Product Security for Bosch (IoT) Products

Stephan van Tienen, BT-CO/ETP1, 12th June 2023

BOSCH

01

Introduction

BOSCH

Introduction
Stephan van Tienen

- Business Sector **Energy and Building Technology**
- Company **Bosch Building Technologies**
- Product Business **Security and Safety Systems**
- Business Unit **Communication Systems**
- System Architect of **Platform Engineering Team**

- Product Security Partner for the Business Unit
- MSc Electrical Engineering (Delft University of Technology)

Introduction
Bosch business sectors

Mobility Solutions

Industrial Technology

Energy and Building Technology

Consumer Goods

Introduction
Mobility Solutions

Link to IoT:

- Software-Defined Vehicle
- Connected mobility solutions

BOSCH

Introduction
Industrial Technology

Link to IoT:

- Connected Industry (Industry 4.0)
- Manufacturing and Logistics

BOSCH

Introduction
Energy and Building Technology

Link to IoT:

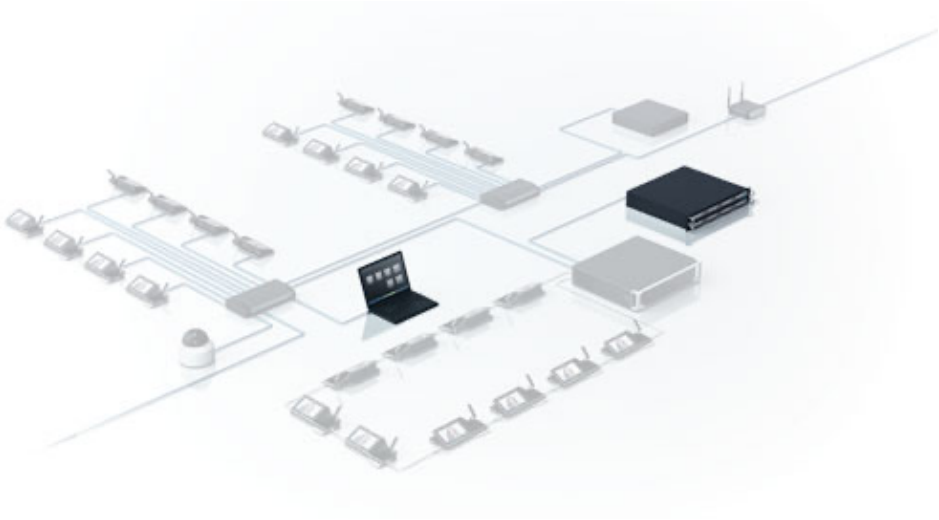
- Connected residential and commercial products and services
- Connected business management products and services

Introduction
Consumer Goods

Link to IoT:

