Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative

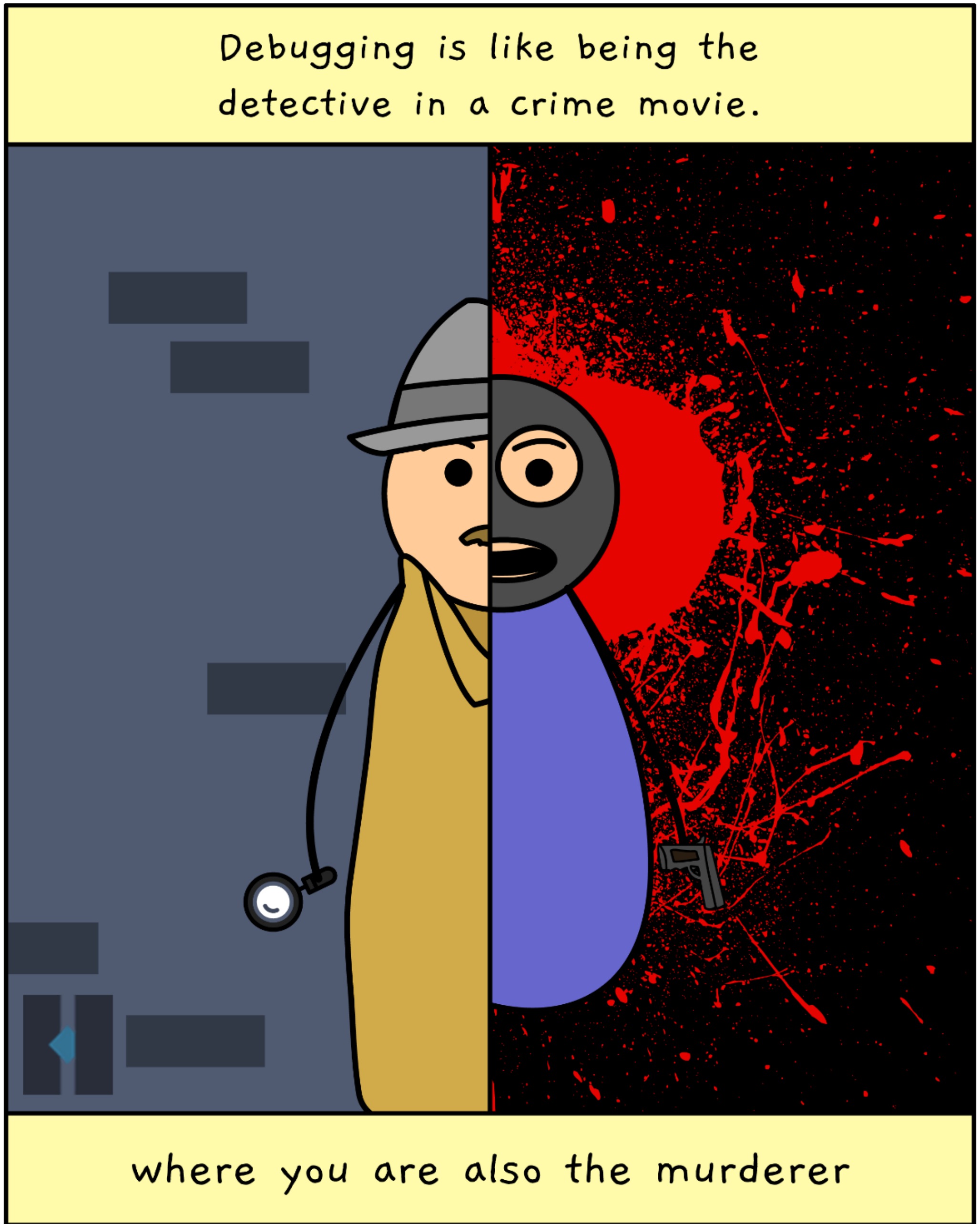
SUPSI

Security by Design

Standards & Principles

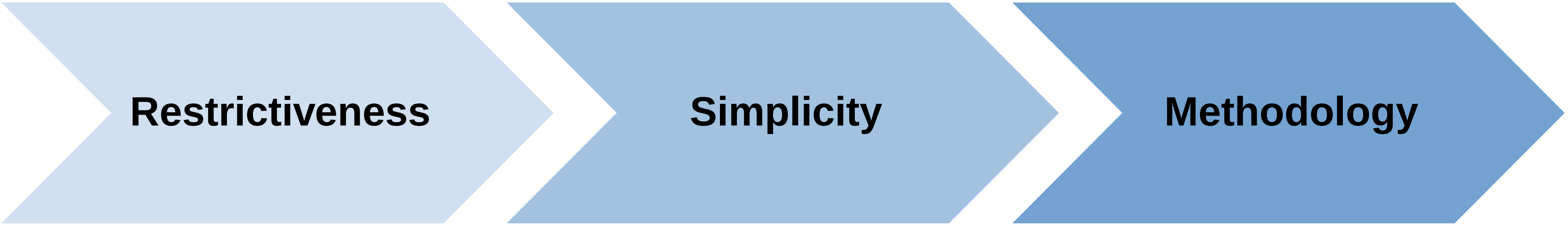
SUPSI DTI
Anno scolastico 2025 - 2026

Responsabile: Angelo Consoli


Debugging is like being the detective in a crime movie.
where you are also the murderer

Source: *https://i.pinimg.com/originals/37/4a/fc/374afc55bb55bddba2806763747aefa8.png*

Design Principle Categories



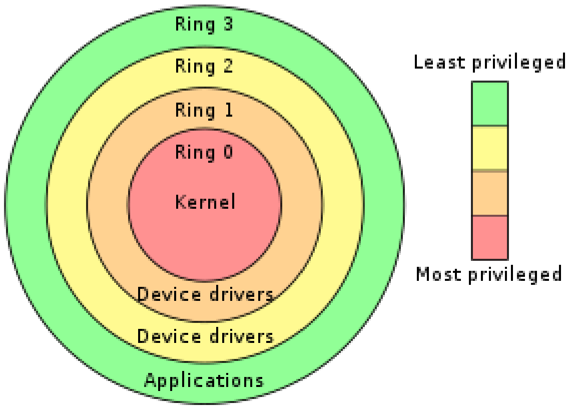
Restrictiveness

Simplicity

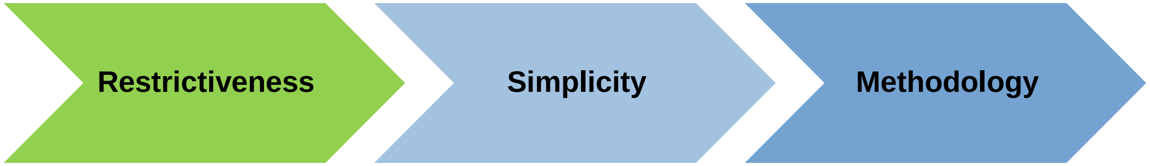
Methodology

1 - Least Privilege

Software (users or systems) should be given only those privileges that it needs to complete its task and only for the minimum time necessary.



- Reduce risk of breach.
- The security impact of a security incident or corruption of the component will be minimized.
- The security analysis of the component will be simplified (e.g. forensic analysis on a breach will be easier).


Restrictiveness

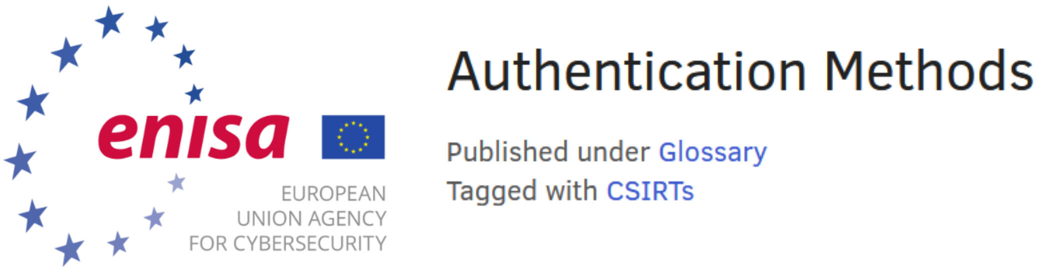
Simplicity

Methodology

2 - Authenticate

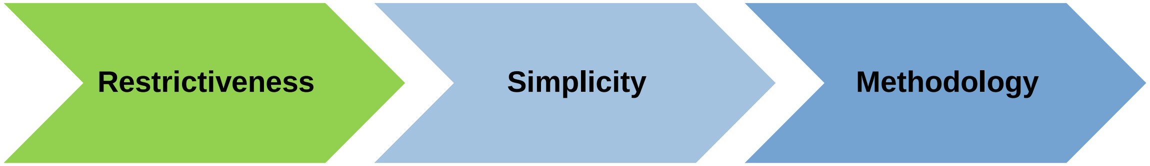
Use an authentication mechanism that cannot be bypassed or tampered.

Brief good read: https://www.enisa.europa.eu/news/enisa-news/tips-for-secure-user-authentication



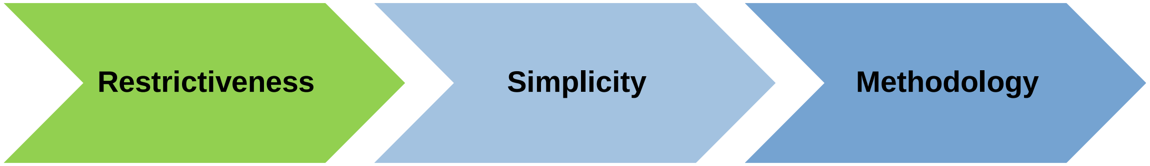
Authentication Methods

Published under Glossary
Tagged with CSIRTs



3 - Authorization

- Authorization is the act of checking that an entity (user or other system) is allowed or disallowed to perform certain actions based upon their authentication.
- Authorization of a sensitive operation (e.g. bank environment) may require a re-authentication or a higher level of authentication.
- The user interface should allow the user to easily review any active authority relationships that would affect security-relevant decisions.
- The interface should allow accounts/privileges to be easily disabled/revoked.



Restrictiveness

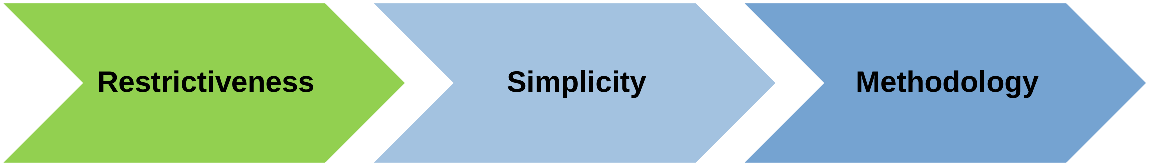
Simplicity

Methodology

4 - Fail-safe defaults

Unless a subject is given explicit access to an object, it should be denied access to that object.

- Some examples:
  - When credit card authorization system is down, the clerk may approve charges (potentially with an old manual system).
  - Database or other stack trace information given when an uncaught exception is thrown. (information disclosure).
- The attacker then “just” has to figure out how to cause a failure.
- If the subject is unable to complete its action or task, all interim changes should be undone and the security state restored prior to termination.



Restrictiveness

Simplicity

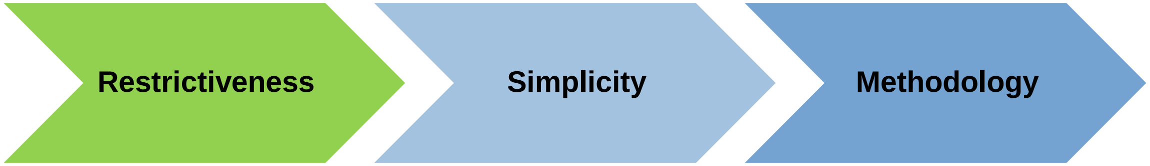
Methodology

5 - Complete Mediation

Software should validate every access to every object to ensure that the access is allowed.

Checks should ensure attempted access do not violate security properties and security policies.

- Caching permissions can increase the performance of a system, but at the cost of allowing secured objects to be accessed.
- This principle is easier to implement and evaluate if principle of least common mechanism has been followed.



