DevSecOps 101

(ou tudo que você tem que saber pra começar)

Olá, eu sou o Fausto :)

2008 / IBM

Tecnologia em Software Livre (2006)

2010 / CTI Renato Archer

MBA em Gestão de Segurança da Informação (2015)

2016 / Techbiz Forense Digital

MBA em Arquitetura de Software (2022)

2018 / Embraer

2020 / RNP

carro véio/moto barulhenta/boteco/rock/ velocidade/churrasco

Teoria + Prática

Não, isso não será só um curso passa-slide.

Não existe bala de prata

Ninguém está a salvo - mas não é fim do mundo.

Risco não mapeado é risco assumido

Depois não adianta chorar.

Agenda

Teoria (2h)

- Shift Left
- Framework CALMS
- Systems Development Life Cycle – SDLC
- Security Software Development Lifecycle – SSDLC
- Frameworks
- Automatização de Testes
- Continuous Integration
- DevOps & DevSecOps
- Pipelines
- Integrações

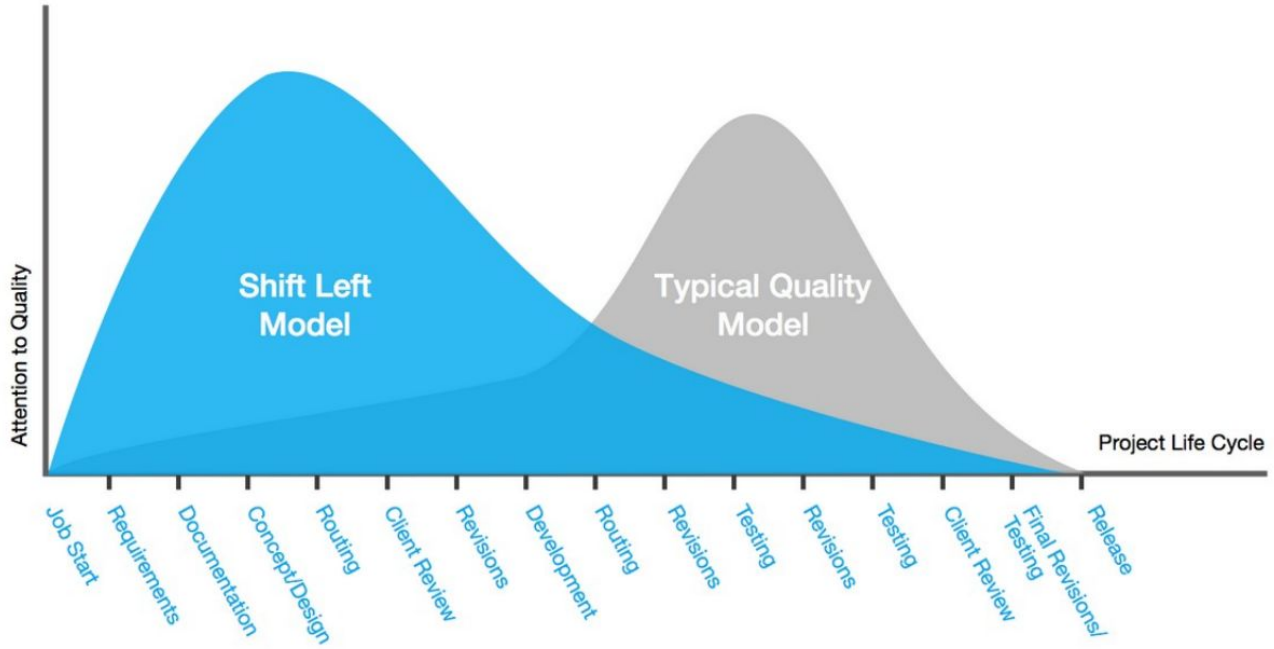
Prática (2h)

- Análise estática de código - SAST
- Escopo
- O que é analisado?
- Como é analisado
- Escopo dos testes
- O que é apresentado após a análise?
- Como utilizar os resultados?

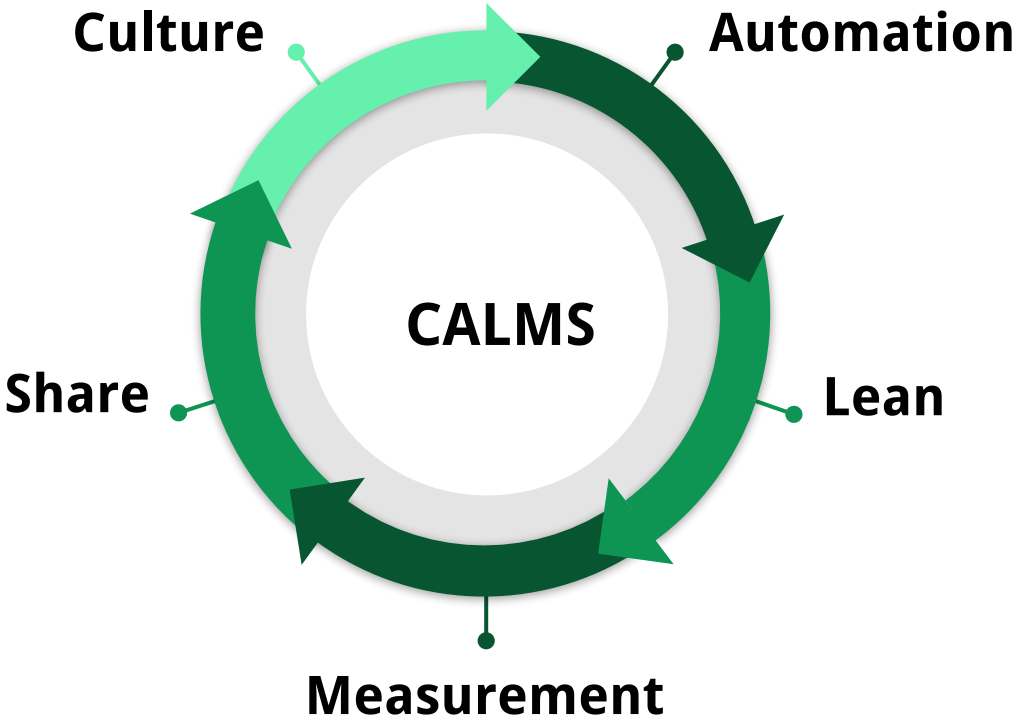
Eu acredito em 50/50

(em outras palavras, não seja um piloto de ferramenta.)

Shift Left

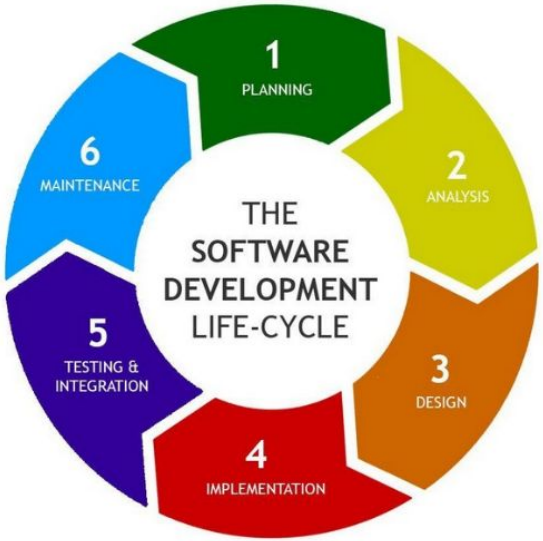

Attention to Quality
Shift Left Model
Typical Quality Model
Project Life Cycle
Job Start
Requirements
Documentation
Concept/Design
Routing
Client Review
Revisions
Development
Routing
Revisions
Testing
Revisions
Testing
Client Review
Final Revisions/ Testing
Release

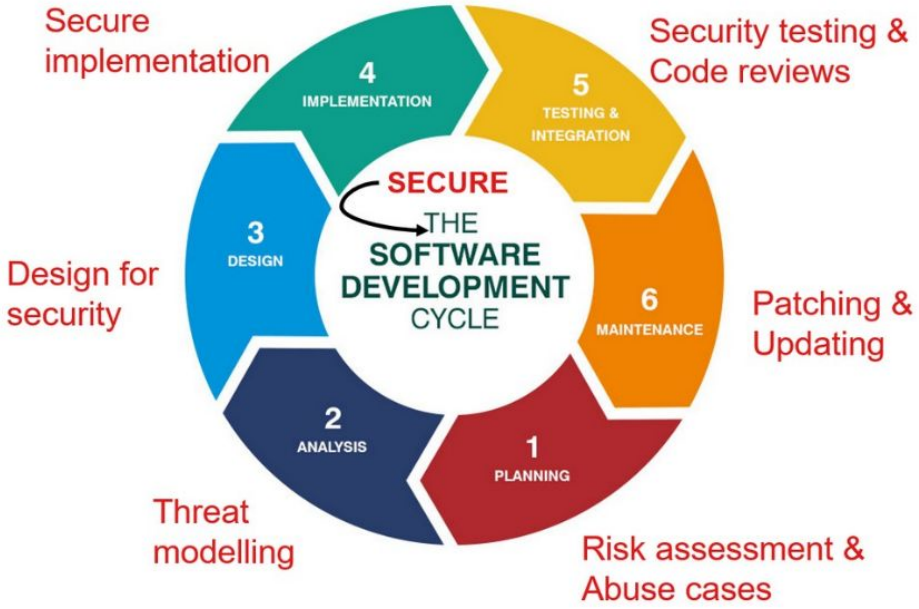
CALMS Framework


Culture
Automation
CALMS
Share
Lean
Measurement

SDLC
Systems Development Life Cycle

SSDLC
Security Software Development Lifecycle


1
PLANNING
2
ANALYSIS
3
DESIGN
4
IMPLEMENTATION
5
TESTING &
INTEGRATION
6
MAINTENANCE
THE
SOFTWARE
DEVELOPMENT
LIFE-CYCLE


