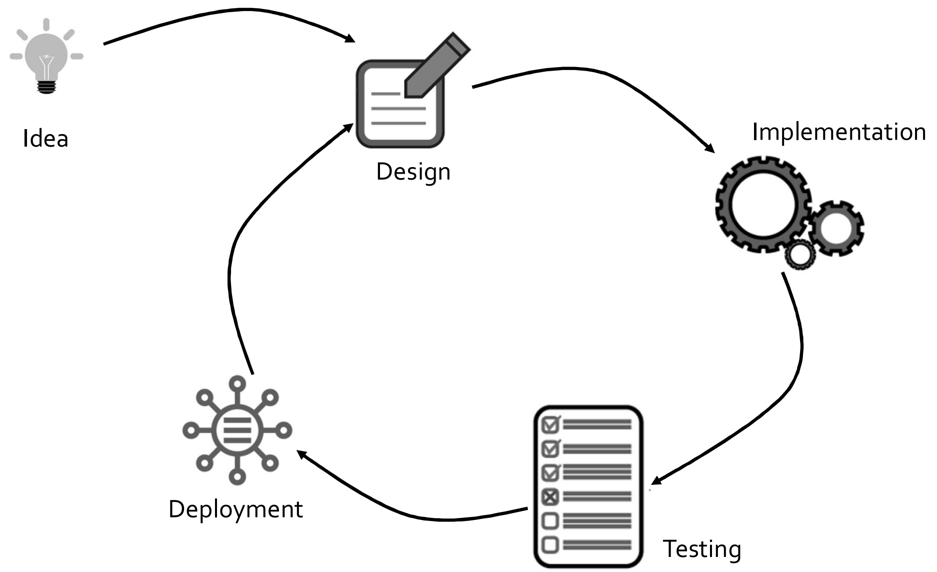
Software Security Engineering Components

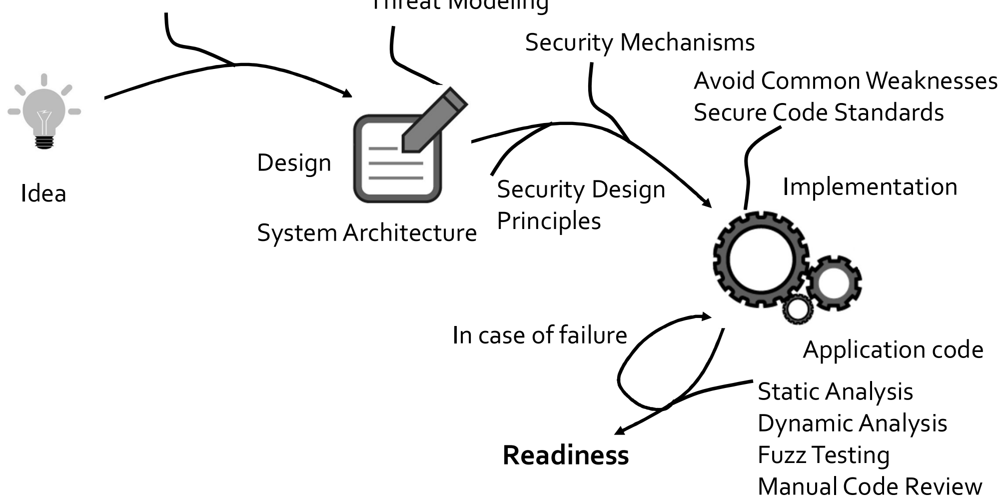
SECURE SOFTWARE ENGINEERING

APM@FEUP

A Simple Software Development Process



Security Tasks Incorporation



Security Policy and Mechanism

- **Policy**
  - **A statement of what is, and is not, allowed**
- **Mechanism**
  - **a procedure, tool, or method of enforcing a policy**
- **Security mechanisms also can implement functions that help prevent, detect, respond, and recover from security attacks**
- **Many security functions on systems are typically made available to users as a set of security services through APIs or integrated interfaces, used by several applications**
- **Cryptography underlies many of the security mechanisms**

