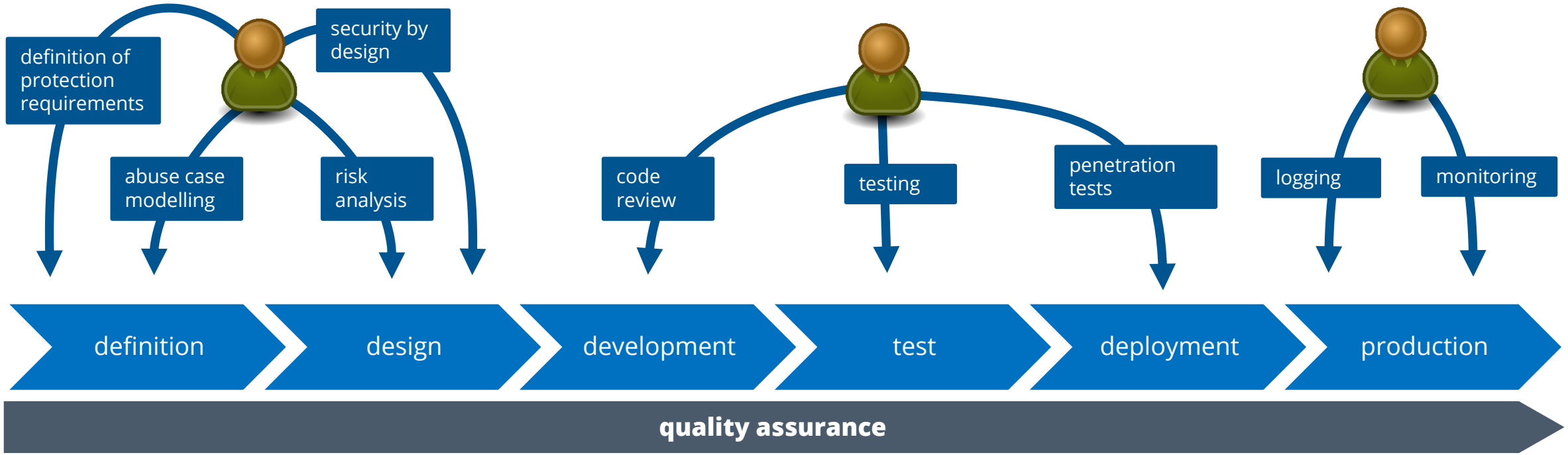
Secure Coding / SSDLC

Security in the SDLC

Points of Action



create awareness train developers and people in charge	adjust contracts with external service providers adjust contract specifications
secure coding guidelines secure coding checklist	Adjustment of processes and organisation
buildup of a second security line (WAF) align infrastructure with WAS interests	

How to Introduce Security into Agile Development Processes

1. TRAINING	2. REQUIREMENTS	3. DESIGN	4. IMPLEMENTATION	5. VERIFICATION	6. RELEASE	7. RESPONSE
1. Core Security Training	2. Establish Security Requirements	5. Establish Design Requirements	8. Use Approved Tools	11. Perform Dynamic Analysis	14. Create an Incident Response Plan	Execute Incident Response Plan
	3. Create Quality Gates/Bug Bars	6. Perform Attack Surface Analysis/ Reduction	9. Deprecate Unsafe Functions	12. Perform Fuzz Testing	15. Conduct Final Security Review	
	4. Perform Security and Privacy Risk Assessments	7. Use Threat Modeling	10. Perform Static Analysis	13. Conduct Attack Surface Review	16. Certify Release and Archive	

src: https://www.microsoft.com/en-us/SDL/process/training.aspx

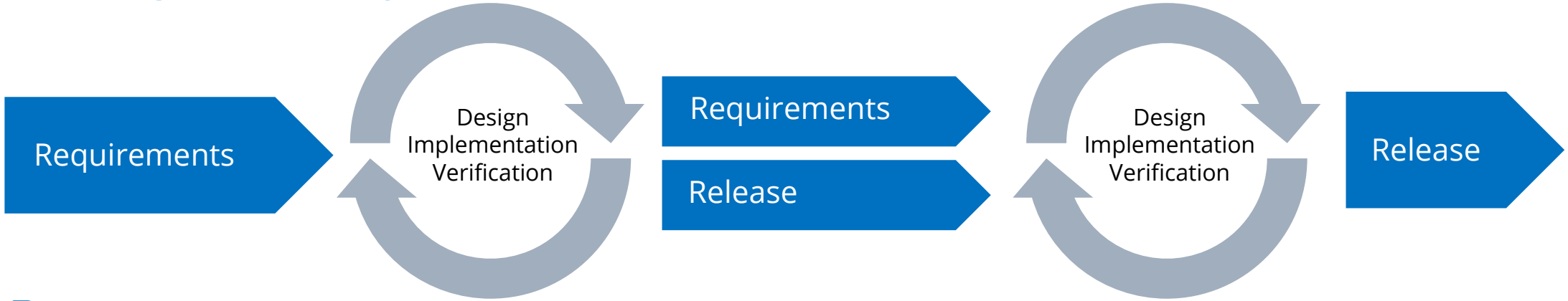
From Waterfall…

How to Introduce Security into Agile Development Processes

1. TRAINING	2. REQUIREMENTS	3. DESIGN	4. IMPLEMENTATION	5. VERIFICATION	6. RELEASE	7. RESPONSE
1. Core Security Training	2. Establish Security Requirements	5. Establish Design Requirements	8. Use Approved Tools	11. Perform Dynamic Analysis	14. Create an Incident Response Plan	Execute Incident Response Plan
	3. Create Quality Gates/Bug Bars	6. Perform Attack Surface Analysis/ Reduction	9. Deprecate Unsafe Functions	12. Perform Fuzz Testing	15. Conduct Final Security Review	
	4. Perform Security and Privacy Risk Assessments	7. Use Threat Modeling	10. Perform Static Analysis	13. Conduct Attack Surface Review	16. Certify Release and Archive	

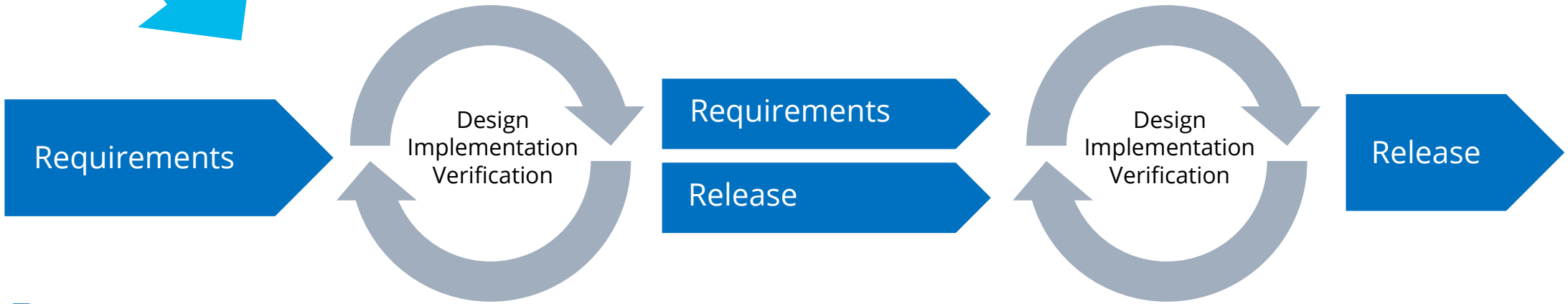
src: https://www.microsoft.com/en-us/SDL/process/training.aspx

… to agile Development



How to Introduce Security into Agile Development Processes

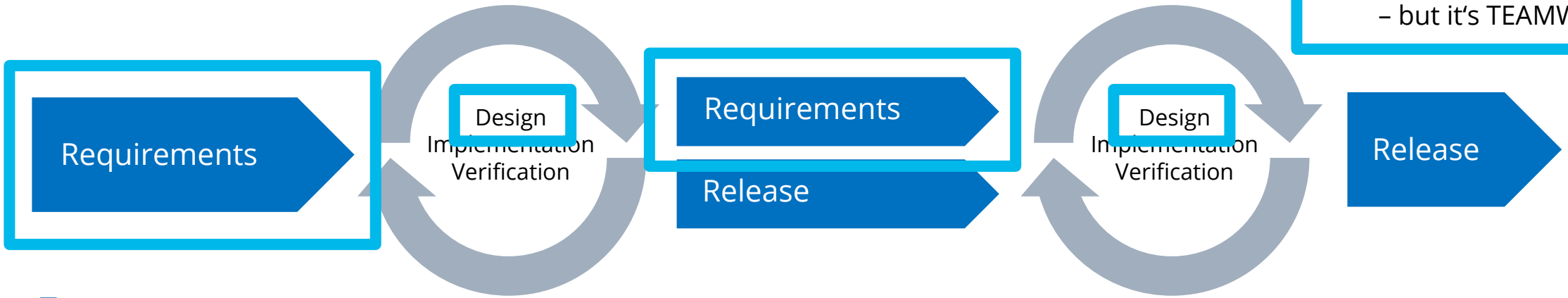
1. TRAINING	2. REQUIREMENTS	3. DESIGN	4. IMPLEMENTATION	5. VERIFICATION	6. RELEASE	7. RESPONSE
1. Core Security Training	2. Establish Security Requirements	5. Establish Design Requirements	8. Use Approved Tools	11. Perform Dynamic Analysis	14. Create an Incident Response Plan	Execute Incident Response Plan
	3. Create Quality Gates/Bug Bars	6. Perform Attack Surface Analysis/ Reduction	9. Deprecate Unsafe Functions	12. Perform Fuzz Testing	15. Conduct Final Security Review	
	4. Perform Security and Privacy Risk Assessments	7. Use Threat Modeling	10. Perform Static Analysis	13. Conduct Attack Surface Review	16. Certify Release and Archive	



How to Introduce Security into Agile Development Processes

1. TRAINING	2. REQUIREMENTS	3. DESIGN	4. IMPLEMENTATION	5. VERIFICATION	6. RELEASE	7. RESPONSE
1. Core Security Training	2. Establish Security Requirements	5. Establish Design Requirements	8. Use Approved Tools	11. Perform Dynamic Analysis	14. Create an Incident Response Plan	Execute Incident Response Plan
	3. Create Quality Gates/Bug Bars	6. Perform Attack Surface Analysis/ Reduction	9. Deprecate Unsafe Functions	12. Perform Fuzz Testing	15. Conduct Final Security Review	
	4. Perform Security and Privacy Risk Assessments	7. Use Threat Modeling	10. Perform Static Analysis	13. Conduct Attack Surface Review	16. Certify Release and Archive	

This is the major field of action – but it's TEAMWORK!



mgm

How to Introduce Security into Agile Development Processes

1. TRAINING	2. REQUIREMENTS	3. DESIGN	4. IMPLEMENTATION	5. VERIFICATION	6. RELEASE	7. RESPONSE
1. Core Security Training	2. Establish Security Requirements	5. Establish Design Requirements	8. Use Approved Tools	11. Perform Dynamic Analysis	4. Create an Incident Response Plan	Execute Incident Response Plan
	3. Create Quality Gates/Bug Bars	6. Perform Attack Surface Analysis/ Reduction	9. Deprecate Unsafe Functions	12. Perform Fuzz Testing	5. Conduct Final Security Review	
	4. Perform Security and Privacy Risk Assessments	7. Use Threat Modeling	10. Perform Static Analysis	13. Conduct Attack Surface Review	6. Certify Release and Archive	

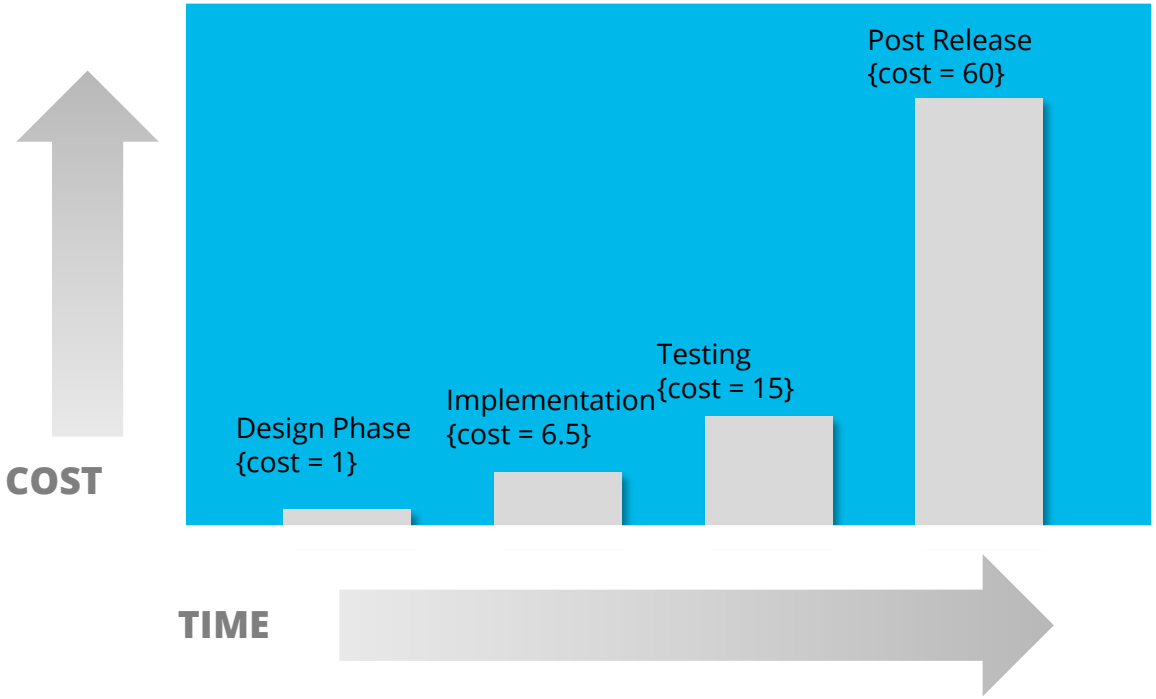
Utilize framework-level security functions + AUTOMATED testing!

