Electronics and
Computer Science

University of Southampton

Software Security

COMP6226: Software Modelling Tools and Techniques for Critical Systems

Dr A. Rezazadeh (Reza)
Email: ra3@ecs.soton.ac.uk or ar4k06@soton.ac.uk

November 24

University of Southampton

Overview

- Security Engineering
- Security Related Concepts
- Security Dimensions – The CIA Triad
- What is Secure Software?
- Secure Software Development
- Identifying Underlying Issues – Threat Modelling
- Analysis & Design
- Secure Systems Design
- Secure Systems Implementation

University of Southampton

Objectives

- To understand different Aspects of Security
- To make sense of different Security Concepts
- List main Security Dimensions
- Compare Secure Software Development with classical software development
- Understand Security Requirement Engineering and Threat Modelling approaches
- Make sense of Security Design and Implementation Principals

University of Southampton

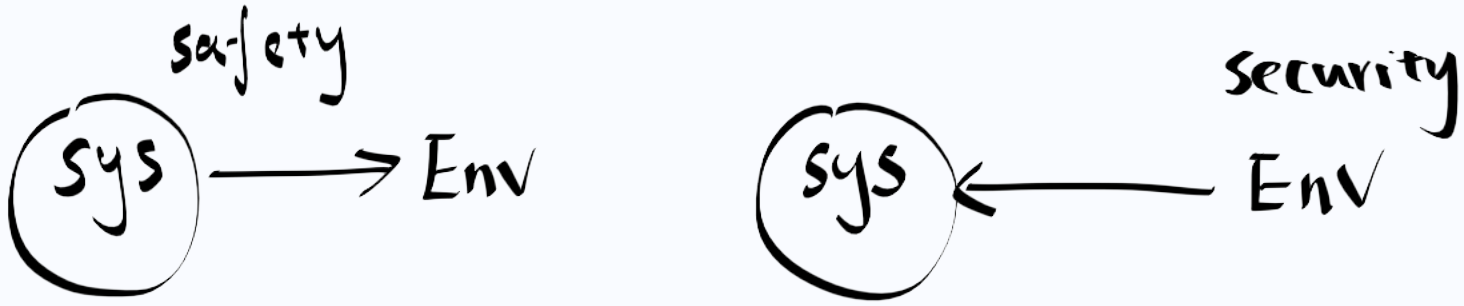
Security Engineering

- The goal of software security is to support the development of systems capable of withstanding malicious attacks that aim to harm a computer-based system or its data.
- A sub-field of the broader field of computer security.
  - Data Security
  - Component Security
  - System Security
  - Organizational Security
  - Software Security
  - Connection Security
  - Human Security
  - Societal Security
- According to "*The Cybersecurity Body of Knowledge*"