- Connected power tools (home/professional)
- Connected home appliances

Introduction
Communication Systems

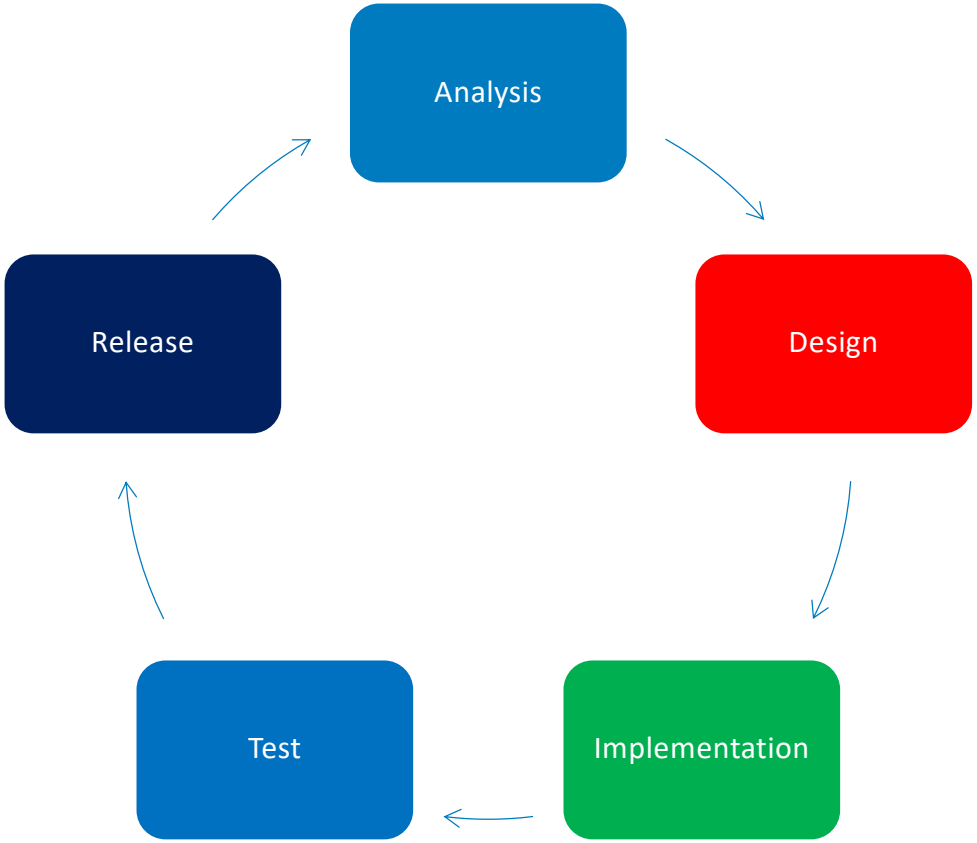


BOSCH

02

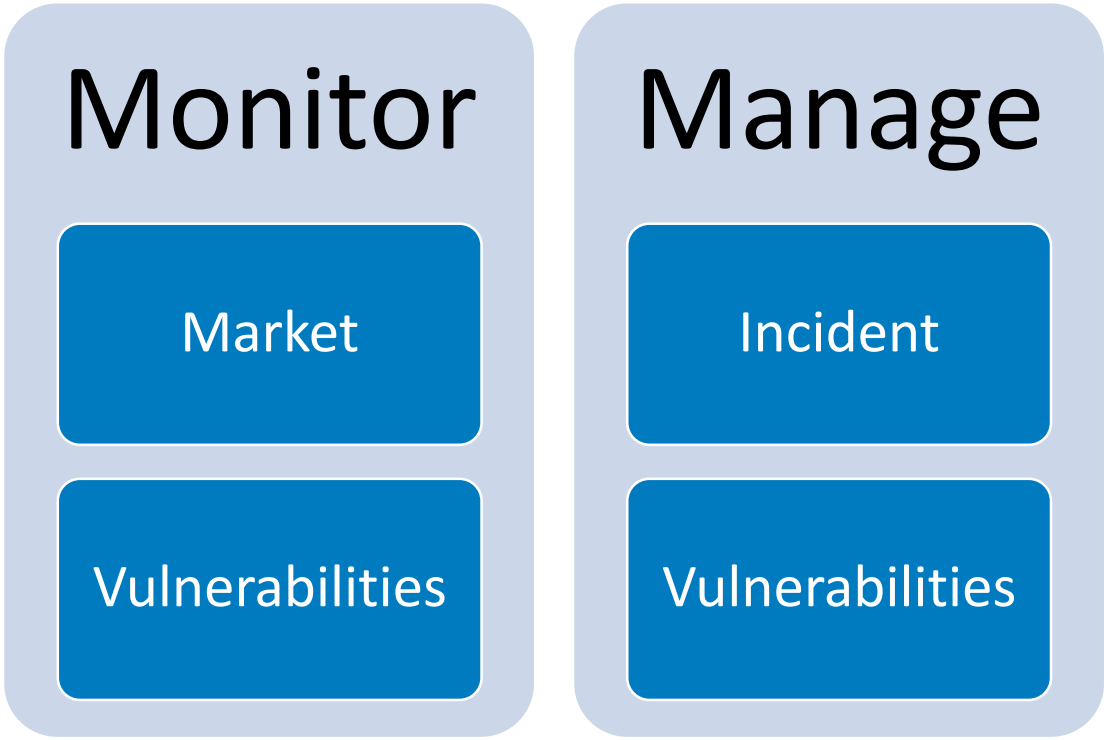
Security Engineering Process

Security Engineering Process
During Development Project



BOSCH

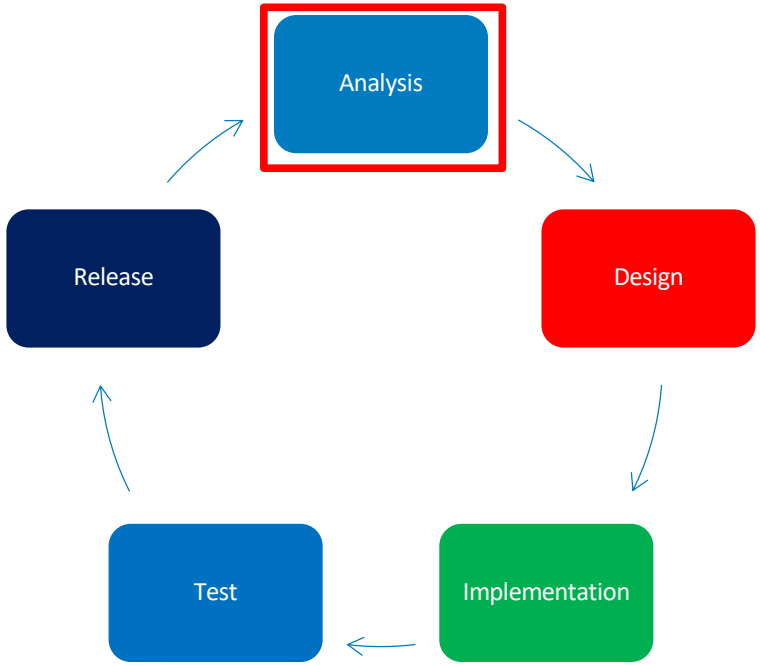
Security Engineering Process
After Market Introduction



- Bosch Product Security Incident Response Team (PSIRT)

