Keycloak における OAuth 2.0 ベースのセキュリティプロファイルの柔軟な実現方法 (第二報)

乗松 隆志[1,*]　中村 雄一[1]

概要：筆者らは，OSS である keycloak で OAuth 2.0 ベースのセキュリティプロファイルを実現する Client Policies を開発し, keycloak にコントリビューションした. これを利用し, 金融サービス向け API セキュリティ仕様である Financial-grade API Security Profile を実装し，動作検証を行った．本論文では，その動作検証の結果から，Client Policies の導入の効果として想定していた通り，セキュリティプロファイルの実装および変更があった場合の対応が容易であることを示す.

キーワード：OAuth 2.0，Keycloak，セキュリティプロファイル，FAPI

Flexible Way for Realizing OAuth 2.0 Based Security Profiles on Keycloak (part 2)

Takashi Norimatsu[1,*]　Yuichi Nakamura[1]

Abstract: We had developed and contributed the feature "Client Policies" for Keycloak to support OAuth 2.0 based security profiles flexibly. By using client policies, we have implemented Financial-grade API (FAPI) security profile to keycloak. In this paper, we show that we can implement new security profile or modify existing security profile easily by using client policies and prove that these security profiles pass FAPI conformance tests.

Keywords: OAuth 2.0, Keycloak, Security Profile, FAPI

1. はじめに

OAuth 2.0 という Web ベースの認可プロトコル[1]がある. 近年では，[2]に示されるように，REST API 連携でこの OAuth 2.0 が広く使われるようになっている.

OAuth 2.0 では，リソースオーナー，リソースオーナーが保有するリソースへのアクセスを API で提供するリソースサーバー，API をコールするクライアント，リソースオーナーから認可を取得し，それを示すデータであるアクセストークンを発行・管理する認可サーバーというエンティティが登場する.

1.1 OAuth 2.0 とセキュリティプロファイル

OAuth 2.0 はこれらのエンティティ間で認可情報をやり取りするフレームワークであり，様々な使い方が可能である．ただし，使い方を誤るとセキュリティホールが作りこまれることもあるため，OAuth2.0 の安全な使い方が求められるようになった．これはセキュリティプロファイルと呼ばれ[3]，様々は標準化団体がセキュリティプロファイルを策定している[4][5][6][7][8].

例えば，金融セクタなど高いセキュリティを要求されるセキュリティプロファイルの標準仕様としては， OpenID Foundation（OID-F）による Financial-grade API Security Profile 1.0 - Part 1: Baseline （FAPI 1.0 Baseline）[4]及び Part 2: Advanced（FAPI 1.0 Advanced） [5] ，英国 Open Banking Implementation Entity（OBIE）による Open Banking Security Profiles[6] ，オーストラリア Australian Competition & Consumer Commission（ACCC）による Consumer Data Right （CDR） Security Profile[7] ，ブラジルの Central Bank of Brazil による Open Banking Brasil Financial-grade API Security Profile 1.0[8]が挙げられる.

他の例として，ネイティブアプリや Single Page Application （SPA）で OAuth 2.0 を安全に使用するための標準仕様[9][10]がある.

これらのセキュリティプロファイルは, OAuth 2.0 に登場するエンティティに対して種々の要件を定めている．認可サーバーが公開するエンドポイントに対する要件，そのエンドポイントをどのようにクライアントやリソースサーバーが使うか，認可サーバーが発行したトークンをクライアント・リソースサーバがどう扱うかという要件等である.

セキュリティプロファイルは，策定された後に内容が改定される場合がある．そのような例として，仕様の脆弱性

1 株式会社 日立製作所
Hitachi, Ltd.
* takashi.norimatsu.ws@hitachi.com

© 2021 Information Processing Society of Japan

発見に伴う修正，仕様が確定する前段階（Implementer's Draft）での仕様のバージョンアップが挙げられる．

1.2 認可サーバーとしての OSS : Keycloak

セキュリティプロファイルに対応したシステムを構築する際，認可サーバーがその要件を満たすために必要な機能を有することが前提となっているため，認可サーバーの実装・設定が最も重要である．Identity Provider （IdP）や Identity and Access Management （IAM）用のソフトウェアが認可サーバーとしての機能を実装しており，プロプライエタリなものもあれば，オープンソース・ソフトウェア（OSS）であるものも存在する．後者の例として Keycloak[11]が挙げられる．Keycloak は Java で実装された Apache License 2.0 である OSS であり，無償で利用できることから世界中で広く利用されており[12]，商用利用向けに有償のサポートサービスも提供されている[13][14].

1.3 第一報からの進捗

筆者らは Keycloak にセキュリティプロファイルである FAPI 1.0 Baseline 及び Advanced の実装を検討し，従来の Keycloak でセキュリティプロファイルを実現するには運用・構築上様々な課題があることが分かった．そこで筆者らは，これらの課題を解決するために Client Policies という機構を Keycloak に実装し，その有効性の検証を第一報として報告した[15]．しかし，Client Policies で FAPI 1.0 Baseline 及び Advanced の実装を行うまでには至らなかった．

その後筆者らは Client Policies で FAPI 1.0 Baseline 及び Advanced を実装し，この実装を施した Keycloak がこれらセキュリティプロファイルの要件を満たしていることを，OID-F が提供している Conformance Test で確認した．そしれこれらの実装を Keycloak にコントリビューションし，マージされた[16].