University of Southampton

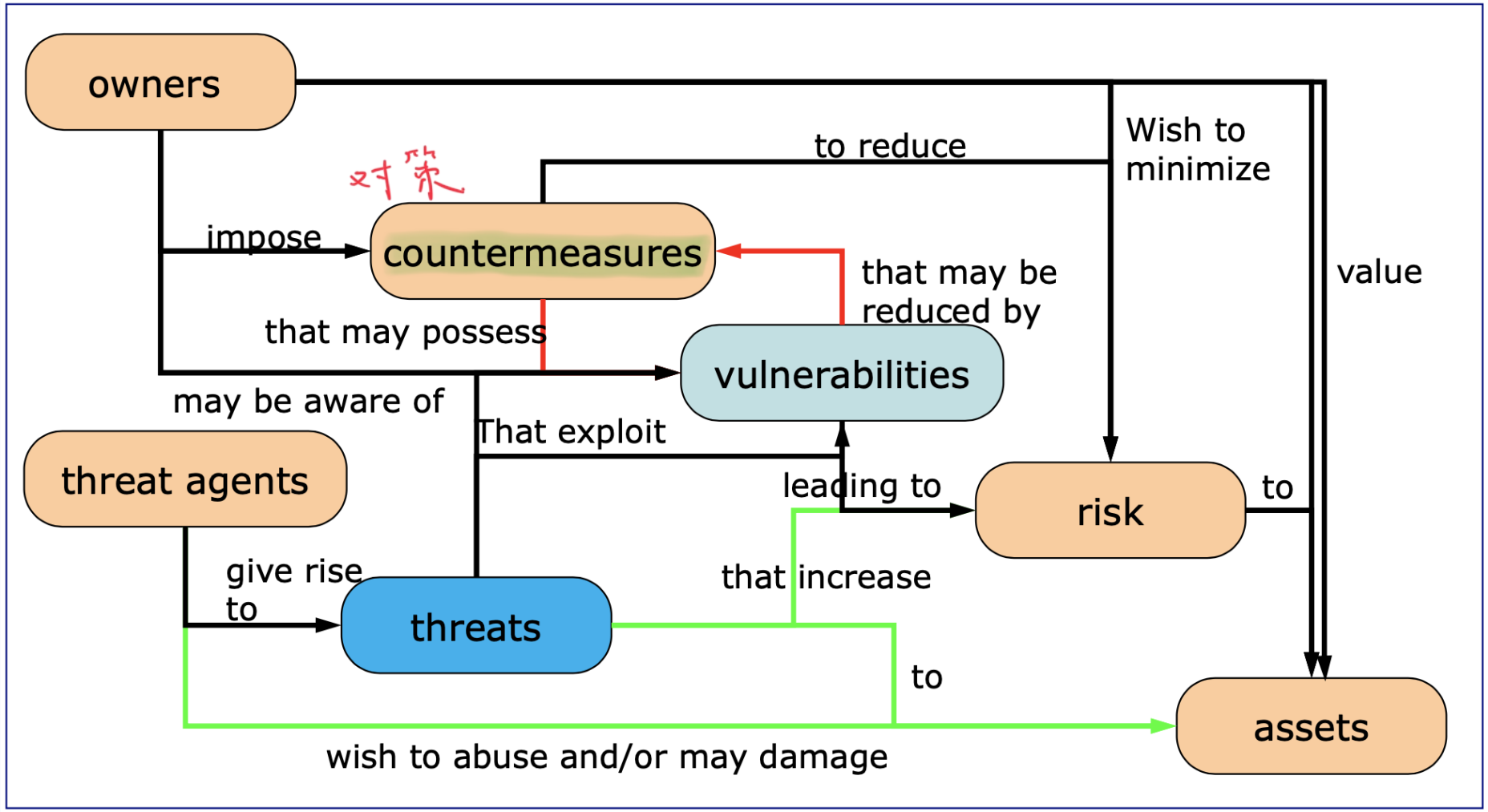
Security Related Concepts

- Security is about protecting assets from malicious intended behaviors
- Asset: A system resource (data and software) of value for the stakeholders
- Policy: Requirements for what is allowed and what is not allowed
- Threat: An event with the potential to violate the policy and inflict damage on a system
- Vulnerability: A weakness that makes a threat possible
- Attack: The action of exercising a threat by exploiting related vulnerabilities