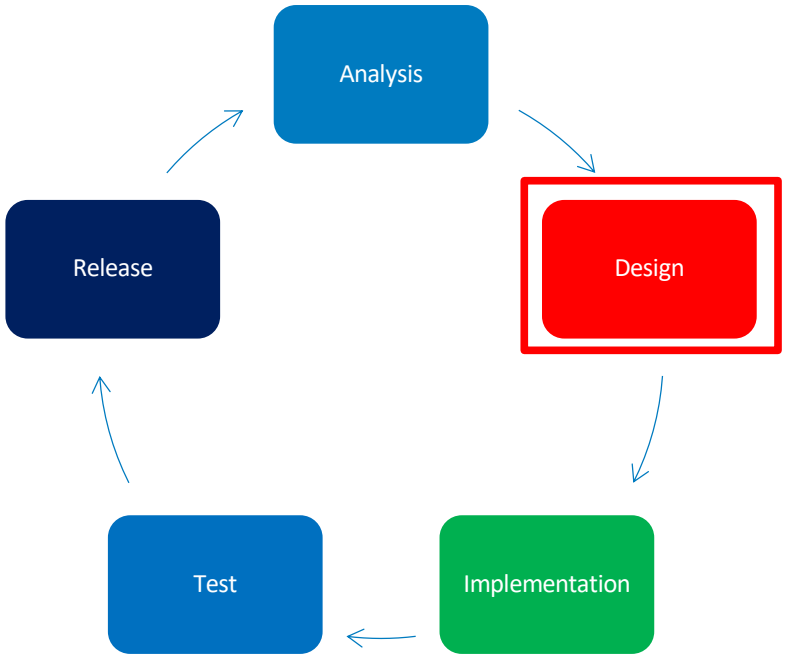
Security Engineering Process
Analysis Phase

- Execute Threat and Risk Analysis
- Business owner and Security expert
- Bosch specific tool Armadillo



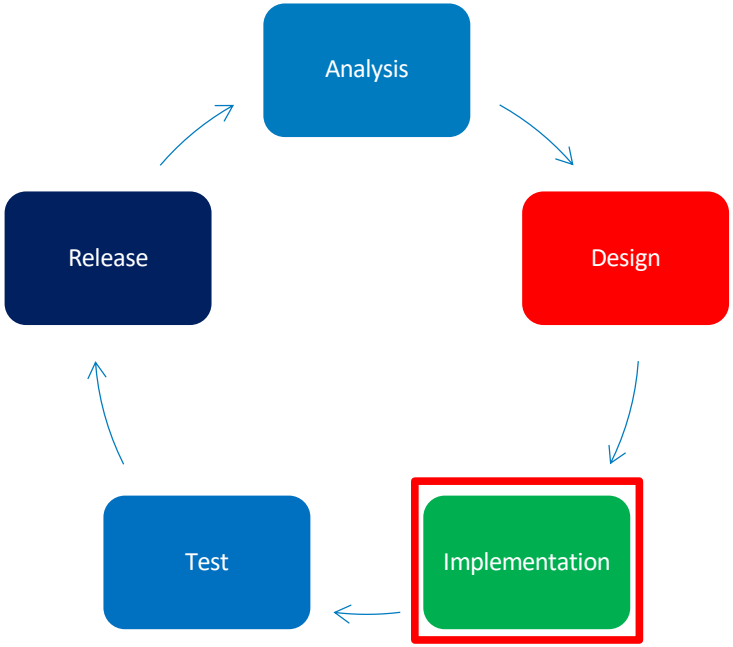
Security Engineering Process
Design Phase

- No Security by Obscurity
- Always use industry standards
- Detailed example later on



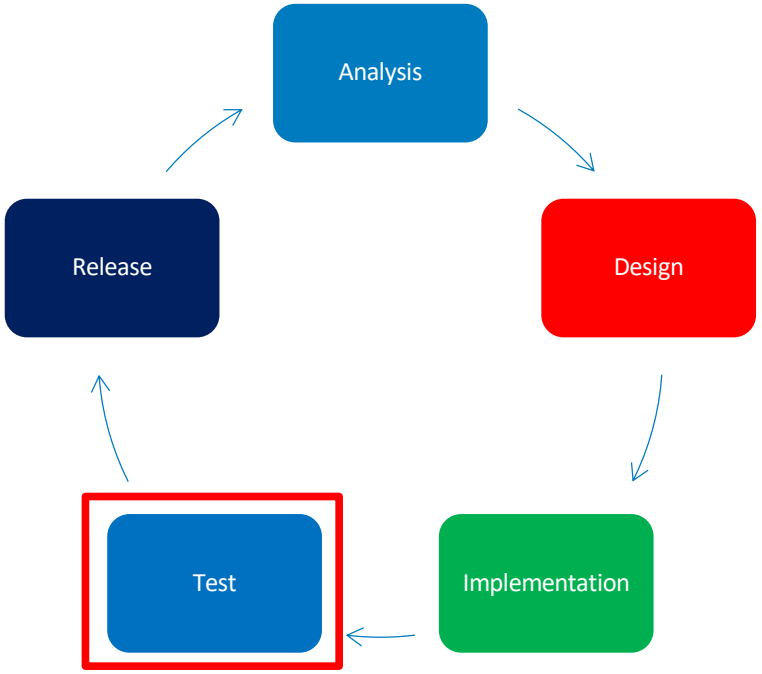
Security Engineering Process
Implementation Phase