Secure
implementation
Security testing &
Code reviews
Design for
security
Patching &
Updating
Threat
modelling
Risk assessment &
Abuse cases
4
IMPLEMENTATION
5
TESTING &
INTEGRATION
3
DESIGN
6
MAINTENANCE
2
ANALYSIS
1
PLANNING
SECURE
THE
SOFTWARE
DEVELOPMENT
CYCLE

Frameworks

(e outras cositas más)

Abordagens Práticas

- Microsoft Security Development Lifecycle (SDL)
- OWASP SAMM

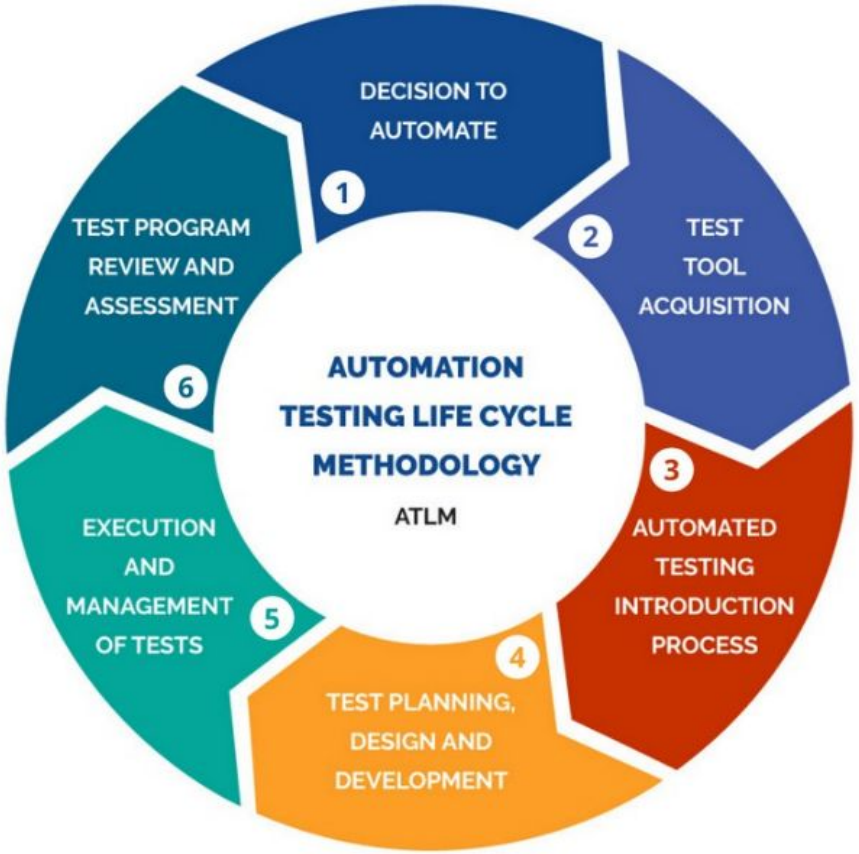
Referências

- NIST SP 800-160 Vol. 1
- NIST SP 800-160 Vol. 2
- ISO/IEC/IEEE 15288:2015

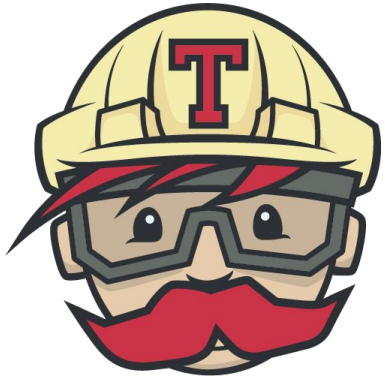
Coisa véia (mas boa)

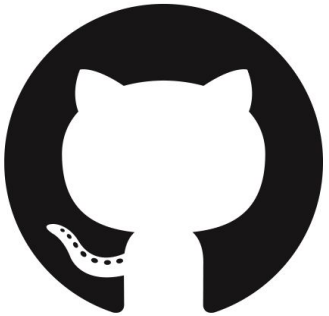
- NIST SP 800-64 Rev. 2
- OWASP CLASP

Automatização de Testes

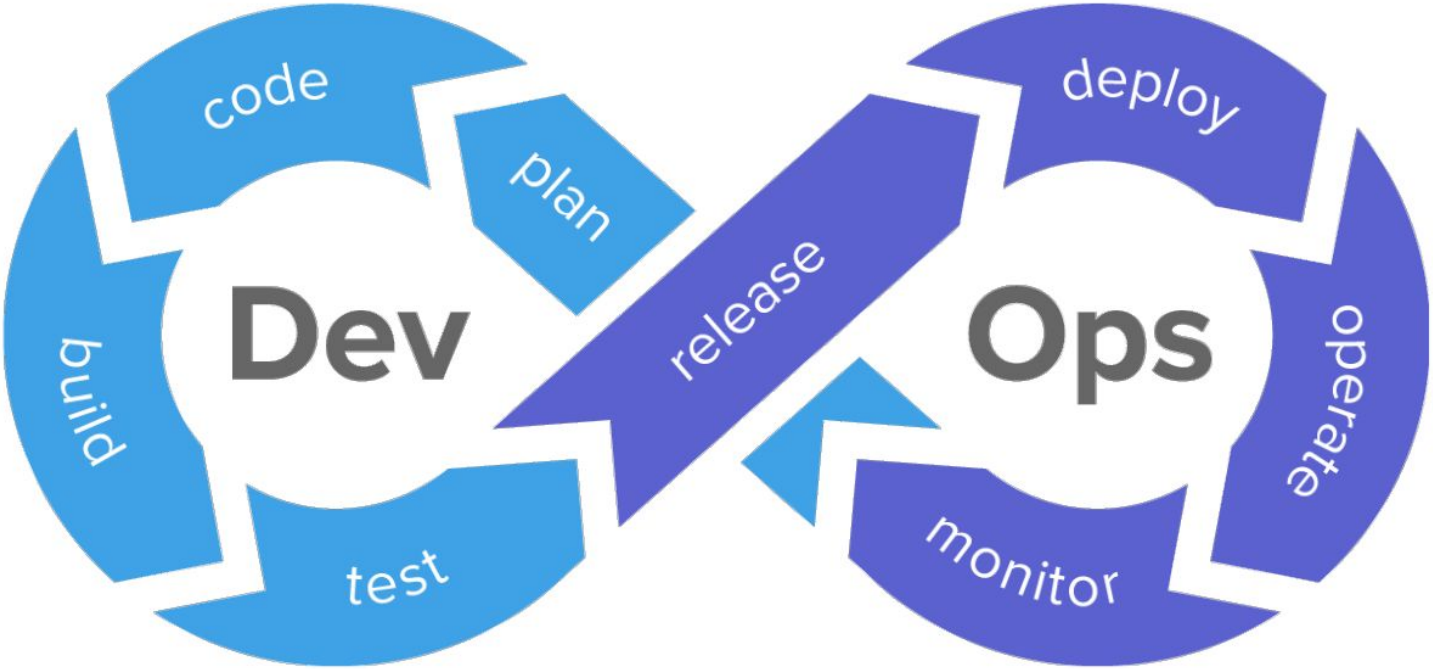

DECISION TO AUTOMATE
1
TEST TOOL ACQUISITION
2
AUTOMATED TESTING INTRODUCTION PROCESS
3
TEST PLANNING, DESIGN AND DEVELOPMENT
4
EXECUTION AND MANAGEMENT OF TESTS
5
TEST PROGRAM REVIEW AND ASSESSMENT
6
AUTOMATION TESTING LIFE CYCLE METHODOLOGY
ATLM