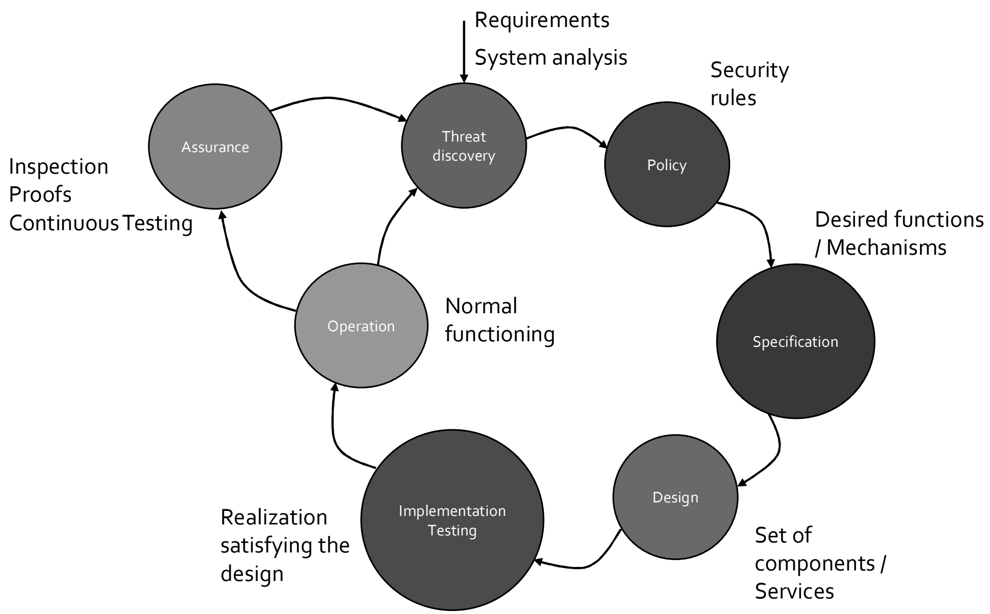
Some Security Mechanisms

- **Authentication**
  - assurance that the communicating entity is the one that it claims to be
- **Access Control**
  - prevention of the unauthorized use of a resource
- **Data Confidentiality**
  - protection of data from unauthorized disclosure
- **Data Integrity**
  - assurance that data received is not modified and is as sent by an authorized entity
- **Non-Repudiation**
  - protection against denial by one of the parties in a communication (origin or destination)

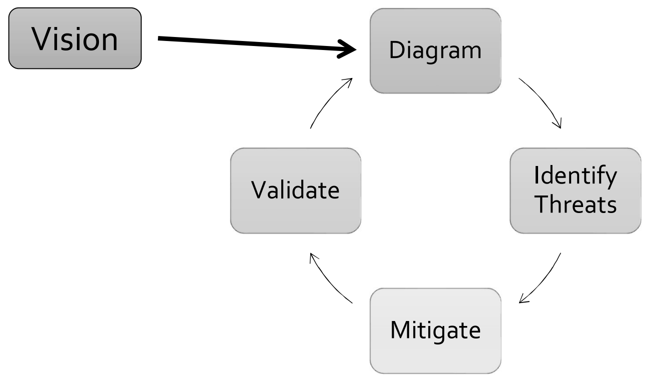
Some Security Services

- **Specific**
  - Encipherment
  - Digital signature
  - Access control
  - Data integrity
  - Authentication exchange
    - Ensure the identity of an entity using message exchange
  - Traffic padding
    - Insertion of arbitrary bits in messages to frustrate traffic analysis
  - Routing control
    - Select physically secure routers and allow route changes when a breach is suspected
  - Notarization
    - Use of a trusted third party
- **Pervasive**
  - Trusted functionality
    - Established by a security policy
  - Security label
    - Marking resources with security attributes
  - Event detection
    - Detection of security related events
  - Security audit trail
    - Collection of data to facilitate auditing
  - Security recovery
    - Takes recovery actions when solicited by other security mechanisms

The Security Life Cycle



Threat Modeling in a Nutshell



- **Threat Modeling is a fundamental component of**
  - Building security in
  - Security by design
  - Shifting security left (in the development process)
    - Should be included as early as possible

