

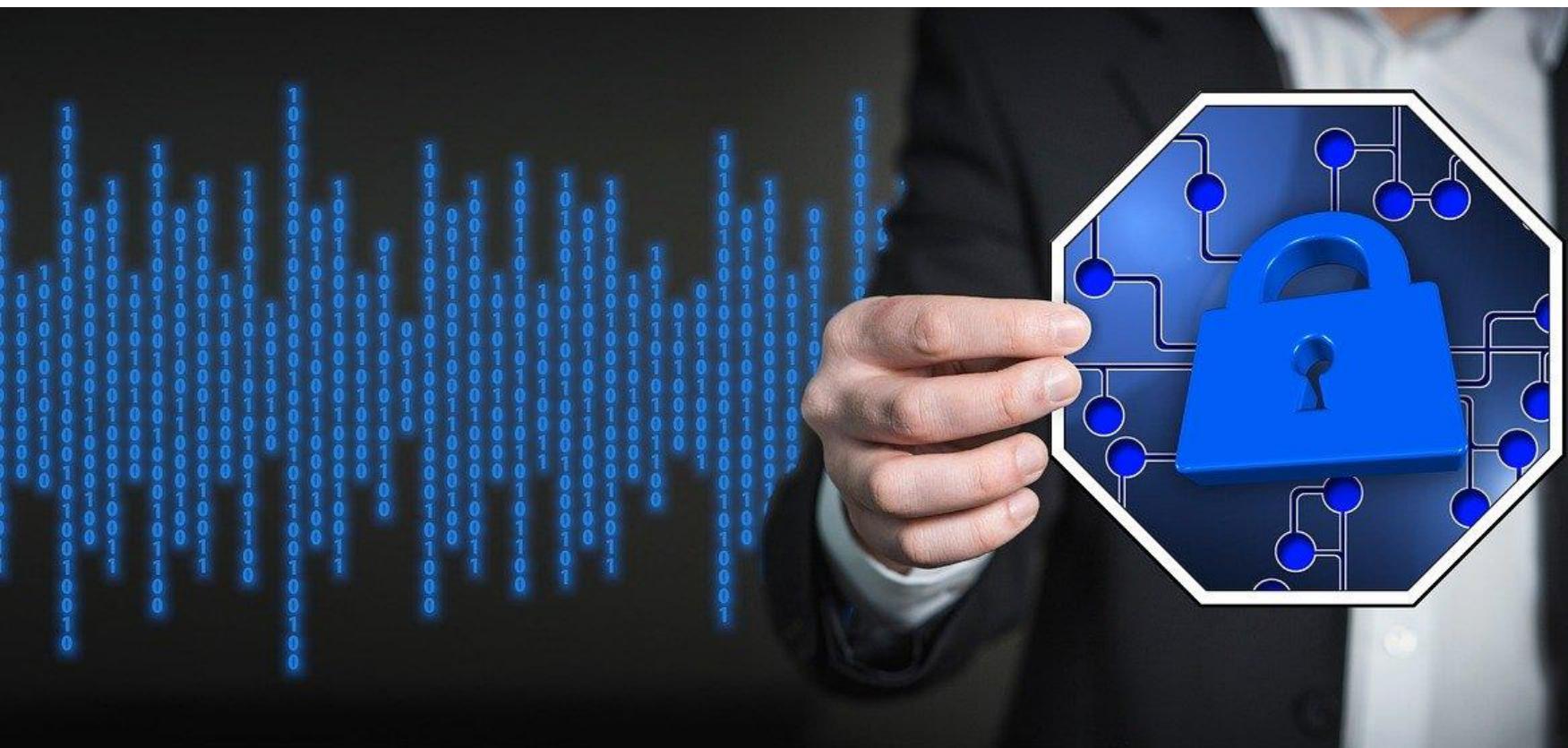


NT521 - Lập trình an toàn & Khai thác lỗ hổng phần mềm



Buổi 04

SSDLC - Qui trình phát triển phần mềm an toàn



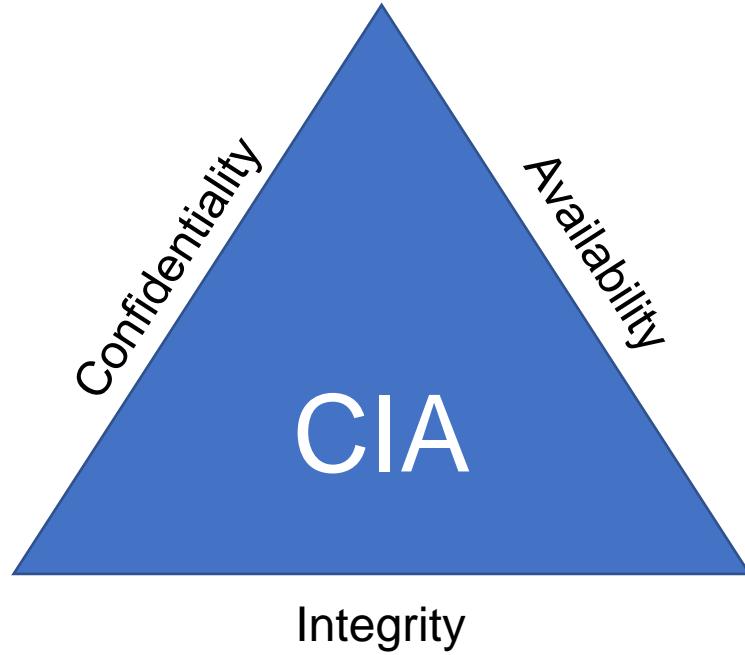
Nội dung



- 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)



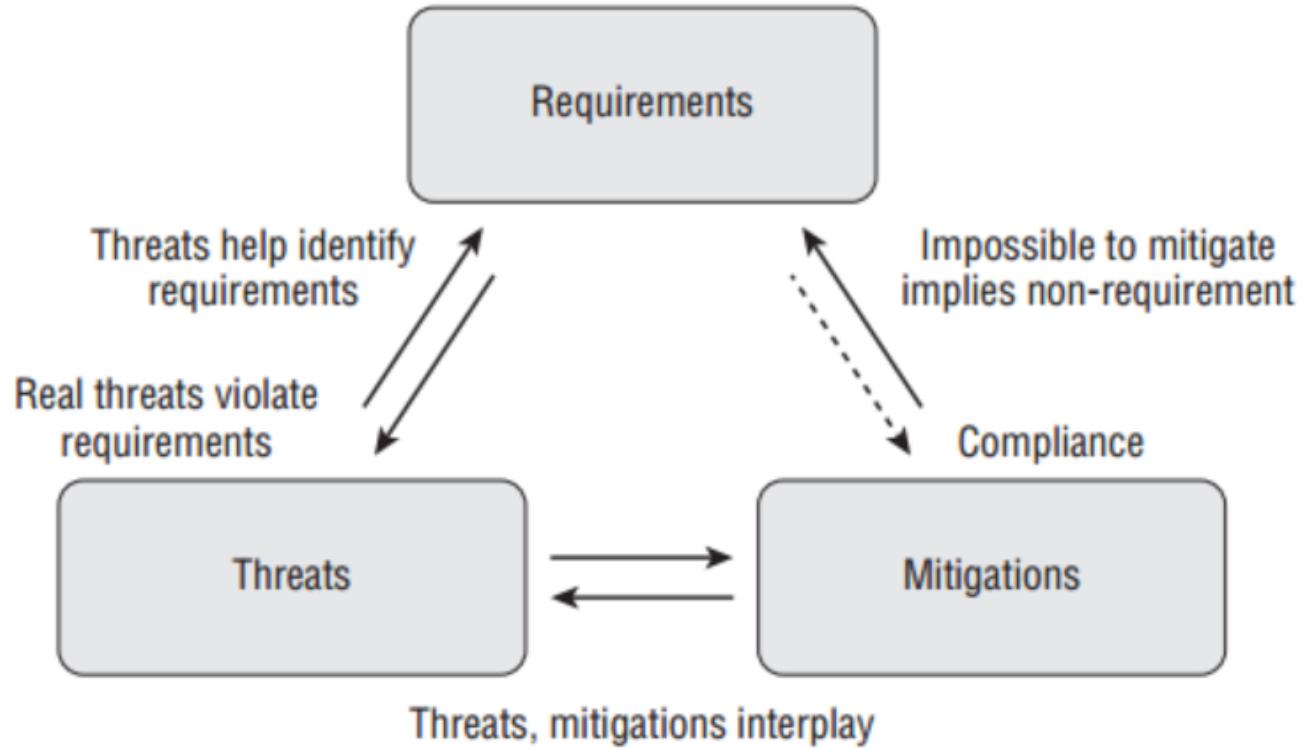
Security: Cơ bản



Bộ ba **CIA** (Bảo mật – Toàn vẹn – Sẵn sàng)
– Lý do team IT Security tồn tại

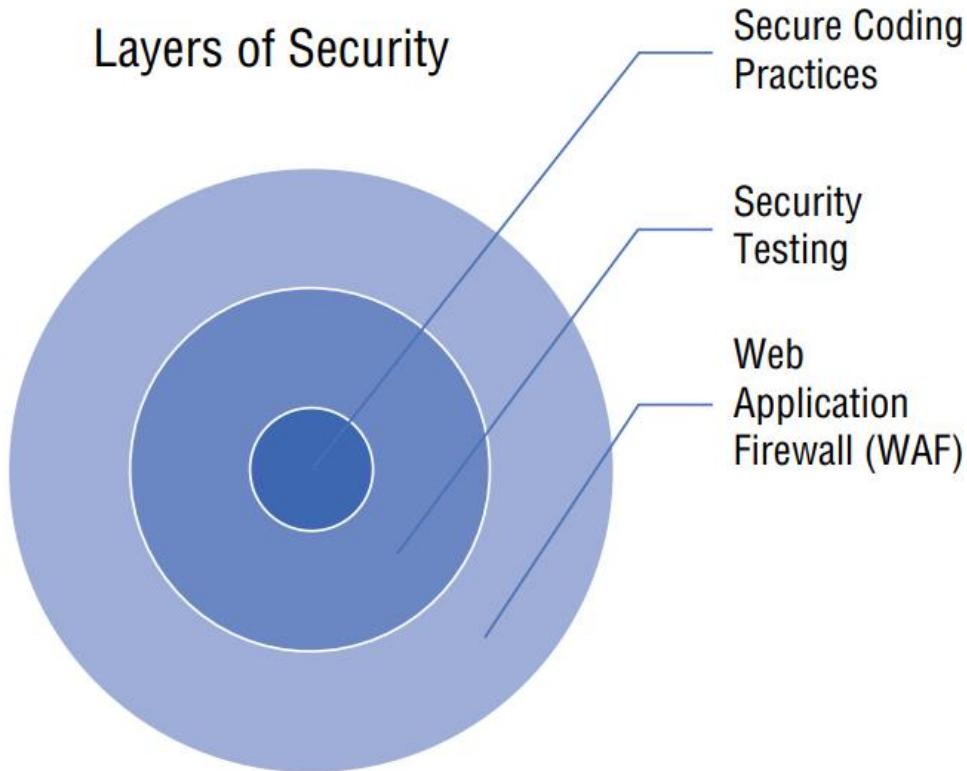


Requirements, Threats, và Mitigations



Defense in Depth

Phòng thủ theo chiều sâu

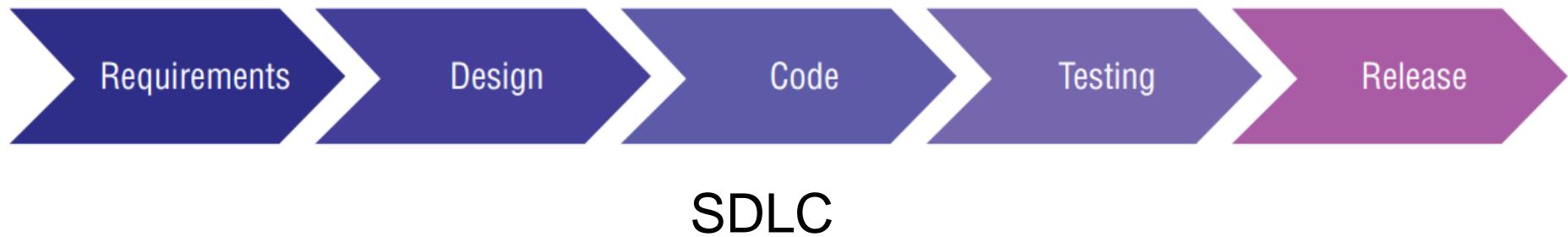


Ví dụ: Web Application Server gồm 3 layer bảo mật – 1 ví dụ của cơ chế phòng thủ theo chiều sâu (defense in depth)



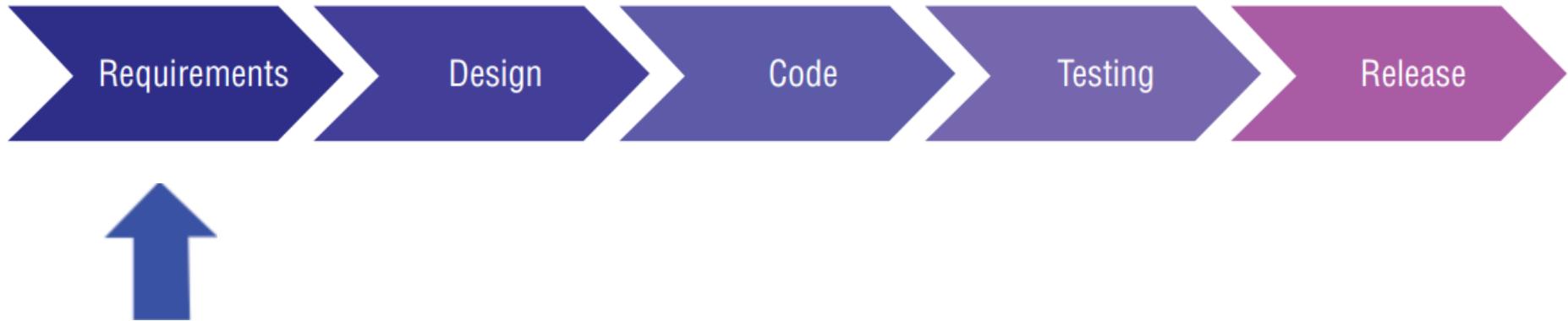


Secure SDLC?





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

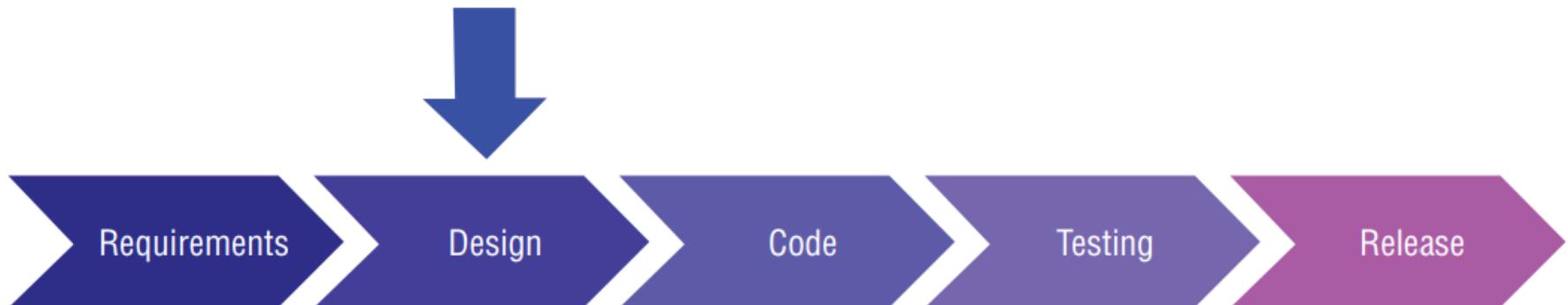




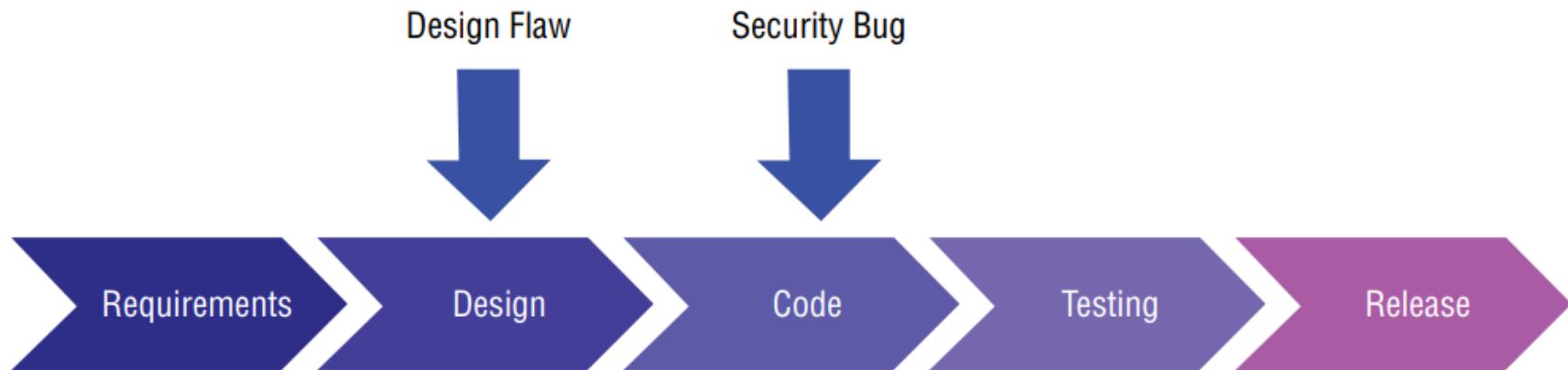
Secure Design

"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 dùng nhập input không hợp lệ.



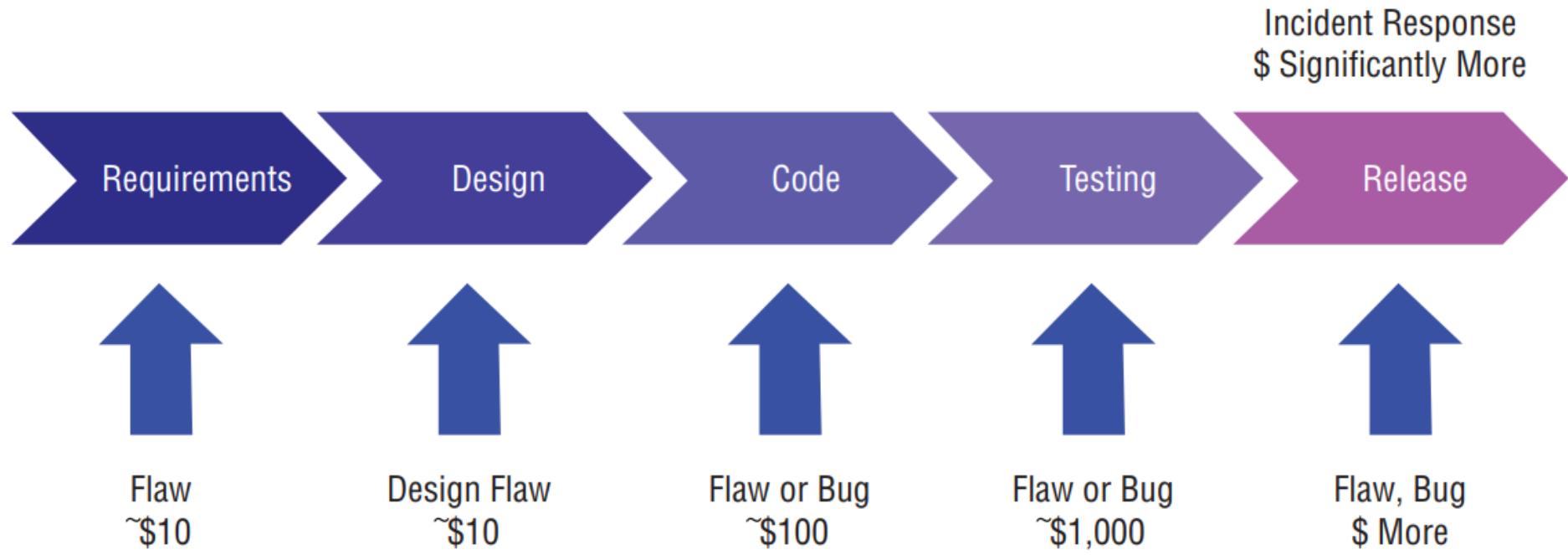
Design Flaw vs. Security Bug



Design Flaw	Security bug
<ul style="list-style-type: none">- 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.	<ul style="list-style-type: none">- 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.

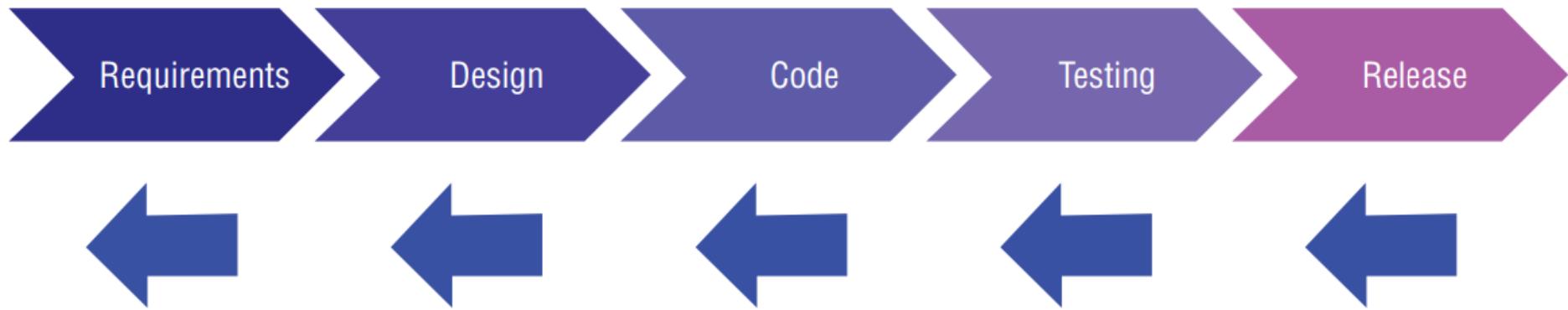


Flaw: Tác động



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.