security → safety

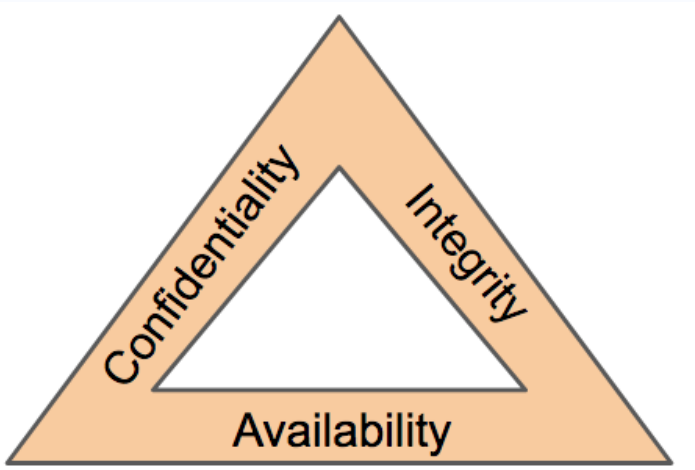
Relation Between Security Concepts



Security Dimensions – The CIA Triad

三个一组

- The three letters in "CIA triad" stand for Confidentiality, Integrity, and Availability.
  - The CIA triad is a common model that forms the basis for the development of security systems.
- *Confidentiality:* Information in a system may be disclosed or made accessible to people or programs that are not authorized to have access to that information.
- *Integrity:* Information in a system may be damaged or corrupted making it unusual or unreliable.
- *Availability:* Access to a system or its data that is normally available may not be possible.



Threats to Confidentiality – Some Examples

- The following are some challenges faced while maintaining confidentiality:
  - Direct attacks involve gaining unauthorized access to systems to view, modify, or alter data.
    - An example is the man-in-the-middle (MITM) attack that aims to intercept the data by invading the information stream. They steal or modify data and use it for malicious purposes.
  - Some other attacks involve network spying to get credentials for system authorisation or clearance of the next level by accessing system privileges.
  - Human error can also pose a threat to confidentiality.
    - Not protecting passwords, sharing credentials with others, hardware threats, and communication channels not encrypted correctly can be harmful.

Threats to Integrity – Some Examples

- The following are the challenges faced while maintaining Integrity:
  - It involves intentional threats such as an attacker altering logs to hide attacks and changing file configurations to allow unauthorised access.
    - They may also bypass an intrusion detection system.
  - Unintentional mistakes such as entering a wrong code can also tamper with the integrity of the data.
  - Inadequate security policies or procedures lead to violation of data integrity without accountability.

University of Southampton

Threats to Availability – Some Examples

- **DDoS attacks** – These are attacks that flood a server or network with traffic, making it difficult or impossible for legitimate users to access the system.
- **Malware** – Malware, or malicious software, can infect systems and disrupt their availability. For example, a ransomware attack could encrypt data on a system and make it unavailable until a ransom is paid.
- **Hardware failures** – Hardware components can fail, leading to system outages.
- **Accidental deletion or modification of data** – Users may accidentally delete or modify data, making it unavailable or unusable.
- **Network outages** – Network outages can occur for various reasons, such as equipment failures or cut cables, and can prevent users from accessing systems and data.

University of Southampton

What is Software Security?

- Software security is the idea of engineering software so that it continues to function correctly under malicious attack.
- The idea behind software security is building software that is secure from the get-go without having to add additional security elements to add additional layers of security (although in many cases this still happens).
  - You should design security into software. add security in early stage
  - It is much more effective to design security into software from the beginning than to try adding it later.
  - When security is part of the initial design, it can be baked into the system architecture and implementation in a more integrated, cohesive way.
  - Retrofitting security often results in a patchwork of disjointed and ineffective controls.