Continuous Integration

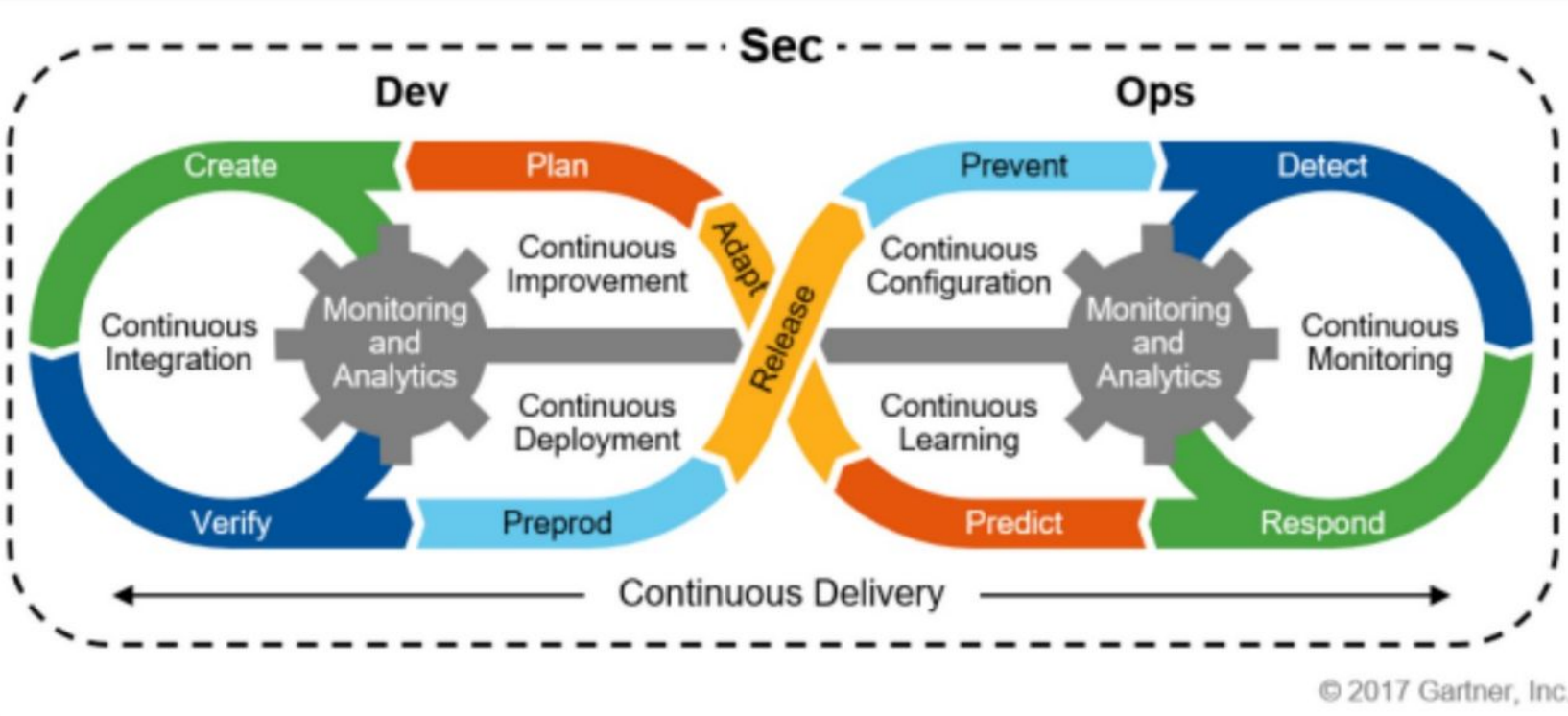




DevOps & DevSecOps

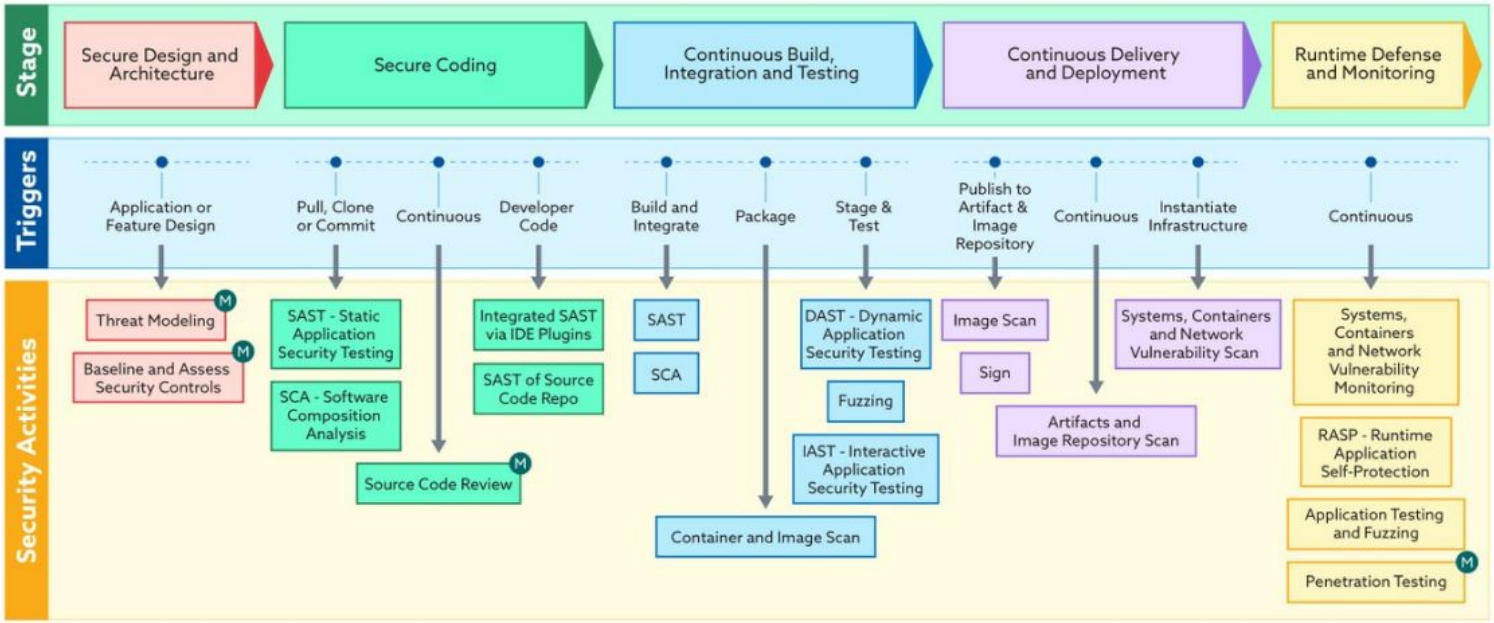

code
plan
build
test
release
deploy
operate
monitor
Dev
Ops


code
deploy
test
monitor
EU NÃO AGUENTO MAIS ISSO!



Bonito seu discurso...
...mas, e na prática?

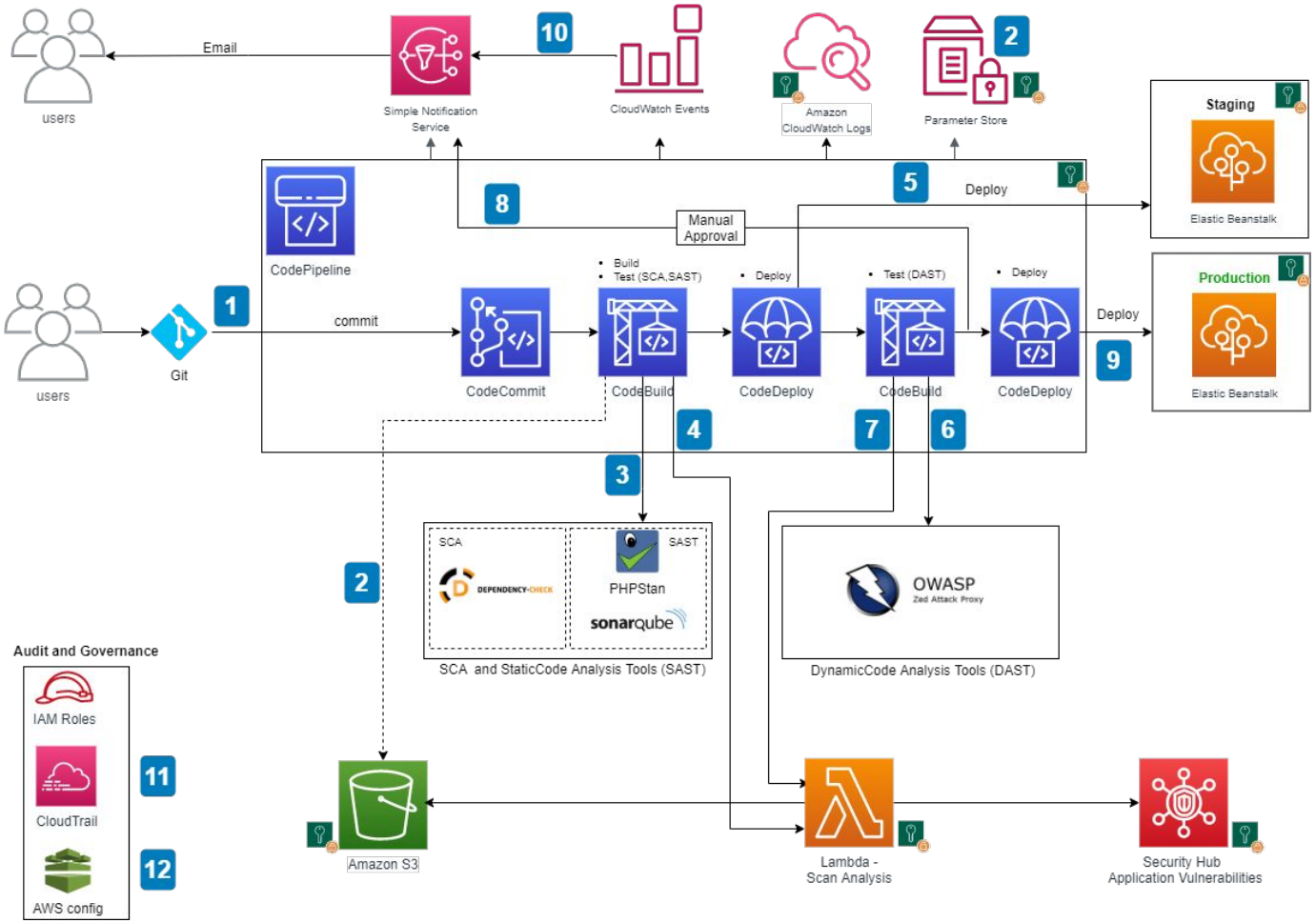
Secure Development Lifecycle - Policies, Standards, Controls and Best Practices