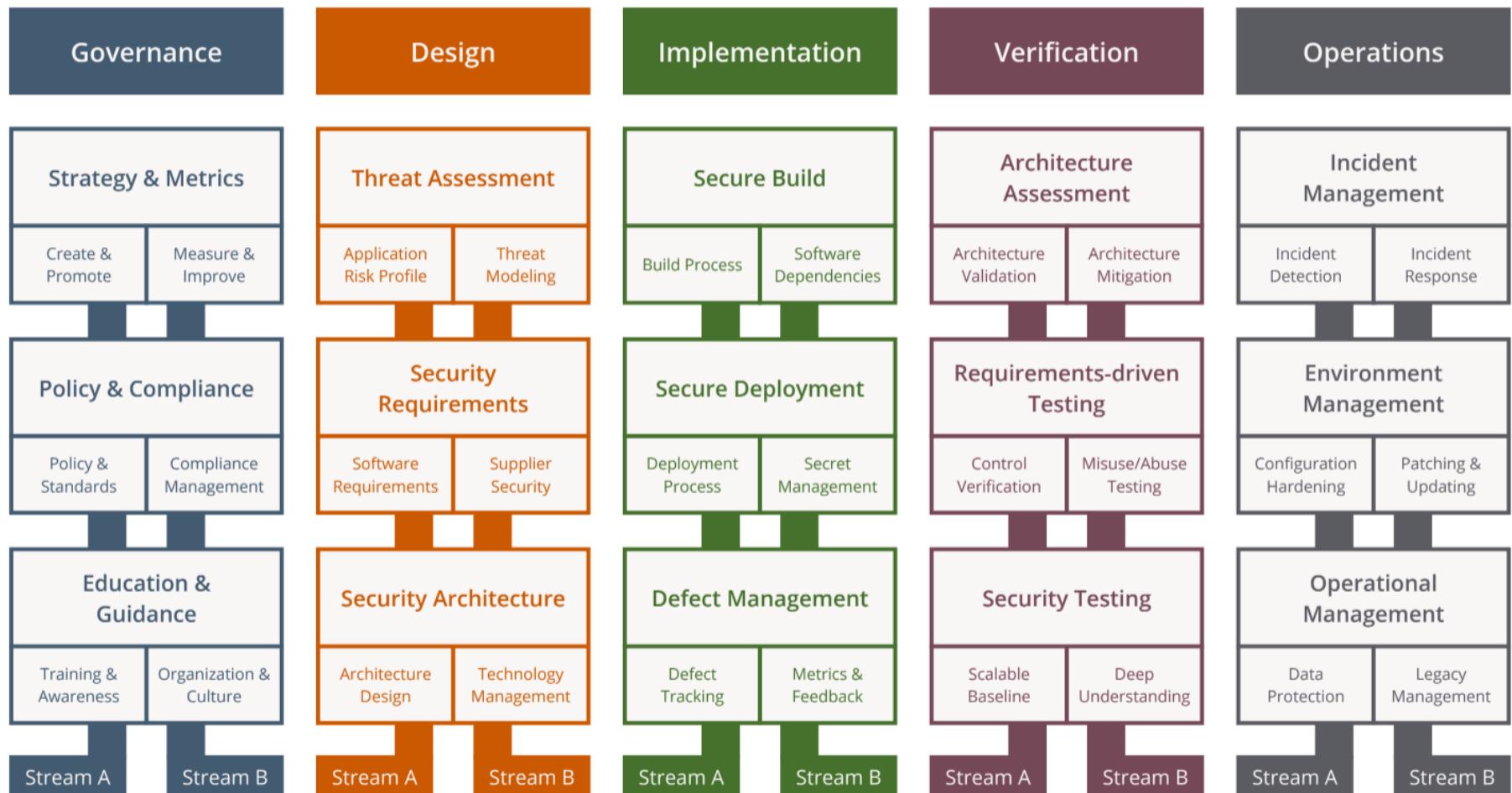


Lỗ hổng phổ biến - Top 10 OWASP

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





<https://owaspSAMM.org/model/>



OWASP SAMM (2)

- **Software Assurance Maturity Model (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.
 - Thiết lập chương trình đảm bảo bảo mật phần mềm cân bằng trong các vòng lặp phát triển.
 - Trình diễn các 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 doanh nghiệp/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

STEPS TO **THREAT MODELING**





Threat Modeling

Mô hình hóa mối đe dọa

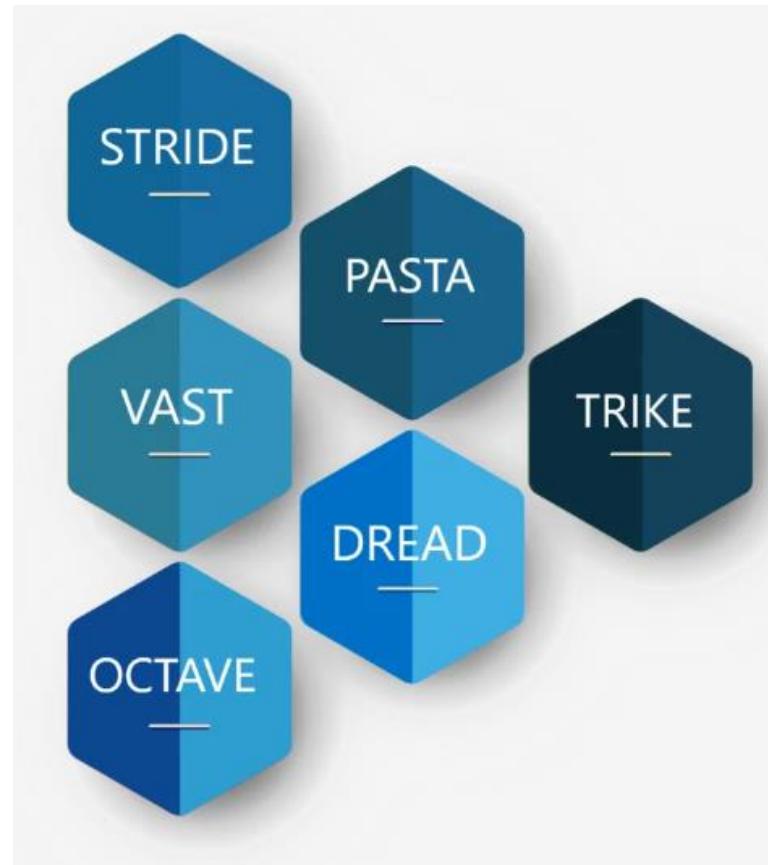
- Đưa ra các câu hỏi và đảm bảo đã cân nhắc các trường hợp xấu có thể xảy ra → concern.
- Đánh giá khả năng, ảnh hưởng → loại bỏ các concern không thể xảy ra hoặc không ảnh hưởng.
- Đánh giá (rate) các concern còn lại ở các mức độ cao, trung bình, thấp
 - Danh sách các threats (các concern đã được xác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

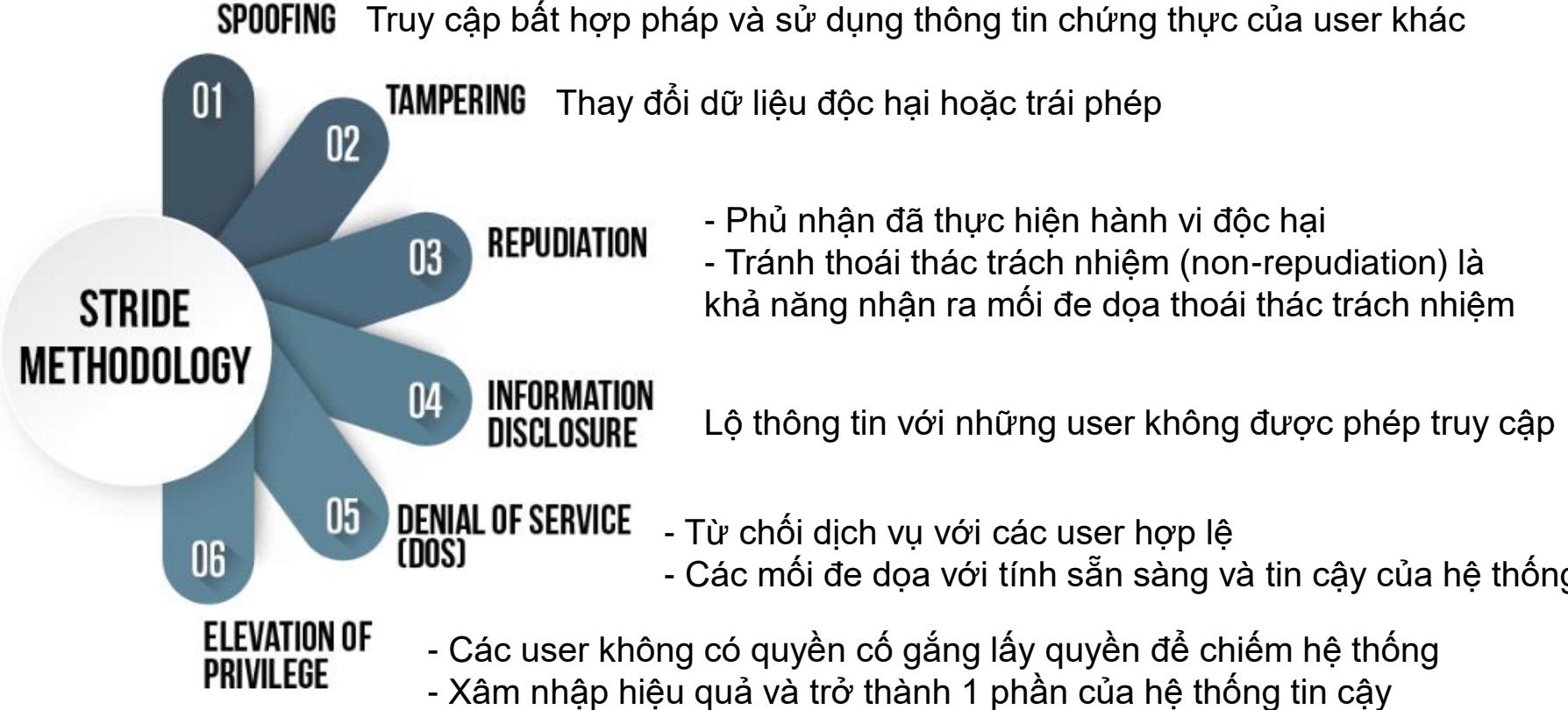
Mô hình hóa mối đe dọa

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

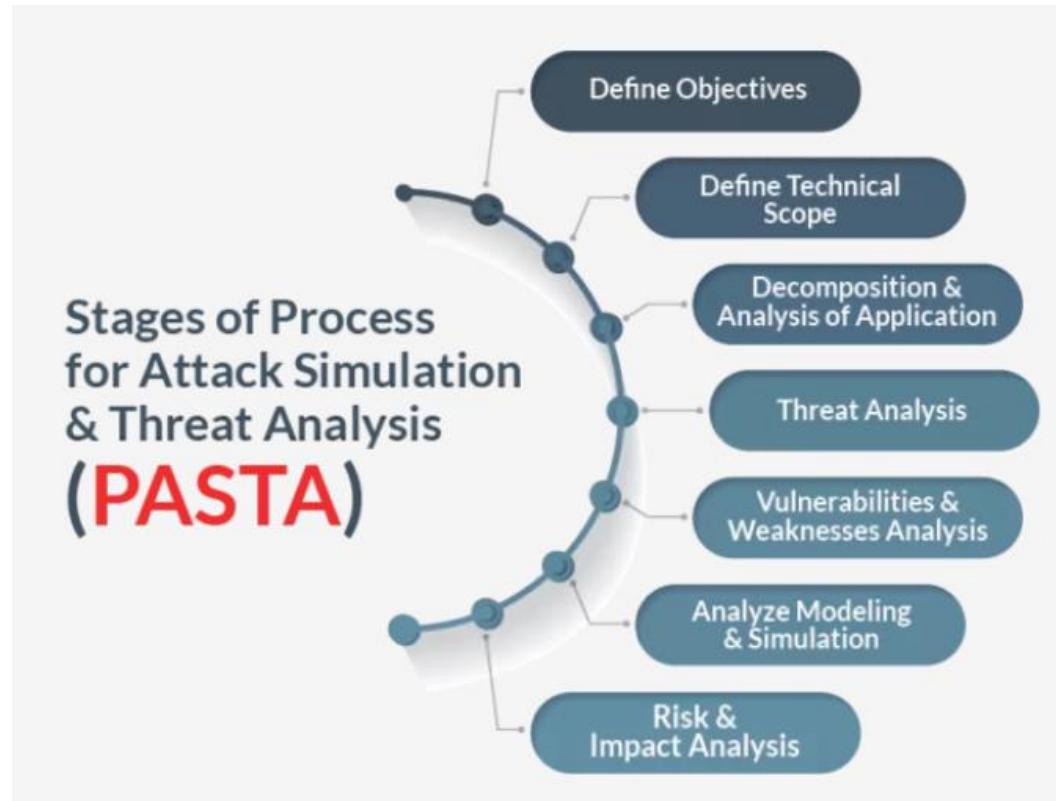


STRIDE

- **STRIDE**, Loren Kohnfelder và Praerit Garg, 1999
 - Xác định các lỗ hổng và mối đe dọa đối với các sản phẩm

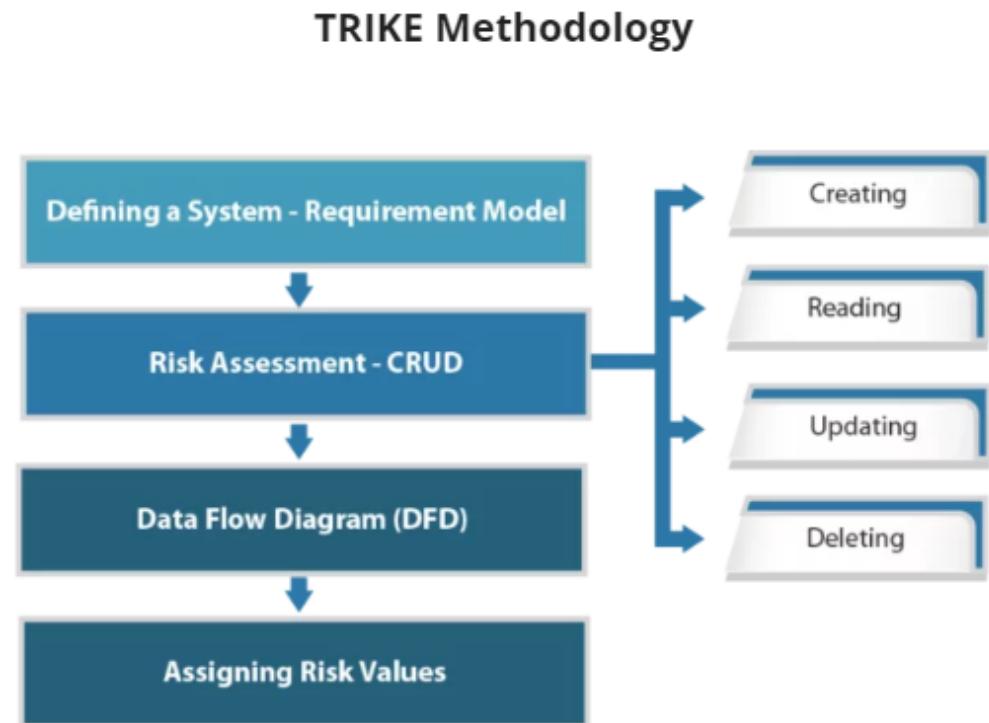


- Process for Attack Simulation and Threat Analysis
- **PASTA:** phương pháp gồm 7 bước để tạo 1 quy trình để **mô phỏng tấn công** vào các ứng dụng IT, **phân tích mối đe dọa, nguồn gốc, rủi ro** gây ra cho tổ chức và **cách giảm thiểu** chúng.
- Mục tiêu: xác định và liệt kê các mối đe dọa, và gán điểm → giúp xác định các biện pháp cần thực hiện để giảm thiểu các mối đe dọa.



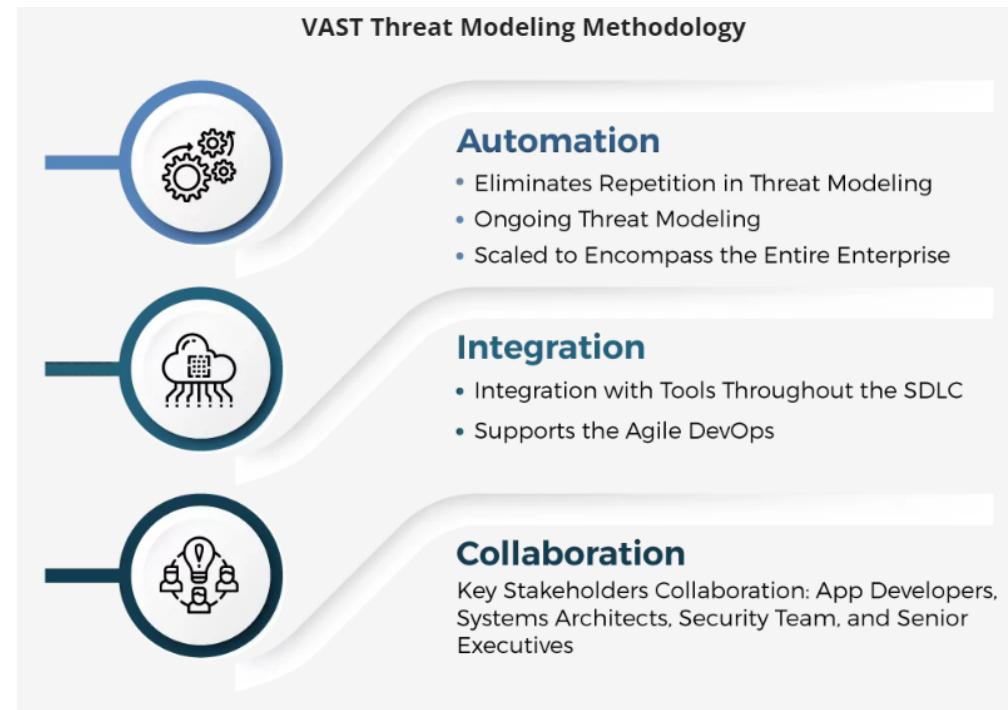
TRIKE

- **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.





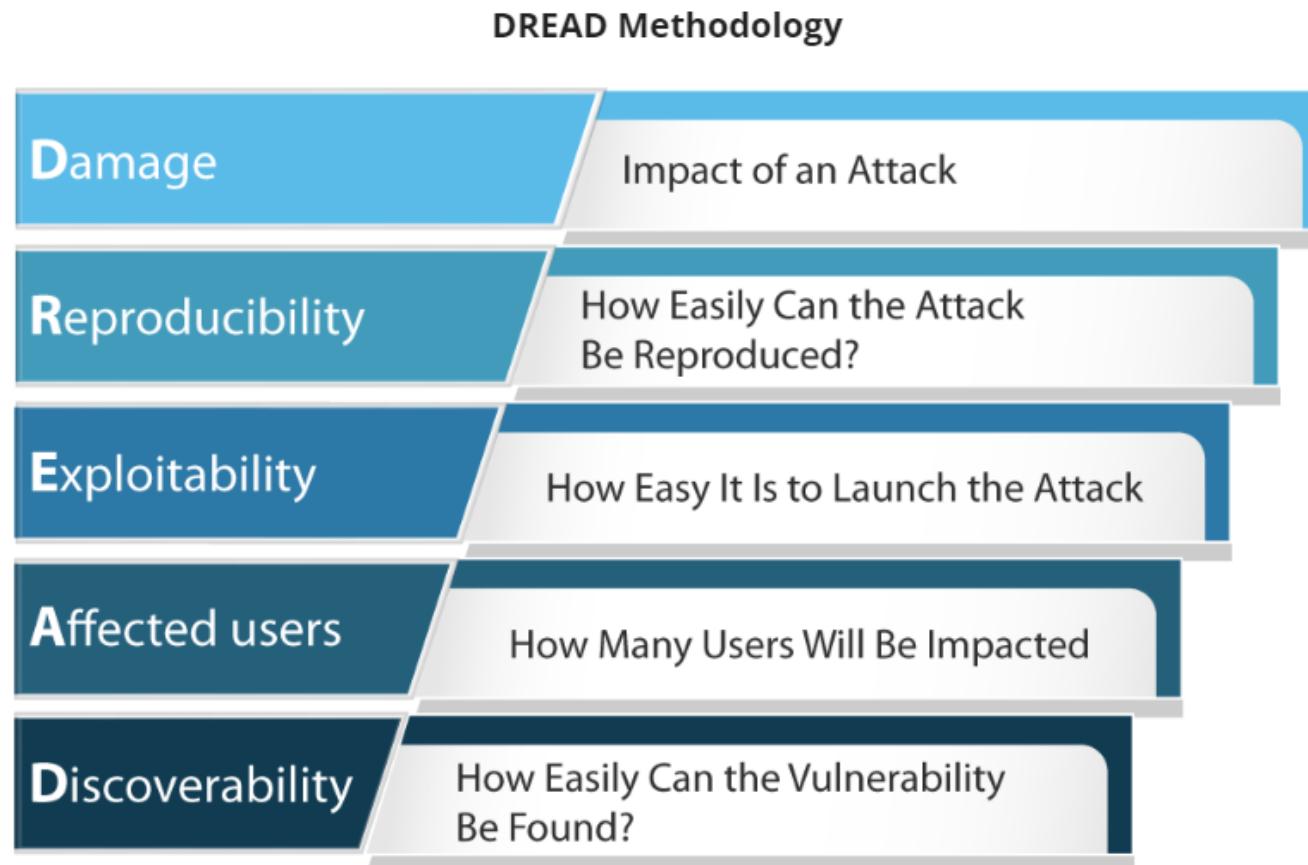
- **VAST** - Visual, Agile, and Simple Threat
- Mô hình hóa mối đe dọa **tự động** bao hàm quy trình phát triển phần mềm trong toàn tổ chức có **tích hợp** phù hợp với các công cụ và **cộng tác** với các bên liên quan như lập trình viên, kiến trúc sư, chuyên gia bảo mật và các lãnh đạo trong tổ chức.





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.



Threat Modeling – Mô hình hóa mối đe dọa

OCTAVE



- OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation)
- 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.





Threat Modeling – Mô hình hóa mối đe dọa

Các công cụ (1)

- Microsoft Threat Modelling Tool

Microsoft Threat Modeling Tool (Preview)

MICROSOFT THREAT MODELING TOOL (PREVIEW)

Threat Model:

Create A Model
Model your system by drawing diagram (s). Make sure you capture important details.

Feedback, Suggestions and Issues

Open A Model

Open an existing model and analyze threats against your system; do not worry, the tool will help you identify them.

