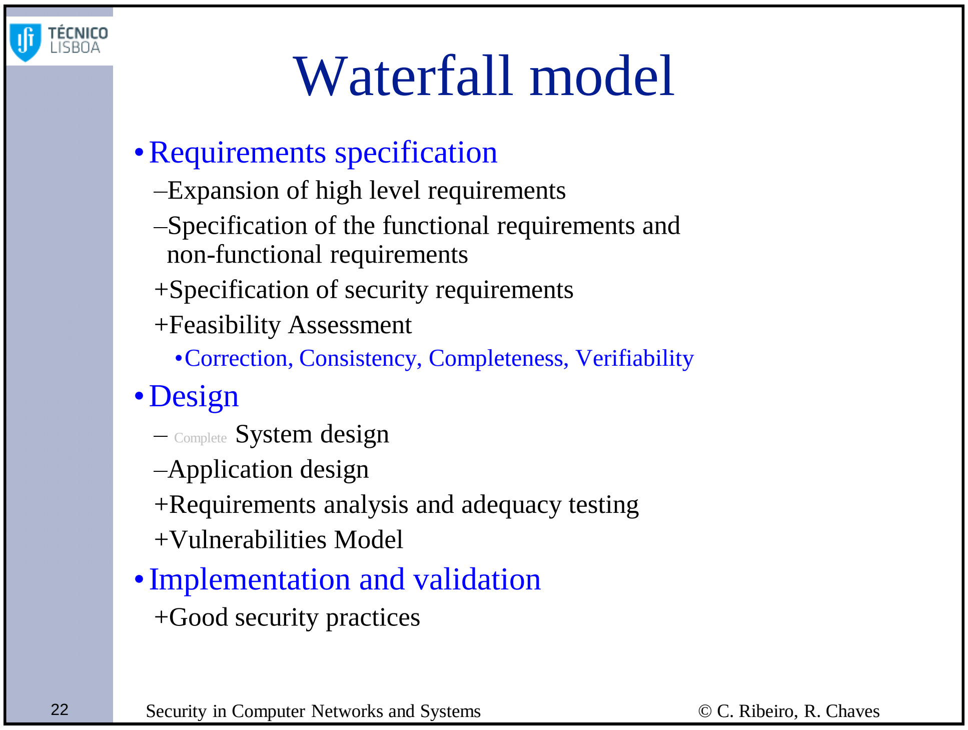
- Assumptions
  - "The lock in the door is secure"
  - "The established policy, defines, non ambiguously, a secure state and an unsecure state"
  - "The mechanisms correctly enforce the policy"
- Trust
  - "If a locksmith exists them he is trustworthy"
  - "The security mechanisms operate/work has expected"
- Assurance
  - Quantification of trust
  - Specification ⇨ Design ⇨ Implementation
  - Certification