Causes of Software security Issues

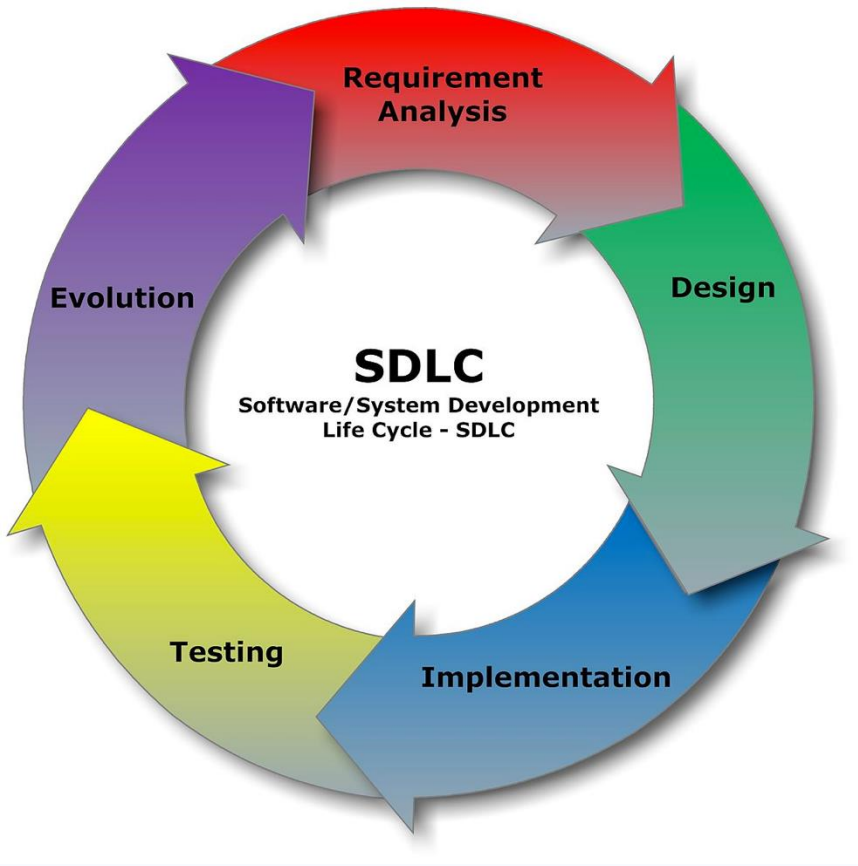
- The application development process in its essence fails to address security issues
- Consequently, security flaws are identified only at the later stages of the application lifecycle. And thus:
  - Much greater cost to fix
  - High maintenance cost
  - Legal liability
- Nearly every company/organisation utilises network security infrastructure (e.g., Firewalls, Intrusion Detection System, etc)
- But very small number of them invest in application security strategy, design, and code review services

Secure Software Development

- For the software industry, the key to meeting demand for improved security, is to implement repeatable processes that reliably deliver measurably improved security.
- Thus, there must be a transition to a more stringent software development process that greatly focuses on security.
  - The secure development life cycle (SDLC) incorporates security practices into every phase of the software development life cycle from inception through deployment and maintenance.
  - Goal: minimize the number of security vulnerabilities in design, implementation, and documentation.
  - Identify and remove vulnerabilities in the development lifecycle ***as early as possible!!!***

Secure Software Development

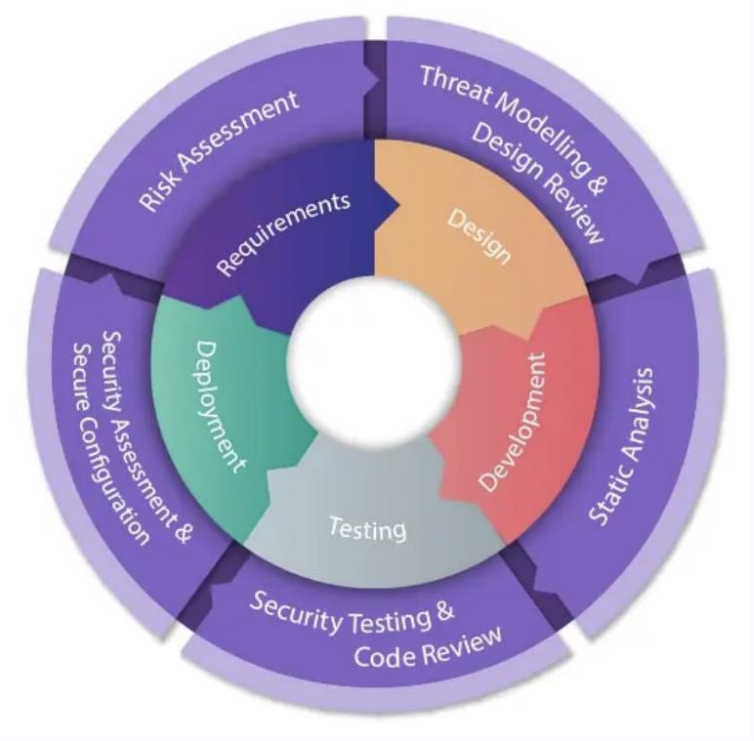
- Consider security throughout the software development lifecycle
  - Requirements
  - Design
  - Implementation
  - Testing
  - Deployment 部署