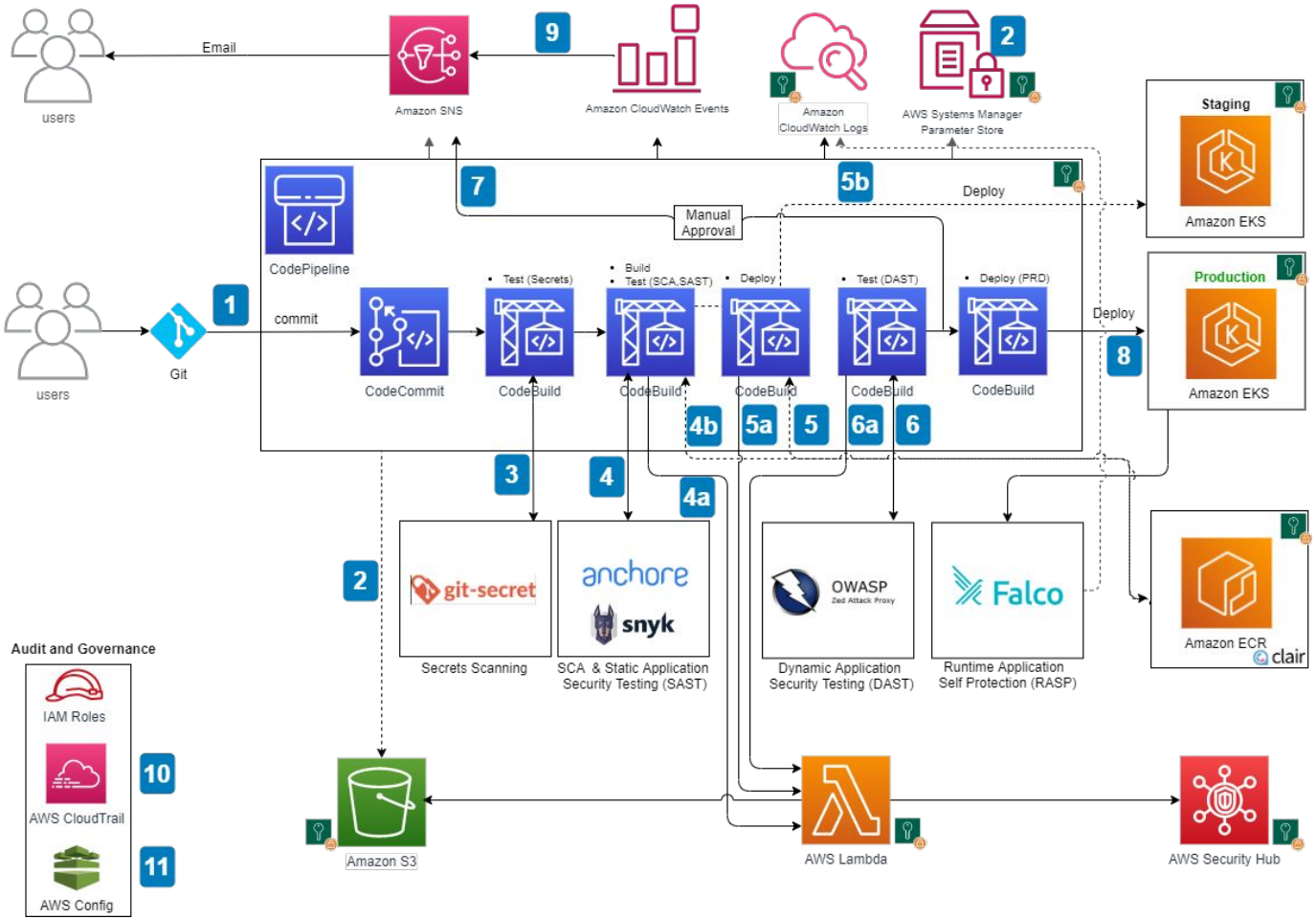


M Manually Performed

**Though the visual gives an impression of a linear flow from one stage to another, a bidirectional feedback loop exists between stages.*

Pipelines

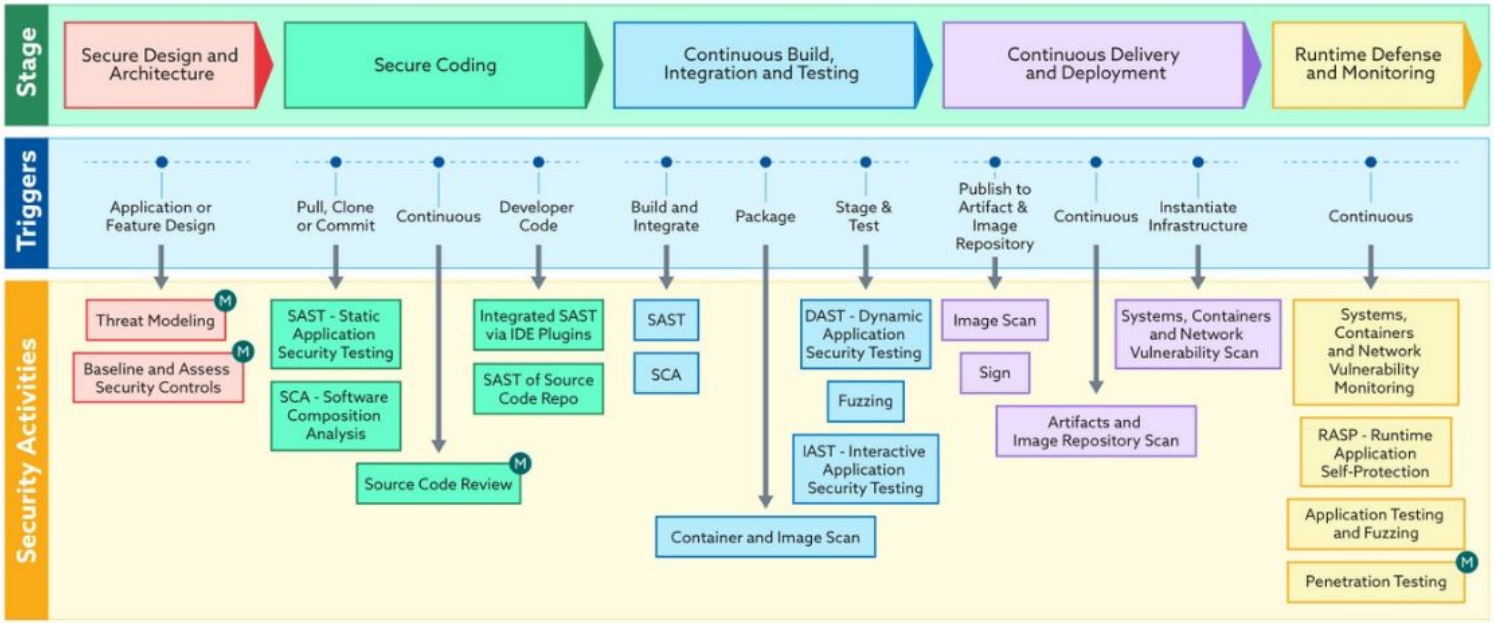

users
Email
Simple Notification Service
10
CloudWatch Events
Amazon CloudWatch Logs
2
Parameter Store
Staging
Elastic Beanstalk
CodePipeline
8
Manual Approval
5
Deploy
users
Git
1
commit
CodeCommit
Build
Test (SCA,SAST)
CodeBuild
Deploy
CodeDeploy
Test (DAST)
CodeBuild
Deploy
CodeDeploy
Deploy
9
Production
Elastic Beanstalk
4
7
6
3
SCA
SAST
DEPENDENCY-CHECK
PHPStan
sonarqube
SCA and StaticCode Analysis Tools (SAST)
OWASP
Zed Attack Proxy
DynamicCode Analysis Tools (DAST)
2
Audit and Governance
IAM Roles
CloudTrail
11
AWS config
12
Amazon S3
Lambda - Scan Analysis
Security Hub Application Vulnerabilities