Design and Implementation of Secure Applications

Development cycle

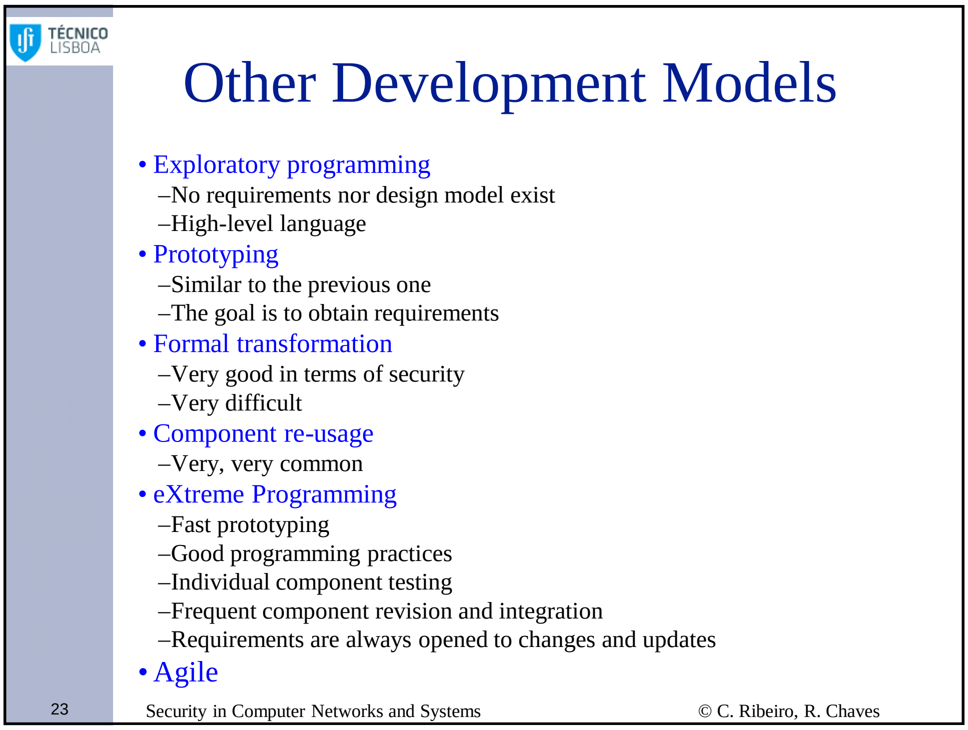
Development cycle





Waterfall model

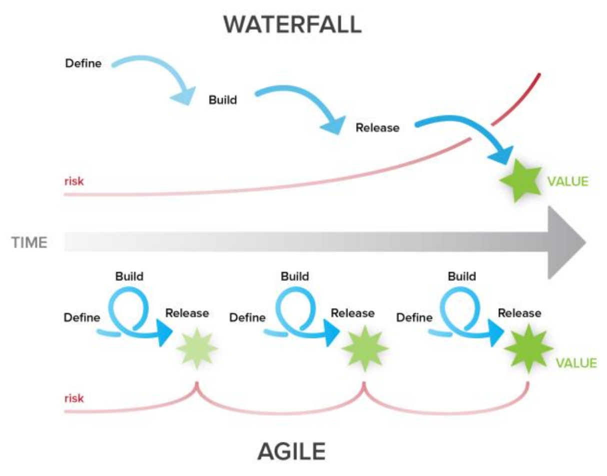
- Requirements specification
  - –Expansion of high level requirements
  - –Specification of the functional requirements and non-functional requirements
  - +Specification of security requirements
  - +Feasibility Assessment
    - •Correction, Consistency, Completeness, Verifiability
- Design
  - – Complete System design
  - –Application design
  - +Requirements analysis and adequacy testing
  - +Vulnerabilities Model
- Implementation and validation
  - +Good security practices



Other Development Models

- Exploratory programming
  - –No requirements nor design model exist
  - –High-level language
- Prototyping
  - –Similar to the previous one
  - –The goal is to obtain requirements
- Formal transformation
  - –Very good in terms of security
  - –Very difficult
- Component re-usage
  - –Very, very common
- eXtreme Programming
  - –Fast prototyping
  - –Good programming practices
  - –Individual component testing
  - –Frequent component revision and integration
  - –Requirements are always opened to changes and updates
- Agile

Agile Development



© Axian

Agile Security Practices

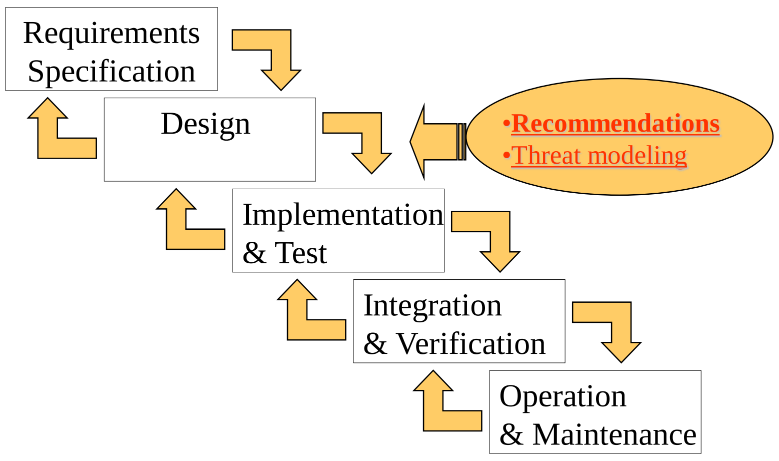
- Inception practices: at the start of the Agile project
  - –Risk Assessment
  - –Requirements Definition
  - –Incident Response
- Iteration practices: should be performed in every release
  - –Threat Assessment
  - –Code Review
  - –Design Review

- Regular practices: spread across multiple sprints during the project
  - –Dynamic Security Testing
  - –Fuzz Testing (misuse)

Adapted from M. Bartoldus (gdssecurity)

Development cycle

Waterfall model



Design of Secure Applications

Recommendations

IEEE Center for Secure Design recommendations

AVOIDING THE TOP 10 SOFTWARE SECURITY DESIGN FLAWS

Iván Arce, Kathleen Clark-Fisher, Neil Daswani, Jim DelGrosso, Danny Dhillon, Christoph Kern, Tadayoshi Kohno, Carl Landwehr, Gary McGraw, Brook Schoenfield, Margo Seltzer, Diomidis Spinellis, Izar Tarandach, and Jacob West

Interested in keeping up with Center for Secure Design activities? Follow @ieeecsd on Twitter, catch up with us via cybersecurity.ieee.org, or contact Kathy Clark-Fisher, Manager, New Initiative Development (kclark-fisher@computer.org).

http://cybersecurity.ieee.org/blog/2015/11/13/avoiding-the-top-10-security-flaws/

Goal of secure design

- Address problems in **design** stage
  - Design flaws
    - Different from implementation bugs or defects
  - Avoiding flaws can significantly reduce the number and impact of security breaches
- The goal of a secure design is to enable a system that supports and enforces the necessary authentication, authorization, confidentiality, data integrity, accountability, availability, and non-repudiation requirements, **even when the system is under attack**

Top 10 recommendations

1. Earn or give, but never assume trust
2. Use authentication that cannot be bypassed or tampered
3. Authorize after you authenticate
4. Strictly separate data and control instructions
5. All data must be explicitly validated
6. Use cryptography correctly
7. Identify sensitive data and how to handle it
8. Always consider the users
9. Understand how external components affect attack surface
10. Be flexible when considering future objects and actors

1/10 Earn or give, but never assume **trust**

- Software systems rely on composition and cooperation of two or more software tiers or components
- Offloading security functions from server to client exposes those functions to a much less trustworthy environment
- When untrusted clients send data to your system or perform a computation on its behalf, the data sent must be assumed to be compromised until proven otherwise

2/10 Use **authentication mechanism** that cannot be bypassed or tampered with

- Authentication is the act of validating an entity's identity
- A securely designed system should also prevent that user from changing identity without re-authentication
- Authentication techniques should require one or more factors for more sensitive operations
  - –Factors:
    - –something you know,
    - –something you are, or
    - –something you have

3/10 **Authorize after** you **authenticate**

- Authorization should be conducted as an explicit check
  - –Necessary even after an initial authentication has been completed
- Authorization depends not only on the **privileges** associated with an authenticated user, but also on the **context** of the request
  - –Time, location, etc.
  - –Handle revocation