Template For New Models

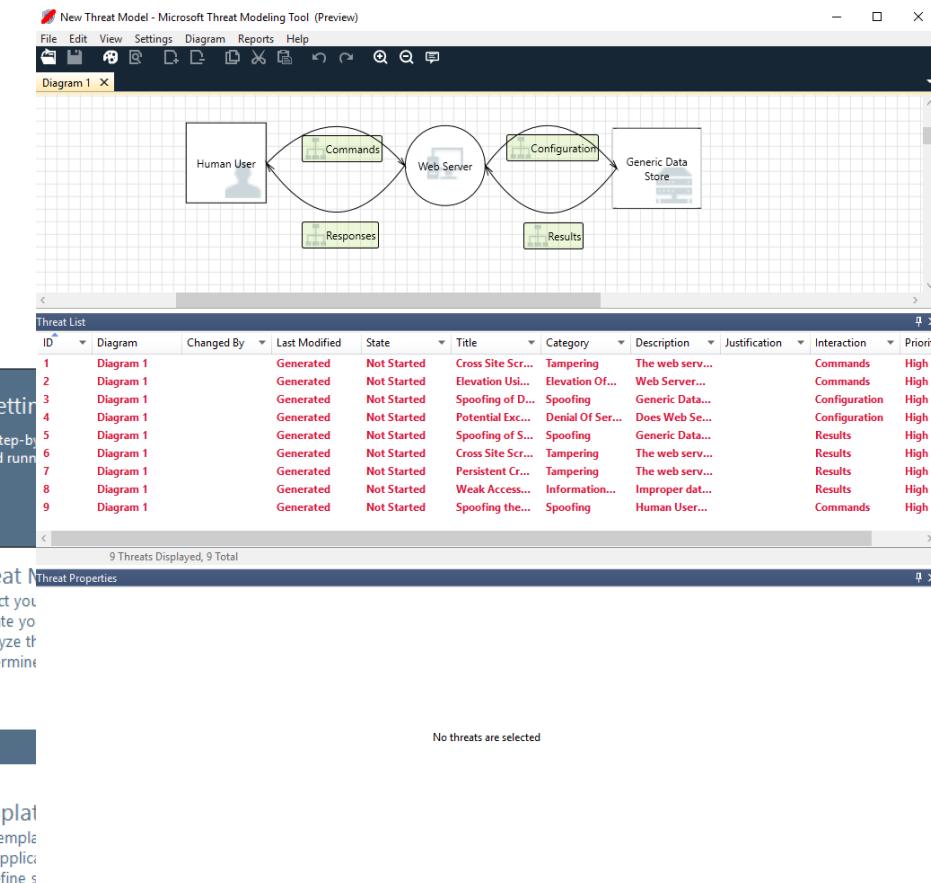
Azure Threat Model Template(1.0.0.20)

Template:

Create New Template
Define stencils, threat types and custom threat properties for your threat model from scratch.

Open Template

Open an existing Template and make modifications to better suit your specific threat analysis.



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/>



Threat Modeling – Mô hình hóa mối đe dọa

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>





Threat Modeling – Mô hình hóa mối đe dọa

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:

Risk Level	Count
critical	1
high	2
elevated	27
medium	46
low	8

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.) from 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.

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.) from 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 SV/nhóm

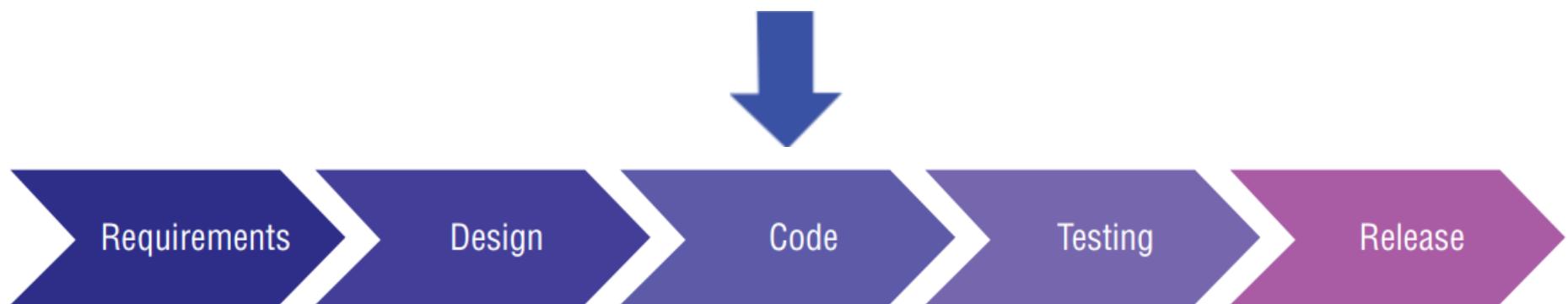
- 1) Kể tên 3 loại thông tin có thể được xem là “bí mật – secret”. Theo bạn, nên lưu trữ các bí mật (secret) của phần mềm ở đâu? Và ứng dụng nên truy cập chúng như thế nào?**
- 2) Giả sử có 1 ứng dụng mobile banking. Kể tên ít nhất 2 mối đe dọa (threat) có thể xảy ra, đánh giá khả năng và thiệt hại của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



Secure Code – Lập trình an toàn

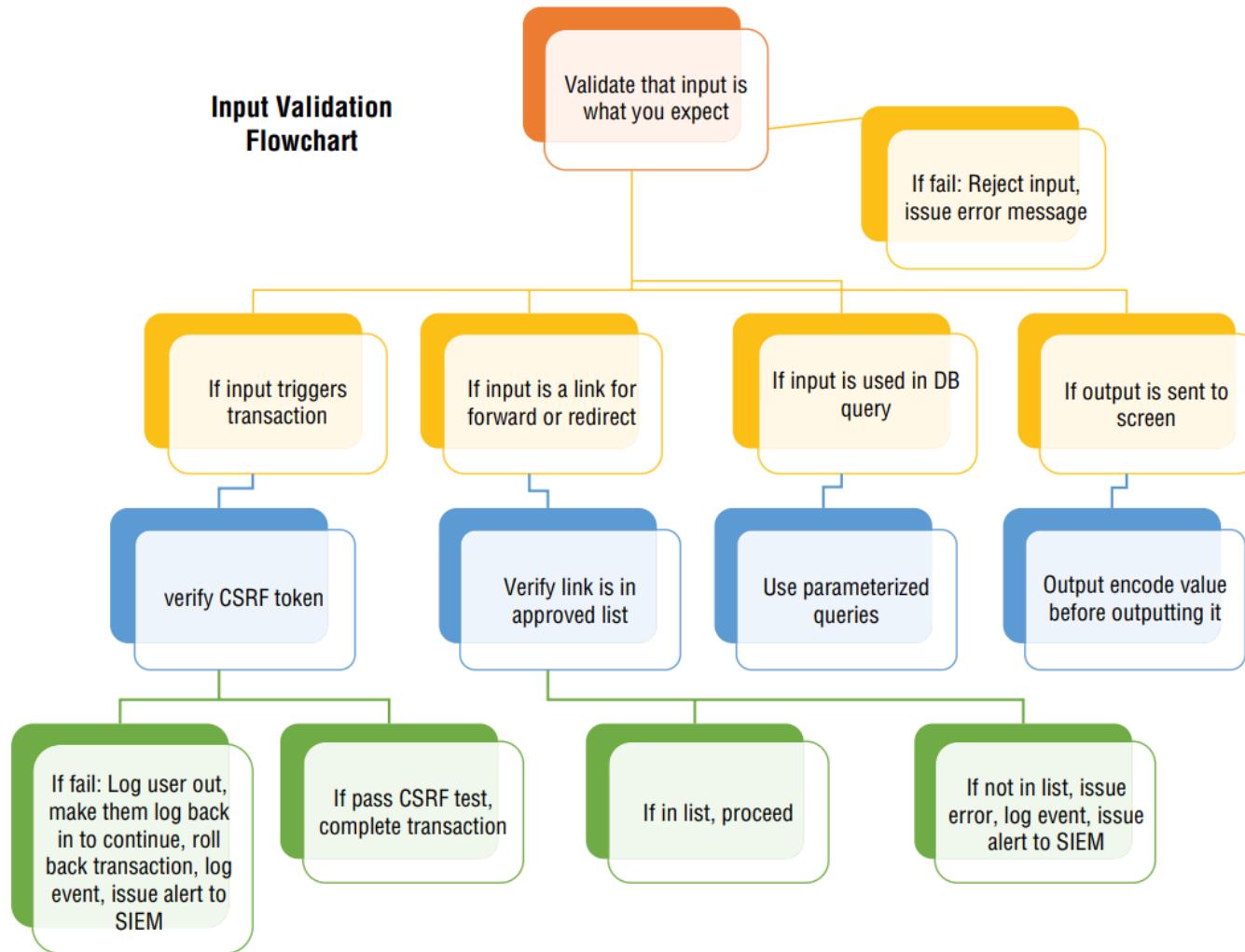
- **Lựa chọn framework và ngôn ngữ lập trình**
- **Ngôn ngữ và framework lập trình:**
 - *Sử dụng framework đang và sẽ được hỗ trợ trong thời gian dài.*
 - *Chọn phiên bản mới nhất hoặc mới-thứ-hai.*
 - *Hỗ trợ security → có đáp ứng được các tính năng cần thiết?*

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

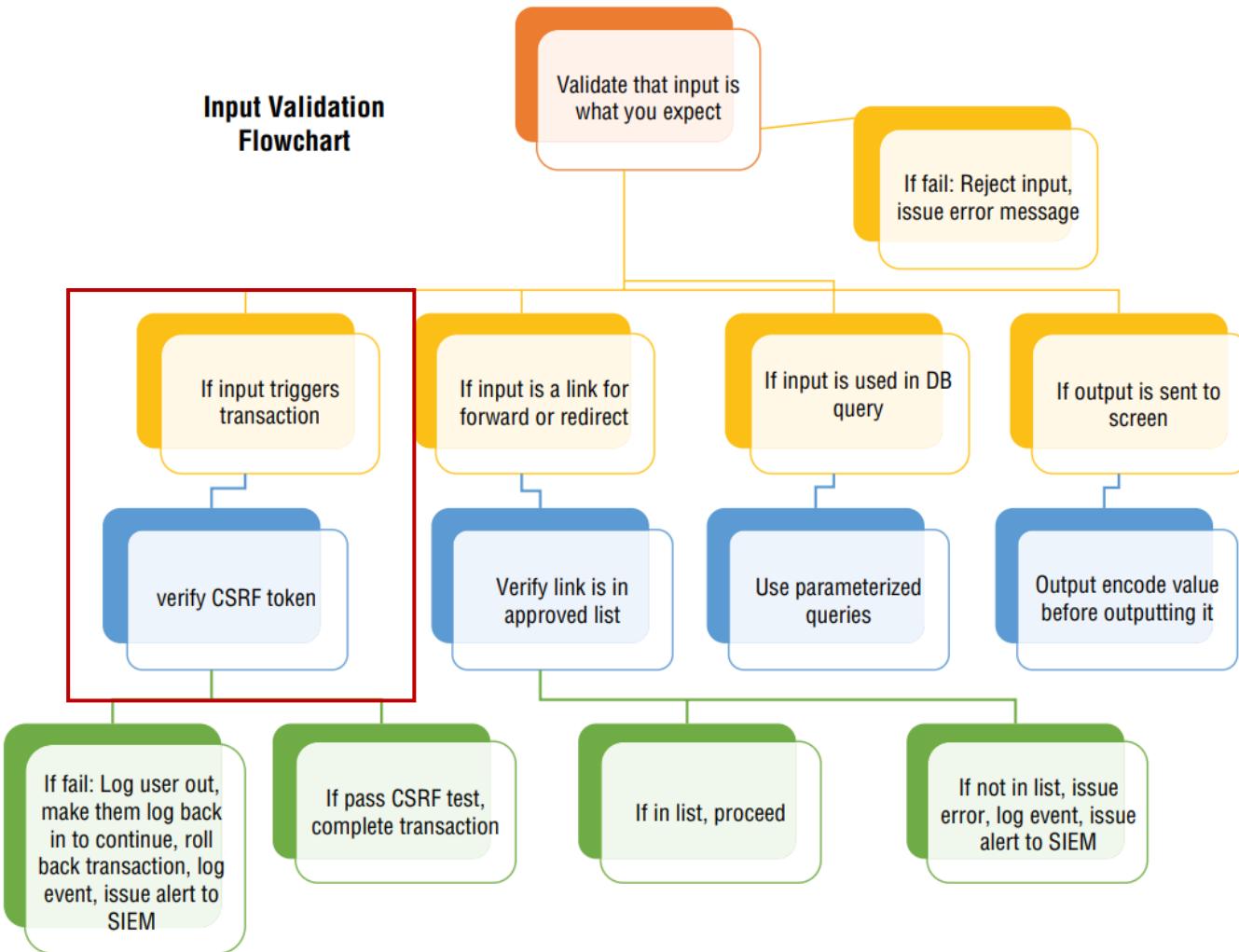


Secure Code

- Untrusted Data
– Dữ liệu không tin cậy



Secure Code

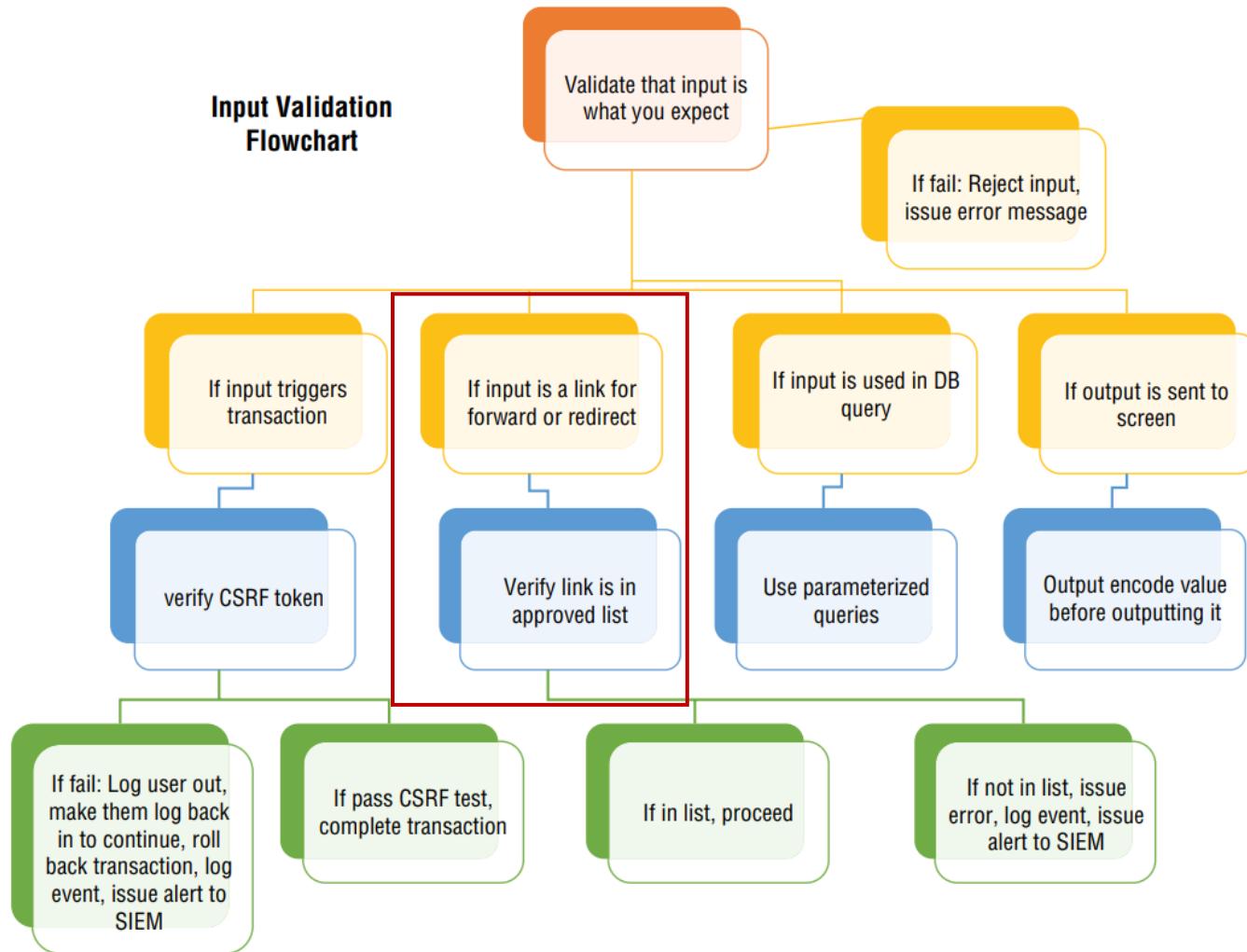


- Untrusted Data
– Dữ liệu không tin cậy

Phòng tránh:
tấn công CSRF



Secure Code

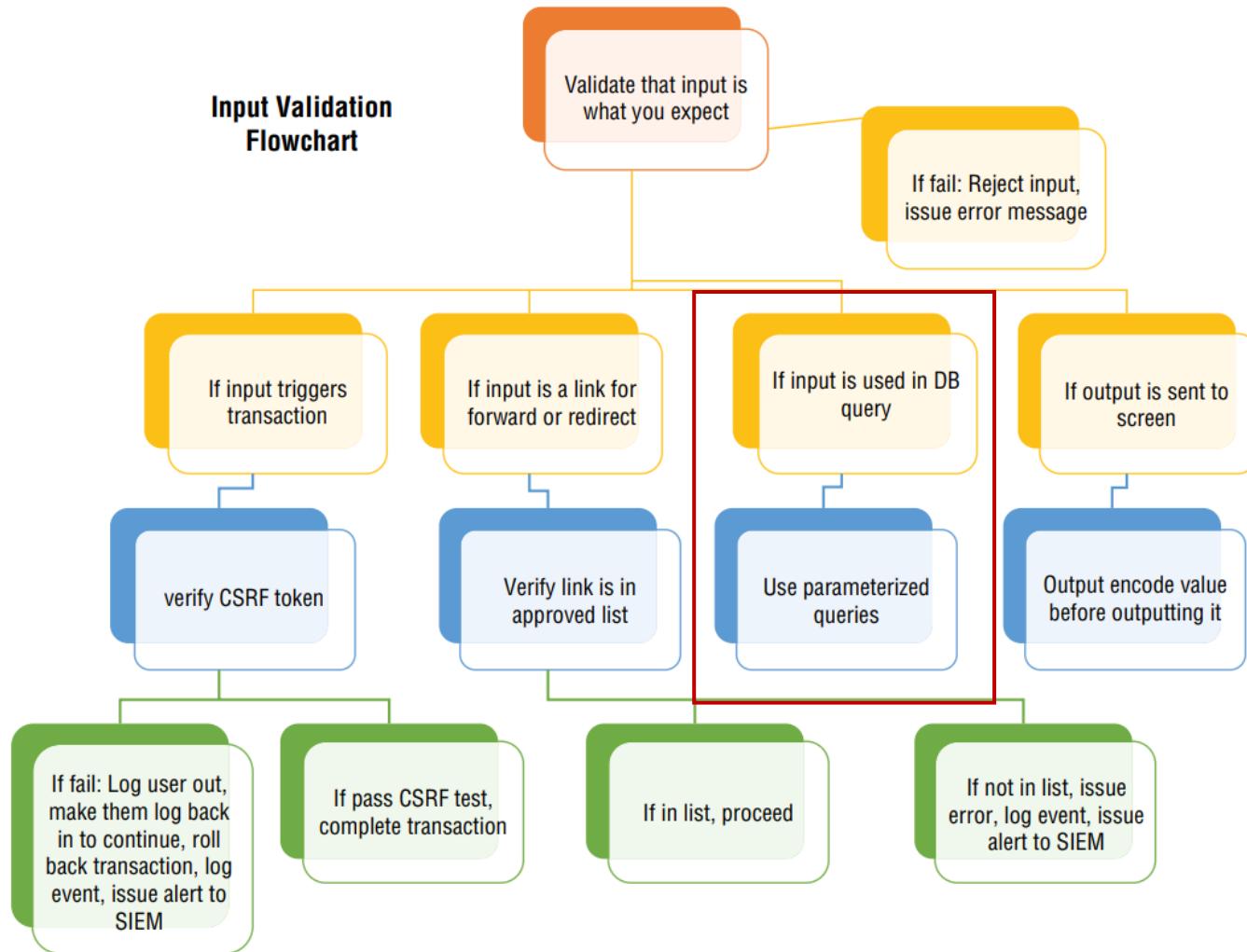


- Untrusted Data
– Dữ liệu không tin cậy

Phòng tránh:
**Reflected XSS/
URL Redirect**



Secure Code

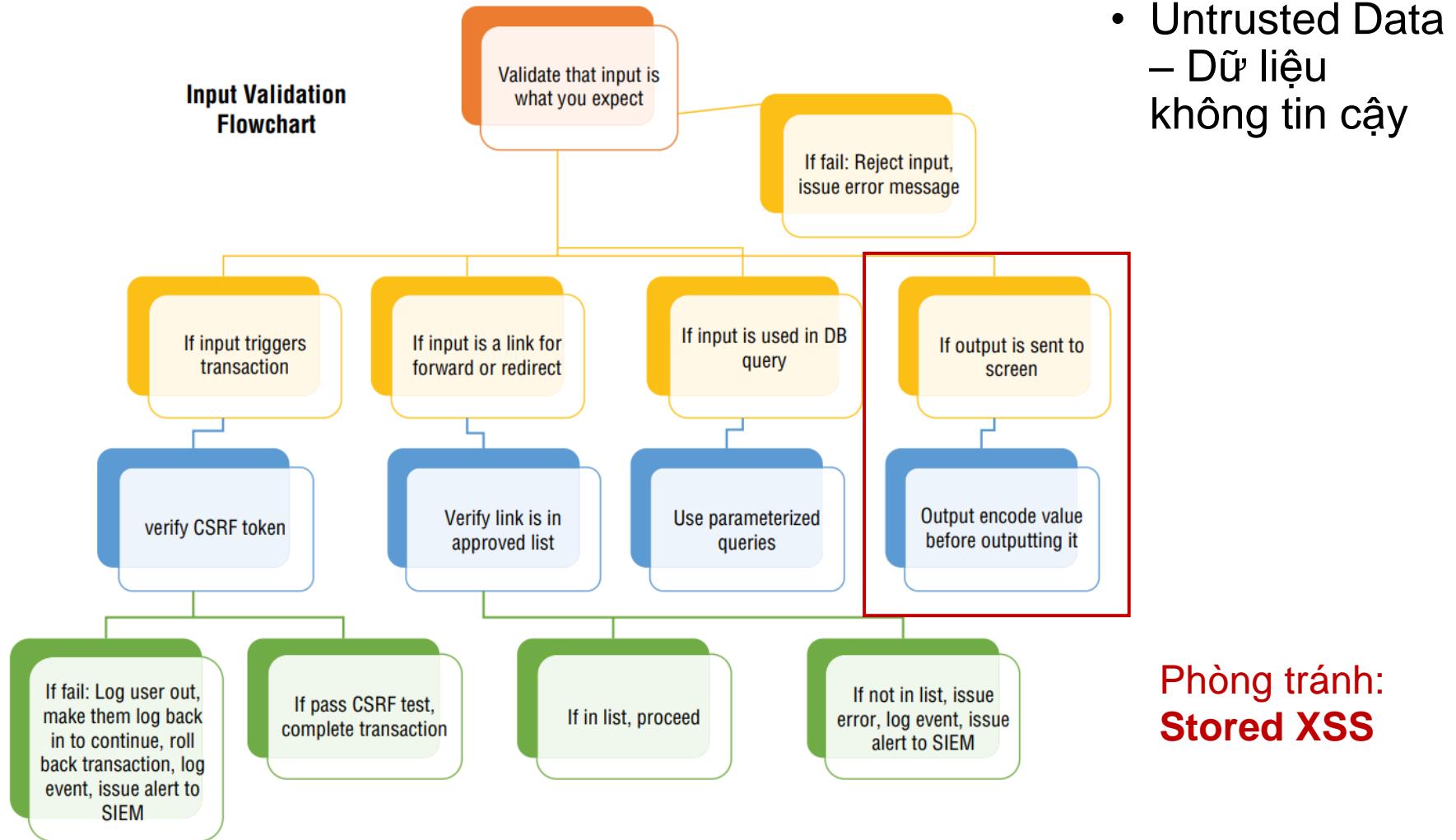


- Untrusted Data – Dữ liệu không tin cậy

Phòng tránh:
SQL Injection



Secure Code





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	<ul style="list-style-type: none">This provides secure coding standards for C, C++, Java, Perl, and Android.
Find Security Bugs	<ul style="list-style-type: none">This provides bug patterns with samples of vulnerable code and solution for Java.
CWE	<ul style="list-style-type: none">This provides vulnerable source code samples from the perspective of common software weaknesses. The coding samples cover C, C++, Java, and PHP.
Android	<ul style="list-style-type: none">Android Application Secure Design and Secure Coding Guidebook



