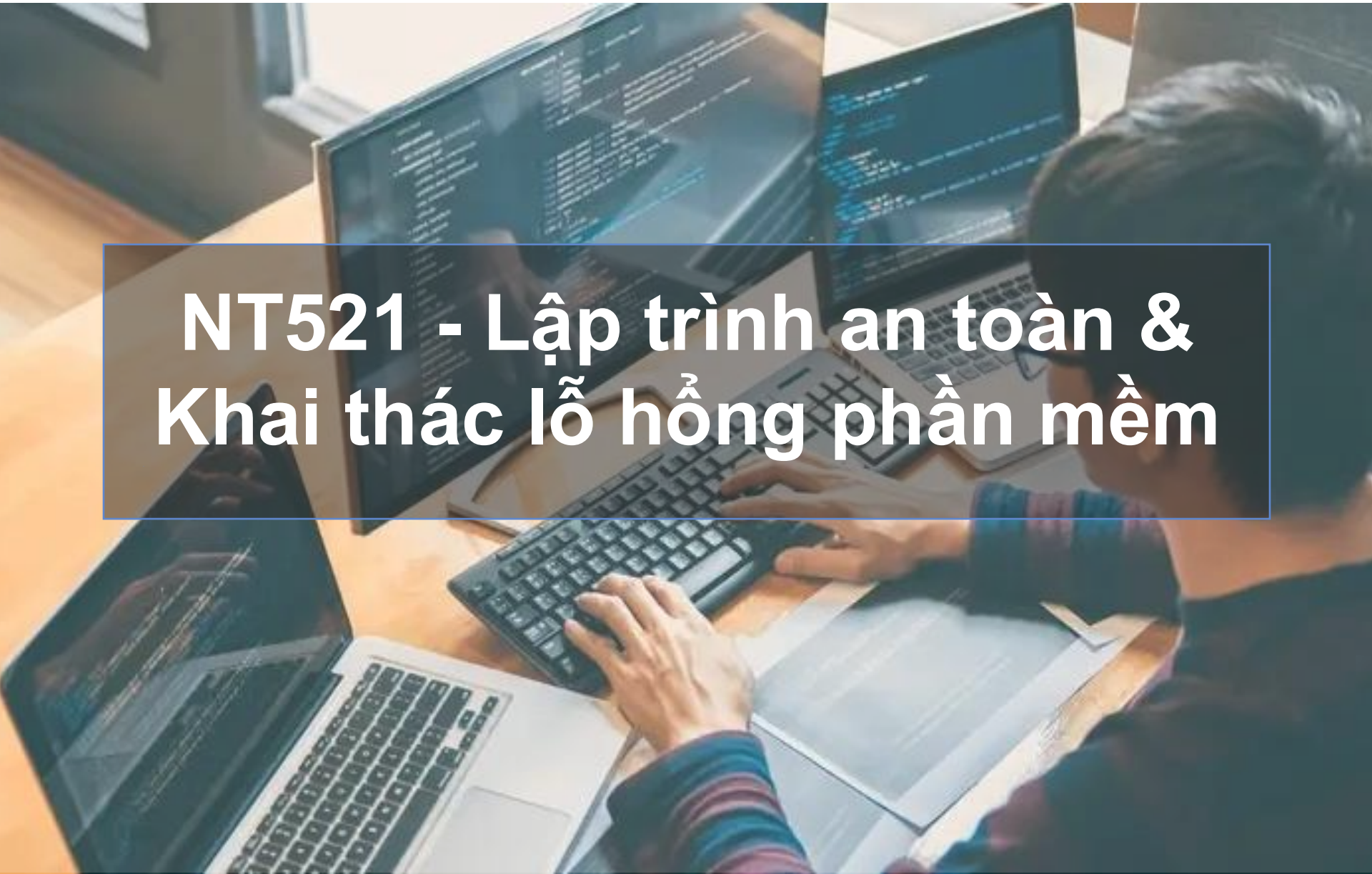


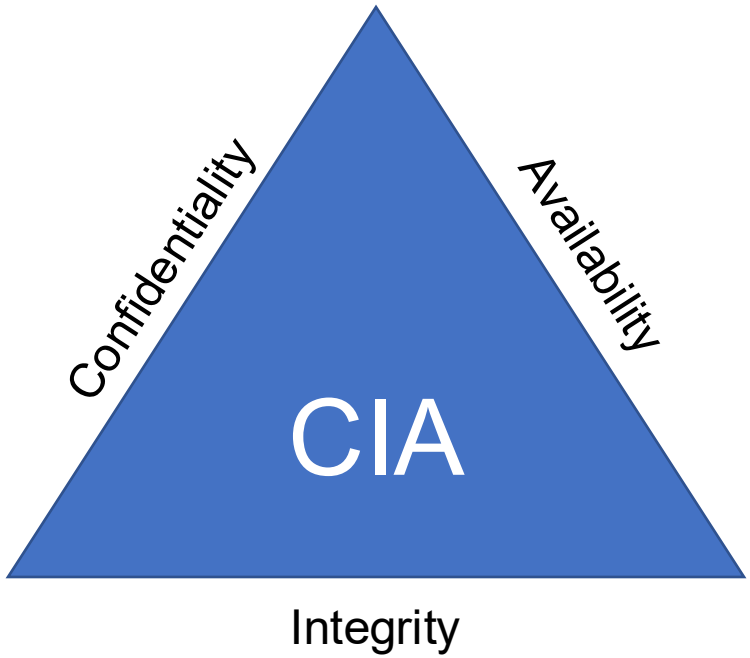
NT521 - Lập trình an toàn & Khai thác lỗ hổng phần mềm

SSDLC - Qui trình phát triển phần mềm an toàn

Nội dung

- Định nghĩa Quy trình phát triển phần mềm an toàn (Secure SDLC – SSDLC)
- Đặc tả yêu cầu an toàn (Security Requirement)
- Thiết kế an toàn (Secure Design)
- Lập trình An toàn (Secure Code)
- Khung phát triển phần mềm an toàn (Security Software Development Framework – SSDF)
- Các lỗi thường gặp (Common Pitfalls)
- Kiểm thử và triển khai an toàn (Secure Testing and Deployment)

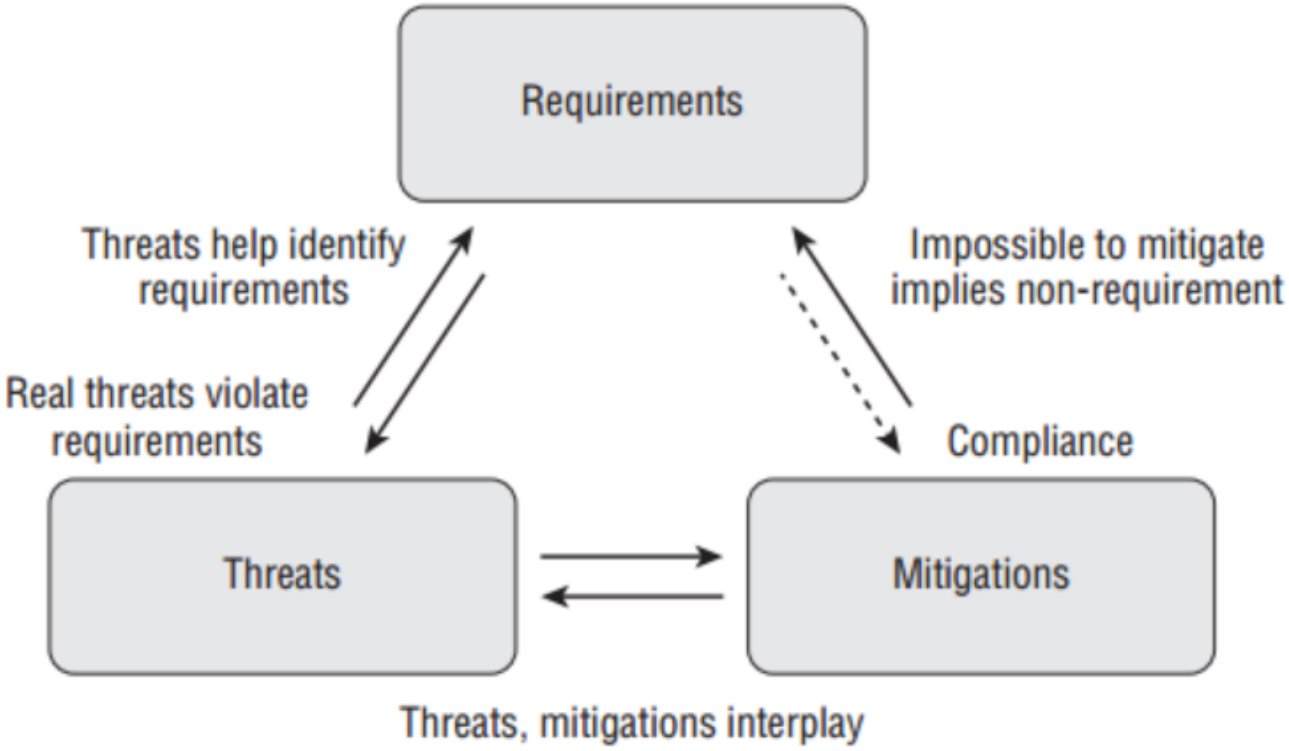
Security: Cơ bản



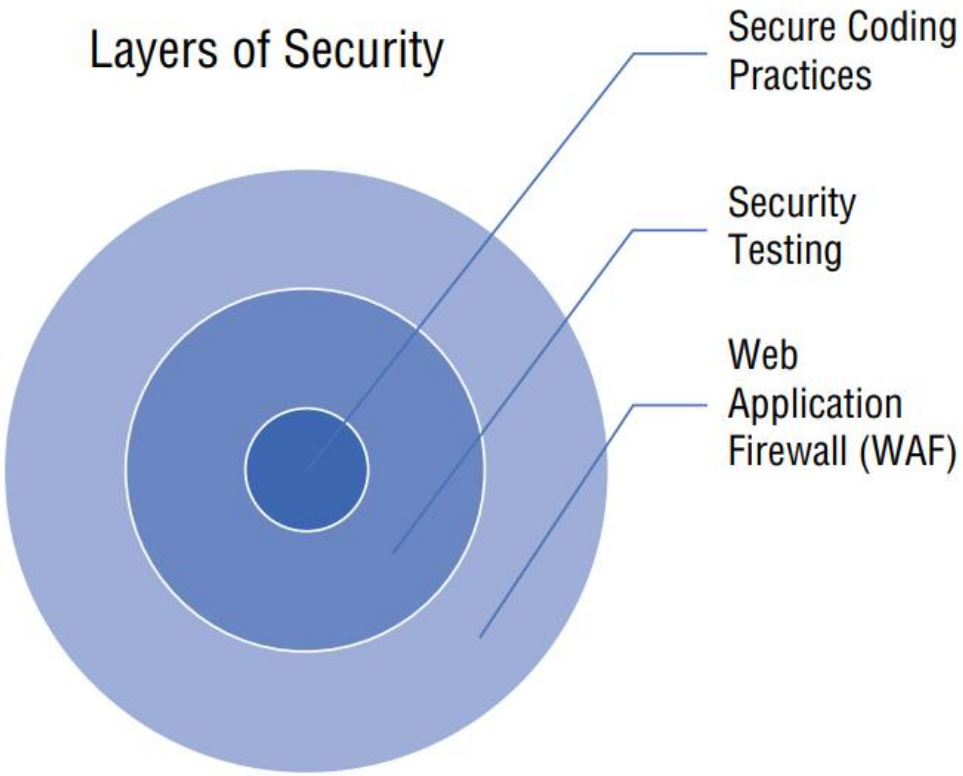
Bộ ba **CIA** (**B**ảo mật – **T**oàn vẹn – **S**ẵn sàng)

