Podczas dyskusji ustaliliśmy:

1. postać komunikatu wejściowego dla autentykacji:

{

"requestId":"52ff9c2c-fb31-40f2-84ae-7fe2e417aeec",

"adURL":"ldaps://192.168.111.9:636",

"user":"labo\\kkr",

"password":"QAdWeFlvTlN6YiE=",

"parameters":["kkr","labo.local"]

}

gdzie:

* requestId – identyfikator żądania w dowolnej postaci tekstowej
* adURL – zawiera kompletny URL do podłączenia do Active Directory; z jego postaci wynika czy jest wykorzystywany SSL, IP i port połączenia. Zrezygnowano z przekazywania nazwy systemowej i jej mapowania na parametry połączenia pobierane z PostgreSQL ze względu na kwestie wydajnościowe i problemy synchronizacji przy cache’owaniu odczytywanych wartości
* user – identyfikator użytkownika w Active Directory; może mieć jedną z postaci:

"CN=Krzysztof Kraska,CN=Users,DC=labo,DC=local"

"[kkr@labo.local](mailto:kkr@labo.local)"

"labo\\kkr"

* password – hasło użytkowania do Active Directory w postaci BASE64
* parameters – tablica wartości przekazywanych parametrów, które mogą zostać wykorzystywane w pliku ‘application.properties’.

Np. w linii:

ENTRY\_FILTER=(&(objectClass=person)(objectClass=user)(sAMAccountName=**{0}**))

wyrażenie {0} zostanie zastąpione przekazanym parametrem o indeksie tablicy 0 przyjmując postać:

ENTRY\_FILTER=(&(objectClass=person)(objectClass=user)(sAMAccountName=kkr))

1. postać komunikatu wyjściowego dla autentykacji:

{

"requestId":"52ff9c2c-fb31-40f2-84ae-7fe2e417aeec",

"userGroups":["CN=SzkolenieECM,CN=Users,DC=labo,DC=local",

"CN=Domain Admins,CN=Users,DC=labo,DC=local"],

"authenticated":true

}

gdzie:

* requestId – identyfikator żądania z komunikatu wejściowego
* userGroups – lista grup, do których należy zautentykowany użytkownik dodatkowo zgodna z filtrem zdefiniowanym w pliku ‘application.properties’ w parametrze:

GROUP\_FILTER=

Filtr przyjmuje postać regex.

Grupy określane są na podstawie atrybutu ‘memberOf’ tego entry, za którego wyszukanie odpowiedzialna jest postać filtra wyszukiwania ldapscherch zdefiniowana w pliku ‘application.properties’ w parametrze:

ENTRY\_FILTER=(&(objectClass=person)(objectClass=user)(sAMAccountName={0}))

* authenticated – informacja o sukcesie autentykacji true/false

1. postać komunikatu generowanego w przypadku wystąpienia błędu:

{

"type":"event",

"origin":"Chris",

"uuid":"fc13945b-be7b-409a-8077-64835b83ece3",

"gateway":null,

"application":"AUTH",

"object":"ITGW-AUTH-2",

"severity":"WARNING",

"text":"Błąd ustanowienia połączenia do Active Directory",

"timestamptz":"2017-11-22 21:31:17.102",

"manager":"inTrackGateway"

}

gdzie:

* type – stała wartość ‘event'
* origin – wartość funkcji InetAddress.getLocalHost().getCanonicalHostName( ) zwracającej FQDN hosta, gdy jest to możliwe, bądź sam name, gdy nie ma takiej możliwości
* uuid: unikalny identyfikator instancji bramki
* gateway – stała wartość ‘null’
* application – wartość konfigurowana w pliku ‘application.properties’ w parametrze:

APPLICATION=

* object – wartość konfigurowana w pliku ‘application.properties’ w parametrze:

OBJECT=

* severity – stała wartość ‘WARNING’
* text – informacja z treści wyjątku wyrzuconego w przypadku błędu
* timestamptz – znacznik czasu z funkcji new Timestamp( System.currentTimeMillis() ).toString()
* manager – stała wartość ‘inTrackGateway’

1. Plik ‘application.properties’ ma postać:

MQTT\_BROKER=127.0.0.1:1883

MQTT\_TOPIC\_AUTH\_REQ=/auth/requests

MQTT\_TOPIC\_AUTH\_RES=/auth/responses

MQTT\_TOPIC\_AUTH\_ERR=/auth/errors

ENTRY\_FILTER=(&(objectClass=person)(objectClass=user)(sAMAccountName=**{0}**))

GROUP\_FILTER=.\*Admin.\*

TRUSTSTORE\_FILE=itgw.truststore

TRUSTSTORE\_PASS=ingrifo

APPLICATION=AUTH

OBJECT=ITGW-AUTH-2

gdzie:

* parametr MQTT\_TOPIC\_AUTH\_ERR definiujący nazwę kolejki zawierającej błędy jest dodatkowo rozwijany o część ‘/{UUID instancji bramki}’.
* parametr TRUSTSTORE\_FILE definiuje względną ścieżkę do pliku truststore
* parametr TRUSTSTORE\_PASS to jawne hasło do pliku truststore

Pozostałe parametry zostały omówione w punktach 1), 2) i 3).

1. UUID instancji bramki znajduje się w pliku ‘itgw.uuid’. Bramka sprawdza istnienie pliku i wczytuje z niego UUID. Bramka po wczytaniu nie weryfikuje poprawności postaci UUID. W przypadku braku pliku bramka sama generuje UUID i zapisuje w pliku ‘itgw.uuid’.
2. Certyfikaty zaufanych serwerów umieszczane są w Truststore. Truststore wykorzystuje domyślny format keystore z JDK.

Jeśli certyfikat serwera AD nie znajduje się w Truststore wówczas wyrzucany jest komunikat na kolejkę błędów a nadawcy zwracana jest odpowiedź o błędnej autentykacji.

Operacje na Truststore wykonywane są manualnie za pomocą keytool, np.:

keytool -v -import -file ingrifo.crt -alias ActiveDirectory \\

-keystore itgw.truststore -storepass ingrifo

keytool -v -delete -alias ActiveDirectory -keystore itgw.truststore \\

-storepass ingrifo

keytool -list -keystore itgw.truststore -storepass ingrifo