Security by Design

Security-by-Design

Benefits

- vulnerability mostly introduced on implementation
- target: address vulnerability as early as possible
- reduce costs up to factor 60!

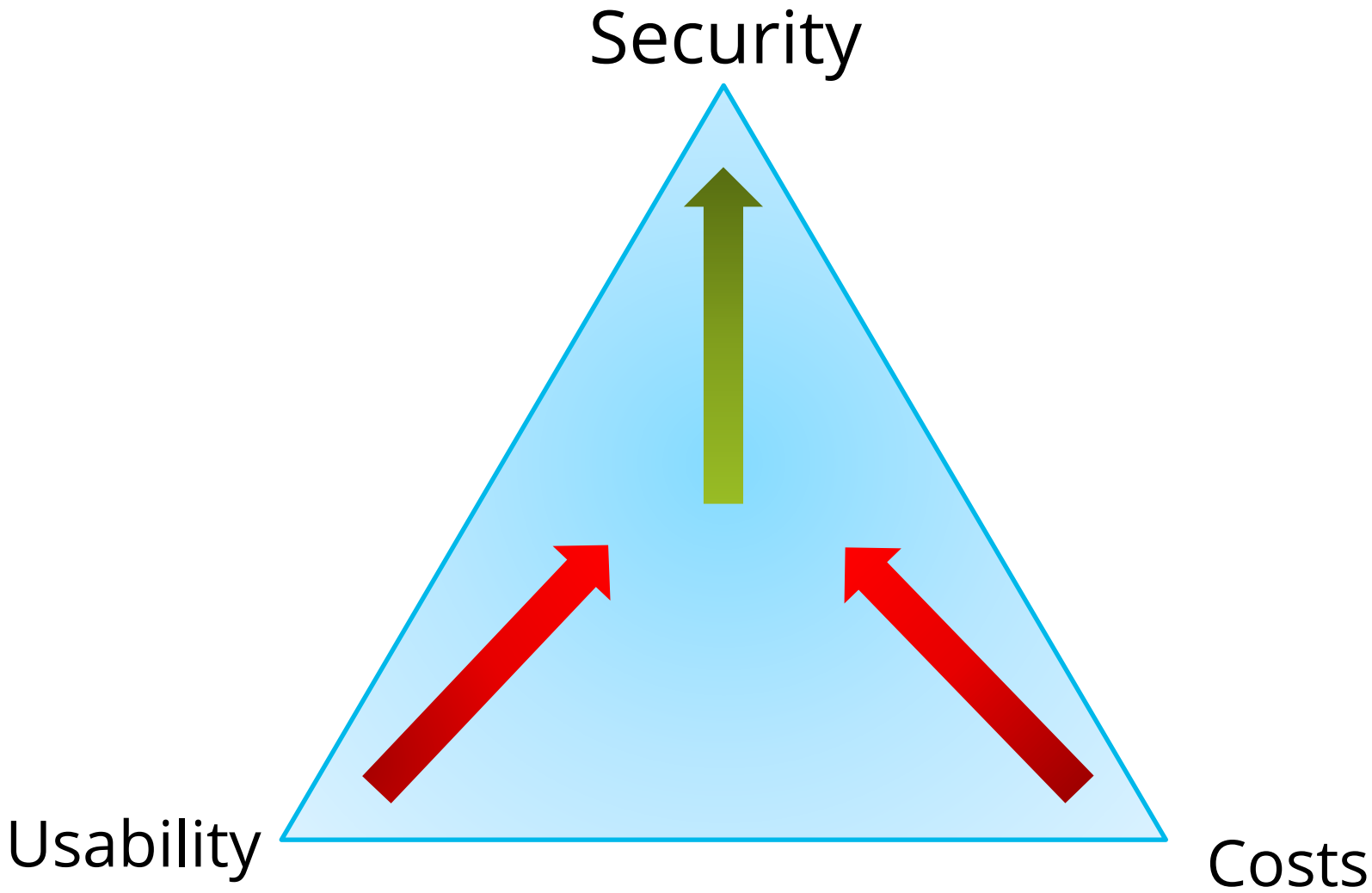


original design from: *Secure Coding Principles & Practices*

Finding the right Protection-Level

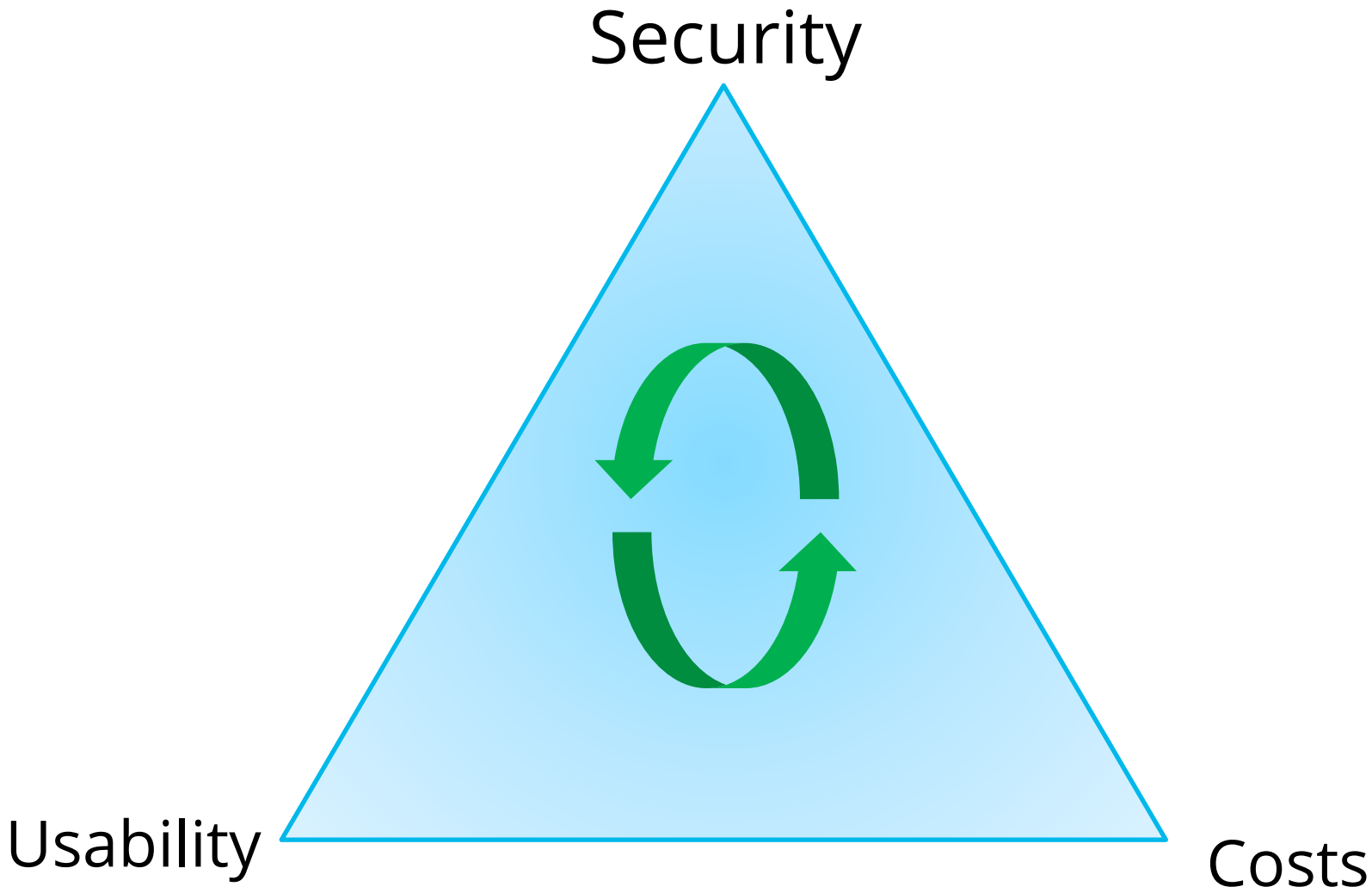
Security

...in conflict with other goals



Security

...in conflict with other goals



Security Principles

... it all depends on the protection requirements

- What data is stored and processed in our application?
  - Sensitive, business critical data?
  - Public data?
  - Personal data?
  - Payment Information?
- Who is using our application?
  - 5 or 5000 users?
  - Intranet or Internet?
- What is our worst case scenario?
- What is the expected impact due to loss of Confidentiality, Integrity or Availability?
- ...

Protection Goals

Primary Protection Goals

Confidentiality

Protection against unauthorized information gathering

Integrity

Protection against unauthorized data tampering

Availability

Protection against loss of functionality and operation capabilities

Secondary Protection Goals

aka derived Protection Goals

Protection of communication …

… contents		… circumstances
Confidentiality	→	**Anonymity** Confidentiality of the relation between subject and object
Integrity	→	**Accountability, Non-Repudiation / Liability, Authenticity** Integrity of the relation between subject and object
Availability	→	**(Reachability / Findability)** Availability of identifier / reference

Wolf & Pfitzmann, 2000; Parker, 1998

Protection Goals

What is the most important protection goal for the following asset?

A user password which is stored in a database.

A. Availability
B. Integrity
C. Confidentiality

Protection Goals

What is the most important protection goal for the following asset?

The user's access control tickets (like a signed: `is_admin:true`) which are stored inside a protected area inside the user's device (e.g. localStorage of the browser).

A. Availability
B. Integrity
C. Confidentiality

Protection Goals

What is the most important protection goal for the following asset?

The functionality of a webshop.

A. Availability
B. Integrity
C. Confidentiality

Protection Goals

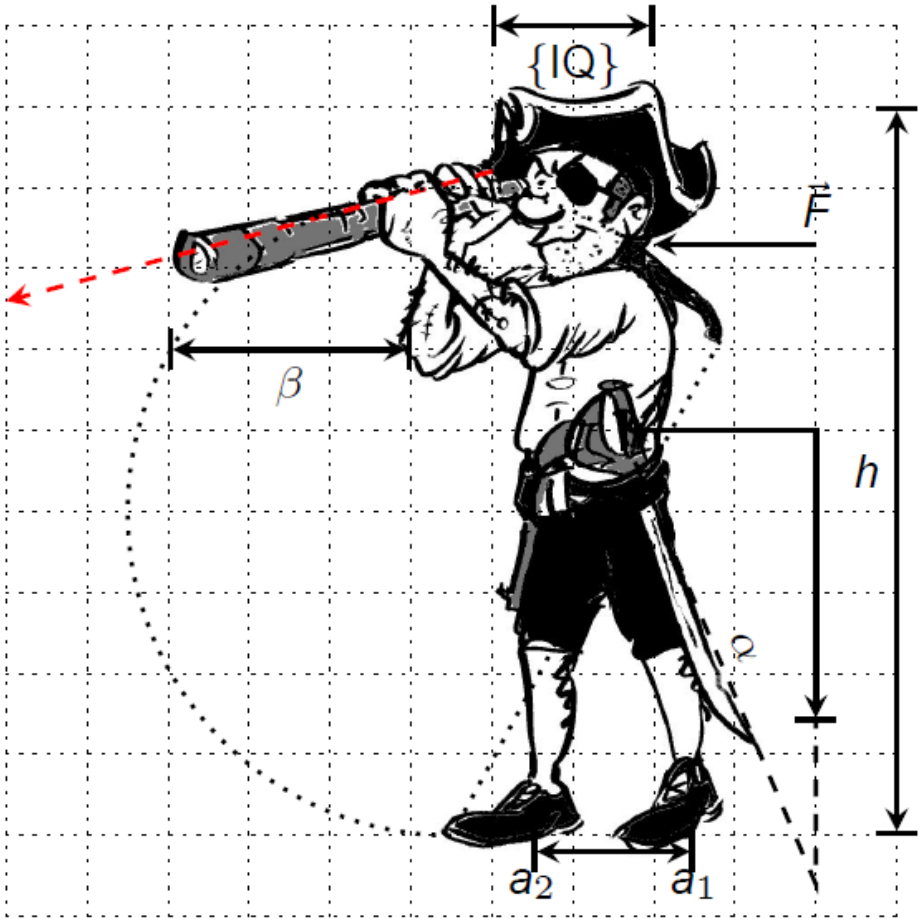
What is the most important protection goal for the following asset?

