ORACLE®

JavaOne™

ORACLE®

Internet of Things:

Threats and counter measures with Java

Florian Tournier
Director, IoT Product Management
Oracle

Patrick Van Haver
Principal Engineer, Internet of Things
Oracle

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Program Agenda

1. Introduction to IoT Security
2. Concerns and threats
3. How Java can help to implement countermeasures
4. Considerations on IoT Infrastructure
5. Conclusion

Program Agenda

1. Introduction to IoT Security
2. Concerns and threats
3. How Java can help to implement countermeasures
4. Considerations on IoT Infrastructure
5. Conclusion

IoT security in the Press

- Some car-related headlines
  - BMW ConnectedDrive hack sees 2.2 million cars exposed to remote unlocking (02/02)
  - DARPA Hacks GM's OnStar To Remote Control A Chevrolet Impala (02/08)
  - US Senate Report: Automakers fail to fully protect against hacking (02/09)
  - Hackers take control of Jeep on the highway (August)
- Medical devices, industry automation, and other things
  - Hackers had struck an unnamed steel mill in Germany (Jan)
  - U.S. government probes medical devices for possible cyber flaws (Oct 14)

Privacy

Spying

Remote Control

Theft

Physical damage

Murder?

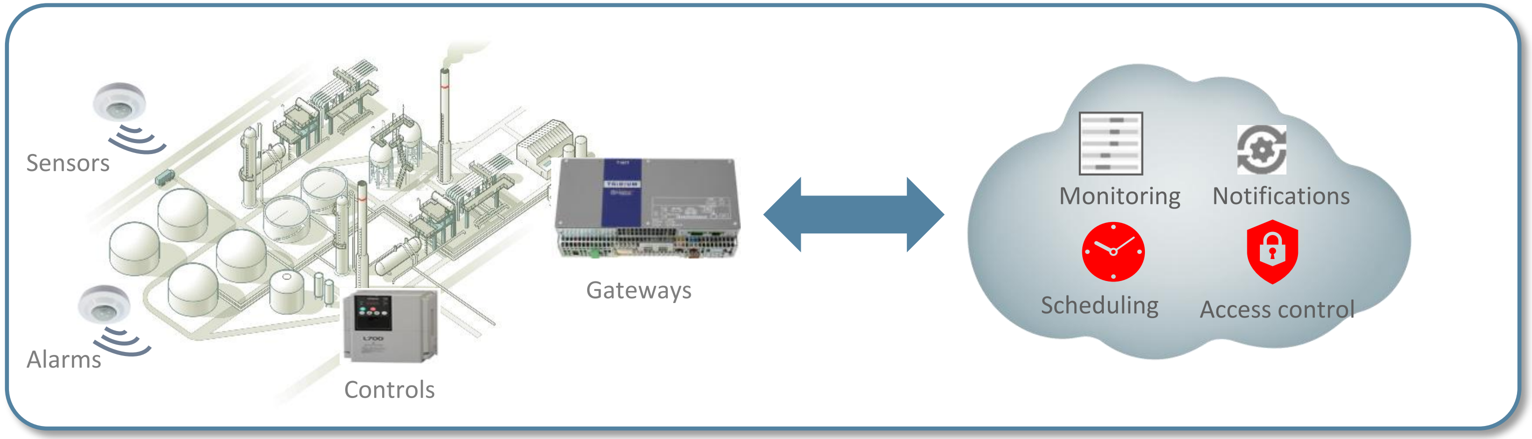
JavaOne ORACLE

Security Use Case : Industrial / Home Alarm System

Today : Alarm components vulnerabilities weaken the system

- Credentials can be reverse-engineered
- Compromised devices can be inserted and weaken the system
- Poor communication encryption enables man in the middle attacks

Hackers aim at modifying the behavior of alarm components	
Best Case	Worse Case
• False positive alarms	• Physical security breach • Damage to equipment / industrial accident