本論文では，その結果から第一報で Client Policies の効果として想定していた通り，セキュリティプロファイルの実装及び変更があった場合の対応が容易であることを示す．

2. Client Policies

第一報で述べた Client Policies は，後に改良が加えられた．本章では，この改良後の Client Policies について述べる．

2.1 Client Policies の概念

Client Policies とは，クライアントに対する設定および処理用のテンプレートに類似したフレームワークである．個々のクライアントに対して設定及び処理をするのではなく，クライアントの集合を規定し，これに対して設定及び処理を行うものである．

Client Policies は，セキュリティプロファイルに関する処理を Keycloak 本体から切り離し，これを動的にロード・アンロードすることを可能とする．これにより，セキュリティプロファイルの新規実装及び既存のセキュリティプロファイルを変更時，Keycloak 本体のソースコードの改変が不要となる．

2.2 Client Policies の設計

図 1 に示されるように，Client Policies は，大きくは Framework と Components から構成される．Framework は，その上で動作する Components のロード・アンロードを管理すると共に，Components を実装するためのインタフェースであるフレームワークを提供する．

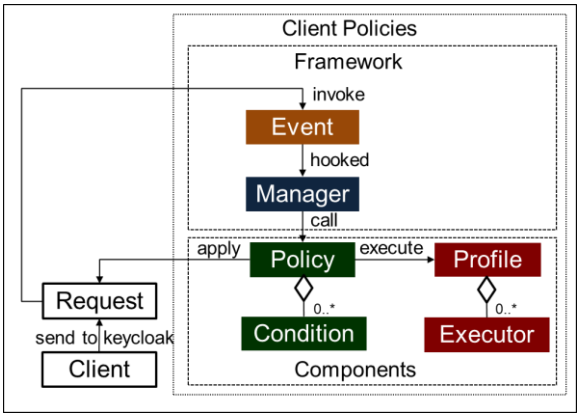

図 1　Client Policies のコンポーネント構成

Components は，Framework 上で動作するために規定されたインタフェースを実装したコンポーネント群であり，Profile，Executor，Policy 及び Condition が含まれる．

Profile は，セキュリティプロファイル全体の要件を満たすために必要な処理を実装したコンポーネントである．Profile は複数の Executor から構成される．Executor は，セキュリティプロファイルの個々の要件を満たすために必要な処理を実装したコンポーネントである．

Policy は，セキュリティプロファイルを適用するクライアント及びそのリクエストを定義するコンポーネントである．Policy は複数の Condition から構成される．Condition は，あるクライアント及びそのリクエストがセキュリティプロファイルの適用対象であるかどうか評価する処理を実装したコンポーネントである．Policy は複数の Profile への参照を保持し，全ての Condition で適用対象と判断されたクライアント及びそのリクエストに対して Profile を実行する．

Policy により，セキュリティプロファイルに関する処理を実装した Profile を適用するクライアント及びそのリクエストの集合を定義できる．そのため，クライアント一つ一つに対し，セキュリティプロファイルを適用するための設定を行う必要がない．

Policy は，クライアントからのリクエストを受け付ける箇所（図 2 の Endpoint）で，クライアントの設定とリクエストを元にして Condition を評価し，Profile を適用するか

どうか決定する. 適用する場合, Profile が包含する Executor を全て実行する．これにより，特定の条件を満たすクライアントとそのリクエストに対してのみ，特定の処理を実施することが可能となる．よって，同じクライアントのリクエスト毎に異なるセキュリティプロファイルを適用することも可能である．

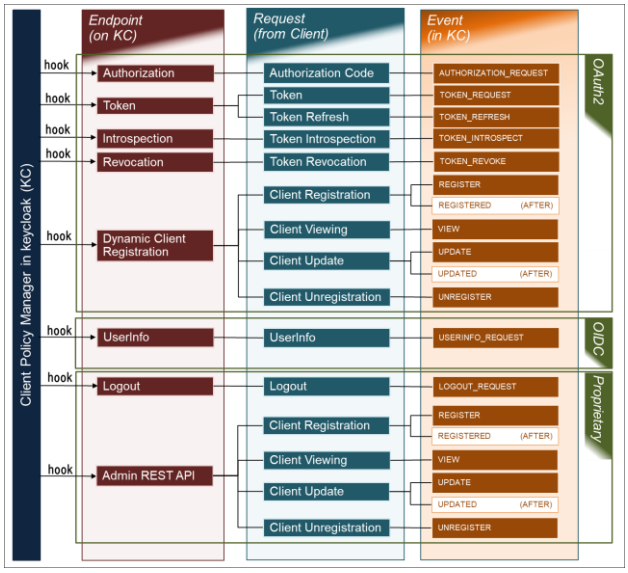

図 2　Client Policies の実行ポイント

2.3 Client Policies の実装

実装面からみると，図 3 で示されるように，Policy や Condition や Executor は，Client Policies の Framework が規定したインタフェースを実装したプラグイン形式のソフトウェアコンポーネントである．これを Java ではプロバイダと呼ぶ. これらプロバイダのビルドは Keycloak 自体のビルドと独立している．また，プロバイダは Keycloak の稼働中に動的にロード・アンロードが可能である．