- Coding Guidelines
- Code reviews
- Tooling



Security Engineering Process
Test Phase

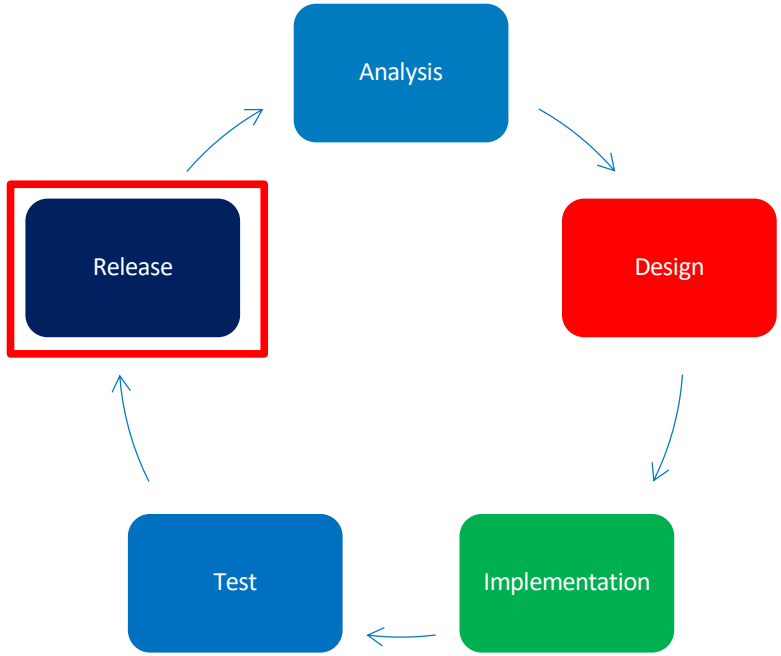
- Create Test Plan
  - Corporate Guidance on type of security test
  - External penetration test mandatory in many cases
- Execute Test Plan
- Some types are challenging
  - Lacking expertise