Best Practices trong doanh nghiệp (2)



OWASP SKF	<ul style="list-style-type: none">• OWASP Security Knowledge Framework.• It can be used as an internal security knowledge base, which includes OWASP ASVS and secure coding knowledge.
PHP Security	<ul style="list-style-type: none">• OWASP PHP Security Cheat Sheet
OWASP Code Review	<ul style="list-style-type: none">• OWASP Code Review Project
Apple Secure Coding Guide	<ul style="list-style-type: none">• Apple Secure Coding Guide
Go	<ul style="list-style-type: none">• Secure Coding Practices for GO language
JavaScript	<ul style="list-style-type: none">• JavaScript Secure Coding Practices
Python	<ul style="list-style-type: none">• OWASP Python Security Project





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)





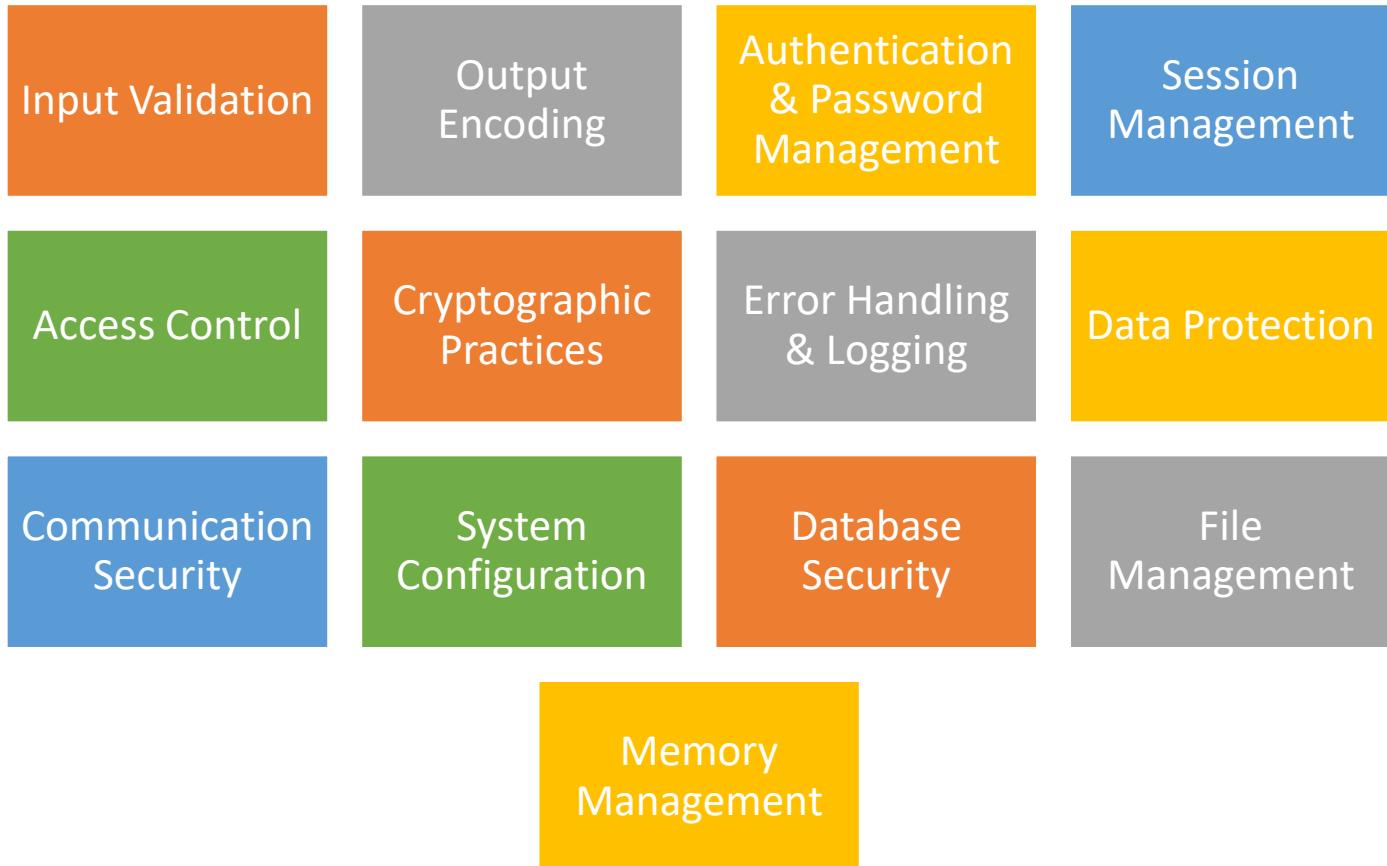
Rules

SEI CERT C++ Coding Standard

- 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/>



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 ngű 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)



Home How To ▾ Bug Patterns Download License

{ } Find Security Bugs

The SpotBugs plugin for security audits of Java web applications.

[Download version 1.11.0](#) [View release notes](#)

(Last updated: October 29th, 2020)

Spread the word:
[Tweet](#)

Follow the project:
[Star](#) 1,706
[Fork](#) 394
[Visit the GitHub project](#)

Features

138 bug patterns

It can detect 138 different vulnerability types with over 820 unique API signatures.

Continuous integration

Can be used with systems such as [Jenkins](#) and [SonarQube](#).

</> Support your frameworks and libraries

Cover popular frameworks including Spring-MVC, Struts, Tapestry and many more.

✓ OWASP TOP 10 and CWE coverage

Extensive references are given for each bug patterns with references to OWASP Top 10 and CWE.

Integrate with your IDE

Plugins are available for [Eclipse](#), [IntelliJ](#), [Android Studio](#) and [NetBeans](#). Command line integration is available with [Ant](#) and [Maven](#).

Open for contributions

The project is open-source and is [open for contributions](#).

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



Training về Mã nguồn an toàn (4)



Security Knowledge Framework

Download

Core Features

Live Demo

Service

Documentation

SKF FEATURES

SECURE CODING STARTS HERE.



PROJECTS

Create projects in SKF and start gathering requirements for your features/sprints



CODE EXAMPLES

An extensive library of common hacks, exploits, and best practices. Learn the hacker mindset and keep your project secure..



CHECKLISTS

Out of the box SKF comes with ASVS and MASVS included.



LABS

Train your hacking skills with over 50+ interactive labs that you can run locally or through the SKF UI in your Kubernetes cluster.



KNOWLEDGE BASE

All requirements are correlated to knowledgebase items to give you more in depth information about attack vectors, impact, mitigation and best practices.



USER MANAGEMENT

Manage your users by adding linking SKF to your favourite OIDC provider



DESIGN PATTERNS

We included the most used user-stories in SKF to get your team get started quickly implementing ASVS in your projects.



SUPPORT

Find us on our Gitter channel to ask us anything about SKF and how to get yourself started.

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



Lựa chọn công cụ

- 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	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - 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

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.



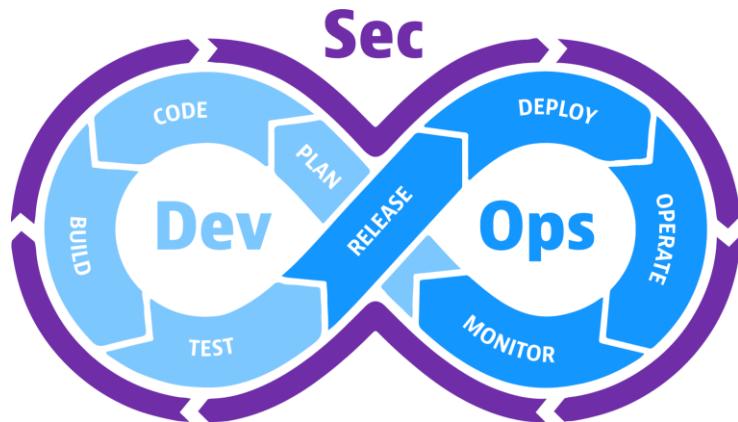
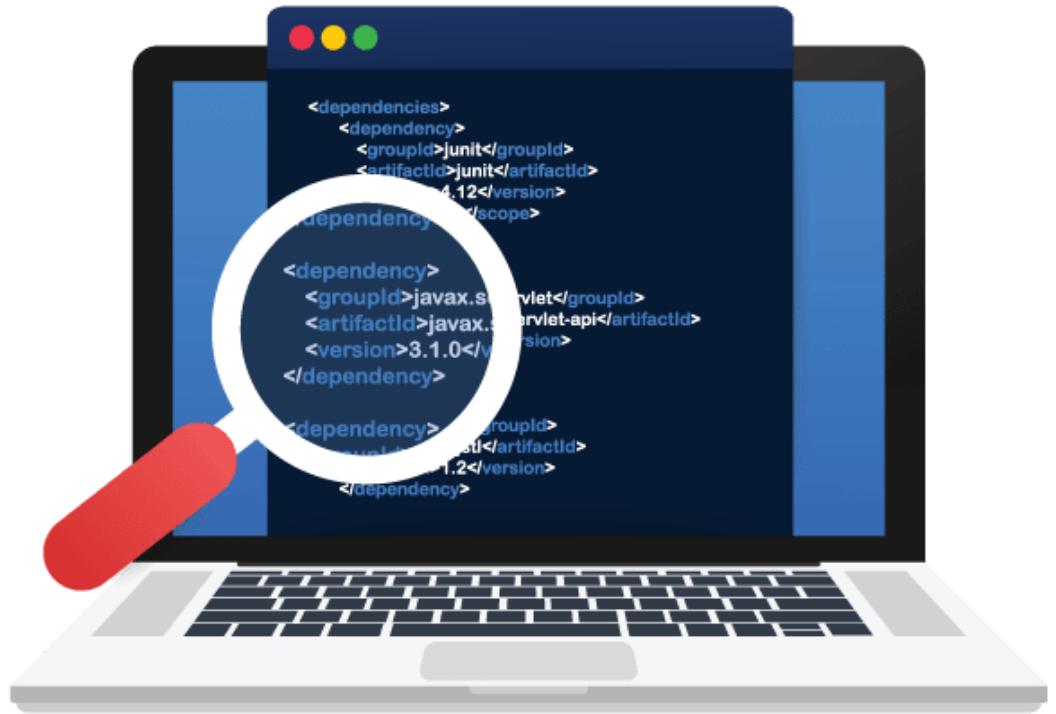
Lựa chọn công cụ: Dự án có lỗ hổng

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.



Phân tích mã nguồn: SAST vs DAST



Static application security testing (SAST)

Dynamic Application Security Testing (DAST)



Công cụ phân tích mã nguồn

Phân tích mã nguồn tĩnh là gì?

- Phân tích mã nguồn tĩnh, hay Phân tích mã nguồn là phương pháp thực hiện trên mã nguồn “**tĩnh**” (**không thực thi**) của phần mềm với các công cụ phân tích mã nguồn tĩnh để phát hiện các lỗ hổng tiềm ẩn.
- Các công cụ phân tích mã nguồn tĩnh kiểm tra toàn bộ mã nguồn để tìm các **lỗ hổng nhất định** cũng như **việc tuân thủ các tiêu chuẩn** lập trình khác nhau.



Công cụ phân tích mã nguồn

Vì sao phải phân tích mã nguồn (tĩnh)?

- Xem xét mã nguồn trước khi thực thi.
- Phân tích nhanh hơn so với phân tích động.
- Có thể tự động hóa quản lý chất lượng mã nguồn.
- Có thể tự động hóa việc tìm bug ở giai đoạn đầu.
- Có thể tự động hóa việc tìm các vấn đề bảo mật ở giai đoạn đầu.
- Một số công cụ phân tích mã nguồn tĩnh đã được tích hợp sẵn trong các IDE (ví dụ Pycharm sử dụng pep8).



https://owasp.org/www-community/Source_Code_Analysis_Tools

Công cụ phân tích mã nguồn



Một số công cụ mã nguồn mở:

- DeepSource
- SonarQube
- Codacy
- DeepScan
- Veracode
- Embold
- Reshift
-

https://owasp.org/www-community/Source_Code_Analysis_Tools



Công cụ phân tích mã nguồn

■ SonarQube

The screenshot shows two code editor panes and a central panel displaying analysis results. The top pane contains Java code with several red annotations (numbered 1 through 6) indicating security vulnerabilities. The bottom pane shows a snippet of SQL code with a red annotation (number 6) for a SQL injection vulnerability. A large callout box highlights a specific issue:

Refactor this code to not construct SQL queries directly from tainted user-controlled data.

Vulnerability +6

Servlet.java

- 1 source: taint value is propagated
- 2 taint value is propagated
- 3 taint value is propagated
- 4 taint value is propagated
- 5 taint value is propagated

SQLInjectionVulnerability.java

- 6 sink: taint value is propagated

alt + ↑ ↓ to navigate issue locations

Refactor this code to not construct SQL queries directly from tainted user-controlled data. [See Rule] 3 months ago L18 %

Vulnerability ⚠ Blocker ⚡ Open Not assigned 30min effort cert, cwe, owasp-a1, sans-top25-inse...

A SonarQube screenshot showing errors





Secure Coding Scanning tool

Tools	Background and key characteristics of the scanning tool		
Retire.JS	<ul style="list-style-type: none"> Detection of vulnerable JavaScript libraries, such as jQuery, AngularJS, Node, and so on. It provides the command line, grunt plugin, and also OWASP ZAP plugin for integration scanning. 	PHPMD	<ul style="list-style-type: none"> PHP Mess Detector is a PHP source code scanner.
Clang Static Analyzer	This provides standalone command line analysis for C, C++, and Objective C.	DawnScanner	Security scanner for Ruby Web applications.
Flawfinder	A simple C/C++ code scanning tool. It's a Python command line scanning tool and can be easily customized based on the needs.	SpotBugs	<ul style="list-style-type: none"> This provides a standalone GUI and command line. SpotBugs can also be used as an Eclipse plugin. It's the successor of FindBugs.
DREK	<ul style="list-style-type: none"> This acts like GREP to search specific security issue by regular expressions, but it can generate scanning results in PDF or HTML format. It's easy to extend any scanning rules by regular expressions. It can be used to scan any programming languages. 	CPP Check	This is a static code analysis tool for C/C++.
Pylint	Pylint is a source code checker for the Python programming language.	Mobile Security Framework (MobSF)	apps. A developer can just upload the APK to the MSF, and the MSF will do all the analysis automatically.
		Clang Static Analyzer	This is a code analysis tool for C/C++ and Objective C.
		ESLint	<ul style="list-style-type: none"> This provides command-line code scanning with JavaScript. Refer here for the secure code scanning rules: https://eslint.org/docs/rules/.





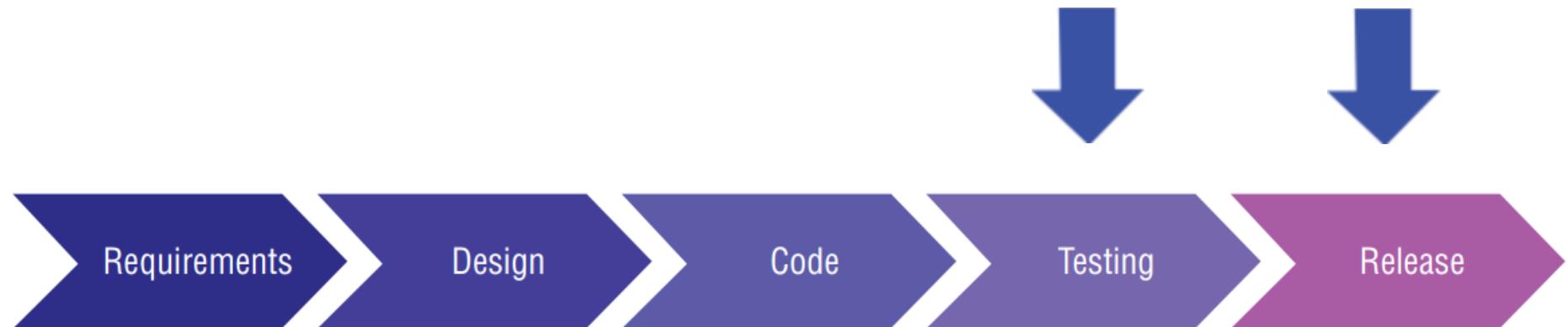
Secure Coding Scanning tool

JSHint	This is for JavaScript code scanning, and also provides command line tools by NodeJS.	PMD	This is a source code analyzer for Java and JavaScript. It's mainly for common programming flaws.
Infer	This is a static code analyzer for Java, C/C++, and Objective C, provided by Facebook.	Graudit	This is a simple script to find potential security issues by using GREP to search for specific code patterns. The signatures database templates provide clues for what to look for.
Phan	Phan is a static analyzer for PHP.	SonarQube	This provides support for more than 20 languages and can integrate with CI frameworks. It is also UI-friendly for quality code scanning results.
PHP Security Checker	This checks PHP project dependencies for known security issues.	Brakeman	Static analysis security scanner for Ruby on Rails.
OWASP Dependency check	This supports a wide range of programming frameworks and checks the disclosed vulnerabilities with updated NVD data feeds. The tool can run as a command line or via integration with Jenkins.	bandit	Security analysis for Python source code.
VisualCodeGrepper (VCG)	VCG is a language-independent scanning tool. The scanning rules can also be easily customized by regular expressions. There are also default rules for commonly banned APIs. It provides a GUI and command line to scan any piece of source code.	Error Prone	Error Prone detects potential Java errors during compile time.
		Dawn	Dawn is a static analysis security scanner for Ruby web applications.