– Lý do team IT Security tồn tại

Requirements, Threats, và Mitigations



Defense in Depth
Phòng thủ theo chiều sâu

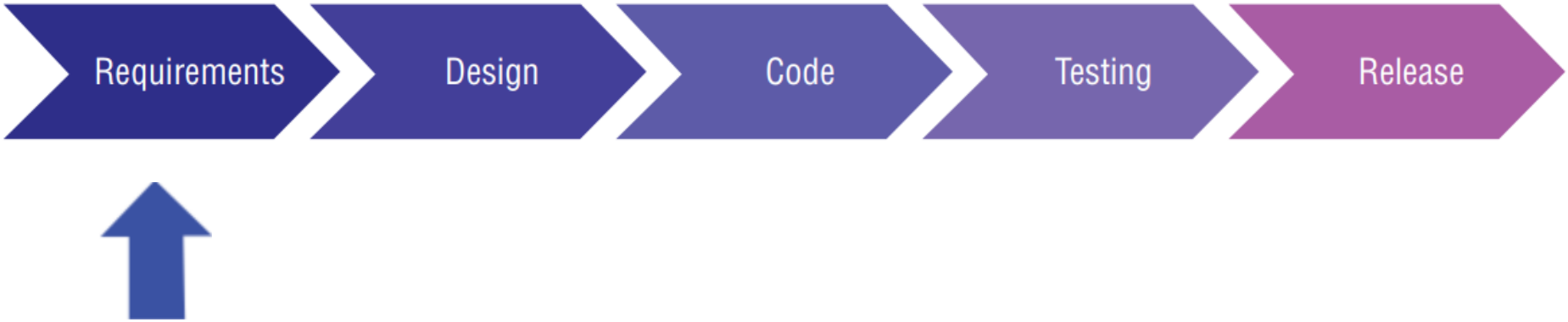


Ví dụ: Web Application Server gồm 3 layer bảo mật – 1 ví dụ của cơ chế phòng thủ theo chiều sâu (defense in depth)

Secure SDLC?

SDLC

Security Requirement



- Hệ thống có chứa hoặc tiếp cận với các loại dữ liệu bí mật, nhạy cảm, hay dữ liệu định danh cá nhân (PII)?
- Dữ liệu được lưu trữ ở đâu và như thế nào? Ứng dụng sẽ được công khai (qua internet) hay sử dụng nội bộ (intranet)?
- Ứng dụng này có thực hiện các tác vụ nhạy cảm hay quan trọng không (ví dụ chuyển tiền, mở khóa, hay vận chuyển thuốc)?
- Ứng dụng này có thực hiện bất kỳ tác vụ rủi ro nào không (ví dụ cho phép người dùng upload các file)?
- Hệ thống cần đảm bảo tính sẵn sàng ở mức độ nào?
- Có cần 99.999% up time hay không? (Note: thực tế là không có hệ thống nào cần mức độ up time này)

Security Requirement

- Encryption: khi lưu trữ hoặc truyền
- Never Trust System Input
- Encoding and Escaping
- Third-party Components
- Security Headers: Seatbelts for Web Apps
- Secure Your Cookies
- Password, Storage: password manager, secret store
- Backup – Rollback
- Framework Security Features
- File Uploads
- Errors and Logging
- Input Validation and Sanitization
- Authorization and Authentication
- Parameterized Queries
- Least Privilege

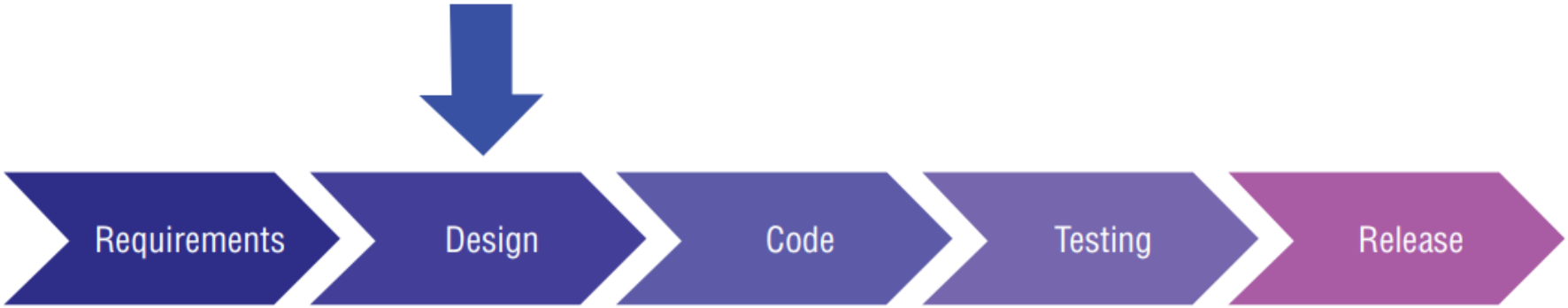
Ví dụ: Security requirements của web

- Mã hóa dữ liệu khi lưu trữ và truyền.
- Kiểm chứng tất cả input của người dùng
- Quét tất cả thư viện, framework của bên thứ 3 để biết các lỗ hổng hiện có trước và trong khi sử dụng, cập nhật bản mới nhất
- Hash và salt tất cả mật khẩu người dùng.
- Sử dụng xác thực nhiều yếu tố cho các tài khoản quan trọng.
- Chỉ cho phép truy cập thông qua HTTPS, chuyển hướng tất cả request HTTP thành HTTPS, sử dụng phiên bản TLS mới nhất.
- Tránh code cứng
- Không để các thông tin nhạy cảm, quan trọng trong phần comment
- Ghi log tất cả các lỗi, đặc biệt các lỗi liên quan đến security
- Đảm bảo catch các exception, error → fail safe
- ...

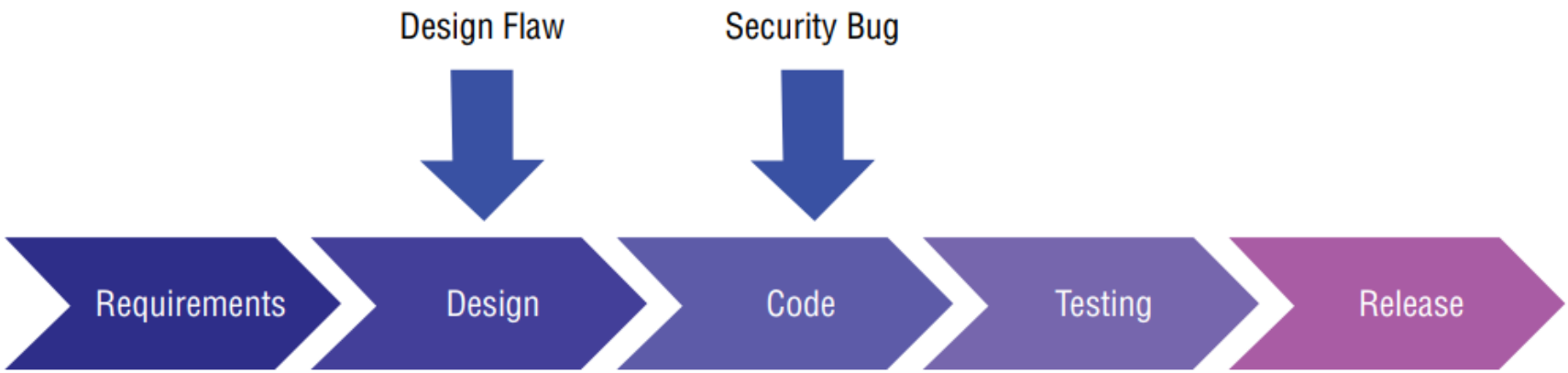
Secure Design

"Secure by design, in software engineering, means that the software has been designed from the ground up to be secure. Malicious practices are taken for granted and care is taken to minimize impact when a security vulnerability is discovered or on invalid user input"

- Phần mềm được thiết kế an toàn ngay từ ban đầu.
- Có tính đến các hoạt động độc hại có thể xảy ra.
- Nhằm giảm thiểu ảnh hưởng khi phát hiện lỗ hổng hoặc người dung nhập input không hợp lệ.