4/10 Strictly **separate Data** and **Control** instructions

- Co-mingling data and control instructions in a single entity, especially a **string**, can lead to injection vulnerabilities
  - –Often leads to untrusted data controlling the execution flow of a software system
  - –Concern at all levels: machine instructions, high-level instructions, domain specific languages

5/10 All **data** must be **explicitly validated**

- It is important to explicitly ensure that assumptions on data hold
  - –Vulnerabilities frequently arise from implicit assumptions about data
- Design software systems to ensure that comprehensive data validation actually takes place and that all assumptions about data have been validated when they are used

6/10 **Use** cryptography **correctly**

- Through the proper use of cryptography, one can ensure the confidentiality of data, protect data from unauthorized modification, and authenticate the source of data
  - –and more
- Common cryptography pitfalls:
  - –Creating your own cryptographic algorithms or implementations
  - –Misuse of libraries and algorithms
  - –Poor key management
  - –Randomness that is not random
  - –Failure to allow for algorithm adaptation and evolution

7/10 Identify **sensitive data** and how it should be handled

- Data sensitivity is context-sensitive
  - –Depends on regulation, company policy, contractual obligations, user expectation, etc.
    - •Examples:
      - –User-input, data computed from scratch, data coming from external sensors, cryptographic material, and Personally Identifiable Information (PII)
- First step: create a policy that explicitly identifies different levels of classification
- Define most important property:
  - –Confidentiality
  - –Integrity
  - –Availability

8/10 Always consider the **users**

- The security of a software system is inextricably linked to what its users do with it
- Always consider the users, and any other stakeholders, in the design and evaluation of systems.
  - –Factors
  - –Trade-offs
- Make the most common usage scenario also secure
  - –“secure by default”
  - –Make relevant settings as easy to find

9/10 Understand how external components affect **attack surface**

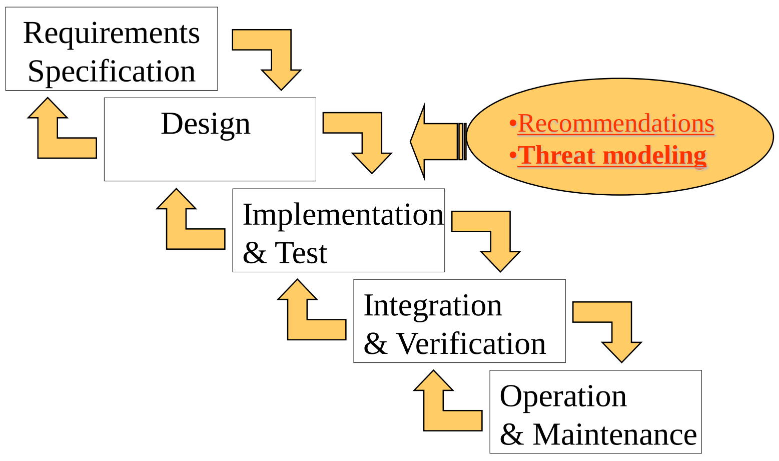
- You must assume that incoming external components are not to be trusted until appropriate security controls have been applied
- Align the component's attack surface and security policy with the overall system’s

10/10 Be flexible when considering future changes to Objects and Actors

- Software security must be designed for change
  - –Environments, threats and attacks
  - –Rather than being fragile, brittle, and static
- Consider the security implications of future changes
  - –Design for security updates
  - –Design for security properties changing over time
  - –Design for changes in components beyond your control
  - –Design with the ability to isolate or toggle functionality
  - –Design for changes to objects intended to be kept secret (keys)
  - –Design for changes in entitlements (dynamic permissions)

Development cycle

Waterfall model



TÉCNICO LISBOA

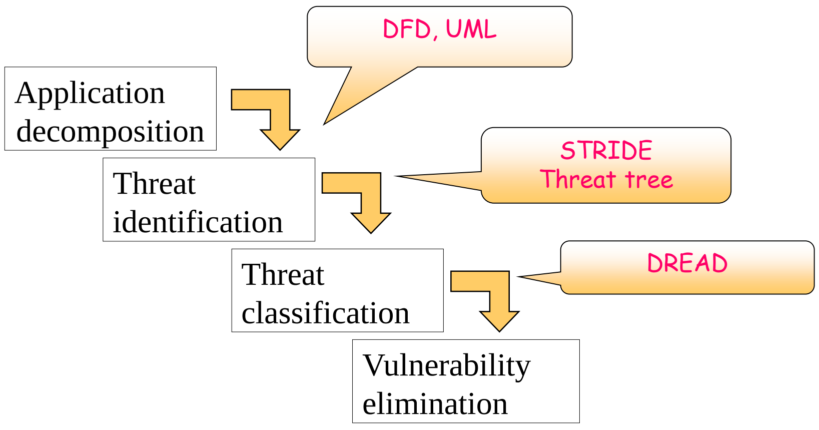
Design of Secure Applications

Threat modeling

42

TÉCNICO LISBOA

Threat modeling

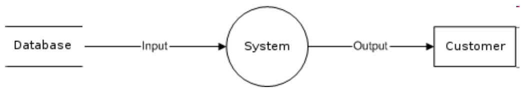


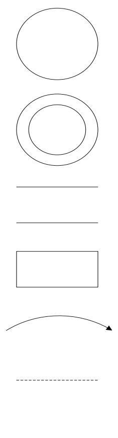
43

Threat modeling: Application decomposition

- DFD – Data Flow Diagrams
  - –Writing Secure Code, Michael Howard e David LeBlanc, Microsoft, 2nd edition.
- UML – Unified Modeling Language
  - –Structure diagrams
    - Component diagram, Class diagram, …
  - –Behavior diagrams
    - Use Case diagrams, Activity diagram, …
  - –Interaction diagrams
    - Communication diagram, Interaction diagram, Sequence diagram, …
  - –Lund, M. S., Braber, F., Stolen, K. and Vraalsen, F. (2004) *"A UML Profile for the Identification and Analysis of Security Risks During Structured Brainstorming"*, SINTEF ICT Cooperative and Trusted Systems, May.