Restrictiveness **Simplicity** **Methodology**

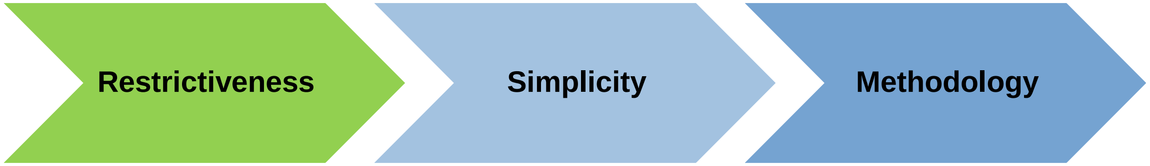
6 - Separation

Software should not grant access to a resource, or take a security-relevant action, based upon a single condition.

Sensitive operations should require the cooperation of more than one check.

For example: UNIX users cannot change from their account to the root account unless:

- User knows the root password.
- User is in the “wheel” group (group with GID 0).



Restrictiveness

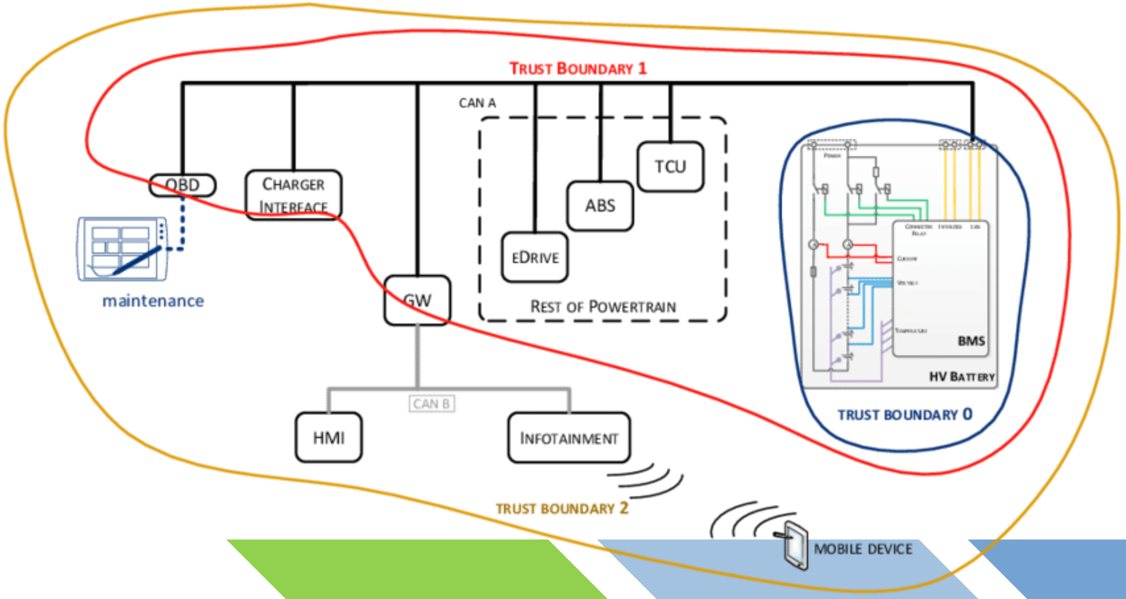
Simplicity

Methodology

7 - Minimize Trust

Software should check all inputs and the results of all security-relevant actions.

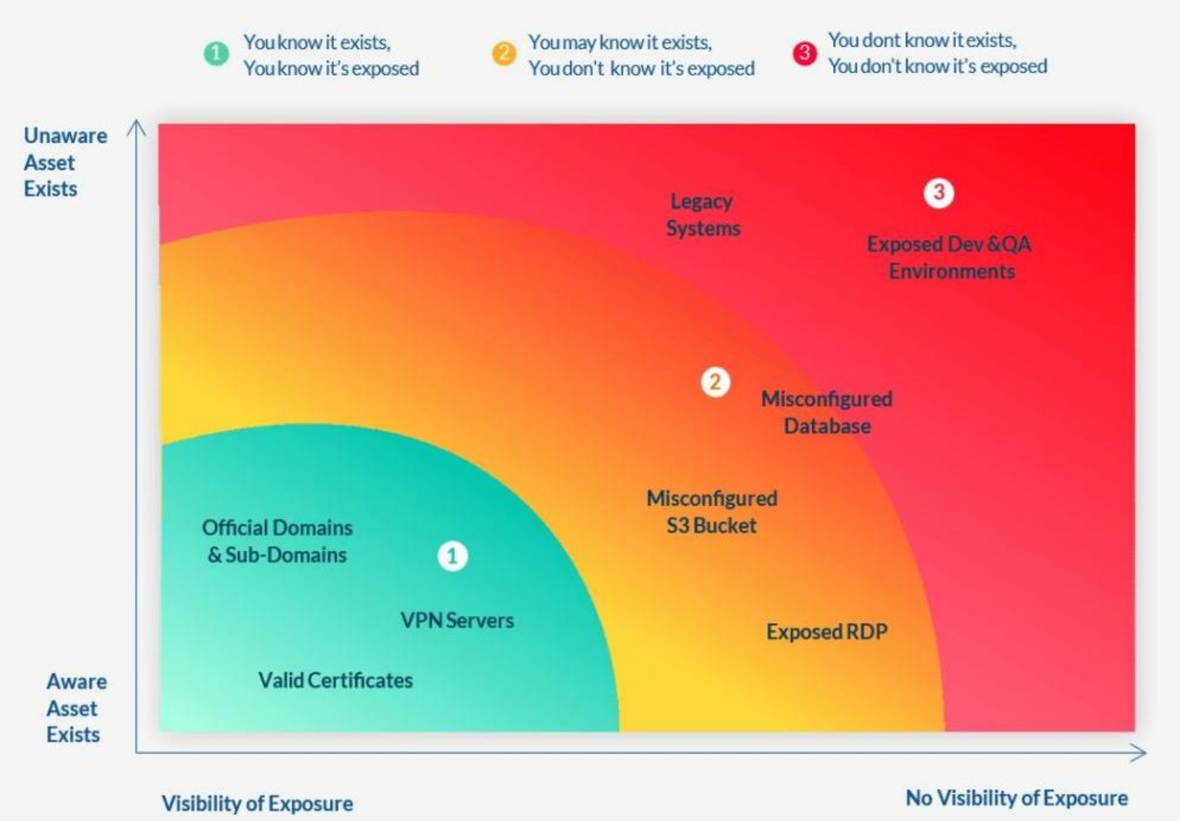
Establish trust boundaries.

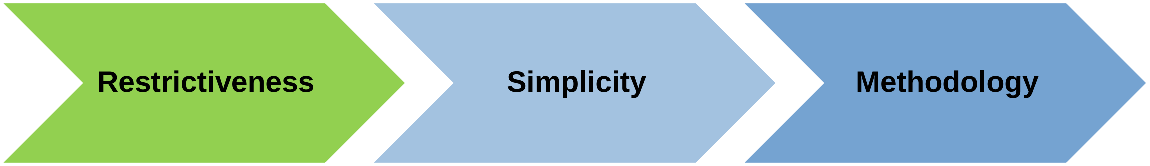


Restrictiveness → Simplicity → Methodology

8 - External components change the attack surface

Software inherits the security weaknesses, security limitations, maintenance responsibilities, and threat model of any software it integrates.

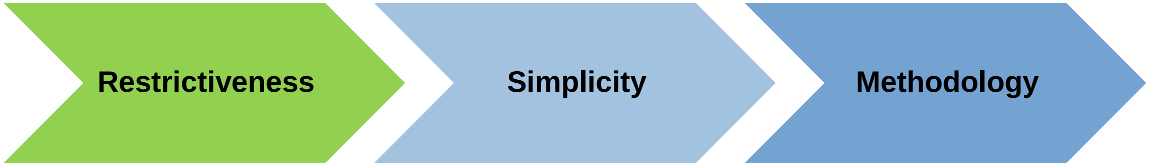

1 You know it exists, You know it's exposed
2 You may know it exists, You don't know it's exposed
3 You dont know it exists, You don't know it's exposed
Unaware Asset Exists
Aware Asset Exists
Legacy Systems
3
Exposed Dev & QA Environments
2
Misconfigured Database
Misconfigured S3 Bucket
Official Domains & Sub-Domains
1
VPN Servers
Exposed RDP
Valid Certificates
Visibility of Exposure
No Visibility of Exposure



9 - Least Common Privilege

Mechanisms used to access resources should not be shared.

- Shared mechanisms, especially involving shared variables, are potential paths between users that can compromise security.
- Separate machines, separate networks, virtual machines can help fulfill this principle and avoid cross-contamination.

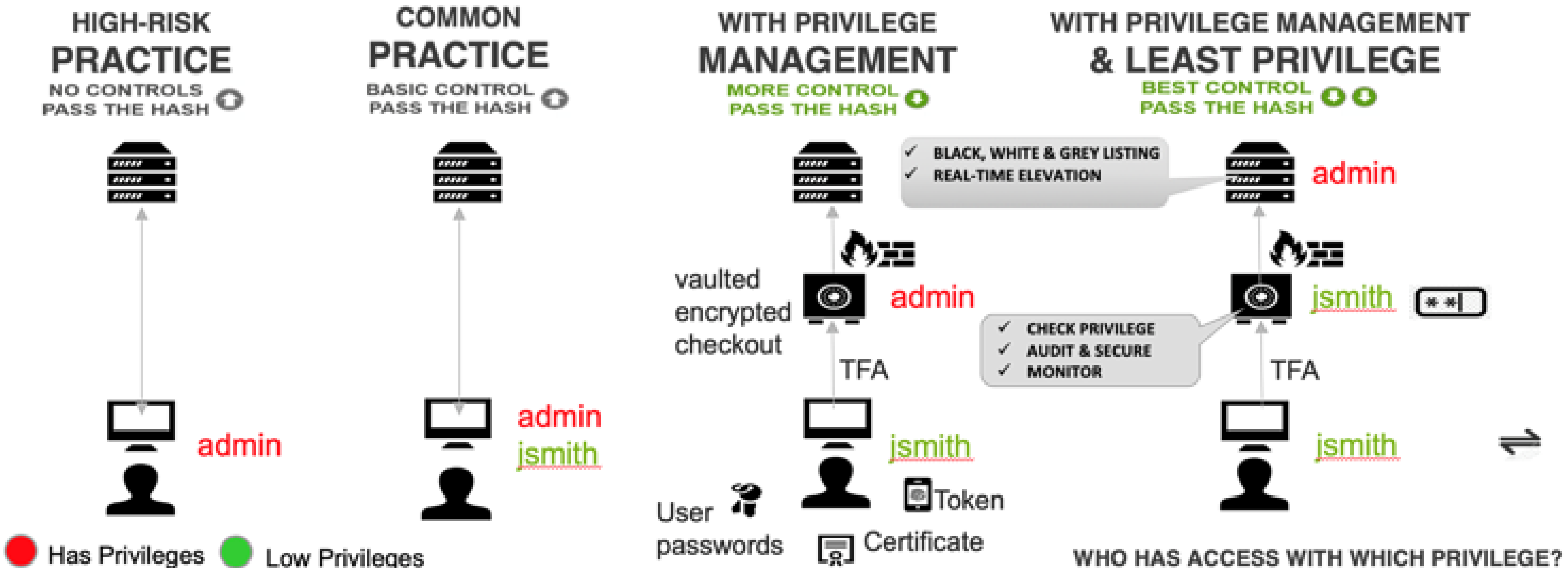


Restrictiveness

Simplicity

Methodology

9 - Least Common Privilege


HIGH-RISK PRACTICE
NO CONTROLS PASS THE HASH
COMMON PRACTICE
BASIC CONTROL PASS THE HASH
WITH PRIVILEGE MANAGEMENT
MORE CONTROL PASS THE HASH
WITH PRIVILEGE MANAGEMENT & LEAST PRIVILEGE
BEST CONTROL PASS THE HASH
✓ BLACK, WHITE & GREY LISTING
✓ REAL-TIME ELEVATION
admin
admin
jsmith
vaulted encrypted checkout
admin
TFA
jsmith
Token
User passwords
Certificate
admin
jsmith
✓ CHECK PRIVILEGE
✓ AUDIT & SECURE
✓ MONITOR
TFA
jsmith
WHO HAS ACCESS WITH WHICH PRIVILEGE?
Has Privileges
Low Privileges

https://thycotic.com/glossary/least-privilege/

Restrictiveness → **Simplicity** → **Methodology**

10 - Economy of mechanism

Keep software design as small and simple as possible.

- Security mechanisms (= security checks, security functionality) should be as simple as possible.
- KISS (Keep it Simple and Small).
- Complex interactions = security verification and checks of systems more difficult.
- Isolate, consolidate, and minimize security controls.