University of Southampton

Requirements

- Identify sensitive data and resources
- Define security requirements for them
  - Confidentiality
  - Integrity
  - Availability

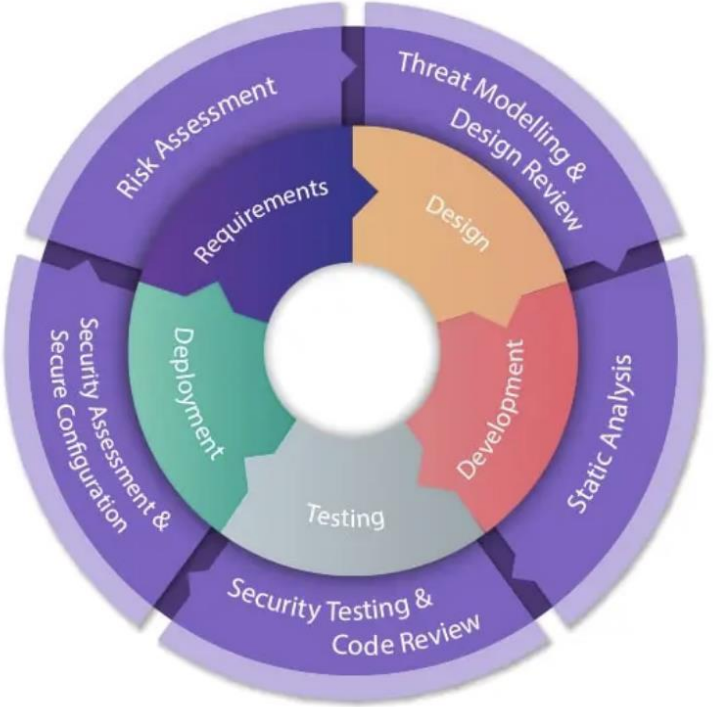


- Consider threats and abuse cases that violate these requirements
- Security Requirements – are constraints on the system functions

University of Southampton

Identifying Underlying Issues – Threat Modelling

- Threat modelling is about identifying potential threats to a given system.
- There are three approaches to thread Modelling:
  1. Attacker-based approach
     - Who are the opponents
     - What are their goals
  2. Asset-based approach
     - What value does the asset have
     - How can the attacker reach the asset
  3. Software/System-based approach
     - What vulnerabilities can the attacker exploit