Identify Threats

- Several methodologies have been proposed
  - Most common one is STRIDE (focused on CIA)
  - Other comprise
    - PASTA – Process for Attack Simulation and Threat Analysis (risk-centric methodology)
    - LINDDUN – An approach including systematic privacy threats, not so focused on the CIA (linkability, identifiability, non-repudiation, detectability, disclosure, unawareness, non-compliance)
    - INCLUDES NO DIRT (STRIDE + LINDDUN + CO) (Clinical error, Overuse)
- Identification
  - For each entry determine how adversaries can attempt to affect assets
  - For every asset, predict what adversaries can try and their goals
- Analysis
  - Decompose threats into individual actions building an attack tree
  - Evaluate the risk of the threat

Identify threats

- Answers to questions like
  - Can an unauthorized network user view confidential data like addresses or passwords ? How ?
  - Can an unauthorized user modify data like payments, purchases in a database, or create them ? How ?
  - Could someone deny legitimate users, access to the application ? How ?
  - Could an authorized user exploit an application feature to raise their privilege to a higher role (e.g., an administrator) ? How ?
- Use databases of known attacks and categories to identify your threats
  - CAPEC – Common Attack Pattern Enumeration and Classification
    - https://capec.mitre.org
  - ATT&CK – adversary tactics and techniques based on real world observation
    - https://attack.mitre.org

STRIDE

- *The STRIDE security threat model should be used by all products to identify various types of threats the product is susceptible to during the design phase. Threats are identified based on the design of the product*

Threat	Property	Definition	Example
Spoofing	Authentication	Impersonating something or someone else.	Pretending to be a legitimate user, or server on the system, or a system update file
Tampering	Integrity	Modifying data or code	Modifying a configuration file on disk, or a packet as it traverses the network
Repudiation	Non-repudiation	Claiming to have not performed an action	"I didn't send it!"
Information Disclosure	Confidentiality	Exposing information to someone not authorized to see it	Reading key material (cryptographic) from an app
Denial of Service	Availability	Deny or degrade service to users	Crashing the web site, sending a packet and absorbing seconds of CPU time, or routing packets into a black hole
Elevation of Privilege	Authorization	Gain capabilities without proper authorization	Allowing a remote internet user to run commands is the classic example, but running kernel code from lower trust levels is also EoP

Mitigations

- Mitigation is the point of threat modeling
  - Designed and performed according to priorities and impact
  - Application of standard solutions to known threats
- Goals of mitigation
  - Address or alleviate a threat
  - Protect customers and assets
  - Design secure software
  - Pass the goals to requirements list and track their fulfilment
- Ways to address threats
  - Redesign to eliminate
    - Remove a functionality to avoid the threat and risk
  - Apply standard mitigations
  - Invent new mitigations (custom)
    - Hard and risky
  - Accept vulnerability in design (for low-risk situations)

Mitigation Common Technologies

Threat	Mitigation Technology	Developer Example	SysAdmin Example
Spoofing	Authentication	Digital signatures, Active directory, LDAP	Passwords, crypto tunnels
Tampering	Integrity, permissions	Digital signatures	ACLs/permissions, crypto tunnels
Repudiation	Fraud prevention, logging, signatures	Customer history risk management	Logging
Information disclosure	Permissions, encryption	Permissions (local), PGP, SSL	Crypto tunnels
Denial of service	Availability	Elastic cloud design	Load balancers, more capacity
Elevation of privilege	Authorization, isolation	Roles, privileges, input validation for purpose, (fuzzing*)	Sandboxes, firewalls

