DevOps Fundamentals

Secure DevOps

Agenda

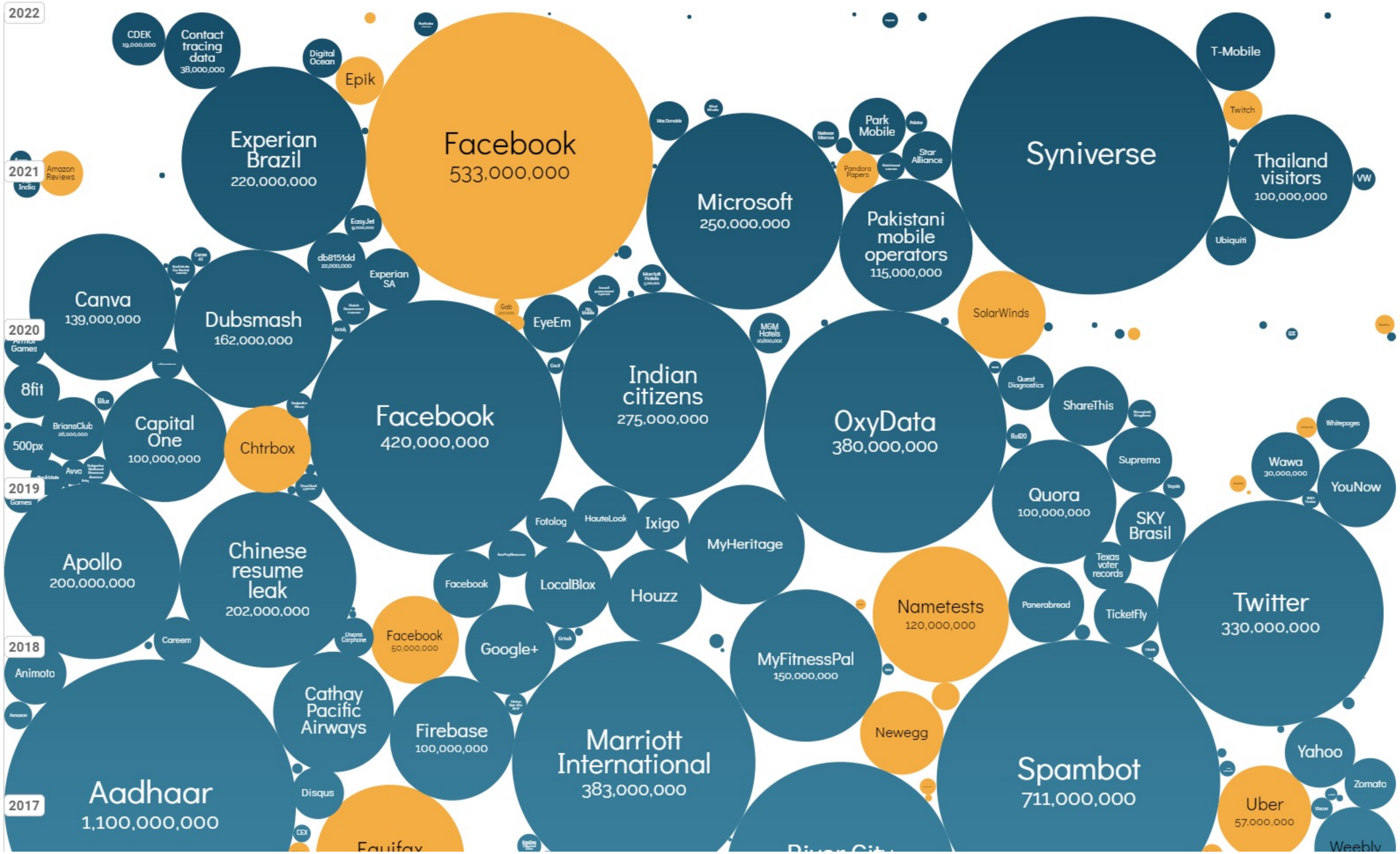
Why Secure DevOps?

Secure DevOps Practices

moOngy.