JavaOne ORACLE

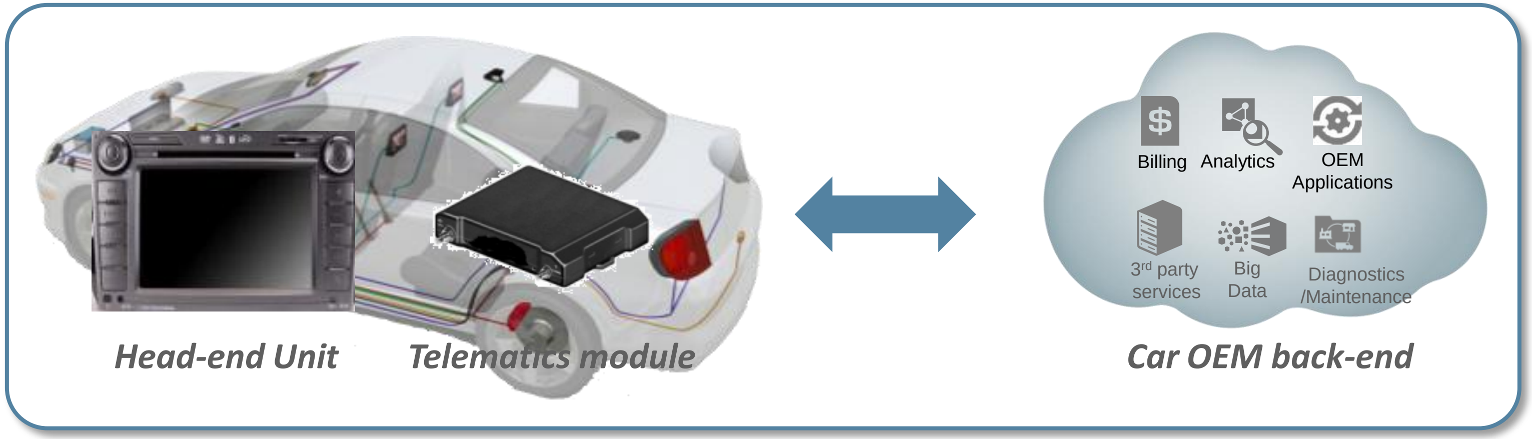
Security Use Case : Connected Car

Today : Multiple vulnerabilities in a car stack

- Credentials accessible in plaintext
- Poor crypto implementations

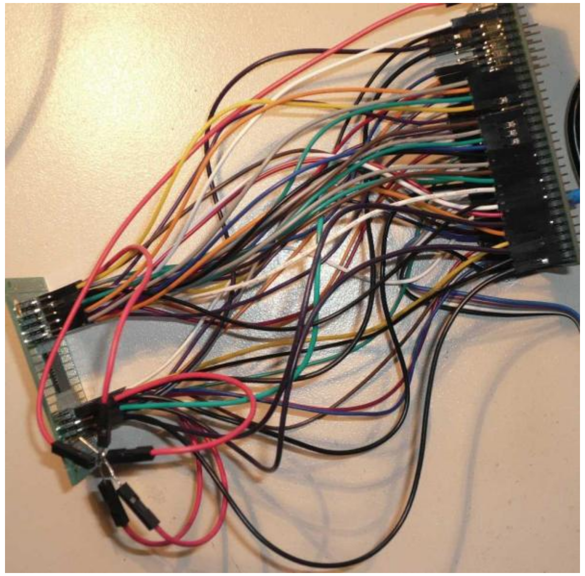
- Poorly implemented protocols
- Poor credential management and provisioning process

Hackers aim at controlling a car head unit or telematics remotely	
Best Case	Worse Case
• Disruption of in-car information / entertainment • Loss of confidential owner data/privacy	• Car unlock / theft • Disruption of car mechanics / loss of life



In Practice: A recent Car Hack

- A lab has been able to remotely open a (high-end brand) car
  – Reverse engineering the Remote Access features to identify **vulnerabilities**
  – Exploiting the vulnerabilities identified through an **attack path**
- The list of vulnerabilities is rather long
  – The same keys are used in all vehicles
  – Some messages are not encrypted
  – Configuration data is not tamper-proof
  – The crypto algorithm used (DES) is outdated and broken
  – The software does not include protection against replay attacks
- One fix: The communication is now encrypted using HTTPS



Safety vs. Security

Safety

- Protects against malfunction
  - Focus on quality
- Principles
  - Coverage analysis
  - Detection, mitigation, reaction
  - Simplicity is better
  - Redundancy helps

Security

- Protects against attackers
  - Focus on robustness
  - Several defence layers
- Principles
  - Coverage analysis
  - Detection, mitigation, reaction
  - Simplicity is better
  - Redundancy helps

Car Hack: Poor Decisions

Poor decision	Safety reasoning	Security reasoning
Using the same keys	Simple process No complex infrastructure	Keys need to be diversified A key needs to be broken on every car
No systematic encryption	Only critical messages are encrypted	A secure channel protects against reverse engineering
Configuration data no tamper-proof	Configuration data integrity is protected by a checksum	Configuration data authenticity is protected by a cryptographic checksum
The vehicle ID is in error messages	Simplify diagnosis by having the data	A remote attacker doesn't have the ID, so let's protect it
Using DES	Well-known, fast algorithm	DES is broken, let's mandate AES
No protection against replay attacks	Same message, same action	A recorded message cannot have the same effect when replayed

Threat Analysis

Thinking like an attacker

- Very important to validate a design
  - Identify the key assets and their flows
  - Analyze how security protections can be bypassed
  - Consider vulnerabilities as opportunities

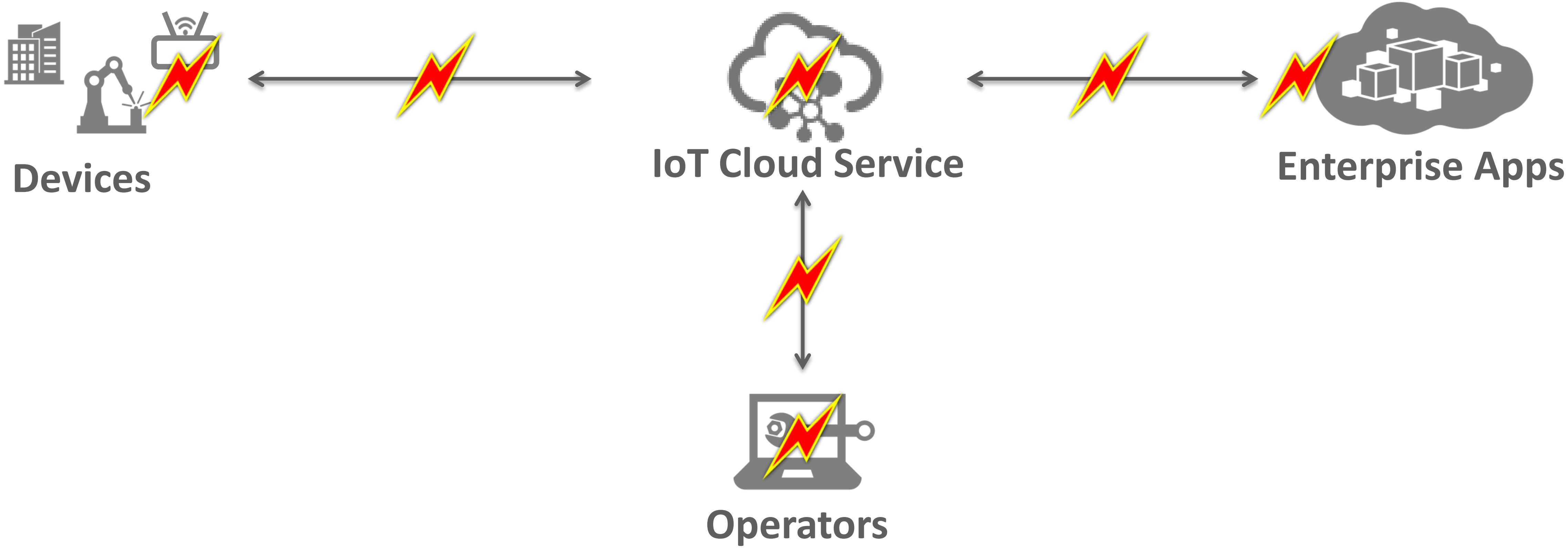
- Identify countermeasures to be added to the design
  - And loop again on the analysis