users
Email
Amazon SNS
9
Amazon CloudWatch Events
Amazon CloudWatch Logs
2
AWS Systems Manager Parameter Store
Staging
Amazon EKS
7
5b
Deploy
Manual Approval
CodePipeline
users
1
Git
commit
CodeCommit
Test (Secrets)
CodeBuild
Build
Test (SCA,SAST)
CodeBuild
Deploy
CodeBuild
Test (DAST)
CodeBuild
Deploy (PRD)
CodeBuild
Deploy
8
Production
Amazon EKS
4b
5a
5
6a
6
3
4
4a
2
git-secret
Secrets Scanning
anchore
snyk
SCA & Static Application Security Testing (SAST)
OWASP
Zed Attack Proxy
Dynamic Application Security Testing (DAST)
Falco
Runtime Application Self Protection (RASP)
Amazon ECR
clair
Audit and Governance
IAM Roles
AWS CloudTrail
10
AWS Config
11
Amazon S3
AWS Lambda
AWS Security Hub

Considerações

Oficina

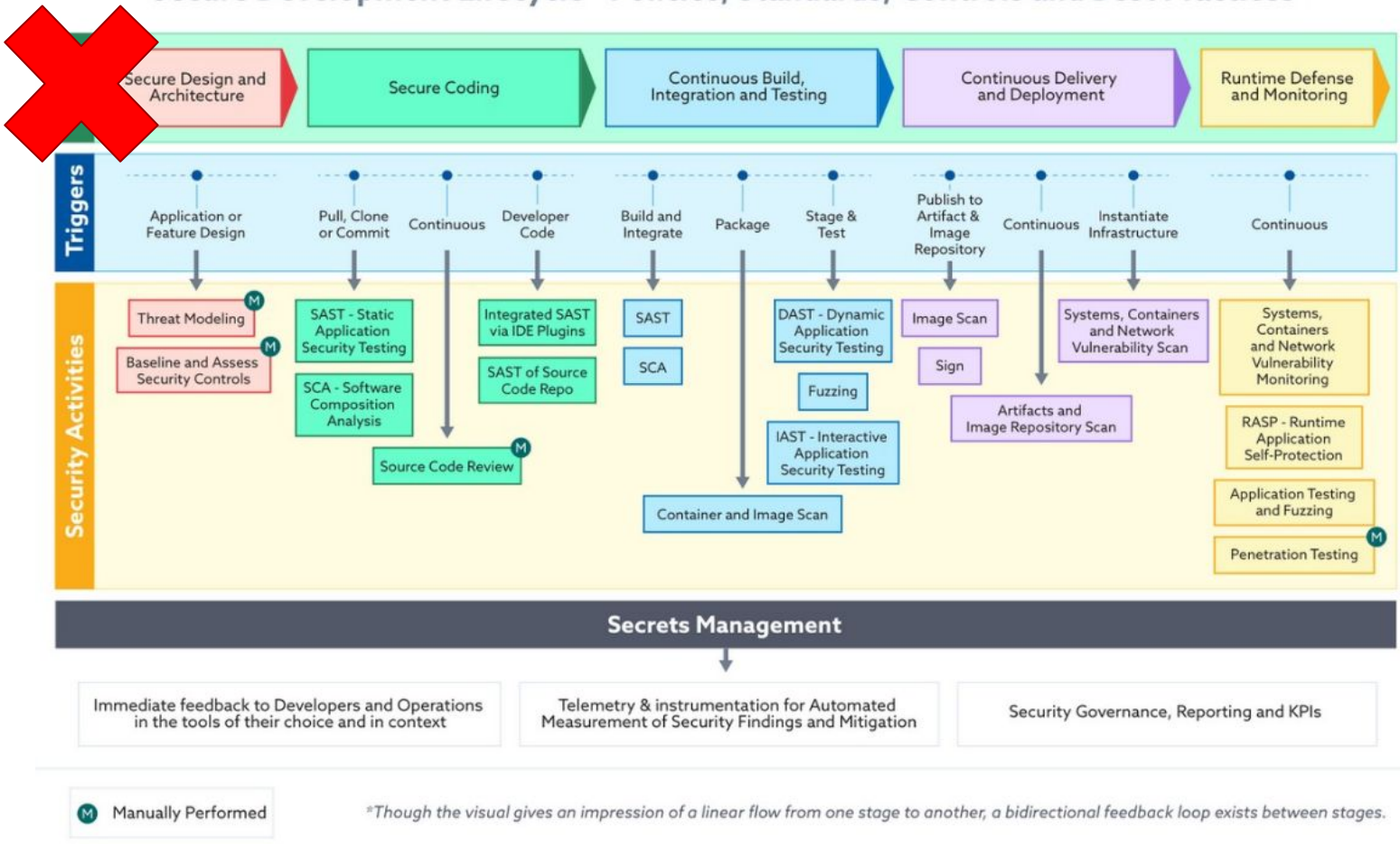
Secure Development Lifecycle - Policies, Standards, Controls and Best Practices



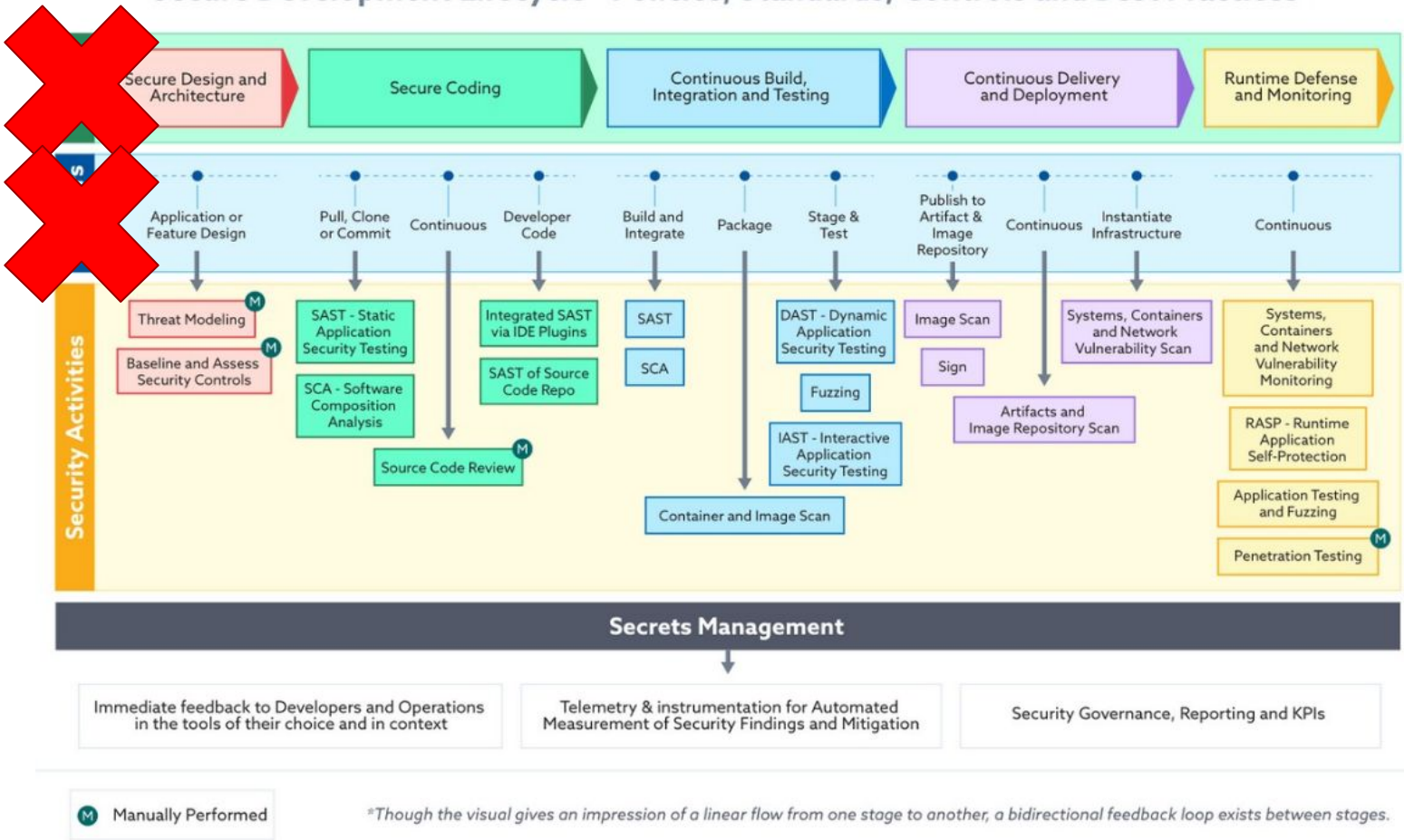
M Manually Performed

**Though the visual gives an impression of a linear flow from one stage to another, a bidirectional feedback loop exists between stages.*