Health care data.

A. Availability
B. Integrity
C. Confidentiality

Attacker Model

Who is authorized?



Security considerations always include an **attacker model** and attacking techniques.

Possible attackers

- outsider
- system user
- administrator
- operations and maintenance
- producer / provider
- external employee
- temporary employee (e.g. student)
- terrorist
- flood / storm
- component becoming obsolete
- overvoltage / power outage
- ...

humans

elements, laws of nature

mgm

Attacker model

computing capacity

time

prior knowledge

distribution

passive

for each point of attack

active

financial power

Attacker types

attacker's strategy

opportunistic

- many easy targets
- looking for "weakest target"
- **standardization:** little effort per target

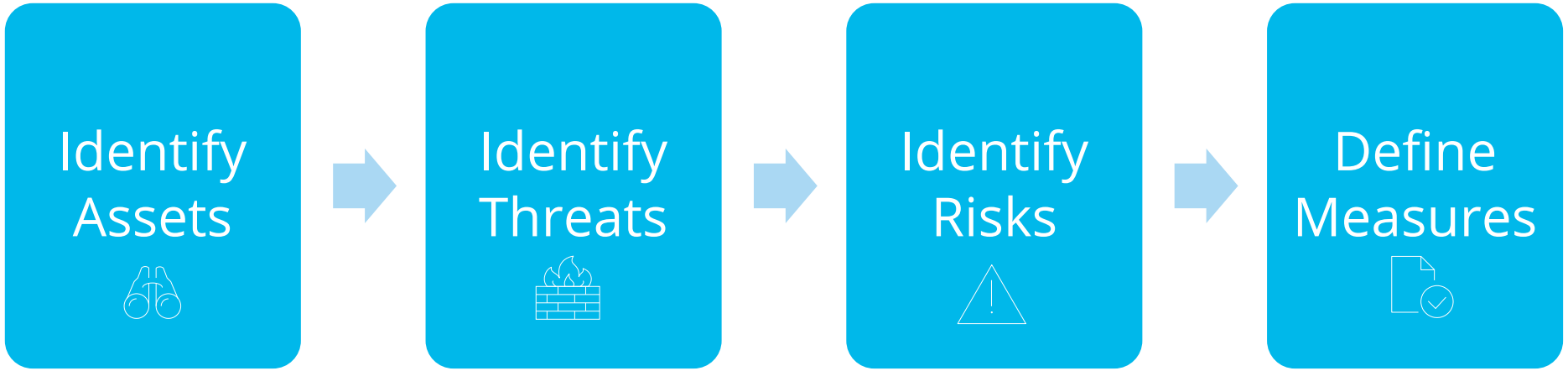
directed

- little profitable targets
- looking for "weakest link"
- **specialization:** much effort per target

Protection Need Analysis

Protection Need Analysis

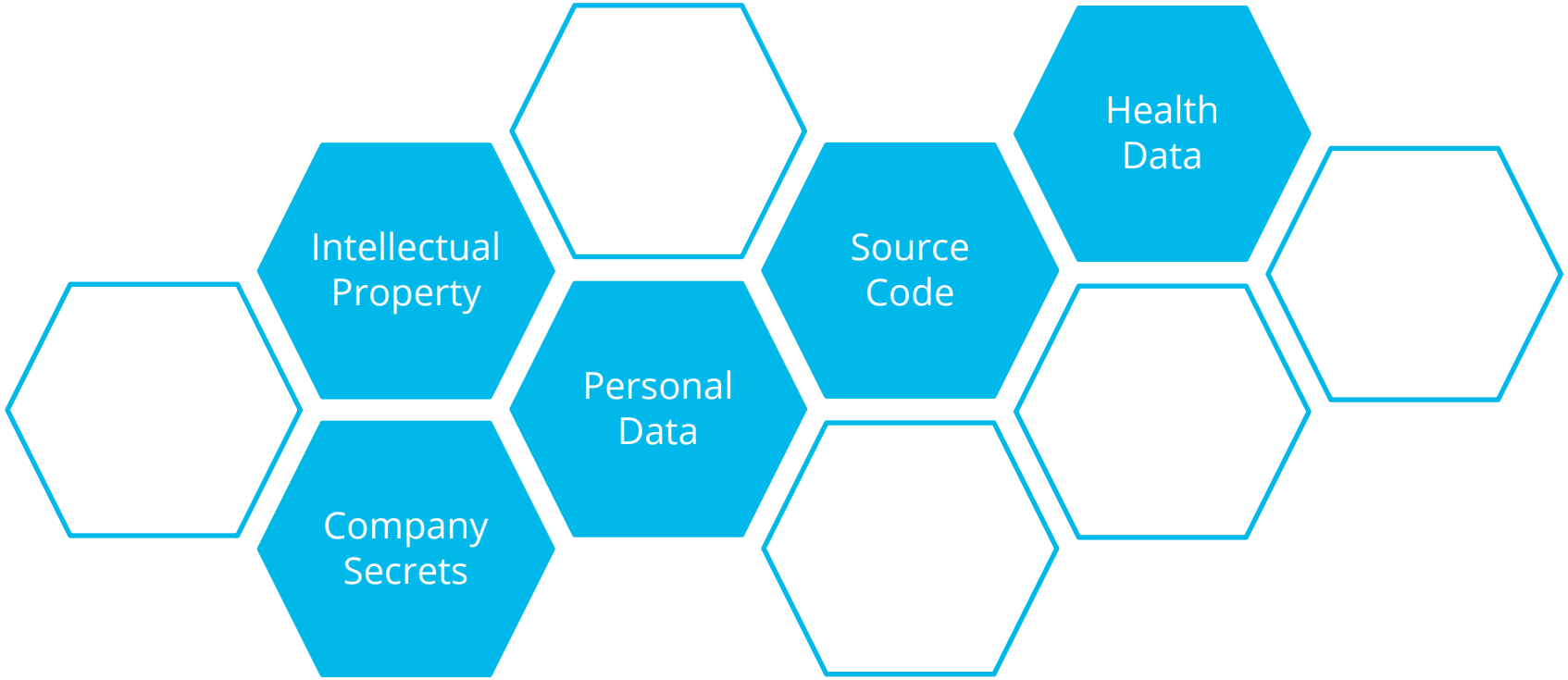
... process overview



Protection Need Analysis

... Identification of assets

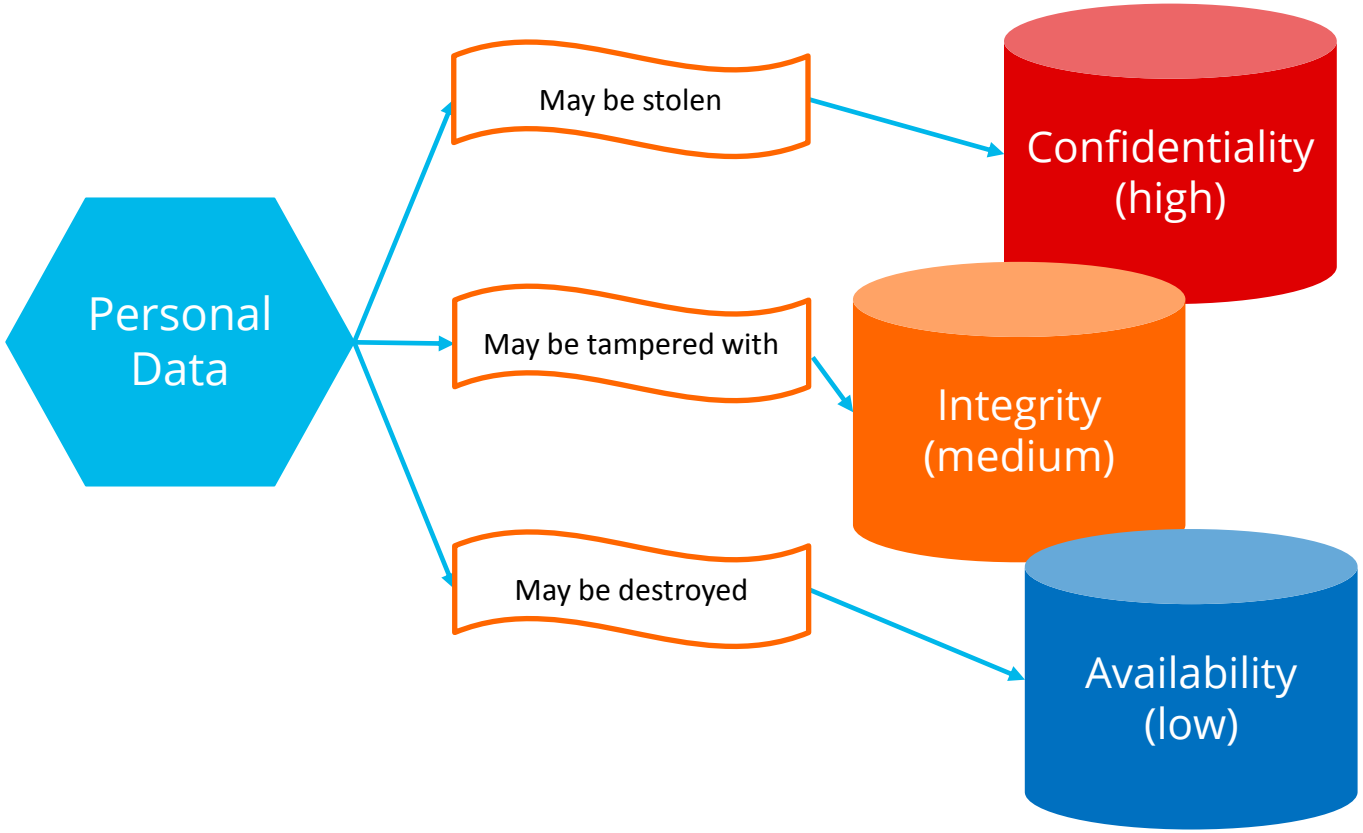
- What are the assets of the application that need to be protected?



Protection Need Analysis

... identification of threats

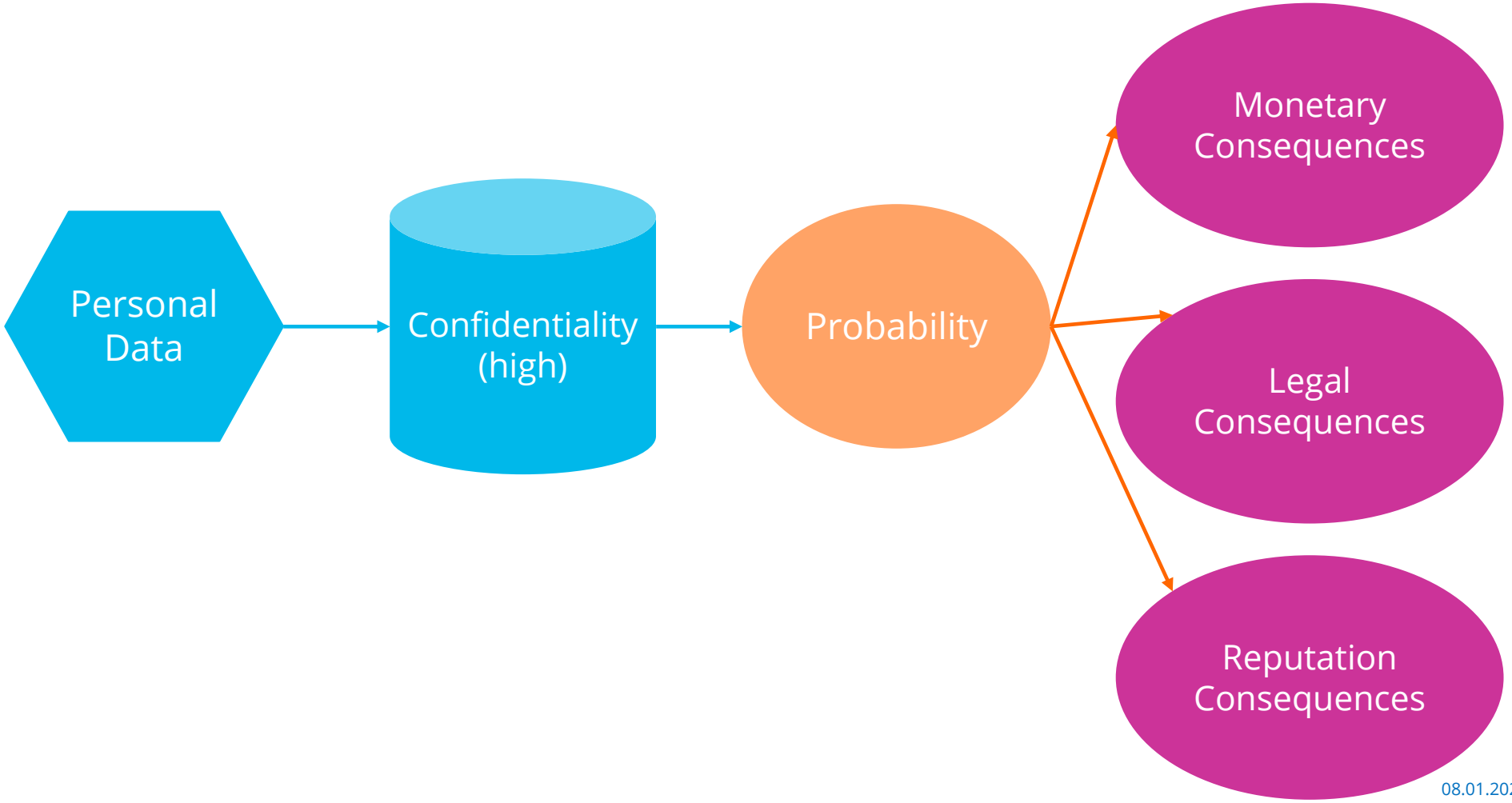
- What dangers are posed to the assets? How are they rated?