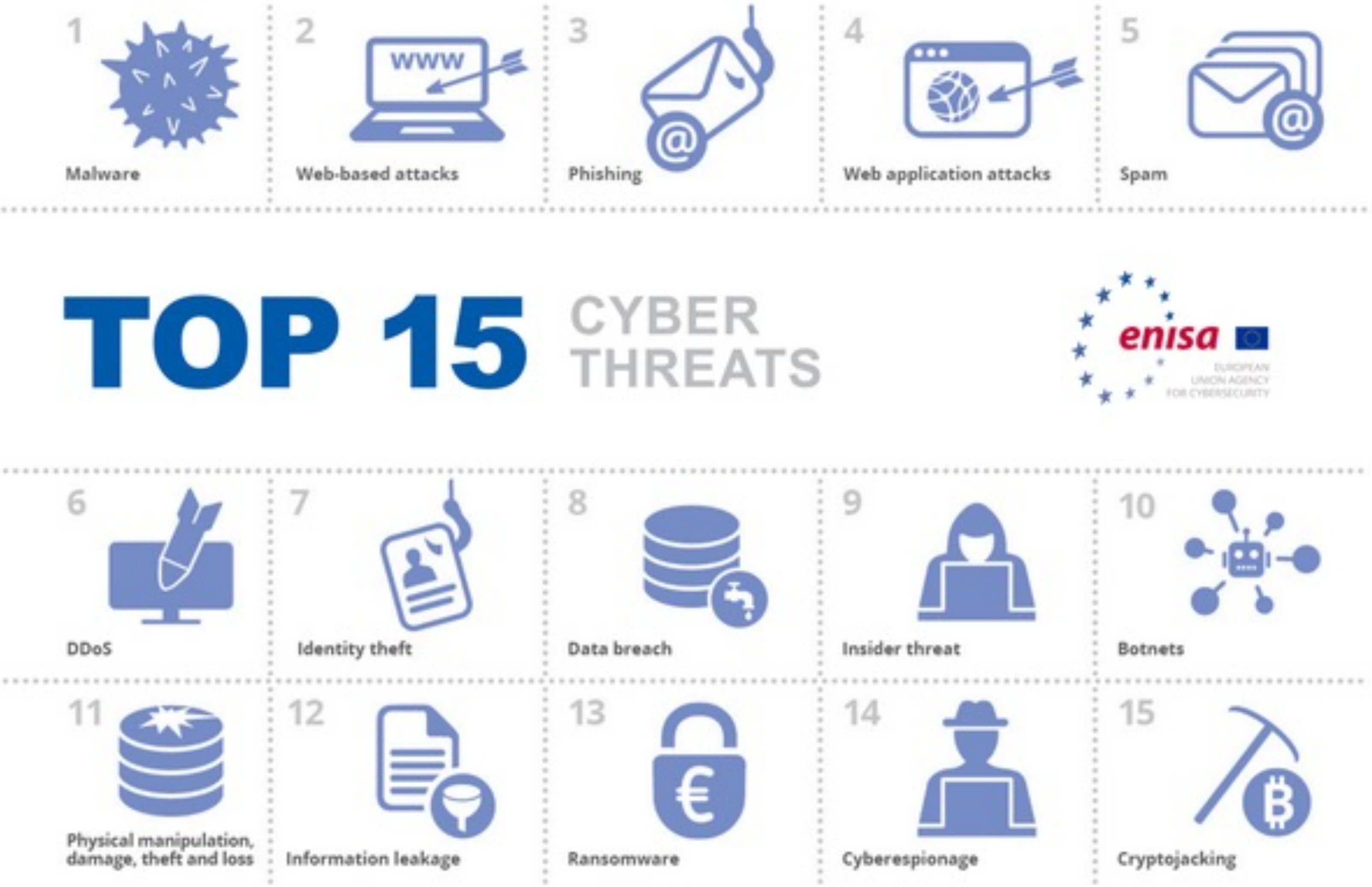
Why Secure DevOps?

World's Biggest Data Breaches & Hacks



[World’s Biggest Data Breaches & Hacks](#)