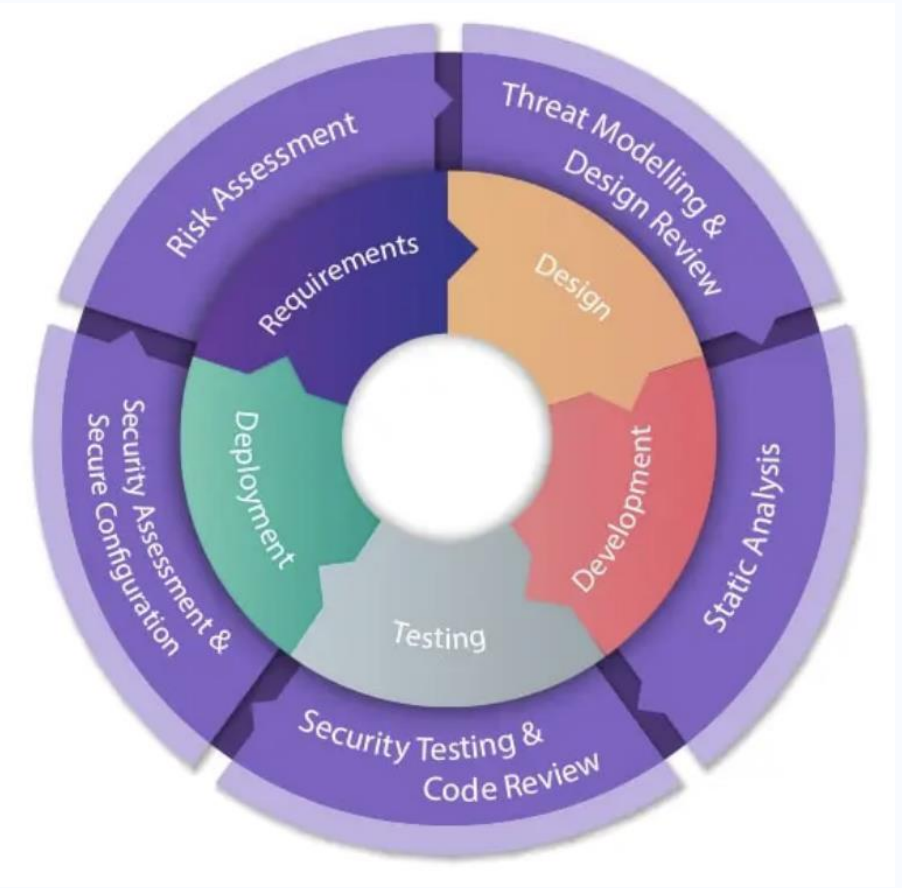


University of Southampton

Analysis & Design

Secure Systems Design

- Apply principles for secure software design
  - Prevent, mitigate and detect possible attacks
- Security principles apply this into design
  - Favor Simplicity
  - Trust with Reluctance (Principle of least privilege)
  - Defend in Depth (Create Multiple Security Layers)
  - Deny Access by Default



Security Architecture

- Security architecture: a software design that enforces the security policy of and mitigates the threats to the software
- Security perspective: view of the software architecture that considers security requirements
- Security design principles: a set of principles for designing secure software, e.g., *least privilege*
- Security patterns: solutions to recurring *security problems*

Secure Systems Implementation and Testing

- After the project design stage is completed, the actual development of the software can begin.
  - In this context, development refers to the actual coding and programming of the application.
- Development must take place using secure coding standards.
  - Programmers should have up-to-date knowledge of the relevant security standards and how they apply to the current project.
- Development must appropriately implement secure design patterns and frameworks.
- Development must take advantage of the latest secure coding practices.
  - This typically means using updated versions of programming languages that best address current security standards.

Secure Systems Testing

- Use automated code review techniques to find potential vulnerabilities components
  - Static Analysis
  - Symbolic execution
  - Penetration Testing to find potential flaws in the real system. In penetration testing, a security professional will attempt to hack into your system as an outsider would using any number of commonly utilized methods.
  - Fuzz testing or fuzzing is an automated software testing method that injects invalid, malformed, or unexpected inputs into a system to reveal software defects and vulnerabilities.

