# Developing the Flexible Conformance Test Execution Platform for OAuth 2.0-based Security Profiles

Takashi Norimatsu[1,2,a)] Yuichi Nakamura[1] Toshihiro Yamauchi[3]

Received: February 28, 2024, Accepted: August 27, 2024

***Abstract:*** Developers of OAuth 2.0's authorization server or OpenID Connect 1.0's OpenID provider software that support multiple OAuth 2.0-based security profiles need their products to pass conformance tests provided by the OpenID Foundation. However, they usually encounter several challenges. Specifically, they require extensive man-hours to create programs other than the product targeted for the conformance tests, provide support for execution of a new conformance test if required by a new security profile, and execute multiple conformance tests. Together with the Open-source Software community OAuth Special Interest Group, we developed a conformance test execution platform to resolve these issues, using Keycloak as the target for conformance tests. We evaluated the platform and confirmed that it resolves these issues. Using the platform, we executed conformance tests of the Financial-grade API (FAPI) and Open Banking security profiles to Keycloak and confirmed that Keycloak passed the conformance tests of these security profiles. This implies that Keycloak complies with their specifications. We confirmed by the evaluation of the platform that automating execution of a conformance test reduced its completion time by 56.8%, parallelizing execution of nine conformance tests reduced its completion time by 62.4% and lines of code of programs the developer needs to write was reduced by 85.7% by the platform. Finally, we published the platform on the GitHub repository for public use.

***Keywords:*** conformance test, security profile, OAuth 2.0, Financial-grade API (FAPI), Keycloak

## 1. Introduction

The Financial-grade API (FAPI) security profile [1] has been developed by the OpenID Foundation (OIDF) as a security specification for accessing Application Programming Interfaces (APIs) that require high security, such as ones providing financial services. The FAPI security profile is based on OAuth 2.0 [2] --- an authorization protocol, and OpenID Connect 1.0 (OIDC) [3] --- an authentication protocol based on OAuth 2.0.

Some of the security profiles of Open Banking, which is an ecosystem that supports APIs and provides financial services, are based on FAPI security profile (**Table 1**).

The OIDF provides a Conformance Suite [11] as open-source software (OSS), which performs conformance tests of the FAPI security profiles and security profiles of open banking, as shown in Table 1. In addition, the OIDF certifies the software's compliance with the security profile [12]. An OAuth 2.0's authorization server or OpenID Connect 1.0's OpenID Provider (OP) software product, which supports the security profile, can be certified by the OIDF if the product passes a conformance test of the security profile provided by the Conformance Suite and sends the result of the conformance test to the OIDF.

An authorization server or OP software product certified by the OIDF has business benefits. For example, Open Finance Brazil only allows authorization servers or OP software products certified by the OIDF to connect to its ecosystem. Generally, customers who want to procure an authorization server or OP that supports security profiles tend to prefer products certified by the OIDF as compliant with the security profiles.

If the conformance tests are executed in parallel as a part of existing continuous integration (CI) of an authorization server or OP, the conformance tests should be executed automatically and the completion time of the conformance tests should be almost the same or shorter than the existing CI. If the completion time of the conformance tests is longer than the one of the existing

**Table 1** Open Banking security profiles.

| Open Banking | Basis of security profile |
|---|---|
| UK Open Banking [4] | FAPI Read and Write [5] |
| Australia Consumer Data Right [6] | FAPI 1.0 Advanced [7] |
| Open Finance Brazil [8] | FAPI 1.0 Baseline [9], FAPI 1.0 Advanced |
| Saudi Arabia KSA Open Banking [10] | FAPI 1.0 Advanced |

[1] Hitachi, Ltd., Chiyoda, Tokyo 100–8280, Japan
[2] Graduate School of Natural Science and Technology, Okayama University, Okayama 700–8530, Japan
[3] Faculty of Environmental, Life, Natural Science and Technology, Okayama University, Okayama 700–8530, Japan
[a)] takashi.norimatsu.ws@hitachi.com

CI, adding the conformance tests to the CI increases the completion time of CI. Increasing the completion time of CI stalls the speed of development of the authorization server or OP applying the CI, which is a problem.