Top 15 Cyber Threats



TOP 15 CYBER THREATS

enisa

EUROPEAN UNION AGENCY FOR CYBERSECURITY

1 Malware

2 Web-based attacks

3 Phishing

4 Web application attacks

5 Spam

6 DDoS

7 Identity theft

8 Data breach

9 Insider threat

10 Botnets

11 Physical manipulation, damage, theft and loss

12 Information leakage

13 Ransomware

14 Cyberespionage

15 Cryptojacking

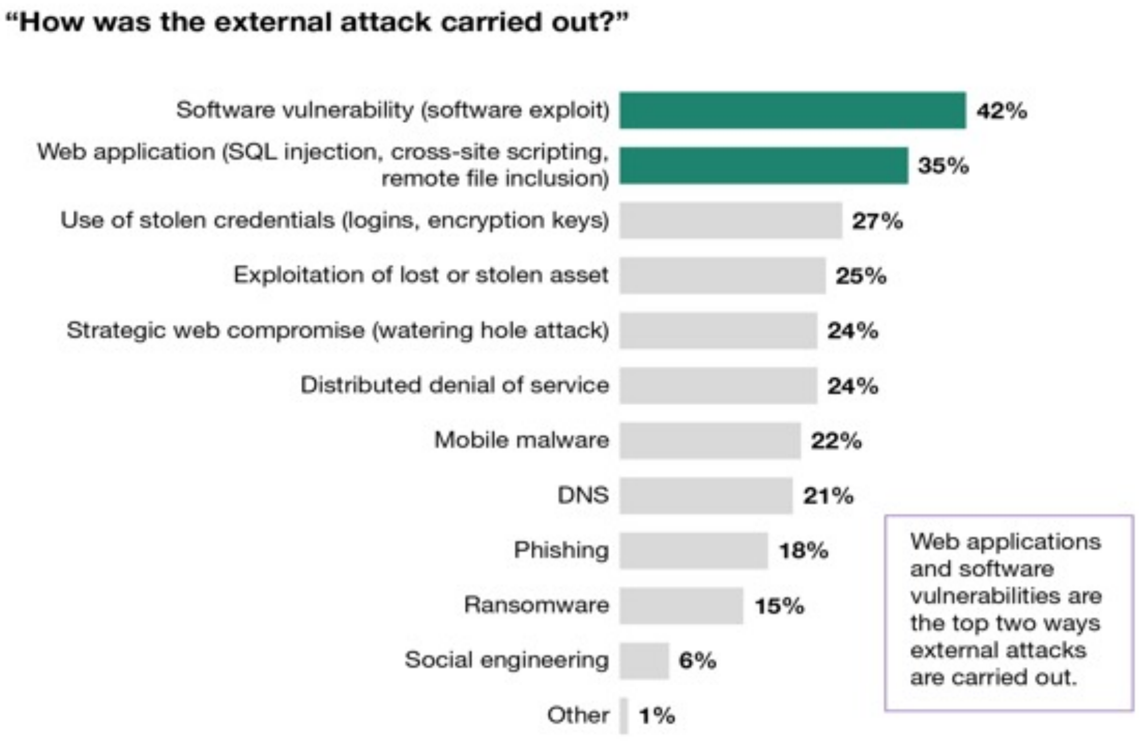
Application as attack vector

Applications remains the most common attack vector

Implementing security means be secure in several layers to make it harder to be breached

Crucial to understand and control all attack vectors

"You are only as secure as your weakest link"



Web applications and software vulnerabilities are the top two ways external attacks are carried out.

Assume Breach

“Fundamentally, if somebody wants to get in, they're getting in…accept that. What we tell clients is: Number one, you're in the fight, whether you thought you were or not. Number two, you almost certainly are penetrated.”

Michael Hayden, CIA

“There are only two types of companies: those that have been hacked, and those that will be.”

Robert Mueller, FBI Director

You need to change your mindset (and your tools) to practice this approach.

First, you need to clearly understand your security vulnerabilities!

Impact caused by a security breach

Costs and efforts related with response and notification

Lost employee productivity

Lawsuits and settlements

Regulatory fines and response

Cost of fixing infrastructure

Brand recovery costs & liabilities

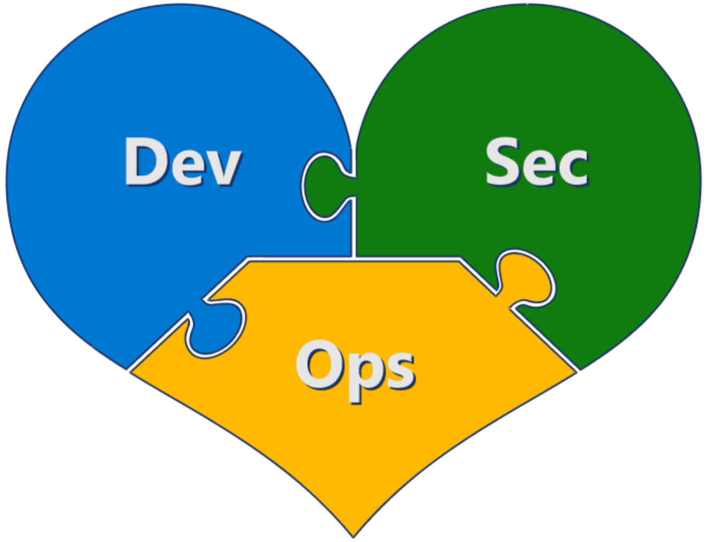
What is security?