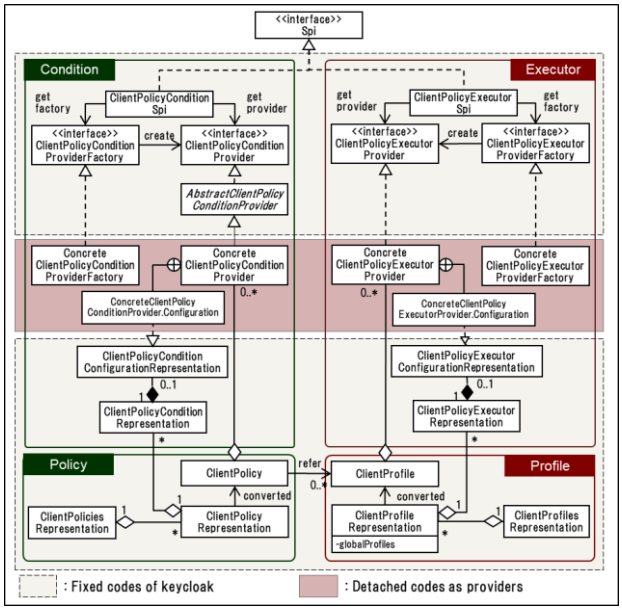

図 3　Client Policies のクラス構成

具体的なセキュリティプロファイルの実現のための Components である Condition 及び Executor と，これら Components の処理を行う Framework のコードは分離されている．そのため，セキュリティプロファイルのバージョンアップや，新しいセキュリティプロファイルをサポートする場合，Executor をプロバイダとしてこれらを実装すればよいので, Keycloak 本体のコードを修正する必要がない．

セキュリティプロファイルのバージョンアップや，新しいセキュリティプロファイルをサポートする場合，これに対応する Condition や Executor を実装して Keycloak で稼働させる必要がある．これらはプロバイダとして実装されるため，Keycloak の稼働中にロードすることが可能である．よって，Keycloak を停止させることなくセキュリティプロファイルのバージョンアップや，新しいセキュリティプロファイルに対応できる．

2.3.1 Condition

Condition は，クライアントの性質及びクライアントの陸セスの内容を基に Profile を適用するかどうか評価する．

評価結果は Yes，No，Abstain の 3 値で表現される．Yes は Profile を適用することを意味する．No は適用しないことを意味する．Abstain は判定自体しないことを意味する．

Condition は，評価結果である Yes と No を反転させる機能（Negative Logic）を持っている．これにより Condition で Profile の適用判定を受けるクライアントのリクエストの補集合に対し適用の判定を出すことができる．

実際に Keycloak に実装した Condition を以下に記す．

- クライアントの性質に基づく Condition
  - (1) 全てのクライアント及びそのクライアントからのリクエスト．
  - (2) クライアントの種別（OAuth 2.0 における Client Type）が特定の種別であった場合．
  - (3) クライアントに適用されているロールが特定のロールであった場合．
- クライアントのリクエストの性質に基づく Condition
  - (4) クライアントのリクエストに含まれるパラメータであるスコープ（OAuth 2.0 における scope）が特定の値を含んでいた場合．
- クライアント登録・更新の方法および実行者の性質に基づく Condition
  - (5) クライアント登録・更新のリクエストの方式が特定の方式であった場合．
  - (6) クライアント登録・更新のリクエストの実行者が所属するグループが特定のグループであった場合．
  - (7) クライアント登録・更新のリクエストの送信元ホスト名が特定のホスト名であった場合．

2.3.2 Executor

FAPI 1.0 Baseline 及び Advanced の要求事項を満たすための処理を実装した．その具体的な内容はそれぞれ 3.1 及び 3.2 節で述べる．

クライアントにセキュリティの観点から不用意な設定がされるのを防ぐために，自動的に妥当な設定を行う機能（Auto Configuration）を設けた．これにより，特定のセキュリティプロファイルの要求事項を満たすような設定をクライアントに自動的に実行することができる．

3. FAPI Security Profile の要求事項を満たす Executor の実装

2 章で述べた Client Policies により FAPI Security Profile の要求事項を満たす Executor を実装した．

3.1 FAPI 1.0 Baseline Security Profile の要求事項を満たす Executor の実装

FAPI 1.0 Baseline の要求事項は，既に Keycloak が対応できるものとできないものとに分かれる．この後者の要件を Executor として実装した．そして，これら Executor を含む Profile を作成した．この Profile により，Policy で規定されるクライアントのリクエストに対し FAPI 1.0 Baseline を適用することができる．

