Codyze

Werkzeug zur Prüfung der korrekten Nutzung von Kryptobibliotheken

22.01.2020

Dennis Titze

Fraunhofer

Informationen zum Projekt

- Entwicklung eines automatischen Werkzeugs zur Prüfung korrekter Nutzung kryptografischer Bibliotheken
- Auftraggeber: Bundesamt für Sicherheit in der Informationstechnik
- Bearbeitung durch das Fraunhofer Institut für Angewandte und Integrierte Sicherheit (AISEC)
- Laufzeit: Februar 2019 – April 2020

Fraunhofer AISEC

Motivation

Fraunhofer AISEC

Motivation

- Was bedeutet korrekte Nutzung?
  - Funktional korrekt?
  - Performant?
  - Sicher?
- Woher kommt inkorrekte Nutzung?

Fraunhofer AISEC

Motivation – was bedeutet korrekte Nutzung?

```
public class EncryptUtil {

    private static byte[] key = { 0x74, 0x68, 0x69, 0x73, 0x49, 0x73, 0x41, 0x53, 0x65, 0x63, 0x72, 0x65, 0x74, 0x4b, 0x65, 0x79 };

    public static String encrypt(String strToEncrypt) {
        try {
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
            final SecretKeySpec secretKey = new SecretKeySpec(key, "AES");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);
            // for encrypt password like LDAP password
            final String encryptedString = Base64.encodeBase64String(cipher.doFinal(strToEncrypt.getBytes(StandardCharsets.UTF_8)));
            return encryptedString;
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage(), e);
        }
    }
```

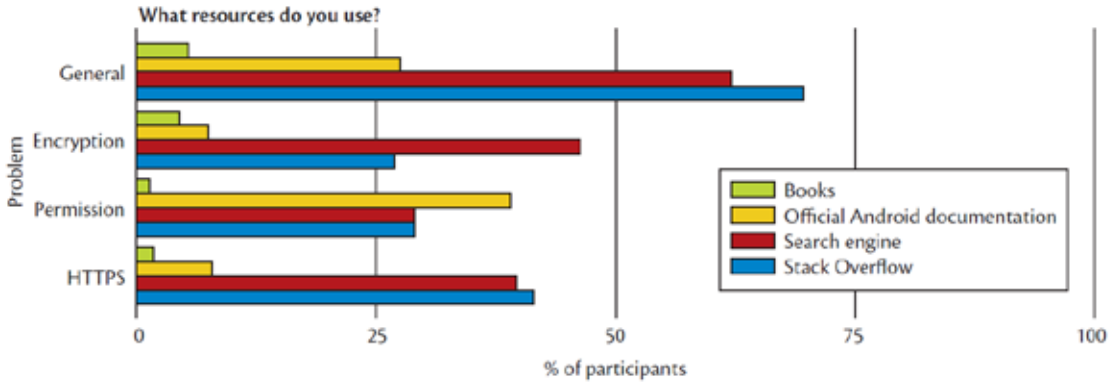

Fraunhofer AISEC

Motivation

- Woher kommt inkorrekte Nutzung?

Fraunhofer AISEC

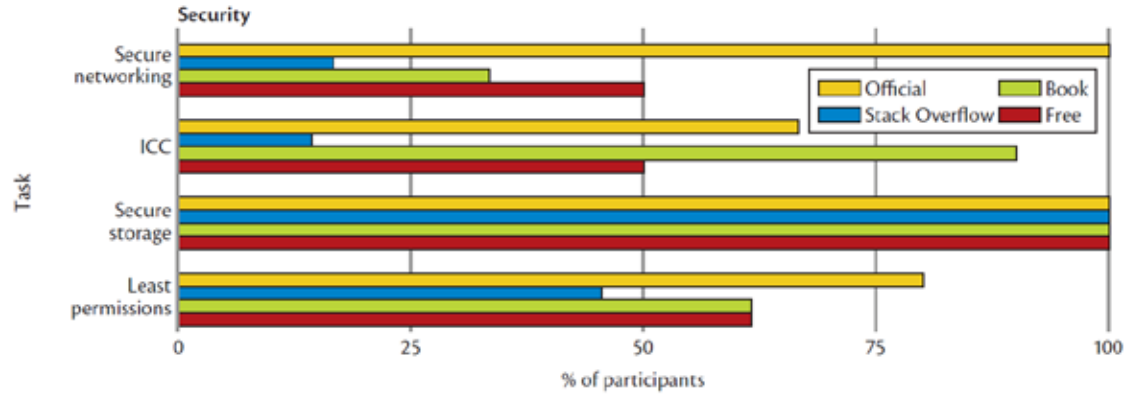
Motivation – woher kommt inkorrekte Nutzung?