“Security is application of technologies, processes, and controls to protect systems, networks, programs, devices, and data from unauthorized access or criminal use.”

Security provides **confidentiality**, **integrity**, and **availability** assurances against deliberate attacks and abuse of valuable data and systems.

What is Secure DevOps?

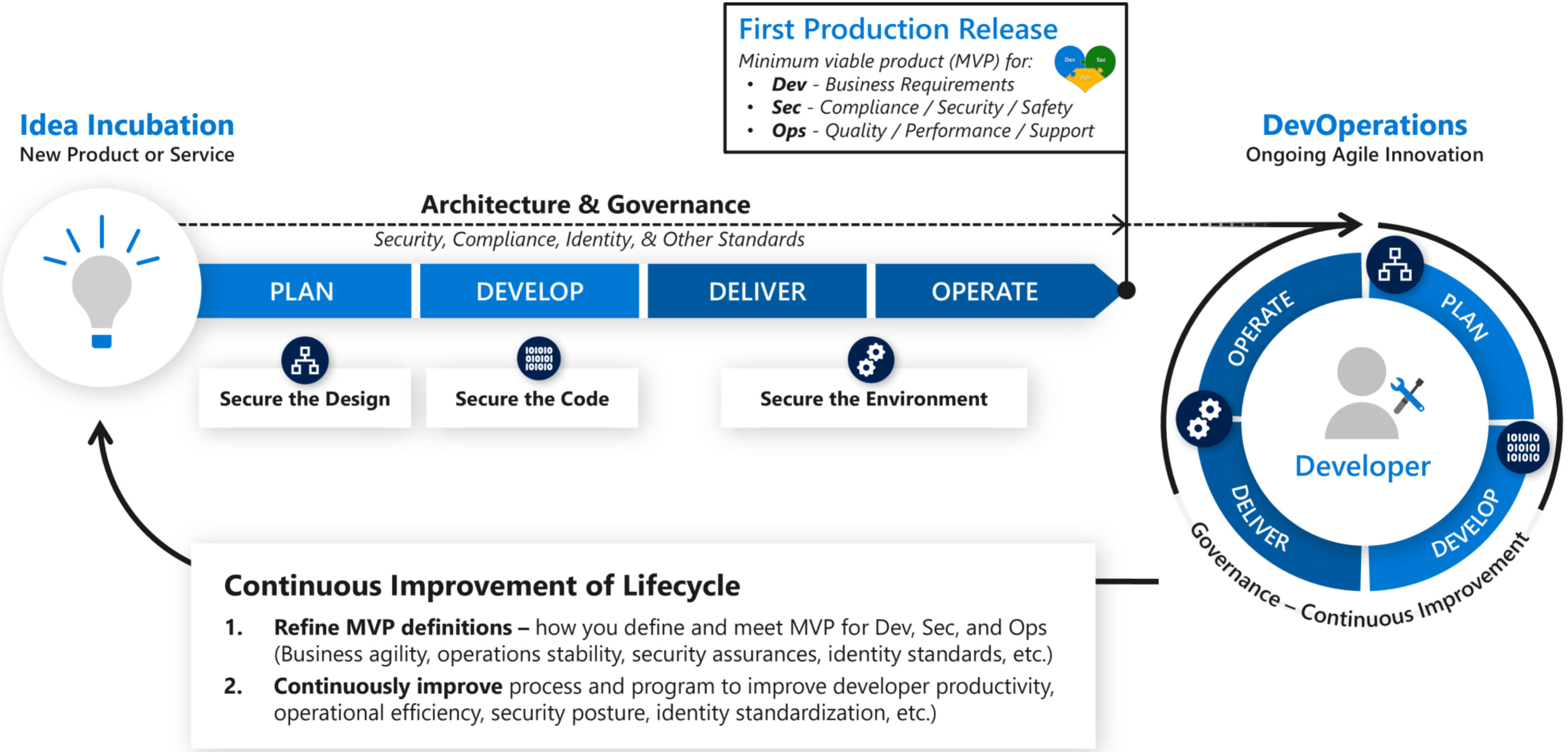
Responsive to Needs
Meets business and customer requirements for market relevance



Safe and Secure
Provides confidentiality, integrity, & availability + regulatory compliance

Quality and Performance
meets the quality, speed, scalability, reliability, and other expectations

What is Secure DevOps?