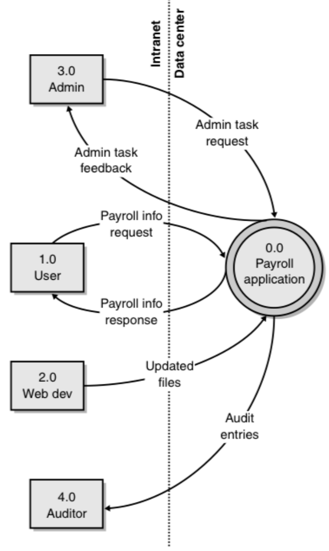
DFD entities



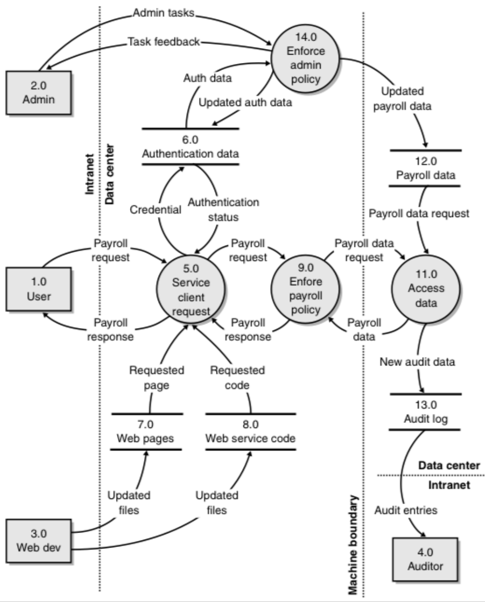

Process – transforms or manipulates data
Multiple processes
Data repository: File, data base, …
I/O
Data flow
Boundary – machines, physical, Trusted/Untrusted areas

- Start with a high level DFD diagram (Layer 1)
- Identify all components
- Properly define all borders
- For each entity perform a set of questions, according to the STRIDE model
- Add more detail with further diagrams (Layer n)

DFD level 0: payroll app



DFD level 1: payroll app



STRIDE model (Threat Classification)

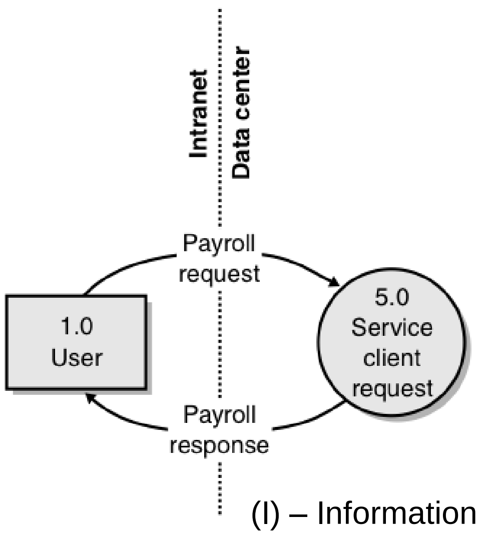
- S: Spoofing identity
  - Users, servers
  - Passwords, DNS
- T: Tampering with data
  - Information modification (file system, flowing data, database, etc.)
  - Incorrect authorization
- R: Repudiation
  - Action repudiation (sent, received, signature, etc.)
- I: Information Disclosure
  - Privacy breach
  - Data Leak
- D: Denial of Service
- E: Elevation of privilege
  - Becoming root
  - Through a Trojan

STRIDE model – Types of Threats

Types of threats	Affects processes	Affects repositories	Affects I/O	Affects the data flow
Spoofing	Yes[1]		Yes	
Tampering	Yes[2]	Yes		Yes
Repudiation		Yes	Yes	Yes
Inf. Disclosure	Yes[3]	Yes	[4]	Yes
DoS	Yes	Yes	[5]	Yes
Elevat. of Privilege	Yes[6]			

[1]Spoofing of user’s process or server.
[2]Modifing a process image in the hard drive or memory.
[3]How they operate or which secrets they contain.
[4]Only makes sense for the data it self.
[5]A DoS is not possible to the I/O directly; but it is possible to the flow of data, processes and repositories.
[6]It is only possible to elevate the privilege of a process, but other threats may exist that lead to this one, e.g.: seeing the root’s password.