Program Agenda

1. Introduction to IoT Security
2. Concerns and threats
3. How Java can help to implement countermeasures
4. Considerations on IoT Infrastructure
5. Conclusion

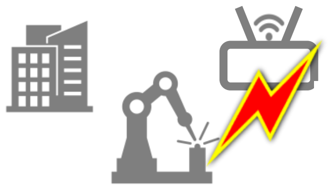
Attack surface

IoT Infrastructure – Attack surface



IoT Infrastructure – Attack surface

Attacking the device



Devices

Thinking like an attacker

- Is the software stack up-to-date?
- Is there any software update mechanism? How robust? Can I block-it? Use-it?
- Can I install my software?
- Is there any device authentication mechanism?
- What about devices connected via a gateway?
- Can I steal devices and reverse engineer? extract credentials?
- Are these credentials diversified or can be reused on other devices? Renewed?
- Need for RTC? Can I abuse RTC? Effects?
- What about physical attacks?
- …

IoT Infrastructure – Attack surface

Attacking the network traffic

Thinking like an attacker

- Can I replay messages?
- Can I spy messages?
- Can I send fake messages (data, signaling, …)?
- Can I modify messages? Modify priority? Timestamp? Payload?
- Can I do buffer overflow? code injection?
- Can I perform MITM attack? DoS?
- Can I insert a fake device to abuse the server? A fake server?
- How does it behaves if I simulate network contentions?
- …

IoT Infrastructure – Attack surface

Attacking the users

Devices

Thinking like an attacker

- How the system is configured & managed?
- Which authentication mechanism?
- What operations are authorized?
- Type of workstation used? Dedicated?
- Can I install software on these machines?
- What about social engineering attacks?
- …

terprise Apps

Operators

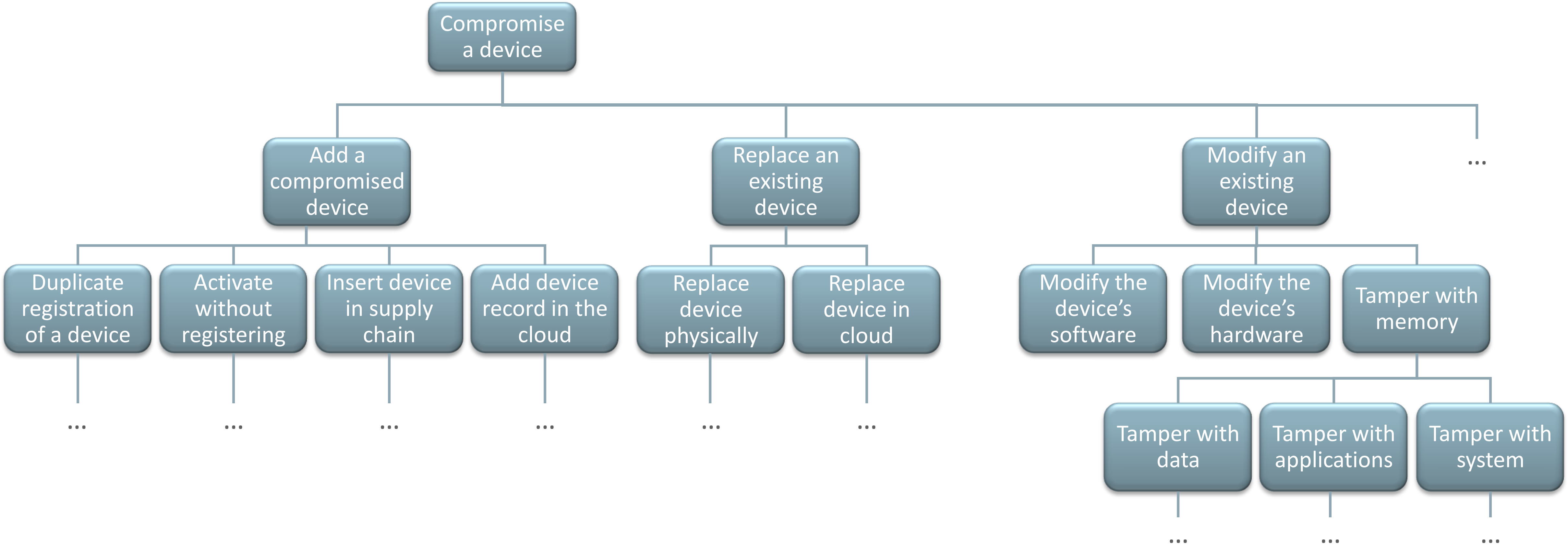
Attack surface

- VERY large attack surface
  - Local or remote attacks
  - Logical or physical attacks
  - Multiple targets (client device, client applications, server, users, operators, …)
- Requires a consistent and global approach