First Production Release

Minimum viable product (MVP) for:

- ***Dev** - Business Requirements*
- ***Sec** - Compliance / Security / Safety*
- ***Ops** - Quality / Performance / Support*

Idea Incubation
New Product or Service

Architecture & Governance
Security, Compliance, Identity, & Other Standards

DevOperations
Ongoing Agile Innovation

Continuous Improvement of Lifecycle

1. **Refine MVP definitions –** how you define and meet MVP for Dev, Sec, and Ops (Business agility, operations stability, security assurances, identity standards, etc.)
2. **Continuously improve** process and program to improve developer productivity, operational efficiency, security posture, identity standardization, etc.)

What is Secure DevOps?

Practice that assures security is a crucial part of software delivery lifecycle

Secure DevOps have 3 main principles

- Security culture
- Secure software delivery
- Secure infrastructure

Secure DevOps Principles

Secure DevOps Principles

Security culture

Secure software delivery

Secure infrastructure

Security Culture

“Security culture refers to the set of values, shared by everyone in an organization, that determine how people are expected to think about and approach security.”

CPNI UK (Centre for the Protection of National Infrastructure)

How to promote security culture?

Promote awareness and training

Build your principles

Put principles in practice

Secure DevOps Principles

Security culture

Secure software delivery

Secure infrastructure

Secure Software Delivery

Makes mandatory that everyone learn about basic security practices

3 main goals

- Reduce the number of vulnerabilities
- Reduce the severity of vulnerabilities you miss
- Prepare for potential attacks

SDL practices are implementing through 3 types of learning

- Learning by doing
- Learning by simulating
- Learning by playing

SDL Practices

Provide security training for all your organization (developers, operations, …)

Define security requirements with metrics & compliance reporting

Perform threat modeling (OWASP at least)

Define and use cryptography standards

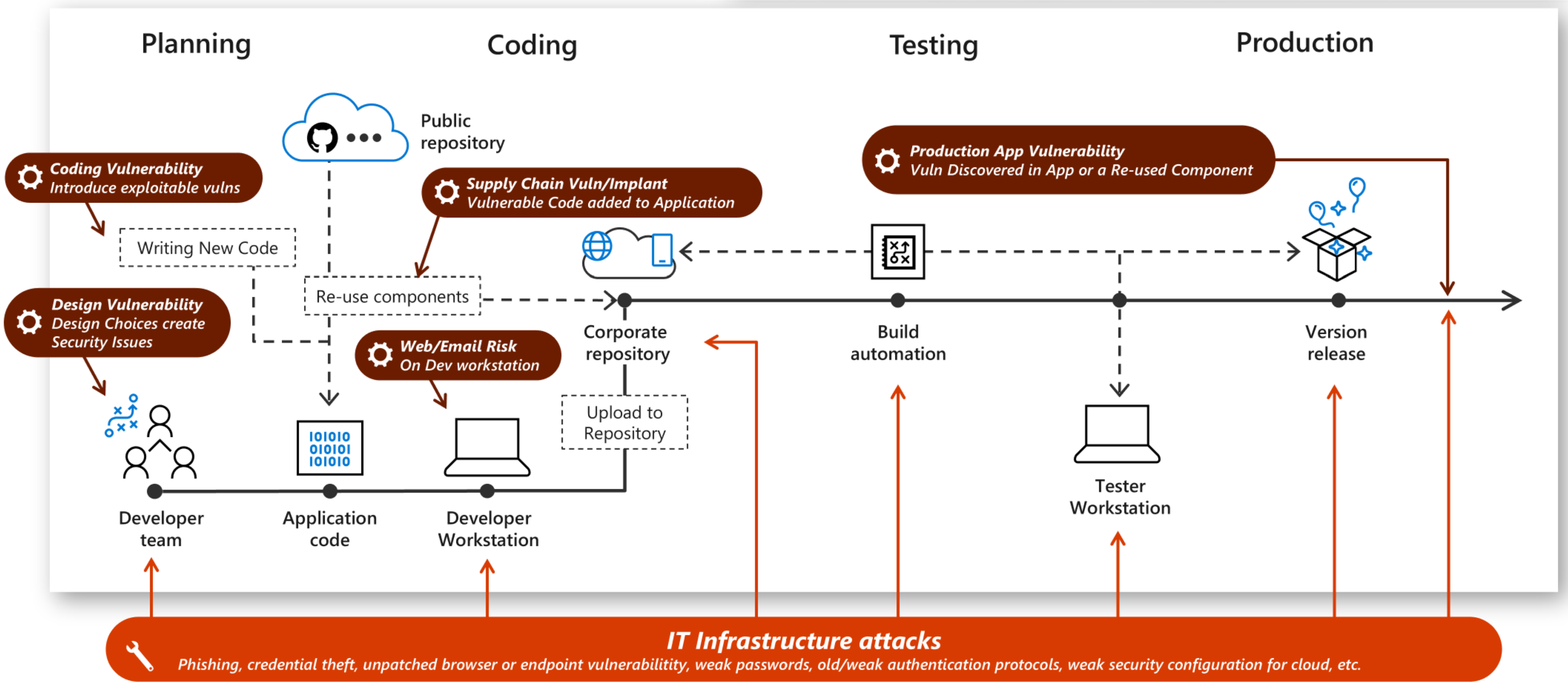
Manage security risk of using 3rd party components