FAPI 1.0 Baseline の要件と対応する Executor を以下に記す．

(1) クライアントの認証方式の限定

クライアントの認証方式として，RFC 8705 OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens（MTLS）[17]の 2 節で規定される Mutual TLS for OAuth Client Authentication，OpenID Connect Core 1.0（OIDC）[18]の 9 節で規定される private_key_jwt 或いは client_secret_jwt のどれかの方法のみ許容する．

これに対応する要件は FAPI 1.0 Baseline の 5.2.2.節 4-1 及び 2 項で規定されている．

本 Executor は，クライアントが明示的に自身の認証方式を設定していない場合 Auto Configuration により private_key_jwt を自動的に設定する．

(2) Proof Key for Code Exchange（PKCE）の適用

認可要求及びトークン要求時に，code_challenge_method を S256 とした RFC 7636 Proof Key for Code Exchange by OAuth Public Clients（PKCE）[19]を実行する．

これに対応する要件は FAPI 1.0 Baseline の 5.2.2.節 7 項で規定されている．

本 Executor は，クライアントが明示的に code_challenge_method を設定していない場合 Auto Configuration により S256 を自動的に設定する．

(3) OAuth 2.0 の redirect_uri パラメータの安全な使用

クライアントが予め登録する redirect_uri パラメータの値の scheme 部は https であり，かつクライアントは認可要求のリクエストを認可サーバーに送信する時に付与する redirect_uri パラメータの値と予め登録されたものとが完全一致で比較し一致する場合のみリクエストを受け付ける．

これに対応する要件は FAPI 1.0 Baseline の 5.2.2.節 8,9,10 および 20 項で規定されている．

(4) リソースオーナーからの同意の取得

クライアントは，リソースオーナーのリソースに対するアクセスを示す OAuth 2.0 の scope パラメータを認可要求のリクエストに含め，認可サーバーに送信する．認可サーバーは，クライアントによる scope パラメータで示されるリソースアクセスについてリソースオーナーから同意をとる必要がある．

これに対応する要件は FAPI 1.0 Baseline の 5.2.2.節 12 項で規定されている．

本 Executor は，scope パラメータで示されるリソースアクセスの同意がリソースオーナーからとられていない場合，リソースオーナーに対し，同意するかどうかを確認する画面を必ず出す．

(5) トークン不正取得を検知可能な仕組みの適用

OAuth 2.0 は認可要求とトークン要求の 2 つのパートに分かれており，それぞれで独立した HTTP リクエスト・レスポンスが交わされる．これらを合わせて一つの論理的なセッションとみなすための仕組みが必要である．

OAuth 2.0 では認可コード（code）というパラメータが認可要求とトークン要求とを関連付けている．然しながら，これのみでは悪意のある第三者が他者のトークンを取得，又は他者に自身のトークンを掴ませる可能性が生じる．

OAuth 2.0 の scope パラメータ，或いは OIDC の nonce パラメータを正しく用いることで，これを検知することが可能となる．本 Executor は，クライアントから送信される認可要求のリクエストに，scope パラメータか nonce パラメータが含まれていない場合，エラー応答する．

これに対応する要件は FAPI 1.0 Baseline の 5.2.2.2 節 1 項および 5.2.2.3 節の第 1 項で規定されている．

3.2 FAPI 1.0 Advanced Security Profile の要求事項を満たす Executor の実装

基本的に FAPI 1.0 Advanced は FAPI 1.0 Baseline の要求事項に加え，追加の要求事項を規定している．だが，例外として FAPI 1.0 Baseline の要求事項の内一部の除外と変更とがある．

FAPI 1.0 Baseline の要求事項の内除外されるものは以下となる．

(1) PKCE の適用除外

FAPI 1.0 Advanced では，クライアント種別を Confidential に限定したことと，クライアント認証方式をより安全なも

のに限定したことから，悪意のある第三者によるクライアントのなりすましが困難となったためである．

本件は FAPI 1.0 Advanced の 5.2.2.節の序文で記されている．

FAPI 1.0 Baseline の要求事項の内変更されるものは以下となる．

(2)　クライアントの認証方式の限定

MTLS の 2 節で規定される Mutual TLS for OAuth Client Authentication と, OIDC の 9 節で規定される private_key_jwt の 2 つのみ許容する．

本件は FAPI 1.0 Advanced の 5.2.2.節 14 項で規定されている．

FAPI 1.0 Advanced で新たに追加されるものは以下となる．

(3)　クライアントの種別の限定

クライアント種別が Confidential なクライアントからの要求のみを受け付ける．

これに対応する要件は FAPI 1.0 Advanced の 5.1 節で規定されている．

(4)　Sender Constrained なアクセストークンの適用

認可サーバーからアクセストークンを受領したクライアントのみが，そのアクセストークンを行使しリソースオーナーのリソースにアクセスできるようにする（Sender Constrained）．また，その具体的な実現方法として，MTLS の 3 節で規定の Mutual-TLS Client Certificate-Bound Access Tokens を適用する．