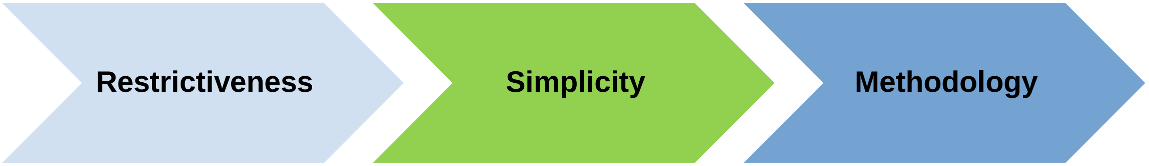
Restrictiveness

Simplicity

Methodology

11 - Psychological acceptability

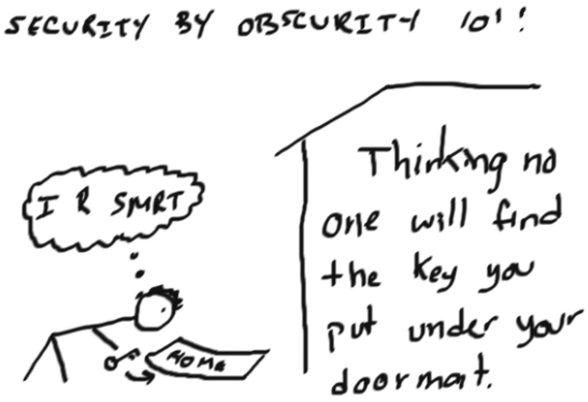
Security features of software and the security mechanisms it implements should be designed so their operation is as logical and simple as possible.

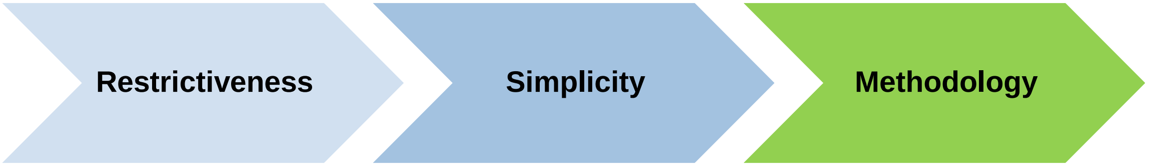

Restrictiveness
Simplicity
Methodology

12 - Open Design

Security of software and of what that software provides should not depend on the secrecy of its design or implementation.

Security by obscurity is not a solution.


SECURITY BY OBSCURITY 101!
I R SMRT
HOME
Thinking no one will find the key you put under your doormat.

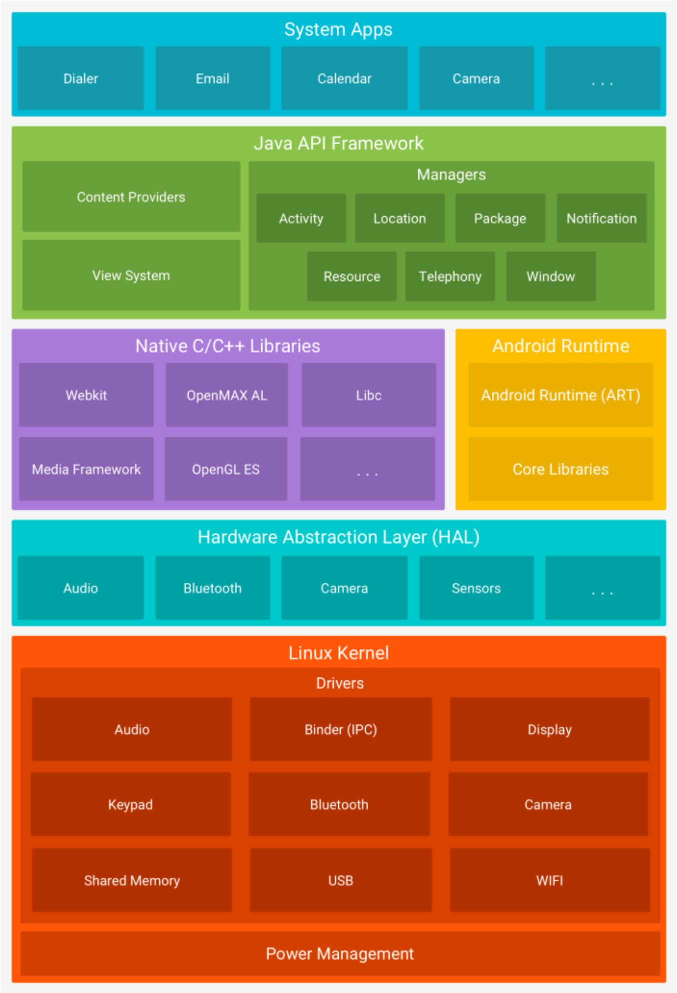

Restrictiveness

Simplicity

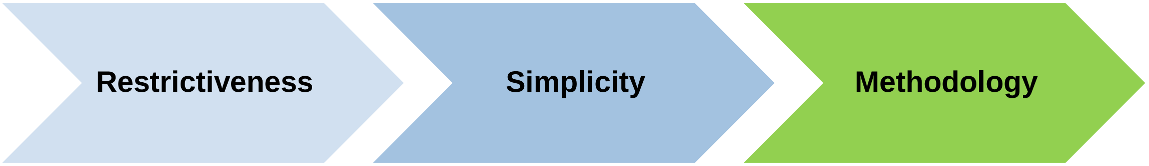
Methodology

13 - Layering

Organize software in layers so that modules at a given layer interact only with modules in the layers immediately above and below it. This allows you to test the software one layer at a time, using either top-down or bottom-up techniques and reduces the access points, enforcing the principle of separation.

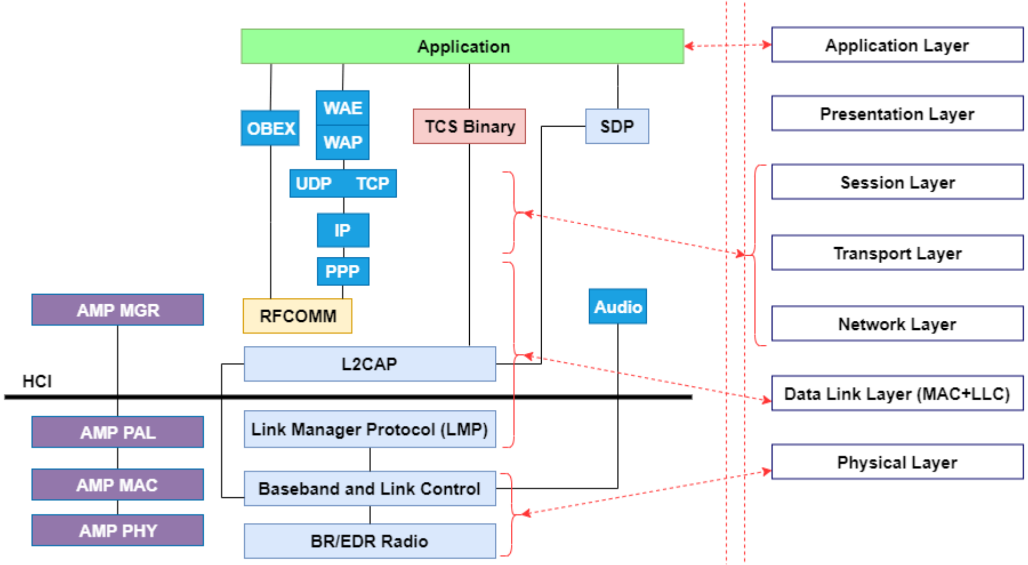


https://developer.android.com/guide/platform


Restrictiveness → **Simplicity** → **Methodology**

14 - Abstraction

Hide the internals of each layer, making only the interfaces available; this enables you to change how a layer carries out its tasks without affecting components at other layers.



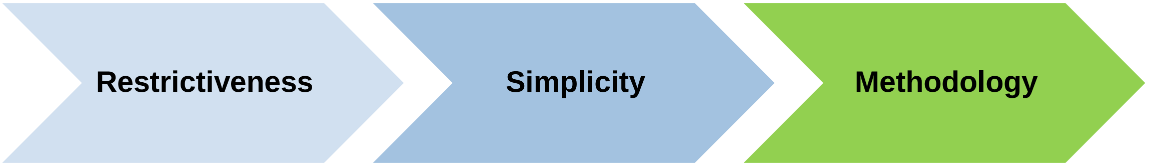
Restrictiveness

Simplicity

Methodology

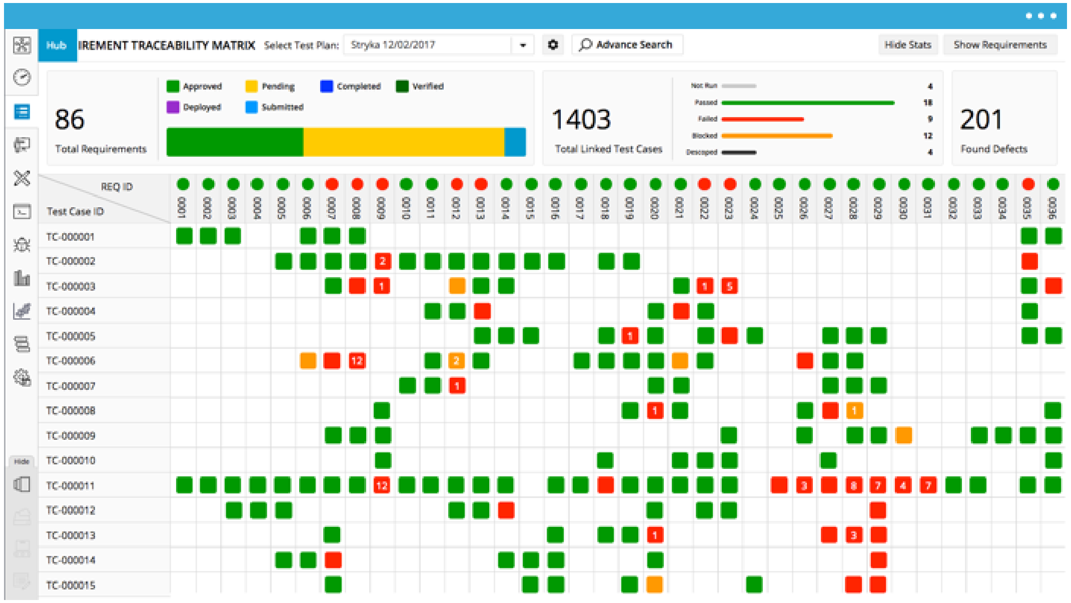
15 - Modularity

Design and implement the software as a collection of co-operating components (modules); indeed, each modules interface is an abstraction.



16 - Complete Linkage

Tie software security design and implementation to the security specifications for that software.