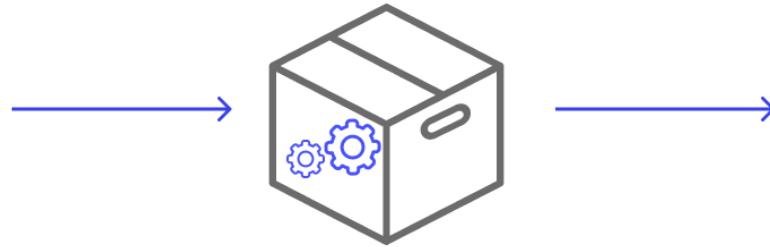
Secure Testing & Deployment



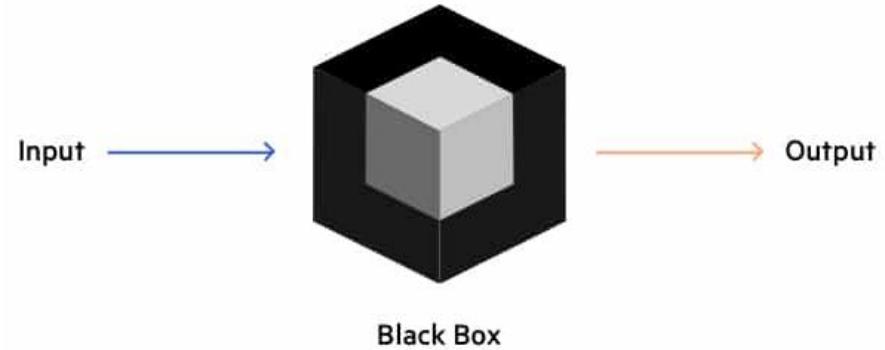
Kiểm thử hộp trắng vs Kiểm thử hộp đen



White Box Testing

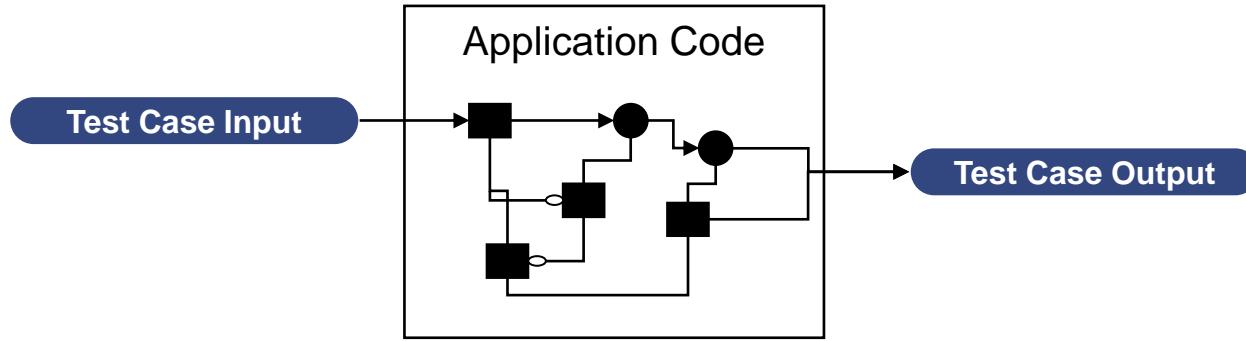


Black Box Testing



Kiểm thử hộp trắng

WHITE BOX TESTING APPROACH

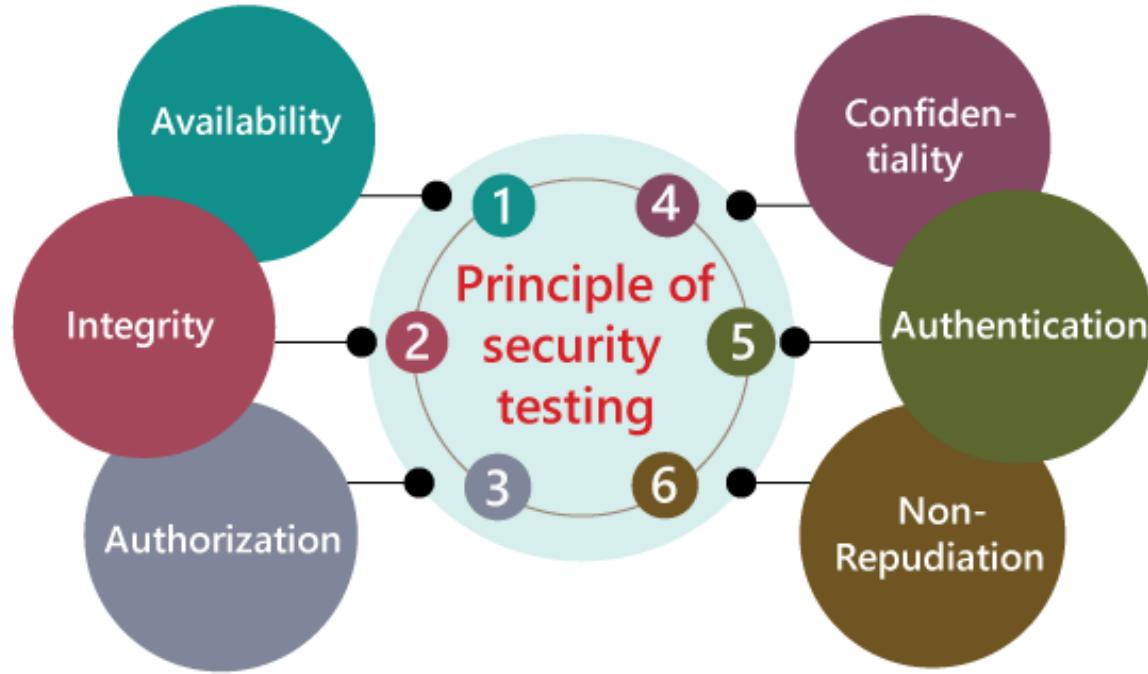




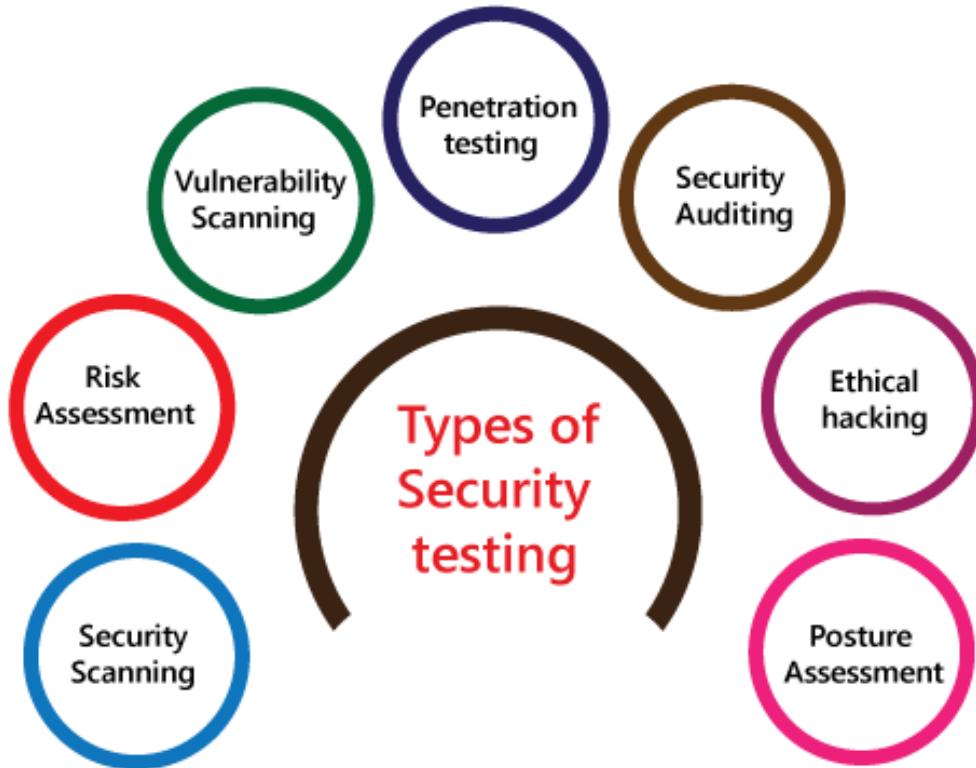
- Kiểm thử mã nguồn
- Kiểm thử ứng dụng
- Kiểm thử hạ tầng
- Kiểm thử cơ sở dữ liệu
- Kiểm thử APIs và các dịch vụ Web
- Kiểm thử tích hợp
- Kiểm thử mạng
- ...Triển khai



Kiểm thử an toàn...

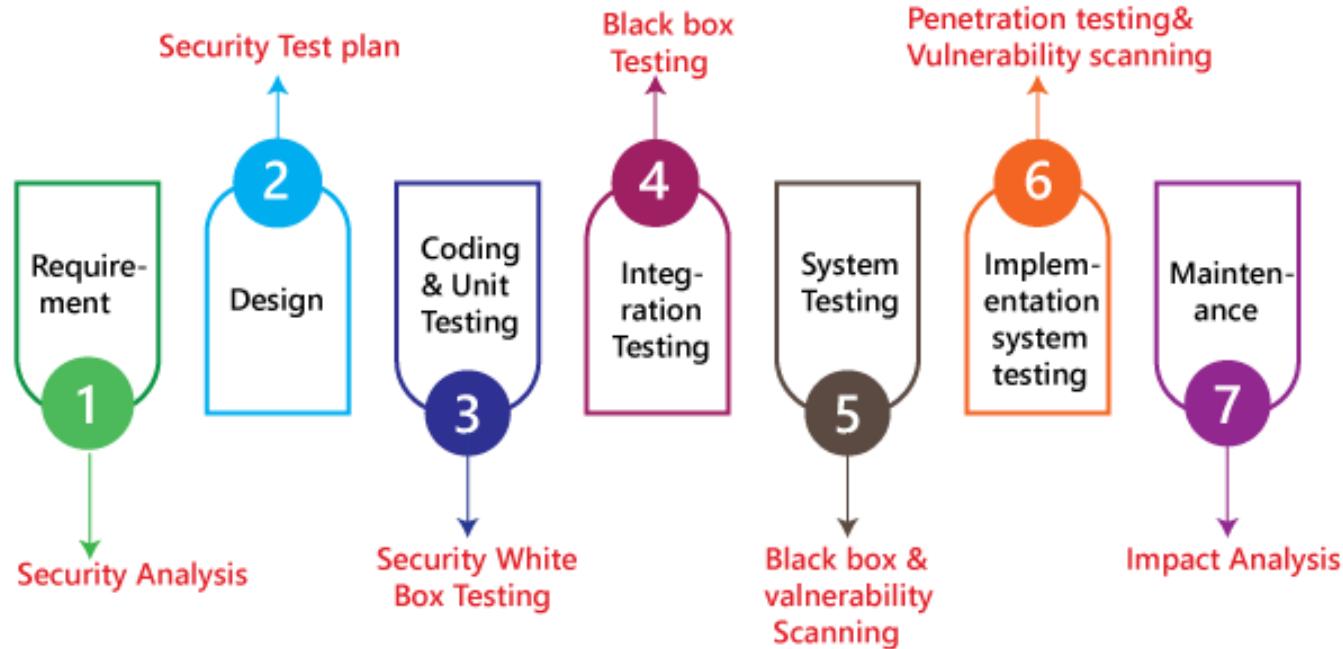


Kiểm thử an toàn...



Kiểm thử an toàn với SDLC

Security Testing along with SDLC



Kiểm thử an toàn...





Bộ khung phát triển phần mềm an toàn

- Bộ khung phát triển phần mềm An toàn
 - Secure Software Development Framework (SSDF):
 - Được đề xuất bởi NIST, Hoa Kỳ, nhằm mục đích đưa các khuyến nghị cho việc giảm thiểu rủi ro của các lỗ hổng phần mềm.
 - Toàn văn tài liệu:

<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218.pdf>

<https://csrc.nist.gov/Projects/ssdf>

NIST Special Publication 800-218

Secure Software Development Framework (SSDF) Version 1.1:

Recommendations for Mitigating the Risk of Software Vulnerabilities

Murugiah Souppaya
Computer Security Division
Information Technology Laboratory

Karen Scarfone
Scarfone Cybersecurity
Clifton, VA

Donna Dodson*

*Former NIST employee; all work for this publication was done while at NIST.

This publication is available free of charge from:
<https://doi.org/10.6028/NIST.SP.800-218>

February 2022

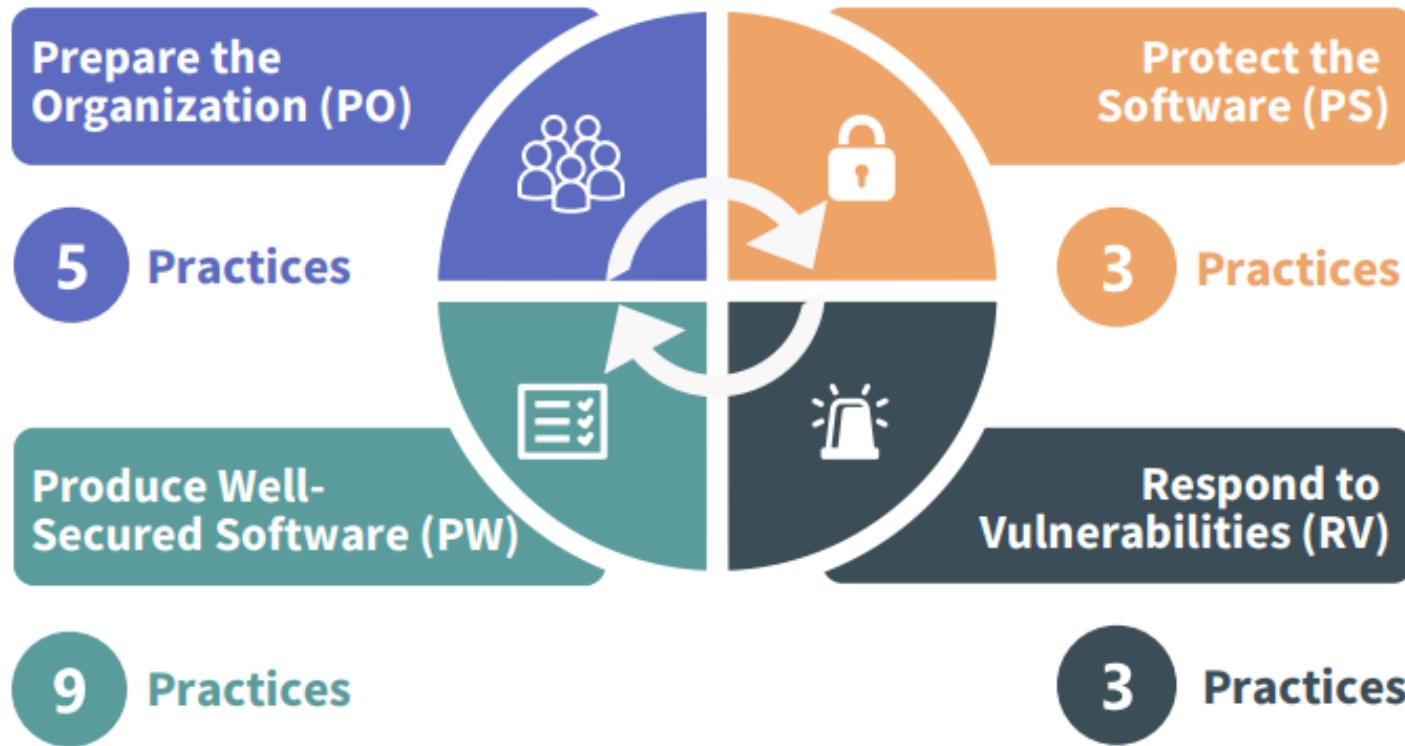


U.S. Department of Commerce
Gina M. Raimondo, Secretary

National Institute of Standards and Technology
James K. Olthoff, Performing the Non-Exclusive Functions and Duties of the Under Secretary of Commerce for Standards and Technology & Director, National Institute of Standards and Technology



Bộ khung phát triển phần mềm an toàn (2)

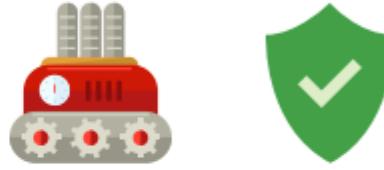


Bộ khung phát triển phần mềm an toàn (3)

Prepare the Organization - Examples



Make sure the security requirements are defined and understood by your entire organization early. Update security requirements annually – at least. (PO 1.1)



Treat build systems like production systems by securing and hardening development endpoints. (PO 5.2)

Bộ khung phát triển phần mềm an toàn (3)

Protect the Software - Examples



Use code signing to help protect the integrity of executables (PS 1.1)



Use an established certificate authority for verifying release integrity (PS 2.1)



Share provenance data e.g. in a software bill of materials [SBOM] (PS 3.2)

Bộ khung phát triển phần mềm an toàn (4)

Produce Well Secured Software - Examples



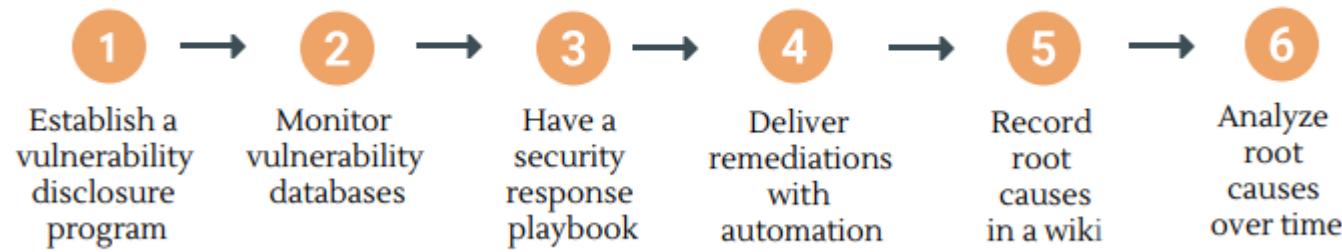
Reuse existing, well secured software (e.g. open source frameworks) instead of duplicating functionality. (PW 4.1)



Implement "clean builds" and perform all builds in a dedicated, highly controlled build environment (PW 6.2)

Bộ khung phát triển phần mềm an toàn (5)

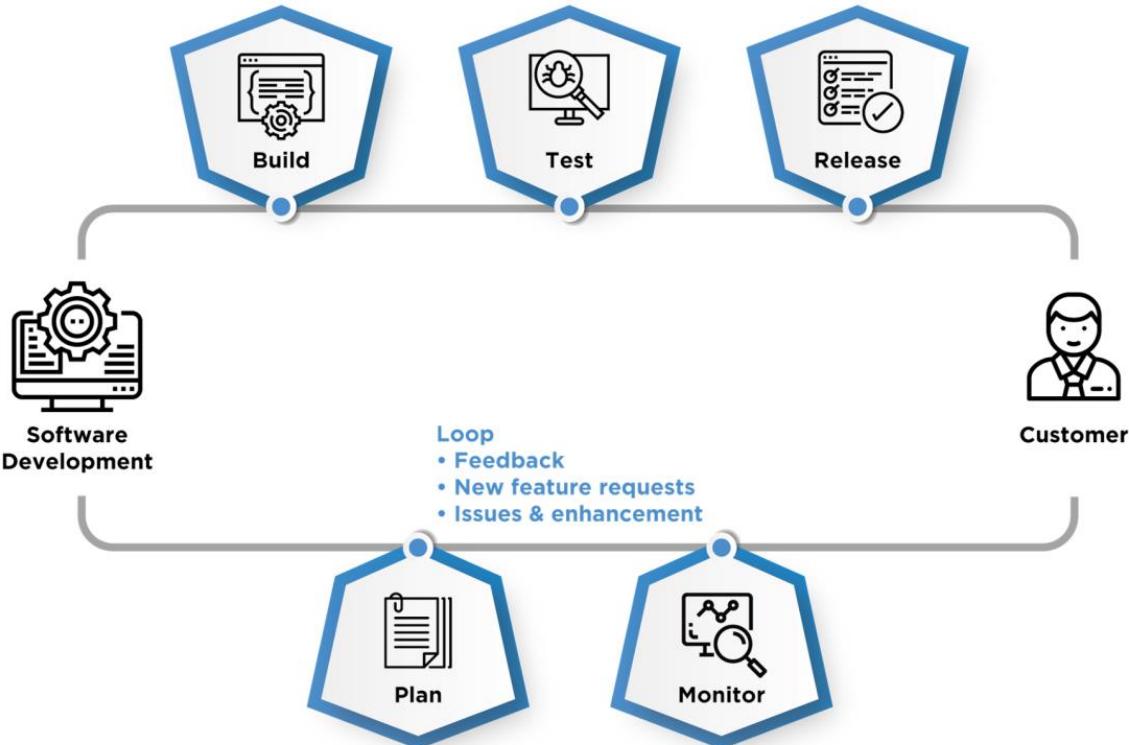
Respond to Vulnerabilities - Examples



DevOps

devops

- là **sự kết hợp** giữa **phát triển (development)** và **vận hành (operations)** phần mềm.
- Là văn hóa làm việc trong các công ty sử dụng quy trình phát triển Agile.



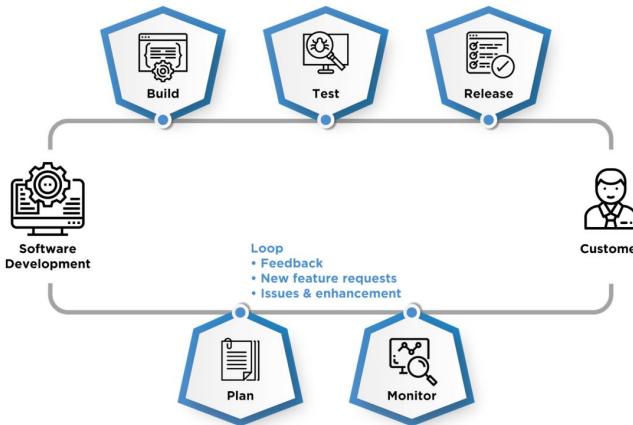
DevOps

Phương pháp mới như **Continuous Integration, Continuous Delivery and Continuous Deployment** cùng với sự phát triển của DevOps, tập trung vào:

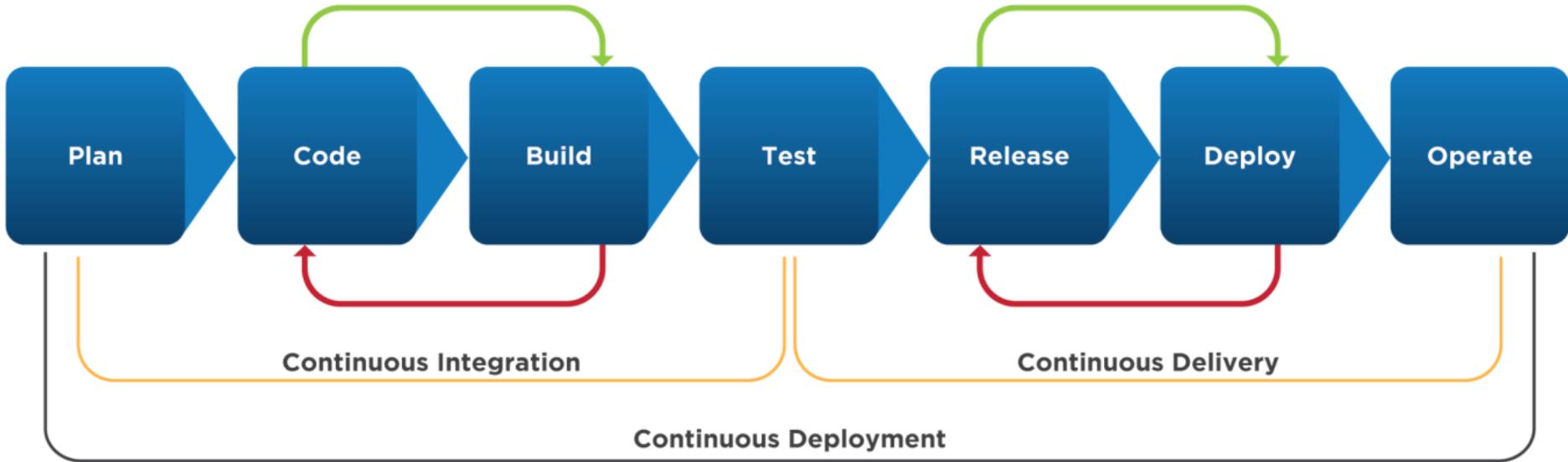
- Tương tác, phối hợp và gắn kết giữa các team.
- Áp dụng các phương pháp để tự động hóa thay đổi, cấu hình và triển khai.
- Chuyển giao giải pháp nhanh hơn.
- Theo dõi và lập kế hoạch cập nhật sản phẩm nhanh chóng.

Lợi ích của DevOps

- Cải thiện sự hợp tác, hỗ trợ chức năng và khắc phục lỗi nhanh hơn.
- Tăng tính linh hoạt, nhanh chóng và tin cậy.
- Bảo mật cơ sở hạ tầng và bảo vệ dữ liệu.
- Bảo trì và cập nhật nhanh hơn.
- Chuyển đổi các dự án với chiến lược số hóa.
- Tăng tốc độ, năng suất của team kinh doanh và CNTT.