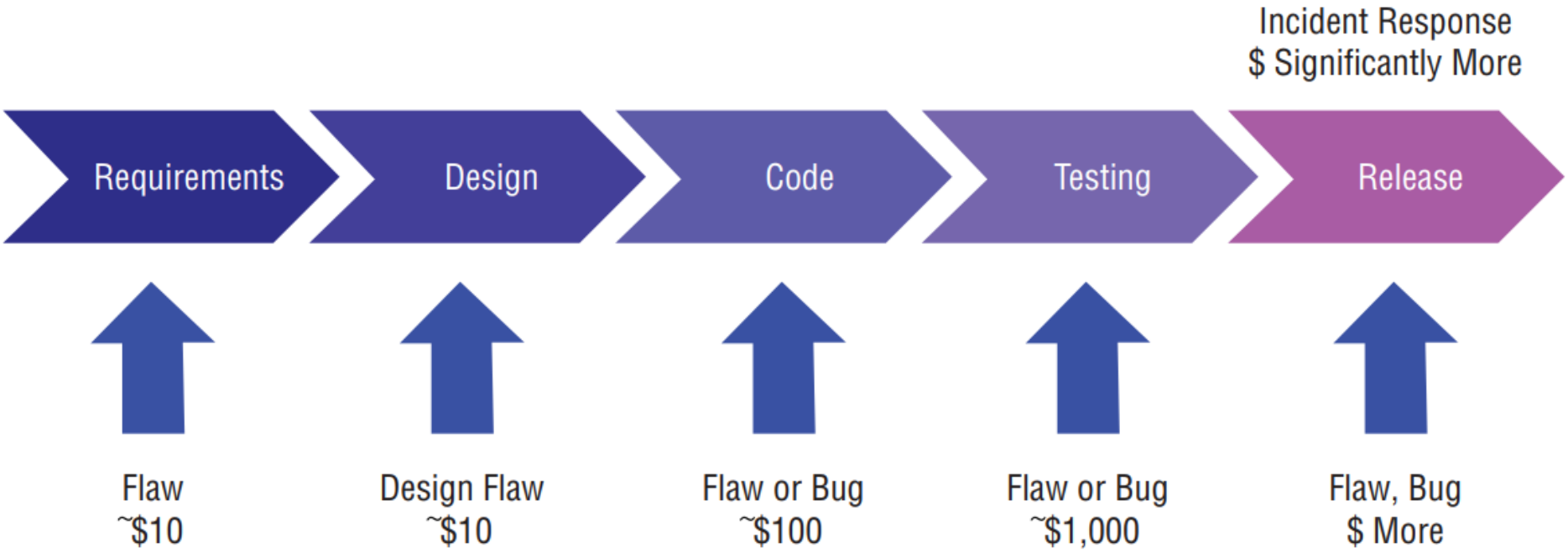


Design Flaw vs. Security Bug



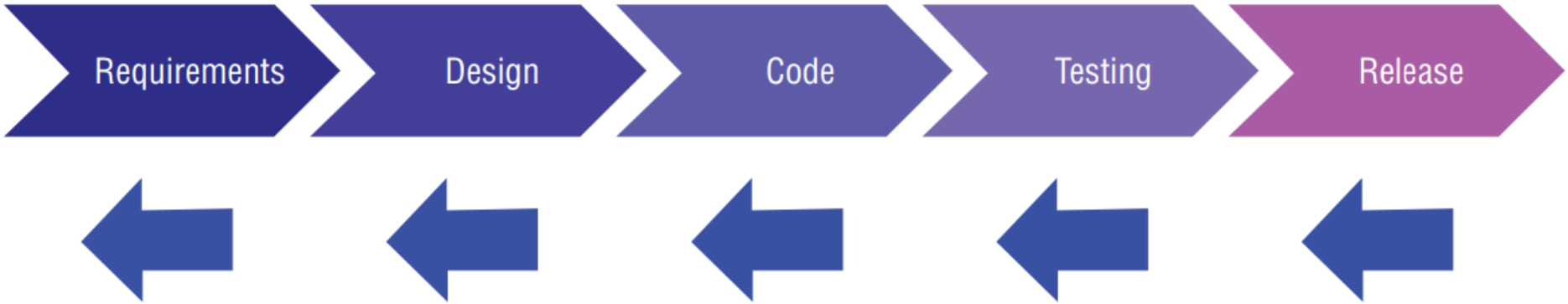
Design Flaw	Security bug
- Lỗi trong **thiết kế** phần mềm. - Cho phép user thực hiện các hành động vốn không được thực hiện. - Biện pháp: dùng security design concepts, requirements và threat modeling.	- Vấn đề trong **hiện thực** phần mềm, trong lập trình. - Cho phép user dùng ứng dụng một cách độc hại hoặc không chính xác. - Biện pháp: đánh giá mã nguồn, kiểm thử security, training và hướng dẫn về secure code.

Flaw: Tác động



Giải quyết vấn đề **càng trễ** trong SDLC, chi phí bỏ ra **càng lớn**.

Nguyên tắc Pushing Left/Shifting Left



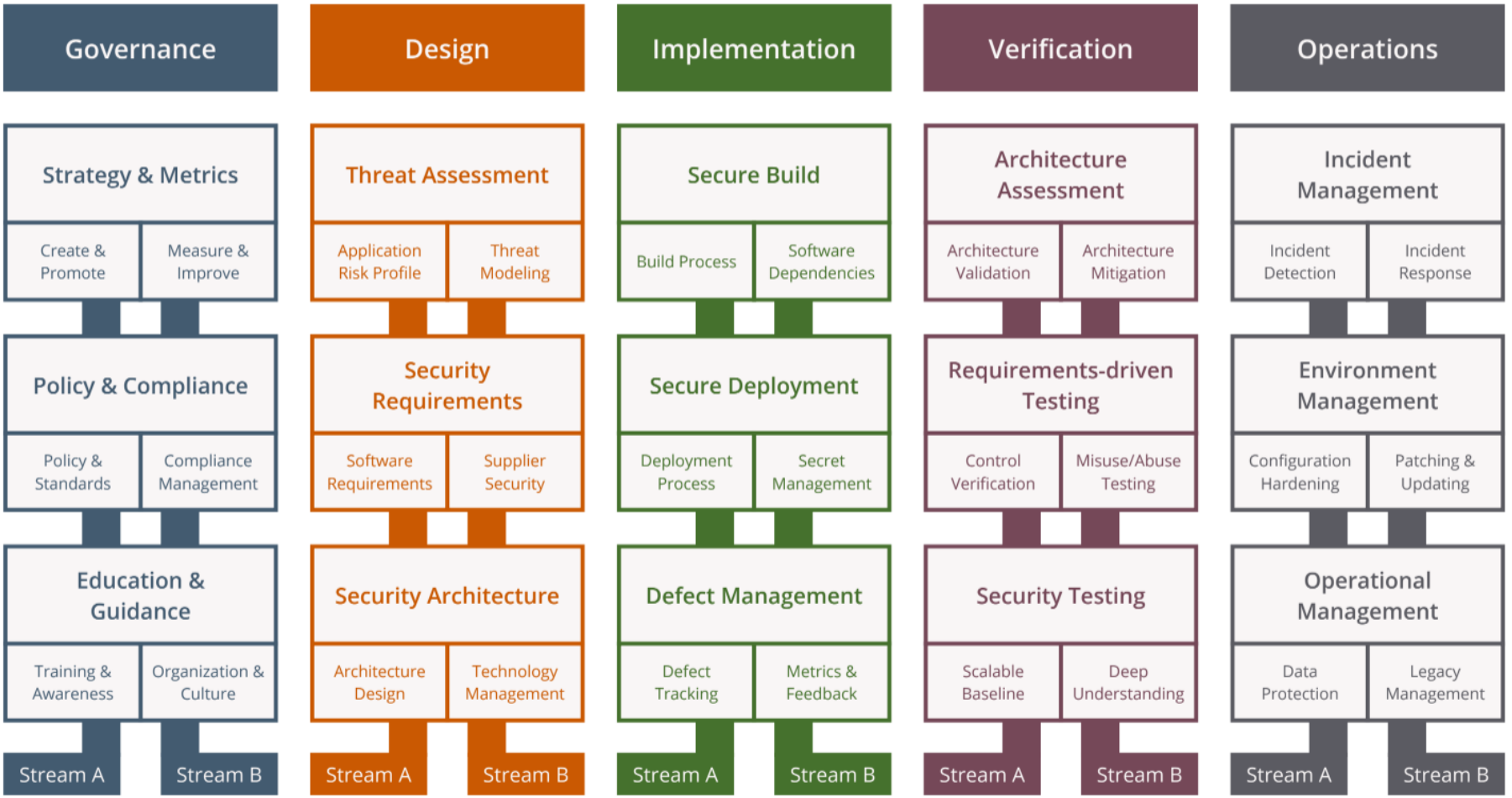
Cần đảm bảo security **ngay từ khi bắt đầu** dự án, chứ không phải lúc kết thúc.

Lỗ hổng phổ biến - Top 10 OWASP

OWASP Top 10 Vulnerabilities

OWASP Top 10 Vulnerabilities 2021	Position in 2017
1. Broken Access Control	5th
2. Cryptographic Failures	3rd
3. Injection	1st
4. Insecure Design	**New Category**
5. Security Misconfiguration	6th
6. Vulnerable and Outdated Components	9th
7. Identification and Authentication Failures	2nd
8. Software and Data Integrity Failures	New Category
9. Security Logging and Monitoring Failures	10th
10. Server-Side Request Forgery	New Category

Secure Design
OWASP SAMM



https://owaspsamm.org/model/

OWASP SAMM (2)

- **S**oftware **A**ssurance **M**aturity **M**odel (SAMM).
- Framework mở hỗ trợ hình thành và hiện thực chiến lược bảo mật phần mềm phù hợp với các rủi ro mà tổ chức phải đối mặt.
- Hỗ trợ:
  - Đánh giá các phương pháp bảo mật phần mềm hiện có của tổ chức với các mức độ và tiêu chí có sẵn.
  - Thiếp lập chương trình đảm bảo bảo mật phần mềm cân bằng trong các vòng lặp phát triển.
  - Demo các giải pháp cải tiến cụ thể đối với chương trình bảo mật.
  - Định nghĩa và đo lường các hoạt động liên quan đến bảo mật trong tổ chức.

Làm sao để tránh Insecure Design?

- **Tạo các user story tốt**
  - Cần các yêu cầu chức năng và phi chức năng về bảo mật.
  - User story: mô tả ngắn gọn, dễ hiểu của 1 chức năng phần mềm dưới góc nhìn của người dùng cuối.
  - Trong user story cũng cần ghi những flaw có thể xảy ra của phần mềm.
- **Yếu tố security trong quy trình phát triển**
  - Phát triển phần mềm an toàn: tính đến an toàn phần mềm ngay từ đầu và có kiểm thử bảo mật tích hợp.
- **Bảo mật và tách biệt các layer, thư viện**
  - Đảm bảo các layer (ứng dụng và mạng) tách biệt rõ ràng và sử dụng các thư viện design pattern an toàn.

Concepts

- Protecting Sensitive Data - Bảo vệ các dữ liệu nhạy cảm
- Never Trust, Always Verify/Zero Trust/Assume Breach - Không tin bất kỳ ai, luôn phải kiểm tra
- Backup and Rollback - Sao lưu và cho phép khôi phục
- Server-Side Security Validation - Xác thực bảo mật ở phía server
- Framework Security Features - Các tính năng bảo mật của framework
- Security Function Isolation - Cô lập các hàm bảo mật
- Application Partitioning - Chia nhỏ ứng dụng
- Secret Management (not mean the secrets that users store inside the software) - Quản lý các bí mật
- Re-authentication for Transactions (Avoiding CSRF) - Xác thực lại các giao dịch (tránh CSRF)

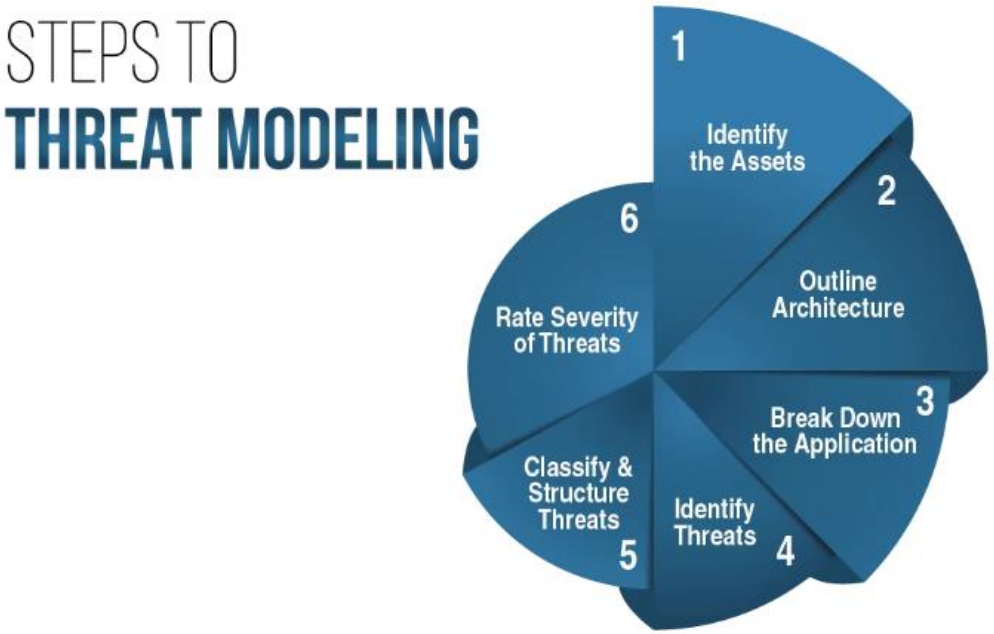
Dữ liệu trong ứng dụng

- Production data: dữ liệu mà phần mềm sử dụng khi chạy thực tế
  - Không nên dùng khi kiểm thử, phát triển phần mềm hay bất kỳ mục đích nào khác mục đích của tổ chức.
  - Phát triển và kiểm thử nên dùng dữ liệu đã che bớt thông tin (masked – anonymized).
  - Dữ liệu thật chỉ nên dùng khi chạy thực tế.
- Bảo vệ mã nguồn - “Security through obscurity”