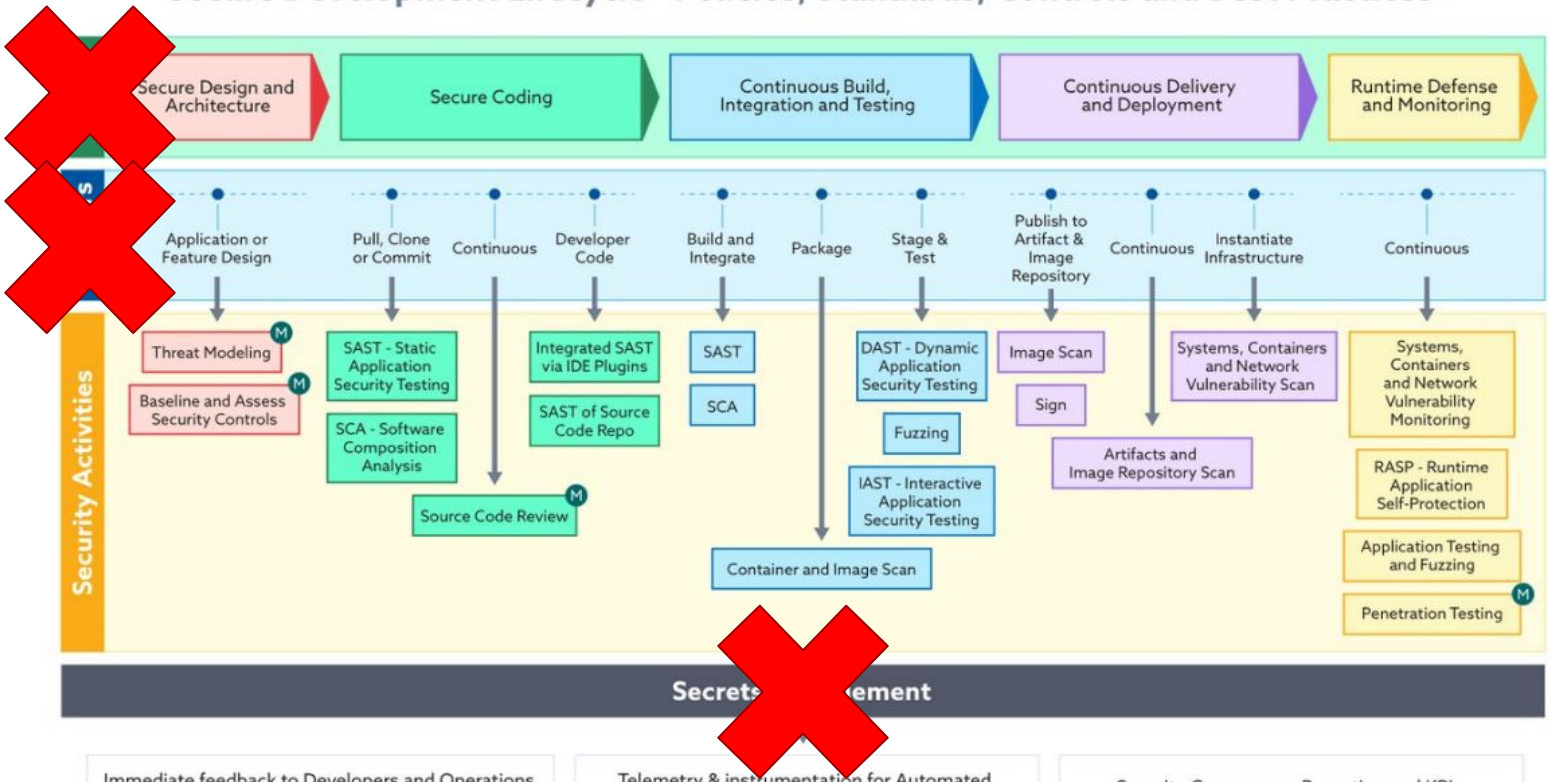
Secure Development Lifecycle - Policies, Standards, Controls and Best Practices



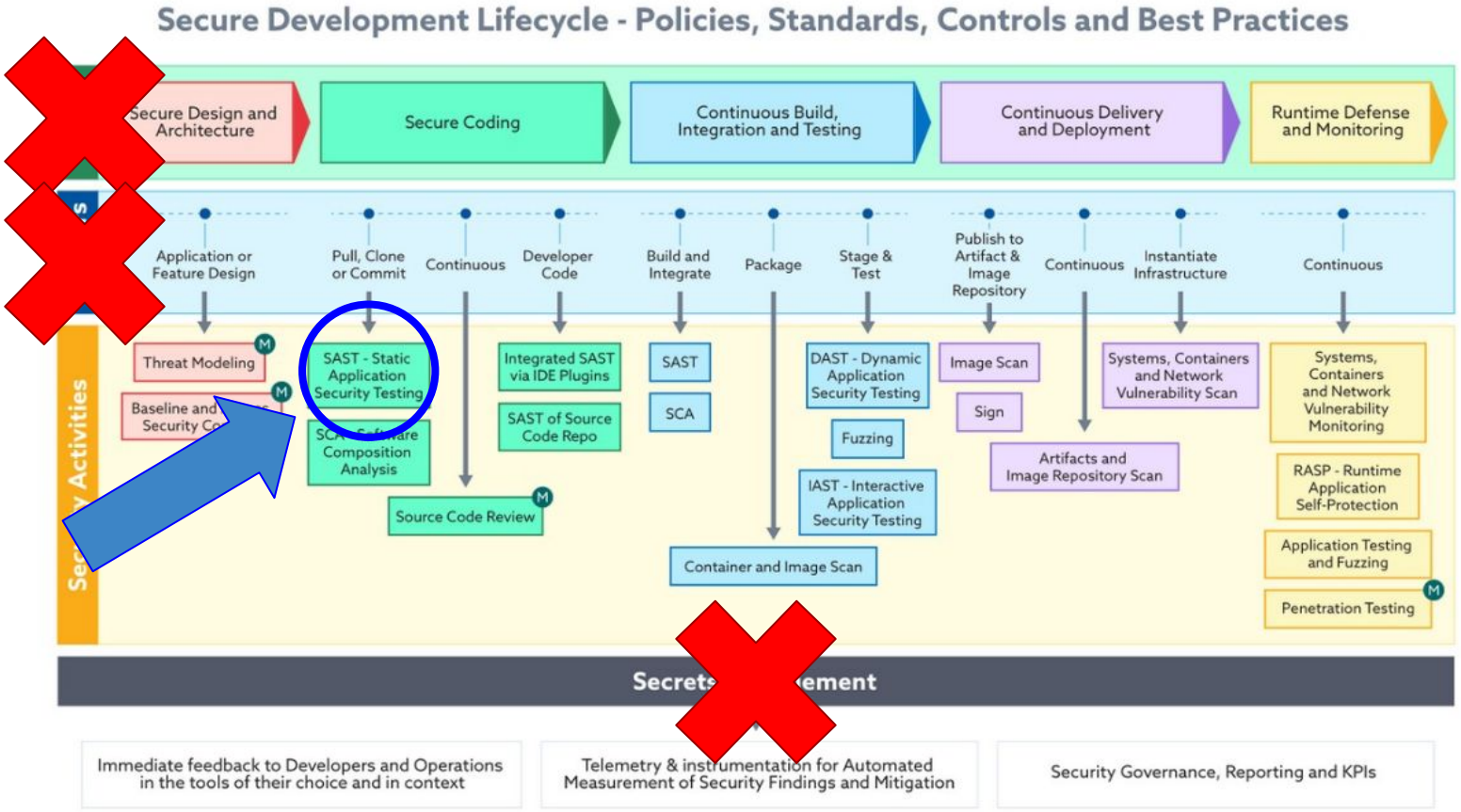
Secure Development Lifecycle - Policies, Standards, Controls and Best Practices



Secure Development Lifecycle - Policies, Standards, Controls and Best Practices



*Though the visual gives an impression of a linear flow from one stage to another, a bidirectional feedback loop exists between stages.