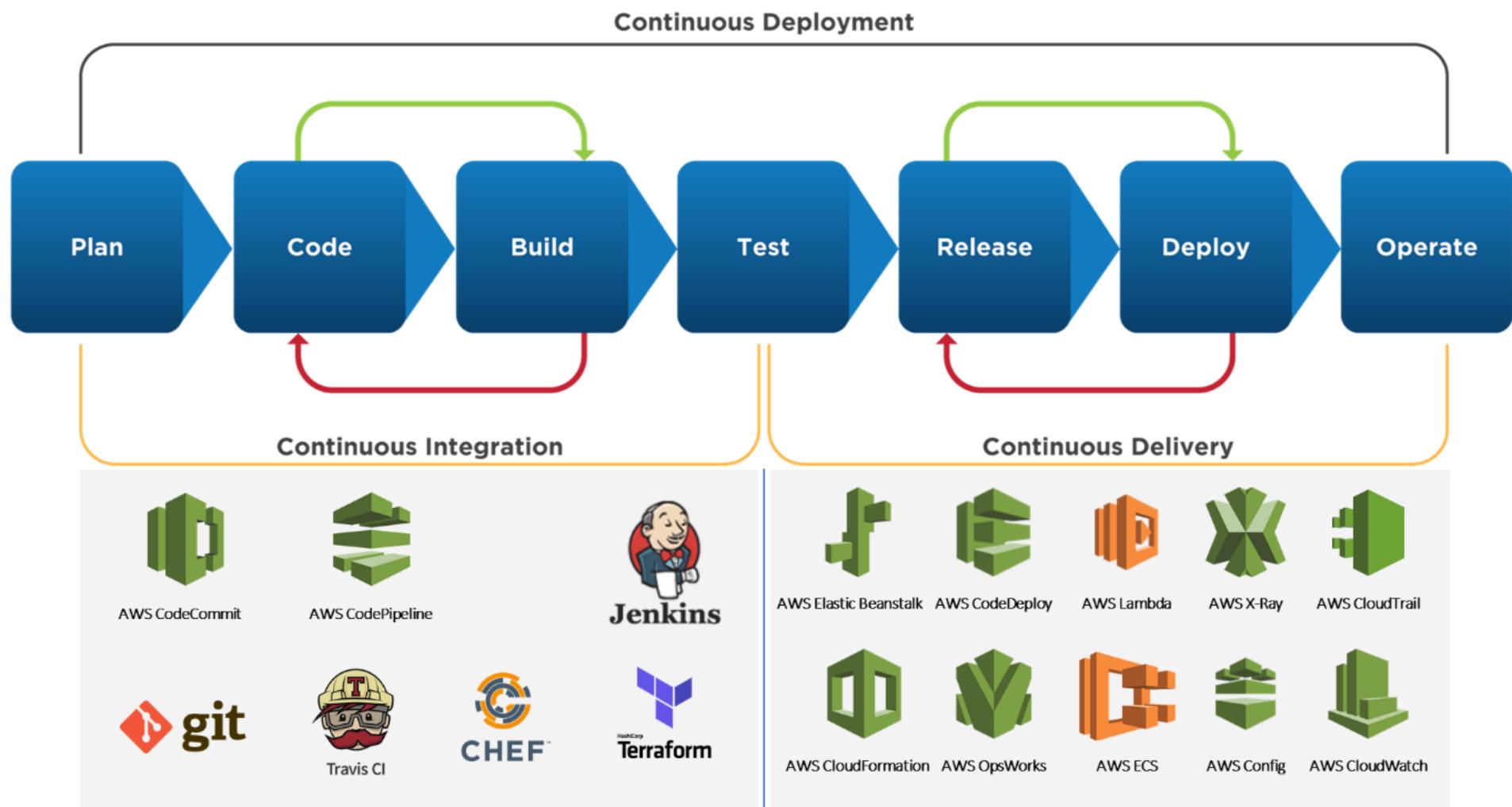
DevOps



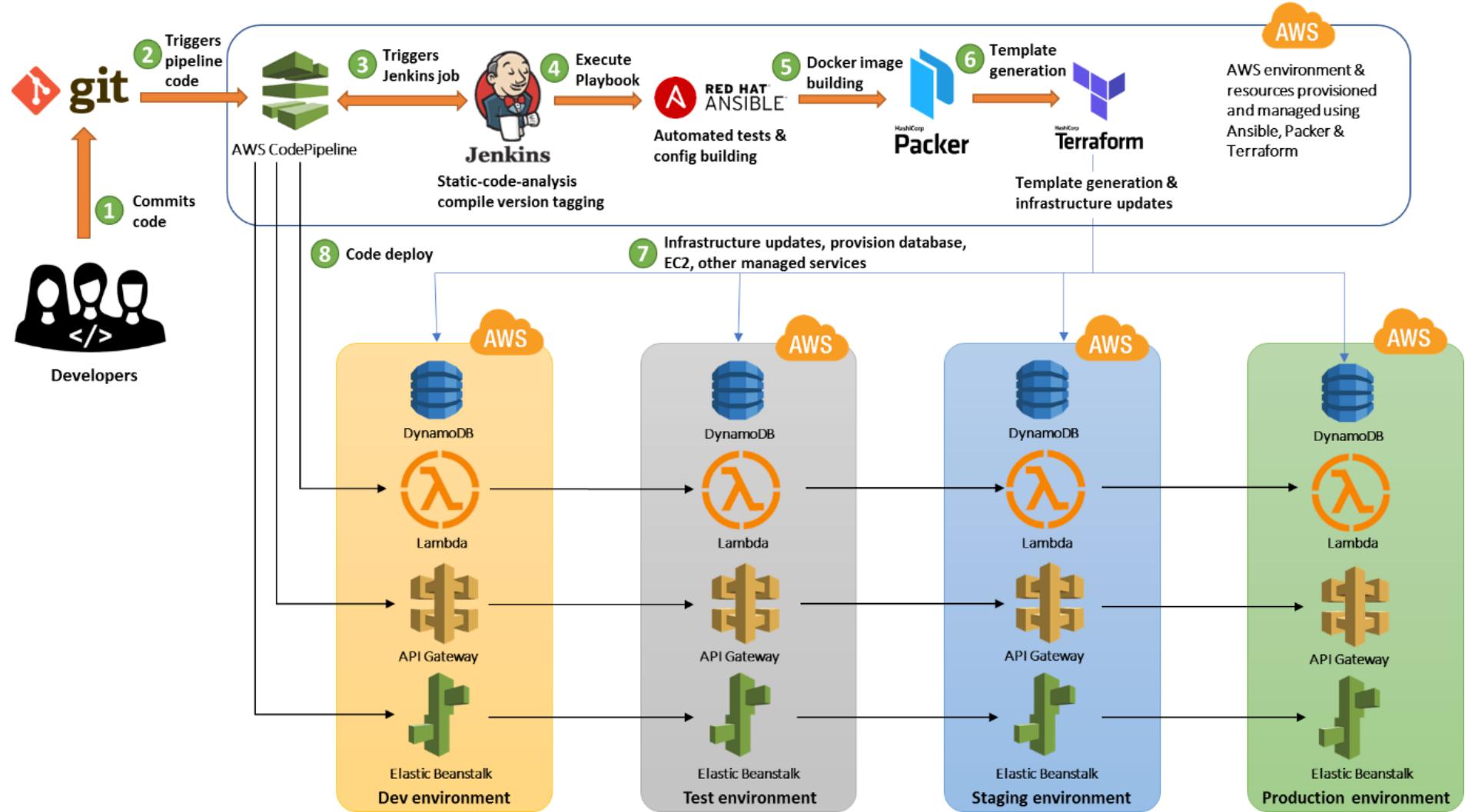
Planning	Code	Build	Test	Release	Deploy	Operate
<ul style="list-style-type: none"> Requirement finalization Updates & new changes Architecture & design Task assignment Timeline finalization 	<ul style="list-style-type: none"> Development Configuration finalization Check-in source code Static-code analysis Automated review & peer review 	<ul style="list-style-type: none"> Compile code Unit testing Code-metrics Build container images or package Preparation or update in deployment templated Create or update monitor dashboards 	<ul style="list-style-type: none"> Integration test with other component Load & stress test UI testing Penetration testing Requirement testing 	<ul style="list-style-type: none"> Preparing release notes Version tagging Code freeze Feature freeze 	<ul style="list-style-type: none"> Updating the infrastructure i.e staging, production Verification on deployment i.e smoke tests 	<ul style="list-style-type: none"> Monitor designed dashboard Alarm triggers Automatic critical events handler Monitor error logs



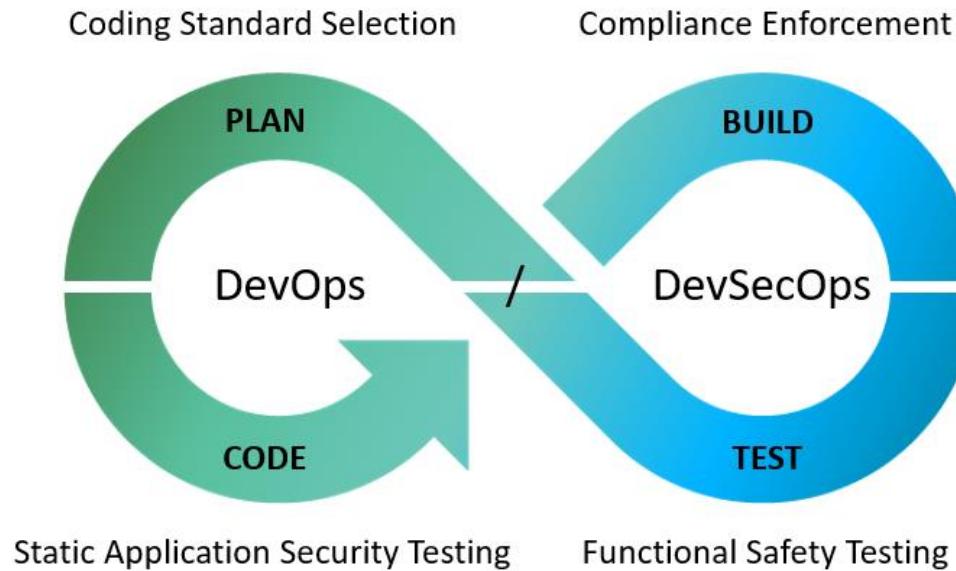
DevOps: Các công cụ



DevOps: Ví dụ



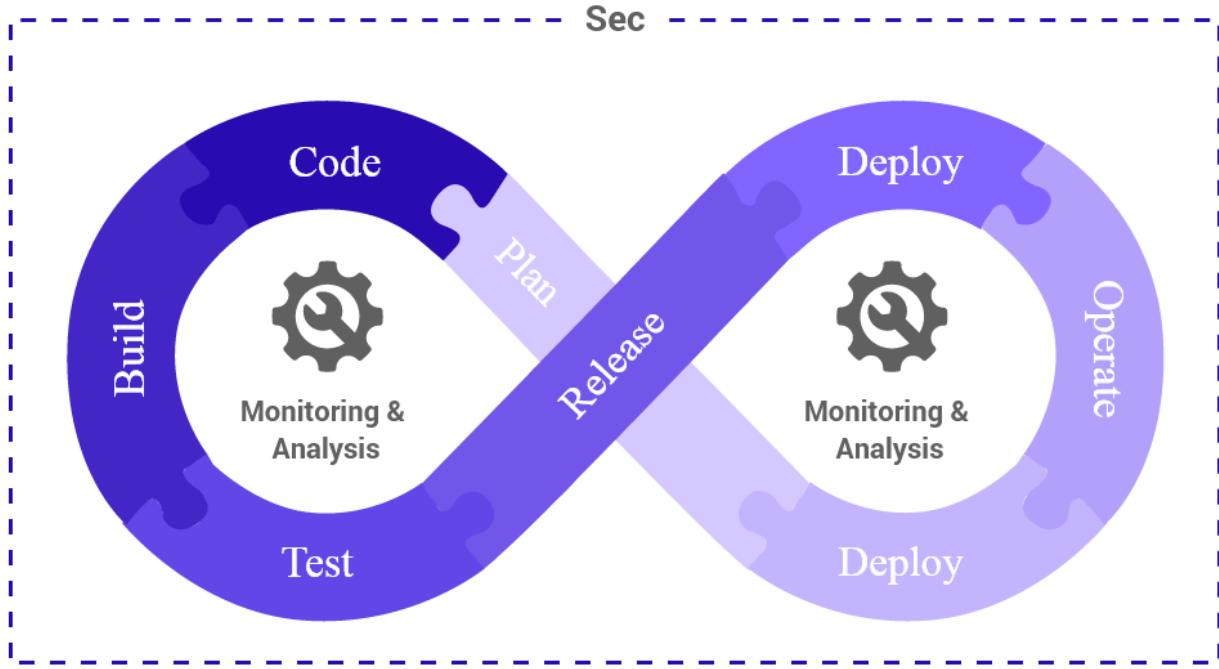
Lập trình an toàn với CI/CD



Continuous Integration (CI)
Continuous Delivery (CD)



DevSecOps: Triển khai Security vào pipeline

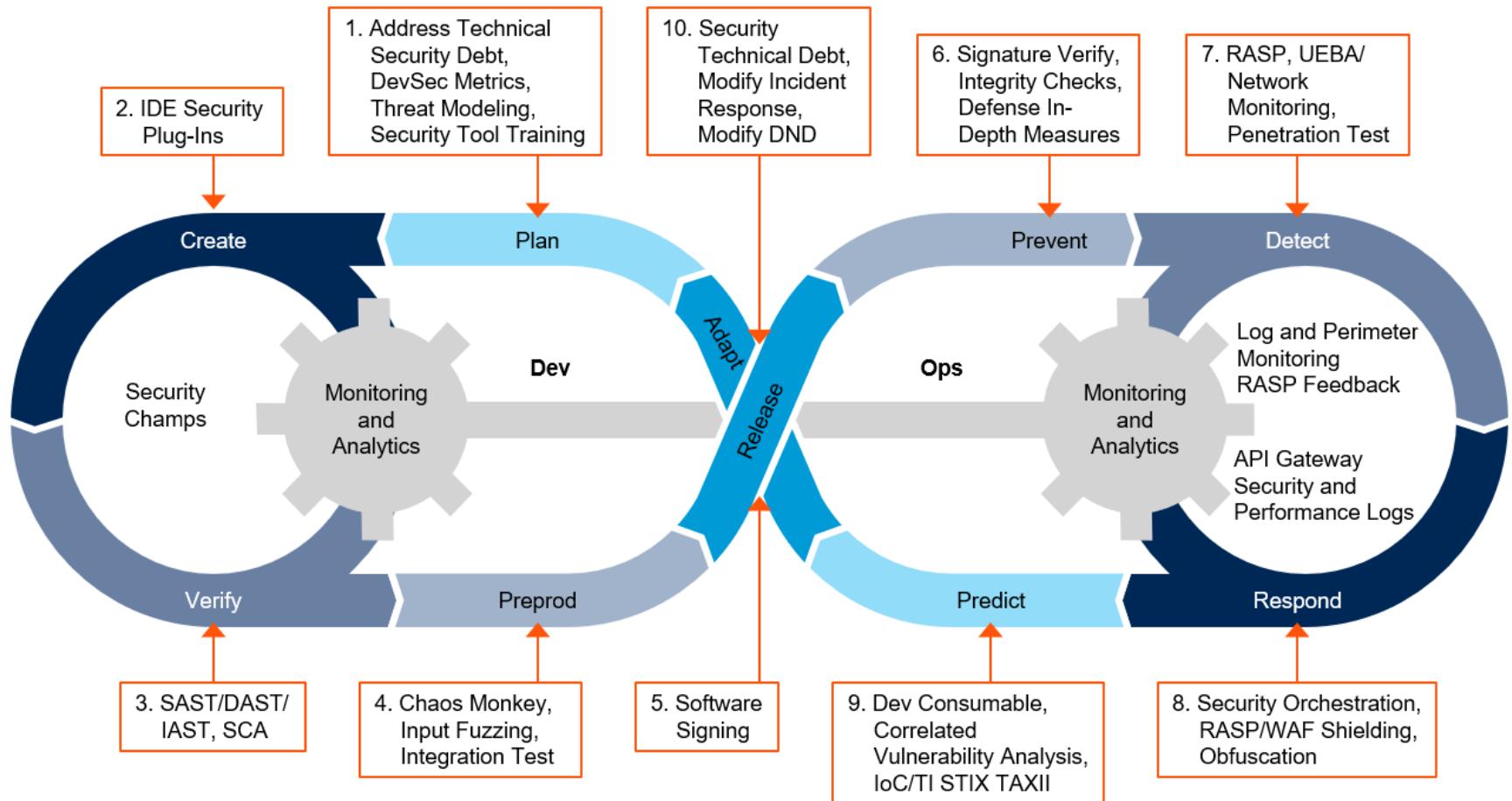


- Continuous Integration (CI)
- Continuous Delivery (CD)
- Continuous Inspection (CI)

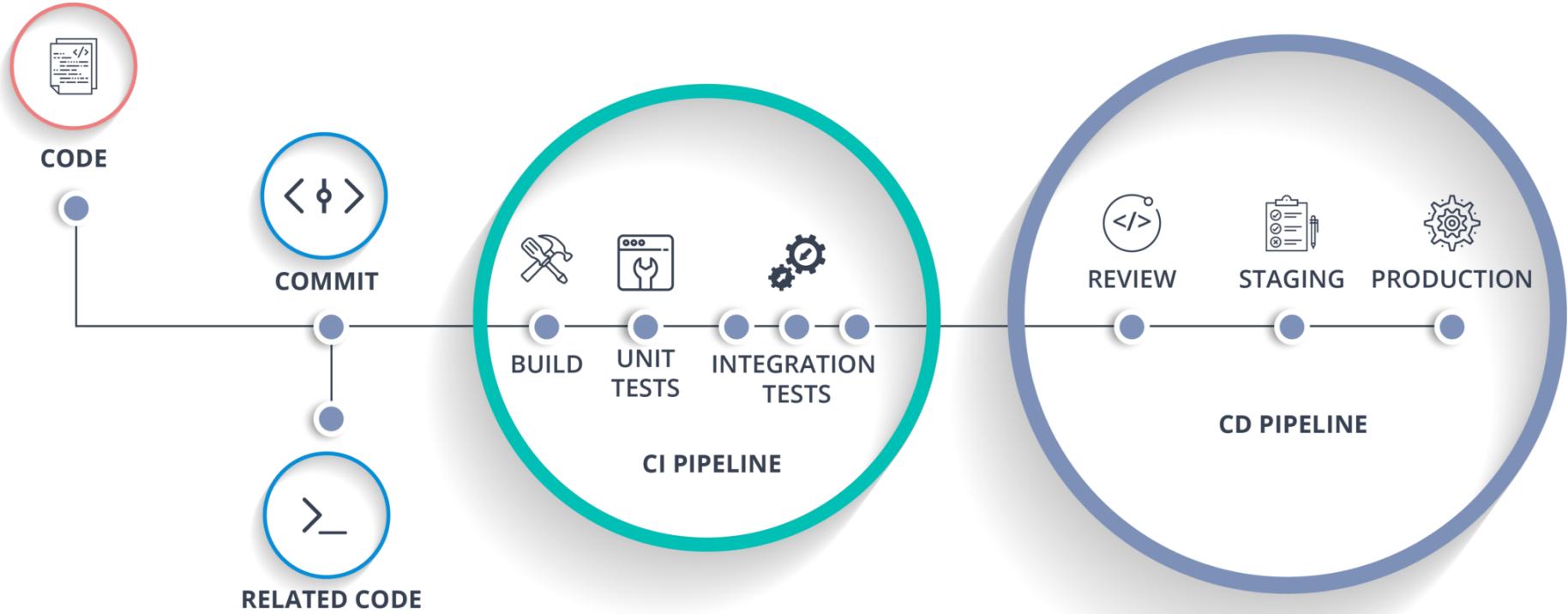


DevSecOps: Gartner Model

The DevSecOps Toolchain

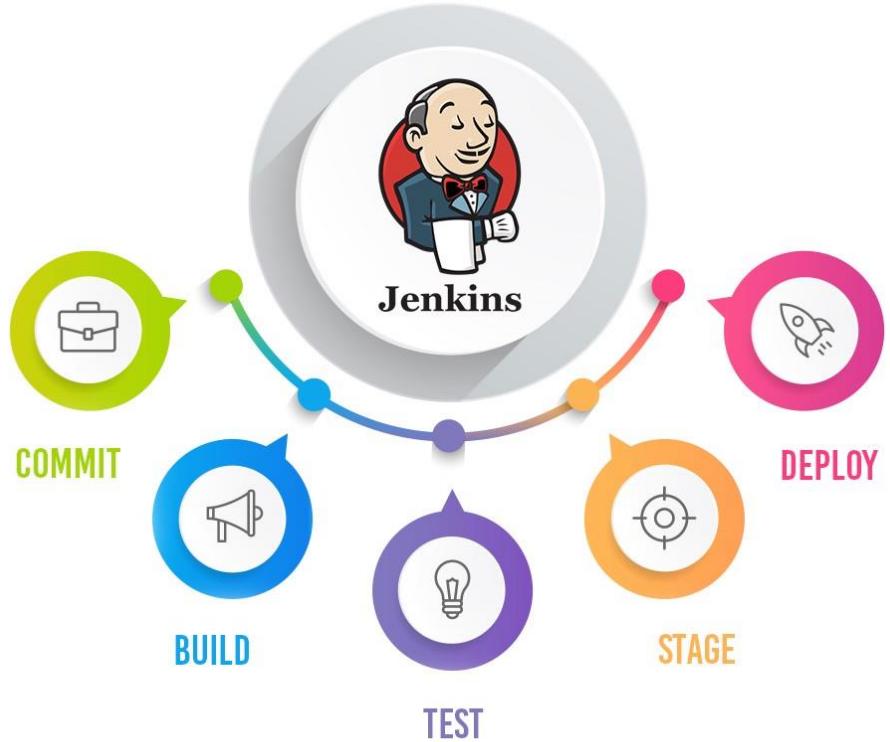
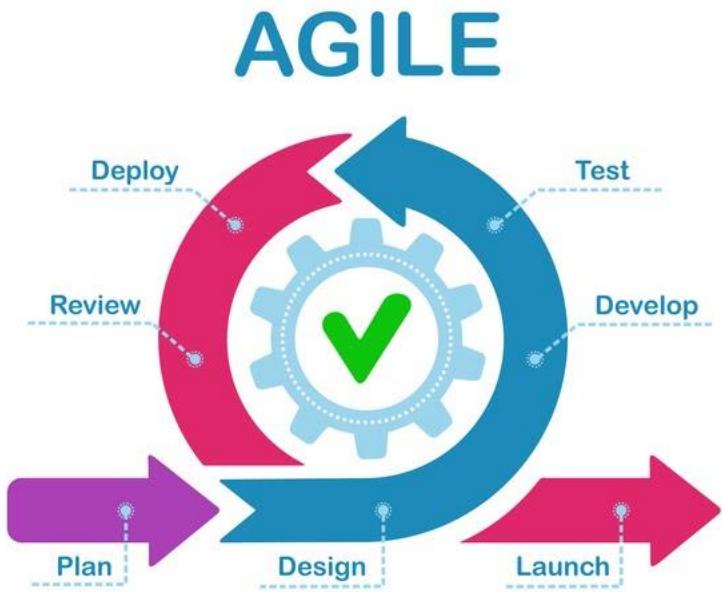