The developers of authorization servers or OP that support multiple FAPI and Open Banking security profiles as shown in Table 1, face three issues during conformance testing. These are related to the increase in man-hours for software development and maintenance;

( 1 ) Automation: Preparing programs other than the authorization server or OP being tested for automating execution of conformance tests
( 2 ) Cost Reduction of New Conformance Test Execution: Supporting conformance test execution of a new security profile
( 3 ) Parallelization: Automating execution of multiple conformance tests in parallel

We assumed Keycloak[*1] as a target for the conformance tests and developed a conformance test execution platform that includes a Conformance Suite to resolve these issues. The authors contributed implementation of several security profiles to Keycloak [13]-[16]. We executed conformance tests of the security profiles. Consequently, we confirmed that Keycloak passes the conformance tests, and thus complies with the specifications of the security profiles.

Members of the Keycloak community other than the authors received certification from the OIDF [12], indicating that Keycloak complies with the standard specifications and security profiles based on the conformance test results obtained using our conformance test execution platform. Keycloak is an OSS; therefore, anyone can use it as an OIDF-certified authorization server/OP.

Using this conformance test execution platform, we executed conformance tests against Keycloak as part of regression testing whenever a new version of Keycloak is released, which ensures that the new version remains compliant with supported security profiles. The test results are published on the Keycloak's OAuth Special Interest Group (SIG) website[*2] for easy accessibility.

The conformance test execution platform was also published on GitHub[*3] as a sub-project of Keycloak for easy accessibility.

The contributions of this study are as follows:

- Design and development of conformance test execution platform. We assumed Keycloak as the target of conformance tests. Hence, we designed and developed a conformance test execution platform that resolves three issues faced by software developers of an authorization server or OP that supports multiple standard specifications and the security profiles defined by OIDF with the OSS community OAuth SIG.
- Contribution to Keycloak's acquisition of certification. Based on the test results obtained using the developed conformance test execution platform, members of the Keycloak community received certification from the OIDF that Keycloak complied with the standard specifications and security profiles. Because Keycloak is an OSS, anyone can use it as an OP or authorization server certified by the OIDF.
- Quality assurance of Keycloak. Each time a new Keycloak is released, regression testing is performed using the developed conformance test execution platform. This is to ensure that the new version remains compliant with the supported standards and security profiles.
- Publication of the study results. The developed conformance test execution platform was published on GitHub as a sub-project of Keycloak for easy accessibility.

## 2. Issues

### 2.1 Issue 1: Automation

To execute a conformance test using a Conformance Suite [11], it is necessary to prepare programs other than the authorization server or OP being tested. The program type is attributable to the security profile. For example, the FAPI security profile requires an OAuth 2.0's resource server. Several man-hours are required for the developer of the authorization server or OP to prepare such programs.

Moreover, if the automated conformance test can be executed as a part of existing CI of the authorization server or OP in parallel, the completion time of the conformance tests should be almost the same or shorter than the existing CI.

To resolve the issue, the following requirements need to be satisfied:

( 1 ) Test Automation: A conformance test using a Conformance Suite can be automatically executed.
( 2 ) Reduced Time of Test Automation: The completion time of the conformance test executed automatically is less than the one executed manually.
( 3 ) Test Applicability to CI: When the conformance test is executed as a part of existing CI of an authorization server or OP in parallel, the completion time of the conformance test is almost the same or shorter than the one of the existing CI.

*1 https://www.keycloak.org/
*2 https://github.com/keycloak/kc-sig-fapi?tab=readme-ov-file#passed-conformance-tests-per-keycloak-version
*3 https://github.com/keycloak/kc-sig-fapi/tree/main/conformance-tests-env

### 2.2 Issue 2: Cost Reduction of New Conformance Test Execution

When the developer of the authorization server or OP supports a new security profile, it is necessary to execute a new conformance test of the security profile using the Conformance Suite. This requires considerable effort by the developer of the authorization server or OP.

To resolve the issue, the following requirements need to be satisfied:

( 1 ) Execute New Security Profile's Conformance Test: The conformance test execution platform is developed and it can execute a new security profile's conformance test.