Quelle: How Internet Resources Might Be Helping You Develop Faster but Less Securely

Fraunhofer AISEC

Motivation – woher kommt inkorrekte Nutzung?



Quelle: How Internet Resources Might Be Helping You Develop Faster but Less Securely

Fraunhofer AISEC

Motivation

- Entwickler oft überfordert, wie Bibliotheken korrekt eingesetzt werden müssen
- Hohe Anzahl an Sicherheitslücken durch falsche Nutzung kryptografischer Bibliotheken

➢ Tool zur Überprüfung von Quelltext auf korrekte Nutzung von (Krypto-)Bibliotheken notwendig
  - Leichte Konfigurierbarkeit der *Regeln* für Bibliotheksentwickler
  - Leichte Bearbeitung der *Regeln* auch für nicht-IT-Experten
  - Unterstützung direkt bei der *Entwicklung*

Fraunhofer AISEC

Codyze

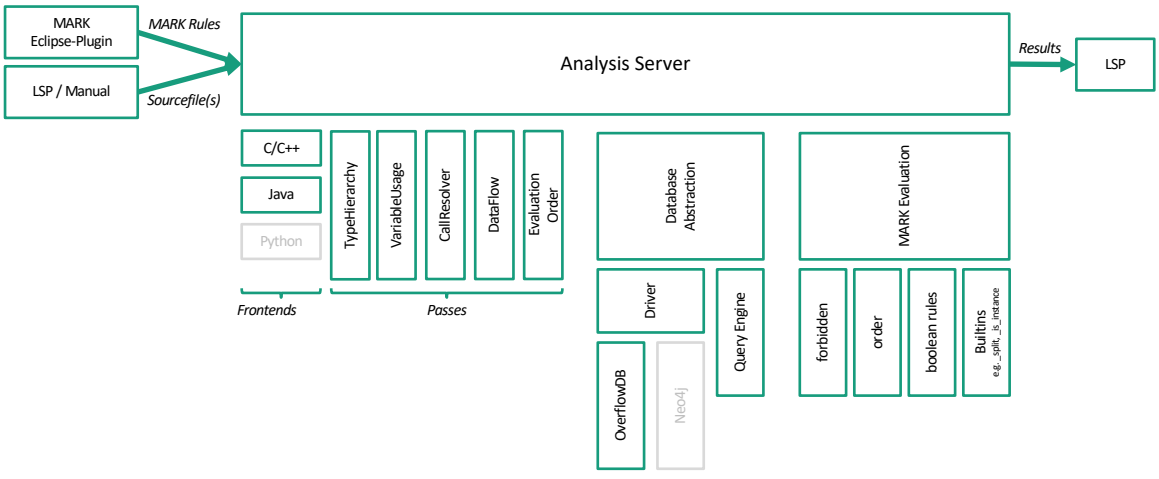
Fraunhofer AISEC

Projektaufbau

- Quelltext (Java, C, C++) wird in einen sog. ***C**ode **P**roperty **G**raph (CPG)* übersetzt
  - Möglich auch für unvollständigen Code (bspw. bei fehlenden Includes)
  - CPG enthält mehr Informationen als der reine Quelltext (bspw. Datenflüsse)
- Regeln zur korrekten Benutzung kryptografischer Bibliotheken werden in der ***M**odellierungssprache für **A**nforderungen und **R**ichtlinien der **K**ryptografie (MARK)* geschrieben
  - MARK-Regeln definieren die Auswertungsschritte
  - Übersetzung der Regeln in Anfragen an den CPG und weitere Auswertung
- Ergebnisse werden direkt in der Entwicklungsumgebung angezeigt