Threat Modeling
Mô hình hóa mối đe dọa

- Vì sao cần?
  - Thiết kế phần mềm chủ yếu tập trung đảm bảo có tất cả các chức năng theo yêu cầu, thay vì đảm bảo phần mềm ***chỉ*** *hoạt động* theo cách mong muốn.
- **Threat modeling — quy trình xác định các mối đe dọa tiềm ẩn** đối với ứng dụng và **đảm bảo các biện pháp giảm thiểu** thích hợp đã được áp dụng.
- **Evil brainstorming –**

Xem xét phần mềm dưới góc nhìn của kẻ tấn công



Threat Modeling
Mô hình hóa mối đe dọa

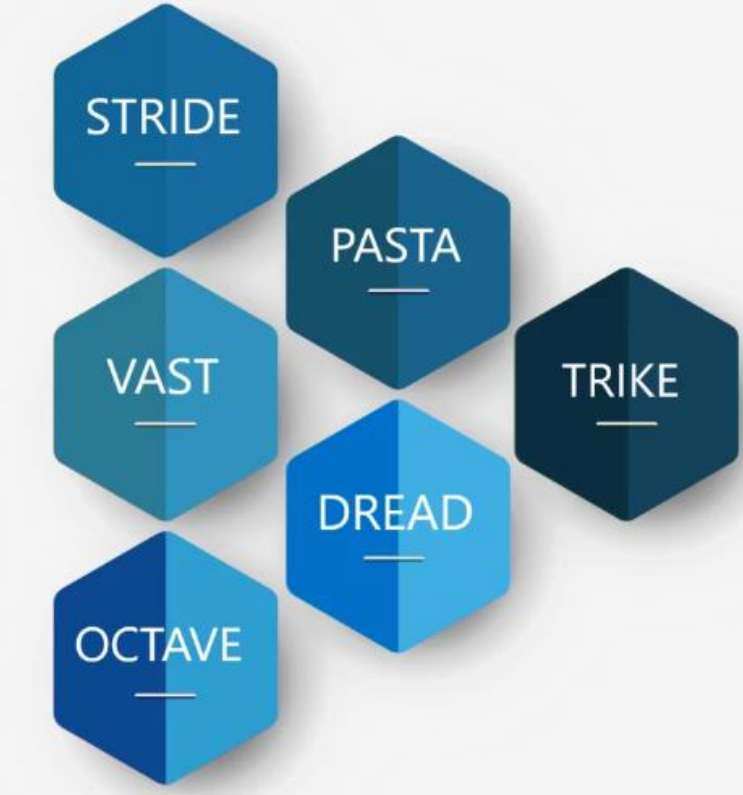
- Đưa ra các câu hỏi và đảm bảo đã cân nhắc các trường hợp xấu có thể xảy ra → concern.
- Đánh giá khả năng, ảnh hưởng → loại bỏ các concern không thể xảy ra hoặc không ảnh hưởng.
- Đánh giá (rate) các concern còn lại ở các mức độ cao, trung bình, thấp

  → Danh sách các threats (các concern đã được xác nhận)

  → Danh sách các risk (đánh giá cao/trung bình/thấp)
- Đưa ra kế hoạch: giảm thiểu (sửa/loại bỏ), tài liệu hóa, chấp nhận...

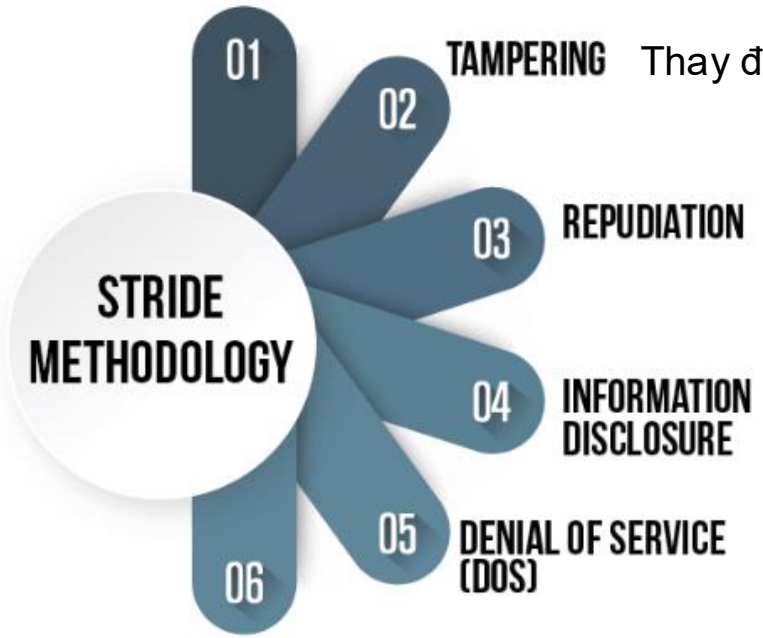
Threat Modeling
Mô hình hóa mối đe dọa

- Một số phương pháp mô hình hóa mối đe dọa: STRIDE, PASTA, TRIKE, DREAD, VAST, OCTAVE.



STRIDE

- **STRIDE**, Loren Kohnfelder và Praerit Garg,1999
  - Tập trung vào các vấn đề xung quanh chứng thực, phân quyền, CIA và thóa thác trách nhiệm



SPOOFING Truy cập bất hợp pháp và sử dụng thông tin chứng thực của user khác

TAMPERING Thay đổi dữ liệu độc hại hoặc trái phép

REPUDIATION
- Phủ nhận đã thực hiện hành vi độc hại
- Tránh thoái thác trách nhiệm (non-repudiation) là khả năng nhận ra mối đe dọa thoái thác trách nhiệm

INFORMATION DISCLOSURE Lộ thông tin với những user không được phép truy cập

DENIAL OF SERVICE (DOS)
- Từ chối dịch vụ với các user hợp lệ
- Các mối đe dọa với tính sẵn sàng và tin cậy của hệ thống

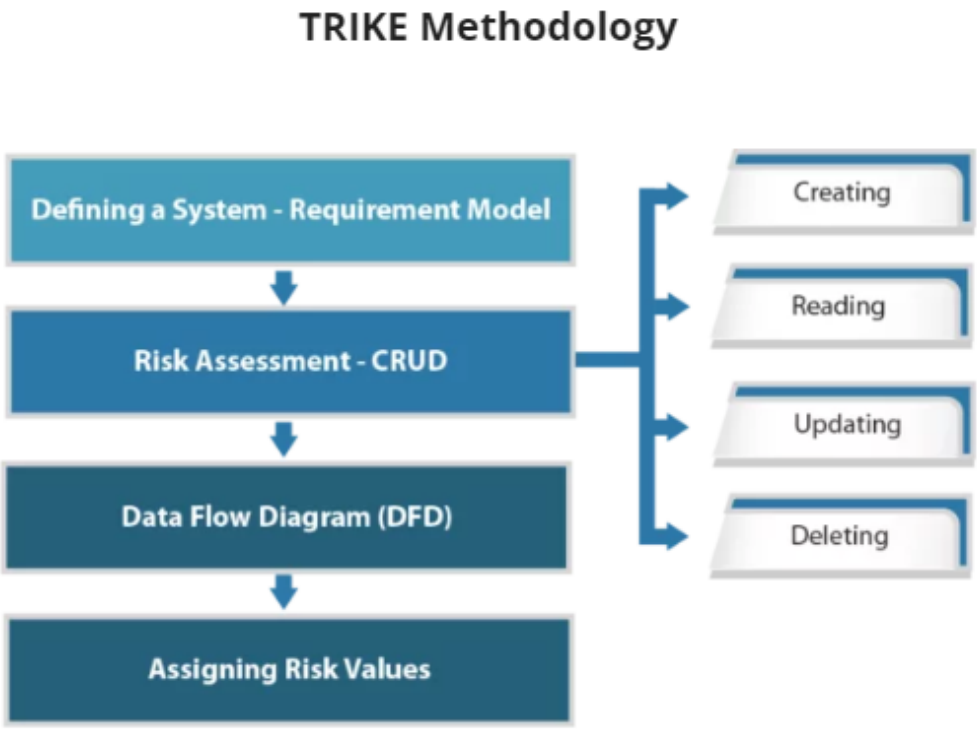
ELEVATION OF PRIVILEGE
- Các user không có quyền cố gắng lấy quyền để chiếm hệ thống
- Xâm nhập hiệu quả và trở thành 1 phần của hệ thống tin cậy

PASTA

- **P**rocess for **A**ttack **S**imulation and **T**hreat **A**nalysis
  - Tập trung vào các yêu cầu kinh doanh, chức năng, sơ đồ dữ liệu...
- **PASTA**: phương pháp gồm 7 bước để tạo 1 quy trình để **mô phỏng tấn công** vào các ứng dụng IT, **phân tích mối đe dọa**, **nguồn gốc, rủi ro** gây ra cho tổ chức và **cách giảm thiểu** chúng.
- Mục tiêu: xác định và liệt kê các mối đe dọa, và gán điểm → giúp xác định các biện pháp cần thực hiện để giảm thiểu các mối đe dọa.

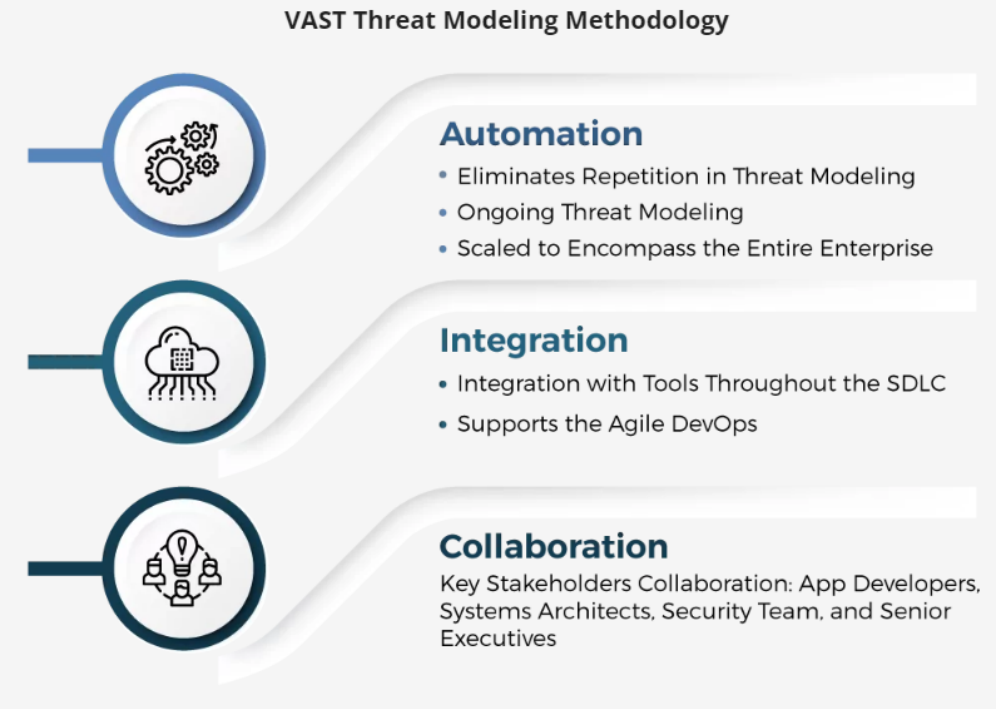
TRIKE

- **TRIKE –** phương pháp mô hình hóa mối đe dọa mã nguồn mở, dùng khi đánh giá bảo mật từ góc nhìn quản lý rủi ro.
- ***Data Flow Diagram*** được tạo để mô phỏng luồng dữ liệu và các user thực hiện các hành vi trong hệ thống.
- Trong mô hình này, các mối **đe dọa** được phân tích nhằm liệt kê và **gán giá trị rủi ro**. Dựa trên đó, các biện pháp kiểm soát an ninh và phòng ngừa được chọn để giải quyết các mối đe dọa theo mức độ ưu tiên và giá trị rủi ro được gán.