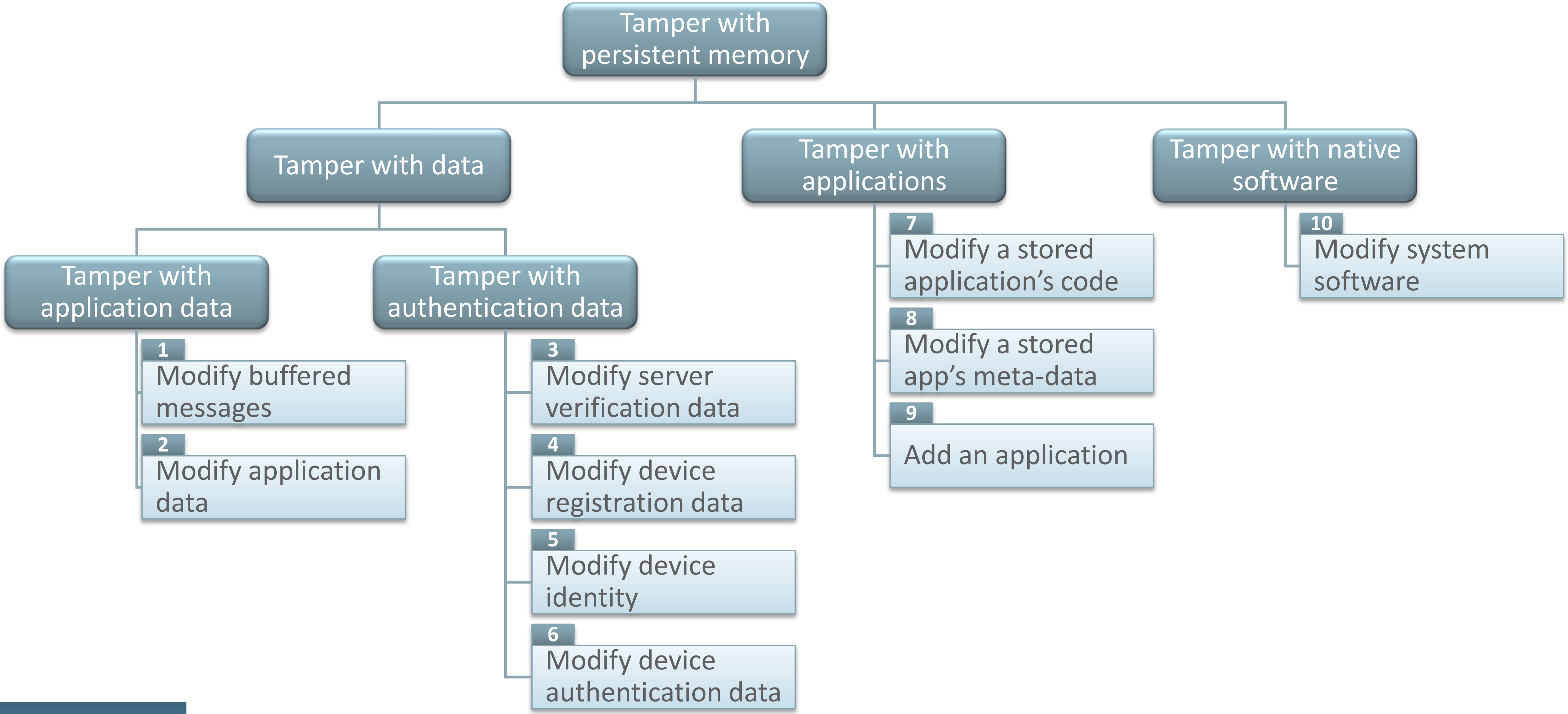
Example

Attacking the authentication credentials used by a device

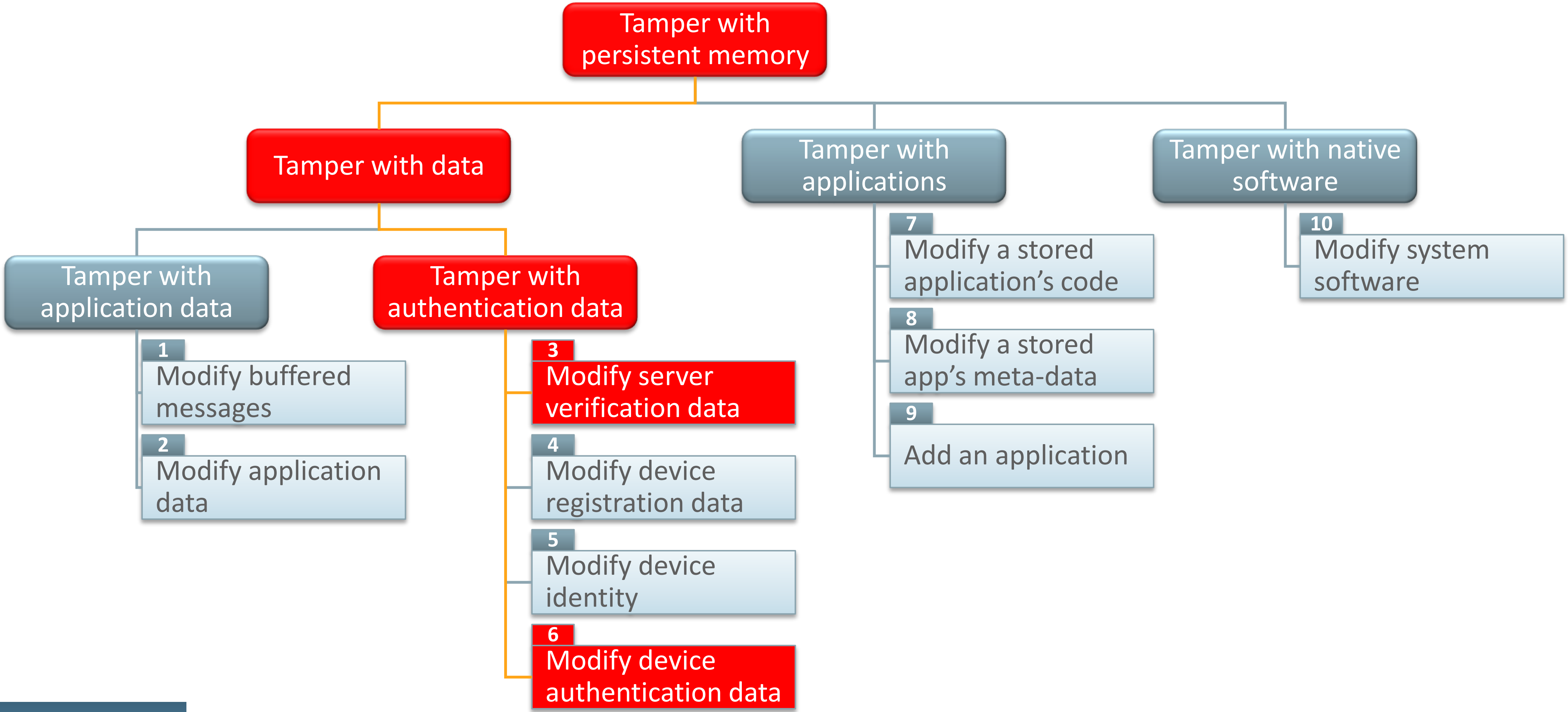
Compromising a Device



Compromising a Device



Compromising a Device



Scenario and Assets involved

- Typical scenario
  - The device connects to the IoT server and authenticate it
  - The device post data to the IoT server (requires authentication & authorization)
- Assets on the device:
  - Trust Anchors: used by device to authenticate the server
  - Key Pair: used by the device to get authenticated by the server

Counter measures to protect these assets

- Do the best to protect assets on device
  - Integrity, confidentiality, access control
- Detect suspicious behavior
  - Observe behavior in regards to lifecycle state, RBAC policy…
- React in case of attack
  - Black list the device, revoke current credentials, …

Device side

Server side

Program Agenda

1 Introduction to IoT Security

2 Concerns and threats

3 How Java can help to implement countermeasures

4 Considerations on IoT Infrastructure

5 Conclusion

Example: Protecting IoT Authentication Assets on a Device

- **Security Design goals**
  - **Separation of concerns**
    - Do not expose application developers to the handling of credentials
    - Encapsulation of the credentials and operations using them
  - **Protection of trust material and credentials**
    - Ensure secure storage (integrity, confidentiality) of Trusted Anchors and device KeyPair
  - **Extensibility to adopt different strategies when hardware allows**
    - Ability to use hardware security when available on a device

JavaOne ORACLE

Encapsulation of credentials

```
public interface TrustedAssetsManager extends javax.net.ssl.X509TrustManager {

   // encapsulation of the trust anchors
   void checkServerTrusted(X509Certificate[] chain, String authType) throws CertificateException;
   void checkClientTrusted(X509Certificate[] chain, String authType) throws CertificateException;
   public X509Certificate[] getAcceptedIssuers();

   // encapsulation of the Private Key
   void generateKeyPair(String algorithm, int size) throws GeneralSecurityException;