これに対応する要件は FAPI 1.0 Advanced の 5.2.2.節 8,9,10 及び 20 項で規定されている．

本 Executor は，クライアントが明示的に Sender Constrained なアクセストークン適用の設定していない場合 Auto Configuration により自動的に設定する．

(5)　Request Object の適用と安全な利用

認可要求のパラメータをクライアントから認可サーバーに安全に届けるために，OIDC の 6 節で規定される Request Object を適用し，かつこれを安全に利用する．

これに対応する要件は FAPI 1.0 Advanced の 5.2.2.節 1,10,13,15 及び 17 項で規定されている．

(6)　認可応答時の Response Type の限定

認可応答に何を返すのかを規定する OAuth 2.0 の response_type パラメータの値として，code と id_token のみ許容する．

これに対応する要件は FAPI 1.0 Advanced の 5.2.2.節 2 項で規定されている．

本 Executor は，認可応答で返される ID トークンが，認証済みのユーザーの情報を格納する本来の用途ではなく，認可応答で返される認可コードや state パラメータの完全性をチェックするための用途となるよう Auto Configuration により設定が自動的に行われる．

(7)　署名アルゴリズムの限定

JSON Web Token（JWT）[20]に対する署名及び署名検証を規定する JSON Web Signature（JWS）[21]において，適用する署名アルゴリズムを JSON Web Algorithms（JWA）[22]で規定される ES256 と PS256 のみ許容する．

OIDC では，JWT のデータ構造をとり署名として JWS が適用されるデータとして，UserInfo，ID トークン，Request Object, private_key_jwt のクライアント認証方式での認証用クレデンシャル（Client Assertion）がある．また，keycloak ではアクセストークンおよびリフレッシュトークンも JWT のデータ構造をとっているため，これらも JWS による署名が適用される．これら全てに対して署名アルゴリズムを ES256 と RS256 に限定する．

これに対応する要件は FAPI 1.0 Advanced の 8.6 節で規定されている．

4. 検証結果

2 節で述べた Client Policies と, 3 節で述べた FAPI Security Profile の要求事項を満たす Executor を利用し，新規セキュリティプロファイルのサポートと，サポート済みの既存セキュリティプロファイルの変更に，柔軟に対応できることを示すことを示す．

FAPI の仕様は Working Draft，Implementer’s Draft，Final の 3 つの段階が存在し，前者 2 つにはバージョンが付与されている．Final は仕様が確定された段階である．それ以前の段階では，仕様の内容の変更があり得る．

Working Draft 及び Implementer’s Draft の段階で FAPI Read Only API Security Profile 及び Read and Write API Security Profile として策定されていた仕様は，Final の段階になる際にそれぞれ FAPI 1.0 Security Profile 及び FAPI 1.0 Advanced Security Profile に改名された．

新規にサポートするセキュリティプロファイルとして，FAPI Read and Write API Security Profile Implementer’s Draft version 2 （FAPI-RW-ID2）[23]を取り上げる．そして，サポート済みの既存セキュリティプロファイルの変更として，FAPI-RW-ID2 を基に Final の段階となった FAPI 1.0 Advanced [5]を取り上げる．

4.1 検証の方法

OID-F が，FAPI 1.0 Advanced 及び関連仕様に製品が適合していることを示す認定制度を設けている[24]．認定を受けるには OID-F が提供しているテスト実行環境 Conformance Suite[25]で，テストにパスする必要がある．この Conformance Suite は MIT License の OSS として公開されている[26].

双方のセキュリティプロファイルについて，これを実装した Keycloak が実際に仕様に合致していることを検証するためために，この Conformance Suite を利用しテストにパ

スすることを示す.

4.2 新規セキュリティプロファイルのサポート

FAPI-RW-ID2 を Client Policies の Profile として規定した. 加えてこの Profile を適用するクライアントを Policy で規定した.

Profile 及び Policy は JSON で表現される. 図 4 に FAPI-RW-ID2 を規定する Profile である「fapi-profile」を示す. 本 Profile が含む Executor が 3.1 節及び 3.2 節のどの Executor に対応しているかを図 4 の右部に記している.

```
{"name": “fapi-profile",
 "description": “The profile for FAPI security profile",
 "executors": [
  {"executor": "secure-session",                          ← 3.1 (5)
   "configuration": {}},
  {"executor": "confidential-client",                     ← 3.2 (3)
   "configuration": {}},
  {"executor": "secure-client-authenticator",             ← 3.2 (2)
   "configuration": {
     "allowed-client-authenticators": [ "client-jwt", "client-x509"],
     "default-client-authenticator": "client-jwt"}},
  {"executor": "secure-client-uris",                      ← 3.1 (3)
   "configuration": {}},
  {"executor": "secure-request-object",                   ← 3.2 (5)
   "configuration": {
     "available-period": "3600",
     "verify-nbf": false}},
  {"executor": "secure-response-type",                    ← 3.2 (6)
   "configuration": {
     "auto-configure": true,
     "allow-token-response-type": false}},
  {"executor": "secure-signature-algorithm",              ← 3.2 (7)
   "configuration": {
     "default-algorithm": "PS256"}},
  {"executor": "secure-signature-algorithm-signed-jwt",
   "configuration": {
     "require-client-assertion": false}},
  {"executor": "consent-required",                        ← 3.1 (4)
   "configuration": {}},
  {"executor": "holder-of-key-enforcer",                  ← 3.2 (4)
   "configuration": {
     "auto-configure": true}}
 ]}
```