( 2 ) Reduced Effort of New Conformance Test: The conformance test execution platform reduces the effort for writing programs required to execute a new security profile's conformance test.

### 2.3 Issue 3: Parallelization

Although the Conformance Suite supports the automatic execution of conformance tests in series, it is not possible to automatically execute conformance tests of multiple security profiles in parallel. Therefore, considerable time is required to complete the execution of multiple conformance tests by the Conformance Suite, and the developer of an authorization server or OP cannot proceed to develop it until the execution is completed.

Moreover, if the automated conformance test can be executed as a part of existing CI of the authorization server or OP in parallel, the completion time of the conformance tests should be the same or shorter than the one of the existing CI.

To resolve the issue, the following requirements need to be satisfied:

( 1 ) Test Parallelization: Conformance tests using a Conformance Suite can be automatically executed in parallel.

( 2 ) Reduced Time of Test Parallelization: The parallelization reduces the completion time of several conformance tests.

( 3 ) Test Applicability to CI: When the conformance tests are executed as a part of existing CI of an authorization server or OP in parallel, the completion time of the conformance tests is almost the same or shorter than the one of the existing CI.

## 3. Design Principles

We devised design policies to develop a conformance test execution platform that resolves the issues described in Section 2.

Conformance tests themselves need to be executed by the Conformance Suite [11]: hence, the Conformance Suite must be included in the conformance test execution platform. Therefore, we first describe the logical structure of the Conformance Suite and subsequently explain the design policies to resolve each issue.

### 3.1 The Logical Structure of the Conformance Suite

We clarified the logical structure of the Conformance Suite as a class diagram (**Fig. 1**) by reverse engineering, reading its documentation and running it.

The Conformance Suite consists of the following components:

- Test Server: A server that executes conformance tests by simulating clients, Relying Parties (RPs), and a user's browser to interact with the authorization server or OP being tested.
- Test Database: A database for storing the data necessary for the execution of a conformance test and its results.
- Reverse Proxy: A reverse proxy in front of the test server.
- Test Configuration: A configuration for the execution of a conformance test. This shows which security-profile conformance test is executed against which authorization server or OP is being tested.
- Test Runner: A program that invokes the test server and executes the conformance test using the test configuration.

The Conformance Suite uses Docker[*4] to run each component in a container, except for the test configuration. The Conformance Suite uses Docker Compose[*5] to set and manage these containers.

The logical structure of the Conformance Suite includes a part that does not depend on the target of the conformance test and a part that depends on it. These are referred to as the test-target in-

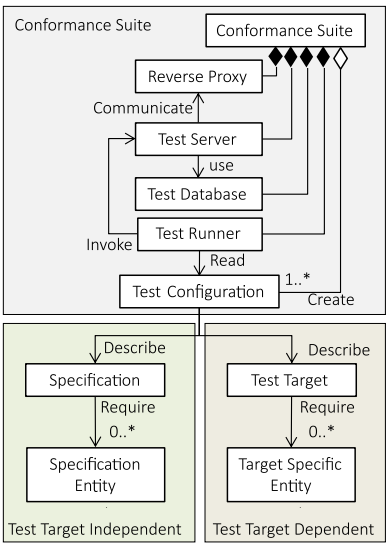


**Fig. 1** Class diagram of the logical configuration of the Conformance Suite.

[*4] https://www.docker.com/
[*5] https://docs.docker.com/compose/

dependent part and test-target dependent part, respectively.

The test-target independent part consists of the following components:

- Specification: A specification of a security profile.
- Specification Entity: An entity required to execute a conformance test for the specification.

The test-target dependent part consists of the following components:

- Test Target: An authorization server or OP being tested.
- Target Specific Entity: An entity required when executing a conformance test of the specification against the test target.

### 3.2 Issue 1: Automation

A developer of an authorization server or OP who uses the Conformance Suite must create the specification entity and the target-specific entity unassisted, which requires considerable amount of man-hours. Therefore, we created and included the specification entity and the target-specific entity in the conformance test execution platform as developer support resource. This saves time and labor.