   PublicKey getPublicKey();
   public byte[] signWithPrivateKey(byte[] data, String algorithm) throws GeneralSecurityException;

}
```

- How does it address design goals?
  - Private key is not exposed to the application to avoid unexpected leaks
  - Trust Anchors are checked internally during authentication process

Generation of the KeyPair

```
import java.security.KeyPairGenerator;

void generateKeyPair(String algorithm, int size) throws GeneralSecurityException {
   […]

   // create a key generator using the specified Provider
   KeyPairGenerator kpg = KeyPairGenerator.getInstance(algorithm, this.getProvider());

   // generate the keypair
   kpg.initialize(keySize);
   this.keyPair = kpg.generateKeyPair();
}
```

- How does it address design goals?
  - Private key never extracted from the device, only public key exported
  - Key is unique per device & renewable (time before expiration can be configured in server policy)

JavaOne ORACLE

Store the Private Key in a KeyStore

```
import java.security.KeyStore;

private void store(KeyPair kp, String alias, KeyStore.Builder builder)
                                         throws GeneralSecurityException {
   // create a key store
   KeyStore ks = builder.getKeyStore();

   // create an entry for the private key
   KeyStore.PrivateKeyEntry entry = new KeyStore.PrivateKeyEntry(
                                                    kp.getPrivate(),
                                                    this.getCertificate(kp.getPublic()));
   // store private key
   ks.setEntry(alias, entry, builder.getProtectionParameter(alias));

}
```

- How does it address design goals?
  - KeyStore provides integrity and confidentiality
  - Use of KeyStore.Builder allows to compute a unique password to access KS (not hardcoded in the application)

Sign data

```
import java.security.Signature;

public byte[] signWithPrivateKey(byte[] data, String algorithm) {
   […]

   Signature s;

   // create a Signature object using the specified Provider
   s = Signature.getInstance(algorithm, this.getProvider());

   // sign
   s.initSign(this.keyPair.getPrivate());
   s.update(data);
   return s.sign();
}
```