Lập trình an toàn với CI/CD

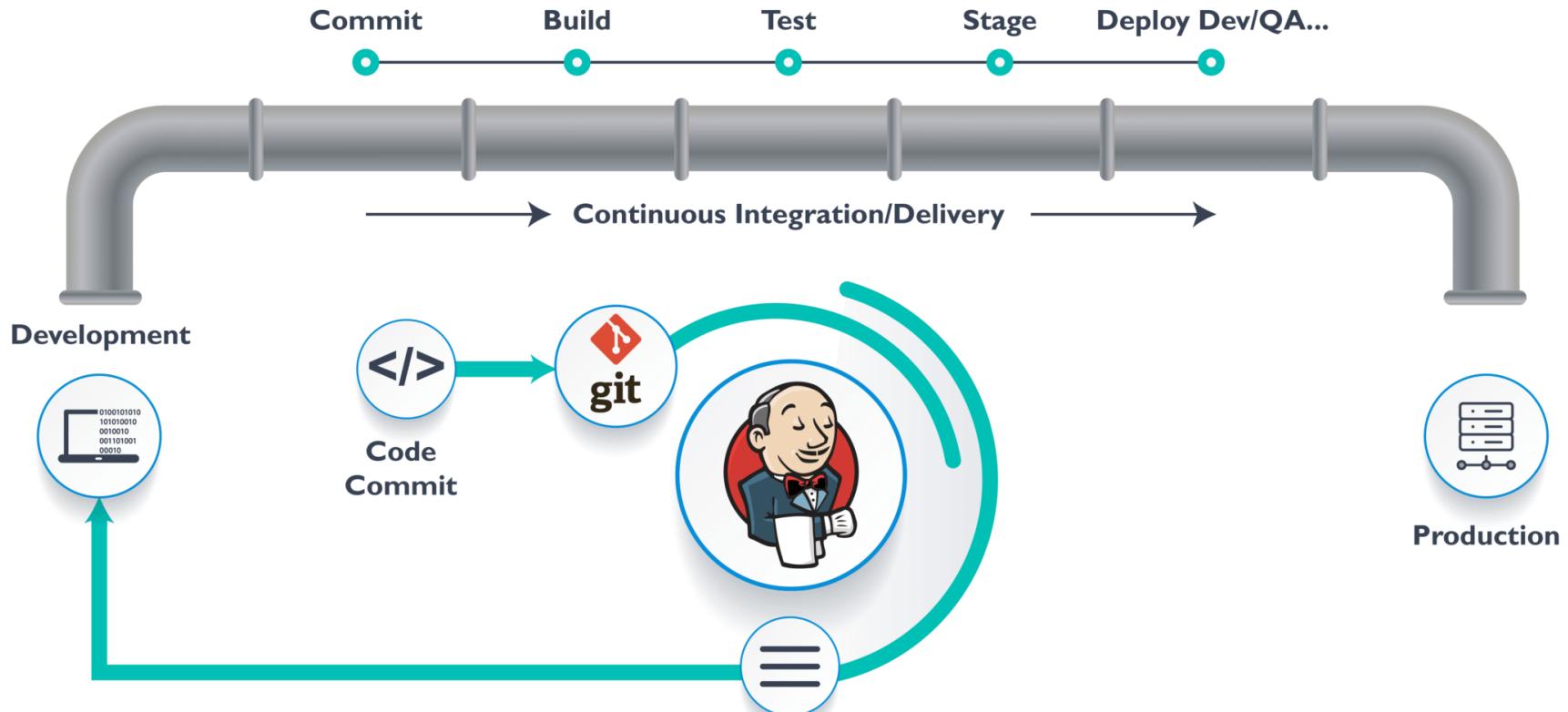


Continuous Integration (CI)
Continuous Delivery (CD)

Lập trình an toàn với CI/CD



Lập trình an toàn với CI/CD





Ví dụ: Java Security

Static Application Security Test (SAST)



Dynamic Application Security Test (DAST)



Static application security testing (SAST)

Giúp xác định các vấn đề trong mã nguồn thông qua tìm kiếm trong mã nguồn trước khi biên dịch.

Dynamic Application Security Testing (DAST):

Xác định các thông tin không thể tự động xác định từ bên trong mã nguồn.





Ví dụ: Java Security

Static Application Security Test (SAST)



Static application security testing (SAST)

Một số vấn đề có thể ngăn ngừa thông qua phân tích tĩnh:

- **Lộ tài nguyên:** phát hiện các tài nguyên bị lộ, có thể là lỗ hổng để tấn công.
- **Các khóa (keys) hay định danh (credentials) được code cứng.**
- **Vấn đề về sự ổn định:** ví dụ rò rỉ bộ nhớ như không giải phóng tài nguyên không còn được sử dụng, hoặc liên quan đến hiệu suất. Kẻ tấn công có thể cố gắng đánh sập hệ thống qua việc lợi dụng các vấn đề hiệu suất.





Ví dụ: Java Security

Dynamic Application Security Test (DAST)



- Quét các ứng dụng Web và phân tích các header được sử dụng.
- Tìm các điểm đầu cuối bị lộ và các chức năng bảo mật của chúng.
- Kiểm tra các giao tiếp/dịch vụ OpenAPI/SOAP/REST/Websocket/
- Kiểm tra lưu lượng mạng giữa trình duyệt và dịch vụ.



Ví dụ: Java Security

Application Dependency Vulnerability Scanning



Phân tích phụ thuộc

OWASP Dependency Check

- Có thể chạy từ Maven build và cung cấp dữ liệu cho Jenkins và SonarQube (bằng cách cài đặt plugin Dependency-Check).
- Ghi lại điểm CVSS (Common Vulnerability Scoring System) cho đánh giá rủi ro.

<https://owasp.org/www-project-dependency-check/>



Ví dụ: Java Security

Container Image Vulnerability Scanning



ANCHORE ENGINE

- **Anchore Engine** là dự án mã nguồn mở, cung cấp dịch vụ để kiểm tra, phân tích và chứng nhận các **image container**.
- **Anchore Engine** được cung cấp dưới dạng một image container Docker, có thể chạy độc lập hoặc trong một nền tảng điều hướng như Kubernetes, Docker Swarm, Rancher, Amazon ECS,...



Continuous Inspection



Continuous Inspection
sonarQube

<https://docs.sonarqube.org/latest/>

SonarQube® là một công cụ đánh giá mã nguồn tự động để phát hiện bug, lỗi hỏng và “code smell” trong mã nguồn. Có thể tích hợp với luồng hoạt động đã có để kích hoạt CI (continuous inspection) trong các nhánh dự án và các yêu cầu pull.





Continuous Inspection

For 27 languages >

Java >

JavaScript >

C# >

Python >

C++ >

COBOL >

58

Fortune
100 use
Enterprise
Editions

Products First

Our prime focus and dedication is in building great products
that have an impact and are loved by their users.

IDE



Fix Issues Before they Exist

[Discover SonarLint →](#)

CLOUD



Enhance Your Workflow with
Continuous Code Quality

[Discover SonarCloud →](#)

ON-PREMISE



Your teammate for Code Quality and
Code Security

[Discover SonarQube →](#)

<https://www.sonarsource.com/>





Lập trình an toàn với CI/CD

Jenkins

Dashboard > Demo Pipeline > main > #26

Back to Project

Status

Changes

Console Output

Edit Build Information

Delete build '#26'

Git Build Data

See Fingerprints

Test Result

Dependency-Check

SpotBugs Warnings

CPD Warnings

PMD Warnings

ZAP report

Maven

Anchore Report (PASS)

Restart from Stage

Replay

Pipeline Steps

Workspaces

Previous Build

Build #26 (Mar 3, 2021 1:13:47 PM)

Build Artifacts

File	Size	Action
anchore_gates.json	3.23 KB	view
anchore_security.json	13.92 KB	view
anchoreengine-api-response-evaluation-1.json	17.34 KB	view
anchorengine-api-response-vulnerabilities-1.json	30.03 KB	view
spring-boot-demo-0.0.1-SNAPSHOT.jar	26.39 MB	view
spring-boot-demo-0.0.1-SNAPSHOT.pom	12.82 KB	view

Changes

- 1. Adding vulnerability ([details](#) / [githubweb](#))

git

- Started by user [admin](#)
- Revision: c6a94e4e3185527f9cd92b19df911ad42f666893
 - main

Test Result (no failures)

SpotBugs: One warning ⓘ

- Reference build: [Demo Pipeline » main #25](#)

CPD: No warnings ⓘ

- No warnings for 16 builds, i.e. since build 11
- Reference build: [Demo Pipeline » main #25](#)

PMD: One warning ⓘ

- Reference build: [Demo Pipeline » main #25](#)

Anchore Report (PASS)





Lập trình an toàn với CI/CD

Jenkins: Building Java and deploying to Kubernetes

Kubernetes is a popular platform to run and manage containerized applications.

- Jenkins as CI/CD platform
- Kubernetes deployed with Kubespray on KVM ([here](#))
- MetallLB as Kubernetes loadbalancer
- GitHub as version control system
- Smee to forward Webhook calls from GitHub to Jenkins
- Maven to build the Java code
- Google Jib to wrap my compiled Java in a container
- DockerHub as container registry

Java Project: <https://github.com/MaartenSmeets/spring-boot-demo>





Lập trình an toàn với CI/CD

Jenkins: Building Java and deploying to Kubernetes

localhost:8080/job/Build%20spring-boot-demo/job/test/

Jenkins

Up Status Changes Build Now View Configuration Full Stage View Open Blue Ocean GitHub Maven Pipeline Syntax Build History trend

Branch test

Full project name: Build spring-boot-demo/test

Last Successful Artifacts

- spring-boot-demo-0.0.1-SNAPSHOT.jar (17.68 MB) view
- spring-boot-demo-0.0.1-SNAPSHOT.pom (3.36 KB) view

Recent Changes

Test Result Trend

Failed Skipped Passed

Stage View

Average stage times: (Average full run time: ~4min 30s)

Declarative: Checkout SCM	Declarative: Tool Install	Build	Create and push container	Deploy to K8s
16s	1min 32s	1min 30s	32s	7s
16s	1min 32s	1min 30s	32s	7s

Latest Test Result (no failures)

#1 Sep 23, 2020 6:25 PM

Atom Feed for all Atom Feed for failures

Permalinks

- Last build (#1), 6 min 41 sec ago
- Last stable build (#1), 6 min 41 sec ago
- Last successful build (#1), 6 min 41 sec ago
- Last completed build (#1), 6 min 41 sec ago

Java Project: <https://github.com/MaartenSmeets/spring-boot-demo>





Lập trình an toàn với CI/CD

Jenkins Pipeline: SonarQube and the OWASP Dependency-Check on Kubernetes

The screenshot shows the Jenkins interface for a 'Dependency-Check' build step. The left sidebar lists various Jenkins features like Back to Project, Status, Changes, Console Output, etc., with 'Dependency-Check' currently selected. The main content area displays the 'Dependency-Check Results' with a 'Severity Distribution' bar at the top. Below is a table with columns: File Name, Vulnerability, Severity, and Weakness. The table lists multiple occurrences of 'spring-boot-demo-0.0.1-SNAPSHOT.jar' with various CVE numbers and severity levels (High, Critical, Medium). A search bar is also present in the header.

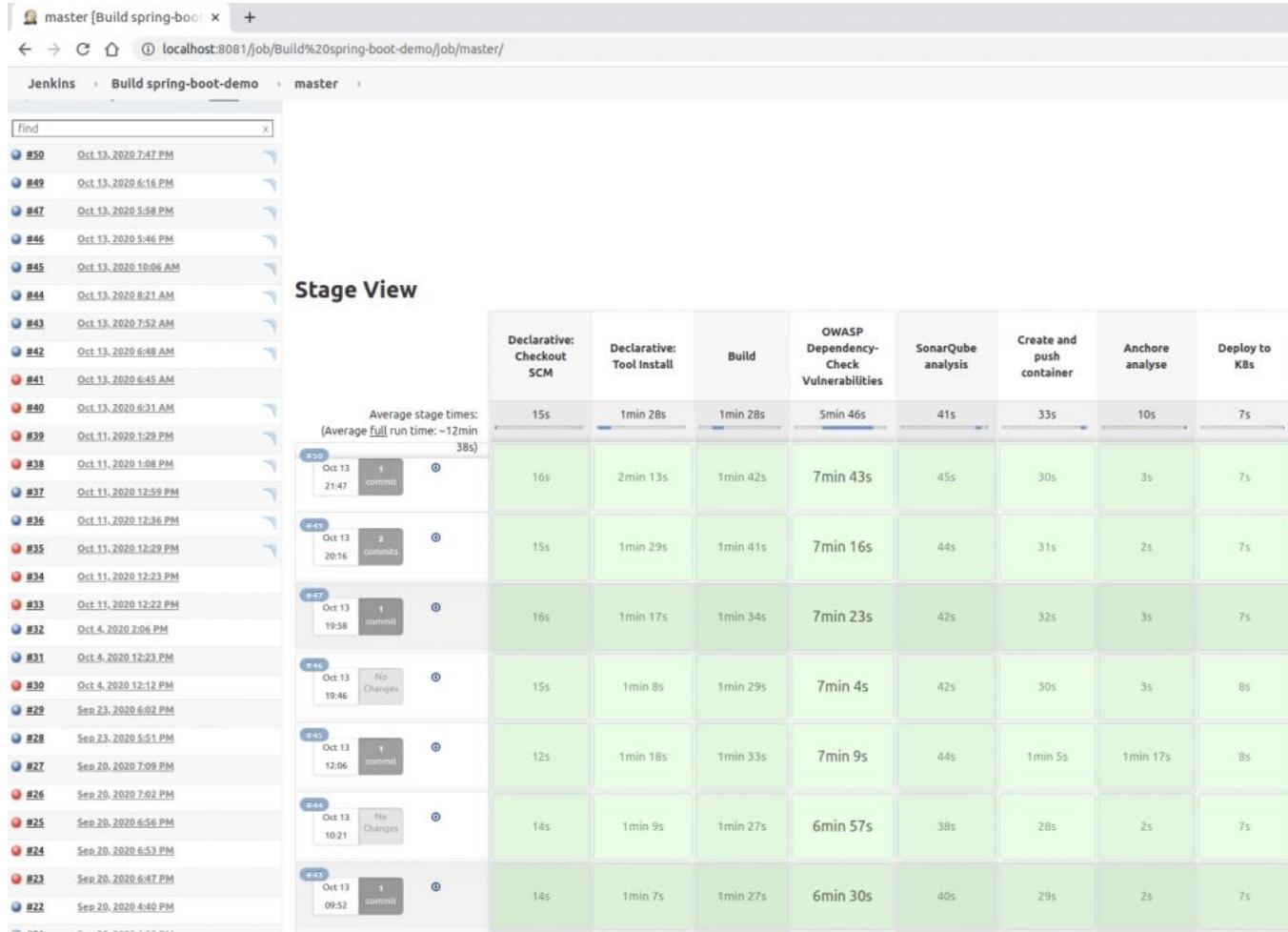
File Name	Vulnerability	Severity	Weakness
+ spring-boot-demo-0.0.1-SNAPSHOT.jar	NVD CVE-2016-9878	High	CWE-22
+ spring-boot-demo-0.0.1-SNAPSHOT.jar	NVD CVE-2017-8046	Critical	CWE-20
+ spring-boot-demo-0.0.1-SNAPSHOT.jar	NVD CVE-2018-1196	Medium	CWE-59
+ spring-boot-demo-0.0.1-SNAPSHOT.jar	NVD CVE-2018-1270	Critical	CWE-358
+ spring-boot-demo-0.0.1-SNAPSHOT.jar	NVD CVE-2018-1271	Medium	CWE-22
+ spring-boot-demo-0.0.1-SNAPSHOT.jar	NVD CVE-2018-1272	High	NVD-CWE-noinfo
+ spring-boot-demo-0.0.1-SNAPSHOT.jar	NVD CVE-2020-5421	Medium	NVD-CWE-noinfo





Lập trình an toàn với CI/CD

Jenkins Pipeline: SonarQube and the OWASP Dependency-Check on Kubernetes





Lập trình an toàn với CI/CD



OWASP VulnerableApp

OWASP VulnerableApp Project: For Security Enthusiasts by Security Enthusiasts:

<https://github.com/SasanLabs/VulnerableApp>

Technology used:

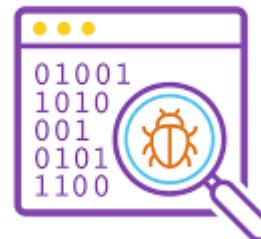
- Java8
- Spring Boot
- Vanilla Javascript





Lập trình an toàn với CI/CD

- Java Security Lab: <https://github.com/CSPF-Founder/JavaVulnerableLab>
- Appsecco: <https://github.com/appsecco>
- Damn Vulnerable Java (EE) Application: <https://github.com/appsecco/dvja>
- KubeSecO: Application Security Workflow Automation using Docker and Kubernetes
Link: <https://github.com/appsecco/kubeseco>





Lập trình an toàn với CI/CD

The screenshot shows the w3af project's homepage. At the top, there's a navigation bar with links for DOWNLOAD, TAKE A TOUR, COMMUNITY, BLOG, and DOCS. The main content area features a large announcement: "Complete rewrite". Below it, a paragraph discusses the transition from previous releases to new ones, mentioning "nasty bugs". Another section talks about "Test Driven Development, unittests, integration tests and continuous integration". To the right, a Python traceback is displayed in a box, with a large red circle and a slash through a green bug icon overlaid on it, indicating a fix or detection.

w3af is a Web Application Attack and Audit Framework. The project's goal is to create a framework to help you secure your web applications by finding and exploiting all web application vulnerabilities.

Our framework is proudly developed using Python to be *easy to use and extend*, and licensed under GPLv2.0.



Our project has [an interesting history](#) which has defined our [long and short term objectives](#) and told us many important lessons. Don't forget to follow our [blog](#) and [twitter](#) account for news, releases and feedback.

RT @fwdcldusec: Today is the big day! The start of fwdcldusec is in a few hours. Live stream links and schedule are at: <https://t.co/...> [2 weeks ago]

RT @AndresRiancho: CVE-2020-17513: SSRF on Airflow.



The easiest way to learn about what w3af is and how you can use it to secure your web applications is to [take our project tour](#) and read our [FAQ](#).

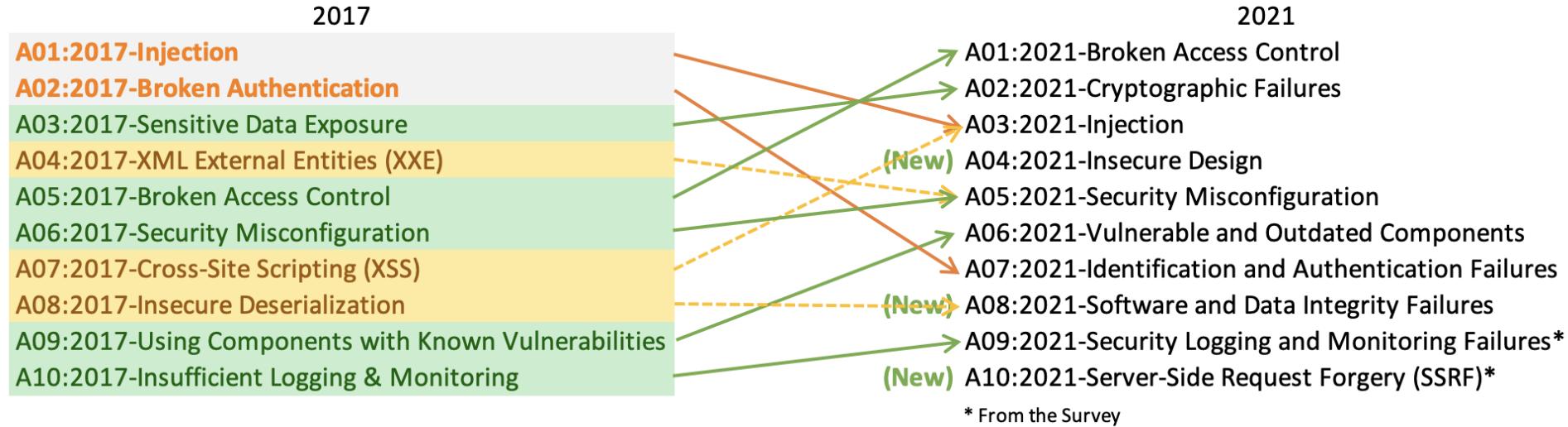
w3af is a **Web Application Attack and Audit Framework**. The project's goal is to create a framework to help you secure your web applications by finding and exploiting all web application vulnerabilities.



Các lỗi thường gặp

- OWASP: Open Web Application Security Project, OWASP mobile security,

...