The specification entity can be used regardless of whether the authorization server or OP is the test target. However, the target-specific entity is different for each authorization server or OP as the test target and cannot be shared among the test targets. In this study, we used the Keycloak as the test target.

For example, when executing a conformance test of FAPI 1.0 Advanced security profile, which was first supported by Keycloak 14, the resource server (A in Fig. 2), is required as the specification entity and the client-key hosting server (B in **Fig. 2**) is required as the target-specific entity.

Because the components of the Conformance Suite run in containers using Docker, the specification and target-specific entities are implemented as programs that run in containers. If these components do not run in containers, they must be operated in the computer environment of each developer of the authorization server or OP. For example, if a developer of an authorization server who uses a Windows machine and another who uses a Linux-based machine wants to execute a conformance test using the Conformance Suite, separate conformance test execution platforms must be created for each machine. This increases the development and maintenance cost of the platform. If the components run in containers, a conformance test execution platform must be developed and maintained.

When executing a conformance test, the specification and target-specific entities must communicate with each component of the Conformance Suite and the test target. Therefore, the specification and target-specific entities can be connected to the Docker network of the Conformance Suite.

### 3.3 Issue 2: Cost Reduction of New Conformance Test Execution

To execute a new security-profile conformance test, new specification and target-specific entities must be added. In addition, the behaviors of the existing specification entity, test target, and target-specific entity must be changed. **Fig. 3** shows a class diagram of the logical configuration of the test configuration that considers these points.

For example, to execute a conformance test of the FAPI-CIBA security profile, which was first supported by Keycloak 15, it is necessary to prepare an element called an authentication entity server as a target-specific entity (A in **Fig. 4**). In addition, it is necessary to change the existing Keycloak settings (B in Fig. 4).

To resolve Issue 2, it is necessary to add new specification entities and target-specific entities. Subsequently, existing specification entities, test targets, and target-specific entities must be

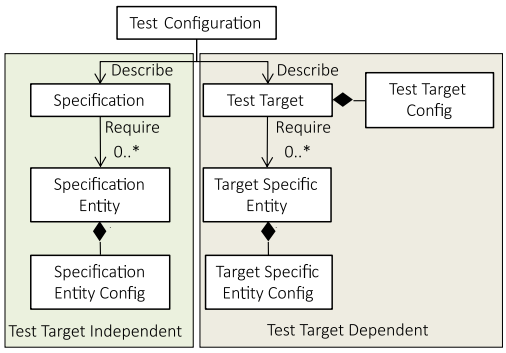


**Fig. 3** Class diagram of the logical configuration of the test configuration.

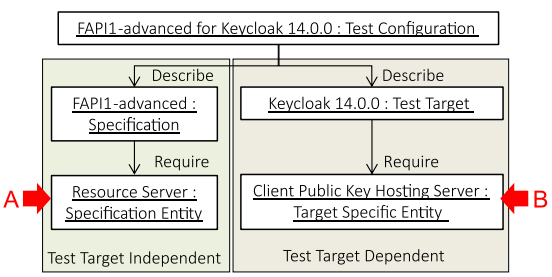


**Fig. 2** Object diagram of the test configuration.

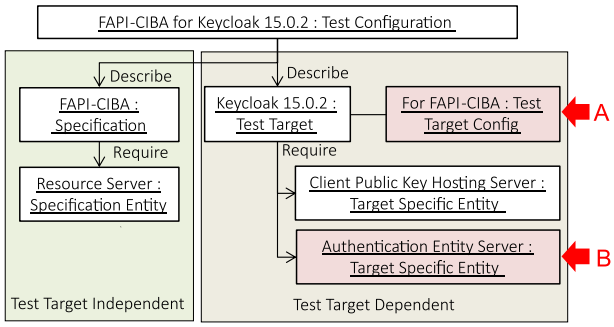


**Fig. 4** Object diagram of the test configuration.

changed. By formulating these tasks, developers of authorization server or OP can avoid the problem of executing new conformance tests through trial and error.

### 3.4 Issue 3: Parallelization

We considered the following design policies for automatically executing multiple conformance tests in parallel:

( 1 ) Automatically execute a single conformance test by Conformance Suite and using a script.