Protection Need Analysis

... Identification of risks

- What is the probability of a threat and what are the consequences of becoming “real”?



How is it done in practice?

- Usage of processes like threat modelling
- Example: STRIDE/DREAD by Microsoft
- Important: Models, not detailed techniques

Threat Modelling

STRIDE

		Threat:	Protection Goals
S	**poofing**	Pretence of wrong facts	Authenticity
T	**ampering**	Manipulation of information	Integrity
R	**epudiation**	Non-Tracability/ Deniability of actions	Non-Repudiation
I	**nformation Disclosure**	Disclosure of protected information	Confidentiality
D	**enial of Service**	Prevention of access	Availability
E	**levation of Privilege**	Gaining of elevated access	All

Threat Modelling with STRIDE

Process

1. Identification of assets
2. Analysis of each asset regarding the STRIDE threats
3. Introduction of individual measures to mitigate threat
4. Repetition of process until no further threats can be identified

Threat Modelling: Risk Assessment

DREAD

		Category:
D	**amage**	Impact of attack
R	**eproducibility**	Ease of repetition
E	**xploitability**	Effort needed for attack
A	**ffected Users**	Number of impacted people
D	**iscoverability**	Ease of threat recognition

Threat/Risk Modelling with DREAD

Process

1. Identification of threats
2. Analysis of each threat regarding the DREAD categories
3. For each category a number (rating) is assigned
4. E.g. the sum of ratings for each threat can be used for prioritizing

Threat/Risk Modelling with DREAD

Notes

Problem:

- Ratings for categories often not consistent
- Especially “Discoverability” subject of debate (Security by Obscurity?)

Important:

- Rating scale of categories must be individually determined for each project/company!

Security Principles

References

- OWASP ASVS
  - Different Levels depending on protection requirements
  - https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project

- BSI Grundschutz
  - Very detailed and complex guidelines to determine protection requirements and assets
  - https://www.bsi.bund.de/DE/Themen/ITGrundschutz/itgrundschutz_node.html

External Showcase

Protection Need/Risk Analysis (STRIDE/DREAD)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
1									"Mapping" to Protection Goals																		
2									Authenticity	Integrity	Non Repudiation	Confidentiality	Availability	Accountability													
3									STRIDE							0 (irrelevant) 1 (low) 2 (medium) 3											
4																											
5																											
6																											
7																											
8	Nr.	Assets	Risks	Threat	Vulnerability	Already existing Measure	Note	internal Reference	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege	Privacy	Damage Potential	Reproducibility	Exploitability	Affected users	Discoverability	Reputational damage	TOTAL Risk	Risk Owner	Tracking	Type of Handling	Verification	(Planned) Measures
11				Finding out passwords to dedicated / many usernames	Enumeration of passwords possible	Application "slows down" response time for too many login attempts with wrong password	Application allows trivial Password	JIRA #34	x							2	3	2	3	2	1	13			Take Action		Anti-automation
12				Users tend to use simple passwords	Application does not support here, or has too simple policy				x							2	2	1	3	3	1	12			Take Action	ASVS: 2.25, 2.27	Establishment of a strong password policy and ban on trivial passwords
13	2	Session / Session-ID																									
14				Stolen Session-ID is used over a long time	Errors in the session timeout configuration or can exist as many active sessions per user				x	x						2	1	1	3	1	1	9			Take Action	ASVS: 3.3, 3.4, 3.5, 3.6, 3.7, 3.12, 3.16, 3.17, 3.18, 4.14	Introduce session timeout on inactivity, but also on activity. Limit the number of simultaneous active sessions per user
15				Calculate valid session Ids	No usage of common mechanisms for SessionID generation				x	x						2	1	1	2	2	0	8			Take Action	ASVS: 3.11, 7.6, 7.7, 7.15	Use a container or other common library to create SessionIDs
16																											

OpenSAMM

OWASP SAMM

What is it?

https://owaspsamm.org

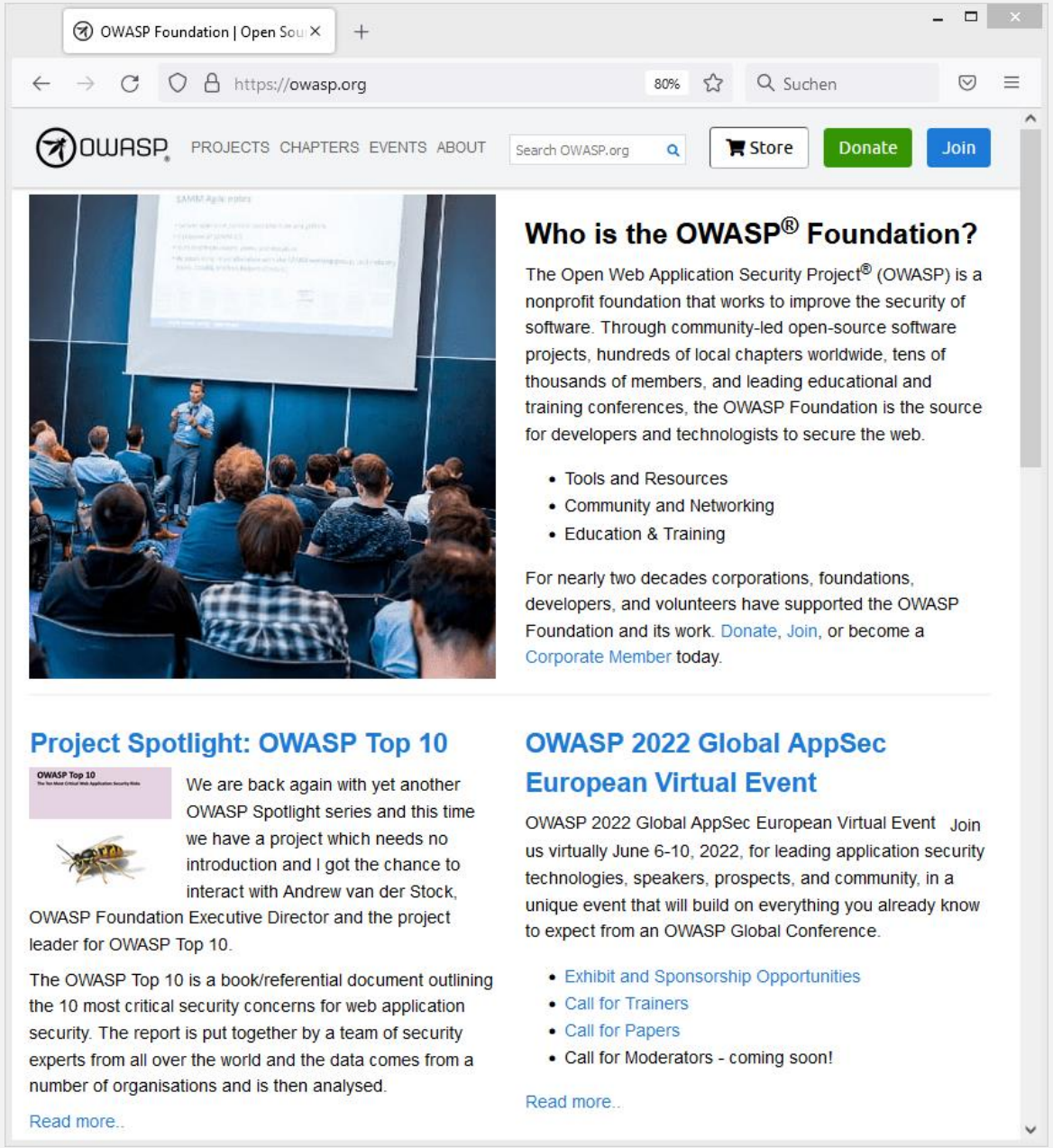
- an open framework which is simple to use
  - multiple Tool-Support (Excel, Website etc.) exists
- powered (mostly) by a security affine **community**
- defined with „what security-experts think is important to be part of an SSI“ (Software Security Initiative)
- main-characteristics:
  - measurable through defined maturity-levels and by calculating a metric
  - actionable through clear pathways for improvements
  - versatile through its agnostic definitions

Development

OWASP

Open **W**eb **A**pplication **S**ecurity **P**roject - **www.owasp.org**

- Chapters
  - Local Discussions (aka Stammtisch)
  - Mailinglists
- Events
  - Conferences (AppSec EU, Global AppSec)
- Projects
  - Top 10
  - Application Security Verification Standard (ASVS)
  - Cheat Sheet Series
  - Security Testing Guides
  - Software Assurance Maturity Model (SAMM)
  - Zed Attack Proxy (ZAP)
  - Dependency Track
  - Vulnerable Web Applications Directory / Juice Shop
  - ModSecurity Core Rule Set



mgm

OWASP

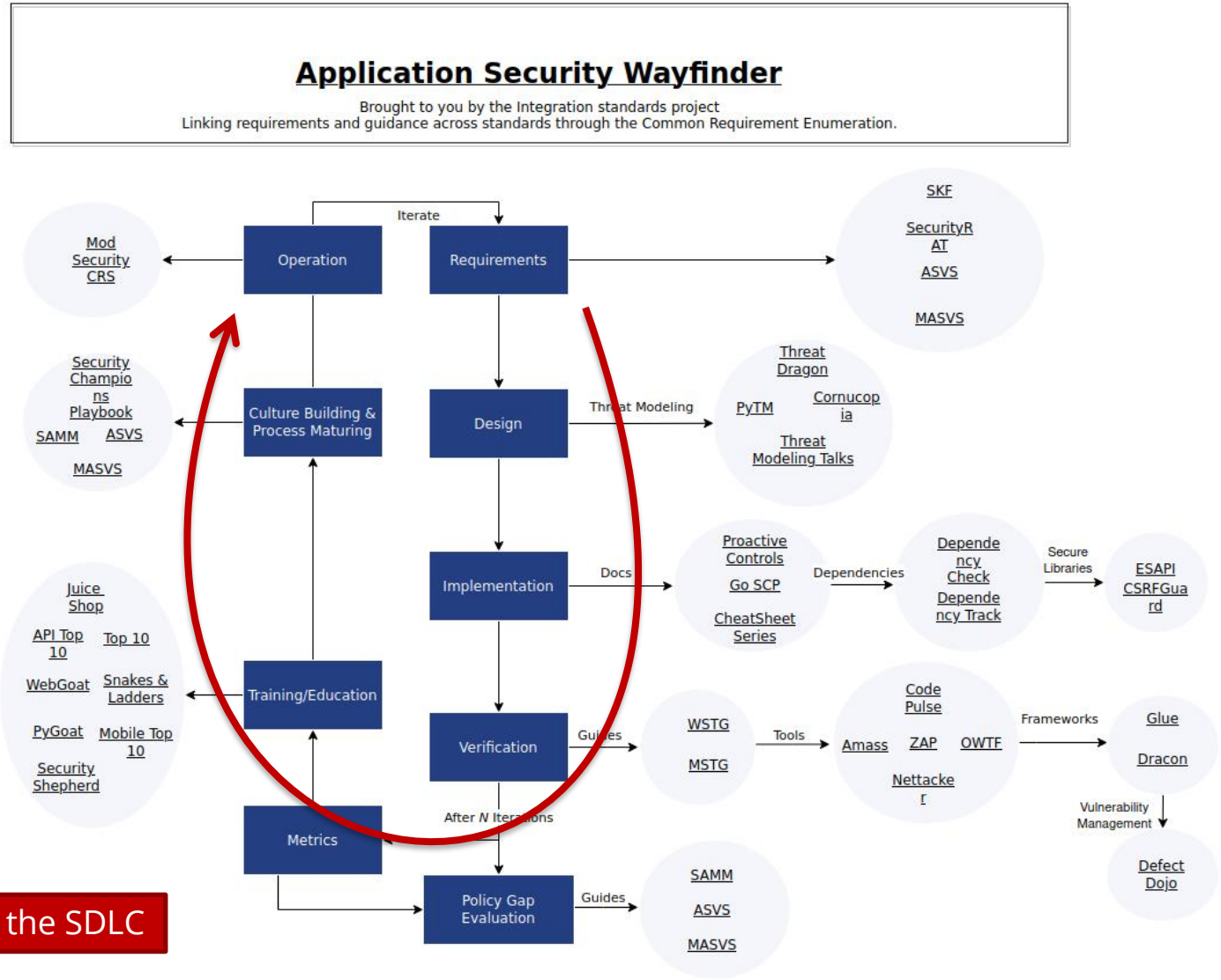
Project - Wayfinder

https://owasp.org/projects/

Suchen

OWASP Projects, the SDLC, and the Security Wayfinder

Thanks to the OWASP Integration Standards Project for mapping OWASP projects in a diagram of the Software Development LifeCycle. This resource should help you determine which projects fit into your SDLC.