Hub IREMENT TRACEABILITY MATRIX Select Test Plan: Stryka 12/02/2017 Advance Search Hide Stats Show Requirements

86 Total Requirements

🟩 Approved 🟨 Pending 🟦 Completed 🟩 Verified
🟪 Deployed 🟦 Submitted

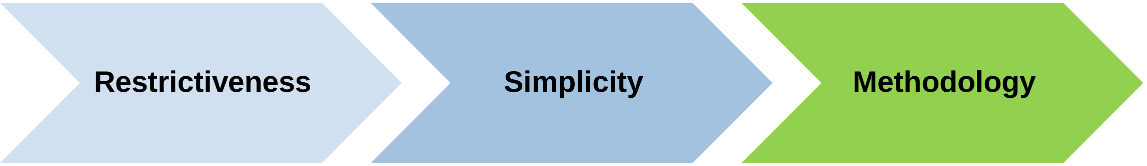
1403 Total Linked Test Cases

Not Run	4
Passed	18
Failed	9
Blocked	12
Descoped	4

201 Found Defects

REQ ID Test Case ID	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015	0016	0017	0018
	🟢	🟢	🟢	🟢	🟢	🟢	🔴	🔴	🔴	🟢	🟢	🔴	🔴	🟢	🟢	🟢	🟢	🟢
TC-000001	🟩	🟩	🟩			🟩	🟩	🟩										
TC-000002					🟩	🟩	🟩	🟩	🟥 2	🟩	🟩	🟩	🟩	🟩	🟩	🟩		🟩
TC-000003							🟩	🟥	🟥 1			🟧	🟩	🟩				
TC-000004											🟩	🟩	🟥					
TC-000005													🟩	🟩	🟩			🟩
TC-000006						🟧	🟥	🟥 12			🟩	🟧 2	🟩				🟩	🟩
TC-000007										🟩	🟩	🟥 1						
TC-000008									🟩									
TC-000009							🟩	🟩	🟩									
TC-000010									🟩									🟩
TC-000011	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟥 12	🟩	🟩	🟩	🟩	🟩		🟩	🟩	🟥
TC-000012			🟩	🟩	🟩							🟩	🟩	🟥				
TC-000013							🟩									🟩		🟩
TC-000014					🟩	🟩	🟥							🟩	🟩	🟩		
TC-000015							🟩								🟩	🟩		

REQ ID Test Case ID	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031	0032	0033	0034	0035	0036
	🟢	🟢	🟢	🔴	🔴	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🔴	🟢
TC-000001																	🟩	🟩
TC-000002	🟩																🟥	
TC-000003			🟩	🟥 1	🟥 5												🟩	🟥
TC-000004		🟩	🟥	🟩													🟩	
TC-000005	🟥 1	🟩		🟩	🟥	🟩			🟩	🟩	🟩						🟩	🟩
TC-000006	🟩	🟩	🟧	🟩				🟥	🟩	🟩								
TC-000007		🟩	🟩						🟩	🟩	🟩							
TC-000008	🟩	🟥 1	🟩					🟩	🟥	🟧 1								🟩
TC-000009					🟩			🟩		🟩	🟩	🟧			🟩	🟩	🟩	🟩
TC-000010			🟩	🟩	🟩				🟩									🟩
TC-000011	🟩	🟩	🟩	🟩	🟩		🟥	🟥 3	🟥	🟥 8	🟥 7	🟥 4	🟥 7	🟩	🟩		🟩	🟩
TC-000012		🟩		🟩	🟩						🟥							
TC-000013	🟩	🟥 1							🟥	🟥 3	🟥							
TC-000014		🟩									🟥							
TC-000015	🟩	🟧				🟩				🟥	🟥							



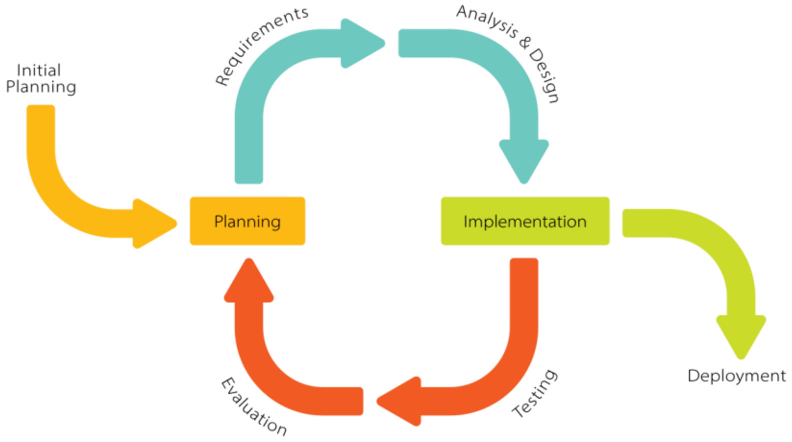
Restrictiveness

Simplicity

Methodology

17 - Design for iteration

Plan the design in such a way that it can be changed, if needed. This minimizes the effects with respect to the security of changing the design if the specifications do not match in an environment that the software is used in.

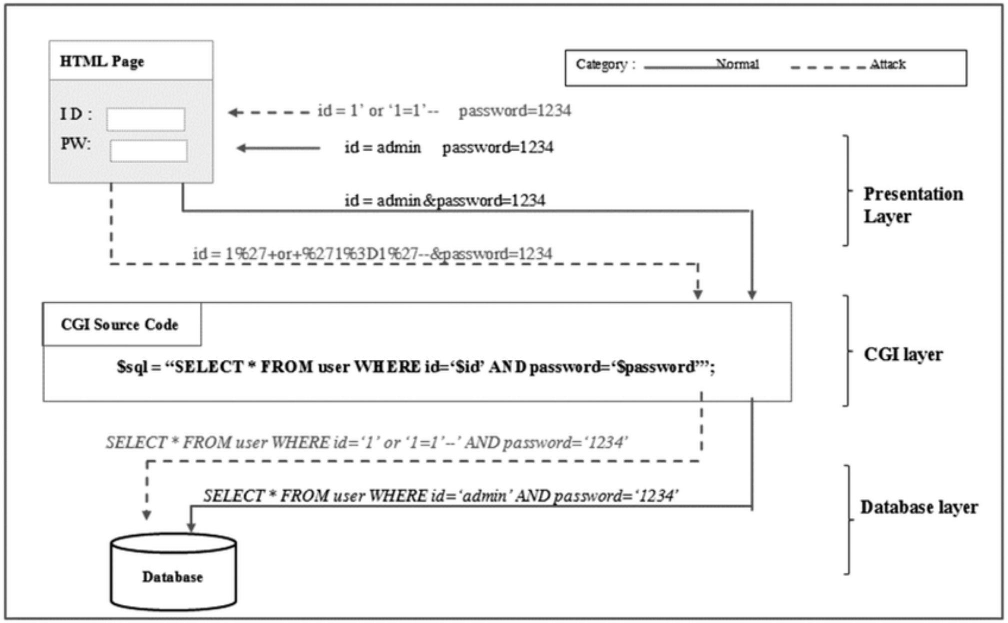


https://en.wikipedia.org/wiki/Iterative_and_incremental_development

Restrictiveness **Simplicity** **Methodology**

18 - Separate data and control instructions

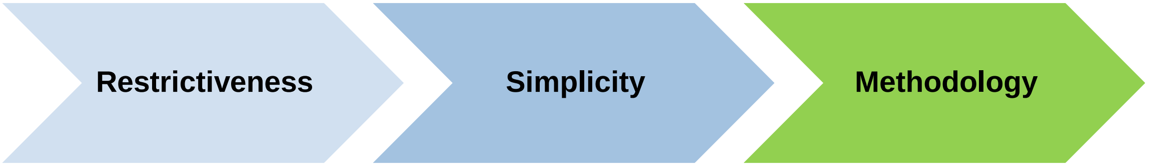
Strictly separate data and control instructions and never process control instructions from untrusted sources.



```
$sql = "SELECT * FROM user WHERE id='$id' AND password='$password'";
```

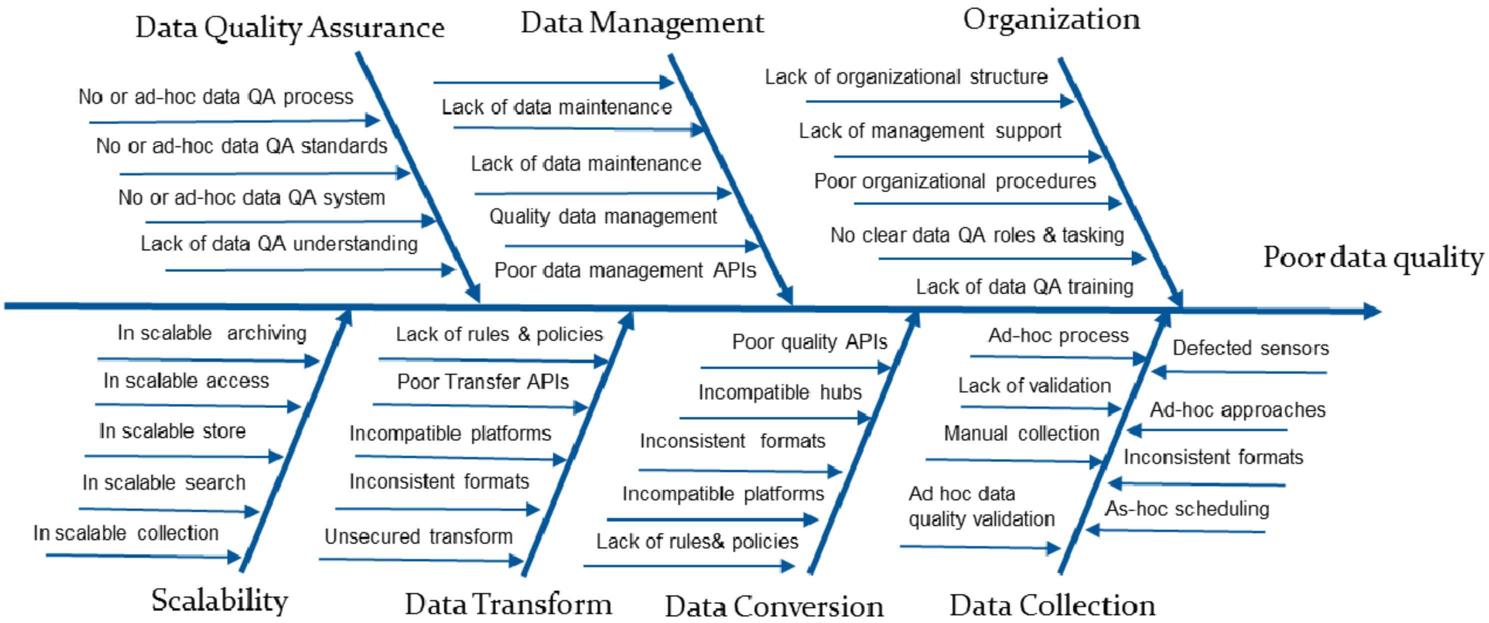
```
SELECT * FROM user WHERE id='1' or '1=1'--' AND password='1234'
```

```
SELECT * FROM user WHERE id='admin' AND password='1234'
```



19 - Data Validation

Define an approach that ensures all data are explicitly validated.



https://d3i71xaburhd42.cloudfront.net/ea892cd1b650bb3b8b9604e2d741486a962d7847/3-Figure1-1.png

Restrictiveness

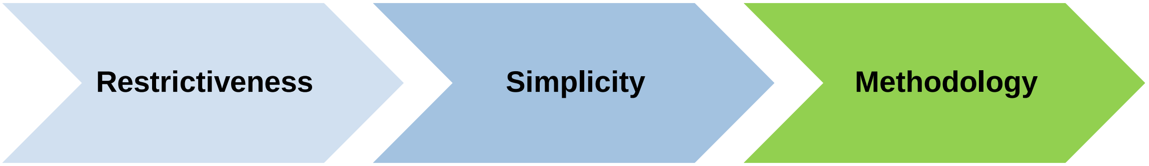
Simplicity

Methodology

20 - Cryptography

Use cryptography CORRECTLY.

Allowing to protect the confidentiality of data, protect data from unauthorized modification, and authenticate the source of data.



Restrictiveness

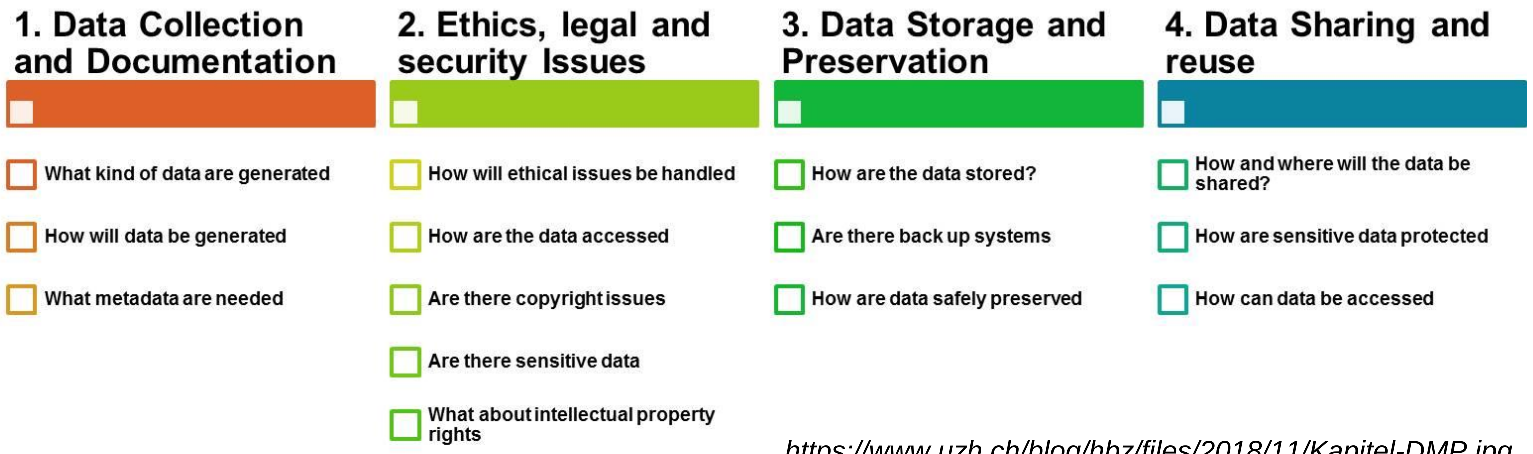
Simplicity

Methodology

21 - Data sensitivity

Identify sensitive data and how they should be handled.