図 4 Profile 「fapi-profile」

この Profile を適用するクライアントを規定する Policy である「fapi-policy」を図 y に示す.

```
{"name" : "fapi-policy",
"description" : "The policy for FAPI security profile",
"enabled" : true,
"conditions" : [{                                         ← 2.3.1 (3)
   "condition" : "client-roles",
   "configuration" : {"roles" : [ "sample-client-role" ]}}],
"profiles" : [ "fapi-profile" ]}
```

図 5 Policy 「fapi-policy」

図 5 で示される Policy「fapi-policy」は，ロール「sample-client-role」が割り当てられたクライアントに Profile「fapi-profile」を適用するようにする. よって FAPI-RW-ID2 を適用したいクライアントにこのロールを付与するのみで済む.

4.2.1 Client Policies 導入の効果

第一報の 2.2 節にある通り Client Policies 導入以前の Keycloak の場合，クライアントにセキュリティプロファイルを適用する処理が行われるように，クライアント毎に多数の設定を行う必要があった. Client Policies では図 5 にある Policy の作成と，FAPI-RW-ID2 を適用したいクライアントにロールを付与するのみで済む.

また，Client Policies 以前の Keycloak の場合，FAPI-RW-ID2 の適用対象となっているクライアントを適用対象から外したい場合は，そのクライアントの設定を不正な動作を行わないよう正しく変更する必要があった. これに対して Client Policies では FAPI-RW-ID2 の適用対象となっているクライアントを適用対象から外したい場合は，そのクライアントからロールを外すのみで済む.

さらに，2.3.1 節(4)で記されているように，Client Policies の Policy はクライアントの個々のリクエストの内容から，Profile を適用するかどうかが決定できる. Client Policies 以前の Keycloak ではこれは実現できず，第一報の 2.3 節にあるようにレルムを複数作成し同じクライアントを登録し設定を管理する必要があった. Client Policies を導入することで，その必要が無くなった.

以上より，Client Policies の導入することで，柔軟にセキュリティプロファイルの新規適用と運用を行えるようになった.

4.2.2 仕様との適合性検証

次に Profile「fapi-profile」により Keycloak が FAPI-RW-ID2 の仕様に適合していることを示す.

筆者らが参画している OSS コミュニティ活動である Financial-grade API Security : Special Interest Group（FAPI-SIG）[27]が，OID-F の Conformance Suite の自動実行環境を提供している. これを利用し，テストにパスすることを確認した. Linux 系の OS のマシンを利用している場合の手順を図 6 に示す.

```
$ git clone https://github.com/keycloak/kc-sig-fapi.git
$ cd kc-sig-fapi
$ git checkout -b  fapi-rw-id2 refs/tags/fapi-rw-id2
$ docker-compose -p keycloak-fapi up –build
```

図 6 FAPI-RW-ID2 用 Conformance Suite の自動実行

テスト結果は，Conformance Suite のサーバーの Web ページ（https://localhost:8443/plans.html）とテキストファイルに出力される. 前者をテスト結果のサマリーページを図 7 に示す.

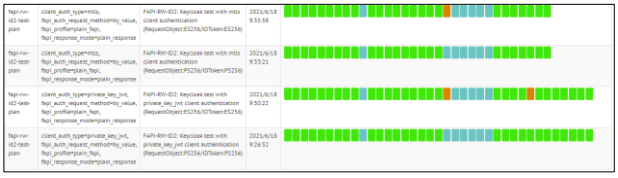

図 7 FAPI-RW-ID2 用 Conformance Suite の結果サマリ

このテスト結果を精査した結果，全てのテスト項目にパスしたことを確認した．

各テスト項目の詳細な結果を同 Web ページから確認できる．個々のテストの結果（result）について，グリーン（PASSED）となっているものは，テストにパスしたことを示している．

水色（REVIEW）となっているものは，そのテスト項目の性質上テストにパスしたことを自動的に判定できないことを示している．ここではリソースオーナーのブラウザに対してエラー応答を返すテスト項目がこれに該当する．仕様では人間を相手にした場合のエラー応答について，伝えるべき内容は規定しているが表現までは規定しておらず，その表現は対象製品で千差万別であるため Conformance Suite のプログラムでは自動的に判定できない．OID-F から認定を受ける場合は，テストのログとブラウザに対して表示した画面のスナップショットを OID-F に送付する必要がある．本論文は OID-F から認定を受けるまでは対象外であるため，テスト結果が REVIEW となっているものもテストにパスしたと見做した．