Perform static analysis security testing (development time), dynamic analysis security testing (testing time) and penetration testing (production time)

Establish standard incident response process

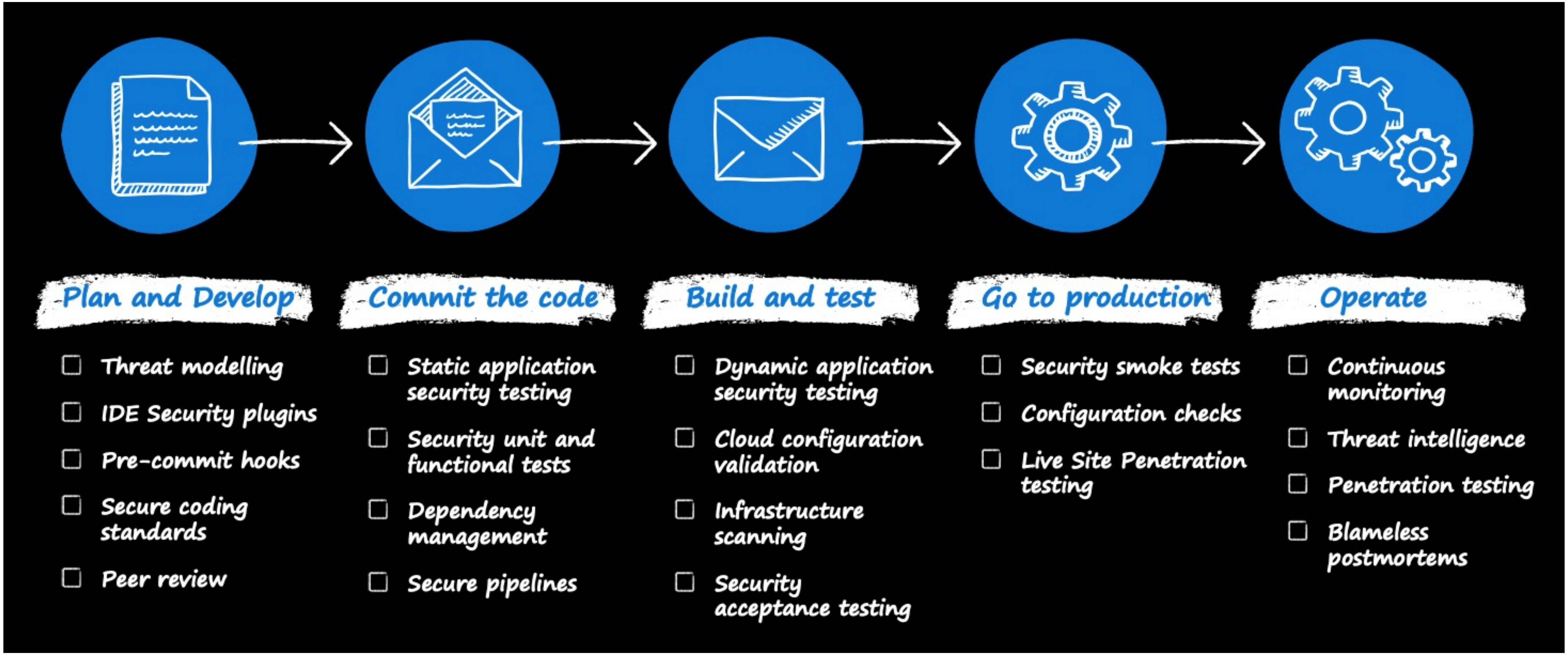
SDL: Attacker Opportunities & Lateral Moves