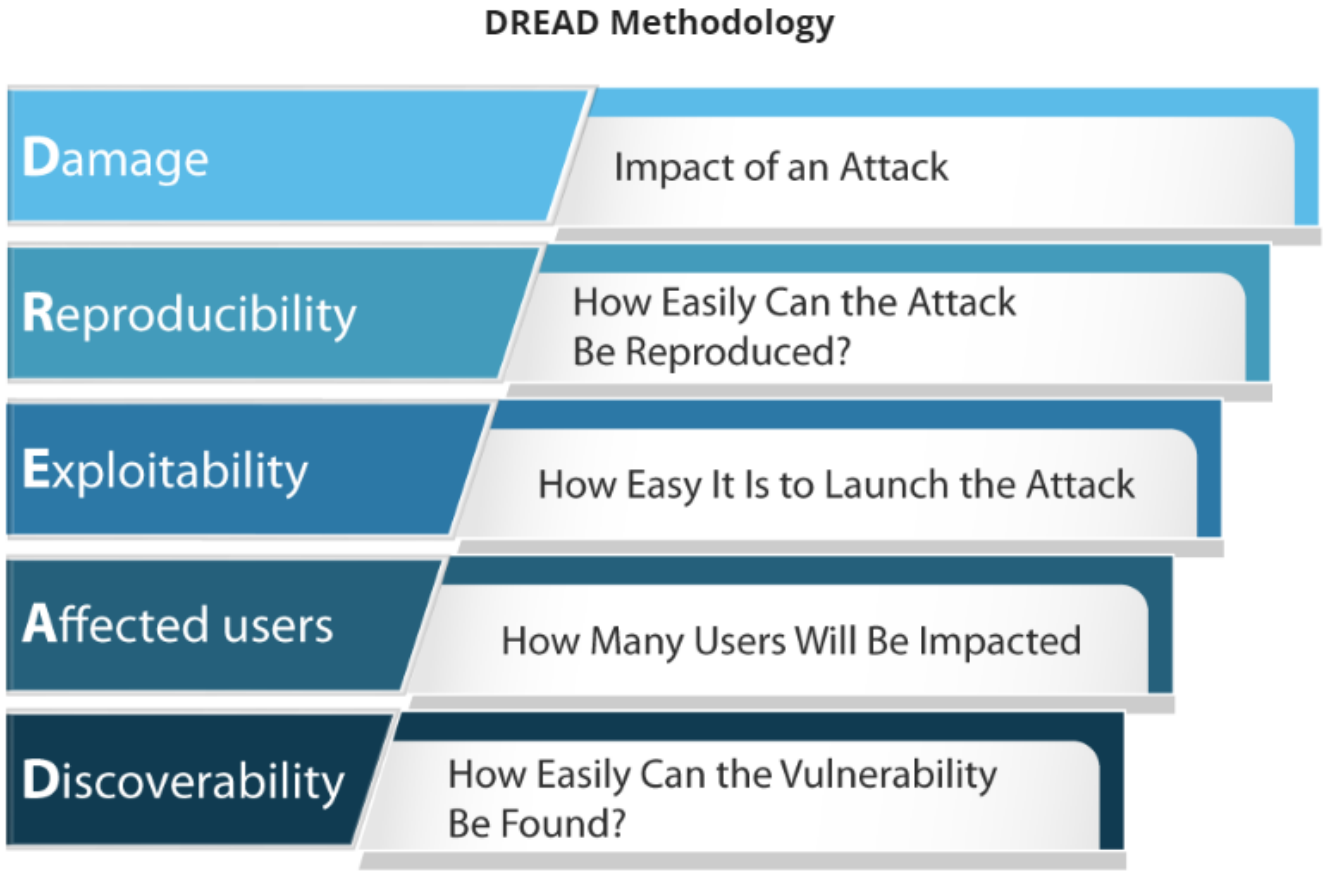
VAST

- **VAST** - **V**isual, **A**gile, and **S**imple **T**hreat
- Mô hình hóa mối đe dọa **tự động** bao hàm quy trình phát triển phần mềm trong toàn tổ chức có **tích hợp** phù hợp với các công cụ và **cộng tác** với các bên liên quan như lập trình viên, kiến trúc sư, chuyên gia bảo mật và các lãnh đạo trong tổ chức.



DREAD

- **DREAD** nhằm đánh giá, phân tích và tìm xác suất xảy ra rủi ro bằng cách đánh giá các mối đe dọa ở các phương diện bên dưới.

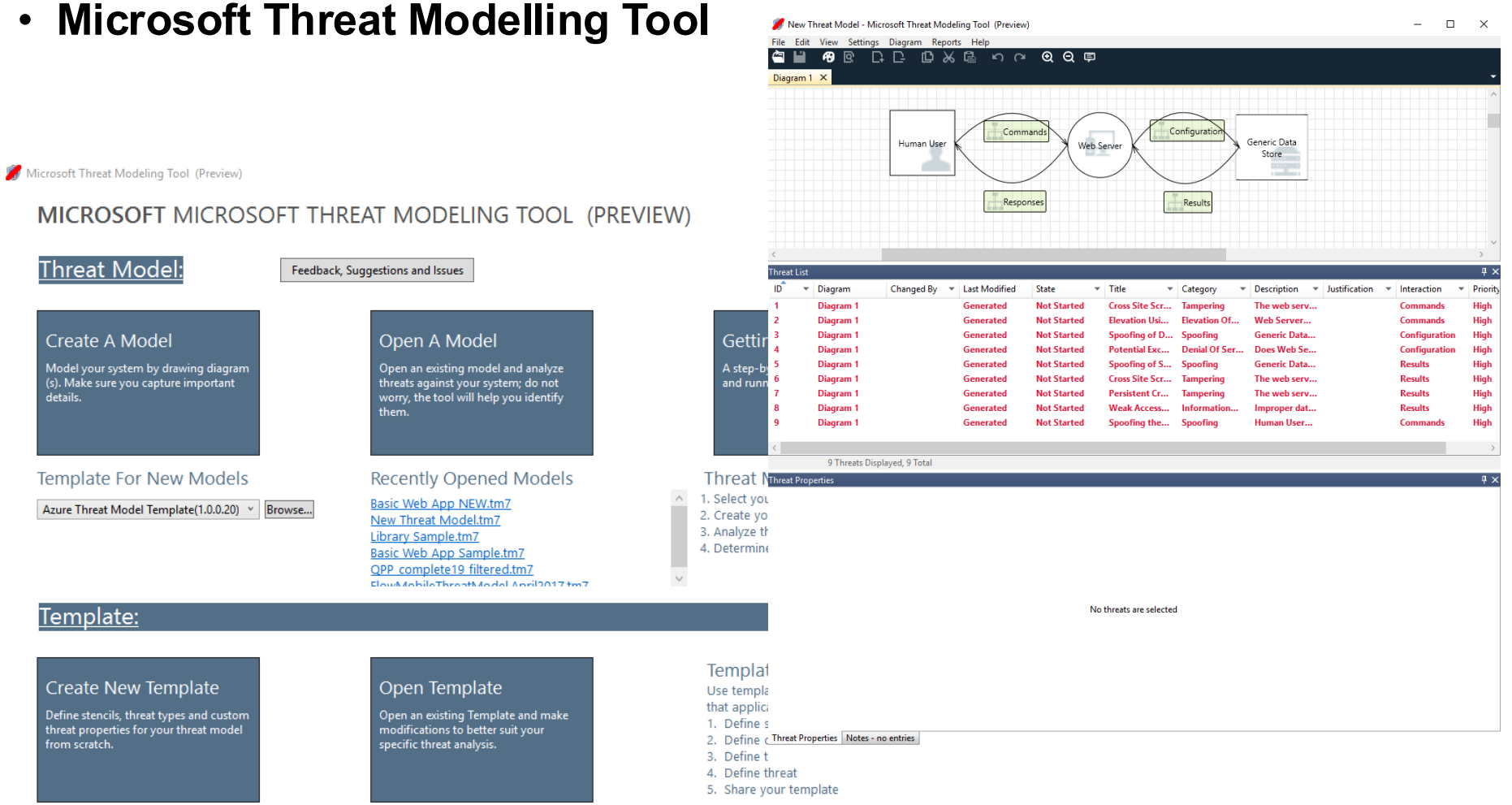


OCTAVE

- **OCTAVE** (**O**perationally **C**ritical **T**hreat, **A**sset, and **V**ulnerability **E**valuation)
- Xác định, đánh giá và quản lý rủi ro cho các tài sản CNTT.
- Xác định các thành phần quan trọng của an toàn thông tin và các mối đe dọa có thể ảnh hưởng đến tính bảo mật, toàn vẹn và sẵn sàng (CIA) của chúng.