Sorted along the SDLC

mgm

Principles for Secure Programming

- **Always treat user input as possibly malicious**
  - …because the user is free to enter whatever he wants
    - Mistakenly wrong input with an impact on security
    - Targeted exploitation of vulnerabilities (user = attacker)
  - “All input is evil until proven otherwise“, Source: Microsoft “Writing Secure Code”

Principles for Secure Programming

Defensive Programming

- **Always expect the worst case … and prepare for that**
- **Example:**

```
int risky_programming(char *input){
  char str[1000+1];     // one more for the null character
  strcpy(str, input);   // copy input
}
```

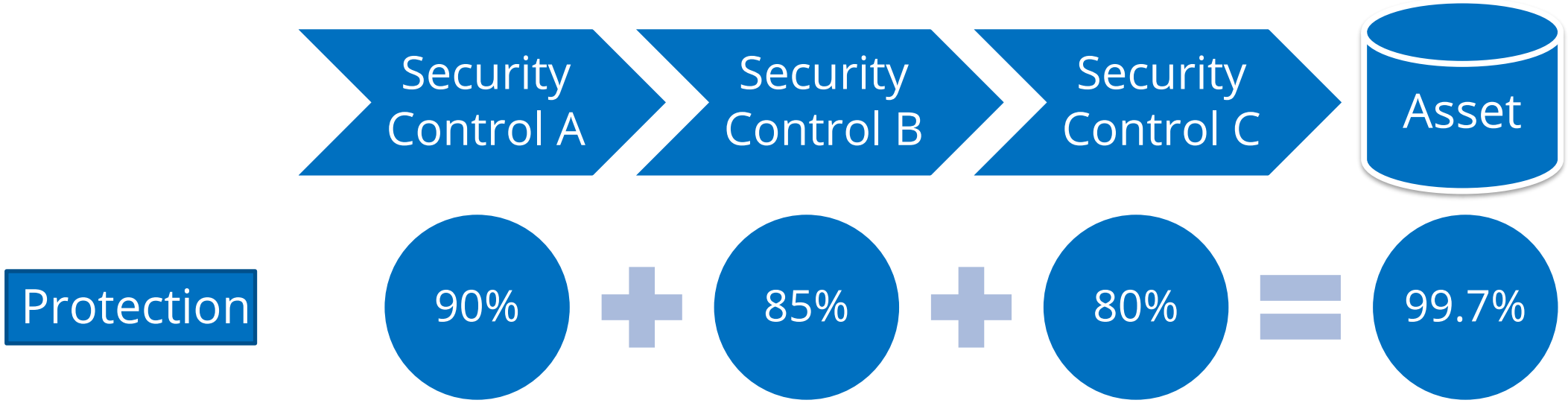
Buffer overflow if more than 1000 characters are submitted!

Defensive Programming avoids this problem:

```
int secure_programming(char *input){
  char str[1000];
  strncpy(str, input, sizeof(str));  // copy input without exceeding the length of the destination
  str[sizeof(str) - 1] = '\0';  // if strlen(input) == sizeof(str) then strncpy won't NUL terminate
}
```

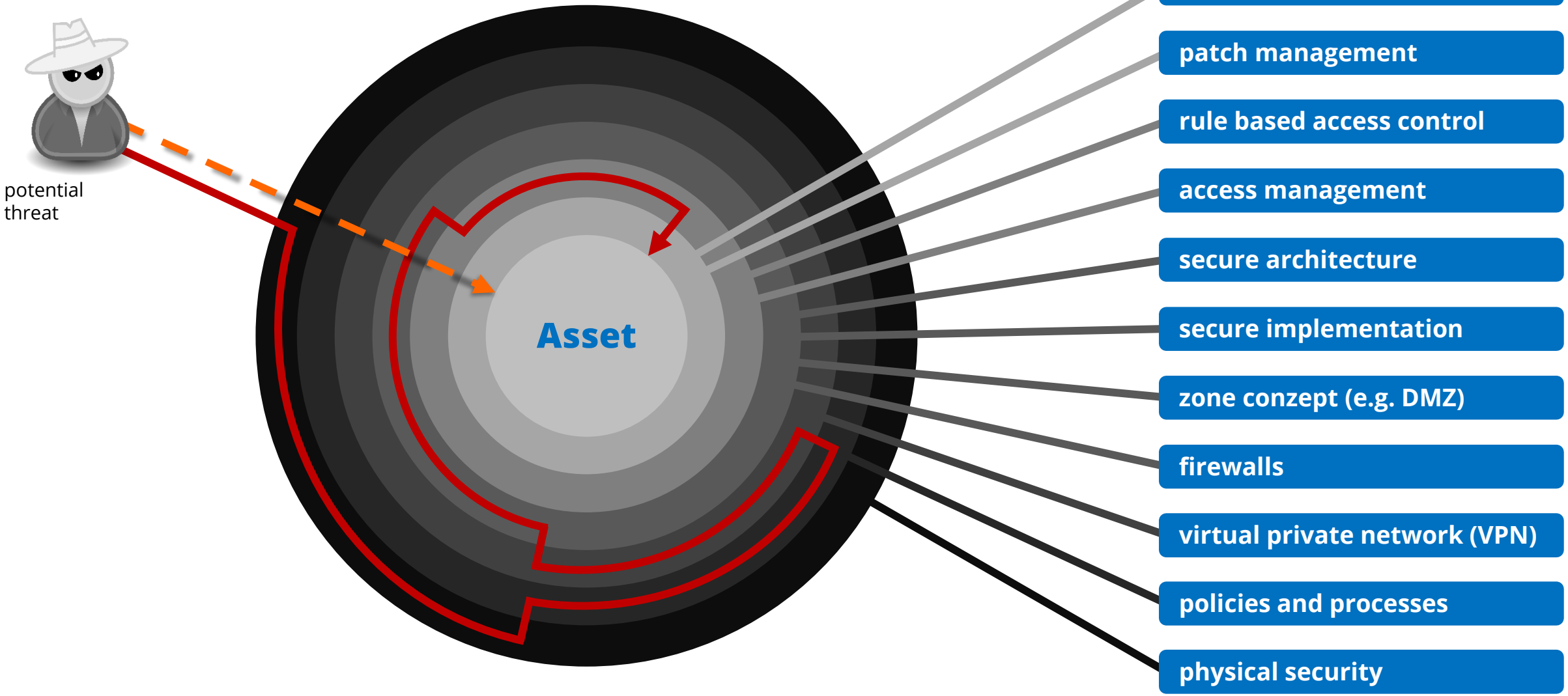
Principles for Secure Programming

- **Defense in Depth**
  - Limit negative impact by multiple security controls: When the first control is overcome, the second may help stopping further damage.
  - Example: Improper data validation (1st control) enables SQL injection. Access control on database level (2nd control) may limit the impact to the respective table.



Defense in Depth

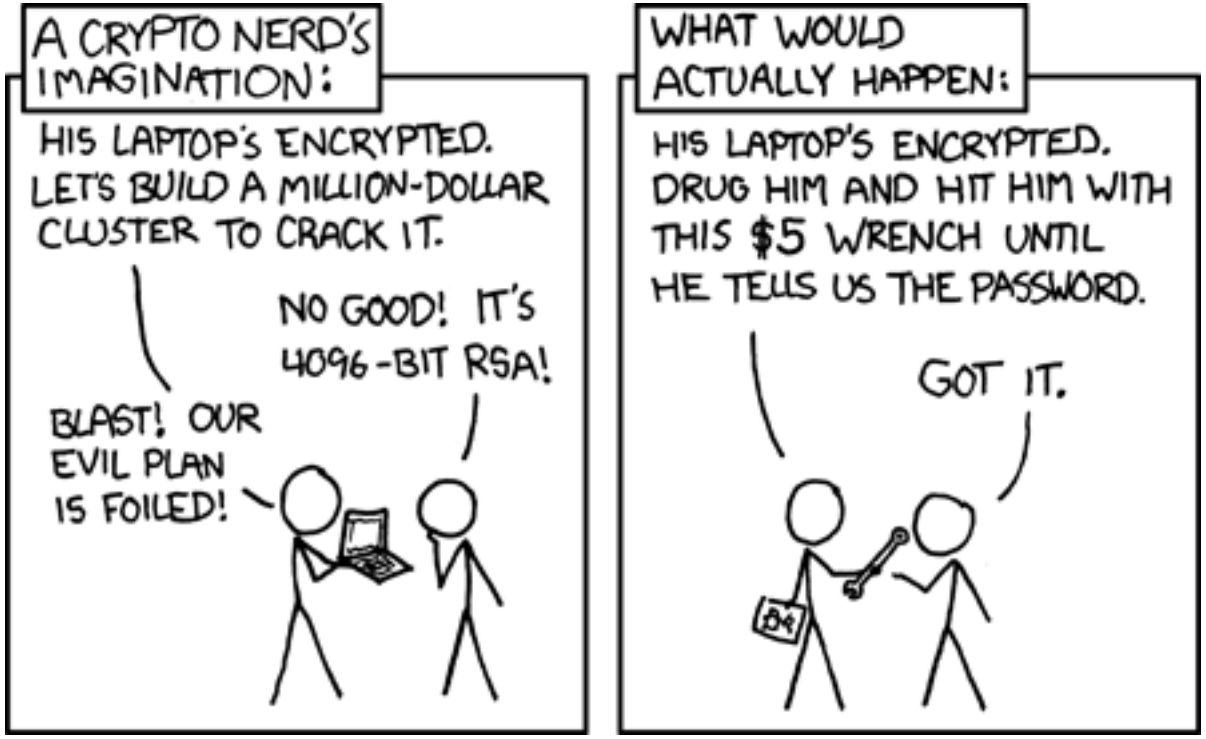
Example



Principles for Secure Programming

Weakest-Link-Principle

- The security of the overall system depends on the security of the weakest link.
- All elements of the system must be put on the intended security level



https://xkcd.com/538/

Principles for Secure Programming

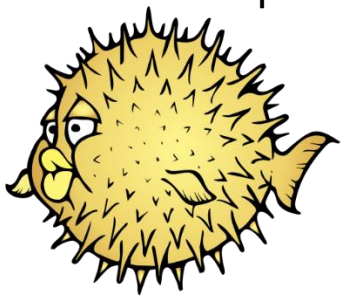
- **Layer of Indirection**
  - Access parameters do not use global internal IDs but mapped single-purpose IDs. A mapping function translates one to the other.
  - This way, the manipulation of access parameters only affects the intended scope and thus always references allowed resources.

  Example:
    - External index instead of internal primary key (for access to database entries)
    - Alias instead of real file name (for access to previously uploaded files)

Principles for Secure Programming

- **Fail Securely**
  - Faults must not lead to uncontrollable states. This usually leads to data loss or leakage.
  - Special attention required for error handling!

- **Secure Defaults**
  - After installation, the system has the most secure configuration
  - Security comes first, the range of functions after
  - The aware user may then change the secure configuration. A section on “Security“ in the manual explains respective risks.



OpenBSD

Only two remote holes in the default install, in a heck of a long time!

Principles for Secure Programming

- **No Security by Obscurity**
  - Security can not be achieved by “hiding“.
  - A thorough measure withstands an attack even if the stash is uncovered.
  - Nevertheless: need-to-know principle

JOURNAL

DES

SCIENCES MILITAIRES.

Janvier 1883.

LA CRYPTOGRAPHIE MILITAIRE.