Attacker thinks in graph, defender think in lists



Planning
Coding
Testing
Production
Public repository
Coding Vulnerability
Introduce exploitable vulns
Supply Chain Vuln/Implant
Vulnerable Code added to Application
Production App Vulnerability
Vuln Discovered in App or a Re-used Component
Writing New Code
Re-use components
Design Vulnerability
Design Choices create Security Issues
Web/Email Risk
On Dev workstation
Corporate repository
Build automation
Version release
Upload to Repository
Tester Workstation
Developer team
Application code
Developer Workstation
IT Infrastructure attacks
Phishing, credential theft, unpatched browser or endpoint vulnerabilitity, weak passwords, old/weak authentication protocols, weak security configuration for cloud, etc.

SDL Practices



Plan and Develop

- □ Threat modelling
- □ IDE Security plugins
- □ Pre-commit hooks
- □ Secure coding standards
- □ Peer review

Commit the code

- □ Static application security testing
- □ Security unit and functional tests
- □ Dependency management
- □ Secure pipelines

Build and test

- □ Dynamic application security testing
- □ Cloud configuration validation
- □ Infrastructure scanning
- □ Security acceptance testing

Go to production

- □ Security smoke tests
- □ Configuration checks
- □ Live Site Penetration testing

Operate

- □ Continuous monitoring
- □ Threat intelligence
- □ Penetration testing
- □ Blameless postmortems

SDL Practices: Infra as Code

Plays a very important role on recoverability

If your infra suffers an attack and you need to rebuild (or build on another place) IaC allow to be faster and secure

On organization using cloud, Infra as Code is the way to implement Disaster Recovery

SDL Practices: Static Application Security Testing (SAST)

Improve code security and quality on an easy and cost-effective way

Makes an analysis on your source code and return insights about security, performance, maintainability

Fully automated and can be shift-left for developers IDE

Runs to answer by with the question “Is the code secure?”

- Is it vulnerable to injections (like SQL)?
- Does it use any weak encryption algorithms?
- Are cookies used with the right flags?

SDL Practices: Dynamic Application Security Testing (DAST)

Type of testing that looks for security vulnerabilities by safely exploiting a running application from the outside

This type of testing is not dependent on the framework or programming language used

Differs from SAST because runs on top of a running application

Must of the times, a specific environment is created to run DAST tests (and then destroyed ☺)

Uses fuzzy mechanisms to test your application, like send request with body, corrupted headers, etc.

At least should run tests on your application to check OWASP Top 10 Attacks

SDL Practices: Dynamic Application Security Testing (DAST)

Ideally, and using shift-right approach, you may run it in production expecting issues to arise

Works well together with canary/ring deployment strategies where you may test and affect only a subset of your users

Other techniques for production testing like penetration testing and chaos engineering (monkey testing)

SDL Practices: DAST Tooling

netsparker®
web application security scanner

acunetix

detectify

GitLab
Ultimate

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

SDL Practices: Software Composition Analysis (SCA)

So, Open Source is a bad and dangerous thing? Of course not!

But you need to use it careful and mostly you need to clearly know what are you using!