Secure Development Lifecycle - Policies, Standards, Controls and Best Practices
Secure Design and Architecture
Secure Coding
Continuous Build, Integration and Testing
Continuous Delivery and Deployment
Runtime Defense and Monitoring
Application or Feature Design
Pull, Clone or Commit
Continuous
Developer Code
Build and Integrate
Package
Stage & Test
Publish to Artifact & Image Repository
Continuous
Instantiate Infrastructure
Continuous
Threat Modeling
SAST - Static Application Security Testing
Integrated SAST via IDE Plugins
SAST of Source Code Repo
SAST
SCA
DAST - Dynamic Application Security Testing
Fuzzing
IAST - Interactive Application Security Testing
Image Scan
Sign
Artifacts and Image Repository Scan
Systems, Containers and Network Vulnerability Scan
Systems, Containers and Network Vulnerability Monitoring
RASP - Runtime Application Self-Protection
Application Testing and Fuzzing
Penetration Testing
Source Code Review
Container and Image Scan
Immediate feedback to Developers and Operations in the tools of their choice and in context
Telemetry & instrumentation for Automated Measurement of Security Findings and Mitigation
Security Governance, Reporting and KPIs
Manually Performed
*Though the visual gives an impression of a linear flow from one stage to another, a bidirectional feedback loop exists between stages.

SAST
Static Application Security Testing

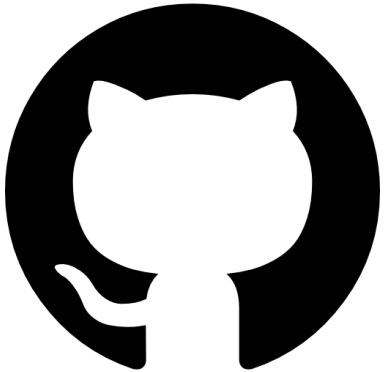
SAST
Static Application Security Testing

código fonte | estática | repositório | IDE

SAST
Static Application Security Testing

código fonte | estática | repositório | IDE
Não testa binários e pacotes. Só fonte!

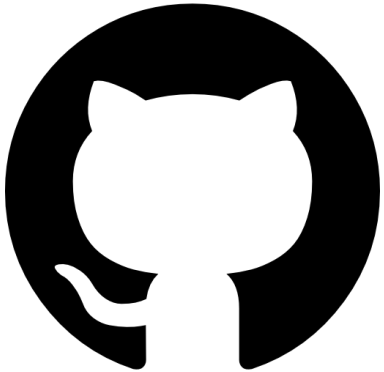
Github



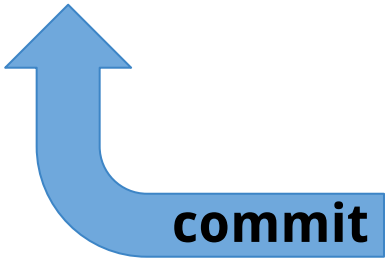
SonarCloud

Você

Github

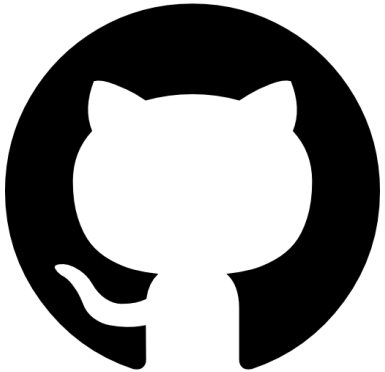


SonarCloud



Você

Github

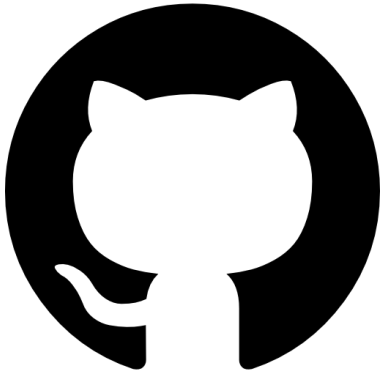


integração

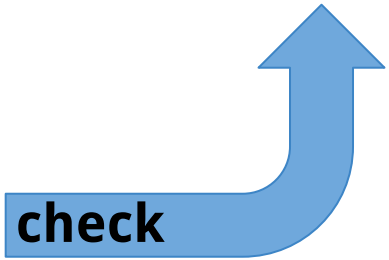
SonarCloud

Você

Github

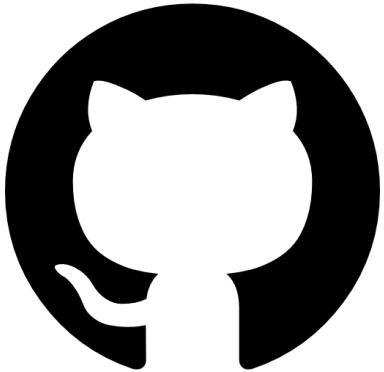


SonarCloud

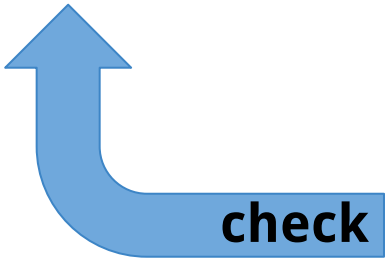


Você

Github



SonarCloud



Você

Dúvidas?

Interlude

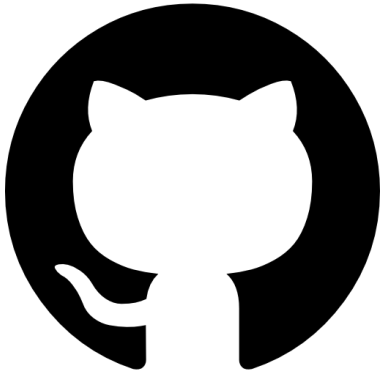
DevSecOps 101

(ou tudo que você tem que saber pra começar)

Episódio II: Lab de SAST

Oficina

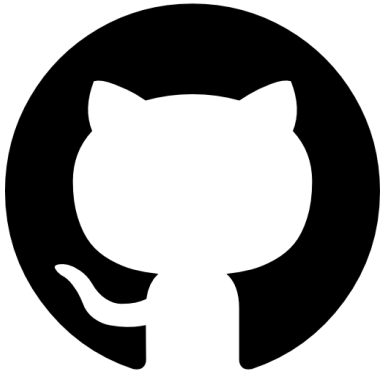
Github



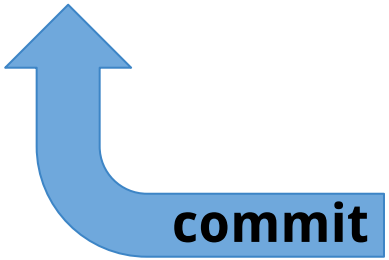
SonarCloud

Você

Github

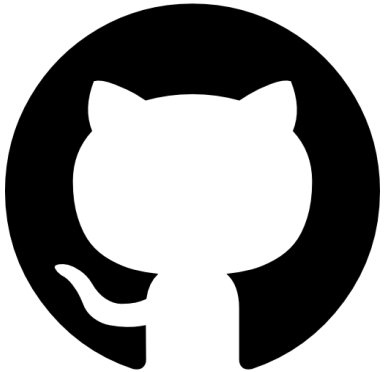


SonarCloud



Você

Github



integração

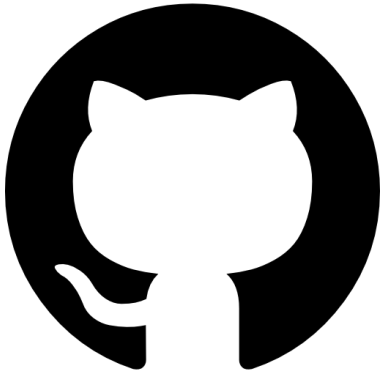
SonarCloud

Você

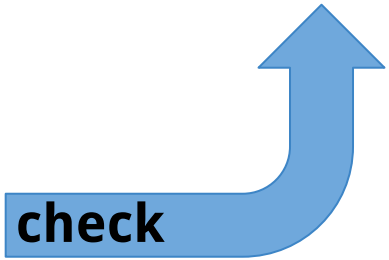
Integração sem CI?

- Feature “Applications”: serviços parceiros, por via sistêmica - algo como um webhook
- O repositório deve ser público: não é possível integrar com repos privados
- Atenção: serviço de SAST disponível gratuitamente para projetos Open Source!