Các công cụ (1)

- **Microsoft Threat Modelling Tool**



Các công cụ (2)

- **OWASP Threat Dragon** là công cụ mô hình hóa được dùng để tạo các diagram mô hình hóa mối đe dọa trong quy trình phát triển phần mềm an toàn.
- Threat Dragon tuân theo các giá trị và nguyên tắc của **threat modeling manifesto**.
- OWASP Threat Modeling Cheat Sheet: https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html

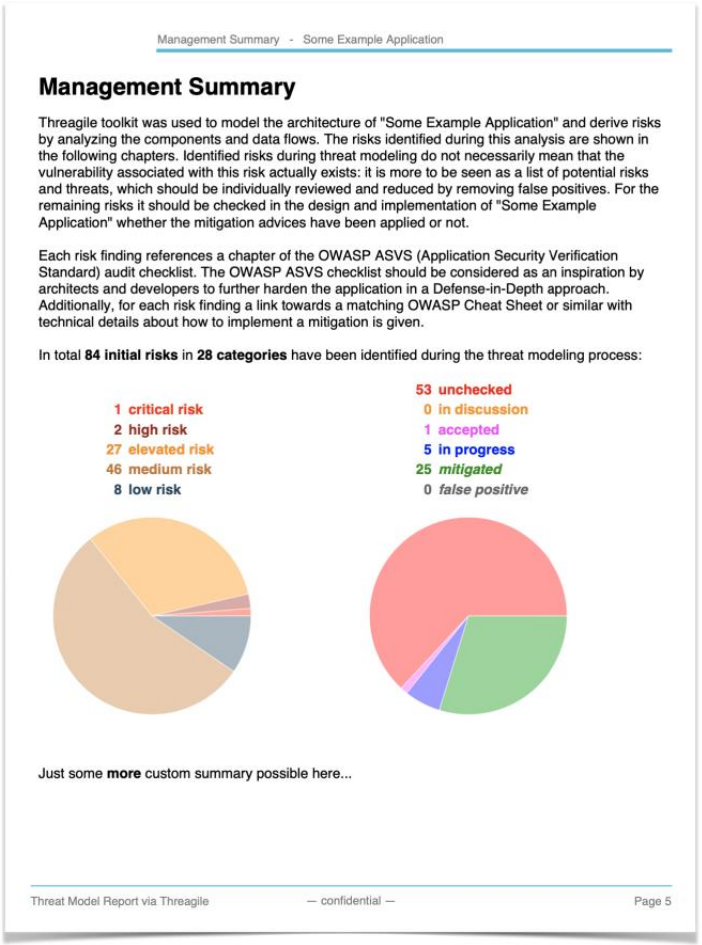
https://www.threatmodelingmanifesto.org/

Các công cụ (3) - Threagile

- **Threagile** cho phép mô hình hóa mối đe dọa Agile một cách liền mạch, và có khả năng tích hợp cao với môi trường DevSecOps.
- Threagile được phát hành tại Black Hat Arsenal 2020 and DEF CON 2020 AppSec Village conferences.
- Bộ tool mã nguồn mở, cho phép mô hình hóa một kiến trúc với các asset theo kiểu khai báo nhanh (agile) dưới dạng file YAML trong IDE hoặc bất kỳ trình soạn thảo YAML nào. Sau khi sử dụng Threagile, một tập các rule rủi ro được chạy để kiểm tra bảo mật mô hình kiến trúc và tạo một báo cáo với các rủi ro tiềm ẩn và đề xuất giải pháp giảm thiểu.

https://threagile.io/
https://christian-schneider.net/slides/DEF-CON-2020-Threagile.pdf

Các công cụ (3) - Threagile



Management Summary - Some Example Application

Management Summary

Threagile toolkit was used to model the architecture of "Some Example Application" and derive risks by analyzing the components and data flows. The risks identified during this analysis are shown in the following chapters. Identified risks during threat modeling do not necessarily mean that the vulnerability associated with this risk actually exists: it is more to be seen as a list of potential risks and threats, which should be individually reviewed and reduced by removing false positives. For the remaining risks it should be checked in the design and implementation of "Some Example Application" whether the mitigation advices have been applied or not.

Each risk finding references a chapter of the OWASP ASVS (Application Security Verification Standard) audit checklist. The OWASP ASVS checklist should be considered as an inspiration by architects and developers to further harden the application in a Defense-in-Depth approach. Additionally, for each risk finding a link towards a matching OWASP Cheat Sheet or similar with technical details about how to implement a mitigation is given.

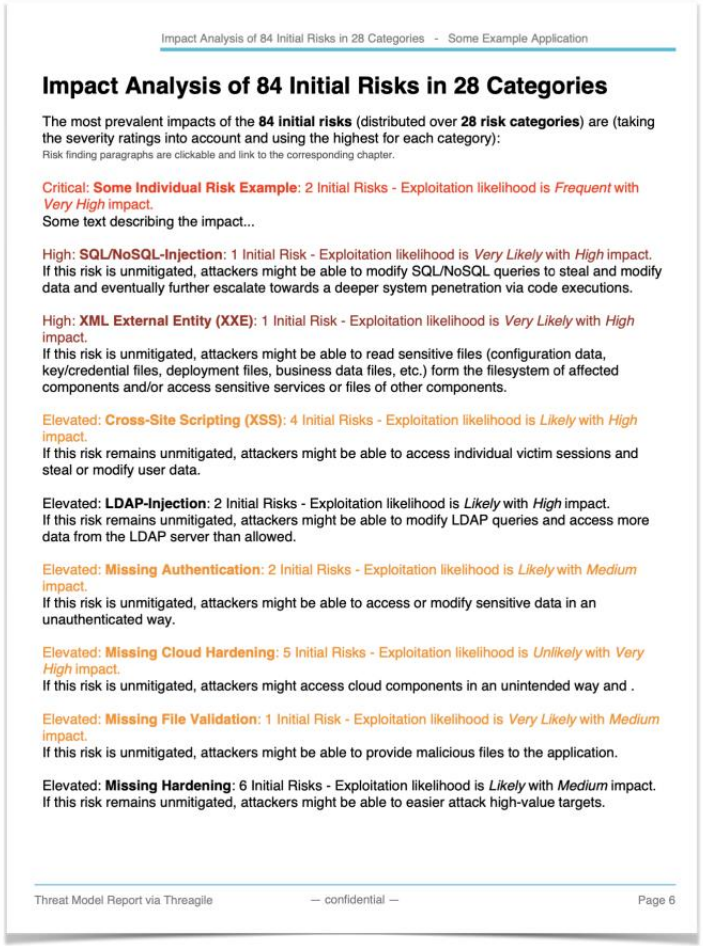
In total **84 initial risks** in **28 categories** have been identified during the threat modeling process:

1 critical risk
2 high risk
27 elevated risk
46 medium risk
8 low risk

53 unchecked
0 in discussion
1 accepted
5 in progress
25 *mitigated*
0 *false positive*

Just some **more** custom summary possible here...

Threat Model Report via Threagile — confidential — Page 5



Impact Analysis of 84 Initial Risks in 28 Categories - Some Example Application

Impact Analysis of 84 Initial Risks in 28 Categories

The most prevalent impacts of the **84 initial risks** (distributed over **28 risk categories**) are (taking the severity ratings into account and using the highest for each category):
Risk finding paragraphs are clickable and link to the corresponding chapter.

Critical: **Some Individual Risk Example**: 2 Initial Risks - Exploitation likelihood is *Frequent* with *Very High* impact.
Some text describing the impact...

High: **SQL/NoSQL-Injection**: 1 Initial Risk - Exploitation likelihood is *Very Likely* with *High* impact.
If this risk is unmitigated, attackers might be able to modify SQL/NoSQL queries to steal and modify data and eventually further escalate towards a deeper system penetration via code executions.

High: **XML External Entity (XXE)**: 1 Initial Risk - Exploitation likelihood is *Very Likely* with *High* impact.
If this risk is unmitigated, attackers might be able to read sensitive files (configuration data, key/credential files, deployment files, business data files, etc.) form the filesystem of affected components and/or access sensitive services or files of other components.

Elevated: **Cross-Site Scripting (XSS)**: 4 Initial Risks - Exploitation likelihood is *Likely* with *High* impact.
If this risk remains unmitigated, attackers might be able to access individual victim sessions and steal or modify user data.

Elevated: **LDAP-Injection**: 2 Initial Risks - Exploitation likelihood is *Likely* with *High* impact.
If this risk remains unmitigated, attackers might be able to modify LDAP queries and access more data from the LDAP server than allowed.

Elevated: **Missing Authentication**: 2 Initial Risks - Exploitation likelihood is *Likely* with *Medium* impact.
If this risk is unmitigated, attackers might be able to access or modify sensitive data in an unauthenticated way.

Elevated: **Missing Cloud Hardening**: 5 Initial Risks - Exploitation likelihood is *Unlikely* with *Very High* impact.
If this risk is unmitigated, attackers might access cloud components in an unintended way and .

Elevated: **Missing File Validation**: 1 Initial Risk - Exploitation likelihood is *Very Likely* with *Medium* impact.
If this risk is unmitigated, attackers might be able to provide malicious files to the application.

Elevated: **Missing Hardening**: 6 Initial Risks - Exploitation likelihood is *Likely* with *Medium* impact.
If this risk remains unmitigated, attackers might be able to easier attack high-value targets.

Threat Model Report via Threagile — confidential — Page 6

https://threagile.io/
https://christian-schneider.net/slides/DEF-CON-2020-Threagile.pdf

Threagile
Agile Threat Modeling

Các công cụ (4)

Một số công cụ khác:

- ThreatModeler
- securiCAD Professional
- IriusRisk
- SD Elements
- Tutamen

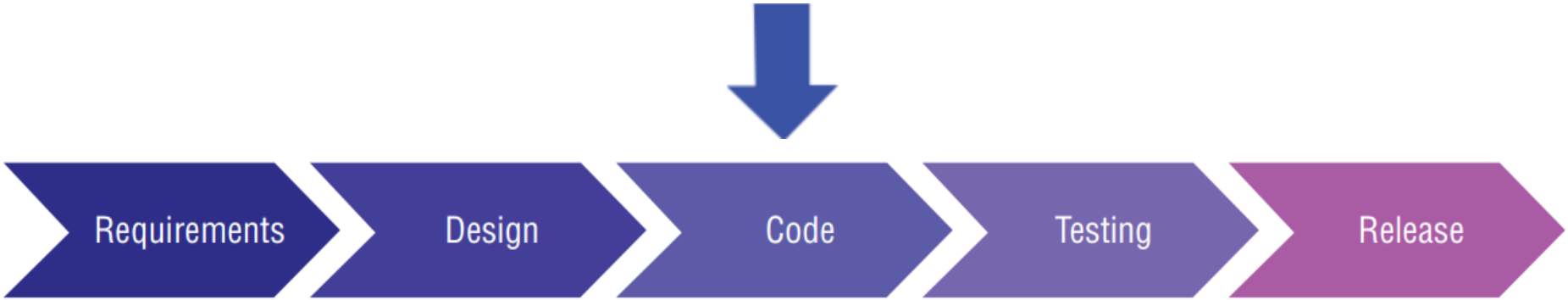
Câu hỏi thảo luận – 2-3 SV/nhóm

1) **Kể tên 3 loại thông tin có thể được xem là "bí mật – secret".** Theo bạn, nên **lưu trữ** các bí mật (secret) của phần mềm ở đâu? Và ứng dụng nên **truy cập** chúng như thế nào?
2) **Giả sử có 1 ứng dụng mobile banking**. Kể tên ít nhất **2 mối đe dọa (threat)** có thể xảy ra, đánh giá khả năng và thiệt hại của chúng với các mức độ (cao, trung bình, thấp)? Đề xuất biện pháp khắc phục?