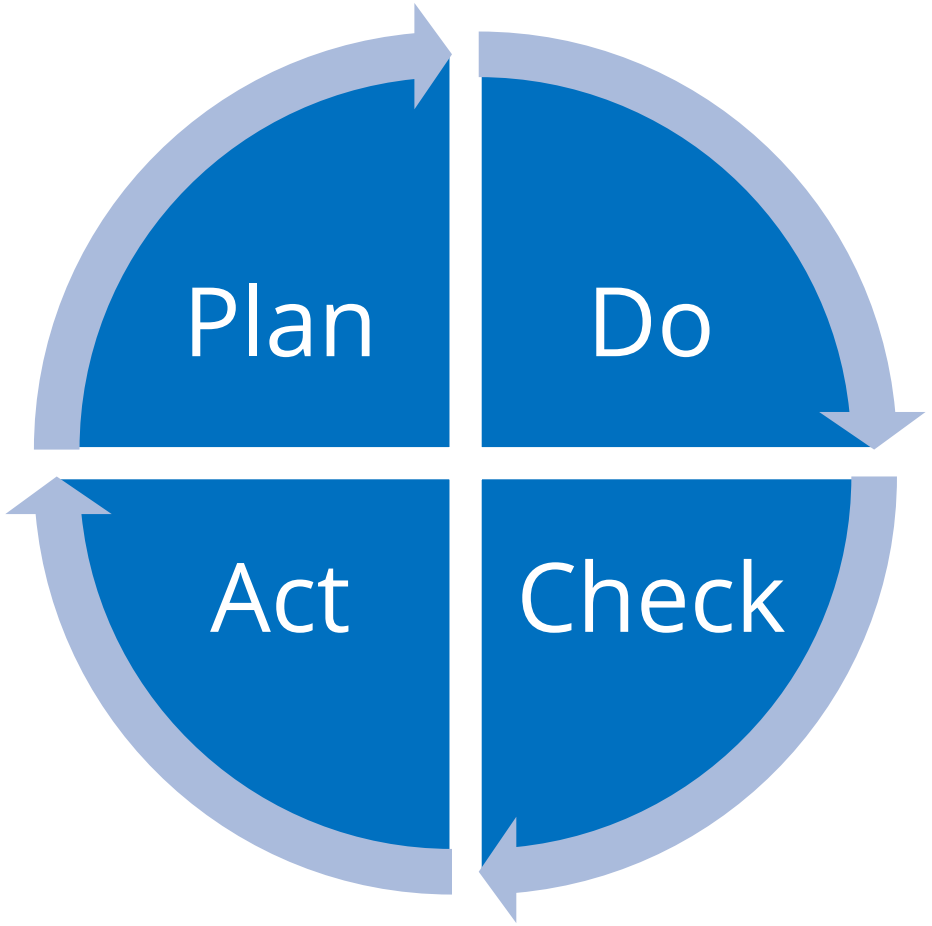
2° Il faut qu’il n’exige pas le secret, et qu’il puisse sans inconvénient tomber entre les mains de l’ennemi ;

“It must not require secrecy, and must be able to fall into the hands of the enemy without difficulty.”
– Auguste Kerckhoffs, «La cryptographie militaire», Journal des sciences militaires, vol. IX, pp. 5–38, Janvier 1883

- **Need-to-know principle**
  - The less an attacker can find the less is the risk!

Principles for Secure Programming

- **Security as a Process**
  - What do security and order have in common?
  - Right: Order is no state but a process!



Deming Cycle

Principles for Secure Programming

- **Least Privilege**
  - An access concept ensures that each process has minimal access rights with respect to the intended operations
  - Example:
    - The normal user having only read access has no option to call the edit-function
    - Moreover, the application implements access control that checks for each call of the edit function whether the calling user has respective rights
  - This way, negative impact is prohibited in the case that measure 1 is implemented falsely and the user manages to call the edit function

- **Separation of duties**
  - If one thing is compromised, only a part fails
  - Example: DOTADIW – “Do One Thing And Do It Well”

Principles for Secure Programming

- **Simplicity**
  - KISS: Keep It Simple and Secure!
  - Complexity and security are mutually exclusive – or require unjustifiable efforts to combine!
  - The times of feature-rich products are over, users seek simple solutions (cf. the iPhone's success story)

Technology Layer
Well-Structured Application!

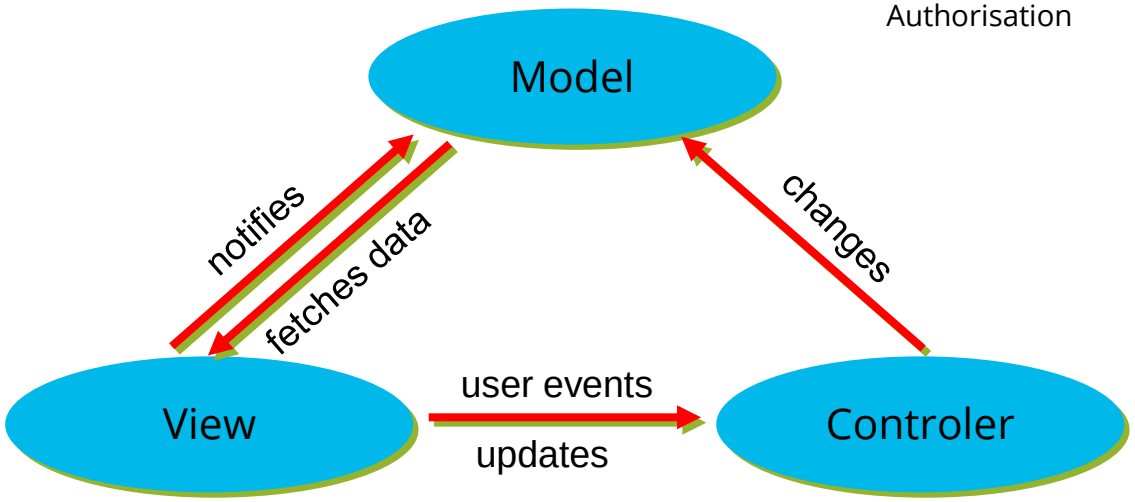
Prerequisite for Countermeasures:

- MVC (Model View Controller) Pattern
  - central Controller Servlet
  - Action Layer
  - Business Logic in the Model
  - DB Access in separate Layer in the Model
  - View Component just for „Rendering“ (no Logic)
- Support by Frameworks
  - Struts
  - Turbine
  - Tapestry
  - …
- Drupal: Presentation–Abstraction–Control (PAC) pattern

Model-View-Controler (MVC)

- The MVC Pattern supports Encapsulation of Security Features
- Java
  - Struts
  - Turbine
  - Tapestry
- PHP
  - CakePHP
  - Symfony
  - CodeIgniter
- .NET
  - ASP.NET MVC oder
  - CastleProject

Data Binding
Validation
Prepared Statements / ORM
Authorisation



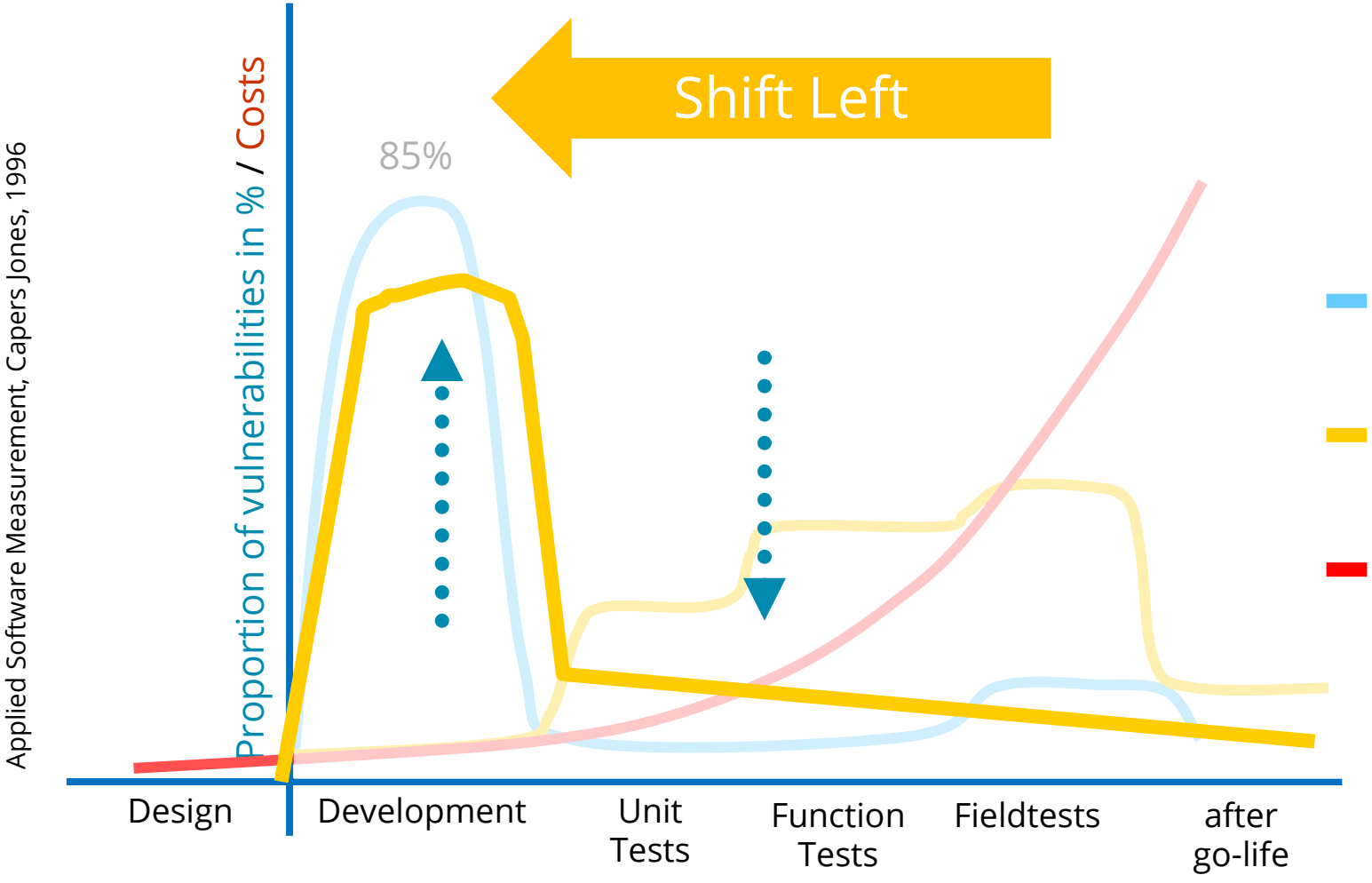
Output Validation

Single Access Point
Central Error Management
Validators (Input Validators)
Authentication

Security Review / Tests

The right moment - shift left

Shift Left



- % of defects <u>introduced</u> in that phase
- % of defects <u>found</u> in that phase
- $ of remediation of a defect <u>revealed</u> in that phase

IBM System Sciences Institute Survey

Unit Cost to fix a software security hole in the deployment phase of an SDLC is 100 x that of fixing the same hole at the design stage

Exponential cost the later you catch (leave) it!

Penetration Tester != Attacker

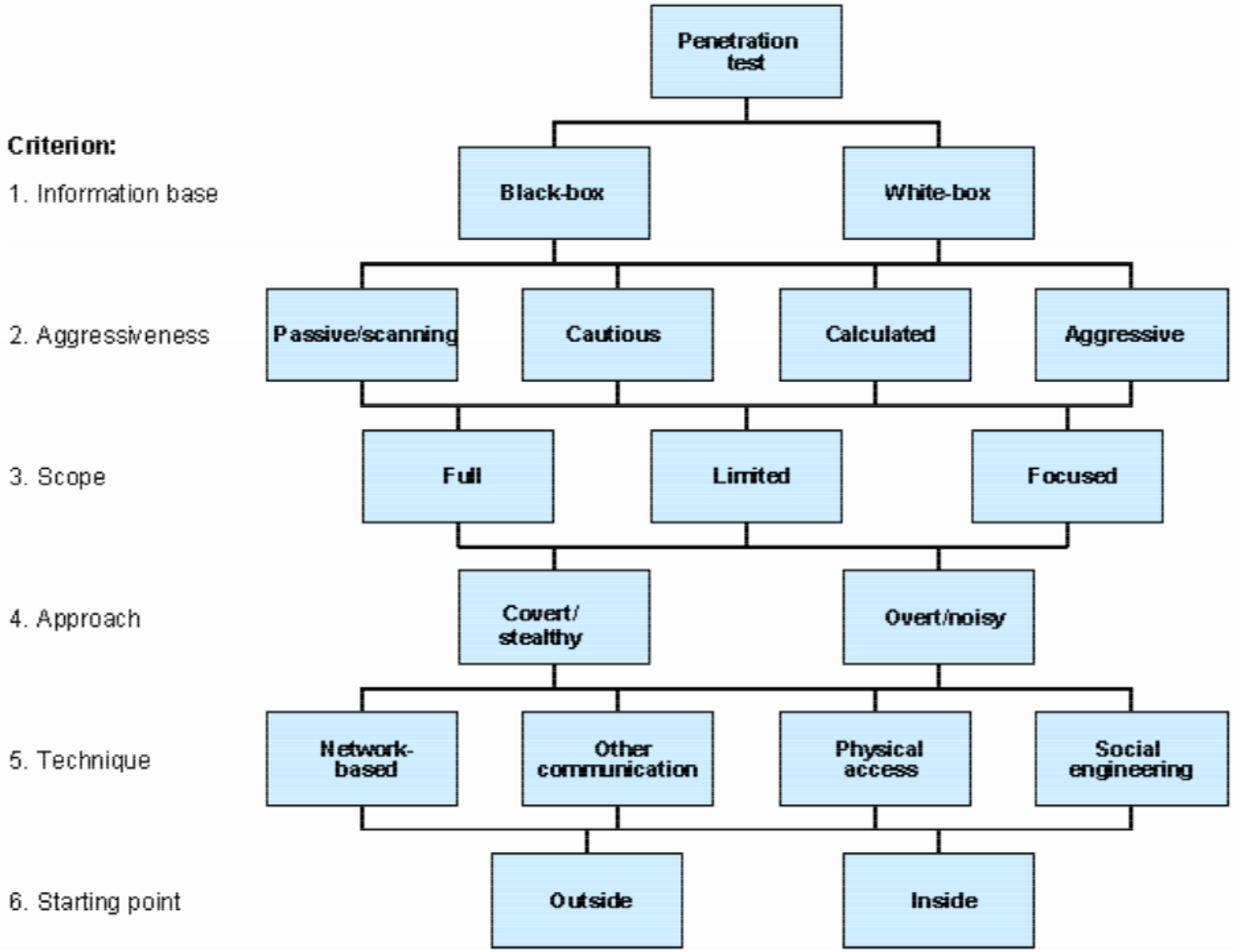
- Think of a construction supervisor:
  - Compares the construction to a list of well-known mistakes, done by building workers
  - Will not figure out the type of construction material
  - Will review the statics and building plans
  - Will ask the workers how they constructed if unclear
  - Will not proof that the building really collapses