Github

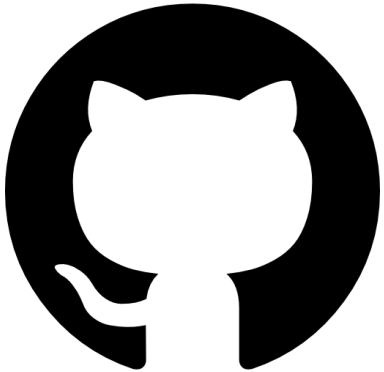


SonarCloud

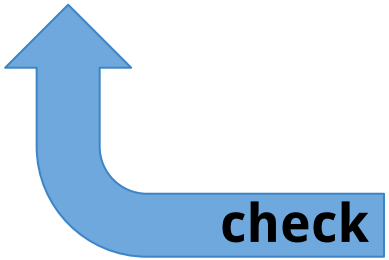


Você

Github



SonarCloud



Você

Considerações

- uma única branch (main)
- cinco commits
  - inicialização do repo
  - código vulnerável
  - correções de segurança
  - correções de security hotspots
  - correções de code smells
  - correções de bugs
- não utilizamos feature branch (desculpa)
- Não precisa codar: vamos de fetch/reset
- Dá pra rodar a app local, mas não precisa

Bora!

Passo 1: Obtendo o código

- **Crie seu diretório do projeto**

```
mkdir labsast && cd $_
```

- **Inicialize o repo**

```
git init
```

- **Defina o remote**

```
git remote add origin https://github.com/ffilho/devsecops101_sastlab
```

- **Defina o fetch**

```
git fetch origin 0e5e5a08f6c5fe3dc0fa7bf7f60c57778364dfc5
```

Passo 1: Obtendo o código

- **Hard reset**

```
git reset --hard FETCH_HEAD
```

Passo 1.1: Executando a aplicação

- **Crie seu virtual environment**

```
virtualenv venv
```

- **Ative o venv**

```
source venv/bin/activate
which python
```

- **Instale as dependências**

```
pip install -r requirements.txt
```

- **Execute a app**

```
FLASK_APP=app.py flask run
```

- **Acesse a app**

```
http://127.0.0.1:5000/
```

Passo 2: Configurando o seu remoto

- Crie um repo público na sua conta
- Copie o endereço do repo
- Altere o remote do seu projeto

```
git remote -v
git remote set-url origin seurepo
git remote -v
```

Passo 3: Integração do SonarCloud

- Faça com o Github em https://sonarcloud.io/sessions/new

  - Criação de Organization
  - Vínculo do Github com o Sonar
  - Criação de projeto
  - Plano de billing do projeto
  - Habilitação da Automatic Analysis

Passo 4: Primeiro commit com análise SAST

- **Prepare o commit**

```
git branch -M main
git add .
git commit -m "Primeiro commit:
codigo vulneravel"
git push -u origin main
```

- **Confira o resultado no Github**
- **Confira o resultado no SonarCloud**

Passo 5: A interface do SonarCloud

- Banner de erro (?)
- Quality Gate
- Bugs
- Vulnerabilities
- Security Hotspots
- Code Smells
- Debt
- Ajuda do produto

Passo 6: Analisando vulns

- Listagem de vulnerabilidades
- Snippet de código
- Interface de uso do console de vulns
- Why is this an issue?
- Base de conhecimento
- Tipo de detecção (ícone do cadeado)
- Tipo de vulnerabilidade (ícone da seta)
- Tracking (ícone em círculo)
- Colaboração (avatar do dev)

Passo 7: Analisando Security Hotspots

- Entendendo o risco
- Aceitando o risco
- Corrigindo o problema

Passo 8: Rules de SAST

- Contexto da Organização
- Cobertura por linguagens
- Criando suas regras:
  - https://docs.sonarqube.org/latest/user-guide/rules/
  - https://docs.sonarqube.org/latest/extend/adding-coding-rules/

Passo 9: Primeiro fix Vulnerabilities

- **Fetch/reset no commit deste passo**

```
git fetch https://github.com/ffilho/devsecops101_sastlab/357c240bec4edbbe8e37f0e1e925698819492309
git reset --hard FETCH_HEAD
```

- **Commit no seu repo**

```
git push -u origin main
```

- **Interface do SonarCloud quando em análise**
- **Interface do Github após a análise**
- **Indicadores de cobertura**
- **Relatórios customizados**

Passo 10: Segundo fix Security Hotspots

- **Fetch no commit deste passo**

```
git fetch https://github.com/ffilho/devsecops101_sastlab/381c6da77bb1ecb6d1a6ab06357f92e479963092
git reset --hard FETCH_HEAD
```

- **Commit no seu repo**

```
git push -u origin main
```

- **Indicadores de cobertura**
- **Declínio em qualidade**
- **Interface do módulo de SH**

Passo 11: Terceiro fix Code smells

- **Fetch no commit deste passo**

```
git fetch https://github.com/ffilho/devsecops101_sastlab/31ab3602b8df78dc8e871faeeedd11dbe1f1cfd6
git reset --hard FETCH_HEAD
```

- **Commit no seu repo**

```
git push -u origin main
```

- **Indicadores de cobertura**
- **Interface do módulo de code smells**
- **Falsos positivos & colaboração**
- **Indicadores de cobertura**

Passo 12: Quarto fix Bugs

- **Fetch no commit deste passo**

```
git fetch https://github.com/ffilho/devsecops101_sastlab/1ff278054b93e932ba8789c7b5e4c2c86c6b29c9
git reset --hard FETCH_HEAD
```

- **Commit no seu repo**

```
git push -u origin main
```

- **Indicadores de cobertura**
- **Interface do Sonar**
- **Interface do Github**
- **Badges do projeto**

Passo 13: Conclusão

- Histórico de atividades
- Métricas, KPIs e gráficos bonitos
- Apagando sua Organization
- Apagando seu projeto

Opa, pera aí...

Será que agora o código é seguro o suficiente?

```
query = "SELECT name, phone FROM users
WHERE name =
'"+str(request.form.get('output'))+"';"
```

```
SELECT name, phone FROM users WHERE
name = 'Fausto';
```

```
FLASK_APP=cais.py flask run
* Serving Flask app app.py' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production
deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [27/May/2021 19:56:56] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/May/2021 19:56:58] "GET /consulta HTTP/1.1" 200 -
SELECT name, phone FROM users WHERE name = 'F';
127.0.0.1 - - [27/May/2021 19:57:02] "POST /consulta HTTP/1.1" 200 -
SELECT name, phone FROM users WHERE name = 'Fa';
127.0.0.1 - - [27/May/2021 19:57:05] "POST /consulta HTTP/1.1" 200 -
SELECT name, phone FROM users WHERE name = 'Fau';
127.0.0.1 - - [27/May/2021 19:57:10] "POST /consulta HTTP/1.1" 200 -
SELECT name, phone FROM users WHERE name = 'Faus';
127.0.0.1 - - [27/May/2021 19:57:13] "POST /consulta HTTP/1.1" 200 -
SELECT name, phone FROM users WHERE name = 'Faust';
127.0.0.1 - - [27/May/2021 19:57:16] "POST /consulta HTTP/1.1" 200 -
```

```
query = "SELECT name, phone FROM users
WHERE name =
'"+str(request.form.get('output'))+"';"

' OR 1=1 --;

SELECT name, phone FROM users WHERE
name = '' OR 1=1 --;
```

```
FLASK_APP=cais.py flask run
* Serving Flask app app.py' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production
deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [27/May/2021 19:59:25] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/May/2021 19:59:41] "GET /consulta HTTP/1.1" 200 -
SELECT name, phone FROM users WHERE name = '' or 1=1 --';
127.0.0.1 - - [27/May/2021 19:59:49] "POST /consulta HTTP/1.1" 200 -
```


HACKERMAN
HM

wtf?

SQL Injection

Contributor(s): kingthorin

Overview

A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands.

Threat Modeling

- SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server.
- SQL Injection is very common with PHP and ASP applications due to the prevalence of older functional interfaces. Due to the nature of programmatic interfaces available, J2EE and ASP.NET applications are less likely to have easily exploited SQL injections.
- The severity of SQL Injection attacks is limited by the attacker's skill and imagination, and to a lesser extent, defense in depth countermeasures, such as low privilege connections to the database server and

No Silver Bullet
—Essence and Accident in Software Engineering

Frederick P. Brooks, Jr.
University of North Carolina at Chapel Hill

There is no single development, in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity.

No Silver Bullet
—Essence and Accident in Software Engineering

Frederick P. Brooks, Jr.
University of North Carolina

There is no single development, in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity.

1986

SAST + DAST / IAST + RASP

SAST + DAST / IAST + RASP

e ainda assim não é muita coisa

Obrigado! :)

Fausto Filho (ff)

https://www.linkedin.com/in/faustoafilho/