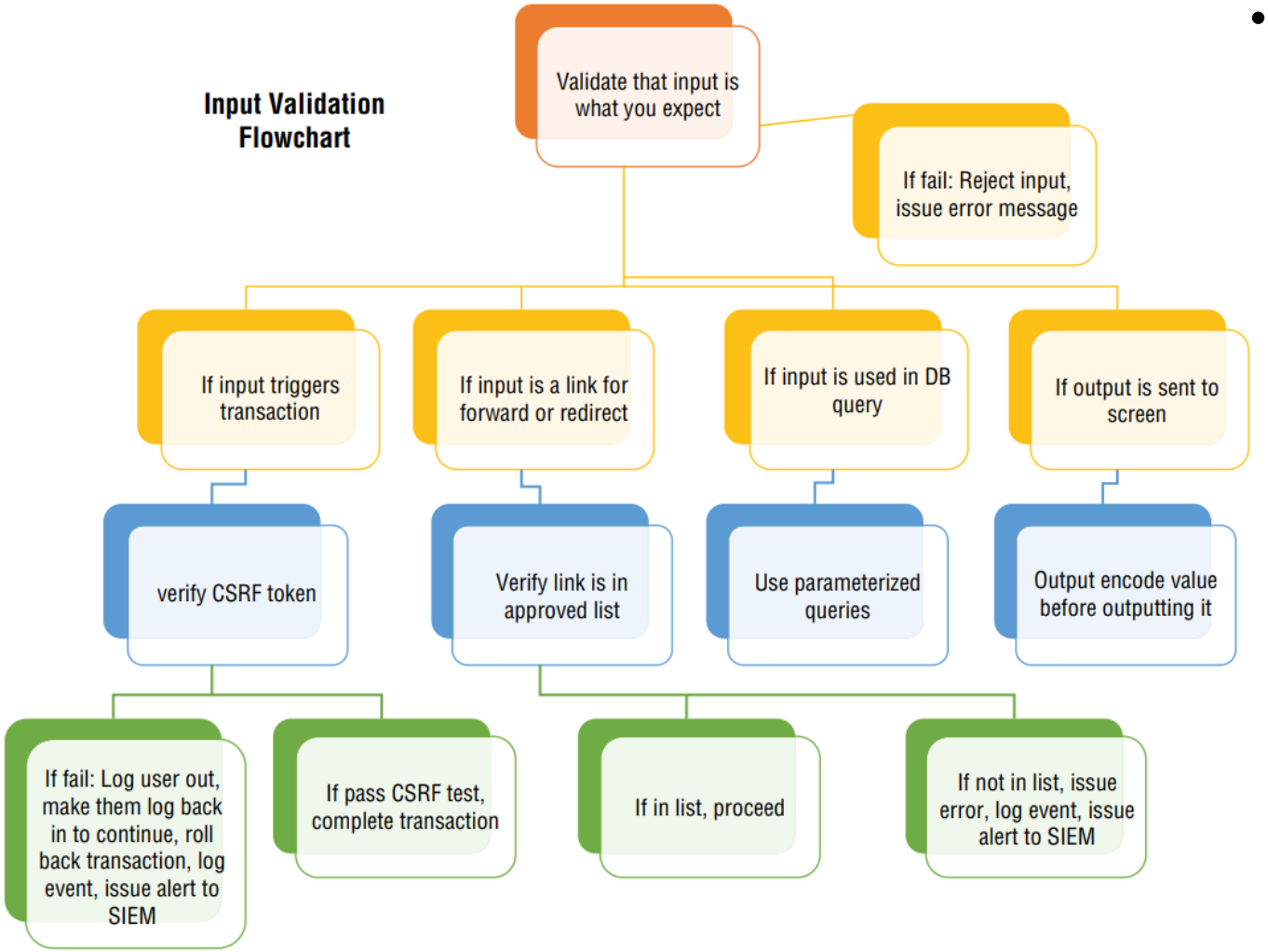
Secure Code – Lập trình an toàn

- **Lựa chọn framework và ngôn ngữ lập trình**
- **Ngôn ngữ và framework lập trình:**
  - *Sử dụng framework đang và sẽ được hỗ trợ trong thời gian dài.*
  - *Chọn phiên bản mới nhất hoặc mới-thứ-hai.*
  - *Hỗ trợ security → có đáp ứng được các tính năng cần thiết?*

SAY NO: framework không an toàn, không còn hỗ trợ, có vấn đề



Secure Code



- Untrusted Data – Dữ liệu không tin cậy

Secure Code

- **HTTP methods:** *tất cả các method HTTP không sử dụng thì nên disable để giảm khả năng tấn công.*
- **Identity – Quản lý định danh:**
  - *Không nên tự tạo hệ thống riêng, nên sử dụng các giải pháp có sẵn. Ví dụ: Active Directory*
  - *Nếu có yêu cầu đặc biệt cần dựng hệ thống riêng, nên sử dụng giao thức chuyên dụng như OAUTH.*
- **Authentication và Authorization – Chứng thực và Phân quyền truy cập**
- **Session Management – Quản lý phiên làm việc**
- **Bounds Checking**
- **Error Handling, Logging, and Monitoring – Xử lý lỗi, ghi log và giám sát**

Best Practices trong doanh nghiệp

Reference standards	Description and reference
CERT Secure Coding	• This provides secure coding standards for C, C++, Java, Perl, and Android.
Find Security Bugs	• This provides bug patterns with samples of vulnerable code and solution for Java.
CWE	• This provides vulnerable source code samples from the perspective of common software weaknesses. The coding samples cover C, C++, Java, and PHP.
Android	• Android Application Secure Design and Secure Coding Guidebook

Best Practices trong doanh nghiệp (2)

OWASP SKF	• OWASP Security Knowledge Framework. • It can be used as an internal security knowledge base, which includes OWASP ASVS and secure coding knowledge.
PHP Security	• OWASP PHP Security Cheat Sheet
OWASP Code Review	• OWASP Code Review Project
Apple Secure Coding Guide	• Apple Secure Coding Guide
Go	• Secure Coding Practices for GO language
JavaScript	• JavaScript Secure Coding Practices
Python	• OWASP Python Security Project

Best Practices trong doanh nghiệp (3)

SEI CERT Oracle Coding Standard for Java

Rules

- Rule 00. Input Validation and Data Sanitization (IDS)
- Rule 01. Declarations and Initialization (DCL)
- Rule 02. Expressions (EXP)
- Rule 03. Numeric Types and Operations (NUM)
- Rule 04. Characters and Strings (STR)
- Rule 05. Object Orientation (OBJ)
- Rule 06. Methods (MET)
- Rule 07. Exceptional Behavior (ERR)
- Rule 08. Visibility and Atomicity (VNA)
- Rule 09. Locking (LCK)
- Rule 10. Thread APIs (THI)
- Rule 11. Thread Pools (TPS)
- Rule 12. Thread-Safety Miscellaneous (TSM)
- Rule 13. Input Output (FIO)
- Rule 14. Serialization (SER)
- Rule 15. Platform Security (SEC)
- Rule 16. Runtime Environment (ENV)
- Rule 17. Java Native Interface (JNI)
- Rule 49. Miscellaneous (MSC)
- Rule 50. Android (DRD)

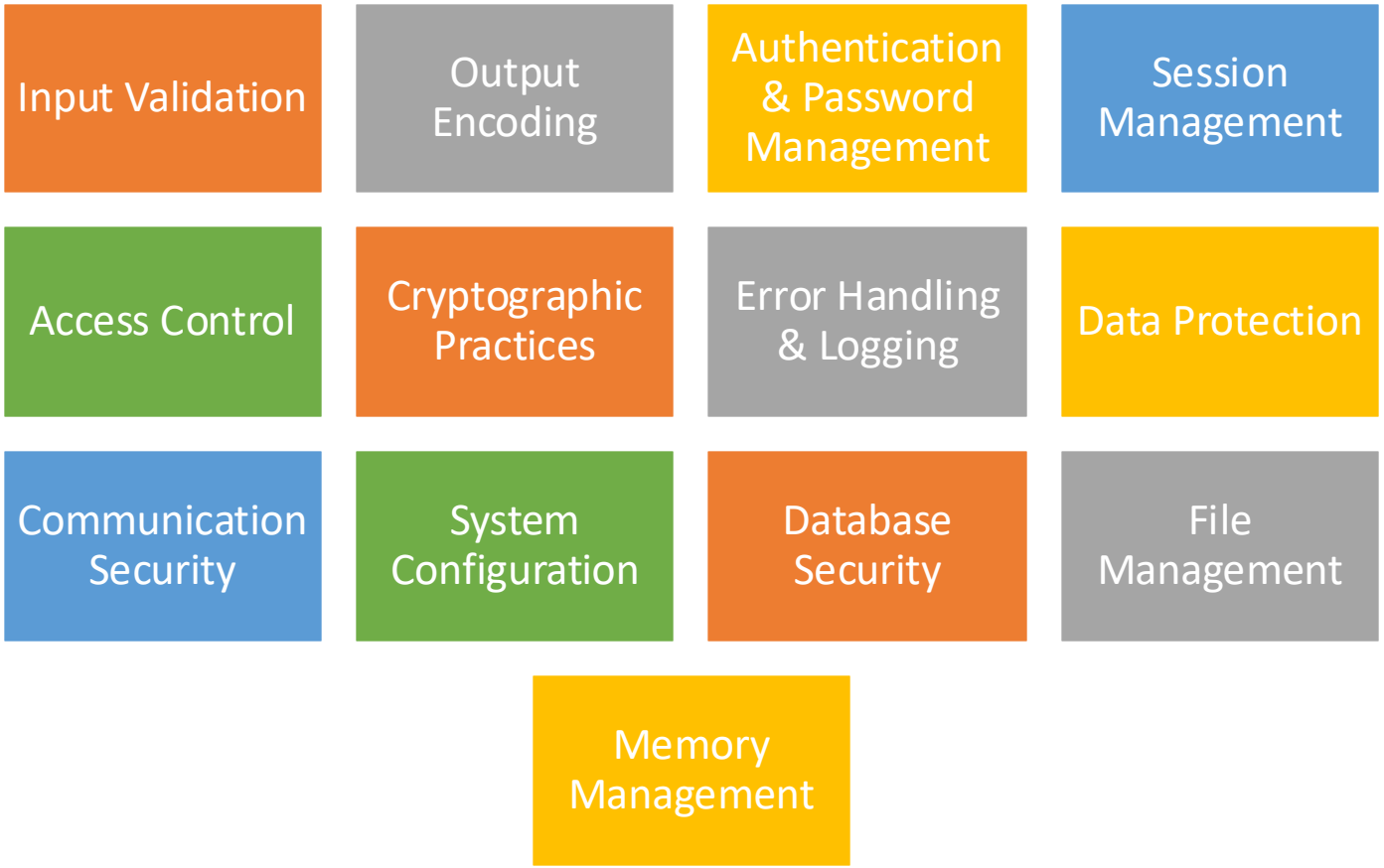
Best Practices trong doanh nghiệp (4)

SEI CERT C++ Coding Standard

Rules

- Rule 01. Declarations and Initialization (DCL)
- Rule 02. Expressions (EXP)
- Rule 03. Integers (INT)
- Rule 04. Containers (CTR)
- Rule 05. Characters and Strings (STR)
- Rule 06. Memory Management (MEM)
- Rule 07. Input Output (FIO)
- Rule 08. Exceptions and Error Handling (ERR)
- Rule 09. Object Oriented Programming (OOP)
- Rule 10. Concurrency (CON)
- Rule 49. Miscellaneous (MSC)

