SECURE INTEGRATION OF CRYPTOGRAPHIC SOFTWARE

Speaker: Stefan Krüger
When a Developer Uses a Crypto API

```java
KeyGenerator keyGen = KeyGenerator.getInstance("AES");
keyGen.init(128);
SecretKey key = keyGen.generateKey();

Cipher cipher = Cipher.getInstance("AES");
cipher.init(Cipher.ENCRYPT_MODE, key, null);
cipher.doFinal(data);
```

Uses Electronic Codebook (ECB)
The Average Developer is no Crypto Expert

- 88% of Android apps contain at least one misuse
- 83% of crypto-related vulnerabilities result from API misuse
- Popular vendors misuse TLS libraries
What shall we do about it?

AES, RSA, CBC ...

Private Data, Passwords, Data Encryption
What shall we do about it?

AES, RSA, CBC

Private Data, Passwords, Data Encryption
What shall we do about it?

AES, RSA, CBC ...

Private Data, Passwords, Data Encryption
What shall we do about it?

AES, RSA, CBC ...

Private Data, Passwords, Data Encryption

JAR
What shall we do about it?

AES, RSA, CBC ...

Private Data, Passwords, Data Encryption
What shall we do about it?

AES, RSA, CBC ...

Private Data, Passwords, Data Encryption

+ AES, RSA, CBC ...

JAR

FIG. 3
What shall we do about it?

AES, RSA, CBC …

Private Data, Passwords, Data Encryption

FIG. 3
What shall we do about it?

AES, RSA, CBC ...

Private Data, Passwords, Data Encryption

FIG. 3
What shall we do about it?

AES, RSA, CBC ...

Private Data, Passwords, Data Encryption

CogniCrypt
What shall we do about it?

AES, RSA, CBC...

Private Data, Passwords, Data Encryption

CogniCrypt

JAR

FIG 3
CogniCrypt supports Developers as an Eclipse Plugin
CogniCrypt supports Developers as an Eclipse Plugin
CogniCrypt’s Components
CogniCrypt\textsubscript{GEN} – Code Generation For Crypto APIs
CogniCrypt\textsubscript{GEN} – Code Generation For Crypto APIs

CogniCrypt\textsubscript{GEN} – Code Generation For Crypto APIs

Task Descriptions
Algorithm Model
Code Template

But… I have to Change My Code!
CogniCrypt\textsubscript{SAST} – Detection of Crypto-API Misuses
CogniCrypt\textsubscript{SAST} – Detection of Crypto-API Misuses


© CRC 1119 CROSSING, TU Darmstadt/Heinz Nixdorf Institut
Folie 22
...within Eclipse...!

- Save
- Static Analysis
- Error Markers
Main Error Types of CogniCrypt$_{SAST}$

- Insecure Parameters
- Incorrect Usage Pattern
- Insecure Class Composition
I.E. Error Type: Insecure Parameters

```java
KeyGenerator keyGen = KeyGenerator.getInstance("AES");
keyGen.init(128);
SecretKey key = keyGen.generateKey();

Cipher cipher = Cipher.getInstance("AES");
cipher.init(Cipher.ENCRYPT_MODE, key, iv);
cipher.doFinal(data);
```

“AES" should rather be „AES/{CBC/GCM}“

[CrySL: An Extensible Approach to Validating the Correct Usage of Cryptographic APIs. Stefan Krüger, Johannes Spaeth, Karim Ali, Eric Bodden, Mira Mezini. ECOOP 2018.]
CrySL – Specifying The Use of Crypto APIs (1/2)

**SPEC** javax.crypto.KeyGenerator

**OBJECTS**
- int keySize;
- javax.crypto.SecretKey key;
- java.lang.String alg;

**EVENTS**
- g1: getInstance(alg);
- g2: getInstance(alg, _);
- Gets := g1 | g2;
- i1: init(keySize);
- i2: init(keySize, _);
- i3: init(_);
- i4: init(_, _);
- Inits := i1 | i2 | i3 | i4;
- gk: key = generateKey();

**ORDER**
- Gets, Inits?, gk

**CONSTRAINTS**
- alg in {"AES"} => keySize in {128, 192, 256};
- alg in {"DES"} => keySize in {56};
- alg in {"Blowfish"} => keySize in {40, 44, 48, 52, 56, ..., 436, 440};

[CrySL: An Extensible Approach to Validating the Correct Usage of Cryptographic APIs. Stefan Krüger, Johannes Spaeth, Karim Ali, Eric Bodden, Mira Mezini. ECOOP 2018.]
CrySL – Specifiying The Use of Crypto APIs (2/2)

ENSURES

generatedKey(key, alg);

SPEC javax.crypto.Cipher

... 

REQUIRES

generatedKey(key, alg(transformation));

ENSURES

encrypted(plaintText, cipherText);

[CRYSL: An Extensible Approach to Validating the Correct Usage of Cryptographic APIs. Stefan Krüger, Johannes Spaeth, Karim Ali, Eric Bodden, Mira Mezini. ECOOP 2018.]
CrySL Rules for the Java Cryptographic Architecture (JCA)

Encryption & Decryption

Signing & Verification

Key Generation

Java

23 Classes & Interfaces

Randomness

Hashing and MACs

Persisting of Key Material

[CrySL: An Extensible Approach to Validating the Correct Usage of Cryptographic APIs. Stefan Krüger, Johannes Spaeth, Karim Ali, Eric Bodden, Mira Mezini. ECOOP 2018.]
Evaluation - Android

10k Android Apps

96% of these Apps are insecure

[CrySL: An Extensible Approach to Validating the Correct Usage of Cryptographic APIs. Stefan Krüger, Johannes Spaeth, Karim Ali, Eric Bodden, Mira Mezini. ECOOP 2018.]
Evaluation - Maven

2.7+ Million Software Artefacts

73% of these Artefacts are insecure
When a Developer Uses a Crypto API

KeyGenerator keyGen = KeyGenerator.getInstance("AES");
keyGen.init(128);
SecretKey key = keyGen.generateKey();

Cipher cipher = Cipher.getInstance("AES");
cipher.init(Cipher.ENCRYPT_MODE, key, iv);
cipher.doFinal(data);

getInstance(String transformation) : Cipher - Cipher
getInstance(String transformation, Provider provider) : Cipher - Cipher
getInstance(String transformation, String provider) : Cipher - Cipher
What shall we do about it?

KeyGenerator
keyGen.in
SecretKey
Cipher c
cipher.in
cipher.do

AES, RSA, CBC...

Private Data,
Passwords,
Data Encryption

CogniCrypt

JAR
When a D

CogniCrypt’s Components

KeyGen
keyGen
SecretKey
Cipher
cipher
Ciphertext

CogniCrypt\_GEN

CogniCrypt\_SAST
Evaluation

10k Android Apps
96% of these Apps are insecure

2.7+ Million Software Artefacts
73% of these Artefacts are insecure
When a Developer Uses a Crypto API

```java
KeyGenerator keyGen = KeyGenerator.getInstance("AES");
keyGen.init(128);
SecretKey key = keyGen.generateKey();
Cipher cipher = Cipher.getInstance("AES");
cipher.init(Cipher.ENCRYPT_MODE, key, iv);
cipher.doFinal(data);
```

What shall we do about it?

CogniCrypt's Components

Evaluation

- 10k Android Apps
- 96% of these Apps are insecure
- 2.7+ Million Software Artefacts
- 73% of these Artefacts are insecure

Maven

The Central Repository