* Fuzzing/fault injection is not a mitigation, but a testing technique

Some Standard Mitigations

Category	Technology	Example Mitigations	Some Concrete implementations
Spoofing	Authentication	• Basic & Digest authentication (principals) • Live authentication (principals) • Cookie authentication (principals) • Kerberos authentication (principals) • PKI systems such as SSL/TLS and certificates • IPSec • Digitally signed packets • Digital signatures (code/data) • Message authentication codes (code/data) • Hashes (code/data)	• Authentication based on key exchange • Decide on single-factor, two-factor, or multi-factor authentication • Offload authentication to another provider • Restrict authentication to certain IP ranges or locations
Tampering	Integrity	• Integrity Controls • ACLs • Digital signatures • Message Authentication Codes	• Data protected from tampering with cryptographic integrity mechanisms • Only enumerated authorized users may modify data
Repudiation	Non-repudiation	• Strong Authentication • Secure logging and auditing • Digital Signatures • Secure time stamps • Trusted third parties	• Maintain logs • Digital signature
Information Disclosure	Confidentiality	• Encryption • ACLs	• Stored data will only be available to authorized users • Existence of data is exposed only to authorized users • Content and existence of communication between two users will only be exposed to these authorized users
Denial of Service	Availability	• ACLs • Filtering • Quotas • Authorization • High availability designs	• Rate limiting or throttling access to a service • Real-time monitoring of log files and other resources to note sudden changes
Elevation of Privilege	Authorization	• ACLs • Group or role membership • Privilege ownership • Permissions • Input validation	• System has a central authorization engine • Authorization controls stored and controlled using ACLs • System limits who can write data to higher integrity level • System uses roles/accounts or permissions to manage access

Threat Modeling Tools

- **Help and automate some of threat modeling process**
  - **IriusRisk – commercial tool with a threat library built from several databases (CAPEC, CWE, OWASP, WASC Threat Classification). It has a free community edition**
  - **SD Elements – commercial. Full cycle security management solution including Threat Modeling (from Security Compass company)**
  - **ThreatModeler – another commercial offer**
  - **Microsoft Threat Modeling Tool – free, from Adam Shostack and Ms SDL team. Uses STRIDE, DREAD and a library of templates (with some Windows specifities)**
  - **OWASP Threat Dragon – free, web and desktop tool, suggesting threats and mitigations**
  - **CAIRIS – open source, is a platform for specifying and modelling secure and usable systems (https://cairis.org)**
  - **Pytm – code-based Threat Modeler - based on a python definition, pytm can generate, a Data Flow Diagram (DFD), a Sequence Diagram and threats to your system. Is an incubating project in OWASP (https://owasp.org/www-project-pytm). It's free.**

Cryptography

- **Derives from Greek words *kryptos* and *graphein***
  - **Kryptos – hidden secret**
  - **Graphein – description**
- **A definition**
  - **Is the practice and study of techniques for secure access, communications, and storage, in the presence of adversaries**
    - **Protocols and processes to prevent reading private messages**
    - **Assurance of data confidentiality**
    - **Also, intervention in data integrity (not modified), data authentication (not forged), and non-repudiation (known author or recipient)**
- **Modern cryptography**
  - **Based on mathematical theory**
  - **Assumes computational hardness, using algorithms hard to break by adversaries**
  - **Adversaries can be an eavesdropper, a man-in-the-middle, or someone accessing data/functionalities without authorization**

Cryptography libraries

- **Many cryptography algorithms are already implemented**
  - Available in many general development frameworks (Java, .NET, …)
  - Many others have wrappers to other open-source libraries
    - PHP, Ruby, Python, …, have a wrapper on openSSL
- **OpenSSL**
  - Has an implementation of the SSL/TLS protocol
  - A very thorough cryptography functions' set (in a library) written in C
  - A command line interface to most of those functions
- **Other**
  - Bouncy Castle – very comprehensive, for Java and C#
  - Crypto++ – written in C++
  - Libgcrypt – in C, by the GnuPG community
  - CryptoComply – commercial (Java, C), very complete

Cryptographic Recommendations

- **Several official organizations produce documents recommending the cryptography algorithms to use and their parameterization**
  - NIST is one of them, for the USA federal information systems
  - They use the notion of cryptographic security strength
    - Estimate of the number of operations needed by the best-known algorithm to break the cryptography in the considered concrete process
      - It is measured in bits; s bits represent a number of operations of 2^s
    - NIST considers currently only five levels: 80, 112, 128, 192, 256 bits
      - 80 bits are currently disallowed, and legacy systems must be immediately replaced
      - 112, 128, and 192 are the current levels for low, medium, and high-security systems
        - Until the limit of 2030
        - After 2030 all those systems should have transitioned to 128, 192, and 256 bits, respectively
        - The transitioning should be considered during the previous 10 years time-frame, so the current one started in 2020
        - All these dates can change, if advances in prime factoring, general discrete-logarithm, elliptic curve discrete-logarithm and other algorithms used in cryptographic implementations and attacks are observed
        - Advances in quantum-computing can also anticipate disallowances for DSA, DH, MQV, and RSA