( 2 ) Prepare computer environments such as multiple Virtual Machines (VMs) and containers, and automatically execute the script to run a conformance test of each security profile using the Conformance Suite in parallel.

Regarding the mechanism for achieving 2, we initially decided not to implement it on the conformance test execution platform. This is because other mechanisms, such as GitHub Action[*6] can accomplish this.

## 4. Developing a Conformance Test Execution Platform

A conformance test execution platform that resolves the issues described in Section 2 was developed according to the design policies described in Section 3. Keycloak software was used.

The initial version of the conformance test execution platform was built after the release of Keycloak 12. However, this study describes a conformance test execution platform that was rebuilt when Keycloak 18 was released. This is because the conformance test execution platform was rebuilt according to the design policies described in Section 3 to resolve the issues described in Section 2 (**Table 2**)[*7,*8,*9,*10,*11,*12,*13,*14,*15,*16,*17,*18,*19].

**Table 2** Standard specifications supported by Keycloak.

| Version | Specifications | Release Date |
|---|---|---|
| 2.3.0 | OIDC | Oct 16, 2016[*7] |
| 12.0.0 | Financial-grade API Read and Write API security profile[*8] | Dec 17, 2020[*9] |
| 14.0.0 | FAPI1 Advanced[*10] | Jun 18, 2021[*11] |
| 15.0.0 | FAPI-CIBA, Brazil Open Banking, Australia CDR[*12] | Jul 30, 2021[*13] |
| 18.0.0 | OIDC for Logout[*14] | Apr 21, 2022[*15] |
| 20.0.0 | UK Open Banking[*16] | Nov 1, 2022[*17] |
| 23.0.0 | FAPI 2.0 security profile Second Implementer's Draft (FAPI2 SP), FAPI 2.0 Message Signing First Implementer's Draft (FAPI2 MS)[*18] | Nov 23, 2023[*19] |

[*6] https://docs.github.com/en/actions
[*7] https://github.com/Keycloak/Keycloak/releases/tag/2.3.0.Final
[*8] https://www.Keycloak.org/docs/latest/release_notes/index.html#Keycloak-12-0-0
[*9] https://github.com/Keycloak/Keycloak/releases/tag/12.0.0
[*10] https://www.Keycloak.org/docs/latest/release_notes/index.html#Keycloak-14-0-0

Based on our study [13], [14], [15], [16] and contributions to Keycloak by the OSS community, Keycloak 18 supports the following four FAPI security profiles:

- FAPI 1.0 Advanced Final (FAPI1 Advanced)
- FAPI 1.0 Client Initiated Backchannel Authentication (FAPI-CIBA)
- Australia Consumer Data Right (Australia CDR)
- Brazil Open Banking

Additionally, Keycloak 18 supports the following one standard specification defined by OIDF:

- OpenID Connect 1.0 for Logout Profiles (OIDC for Logout)

Keycloak already supported the following one standard specification defined by OIDF:

- OpenID Connect 1.0 (OIDC)

Therefore, the conformance test execution platform supports the conformance test execution of the four security profiles and two standard specifications.

### 4.1 Issue 1: Automation

To resolve Issue 1, the conformance test execution platform implements the specification entity and target-specific entity using Docker containers, connects them to the Docker network to communicate with other components and uses Docker Compose to set up and manage the containers according to the design policies described in Section 3.2.

**Fig. 5** shows the container network configuration of the conformance test execution platform and **Fig. 6** shows the main files of the conformance test execution platform.