Attacker vs. Penetration tester

- Search for the needle in the haystack
- Budget = asset_value – worktime_value
- Looks for 1 full exploitation path
  → “Expert in exploitation”

- Search for the needle in the haystack
- Budget ≪ asset_value
  - (development, operation, revenue, …)
- Looks for all possible problems
  → “Expert in finding possible bugs”

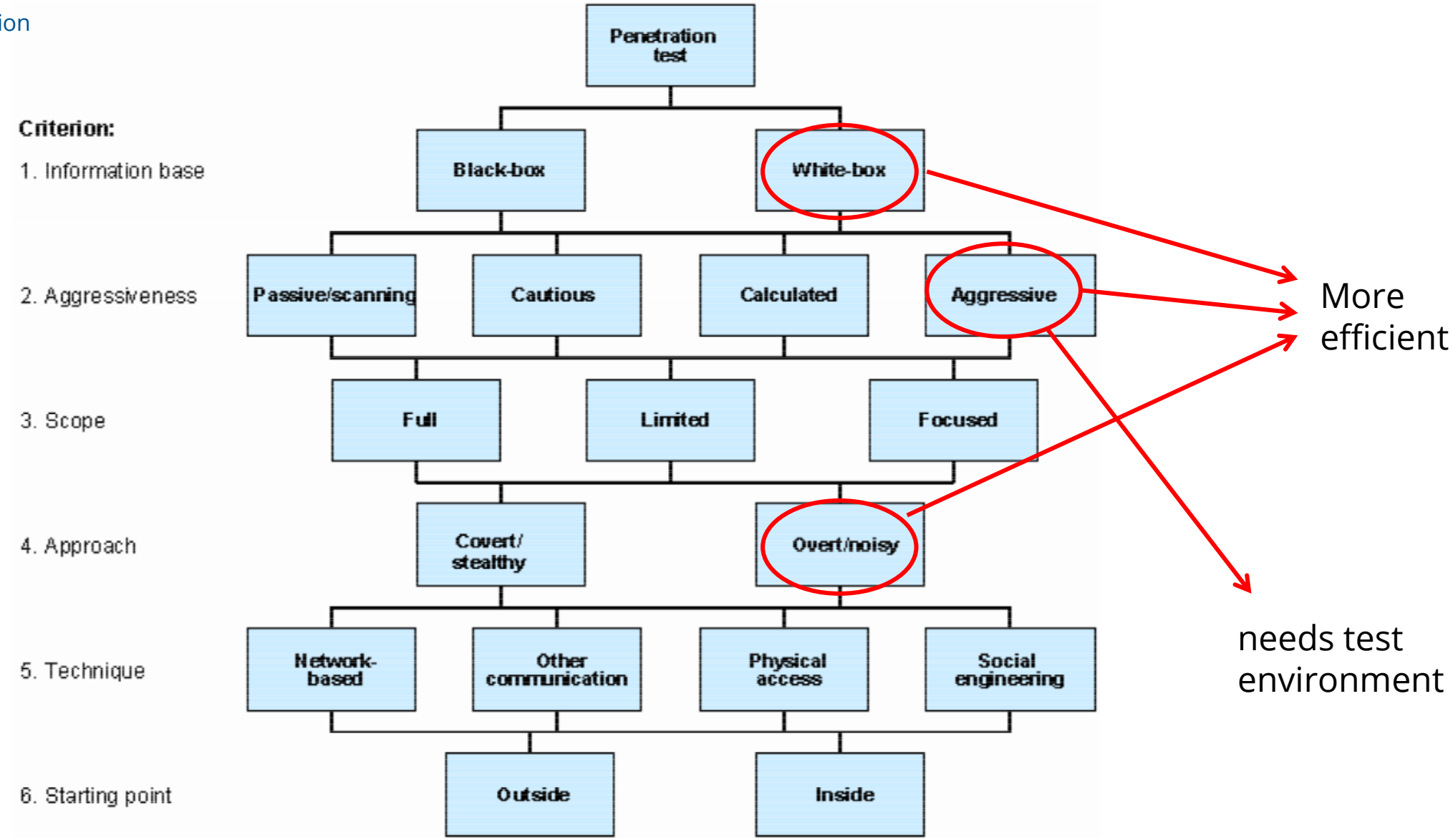
Classification of penetration tests (BSI)



Source: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration_pdf.pdf?__blob=publicationFile&v=1

Classification of penetration tests (BSI)

Recommendation



Source: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration_pdf.pdf?__blob=publicationFile&v=1

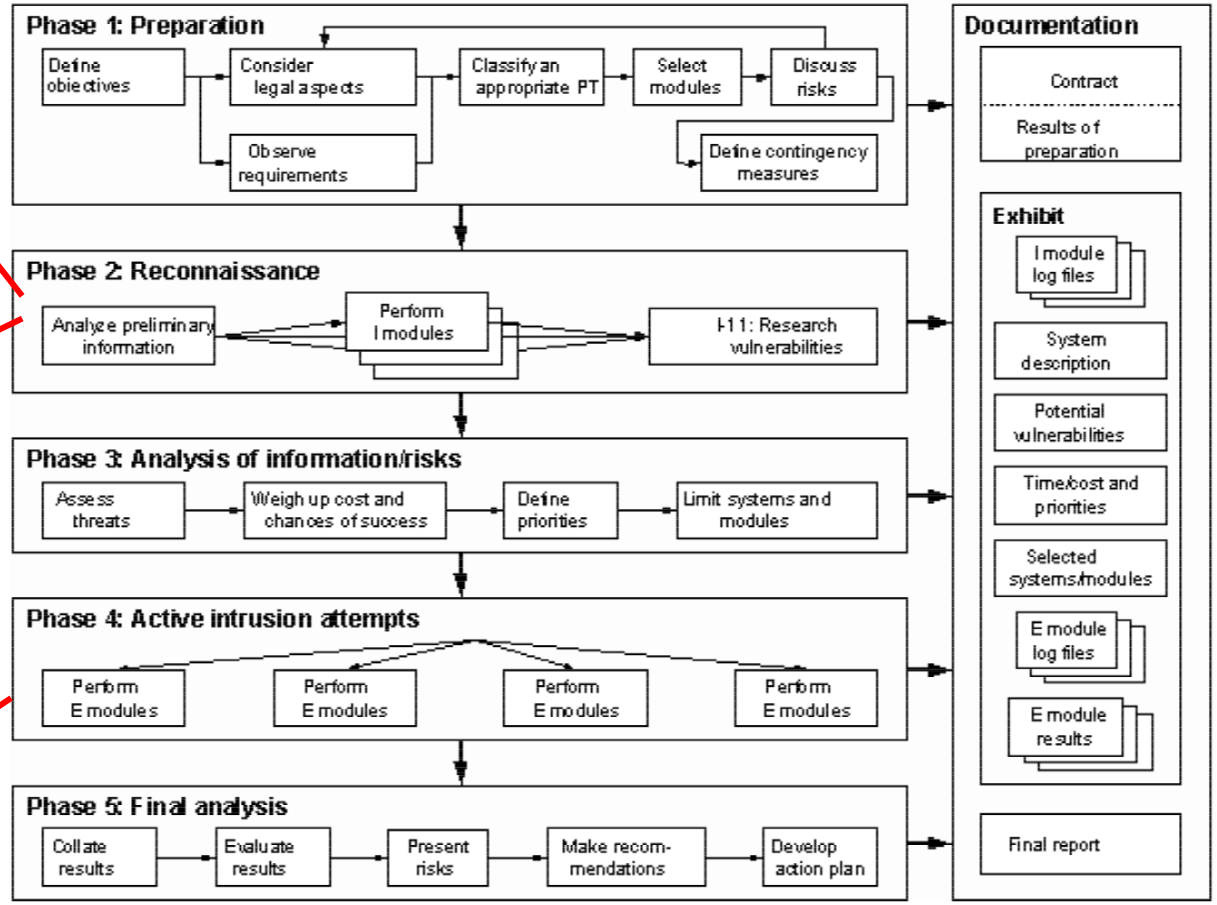
Phases of a penetration test (BSI)

Access via network (aka penetration test)

1. automated (Nessus, nmap, Metasploit)
2. Manual verification (send requests manually, analyse browser/client-traffic, burp, soapui, wireshark, tcpdump,...)

Access via OS (aka host-audit)

1. automated (Nessus+ssh/smb)
2. manual verification (ssh + netstat, rdp+powershell, ...)

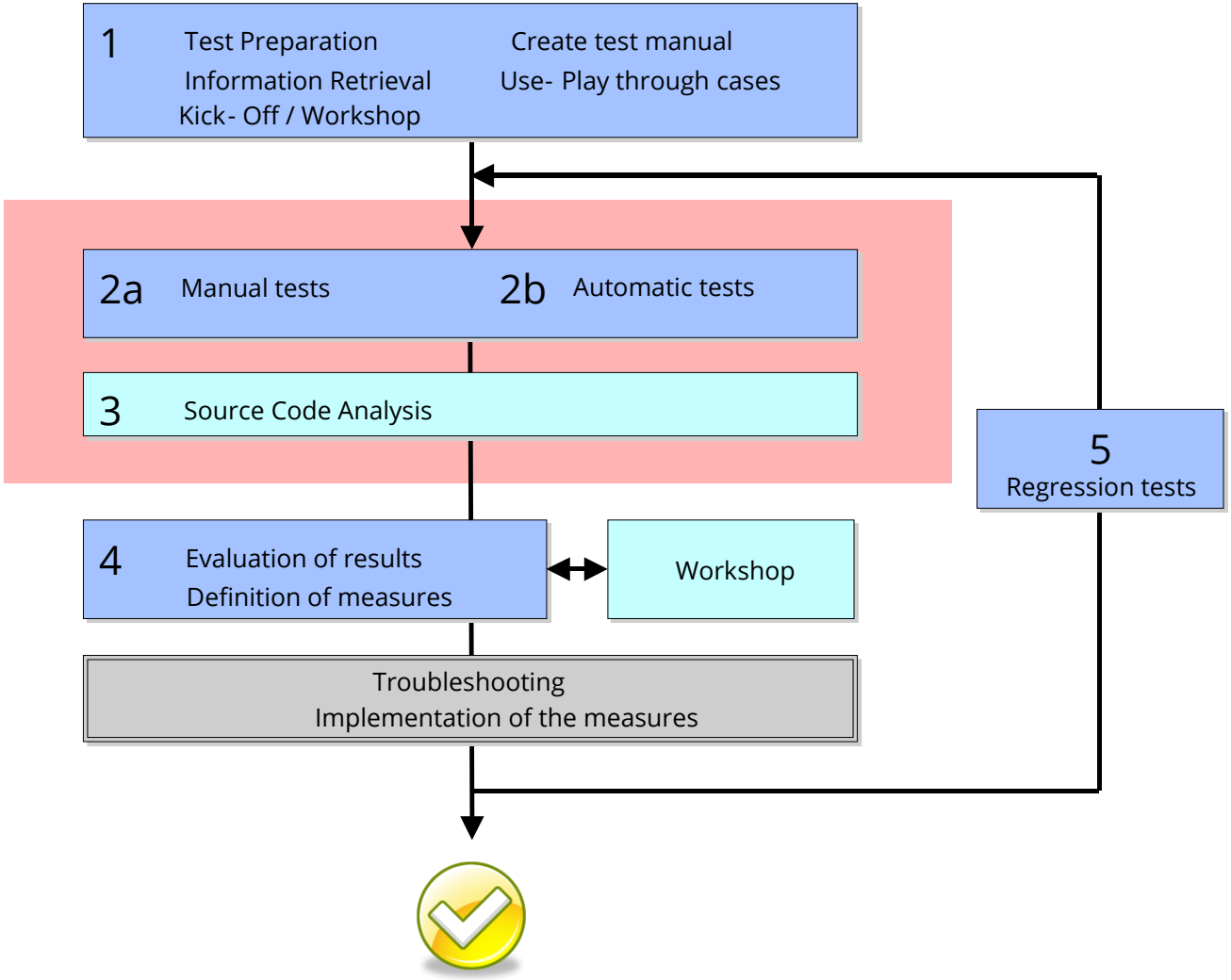


Warning: expensive!

