Zentrum für sichere Informationstechnologie – Austria
Secure Information Technology Center – Austria

A-SIT

Seidlgasse 22 / 9, 1030 Wien
Tel.: (+43 1) 503 19 63–0
Fax: (+43 1) 503 19 63–66

Inffeldgasse 16a, 8010 Graz
Tel.: (+43 316) 873-5514
Fax: (+43 316) 873-5520

http://www.a-sit.at
E-Mail: office@a-sit.at

DVR: 1035461
ZVR: 948166612
UID: ATU60778947

HANDBUCH DEMONSTRATOR ZUM POLICY-ENFORCEMENT IN VERTEILTEN UMGEBUNGEN

Version 1.0 vom 31.10.2019
Bernd Prünster – bernd.pruenster@a-sit.at

Inhaltsverzeichnis

1. Einleitung

Der vorliegende Demonstrator veranschaulicht die Durchsetzung von Zugriffskontrollen für verschlüsselte Daten ohne zentrale Kontrollinstanz. Umgesetzt wurde der Demonstrator als Peer-to-Peer-Netzwerk, bzw. aufbauend auf einer Peer-to-Peer-Netzwerkbasis, welche um Semantik für den Austausch von Policies und deren Auswertung erweitert. Da es sich um einen Demonstrator handelt, welcher lediglich als Basis für die Umsetzung des Konzepts in Produktivumgebungen zu verstehen ist, wurde zur einfacheren Veranschaulichung von Policy-Definitionen auf *Easy Rules*[1] und *MVEL*[2] als Expression Language zurückgegriffen. Exemplarisch wurde eine Policy bereitgestellt, welche vordefinierten Empfängern, bzw. Empfängerinnen Zugriff auf verschlüsselte Daten gewährt.

2. Systemvoraussetzungen

Die Implementierung ist in *Kotlin* umgesetzt und verwendet *Apache Maven* als Buildsystem. Konkret werden Apache Maven 3.5 und Java 11 vorausgesetzt.

3. Installation

Das bereitgestellte Paket beinhaltet neben Source Code einen JUnit-Test-Case zu Demonstrationszwecken und erfordert keine Installation.

[1] https://github.com/j-easy/easy-rules
[2] http://mvel.documentnode.com/

3.1. Kompilieren von Source Code

Die Kompilation erfolgt über Apache Maven durch Ausführen von `mvn compile`.

3.2. Bereitgestelltes Paket

Das bereitgestellte Paket ist in Kotlin umgesetzt und versteht sich als Basis, um dezentrale Zugriffskontrollen umzusetzen. Die dafür notwendigen Mechanismen, sowie der Unterbau in Form eines strukturierten P2P-Netzwerks sind vorhanden und einsatzfähig. Für einen Produktiveinsatz sollte auf ein etabliertes ABAC-Framework (XACML, …) zurückgegriffen werden.
Experimentell kann jedoch Easy-Rules in Kombination mit MVEL beibehalten werden. Für erweiterte Einsatzmöglichkeiten müssen in diesem Fall zumindest einerseits neue Policy-Templates erstellt werden (siehe Klasse `IdBasedPolicy` für ein Beispiel), sowie Security-Manager-Regeln definiert werden. Zusätzliche Templates sind notwendig, um Datenquellen und eventuelle Callbacks für etwaige Policies zu spezifizieren, damit auf diese im Rahmen von Policy-Definition und -Auswertung auf wohldefinierte Art zugegriffen werden kann. Security-Manager-Regeln sollten festgelegt werden, da ansonsten beliebige Operationen im Rahmen der Policy-Auswertung ausgeführt werden können, was ein erhebliches Sicherheitsrisiko darstellt.
Die eigentliche Programmlogik ist in den Klassen `PolNode` und in der Datei `DataContainer.kt` abgebildet. `PolNode` stellt eine Instanz im Peer-to-Peer-Netz dar, welche Entschlüsselungsanfragen an andere Instanzen stellen kann. Gleichzeitig beinhaltet diese Klasse auch die notwendige Programmlogik, um auf solche Anfragen zu reagieren und Policies auszuwerten. Die Datei `DataContainer.kt` beinhaltet allen notwendigen Code um Daten nach dem im Rahmen dieses Projets entwickelten Schema zu ver- und auf Basis einer zugehörigen Policy (und deren Auswertung) zu entschlüsseln und entsprechend zu speichern.

4. Inbetriebnahme

Der Demonstrator wird als Maven-Projekt bereitgestellt und beinhaltet einen JUnit-Test-Case zur Demonstration, welcher über Maven ausgeführt werden kann (s.u.).

4.1. Konfiguration

Das bereitgestellte Paket verfügt über keinerlei nicht-programmatische Konfigurationsparameter. Netzwerkknoten werden über eine Referenz auf eine Instanz der `Config`-Klasse gestartet (siehe `PolExample`, Zeile 24). Policies werden als YAML-Dateien definiert (siehe `PolExample`, Zeile 55ff).

4.2. Starten

Ein eigenständiger Start ist nicht vorgesehen, lediglich eine Demonstration im Rahmen eines JUnit-Test-Cases (s.u.). Dieser Test-Case konfiguriert und startet ein voll funktionsfähiges Peer-to-Peer-Netzwerk, innerhalb dessen eine dezentrale Durchsetzung von Zugriffs-Policies auf einen Datencontainer demonstriert wird.

4.3. Testen

Eine JUnit-Test-Suite liegt bei.

5. Handhabung

Tests, welche die Funktionalität veranschaulichen, können über Maven (`mv test`) ausgeführt werden.

6. Lizenz

Das bereitgestellte Projekt wird unter den Bedingungen der Open-Source-Lizenz für die Europäische Union (EUPL) V1.1 bereitgestellt (siehe Lizenz.pdf) und verwendet Bibliotheken der Stiftung SIC (siehe SIC_LICENSE.txt).