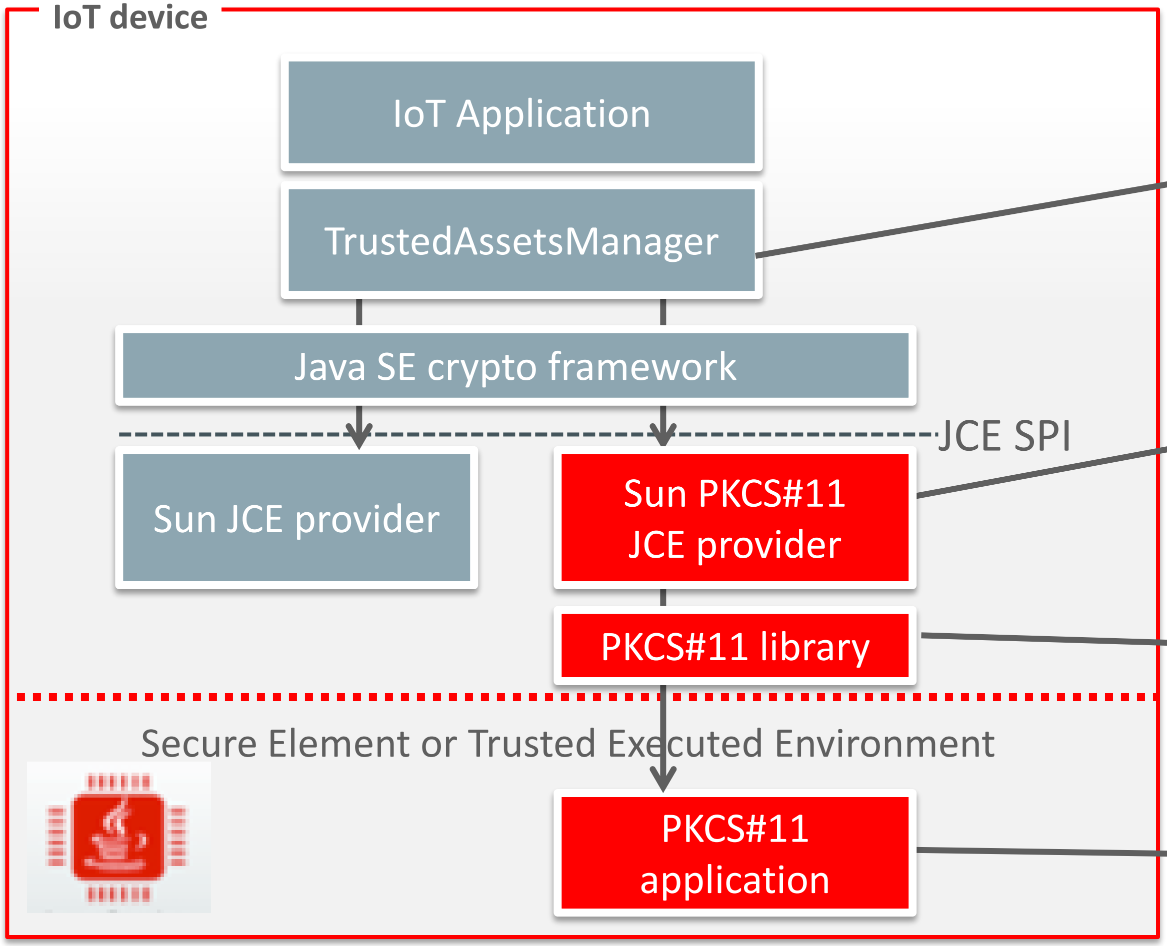
- How does it address design goals?
  - Here the example is simplified (sign everything: might expose the key to unexpected use)
  - Could be improved by generating and signing the authorization/token request

JavaOne ORACLE

Making use of Hardware Security when available

- Goal:
  - Delegate operations to a security token
  - Security token is a dedicated hardware to securely perform crypto operations
    - Could be embedded Secure Element (eSE) or a Trusted Execution Environment (TEE)
    - Provides physical isolation from applications running on the device
    - eSE provides tamper resistance against physical attacks
    - TEE protects against logical attacks and offers more processing power (e.g. could process the whole TLS flow)
- How to do that?
  - Using a specific security JCE Provider…

Making use of Hardware Security when available



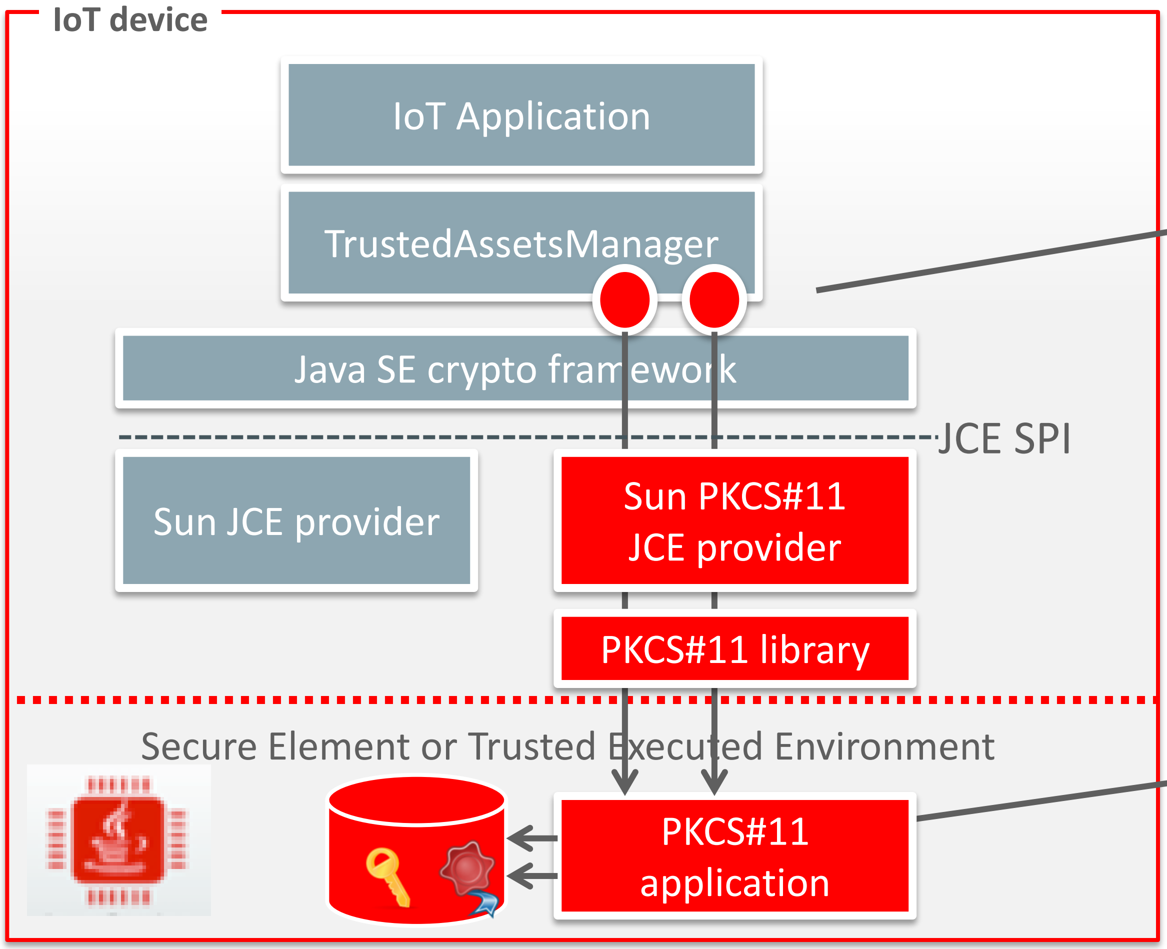
Code need to select and configure PKCS#11 Provider