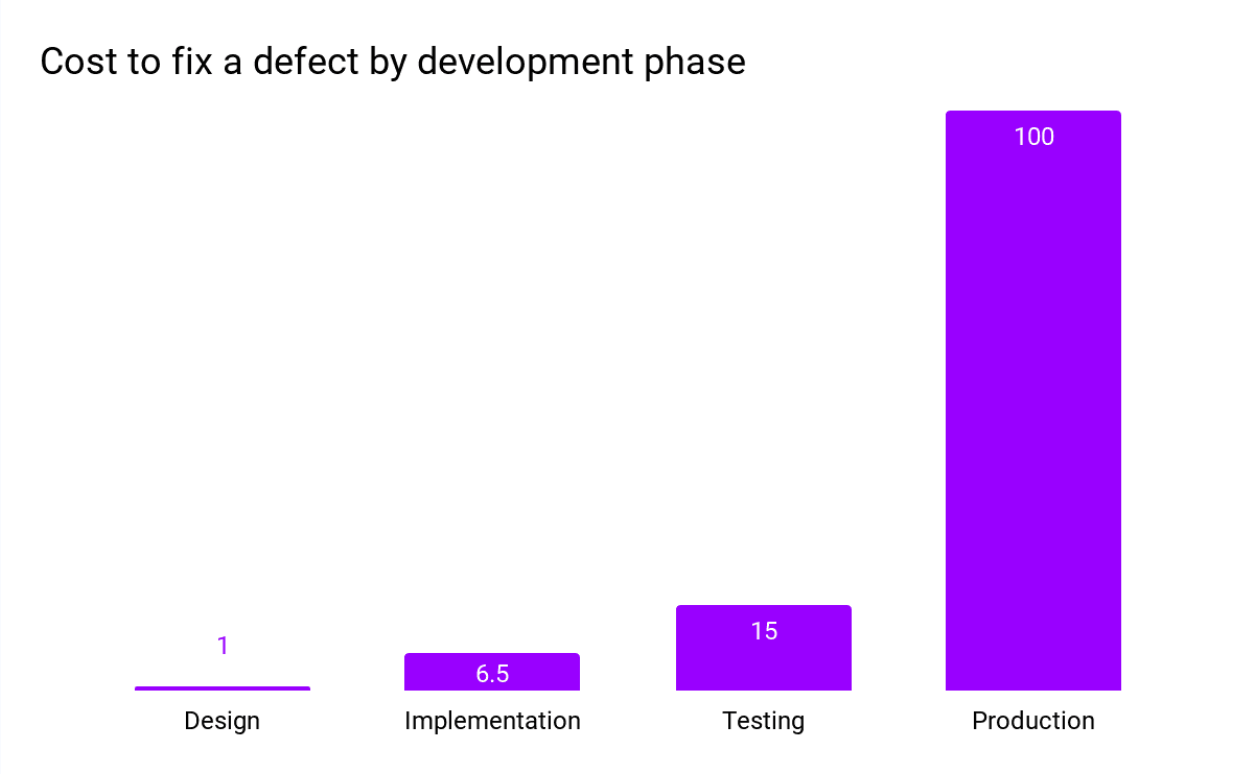
Deployment and Maintenance

- The story doesn't end once the application is released. During the deployment phase, it is important to follow secure deployment practices to ensure that the software is deployed securely.
  - This may include conducting a security assessment of the deployment environment, implementing appropriate controls, and following best practices for securing the software in production.
- After deployment, vulnerabilities that slipped through the cracks may be found in the application long after it's been deployed.
  - These vulnerabilities may be in the code developers wrote but are increasingly found in the underlying components that comprise an application discovered in production by the application's maintainers.
- These vulnerabilities then need to be patched by the development team, a process that may in some cases require significant rewrites of application functionality.

University of Southampton

The Benefits of Secure SDLC

- Reduced Costs: Thanks to early identification of security concerns allowing the embedding of controls in parallel. No more patching post-deployment.



University of Southampton

The Benefits of Secure SDLC – Cont.

- **Security-First:** Secure SDLC builds security-focussed cultures, creating a working environment where security comes first, and everyone's eyes are on it. Improvements happen across the organisation.
- **Development Strategy:** Defining security criteria from the outset improves technology strategy, making all team members aware of the security criteria of the product, and ensuring developer security throughout the lifecycle.
- **Better Security**: Once Secure SDLC processes are embedded, security posture improves across the whole organisation. Organisations that are security aware reduce their risk of cyberattack significantly.

University of Southampton

Different Methodologies

- Microsoft Security Development Lifecycle
  - https://www.microsoft.com/en-us/securityengineering/sdl/
- BSIMM (Building Security In – Maturity Model)
  - http://bsimm.com
- OpenSAMM Software Assurance Maturity Model
  - http://opensamm.org

University of Southampton

Reference

- Chapter 13 of Software Engineering by Ian Sommerville (10ed)

University of Southampton

YOUR QUESTIONS