Security Engineering Process
Release Phase

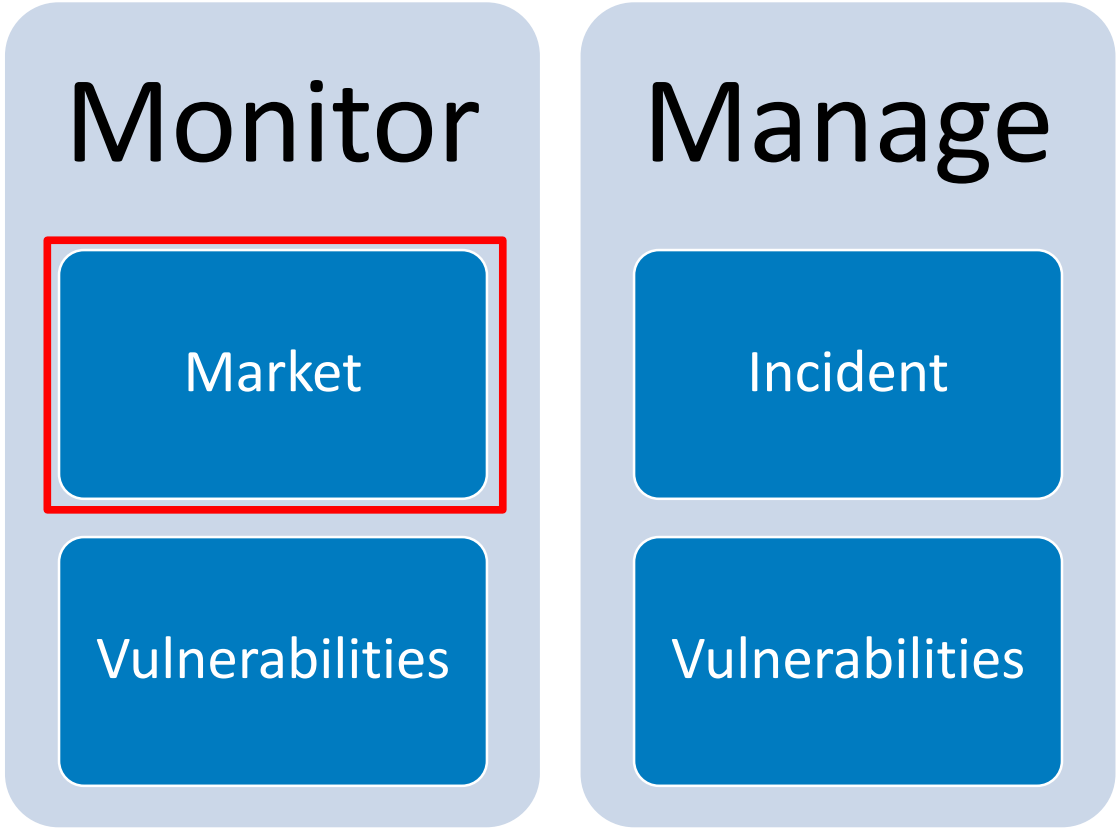
- Secure Delivery

- Strongly depends on type of product/service



Security Engineering Process
Monitor Market

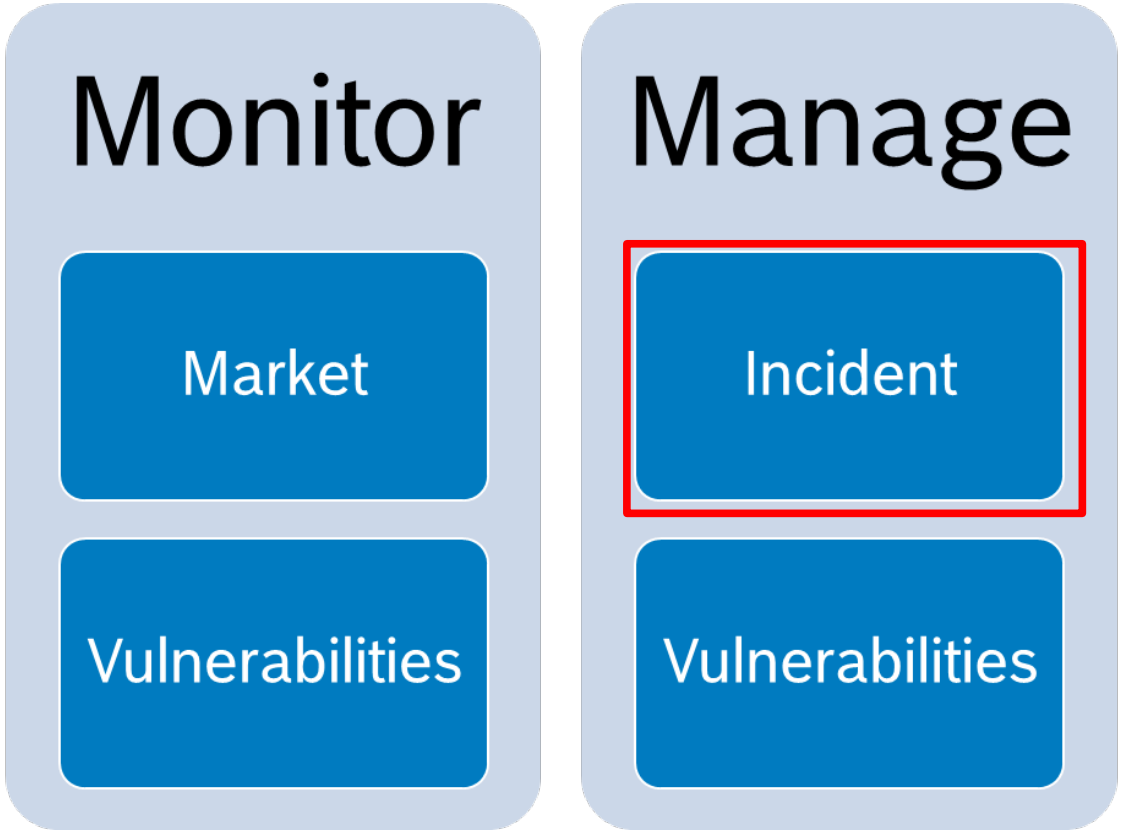
- Consider ‘state of industry’
  - helps in finding right balance
- Strongly depends on type of product/service
  - (Cloud) SIEM / EDR
  - Questionnaire
  - …



Security Engineering Process
Manage Incident

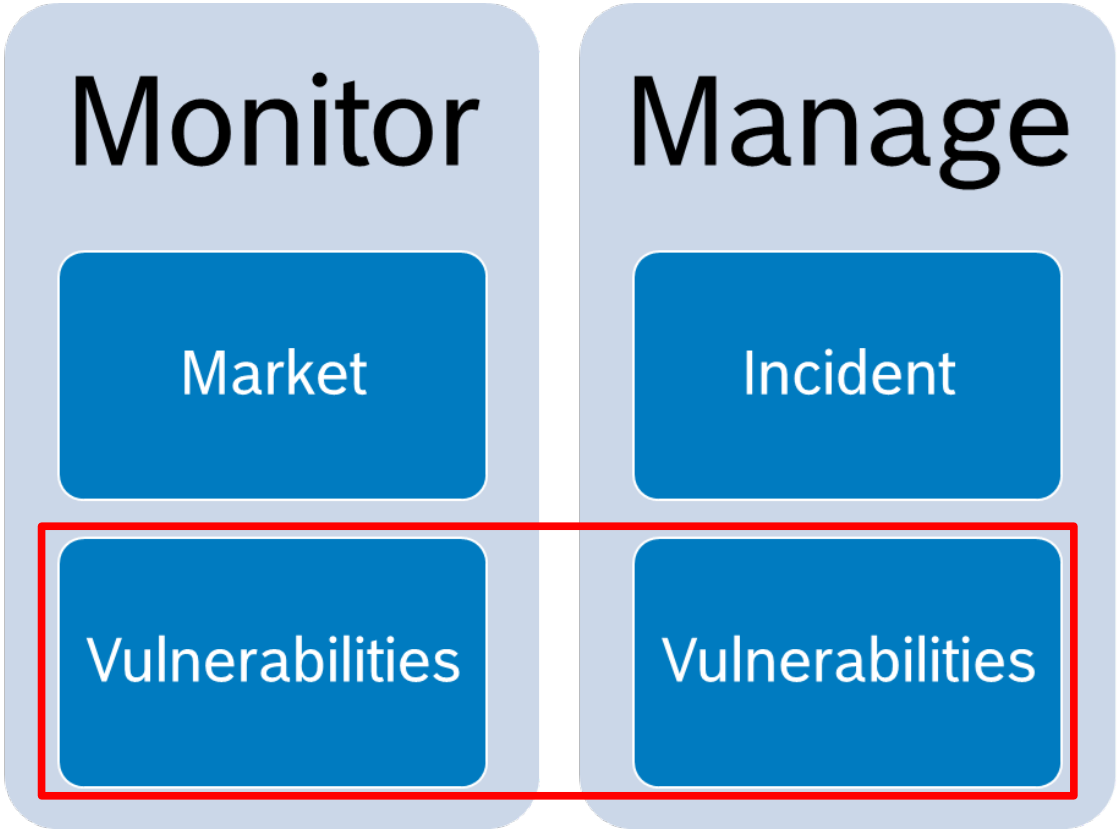
- Cross-functional crisis team is formed
  - Business owner
  - Security expert
  - Quality engineer
  - Etc