Include SunPKCS#11 jar
(Additional package to Java SE Embedded compact profiles)

Include library (& drivers) to access the PKCS#11 hardware token

Processing and storage of credential performed in an isolated environment

Making use of Hardware Security when available



Crypto objects (KeyPair, Certificate) are proxies redirecting calls to

Real credentials are stored and processed in the secure hardware

Impacts on the code?

```
// Use PKCS11 provider instead of default
Provider getProvider() {
   String conf = "library = my.lib …"; // PKCS11 configuration parameters
   return new sun.security.pkcs11.SunPKCS11(new ByteArrayInputStream(conf.getBytes()));
}

// The key generator will use this provider and redirect generation into the hardware token
KeyPairGenerator kpg = KeyPairGenerator.getInstance(algorithm, this.getProvider());

// The KeyStore.Builder must create a PKCS11 key store, from PKCS#11 provider:
// storage into this KS will be redirected o the hardware token
KeyStore.Builder ksb = KeyStore.Builder.newInstance("PKCS11", this.getProvider(), pwdProtection);

// The signature will then be performed in the token
Signature s = Signature.getInstance(algorithm, this.getProvider());
```

Program Agenda

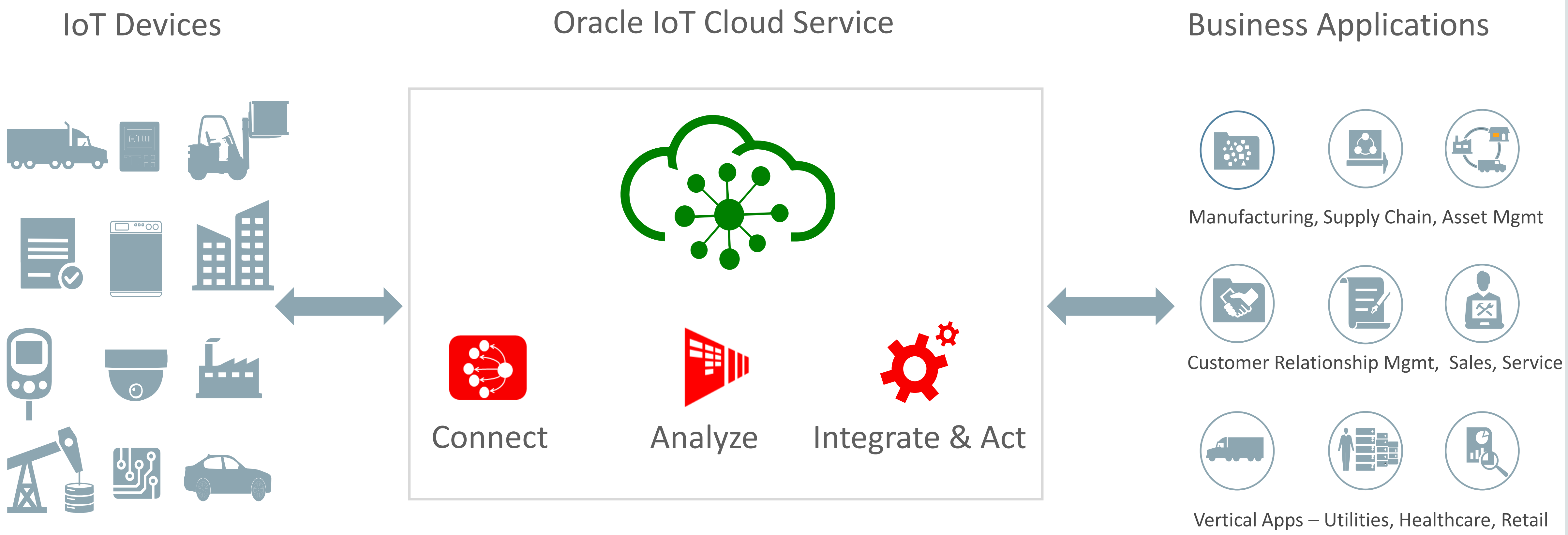
1. Introduction to IoT Security
2. Concerns and threats
3. How Java can help to implement countermeasures
4. **Considerations on IoT Infrastructure**
5. Conclusion