Constantly run a scan on your dependencies is crucial to understand known vulnerabilities on your supply chain

Knowing the vulnerabilities and their severity you may define you plan to fix them

Know what you're using means knowing your dependency graph! Your direct dependency have its own dependencies. That dependencies have their own dependencies and so on…

Secure DevOps Principles

Security culture

Secure software delivery

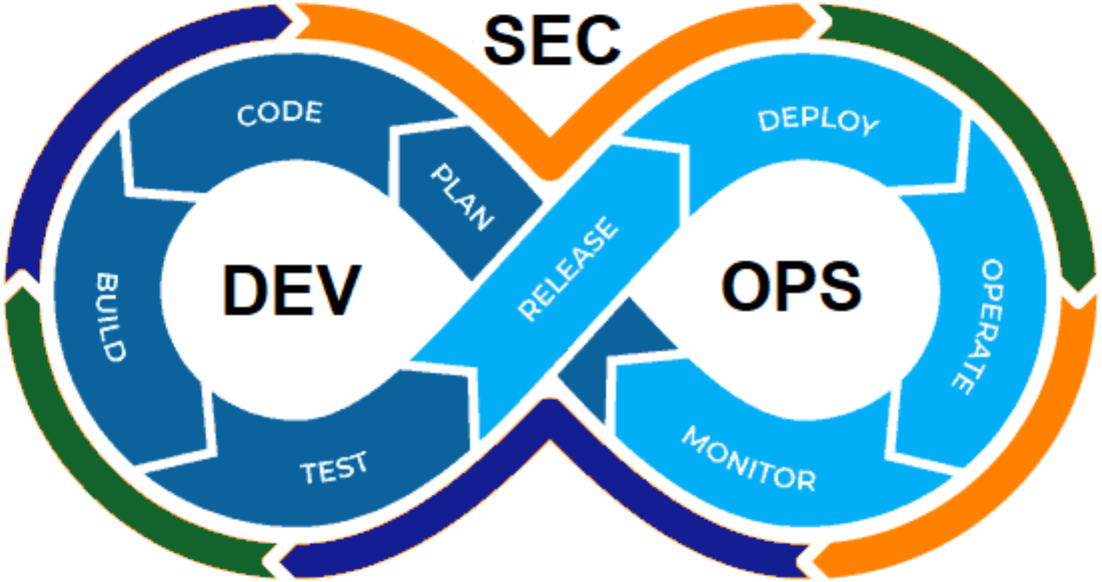
Secure infrastructure

Secure Infrastructure

Secure DevOps proposal is to add security layer on top of any phase of DevOps infinite loop

Any phase uses infrastructure to execute expected outcomes

What means infrastructure in DevOps?



What means infrastructure in DevOps?

For plan, your PO, designer or architect workstations are DevOps Infra

On build phase, developers and any want producing code workstation is DevOps Infra

If you use any internal repository, is DevOps Infra

During CI/CD, your runners are DevOps Infra, even more if you don't control them directly and you're doing deploys on your infra

On testing phases, testers and even customers workstations are DevOps Infra

During operation, all your operations and infra team workstations are DevOps Infra

Oh! And your production (all) environments are DevOps Infra too! ☺

Secure Infra in DevOps

Vulnerable workstations open doors for lateral moves

Constantly update your machines

Zero trust principles, grant access to everything is needed but nothing more

Repository access sharing credentials and adding to the repos

Reuse of credentials without rotation

Isolate your environments to make harder to do lateral moves

Upskill your collaborators and make surprise tests for common tasks, like email phishing

Secure DevOps Practices

Secure DevOps Practices

Secure DevOps Practices acts as the enablers of principles

Making these practices better allow you to implement better processes for your principles

Makes security into your daily workflow

Main practices

- Shifting Left
- Continuous improvement
- Automation

Secure DevOps Practices

Shifting Left

Continuous improvement

Automation

Secure DevOps: Shifting Left

Introducing security controls since the beginning

Security team must be involved since day 1

Initially, can be to make solution compliant with well defined security controls

Since security is an everyone's responsibility, security teams can be focused on upskilling and being always updated

Secure DevOps Practices

Shifting Left

Continuous improvement

Automation

Secure DevOps: Continuous Improvement

Is a basic practice for DevOps, you must be always looking to your processes and try to make them better, faster, more secure

Security topics are evolving every day, with attackers always one step forward than defenders

You need to be informed by security team and your security controls, to identify possible vulnerabilities

Your implementation processes needs to be reviewed constantly to face new possible vulnerabilities

Secure DevOps Practices

Shifting Left

Continuous improvement

Automation

Secure DevOps: Automation

Again, crucial practice to have a proper outcome

Automate allow you to be consistent on analysis, faster on implementation and make it easier to evolve

SCA and SAST processes without automation are not viable

Can be used on every step and allow you to create security guards that only allow you to proceed when meet security principles

moOngy.
Minds on the move

Rua Sousa Martins, nº 10
1050-218 Lisboa | Portugal