オレンジ（SKIPPED）となっているものは，指定された署名アルゴリズムにより意図的にパスされたもので，テストの判定に影響がない事を確認した．

4.3 既存セキュリティプロファイルの変更

4.2 節の Profile を実装した Keycloak が，FAPI-RW-ID2 の改定版である FAPI 1.0 Advanced をサポートするために，Profile 「fapi-profile」を FAPI-RW-ID2 の改定版である FAPI 1.0 Advanced へ変更する．

FAPI 1.0 Advanced では，3.2 節(5)で述べた認可要求時の Request Object の適用と安全な利用に関し，JWT の nbf クレームを含み，その値は Request Object の生成時点から 60 分以内であることが新しく追加された．この変更に対応するため，既存の Profile 「fapi-profile」を図 8 の様に変更した．

```
{"executor": "secure-request-object",        ← 3.2 (5)
 "configuration": {
  "available-period": "3600",
  "verify-nbf": true}},
```

図 8 Profile 「fapi-profile」の変更

Client Policies 以前の Keycloak の場合，クライアントに変更後のセキュリティプロファイルを適用するための処理が行われるよう，クライアント毎に設定変更を行う必要があった．Client Policies でその必要が無くなった．

4.3.1 Client Policies 導入の効果

第一報の 2.2 節にある通り Client Policies 以前の Keycloak の場合，セキュリティプロファイルを変更する場合，そのセキュリティプロファイルを適用している全てのクライアントの設定を正しく変更する必要があった．Client Policies では図 8 にある Profile の変更のみで済む．

セキュリティプロファイルの変更の内容によっては，既存の Keycloak ではサポートできない場合も考えられる．Client Policies 導入以前の Keycloak では，第一報の 2.1 節にある通り，Keycloak 本体のソースコードの改変と，それに伴うクライアントの設定の追加が必要となる場合があった．Client Policies を導入することで，Keycloak 本体ではなく，プロバイダという形式で Executor を実装し，Profile にその設定を追加するのみで済む．

以上より，Client Policies の導入することで，柔軟に既存セキュリティプロファイルの変更と運用を行えるようになった．

4.3.2 仕様との適合性検証

4.2 節と同様，FAPI-SIG の Conformance Suite の自動実行環境を利用し，テストにパスすることを確認した．Linux 系の OS のマシンを利用している場合の手順を図 9 に示す．

```
$ git clone https://github.com/keycloak/kc-sig-fapi.git
$ cd kc-sig-fapi
$ git checkout -b fapi-1-advanced-final refs/tags/fapi-1-advanced-final
$ docker-compose -p keycloak-fapi up –build
```

図 9 FAPI 1.0 Advanced 用 Conformance Suite の自動実行

Conformance Suite のサーバーの Web ページ（https://localhost:8443/plans.html）とテキストファイルに出力される．前者をテスト結果のサマリーページを図 10 に示す．

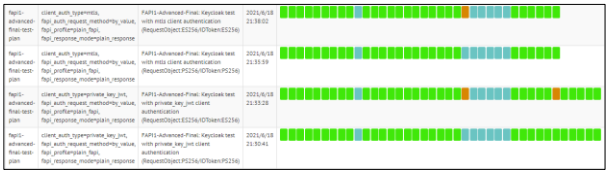

図 10 FAPI 1.0 Advanced 用 Conformance Suite 結果サマリ

このテスト結果を精査した結果，全てのテスト項目にパスしたことを確認した．

5. おわりに

本論文の第一報にて，Keycloak に OAuth 2.0 ベースのセキュリティプロファイルを適用するためにはコード本体の修正が必要な場合があり，またクライアントの設定が煩雑という課題を述べた．同じく第一報[15]で，これら課題を解決するために Client Policies というコンセプトを導入しこれを実装した．これらを Keycloak にコントリビューションし，マージされ，Keycloak 12 から利用可能となった[28]．これにより，Keycloak 本体の修正が不要となり，設定も簡易化されていることを確認した．

この Client Policies を利用し，新しく FAPI-RW-ID2[23]を Client Policies の Profile として実装した．この方法は Client

Policies を実装する以前の Keycloak で同様の設定をクライアントに施すのに比べて柔軟な方式であることを確認した．

次いで Conformance Suite のテストを実行し，全てのテスト項目をパスすることを確認した．

更に，このセキュリティプロファイルを変更し FAPI 1.0 Advanced に対応, Client Policies を実装する以前の Keycloak で同様の設定をクライアントに施すのに比べて柔軟な方式であることを確認した．

同じく Conformance Suite のテストを実行し，全てのテスト項目をパスすることを確認した．

実装した FAPI 1.0 Baseline 及び FAPI 1.0 Advanced のセキュリティプロファイルを Keycloak でサポートするために必要な Profile，Executor，Policy 及び Condition を Keycloak にコントリビューションし，マージされ，keycloak 14 から利用可能となった．

今後は，この Client Policies をベースとして FAPI 1.0 Baseline 及び FAPI 1.0 Advanced とは性質の異なるセキュリティプロファイルの実装及び Keycloak へのコントリビューションを計画している．