OWASP Secure Coding Checklist



https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/

Secure Code

The CWE Top 25

Rank	ID	Name	Score	KEV Count (CVEs)	Rank Change vs. 2021
1	CWE-787	Out-of-bounds Write	64.20	62	0
2	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.97	2	0
3	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	22.11	7	+3 ▲
4	CWE-20	Improper Input Validation	20.63	20	0
5	CWE-125	Out-of-bounds Read	17.67	1	-2 ▼
6	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	17.53	32	-1 ▼
7	CWE-416	Use After Free	15.50	28	0
8	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.08	19	0
9	CWE-352	Cross-Site Request Forgery (CSRF)	11.53	1	0
10	CWE-434	Unrestricted Upload of File with Dangerous Type	9.56	6	0
11	CWE-476	NULL Pointer Dereference	7.15	0	+4 ▲
12	CWE-502	Deserialization of Untrusted Data	6.68	7	+1 ▲
13	CWE-190	Integer Overflow or Wraparound	6.53	2	-1 ▼
14	CWE-287	Improper Authentication	6.35	4	0
15	CWE-798	Use of Hard-coded Credentials	5.66	0	+1 ▲
16	CWE-862	Missing Authorization	5.53	1	+2 ▲
17	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	5.42	5	+8 ▲
18	CWE-306	Missing Authentication for Critical Function	5.15	6	-7 ▼
19	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	4.85	6	-2 ▼
20	CWE-276	Incorrect Default Permissions	4.84	0	-1 ▼
21	CWE-918	Server-Side Request Forgery (SSRF)	4.27	8	+3 ▲
22	CWE-362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	3.57	6	+11 ▲
23	CWE-400	Uncontrolled Resource Consumption	3.56	2	+4 ▲
24	CWE-611	Improper Restriction of XML External Entity Reference	3.38	0	-1 ▼
25	CWE-94	Improper Control of Generation of Code ('Code Injection')	3.32	4	+3 ▲

https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html

Thiết lập secure coding baselines

Secure code issue – predictable random numbers:

The use of a predictable random number can result in vulnerabilities in the session ID, token, or encryption initialization vector. It's suggested to use `java.security.SecureRandom` *instead of* `java.util.Random`*:*

```
// Vulnerable Code
Random rnd = New Random ();

// Suggested Code
SecureRandom rnd = SecureRandom();
```

- Secure coding baselines là các **yêu cầu secure code tối thiểu** và danh sách checklist để team dự án chuyển sang giai đoạn tiếp theo.
- Cũng là 1 phần trong các điều kiện cần đảm bảo để sản phẩm có thể được phân phối. Tất cả các dự án phải được quét với công cụ quét mã nguồn nhất định trước khi phân phối, ví dụ: kiểm tra các cảnh báo, lỗi.
- Bên cạnh đó, secure coding baselines cần **các công cụ** phát triển có liên quan và **hoạt động training trong thực thế** thay vì chỉ là các rule secure code trong các tài liệu.

Training về Mã nguồn an toàn (1)

Mục đích: thông báo cho đội ngũ phát triển biết về các phương pháp secure code sẽ được áp dụng.

Ở giai đoạn đầu của training, tập trung vào:

- Chuẩn hay secure code baseline là gì?
- Các vấn đề trong secure code thường thấy trong doanh nghiệp?
- Chúng sẽ tác động như thế nào đến công việc của lập trình viên?
- Các tiêu chí để quét mã nguồn an toàn?

→ Thay vì các rule về lập trình an toàn đơn thuần, tốt hơn hết nên đưa ra **các case study** hoặc **ví dụ về mã nguồn có lỗ hổng trong 1 ngữ cảnh nào đó.**

Training về Mã nguồn an toàn (2)

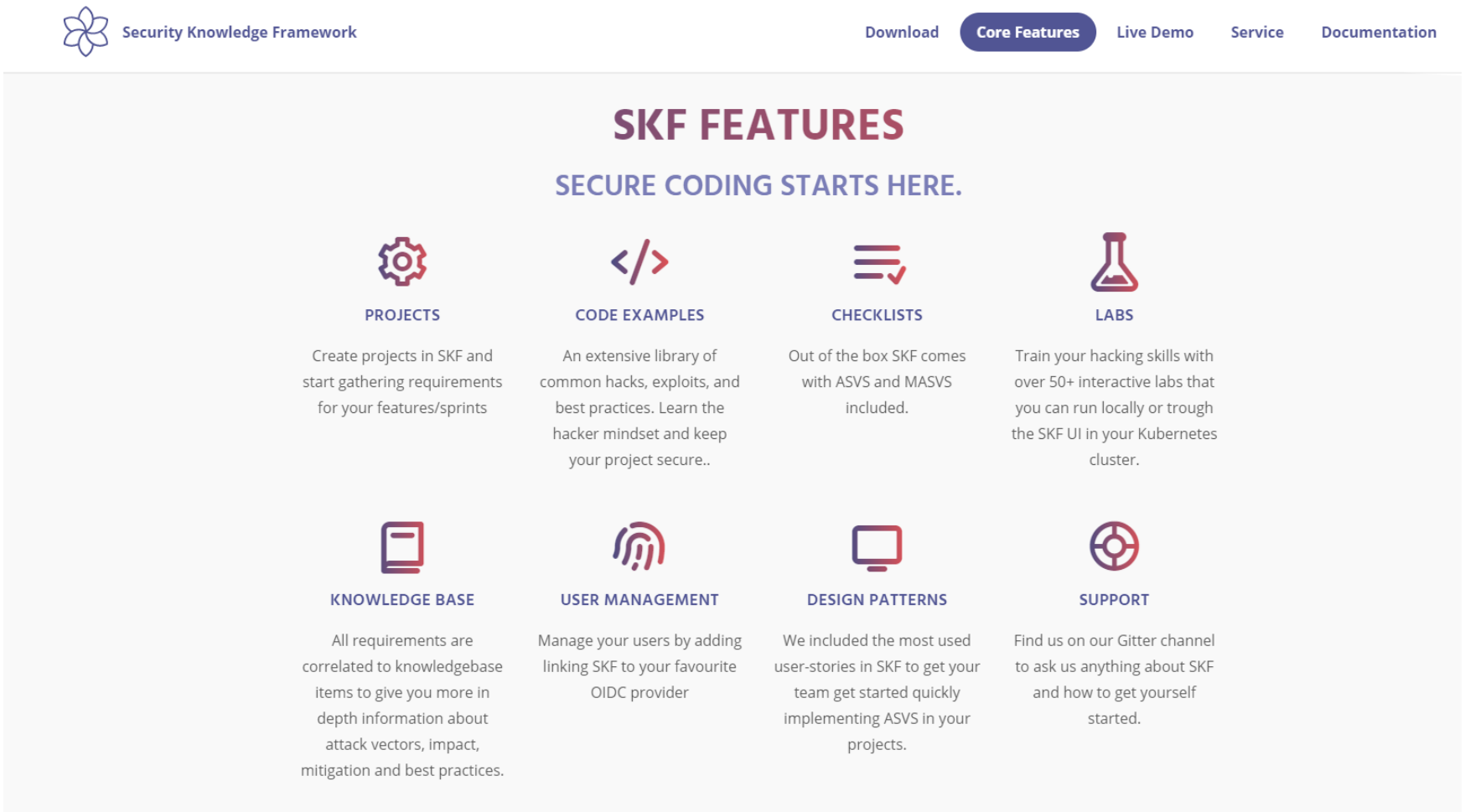
Tài liệu tham khảo với các ví dụ về mã nguồn có lỗ hổng và các phương pháp lập trình an toàn:

- **OWASP Security Knowledge Framework**: ứng dụng web mã nguồn mở, phát triển dựa trên Python-Flask, được thiết kế dùng OWASP Application Security Verification Standard để train về việc viết mã nguồn an toàn.
  Link: https://www.securityknowledgeframework.org/
- **Android Application Secure Design and Secure Coding Guidebook**:
  http://www.jssec.org/dl/android_securecoding_en.pdf
- **Find Security Bugs Patterns for Java:**
  https://find-sec-bugs.github.io/
- **OWASP Teammentor:**
  http://teammentor.github.io/

Training về Mã nguồn an toàn (3)

Dành cho ứng dụng Java: https://find-sec-bugs.github.io/

Training về Mã nguồn an toàn (4)



Dành cho ứng dụng Python-Flask: https://www.securityknowledgeframework.org/

Lựa chọn công cụ đánh giá

- Cần một số công cụ hỗ trợ đánh giá secure code – mã nguồn an toàn.
- Các tiêu chí lựa chọn công cụ:

Tiêu chí	Mô tả
Dễ sử dụng	- Đối tượng sử dụng là các lập trình viên. - Dễ sử dụng = khả năng quét 1 phần mã nguồn, quét các khác biệt, xuất báo cáo, truy vết code ban đầu,...
Chi phí	Nếu là công cụ thương mại, cần cân nhắc chi phí cần bỏ ra cho các licenses
Hỗ trợ ngôn ngữ lập trình	- Hầu hết hỗ trợ C/C++ và Java. - Khảo sát các ngôn ngữ dùng trong các dự án và ưu tiên các ngôn ngữ sẽ được hỗ trợ
Tỉ lệ phát hiện và tỉ lệ dương tính giả	- Tỉ lệ dương tính giả cao thì không tốt, nên tìm công cụ phù hợp nhất với dự án thay vì chọn cái phổ biến nhất. - Có thể đánh giá tỉ lệ phát hiện bằng các dự án có lỗ hổng đã biết.
Cập nhật các rule quét mã	Công cụ cần được cập nhật định kỳ với rule và các bộ quét, đây là ưu điểm của các bản thương mại.

Lựa chọn công cụ đánh giá

Có 2 cách quét mã nguồn:

- 1 – Quét mã nguồn tĩnh với IDE plugin.

Tương tự như công cụ kiểm tra chính tả, cú pháp, giúp lập trình viên học và sửa các vấn đề bảo mật một cách trực quan.

- 2 – Quét mã nguồn hàng ngày để tạo các báo cáo quét hàng ngày.

Lập trình viên xem các báo cáo quét mã hàng ngày để khắc phục hoặc nhận xét các vấn đề bảo mật theo đợt.

Đánh giá các công cụ quét mã nguồn cần đánh giá tỉ lệ phát hiện, tỉ lệ dương tính giả, chi phí, và tính dễ sử dụng cho team phát triển.

Lựa chọn công cụ: Dự án có lỗ hổng

Vulnerable projects	Description	Programming languages
NIST Software	The project provides on-purpose insecure code	Java, C/C++,
Assurance Reference Dataset Project	examples which can be used to test the detection rate of secure code scanning tools	C#, PHP
OWASP Node JS Goat	It's a vulnerable website to practice OWASP top 10 security testing and is built by NodeJS.	Node JS
OWASP WebGoat .Net	It's a vulnerable website to practice OWASP top 10 security testing and is built by .NET.	.NET
OWASP WebGoat PHP	It's a vulnerable website to practice OWASP top 10 security testing and is built by PHP.	PHP
OWASP RailsGoat	It's a vulnerable website to practice OWASP top 10 security testing and is built by Ruby.	Ruby on Rails

- Các dự án mã nguồn có lỗ hổng để đánh giá các công cụ quét mã tĩnh.

Lập trình an toàn & Khai thác lỗ hổng phần mềm

Trường ĐH CNTT TP. HCM