Các lỗi thường gặp CSRF

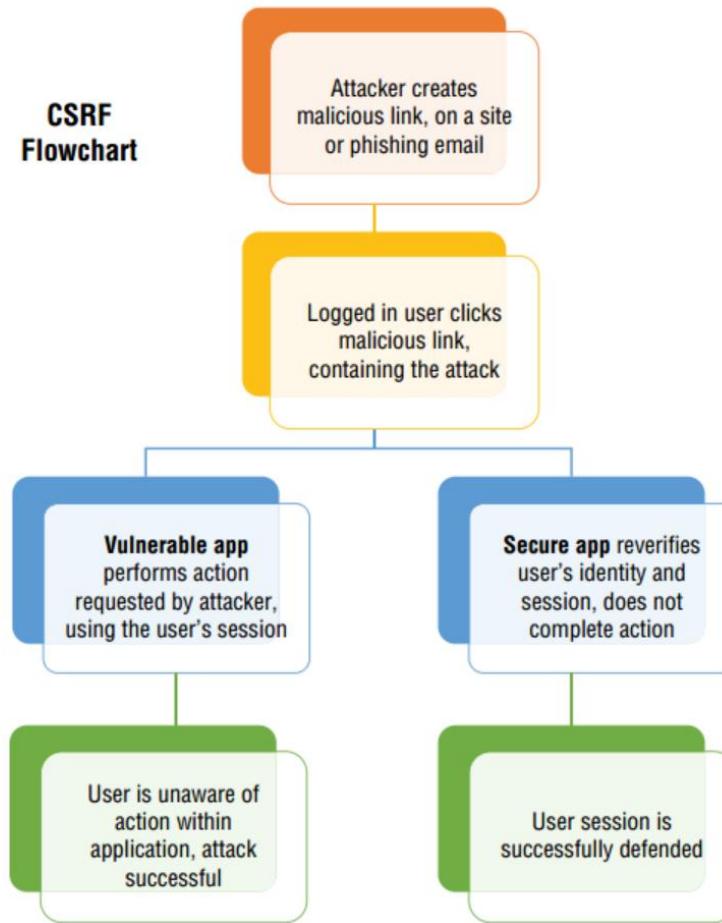


Figure 5-1: CSRF flowchart



Server-Side Request Forgery (SSRF)

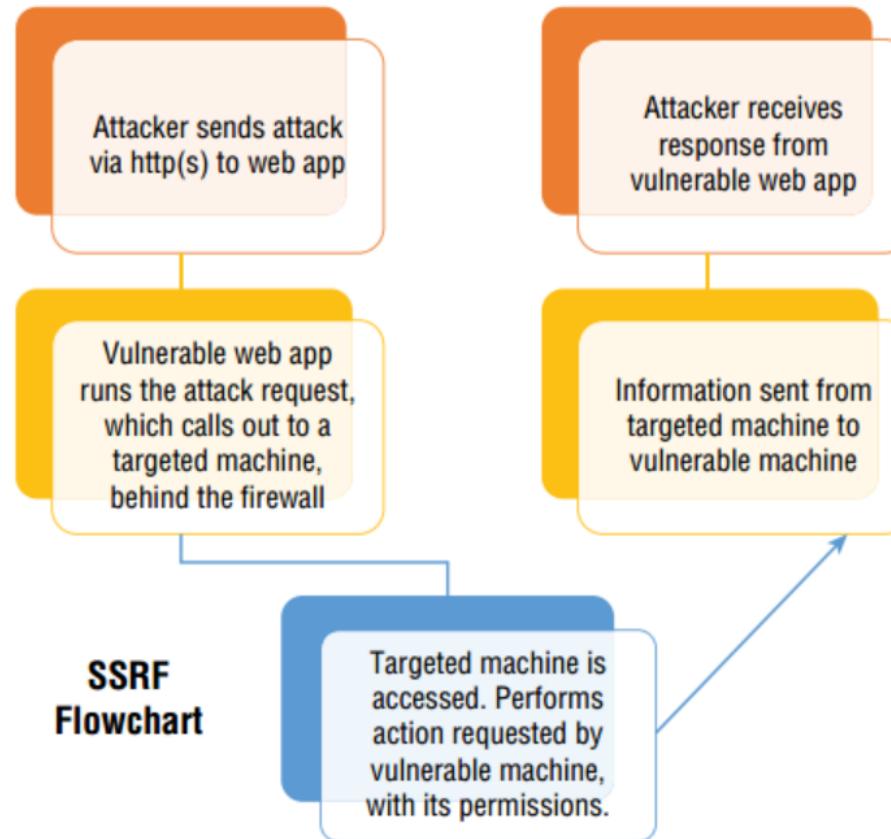


Figure 5-2: SSRF flowchart





Insecure Deserialization

- **Serialization vs Deserialization?**

- **Serialization:** chuyển dữ liệu từ dạng object sang dạng có thể truyền hoặc lưu trữ trong stream hoặc file.
- **Deserialization:** đưa dữ liệu về dạng object ban đầu sau khi truyền hoặc khi cần sử dụng.
- **Insecure Deserialization:** khi kẻ tấn công có thể thay thế dữ liệu đã serialize bằng dữ liệu độc hại nào đó, có thể gây ra hậu quả nghiêm trọng.
- Giải pháp: **hạn chế serialization/ deserialization khi có thể.** Nếu cần sử dụng object đã serialize, **chỉ sử dụng dữ liệu từ các nguồn tin cậy.**





Race Conditions

- **Race condition** xảy ra khi các hoạt động của phần mềm thực thi dựa trên thời gian, và các tiêu trình cần được thiết lập đồng bộ, tuy nhiên không phải như vậy.



Các lỗi khác

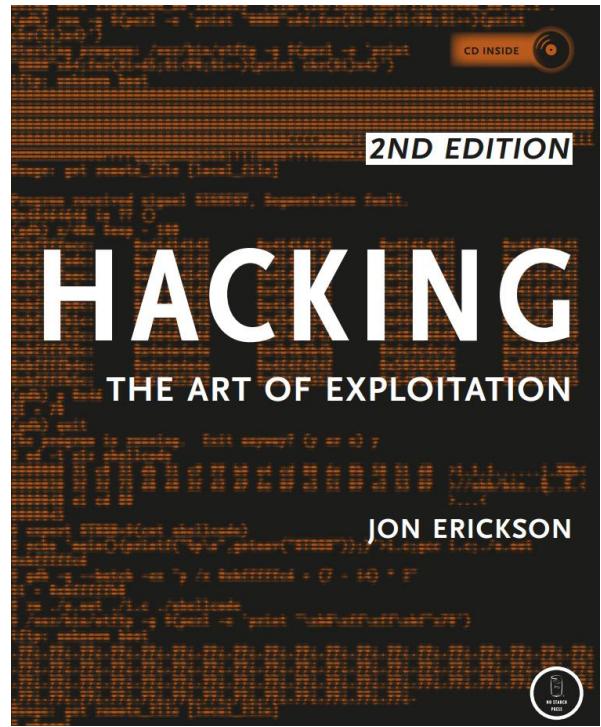
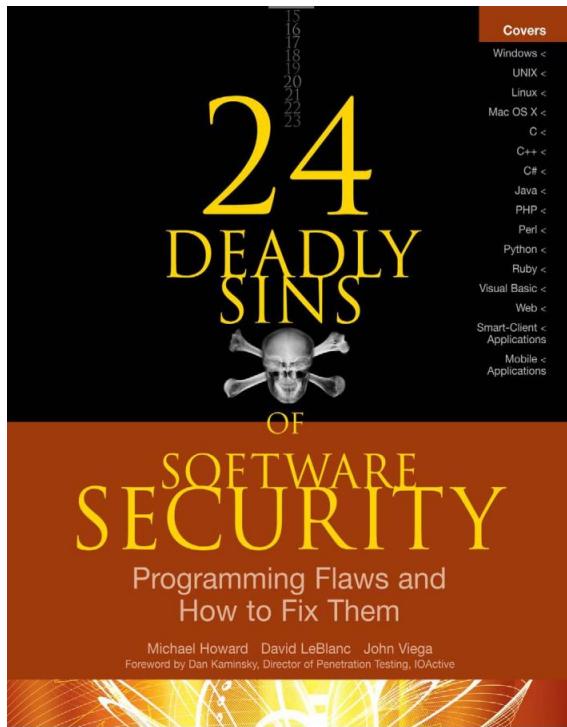
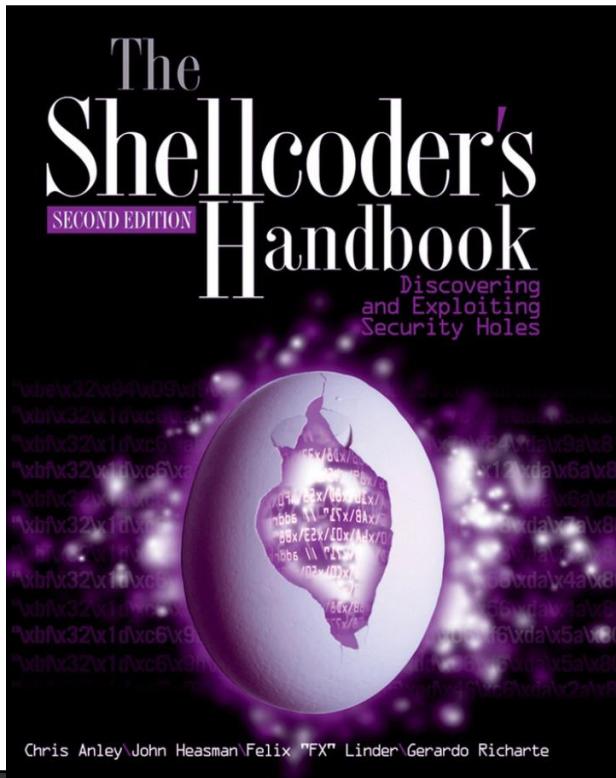


- Xem thêm trong OWASP, CWE,...



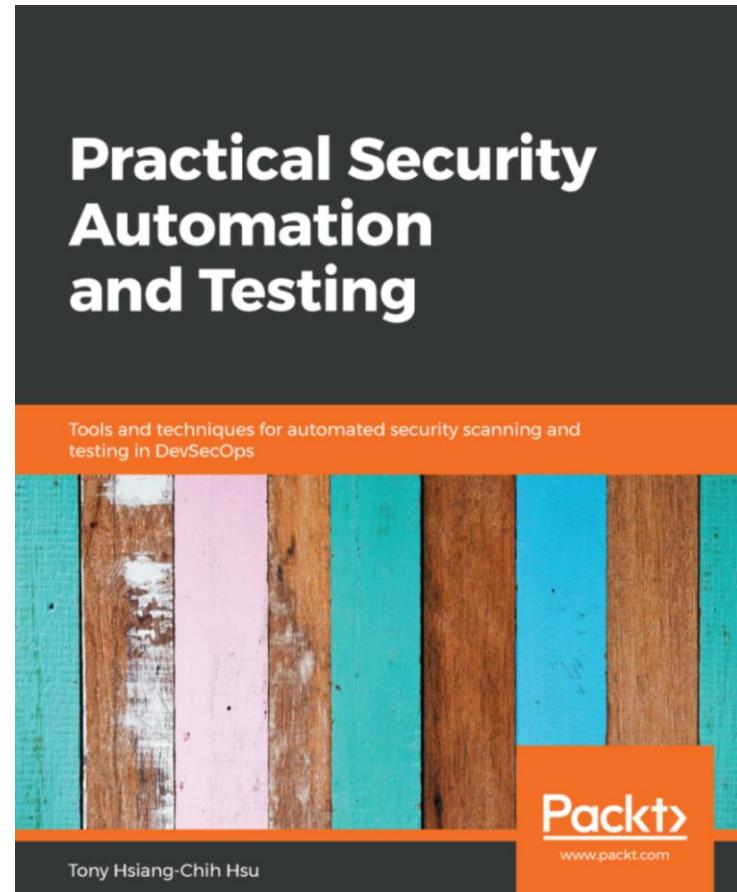
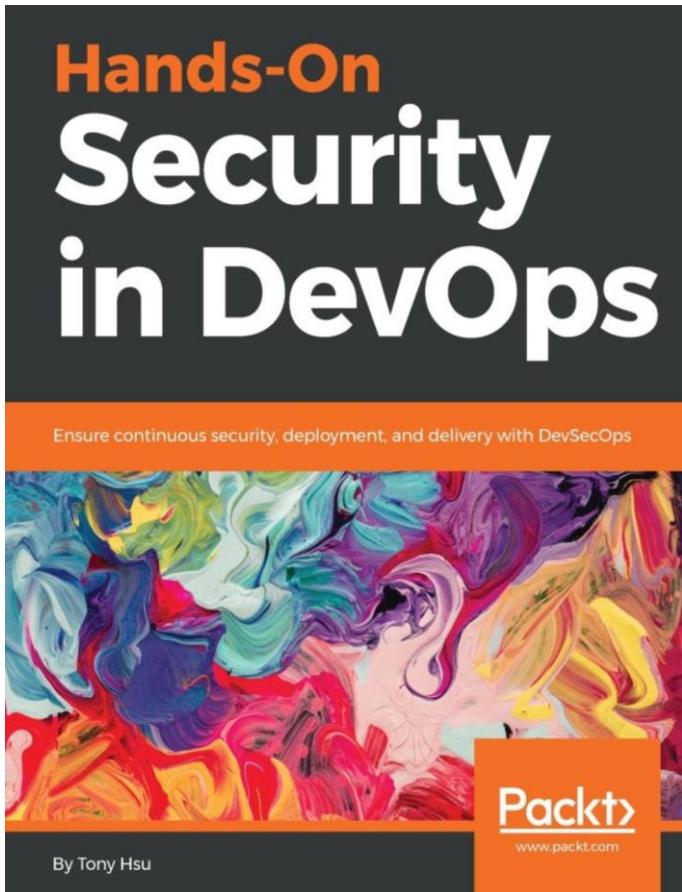


Tài liệu tham khảo





Tài liệu tham khảo





Tài liệu tham khảo

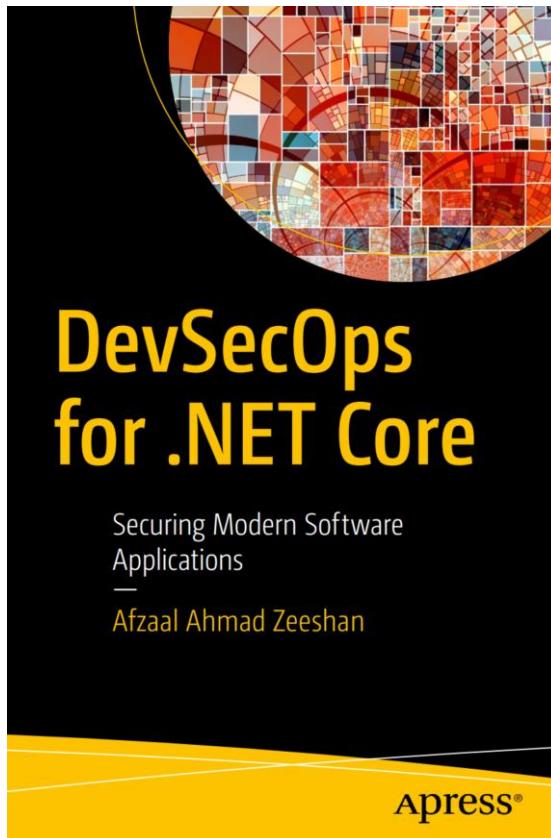
O'REILLY®



Agile Application Security

ENABLING SECURITY IN A CONTINUOUS DELIVERY PIPELINE

Laura Bell, Michael Brunton-Spall,
Rich Smith & Jim Bird



O'REILLY®

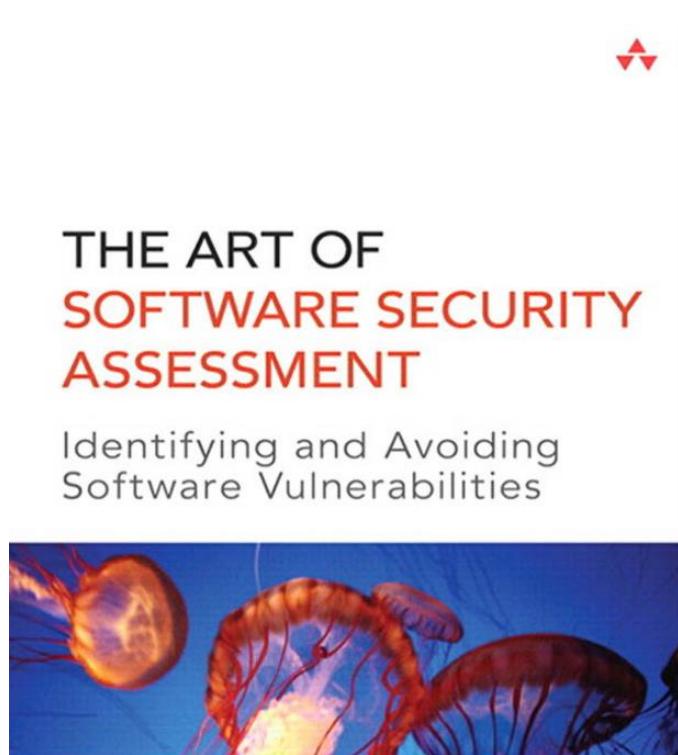
DevOpsSec

Delivering Secure Software
Through Continuous Delivery

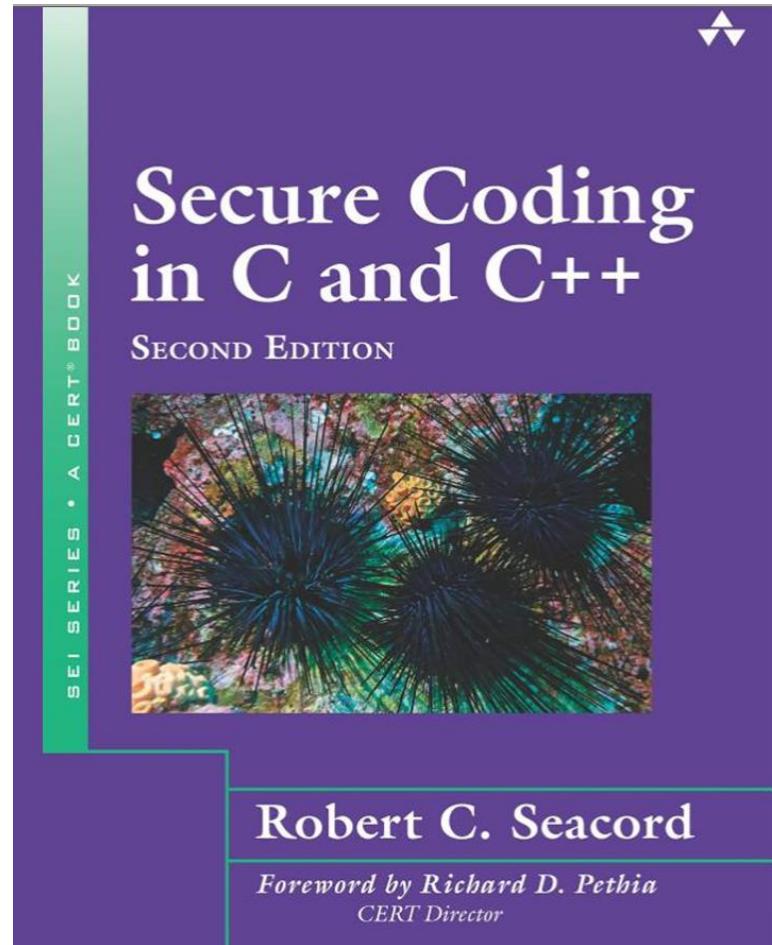




Tài liệu tham khảo

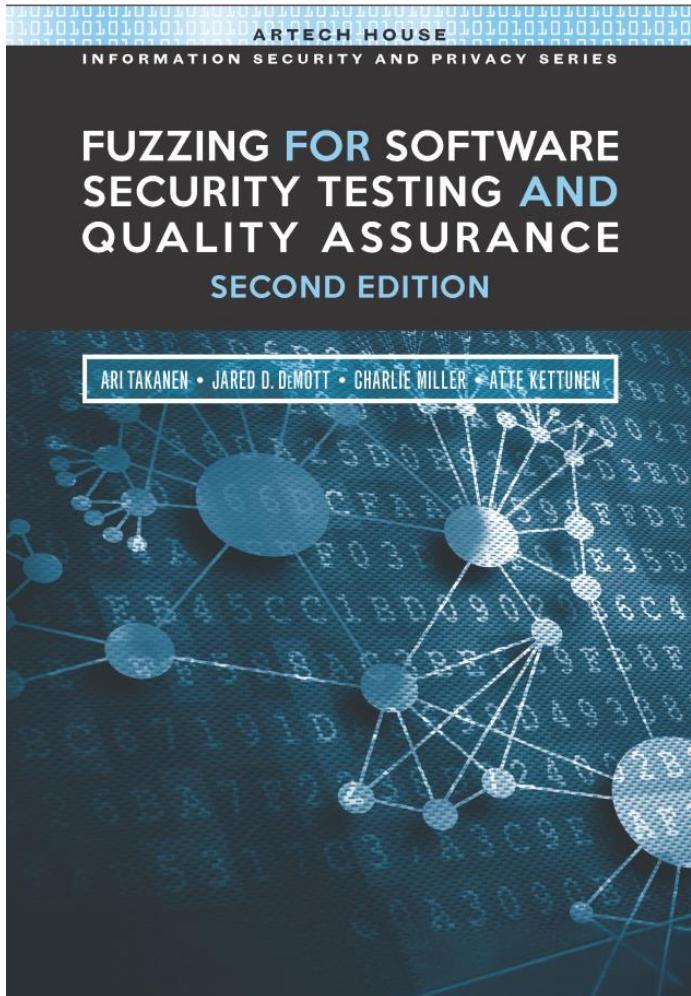


MARK DOWD
JOHN McDONALD

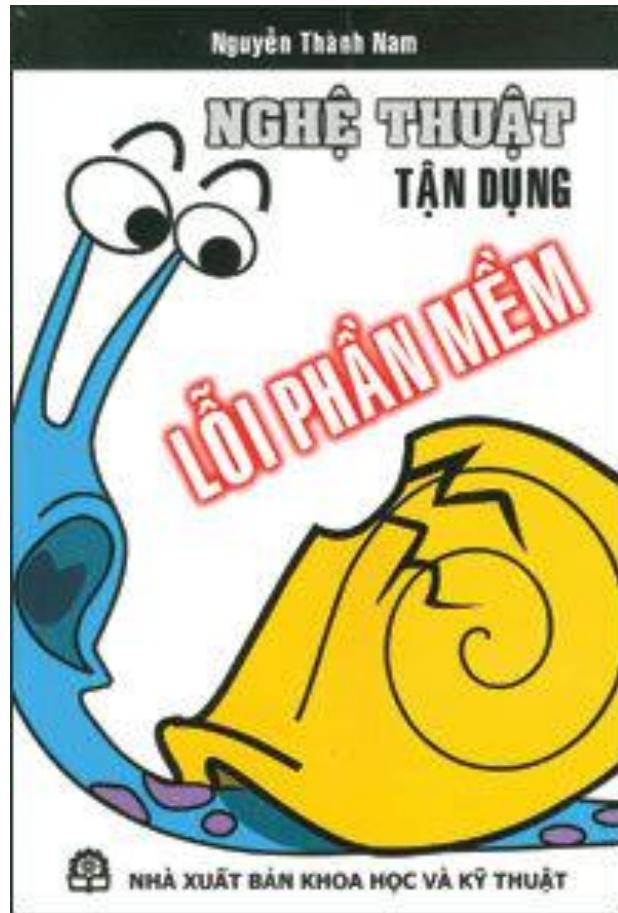




Tài liệu tham khảo



Tài liệu tham khảo





Tài liệu tham khảo

- <https://security.berkeley.edu/secure-coding-practice-guidelines>
- https://wiki.sei.cmu.edu/confluence/display/sec_code/Top+10+Secure+Coding+Practices
- https://owasp.org/www-pdf-archive/OWASP_SCP_Quick_Reference_Guide_v2.pdf
- <https://www.softwaretestinghelp.com/guidelines-for-secure-coding/>
- <http://security.cs.rpi.edu/courses/binexp-spring2015/>
- <https://www.ired.team/>



Lập trình an toàn & Khai thác lỗ hổng phần mềm



Trường ĐH CNTT TP. HCM