- Incident blueprint is followed
  - Corporate guideline



Security Engineering Process
Monitor and Manage Vulnerabilities

- Scanning of OSS components
  - Automated where possible
- In case of a critical issue crisis team is set up
  - Log4Shell most recent example
- Less critical issues are simply patched
  - Release schedule depends on type of product/service

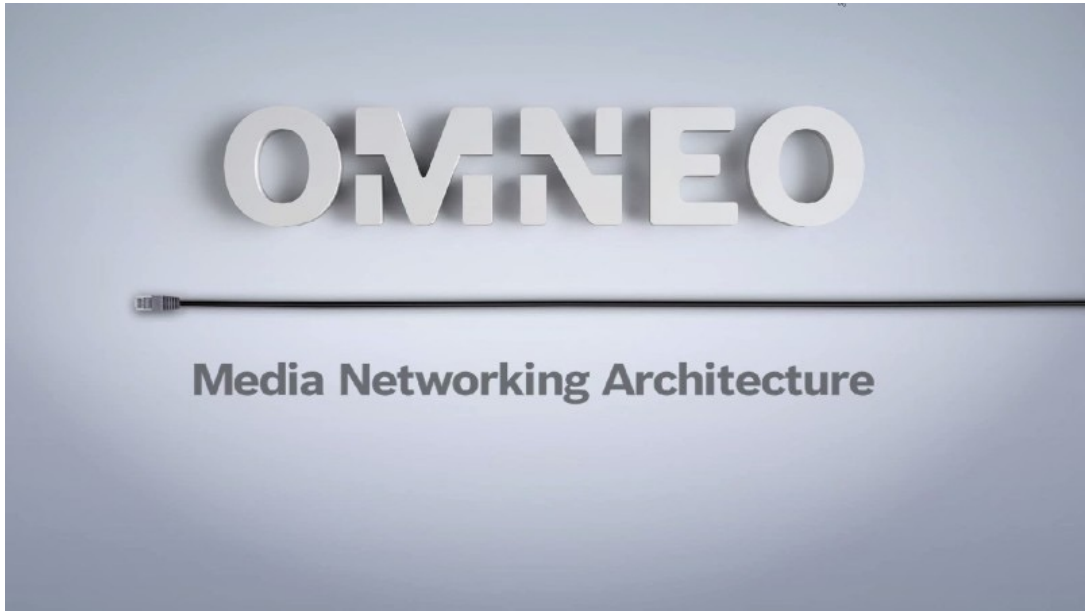


03

Design Phase – A case study

Design Phase – A case study
Introduction

- BT-CO develops OMNEO Media Networking Architecture



- Platform that offers
  - High-quality low-latency audio streaming on standard Ethernet networks
  - Standardized control protocol
  - Audio and control security
- Client/server architecture
  - Embedded controller (client) connects to embedded devices (servers)

BOSCH

Design Phase – A case study
Introduction

- Control security offered using TLS on top of TCP/IP
- TLS(1.2) ciphersuites split up into 5 elements:

- PROTOCOL_SESSKEYEX_SESSAUTH_WITH_ENCRYPTION_MESSAUTH
  - PROTOCOL: TLS or SSL
  - SESSKEYEX: how to set up session specific keys (e.g. for encryption)
  - SESSAUTH: how to authenticate the session
  - ENCRYPTION: what encryption scheme is used after session is successfully established
  - MESSAUTH: what message authentication scheme is used after session is successfully established

Design Phase – A case study
Introduction

- TRA for next generation platform came up with new Security Requirement:
  - **Use Device Certificates instead of Pre-Shared Key for session authentication**
- What then happens in Design Phase?