DFD excerpt

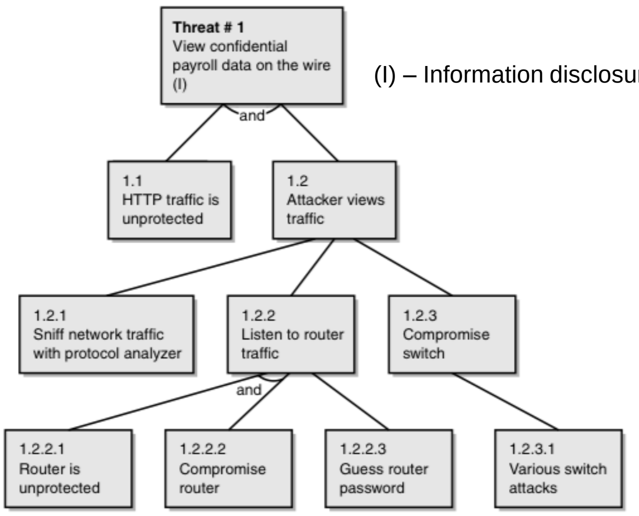


(I) – Information disclosure

Tree of Threats

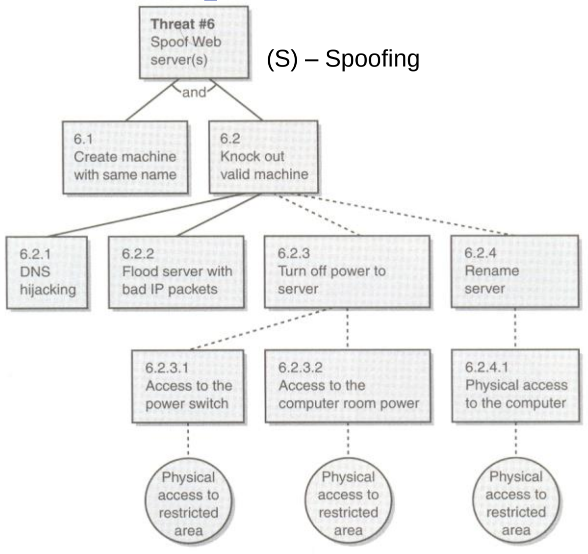
- Threats ≠ vulnerabilities
- From the identified threats, infer which are the existing vulnerabilities
- Create a tree of threats
  - –For each threat, of each entity

Example of a tree of threats


Threat # 1
View confidential payroll data on the wire (I)
and
1.1 HTTP traffic is unprotected
1.2 Attacker views traffic
1.2.1 Sniff network traffic with protocol analyzer
1.2.2 Listen to router traffic
1.2.3 Compromise switch
and
1.2.2.1 Router is unprotected
1.2.2.2 Compromise router
1.2.2.3 Guess router password
1.2.3.1 Various switch attacks

(I) – Information disclosure

Another example tree of threats


Threat #6
Spoof Web server(s)
and
6.1 Create machine with same name
6.2 Knock out valid machine
6.2.1 DNS hijacking
6.2.2 Flood server with bad IP packets
6.2.3 Turn off power to server
6.2.4 Rename server
6.2.3.1 Access to the power switch
6.2.3.2 Access to the computer room power
6.2.4.1 Physical access to the computer
Physical access to restricted area
Physical access to restricted area
Physical access to restricted area

(S) – Spoofing

TÉCNICO LISBOA

Vulnerabilities classification: DREAD (Risk Assessment Model)

- Vulnerabilities are rated in a 1 to 10 scale for each category:
  - –D – Damage potential
    - How serious is the damage, if it exists?
    - Priority elevation = 10.
  - –R – Reproducibility
    - How hard it is to reproduce the attack?
    - Random occurring vulnerabilities are hard to reproduce.
  - –E – Exploitability
    - How much work is it to launch the attack?
    - *Script kiddies* = 10; Experts = 5; Unlimited resources = 1.
  - –A – Affected users
    - How many people will be affected by the attack?
    - 91-100% = 10; 0-10% = 1
  - –D – Discoverability
    - How probable is it of being discovered?
    - Internet = 10; Locked in a vault = 1 !!!
- The relevance of the vulnerability is the average value of all the categories.

TÉCNICO LISBOA

Assessing the risk of identified threat

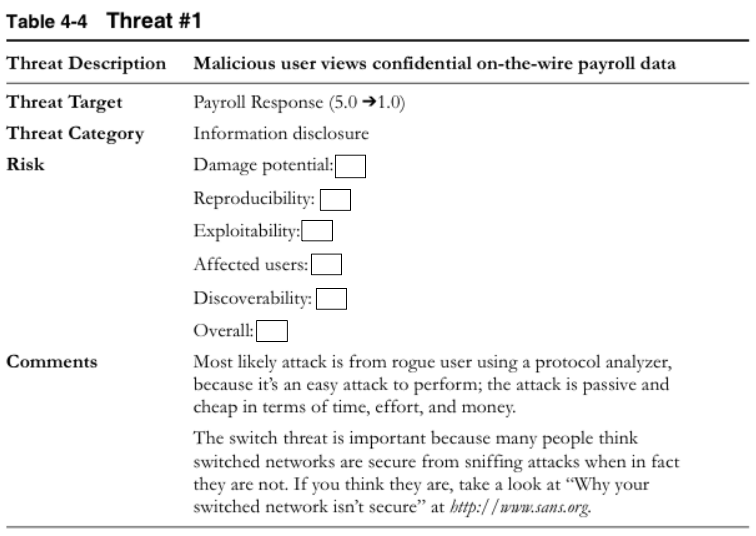


Table 4-4 Threat #1

Threat Description	Malicious user views confidential on-the-wire payroll data
Threat Target	Payroll Response (5.0 →1.0)
Threat Category	Information disclosure
Risk	Damage potential: ☐
	Reproducibility: ☐
	Exploitability: ☐
	Affected users: ☐
	Discoverability: ☐
	Overall: ☐
Comments	Most likely attack is from rogue user using a protocol analyzer, because it's an easy attack to perform; the attack is passive and cheap in terms of time, effort, and money. The switch threat is important because many people think switched networks are secure from sniffing attacks when in fact they are not. If you think they are, take a look at "Why your switched network isn't secure" at *http://www.sans.org*.