→ Data Handling Plan



1. Data Collection and Documentation

□ What kind of data are generated

□ How will data be generated

□ What metadata are needed

2. Ethics, legal and security Issues

□ How will ethical issues be handled

□ How are the data accessed

□ Are there copyright issues

□ Are there sensitive data

□ What about intellectual property rights

3. Data Storage and Preservation

□ How are the data stored?

□ Are there back up systems

□ How are data safely preserved

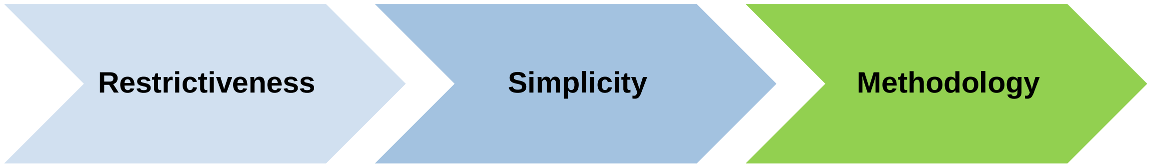
4. Data Sharing and reuse

□ How and where will the data be shared?

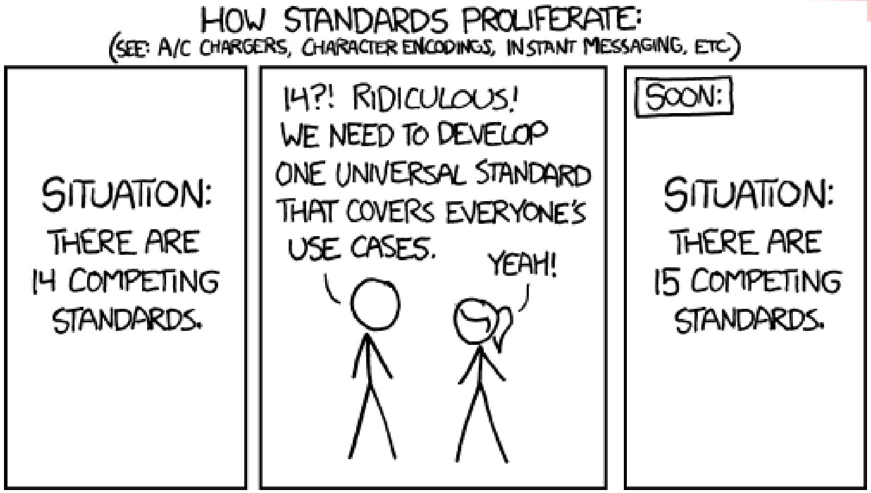
□ How are sensitive data protected

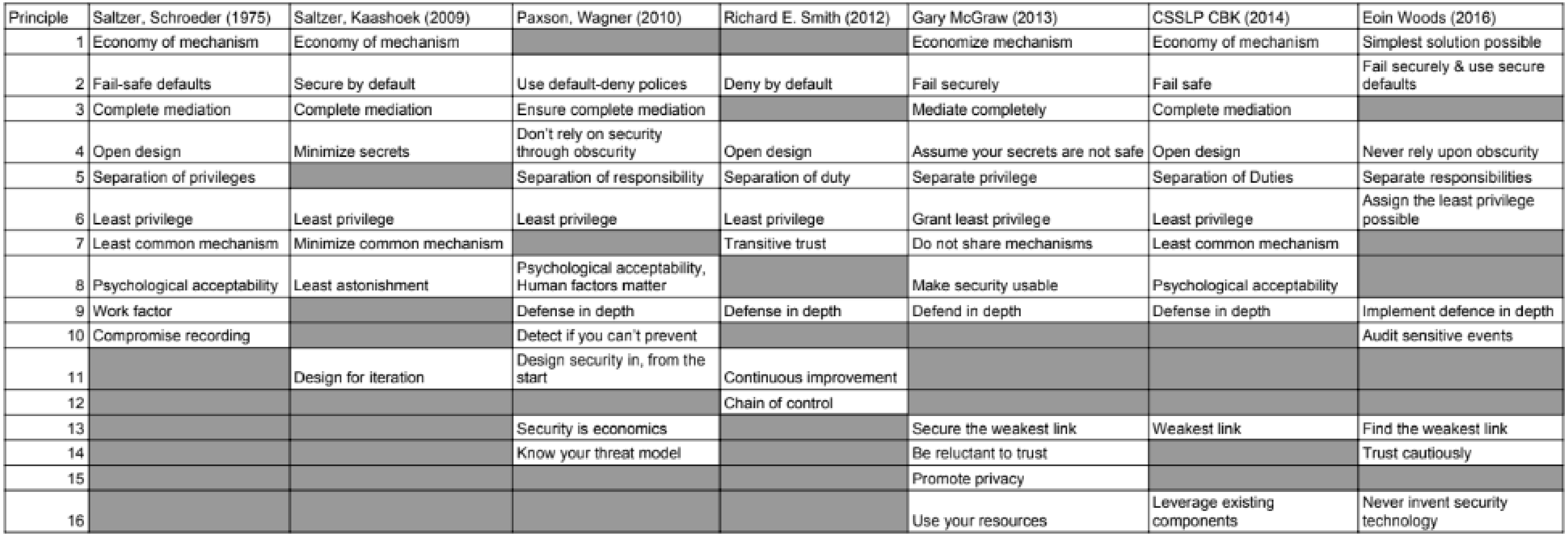
□ How can data be accessed

https://www.uzh.ch/blog/hbz/files/2018/11/Kapitel-DMP.jpg



Different principles and standards Overview

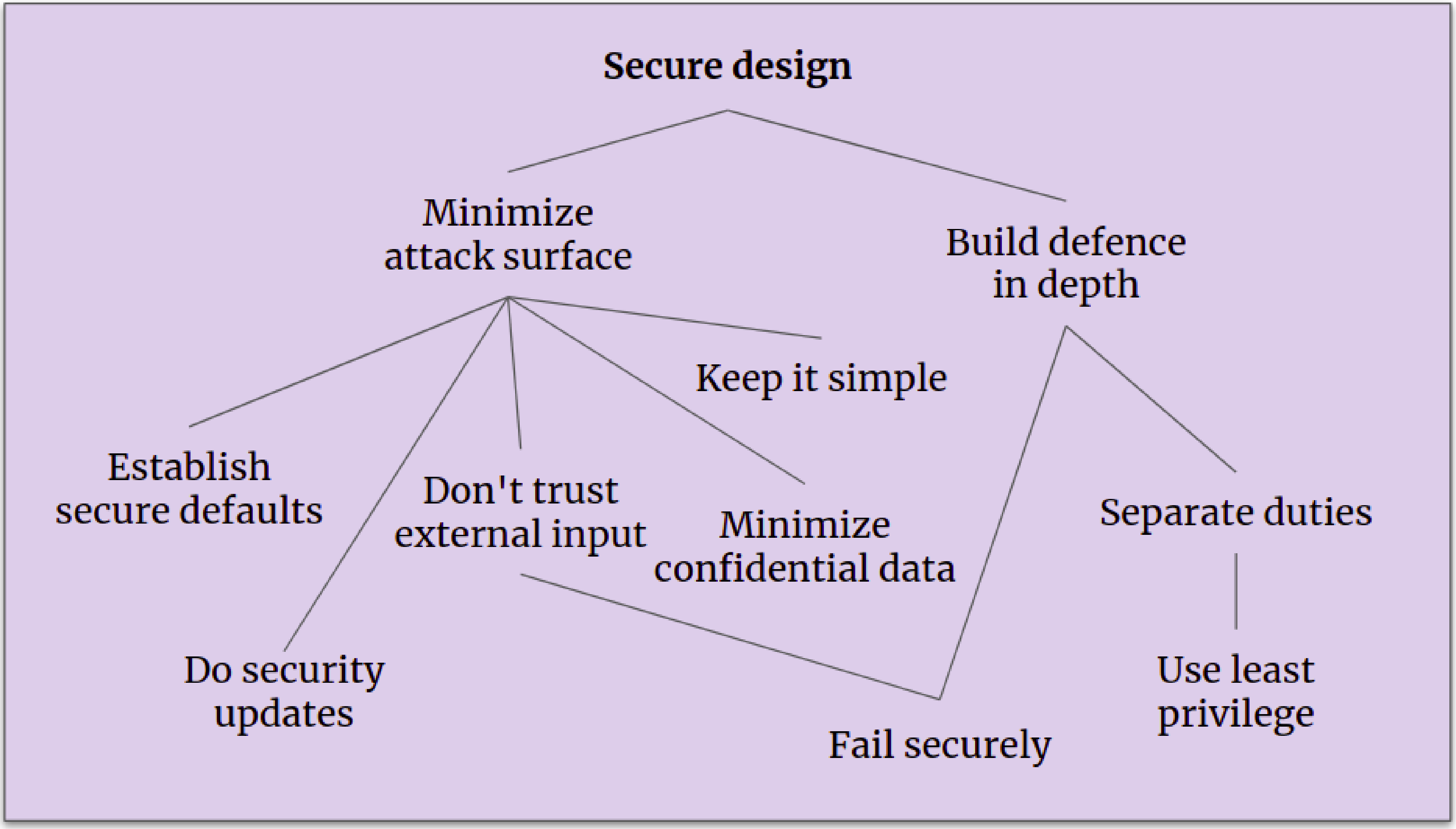

HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.



Principle	Saltzer, Schroeder (1975)	Saltzer, Kaashoek (2009)	Paxson, Wagner (2010)	Richard E. Smith (2012)	Gary McGraw (2013)	CSSLP CBK (2014)	Eoin Woods (2016)
1	Economy of mechanism	Economy of mechanism			Economize mechanism	Economy of mechanism	Simplest solution possible
2	Fail-safe defaults	Secure by default	Use default-deny polices	Deny by default	Fail securely	Fail safe	Fail securely & use secure defaults
3	Complete mediation	Complete mediation	Ensure complete mediation		Mediate completely	Complete mediation	
4	Open design	Minimize secrets	Don't rely on security through obscurity	Open design	Assume your secrets are not safe	Open design	Never rely upon obscurity
5	Separation of privileges		Separation of responsibility	Separation of duty	Separate privilege	Separation of Duties	Separate responsibilities
6	Least privilege	Least privilege	Least privilege	Least privilege	Grant least privilege	Least privilege	Assign the least privilege possible
7	Least common mechanism	Minimize common mechanism		Transitive trust	Do not share mechanisms	Least common mechanism	
8	Psychological acceptability	Least astonishment	Psychological acceptability, Human factors matter		Make security usable	Psychological acceptability	
9	Work factor		Defense in depth	Defense in depth	Defend in depth	Defense in depth	Implement defence in depth
10	Compromise recording		Detect if you can't prevent				Audit sensitive events
11		Design for iteration	Design security in, from the start	Continuous improvement			
12				Chain of control			
13			Security is economics		Secure the weakest link	Weakest link	Find the weakest link
14			Know your threat model		Be reluctant to trust		Trust cautiously
15					Promote privacy		
16					Use your resources	Leverage existing components	Never invent security technology