Cryptographic operations and strength

- **Fundamental cryptographic operations**
  - Encryption / Decryption (confidentiality)
  - Message Authentication Code (integrity)
  - Cryptographic Hash Functions
    - Preimage resistant
    - Weak collision resistant (brute-force: 2^{length})
    - Collision resistant (brute-force: $2^{length/2}$)
  - Secret agreement (DH)
  - Digital signatures (integrity / non-repudiation)
  - Cryptographic random generation

Brute-force security (symmetric encryption)

Key size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 decryptions/μs	Time Required at 10^{13} decryptions/μs
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	$2^{55}\mu s = 1.125$ years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	$2^{127}\mu s = 5.3 \times 10^{21}$ years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	$2^{167}\mu s = 5.8 \times 10^{33}$ years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	$2^{191}\mu s = 9.8 \times 10^{40}$ years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	$2^{255}\mu s = 1.8 \times 10^{60}$ years	1.8×10^{56} years

Strength time-frame and hash strength

General security strength current time-frame

Security Strength		Through 2030	2031 and Beyond
< 112	Applying protection	Disallowed	
	Processing	Legacy use	
112	Applying protection	Acceptable	Disallowed
	Processing		Legacy use
128	Applying protection and processing information that is already protected	Acceptable	Acceptable
192		Acceptable	Acceptable
256		Acceptable	Acceptable

Hash functions are used as components of many other cryptographic algorithms. Recent attacks on SHA-1 raised the assumption that its strength is far less than the one stated, so it should not be used anymore.

Security strengths for hash functions

Security Strength	Digital Signatures and Other Applications Requiring Collision Resistance	HMAC,[70] KMAC,[71] Key Derivation Functions,[72] Random Bit Generation[73]
≤ 80	SHA-1[74]	
112	SHA-224, SHA-512/224, SHA3-224	
128	SHA-256, SHA-512/256, SHA3-256	SHA-1, KMAC128
192	SHA-384, SHA3-384	SHA-224, SHA-512/224, SHA3-224
≥ 256	SHA-512, SHA3-512	SHA-256, SHA-512/256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512, KMAC256

Symmetric and asymmetric strengths

Recommended lengths for keys in symmetric and asymmetric cryptography, corresponding to a given strength

Security Strength	Symmetric Key Algorithms	FFC (DSA, DH, MQV)	IFC* (RSA)	ECC* (ECDSA, EdDSA, DH, MQV)
≤ 80	2TDEA	$L = 1024$ $N = 160$	$k = 1024$	$f = 160\text{-}223$
112	3TDEA[68]	$L = 2048$ $N = 224$	$k = 2048$	$f = 224\text{-}255$
128	AES-128	$L = 3072$ $N = 256$	$k = 3072$	$f = 256\text{-}383$
192	AES-192	$L = 7680$ $N = 384$	$k = 7680$	$f = 384\text{-}511$
256	AES-256	$L = 15360$ $N = 512$	$k = 15360$	$f = 512+$

TDEA - Triple Data Encryption Algorithm (tripleDES) 3TDEA will be disallowed after 2023
FFC – finite field cryptography
IFC – Integer factorization cryptography
ECC – Elliptic curve cryptography
MQV – Menezes-Qu-Vanstone (key establishment)
L, N – sizes of public and private keys for the algorithms that use FFC
k – size of the keys' modulus for RFC
f – range of key size for ECC (the size of the order of the basepoint G)

*IFC and ECC strengths are expected to be severely affected with the generalization of quantum-computing cryptography.
Post-quantum algorithms are already being evaluated for adoption soon. NIST launched a request for standardization in 2017, and the round 3 is now completed with several promising candidates for public key encryption, key establishment, and digital signatures, that should be quantum resistant.