Design Phase – A case study
Device Certificates explained

- Certificates use **public key cryptography**

- Certificates are meant to offer **trustworthiness**

Design Phase – A case study
Device Certificates Explained

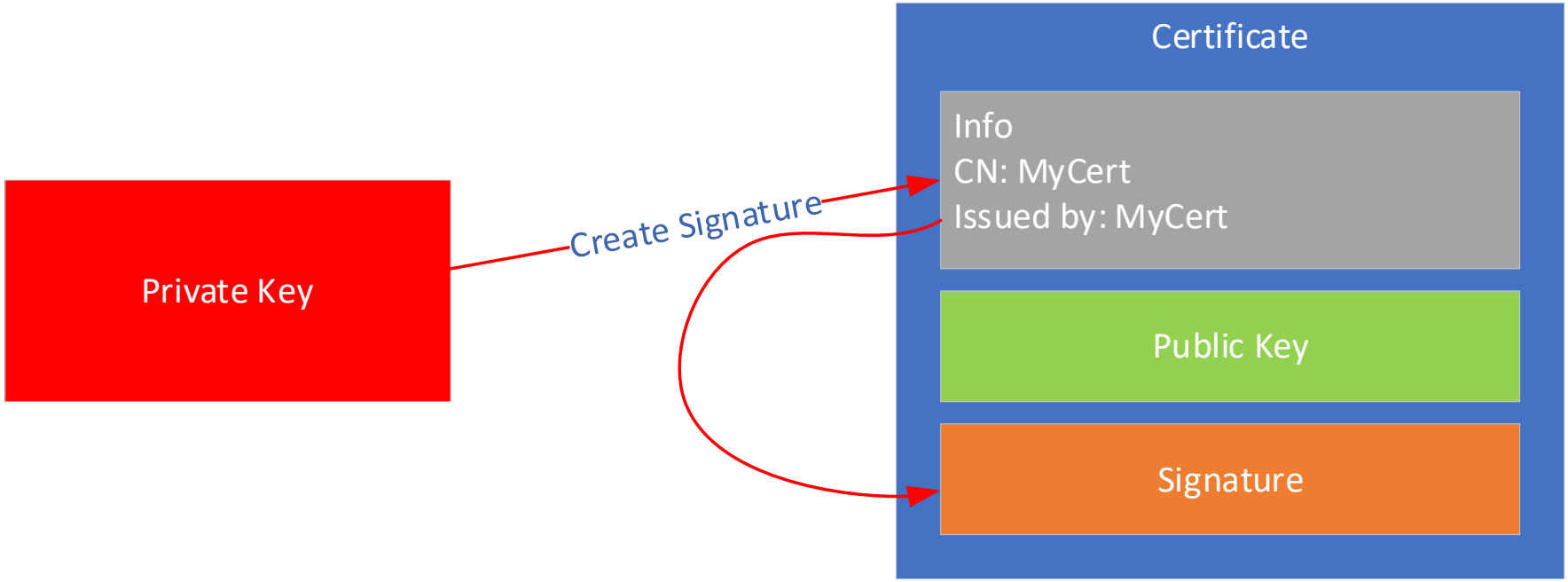
- A Device Certificate actually contains two main elements:
    1. Private key
    2. Certificate containing **information**, **public key**, **signature**

- Private key used to create a signature / decrypt a message

- Certificate:
    - **Information** about the certificate including ‘Issued by’ and ‘Common Name’
    - **Public key** used to validate a signature / encrypt a message
    - **Signature** allows validation of the **Information** of the certificate

Design Phase – A case study
Device Certificates Explained

- Self-Signed Certificate:
  - Generate public/private key pair, use private key to create signature
  - ‘Issued by’ is set to the info of the own certificate



Design Phase – A case study
Device Certificates Explained

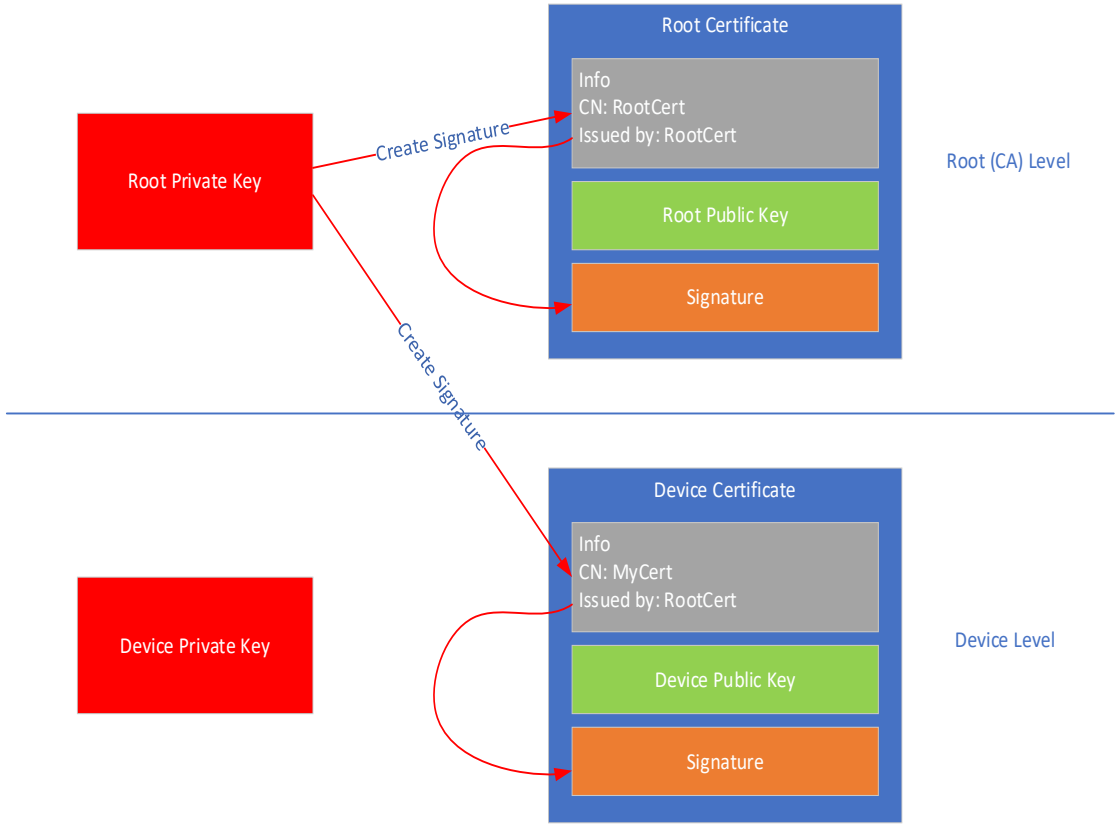
- How does server validation work in TLS?
  - Device (server) sends certificate
  - Client checks info -> date, CN, Issued by
    - With self-signed you have to **explicitly trust the info**
  - Client checks signature with public key
    - Ensures info is valid
  - Client uses public key to encrypt a message
    - Client sends encrypted message to server
  - Device decrypts message and sends back original message to client
    - Proves entity has the private key -> decryption can only be done with that key