https://medium.com/@girishsj2/security-design-principles-466dda3bf4fb

Principles and standards - overview



https://medium.com/ouspg/security-design-with-principles-a8c045765b93

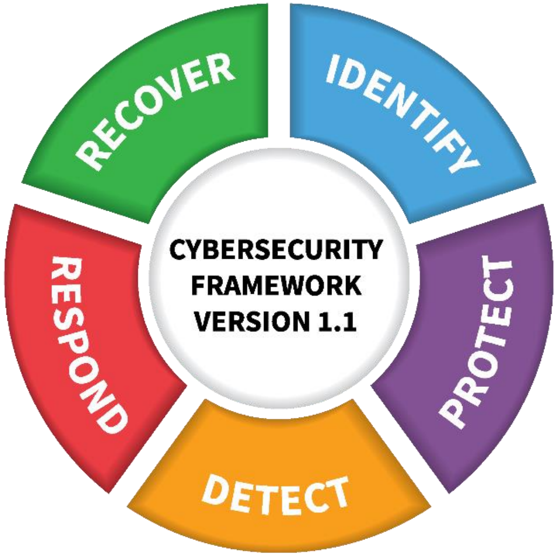
Security by Design

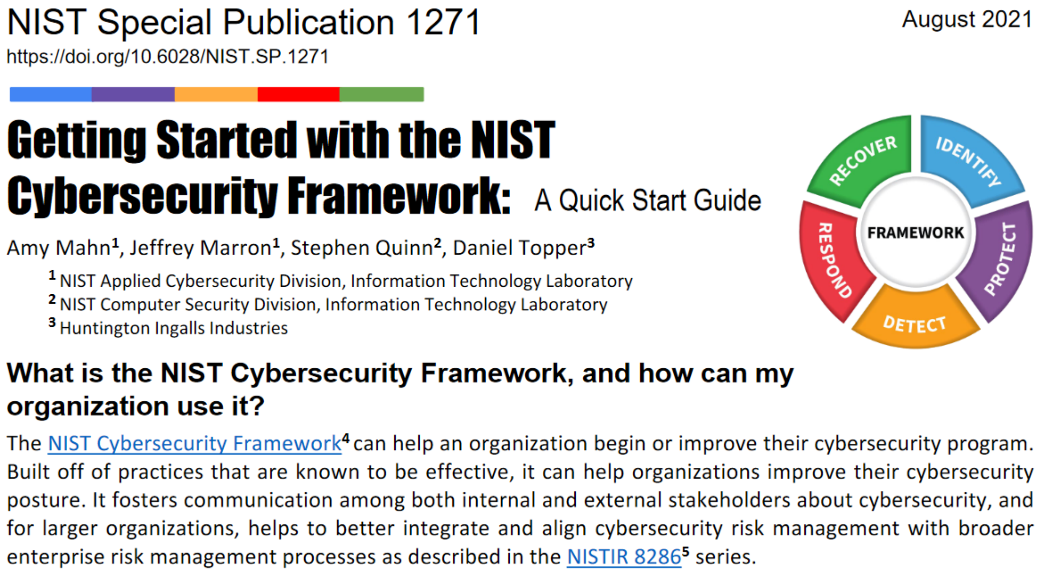
“*This whole economic boom in cybersecurity seems largely to be a consequence of poor engineering*.”
Carl Landwehr, Communications of the ACM, February 2015

What is NIST?

NIST = National Institute of Standards and Technology

Agency of the United State of America Government primarily managing technology aspects in the country.




RECOVER
IDENTIFY
PROTECT
DETECT
RESPOND
FRAMEWORK

NIST Special Publication 1271

August 2021

https://doi.org/10.6028/NIST.SP.1271

Getting Started with the NIST Cybersecurity Framework: A Quick Start Guide

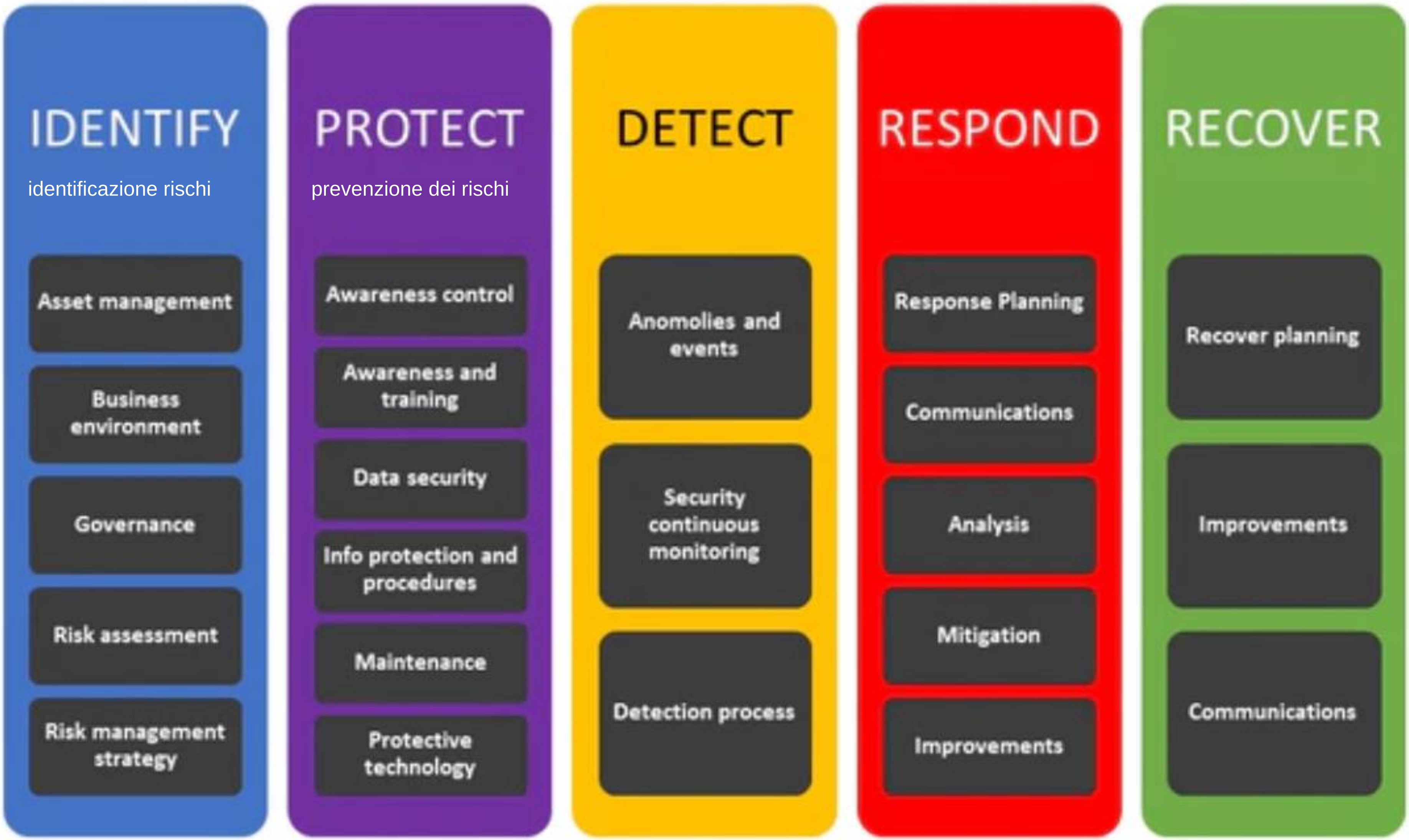
Amy Mahn[1], Jeffrey Marron[1], Stephen Quinn[2], Daniel Topper[3]

[1] NIST Applied Cybersecurity Division, Information Technology Laboratory
[2] NIST Computer Security Division, Information Technology Laboratory
[3] Huntington Ingalls Industries

What is the NIST Cybersecurity Framework, and how can my organization use it?

The NIST Cybersecurity Framework[4] can help an organization begin or improve their cybersecurity program. Built off of practices that are known to be effective, it can help organizations improve their cybersecurity posture. It fosters communication among both internal and external stakeholders about cybersecurity, and for larger organizations, helps to better integrate and align cybersecurity risk management with broader enterprise risk management processes as described in the NISTIR 8286[5] series.

NIST Cybersecurity Framework



IDENTIFY

identificazione rischi

- Asset management
- Business environment
- Governance
- Risk assessment
- Risk management strategy

PROTECT

prevenzione dei rischi

- Awareness control
- Awareness and training
- Data security
- Info protection and procedures
- Maintenance
- Protective technology

DETECT

- Anomolies and events
- Security continuous monitoring
- Detection process

RESPOND

- Response Planning
- Communications
- Analysis
- Mitigation
- Improvements

RECOVER

- Recover planning
- Improvements
- Communications

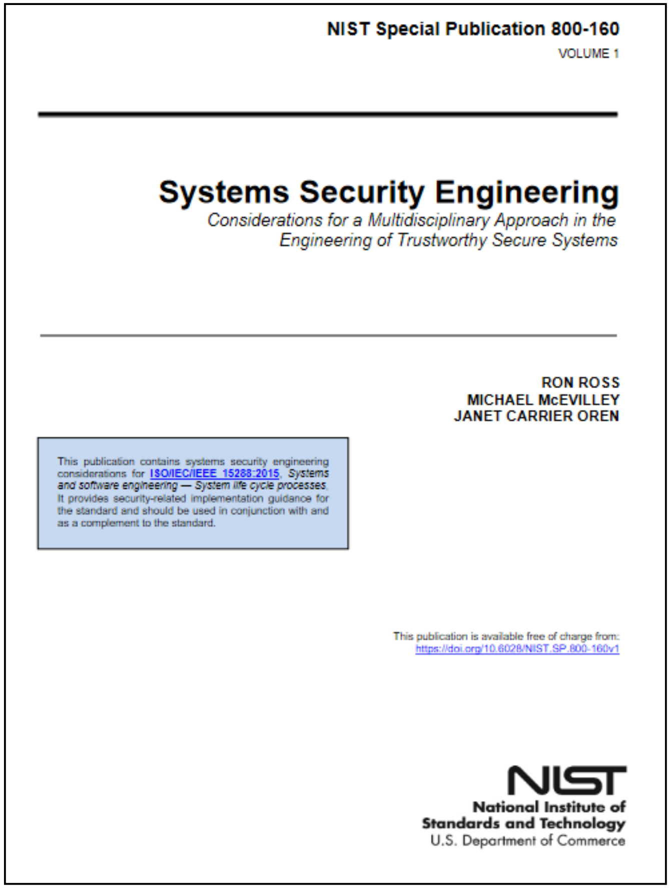
NIST Special Publication

In November 2016 the institute published:

NIST Special Publication 800-160 - Volume 1

Systems Security Engineering

Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems



NIST Special Publication 800-160
VOLUME 1

Systems Security Engineering

Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems

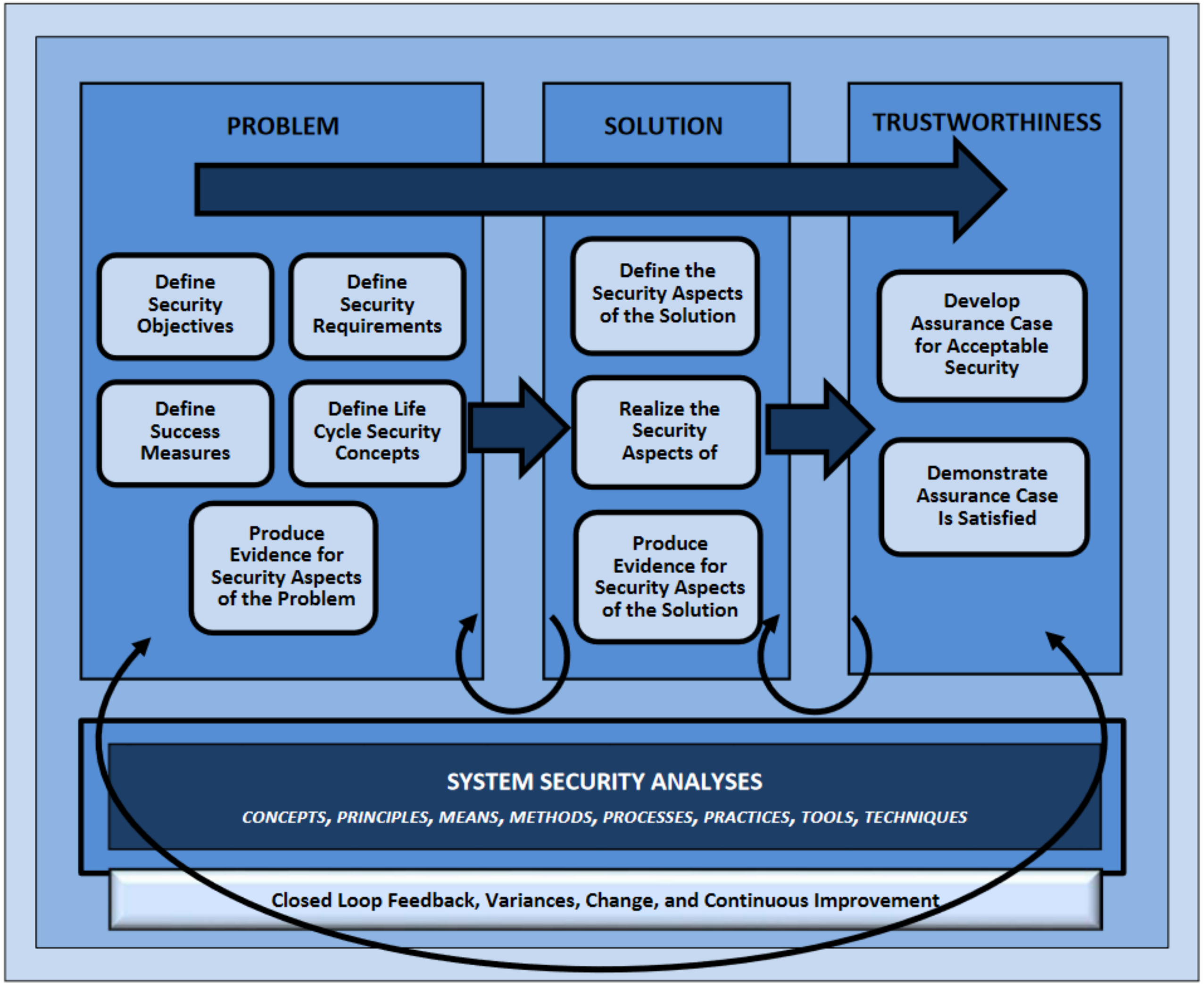
RON ROSS
MICHAEL McEVILLEY
JANET CARRIER OREN

This publication contains systems security engineering considerations for ISO/IEC/IEEE 15288:2015, *Systems and software engineering — System life cycle processes*. It provides security-related implementation guidance for the standard and should be used in conjunction with and as a complement to the standard.