謝辞 Keycloak の Project Lead である Stian Thorgersen 氏からは，Client Policies のコンセプト，その要件や設計議論で数多くのアイデアを頂きました．また Client Policies の実装の改良には Keycloak Development Team のメンバーである Marek Posolda 氏や，FAPI-SIG をはじめ多くの Keycloak コミュニティのコントリビューターの方々に参画いただきました．ここに感謝を述べます．

参考文献

[1] “RFC 6749 The OAuth 2.0 Authorization Framework”. https://datatracker.ietf.org/doc/html/rfc6749, (参照 2021-07-06).
[2] セキュアなシステム間連携を支える OSS 認証認可技術．日立評論 2020 vol.102 No.3. https://www.hitachihyoron.com/jp/archive/2020s/2020/03/03a04/index.html, (参照 2021-07-06)
[3] “An Extensive Formal Security Analysis of the OpenID Financial-Grade API”. https://ieeexplore.ieee.org/document/8835218, (参照 2021-07-06).
[4] “Financial-grade API Security Profile 1.0 - Part 1: Baseline”. https://openid.net/specs/openid-financial-api-part-1-1_0-final.html, (参照 2021-07-06).
[5] “Financial-grade API Security Profile 1.0 - Part 2: Advanced”. https://openid.net/specs/openid-financial-api-part-2-1_0.html, (参照 2021-07-06).
[6] “OpenBanking Security Profiles”. https://standards.openbanking.org.uk/security-profiles/, (参照 2021-07-06).
[7] “Consumer Data Right Security Profile”. https://consumerdatastandardsaustralia.github.io/standards/#security-profile, (参照 2021-07-06).
[8] “Open Banking Brasil Financial-grade API Security Profile 1.0”. https://openbanking-brasil.github.io/specs-seguranca/open-banking-brasil-financial-api-1_ID1.html, (参照 2021-07-06).
[9] “RFC 8252 OAuth 2.0 for Native App”. https://datatracker.ietf.org/doc/html/rfc8252, (参照 2021-07-06).
[10] “OAuth 2.0 for Browser-Based Apps”. https://datatracker.ietf.org/doc/html/draft-ietf-oauth-browser-based-apps-08, (参照 2021-07-06).
[11] “Keycloak”. https://www.keycloak.org/, (参照 2021-07-06).
[12] “Keycloak proposal for submission to CNCF”. https://github.com/cncf/toc/pull/463, (参照 2021-07-06).
[13] “Red Hat Single Sign On”. https://access.redhat.com/products/red-hat-single-sign-on, (参照 2021-07-06).
[14] “OpenStandia”. https://www.nri.com/jp/news/info/cc/lst/2018/0119, (参照 2021-07-06).
[15] Keycloak における OAuth 2.0 ベースのセキュリティプロファイルの柔軟な実現方法. Computer Security Symposium 2020. 2020 年 10 月 28 日.
[16] Keycloak release notes - Keycloak 14.0.0 - Client Policies and Financial-grade API (FAPI) Support. https://www.keycloak.org/docs/latest/release_notes/index.html#keycloak-14-0-0, (参照 2021-07-06).
[17] “RFC 8705 OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens”. https://datatracker.ietf.org/doc/html/rfc8705, (参照 2021-07-06).
[18] “OpenID Connect Core 1.0”. https://openid.net/specs/openid-connect-core-1_0.html, (参照 2021-07-06)
[19] “RFC 7636 Proof Key for Code Exchange by OAuth Public Clients”. https://datatracker.ietf.org/doc/html/rfc7636, (参照 2021-07-06).
[20] “RFC 7519 JSON Web Token (JWT)” . https://datatracker.ietf.org/doc/html/rfc7519, (参照 2021-07-06).
[21] “RFC 7515 JSON Web Signature (JWS)” . https://datatracker.ietf.org/doc/html/rfc7515, (参照 2021-07-06).
[22] “RFC 7518 JSON Web Algorithm (JWA) ”. https://datatracker.ietf.org/doc/html/rfc7518, (参照 2021-07-06).
[23] “Financial-grade API - Part 2: Read and Write API Security Profile”.https://openid.net/specs/openid-financial-api-part-2-ID2.html, (参照 2021-07-06).
[24] “OpenID Certification”. https://openid.net/certification/, (参照 2021-07-06).
[25] “OpenID Foundation Conformance Suite”. https://www.certification.openid.net/login.html, (参照 2021-07-06).
[26] “conformance-suite”. https://gitlab.com/openid/conformance-suite, (参照 2021-07-06).
[27] “FAPI-SIG (Financial-grade API Security : Special Interest Group) ”. https://github.com/keycloak/kc-sig-fapi, (参照 2021-07-06)
[28] Keycloak release notes - Keycloak 12.0.0 - FAPI RW support and initial support to Client policies. https://www.keycloak.org/docs/latest/release_notes/index.html#keycloak-12-0-0, (参照 2021-07-06).