Design Phase – A case study
Device Certificates Explained

- Example: SSH
  - When SSH client connects for the first time it asks if you want to trust the server
- Example: web server
  - Web browser will not connect to a web server that has a self-signed certificate
  - User has to knowingly continue
  - Most browsers no longer allow storing exception
- Trusting certificate is still possible for many clients
  - Grab it from device and validate its CN
  - Import the certificate in some way (add to certificate file, certificate store, etc)
  - **This does not scale well**

Design Phase – A case study
Device Certificates Explained

- Signed Certificates
  - Create a self-signed Root Certificate -> **Certificate Authority**
  - Use its private key to sign ‘derived’ certificates
  - Distribute the Root Certificate



Design Phase – A case study
Device Certificates Explained

- How does server validation work?
  - Client inspects Root Certificate and **explicitly trusts it as Certificate Authority**
    - This is typically a preparation step
  - Device (server) sends certificate (chain)
  - Client checks info -> date, CN, Issued by
    - Issued by is now a **trusted party**
  - Client checks signature with public key **of Root Certificate**
    - Ensures info is valid **and trusted**
  - Client uses public key **of Device Certificate** to encrypt a message
    - Client sends encrypted message to server
  - Device decrypts message and sends back original message to client
    - Proves entity has the **Device** private key -> decryption can only be done with that key

Design Phase – A case study
Device Certificates Explained

- Key points of signing:
  - Root Private Key is **only** used for **signing**
  - Root Public Key is **only** used for **validation**
  - Device Private Key is **only** used for **decryption**
  - Device Public Key is **only** used for **encryption**
- More layers can be added – see for example https://www.ing.nl
  - Each layer up to the point of creating a device certificate is only meant for signing
  - Allows distribution of signing entities
  - Validation involves the complete chain
- Guarding of Private Keys of Signing Certificates is **essential**
  - **Key compromise means no certificates signed by it can be trusted anymore**

Design Phase – A case study
Device Certificates Explained

- Crucial part of certificate security is **trust of Certificate Authority**
  - Trust that the CA guards its private key and only supplies certificates to trusted parties

- Public Certificate Authorities distribute their certificates
  - Taken up into OS certificate stores / Browser certificate stores
  - Part of OS/Browser updates -> can be revoked
  - User never explicitly has to trust the CAs himself
    - Trust Microsoft / Red Hat / Mozilla etc to do it for you

- Private Certificate Authorities distribute their certificates
  - User must explicitly trust them
  - Typically manual action: download and import
  - Advantage is that this immediately **trusts all certificates signed by the chain -> Scalability!**

Design Phase – A case study
OMNEO Design

- Use Device Certificates instead of Pre-Shared Key for session authentication

- Design choices to be made:
  - Self-signed or CA-signed certificates?
  - If CA-signed: what Certificate Authority?
  - How to handle device private key?
  - What specific TLS ciphersuite?

Design Phase – A case study
OMNEO Design

- Self-signed or CA-signed?
  - OMNEO systems are very scalable -> system size up to 10.000 devices
  - CA-signed is the only way to go

- What Certificate Authority?
  - Bosch owns company Escrypt
  - Private Certificate Authority
  - Expertise at guarding root and signing certificate
  - Offer signing solution that can be used during manufacturing

Design Phase – A case study
OMNEO Design

- How to handle device Private key?
  - Add Secure Element to platform
  - Securely store key -> hardened device
  - Key never leaves element -> decryption is performed on Secure Element

- What specific TLS ciphersuite?
  - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
  - Session key exchange is done using ECDHE (Elliptic-Curve Diffie-Hellman Ephemeral)
  - Session Authentication is based on ECDSA (Elliptic-Curve Digital Signature Algorithm)
  - After handshake encryption via AES with session 128-bit key in GCM (Galois/Counter Mode) mode
  - After handshake SHA-256 is used as hashing (message authentication) algorithm

Design Phase – A case study
OMNEO Design

- OMNEO Controller (client) and Device (server) both have certificates
- TLS handshake should show both certificates being exchanged
- Tested using OpenSSL to emulate controller:

```
openssl s_client -connect <ipaddr>:55556 -cert <>.pem -key <>.key -CAfile
<ca_chain>.pem
```

THANK YOU

BOSCH