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-160v1

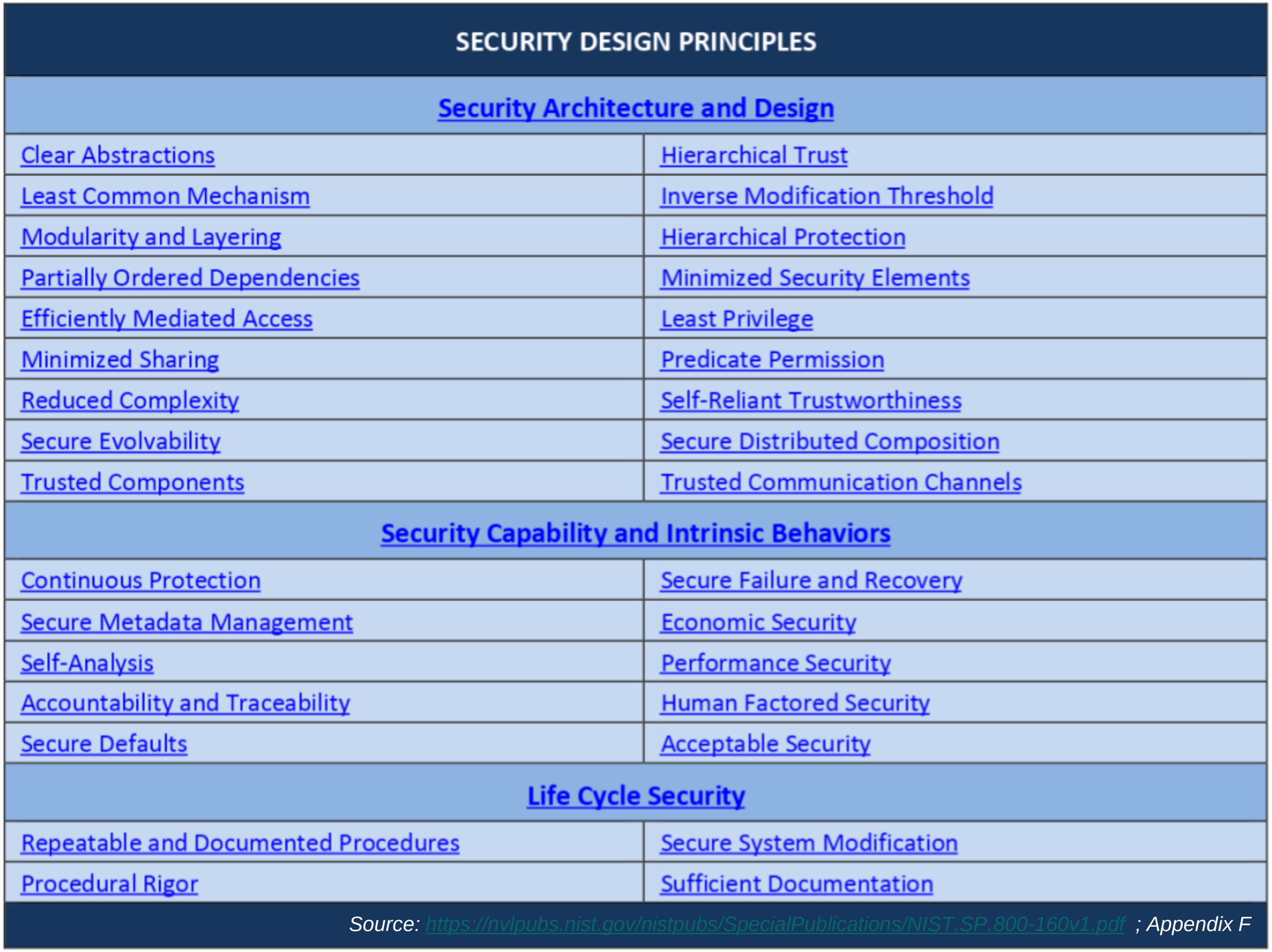
NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

NIST Security


PROBLEM
SOLUTION
TRUSTWORTHINESS
Define Security Objectives
Define Security Requirements
Define Success Measures
Define Life Cycle Security Concepts
Produce Evidence for Security Aspects of the Problem
Define the Security Aspects of the Solution
Realize the Security Aspects of
Produce Evidence for Security Aspects of the Solution
Develop Assurance Case for Acceptable Security
Demonstrate Assurance Case Is Satisfied
SYSTEM SECURITY ANALYSES
CONCEPTS, PRINCIPLES, MEANS, METHODS, PROCESSES, PRACTICES, TOOLS, TECHNIQUES
Closed Loop Feedback, Variances, Change, and Continuous Improvement

Source: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160v1.pdf

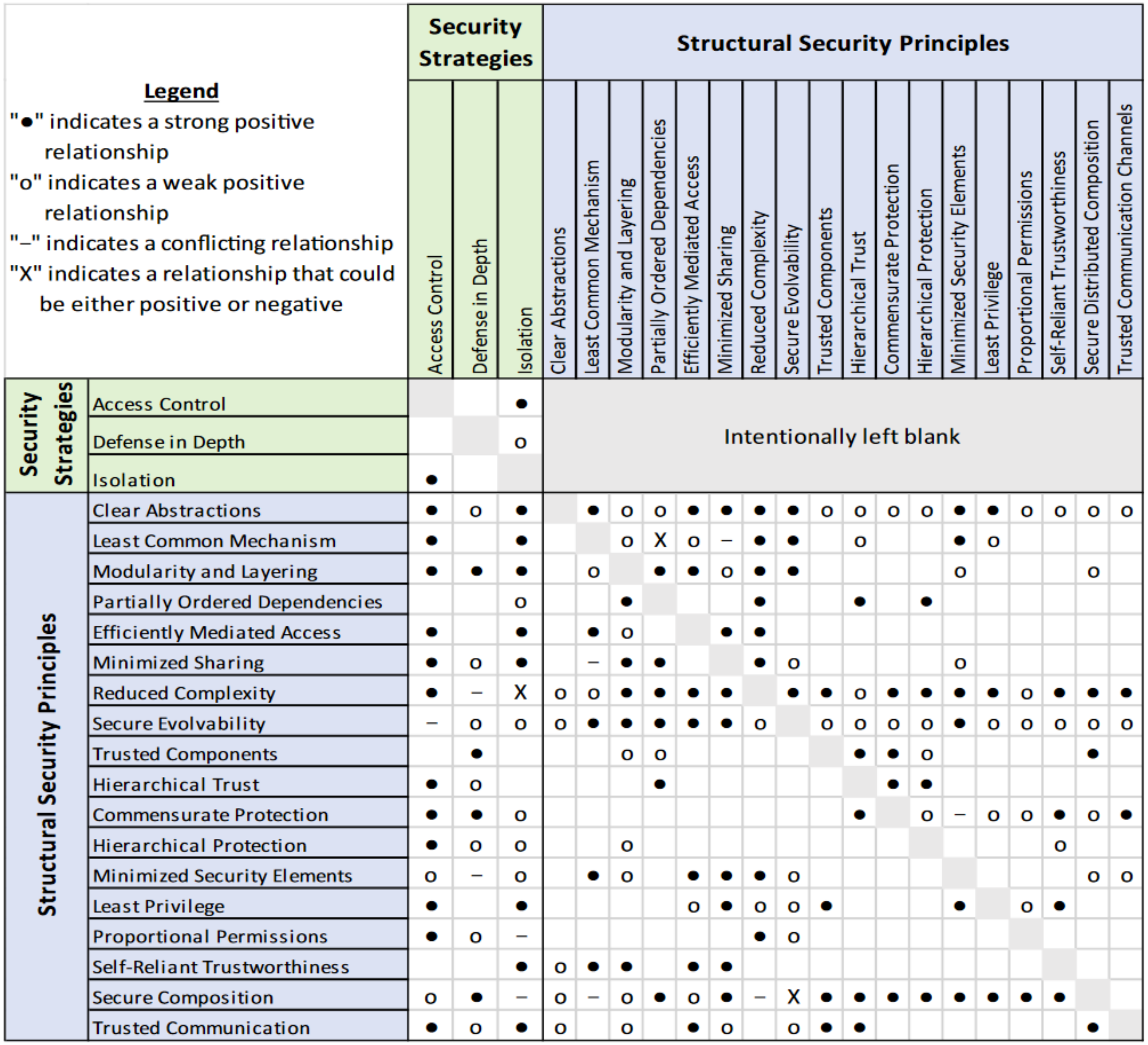
NIST Security Design Principles



SECURITY DESIGN PRINCIPLES	
Security Architecture and Design	
Clear Abstractions	Hierarchical Trust
Least Common Mechanism	Inverse Modification Threshold
Modularity and Layering	Hierarchical Protection
Partially Ordered Dependencies	Minimized Security Elements
Efficiently Mediated Access	Least Privilege
Minimized Sharing	Predicate Permission
Reduced Complexity	Self-Reliant Trustworthiness
Secure Evolvability	Secure Distributed Composition
Trusted Components	Trusted Communication Channels
Security Capability and Intrinsic Behaviors	
Continuous Protection	Secure Failure and Recovery
Secure Metadata Management	Economic Security
Self-Analysis	Performance Security
Accountability and Traceability	Human Factored Security
Secure Defaults	Acceptable Security
Life Cycle Security	
Repeatable and Documented Procedures	Secure System Modification
Procedural Rigor	Sufficient Documentation

Source: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160v1.pdf ; Appendix F