Source: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration_pdf.pdf?__blob=publicationFile&v=1

Web Application Security Pentest

Idealised process



Web Application Security Pentest

Criteria

- Black Box vs. Whitebox
  - Blackbox test sometimes resembles searching for a needle in a haystack
  - The pentester is usually inferior to the attacker
    - Attacker: there are no limits to the effort and intensity // proportional to motivation and interests
    - Pentester tests alone // Number of attackers can be unlimited
  - The more comprehensive the information, the higher the quality of the result
  - ➔ **Give the tester a maximum of information!**
    - User IDs for all roles (also test special admin access)
    - Architectural image/description / Technical/IV concept/specifications
    - Description Use-Cases
    - Interface descriptions
    - User manual / Operating instructions
    - QA Protocols / Load Test Results
    - Results of Risk Analysis / Threat Model

Web Application Security Pentest

Criteria

- Test with or without WAF?
  - The root of the problem is to be found in the application.

  Pentest with WAF only makes sense in justified cases.
- Test environment Production environment
  - Test environment: data loss is not to be feared // Consideration of productive operation not necessary
  - Production environment: All real influences are included in the test.

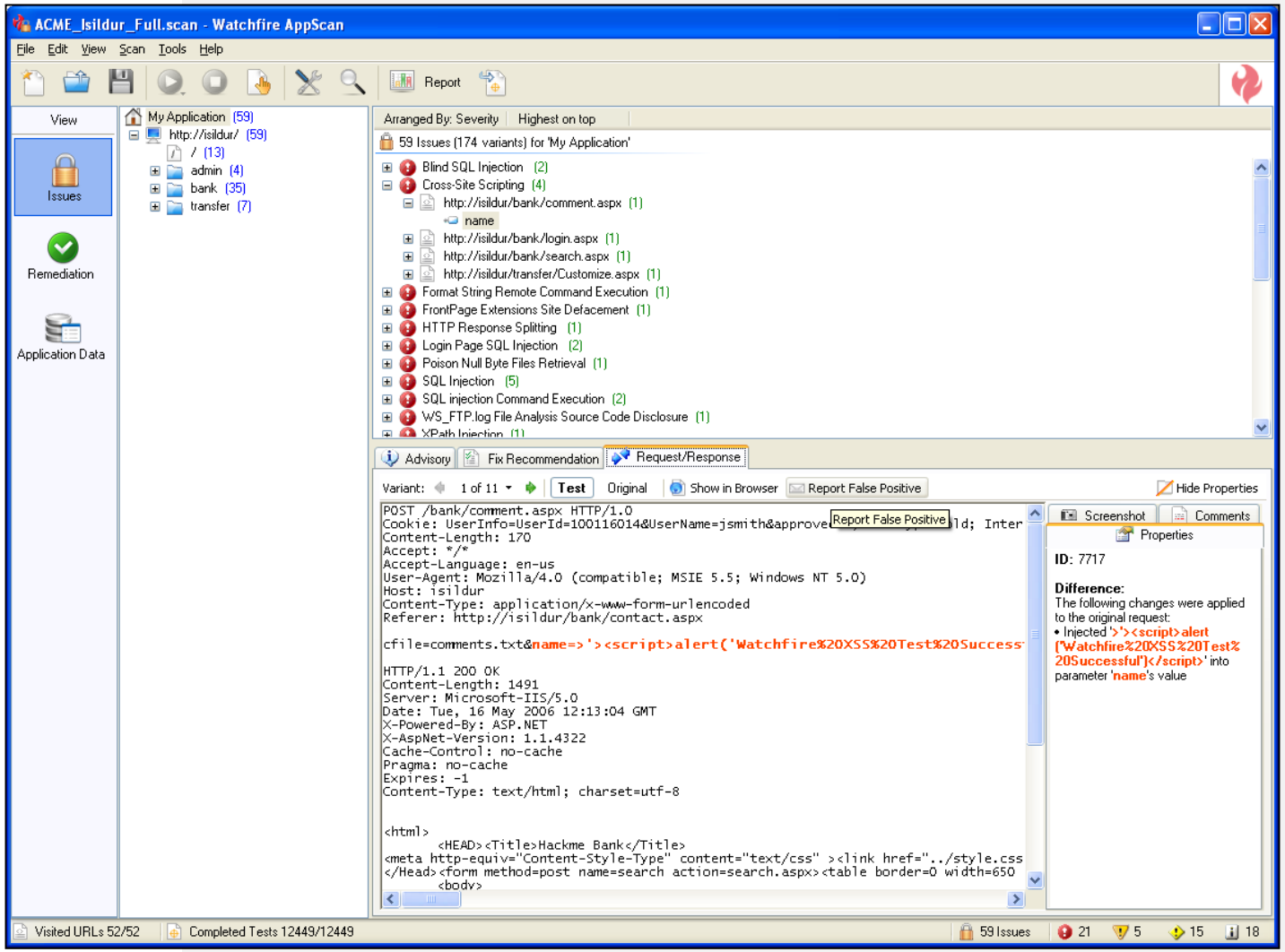
  ➔ **As a rule, a test in a test environment (as exact a copy of the productive environment as possible!) is preferable.**
- Test the internal functions?
  - Many web applications have functional areas (workflows, admin) that are not accessible on the Internet.

  These should be included in the tests, as there are many possibilities of attack.

WHAT Scanner

Using the example of AppScan

- Market leader in application scanners
- Version 7.5 with extension concept
- Enterprise version with management functionalities
- Regression tests
- Integration with Fortify



mgm

Web Application Scanner

- A good overview:
  - WAVSEP: The Web Application Vulnerability Scanner Evaluation Project http://code.google.com/p/wavsep/
  - sectooladdict.blogspot.com/2012/08/commercial-web-application-scanner.html
- Free versions of commercial scanners
  - Acunetix
  - N-Stalker
  - Netsparker
  - Sandcat Mini

5 Layer Model

Layers

	Level	Content	Examples
5	**Semantics**	Preventing Fraud	Phishing Protection Information Disclosure
4	**Logic**	Securing Workflows and Processes	“Forgot Password” Func. User Lock-out
3	**Implementation**	Avoiding Implementation Faults leading to Vulnerabilities	Cross-site Scripting SQL Injection
2	**Technology**	Principles of Secure Coding	Encryption Authentication
1	**System**	Securing the Software used on the System / Platform	Known Vulnerabilities Configuration Issues
0	**Network & Host**	Securing the Host and Network	

5 Layer Model

Tool-Support

	Level	Skills	Tool support
5	**Semantics**	Corporate Identity and business-communication	. . . . .
4	**Logic**	Securing Workflows and Processes	O . . . . .
3	**Implementation**	Avoiding Implementation Faults leading to Vulnerabilities	OOO . . . . . (PenTest) OOOOOO . . (SCA)
2	**Technology**	Principles of Secure Coding	OO . . . . .
1	**System**	Network- and system administration	OOOOOOO . .
0	**Network & Host**		

Types of Web Applications

- A - Customized Web Applications
  - The owner has full access to the implementation
  - He – or the service provider – can patch identified vulnerabilities himself.

Level 2 - 5

- B - Standard Web Applications and Components
  - Owner can not patch.
  - Instead: application of security patches
  - Usually, such software can be configured – which has a security impact

Level 1

- C - Mixes: Customized Web Applications utilizing “half- finished Products“
  - Problems from A and B accumulate
  - No clear distinction of patching responsibilities
  - Example: hybris (Level 1) + Frontend + Customization

Level 1 - 5

Sourcecode Analysis

Code-Review vs. SAST

	Code-Review (without Tool-Support)	SAST (with Tool-Support)
When?	• on (every) Pull Request / Commit • at milestones / gateways • on a component/module basis	• regularly (instantly, every commit, daily ...) • integrated into CI/CD (may also break builds) • at milestones / gateways
Drawbacks	• time / pricing • reliability / human failures (resulting in False-Positives / -Negatives) • strongly depends on reviewer (subjective) • doesn't scale well (done mostly with narrow focus)	• pricing • False-Positives (findings that are no vulnerability) • False-Negatives (misses an vulnerability) • understanding the technology stack • only finds what is described by a formal rule
Advantages	• thoroughly reviewing neuralgic areas and processes • understanding the technology-stack • finding logical / semantical flaws	• (re-)inspecting the <u>whole code base</u> reliably and objective for at least "low hanging fruits" (depends on scan-types) • instantly providing results (e.g. to IDE) • checking policies, guidelines, compliances (if expressable as rule) • outputs metrics / KPIs (Key Performance Indicators) and development over time

A tool-guided manual analysis grants higher benefits!

Comparison

Pentest vs. Source Code Analysis

Penetration Testing	SAST
Belated Securiy (Search for the needle in the haystack)	Inherent Security (Systematic, comprehensive approach)
Findings must be “translated” into the “language“ of the developer	Findings are already in the “language” of the developer
Hard to measure coverage	In theory full coverage (in terms of they “see it” – not necessarily “understand it” → high false-positive rate)
Late in the SDLC	Accompanies the SDLC from the very beginning
Application must normally be in release status (or at least deployable).	Component/module tests possible
Low contribution to the training of the developer	Strong retroactive effect on the cause and the process
Gives statements for the system as a whole (incl. Webserver, Infrastructure etc.)	Statements on application only
Easy feasibility independent of the used technologies (Java, PHP, JSF…)	Sourcecode must be available and technologies must be understood by the scanner.
Typically cheaper in terms of money	Typically more expensive if commercial tools are used
Some vulnerabilities can easily be identified some not.	Some vulnerabilities can easily be identified some not.

Trying to verify Security…

Your boss wants to spend $20k to verify if the security of your work is acceptable. Which measure sounds the best?

You set-up a MS-Exchange server including Outlook Web Access (OWA). You got 4 different quotations from 4 companies.

A. Company **A** wants to do a Web Application Penetration test searching for the OWASP Top 10 like SQL injections and insecure deserialization attacks.
B. Company **B** recommends to review the network architecture and perform a network penetration test looking for system configuration issues and known vulnerabilities
C. Company **C** wants to do a source code analysis (SAST) of the whole application to review in-depth variable flows.
D. Company **D** recommends to do a white-box review of the operating system configuration of the servers in the DMZ using rdp/ssh to look see the registry/config files.

Trying to verify Security…

Your boss wants to spend $20k to verify if the security of your work is acceptable. Which measure sounds the best?

You lead a group of developers who wrote an internet-accessible web application which should be sold and installed at a customers premises. You got 4 different quotations from 4 companies.

A. Company **A** wants to do a Web Application Penetration test searching for the OWASP Top 10 like SQL injections and insecure deserialization attacks.
B. Company **B** recommends to review the network architecture and perform a network penetration test looking for system configuration issues and known vulnerabilities
C. Company **C** wants to do a source code analysis (SAST) of the whole application to review in-depth variable flows.
D. Company **D** recommends to do a white-box review of the operating system configuration of the servers in the DMZ using rdp/ssh to look see the registry/config files.

Trying to verify Security…

You want to assure that your user's passwords are stored securely. Which approach will give you an answer with a high probability?

A. You order a penetration test with your provider.
B. You raise a bug bounty program to make sure that vulnerabilities are reported to you.
C. You buy a dynamic application security testing (DAST) tool to scan your application.
D. You buy a static application security testing (SAST) tool to scan your code.
E. You employ a security expert to review your code and DBMS setup.

Trying to verify Security…

You want to assure that all data transmitted by the user is well encrypted while being sent through the internet. Which approach will give you high certainty?

A. You order a penetration test with your provider.
B. You raise a bug bounty program to make sure that vulnerabilities are reported to you.
C. You buy a dynamic application security testing (DAST) tool to scan your application.
D. You buy a static application security testing (SAST) tool to scan your code.
E. You employ a security expert to review your code and DBMS setup.

Trying to verify Security…

The penetration testers you employed for a five-day test ask you for project documentation and test users for each role. What do you do best?

A. You provide them everything they need to ensure the best results.
B. You deny the request because real attackers have no docs either.
C. You grant them a standard user account but no details.

Deployment …
→ Security on System & Infrastructure Level