Fraunhofer AISEC

Architektur



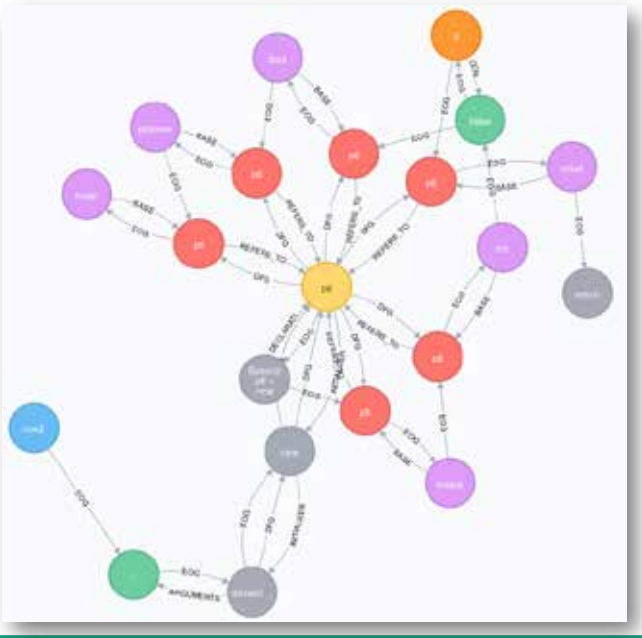
Fraunhofer AISEC

Code **P**roperty **G**raph (CPG)

Fraunhofer AISEC

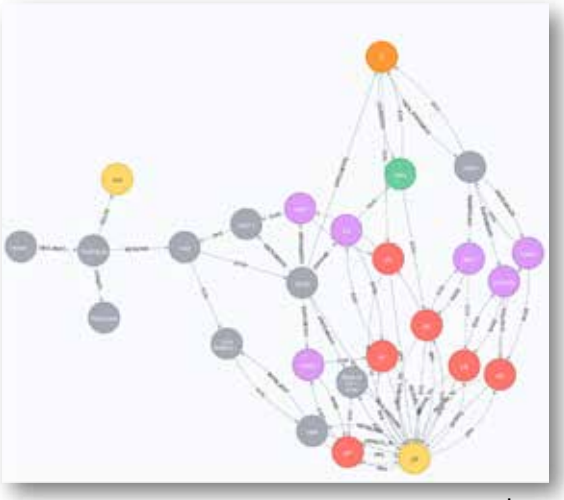
Kleines Beispiel

```
void nok2() {
  Botan2 p6 = new Botan2(1);
  p6.create();
  p6.init();
  if (false) {
    p6.start();
    p6.process();
    p6.finish();
  }
  p6.reset();
}
```