NIST - Mapping of System Security Strategies to Design Principles



Legend

"●" indicates a strong positive relationship

"o" indicates a weak positive relationship

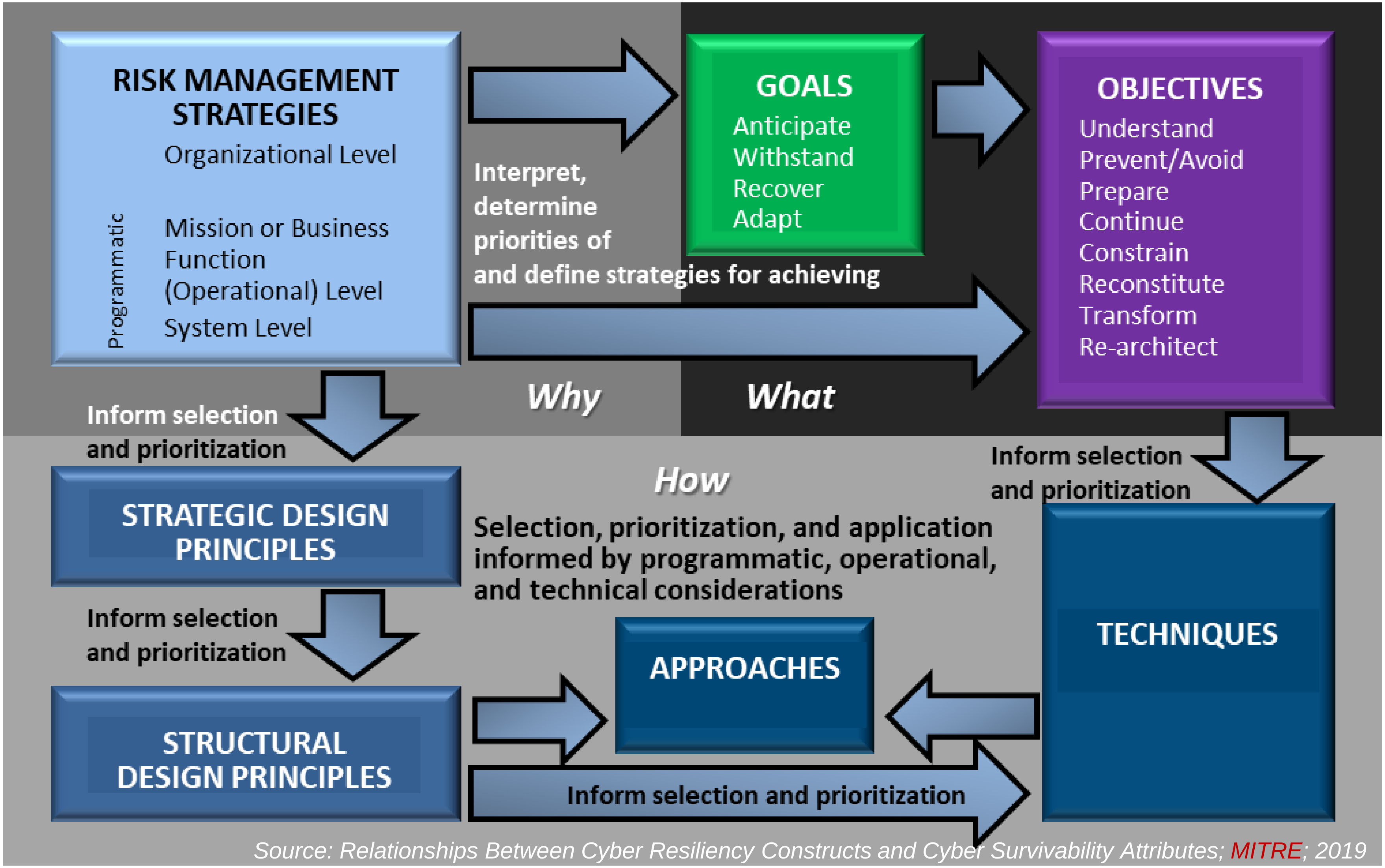
"–" indicates a conflicting relationship

"X" indicates a relationship that could be either positive or negative

		Security Strategies			**Structural Security Principles**																	
		Access Control	Defense in Depth	Isolation	Clear Abstractions	Least Common Mechanism	Modularity and Layering	Partially Ordered Dependencies	Efficiently Mediated Access	Minimized Sharing	Reduced Complexity	Secure Evolvability	Trusted Components	Hierarchical Trust	Commensurate Protection	Hierarchical Protection	Minimized Security Elements	Least Privilege	Proportional Permissions	Self-Reliant Trustworthiness	Secure Distributed Composition	Trusted Communication Channels
Security Strategies	Access Control			●	Intentionally left blank																	
	Defense in Depth			o																		
	Isolation	●																				
Structural Security Principles	Clear Abstractions	●	o	●		●	o	o	●	●	●	●	o	o	o	o	●	●	o	o	o	o
	Least Common Mechanism	●		●			o	X	o	–	●	●		o			●	o				
	Modularity and Layering	●	●	●		o		●	●	o	●	●					o				o	
	Partially Ordered Dependencies			o			●				●			●		●						
	Efficiently Mediated Access	●		●		●	o			●	●											
	Minimized Sharing	●	o	●		–	●	●			●	o					o					
	Reduced Complexity	●	–	X	o	o	●	●	●	●		●	●	o	●	●	●	●	o	●	●	●
	Secure Evolvability	–	o	o	o	●	●	●	●	●	o		o	o	o	o	●	o	o	o	o	o
	Trusted Components		●				o	o						●	●	o					●	
	Hierarchical Trust	●	o					●							●	●						
	Commensurate Protection	●	●	o										●		o	–	o	o	●	o	●
	Hierarchical Protection	●	o	o			o													o		
	Minimized Security Elements	o	–	o		●	o		●	●	●	o									o	o
	Least Privilege	●		●					o	●	o	o	●				●		o	●		
	Proportional Permissions	●	o	–							●	o										
	Self-Reliant Trustworthiness			●	o	●	●		●	●												
	Secure Composition	o	●	–	o	–	o	●	o	●	–	X	●	●	●	●	●	●	●	●		
	Trusted Communication	●	o	●	o		o		●	o		o	●	●							●	

Source: Examination of Security Design Principles from NIST SP 800-160; 2018

Cyber Resiliency Constructs


RISK MANAGEMENT STRATEGIES
Organizational Level
Programmatic
Mission or Business Function (Operational) Level
System Level
Interpret, determine priorities of and define strategies for achieving
GOALS
Anticipate
Withstand
Recover
Adapt
OBJECTIVES
Understand
Prevent/Avoid
Prepare
Continue
Constrain
Reconstitute
Transform
Re-architect
Why
What
Inform selection and prioritization
STRATEGIC DESIGN PRINCIPLES
How
Selection, prioritization, and application informed by programmatic, operational, and technical considerations
Inform selection and prioritization
TECHNIQUES
Inform selection and prioritization
APPROACHES
STRUCTURAL DESIGN PRINCIPLES
Inform selection and prioritization

Source: Relationships Between Cyber Resiliency Constructs and Cyber Survivability Attributes; MITRE; 2019

NIST Security Design Principles

F1 - SECURITY ARCHITECTURE AND DESIGN

NIST Security Design Principles

F1 - Security Architecture and Design

- F1.1 Clear Abstraction
- F1.2 Least Common Mechanism
- F1.3 Modularity and Layering
- F1.4 Partially Ordered Dependencies
- F1.5 Efficiently Mediated Access
- F1.6 Minimized Sharing
- F1.7 Reduced Complexity
- F1.8 Secure Evolvability
- F1.9 Trusted Components

NIST Security Design Principles

F1 - Security Architecture and Design

- F1.10 Hierarchical Trust
- F1.11 Inverse Modification Threshold
- F1.12 Hierarchical Protection
- F1.13 Minimized Security Elements
- F1.14 Least Privilege
- F1.15 Predicate Permission
- F1.16 Self-Reliant Trustworthiness
- F1.17 Secure Distributed Composition
- F1.18 Trusted Communication Channels

NIST Security Design Principles

F2 - SECURITY CAPABILITY AND INTRINSIC BEHAVIORS

NIST Security Design Principles

F2 - Security Capability and Intrinsic Behaviors

- F2.1 Continuous Protection
- F2.2 Secure Metadata Management
- F2.3 Self-Analysis
- F2.4 Accountability and Traceability
- F2.5 Secure Defaults
- F2.6 Secure Failure and Recovery
- F2.7 Economic Security
- F2.8 Performance Security
- F2.9 Human Factored Security
- F2.10 Accetable Security

NIST Security Design Principles

F3 - LIFE CYCLE SECURITY

NIST Security Design Principles

F3 - Life Cycle Security

- F3.1 Repeatable and Documented Procures
- F3.2 Procedural Rigor
- F3.3 Secure System Modification
- F3.4 Sufficient Documentation

NIST Security Design Principles

F4 - APPROACHES TO TRUSTWORTHY SECURE SYSTEM DEVELOPMENT

NIST Security Design Principles

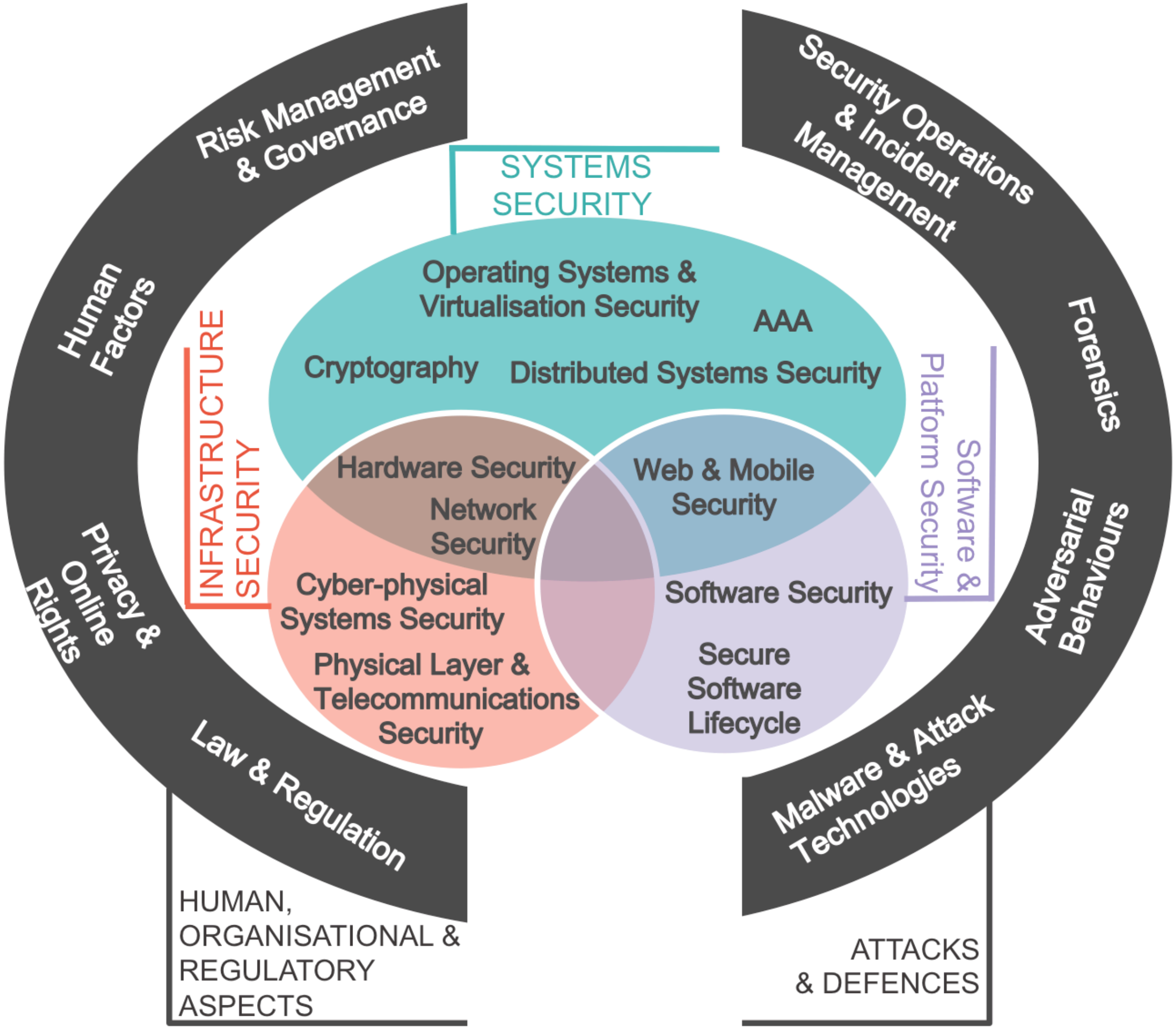
F4 - Approaches to Trustworthy Secure System Development

- F4.1 Reference Monitor Concept
- F4.2 Defense in Depth
- F4.3 Isolation

The Cybersecurity Body of Knowledge

CYBOK

CyBoK Security


Risk Management & Governance
Human Factors
Privacy & Online Rights
Law & Regulation
HUMAN, ORGANISATIONAL & REGULATORY ASPECTS
Security Operations & Incident Management
Forensics
Adversarial Behaviours
Malware & Attack Technologies
ATTACKS & DEFENCES
SYSTEMS SECURITY
Operating Systems & Virtualisation Security
AAA
Cryptography
Distributed Systems Security
INFRASTRUCTURE SECURITY
Hardware Security
Network Security
Cyber-physical Systems Security
Physical Layer & Telecommunications Security
Software & Platform Security
Web & Mobile Security
Software Security
Secure Software Lifecycle