Beyond the Device

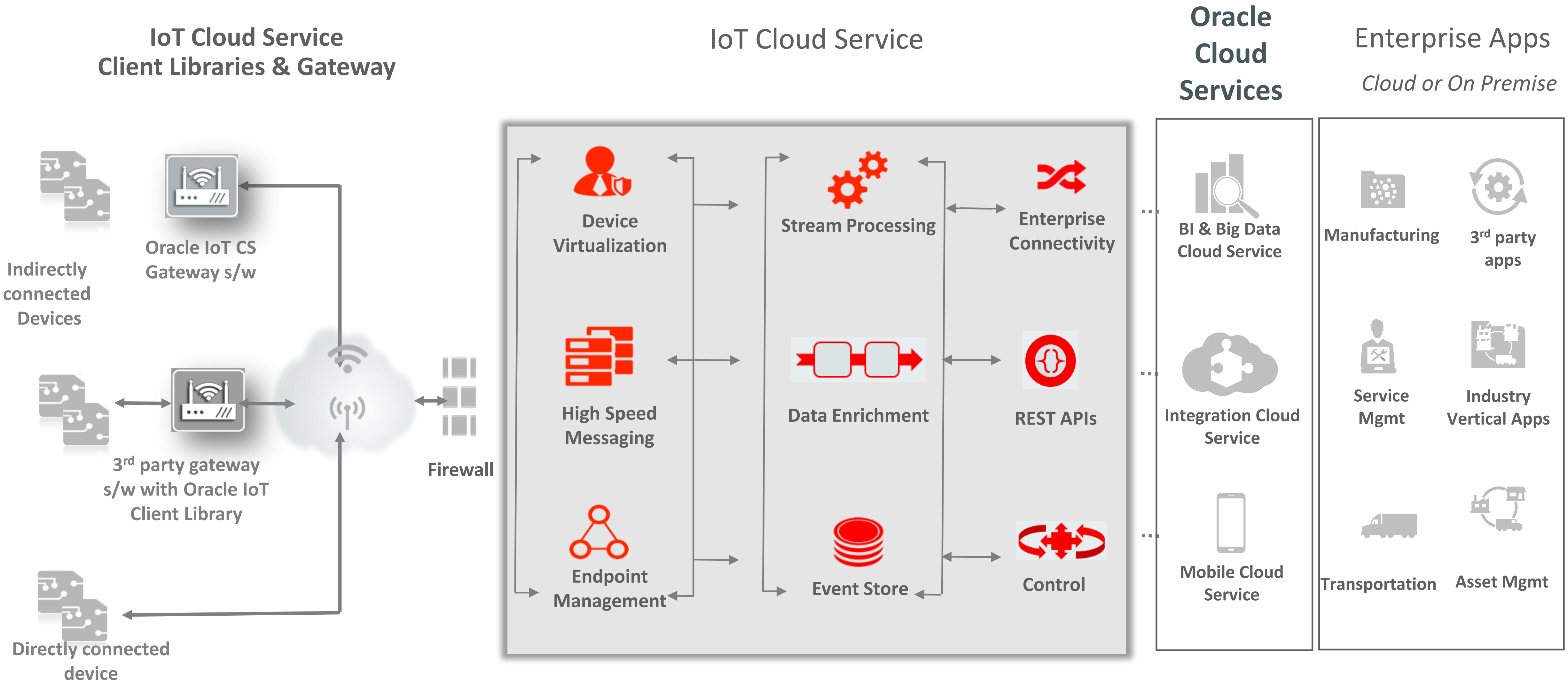
(Some) IoT Infrastructure Requirements for IoT Security

- Protocol security
  - Implementing the protocols securely, with proper options
  - Managing the credentials appropriately
- Device lifecycle management
  - Ensuring that the infrastructure cannot be abused by fake/compromised devices
- Principal authentication and authorization
  - Applying adequate authentication for both devices and users
  - Strictly control access of sensitive operations to authorized principals

An Example of IoT Infrastructure – Oracle IoT Cloud Service



Oracle Internet of Things Cloud Service - Architecture



Oracle IoT Cloud Service – Approach to security

TRUSTED DEVICES

- Security mechani… provision… …ges tr… …nships with de… …ces – uses OAuth2
- Uniquely assigned… …ice identitie… …use of… …redentials ac… …ss devices

Registration

Public Key Auth

NON REPUDIATION

- Enforces authentica… …prior to commu… …any de… …rprise software, ena… …ng proof of origin of data
- Transport level sec… …for all communica… …ure data inte…

OAuth

HTTPS

SECURITY LIFECYCLE

- Secure, managed st… transitions… …ccess from…
- Restricts types of IoT C… operations tha… …n perfo…

Lifecycle

Role-based Access Control

JavaOne ORACLE

Program Agenda

1. Introduction to IoT Security
2. Concerns and threats
3. How Java can help to implement countermeasures
4. Considerations on IoT Infrastructure
5. **Conclusion**

Conclusion

- Internet of Things is growing fast, new products
- Lot of interest from hackers – scale & visibility
- Very large attack surface requiring consistent approach and methodology
- Security is a dynamic process – needs to be continuously improved through threat analysis, code checking and countermeasure implementation
  - Implementing a security roadmap is an important step to stay ahead of the attackers
- There is an ecosystem of providers looking to provide infrastructure and components to strengthen system security
  - Java technologies can help securing the Internet of Things
  - Oracle IoT CS can provide security foundations for the IoT infrastructure

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Integrated Cloud

Applications & Platform Services

JavaOne™

ORACLE®

ORACLE®