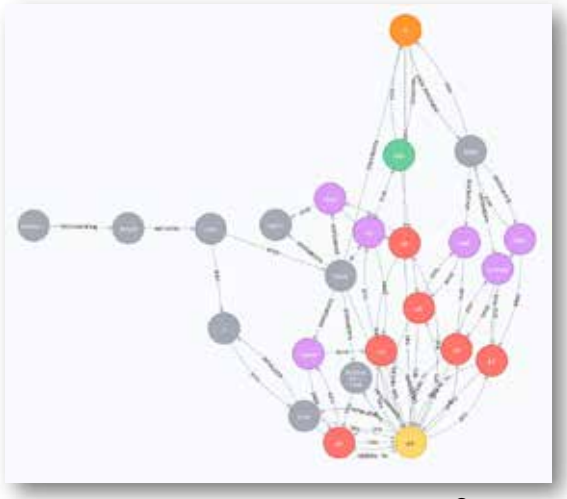


Fraunhofer AISEC

Sprachenunabhängiger* CPG



Java



C++

Fraunhofer AISEC

Großes Beispiel

- Beispiel (C++)
  - 377 kB
  - 12k Codezeilen
  - 486 Methoden
- Ergebnis
  - 28k Knoten
  - 80k Kanten
- Analysedauer
  - 1s Parsen Quelltext
  - 2.7s Konvertieren zu CPG
  - 33s Passes
  - 8s Speichern In-Memory-DB
  - 0.4s MARK Evaluation

Fraunhofer AISEC

CPG - Evaluierung

- 100.181 zufällige Sourcecode Dateien von Github
  - 59.317 C/C++
  - 40.864 Java
- Viele Dateien mit Syntaxfehlern oder fehlenden Defines (13451 C/C++, 66 Java)
- Durchschnittliche Laufzeiten
  - <1s für Parsen und Passes
  - <1s für Persistieren in In-Memory-DB

Fraunhofer AISEC

C/C++ Best-of

```
void foo(int i) {
    switch (i % 2) do {
        case 0:
            printf("0");

        case 1:
            printf("1");

    } while (i-- > 0);
}

int main() {
  foo(2);
  printf("\n");
  foo(1);
}
```

```
> g++ foo.cpp && ./a.out
010101
101
```

- Duff's Device

Fraunhofer AISEC

MARK-Regeln

Fraunhofer AISEC

MARK-Regeln

```
entity Cipher_Mode isa Cipher {
    var nonce : uint8_t;
    var nonce_length : std::size_t;
    var iv : Botan::InitializationVector;
    var algorithm : std::string;

    op create() {
        Botan::get_cipher_mode(algorithm, _);
    }

    op init() {
        Botan::set_key(_);
        Botan::set_key(_, _);
    }

    op start() {
        Botan::start(iv);
        forbidden Botan::start_msg(*);
    }

    […]

}
```

```
rule UseOfBotan_CipherMode {
    using Order as cm
    ensure
        order cm.create(), cm.init(), cm.start()
    onfail WrongUseOfBotan_CipherMode
}

rule BlockCiphers {
    using Botan::Cipher_Mode as cm
    ensure
        _split(cm.algorithm, "/", 0) in [ "AES" ]
    onfail WrongBlockCipher
}

rule UseRandomIV {
    using Botan::Cipher_Mode as cm,
          Botan::AutoSeededRNG as rng
    when _split(cm.algorithm, "/", 1) == "CBC" &&
          cm.direction == Cipher_Dir::ENCRYPTION
    ensure
        _receives_value_from(cm.iv, rng.random)
    onfail NoRandomIV
}
```


Fraunhofer AISEC

Integration IDE

Fraunhofer AISEC

Integration IDE

- Integration via Language Server Protocol (LSP)
  - Plugins für verschiedene IDEs verfügbar (Visual Studio Code, IntelliJ)
  - Eigene Plugins im Projekt für Eclipse und Visual Studio

Fraunhofer AISEC

Ausblick

Fraunhofer AISEC

Verfügbarkeit

- Veröffentlichung auf Github
- Preview möglich (Mail an AISEC)
- Präsentation Evaluierung und finales Werkzeug auf OMNISECURE 2021

Fraunhofer AISEC

Diskussion

- Welche Probleme könnte ein solches Tool bei Ihnen lösen?
- Fragen zum Ansatz
- Fragen zum Werkzeug

Fraunhofer AISEC

Kontakt

Dr. Dennis Titze
dennis.titze@aisec.fraunhofer.de

Fraunhofer AISEC
Service & Application Security
Lichtenbergstr. 11
85748 Garching (near Munich)
https://www.aisec.fraunhofer.de