Assessing the risk of identified threat

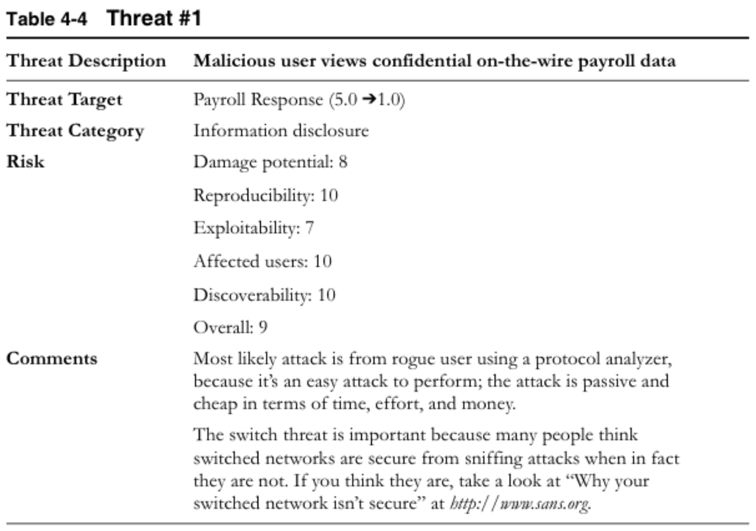


Table 4-4 Threat #1

Threat Description	**Malicious user views confidential on-the-wire payroll data**
Threat Target	Payroll Response (5.0 →1.0)
Threat Category	Information disclosure
Risk	Damage potential: 8
	Reproducibility: 10
	Exploitability: 7
	Affected users: 10
	Discoverability: 10
	Overall: 9
Comments	Most likely attack is from rogue user using a protocol analyzer, because it's an easy attack to perform; the attack is passive and cheap in terms of time, effort, and money.
	The switch threat is important because many people think switched networks are secure from sniffing attacks when in fact they are not. If you think they are, take a look at "Why your switched network isn't secure" at *http://www.sans.org*.

56

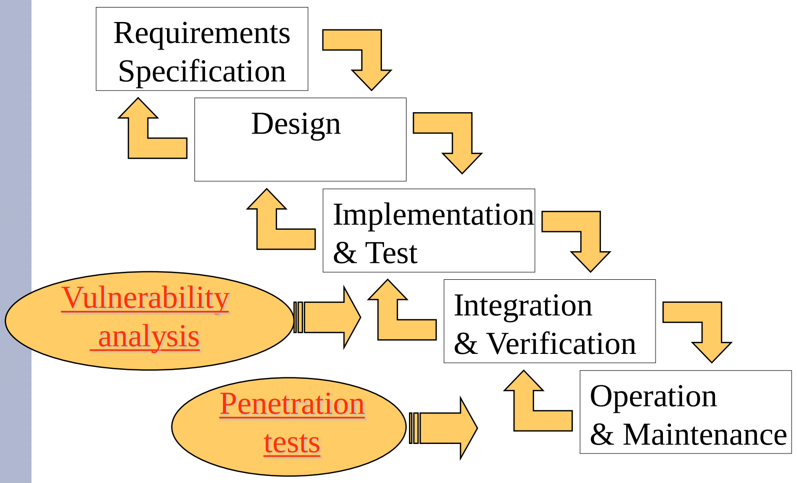
Threat Modeling Procedure

- Decompose application
  - DFD
- For each element of the DFD -> STRIDE
- For each threat -> Tree of threats
- For each vulnerability -> DREAD
- Elimination or mitigation of the discovered vulnerabilities/threats

57

Development cycle

Waterfall model



Implementation of Secure Applications

Penetration testing

- Flaw Hypothesis methodology
  - Information gathering
  - Establish hypothesis
  - Test the hypothesis
  - Flaw generalization
  - Flaw elimination

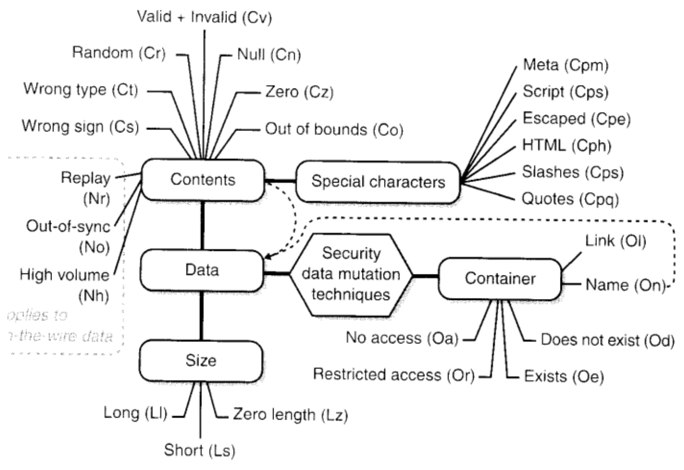
Input mutation

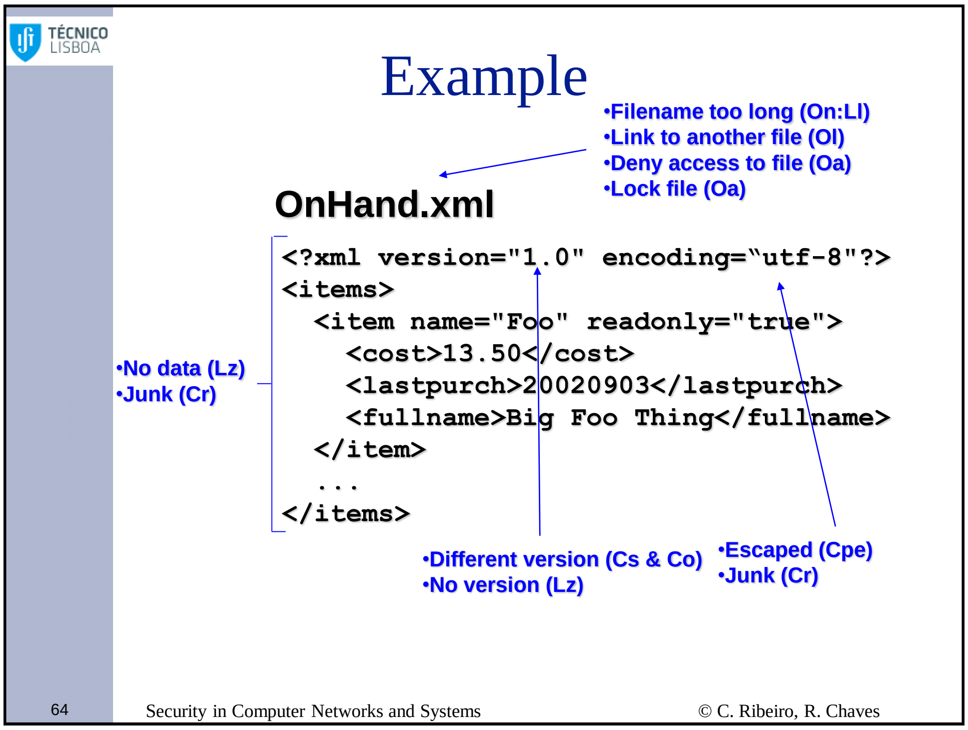
- Identification of the input data
- Controlled data mutation
- Result analysis

Input identification

- Application decomposition
- Identification of the interfaces
- Enumeration of data inputs:
  - Sockets
  - Pipes
  - Registry
  - Files
  - RPC (etc.)
  - Input parameters
  - Etc.
- Enumeration of the data structures
  - C/C++ Data structures
  - HTTP headers
  - HTTP body
  - Search strings
  - Flags
  - Etc.
- Establish the valid constructs

Controlled input mutation





Example

•Filename too long (On:Ll)
•Link to another file (Ol)
•Deny access to file (Oa)
•Lock file (Oa)

OnHand.xml

```
<?xml version="1.0" encoding="utf-8"?>
<items>
  <item name="Foo" readonly="true">
    <cost>13.50</cost>
    <lastpurch>20020903</lastpurch>
    <fullname>Big Foo Thing</fullname>
  </item>
  ...
</items>
```

•No data (Lz)
•Junk (Cr)

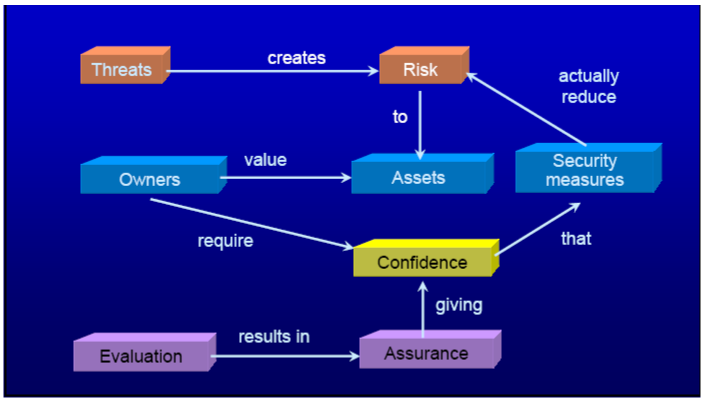
•Different version (Cs & Co)
•No version (Lz)

•Escaped (Cpe)
•Junk (Cr)

Secure Applications

Certification of applications and systems

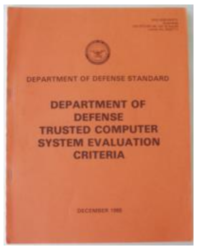
Trust and Certification



Certification

- For operating systems
  - TCSEC (Orange Book) (US)
    - Functional requisites
    - Assurance requisites
    - Class assurance (C1,C2,B1,B2,B3,A1)
  - ITSEC (UK, France, Germany, Nederland's)
- Single domain networks
  - TNI - Trusted Network interpretation
- For cryptography
  - FIPS 140
- Application evaluation
  - ITSEC
  - Common Criteria

TCSEC


DEPARTMENT OF DEFENSE STANDARD
DEPARTMENT OF DEFENSE TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA
DECEMBER 1985

- American standard
- Evaluation Phase:
  - –Design analysis
    - Based on the documentation, no access to the source code
  - –Test analysis
  - –Final review
- Evaluation is performed by trained, government-sponsored independent evaluators.
- Problems:
  - –Based heavily on confidentiality, disregarding other important security services (e.g. integrity)
  - –Narrow scope
    - Oriented for Military operating systems
  - –Long time for evaluation
  - –Ties assurance with functionality in the evaluation:
    - E.g.: B1
      - Mandatory Access Control (MAC)
      - All objects and users have labels
      - Informal model of the security policy
      - More detailed testing and documentation

ITSEC

- European standard
- Functionality and assurance are separate
- Applicable to systems and applications
  - –TOE – *Target of Evaluation*

TCSEC	ITSEC
C1	F1+E2
C2	F2+E2
B1	F3+E3
B2	F4+E4
B3	F5+E5
A1	F5+E6

- Problems:
  - –Assurance and Functionality are evaluated separately
  - –No validation that security requirements make sense
    - May meet the intended evaluation goal, but is that sufficient to the desired assurance level!
  - –Inconsistency in evaluations
    - Not as formally defined as in TCSEC

Common Criteria

- ***Common Criteria Recognition Arrangement***
  - Signed by 19 countries
  - ISO/IEC 15408
- CC
  - CC documentation
  - Evaluation methodology of CC (CEM)
  - National schemas (Country specific)
    - Evaluators selection; certification attributions; interaction between evaluators and vendors, etc.
  - e.g. in the USA NIST accredits commercial organizations
- CC Methodology (CEM)
  - Functional requirements
  - Assurance requirements
  - Evaluation Assurance Levels (EALs)
    - Of the existing 7 levels only the first 5 have been achieved for full systems.
  - e.g. Java Smart Card has an EAL=5+
- Types of evaluation
  - Protection Profile (PP)
  - Security Target (ST)

CC – PP and ST (Examples)

- Protection Profiles (Product independent)
  - Operating systems (C2, CS2, RBAC)
  - Firewalls (Packet Filter, Application Level gateway)
  - SmartCards
- Security Targets (Specific for each product)

–Hitachi Universal Storage Platform V	: EAL2
–Cisco PIX Firewall	: EAL4+
–GemXplore Xpresso V3 Java Card Platform	: EAL5+
–Tenix Interactive Link Data Diode Device	: EAL7+

- List of PPs
  - http://www.commoncriteriaportal.org/pps/
- List of STs (certified products)
  - http://www.commoncriteriaportal.org/products/

CC – PP and ST

Protection Profile (generic)

- Introduction
- Description of the class/family of target products, a.k.a. Target of Evaluation (TOE)
- Description of the execution environment
  - –Assumptions regarding the system operation
  - –Threatened resources
  - –Security policy of the target organization
- Security objectives
  - –Product/system objectives
  - –Environment objectives
- Security requirements
  - –Functional
  - –Assurance
- Rational
  - –Interconnects the previous points

Security Target (specific)

- Introduction
- TOE description
- Description of the execution environment
  - –Assumptions regarding the system operation
  - –Threatened resources
  - –Security policy of the target organization
- Security objectives
  - –Product/system objectives
  - –Environment objectives
- Security requirements
  - –Functional
  - –Assurance
- TOE specification
  - –Security mechanisms
  - –Description on how to assure security
- PP claims
  - –How the PP objectives /requirements are fulfilled
- Rational

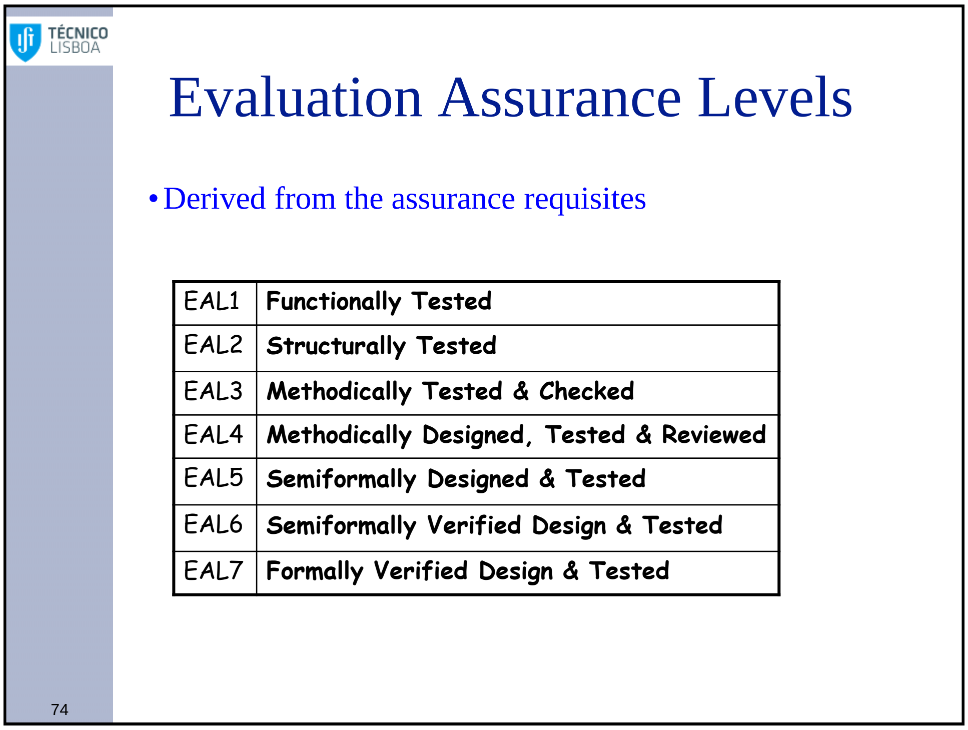
CC security requirements

Functional Requirements

- –Product/system behavior definition regarding security
- –11 classes divided in families that contain components
- –Components have:
  - Requirements definition
  - Dependencies from other requirements
  - Requirements hierarchy
- –Predefined classes:
  - Audit (FAU)
  - Cryptography Support (FCS)
  - Communications (FCO)
  - User Data Protection (FDP)
  - Identification and Authentication (FIA)
  - Security Management (FMT)
  - Privacy (FPR)
  - Protection of the TOE Security Functions (FPT)
  - Resource Utilization (FRU)
  - TOE Access (FTA)
  - Trusted Path/Channels (FTP)

Assurance Requirements

- –Establish confidence in the security features
- –Correction of the implementation
- –Fulfillment of the security objectives
- –10 classes
  - 1 – Evaluation of PPs
  - 1 – Evaluation of STs
  - 1 – Maintenance of Assurance
  - 7 – Product assurances
- –Assurance classes:
  - Development
    - –TOE design, Functional specifications, …
  - Delivery and Operation
  - Configuration
  - Product Documentation
  - Life cycle
    - –Delivery, Flaw remediation, …
  - Testing
    - –Depth, coverage, …
  - Vulnerability analysis



TÉCNICO LISBOA

Evaluation Assurance Levels

- Derived from the assurance requisites

EAL1	**Functionally Tested**
EAL2	**Structurally Tested**
EAL3	**Methodically Tested & Checked**
EAL4	**Methodically Designed, Tested & Reviewed**
EAL5	**Semiformally Designed & Tested**
EAL6	**Semiformally Verified Design & Tested**
EAL7	**Formally Verified Design & Tested**

74