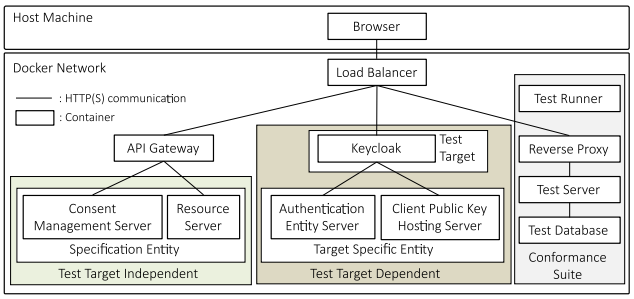


**Fig. 5** Container network configuration diagram.

[*11] https://github.com/Keycloak/Keycloak/releases/tag/14.0.0
[*12] https://www.Keycloak.org/docs/latest/release_notes/index.html#Keycloak-15-0-0
[*13] https://github.com/Keycloak/Keycloak/releases/tag/15.0.0
[*14] https://www.Keycloak.org/docs/latest/release_notes/index.html#Keycloak-18-0-0
[*15] https://github.com/Keycloak/Keycloak/releases/tag/18.0.0
[*16] https://github.com/Keycloak/Keycloak/pull/13068
[*17] https://github.com/Keycloak/Keycloak/releases/tag/20.0.0
[*18] https://www.keycloak.org/2023/11/keycloak-2300-released.html
[*19] https://github.com/keycloak/keycloak/tree/23.0.0

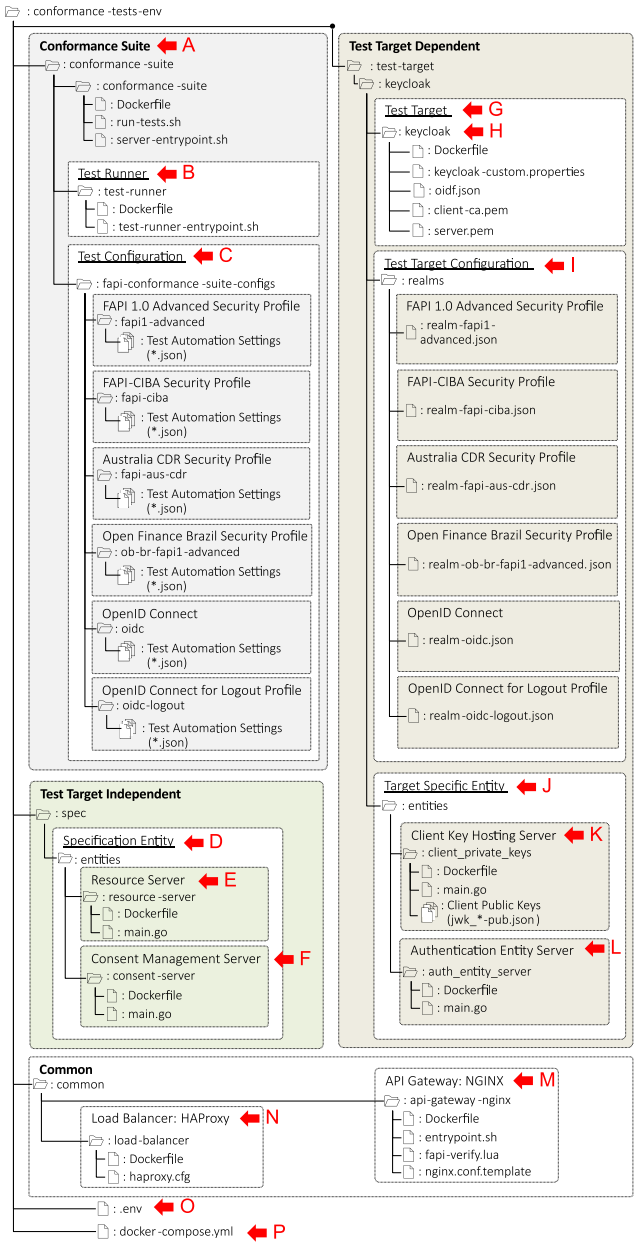


**Fig. 6** Layout of main files of the conformance test execution platform.

- Conformance Suite (A in Fig. 6): The OSS published by OIDF to execute a conformance suite of a security profile or standard specification defined by OIDF against an authorization server or OP.

As described in Section 3, the Conformance Suite consists of three containers: the test server, test database, and reverse proxy. The reverse proxy communicates outside the Conformance Suite. The test server executes the conformance tests, stores the results in the test database, and provides web-pages to show the results to a browser. The test server simulates the clients, RPs, and the user's browser to interact with an authorization server or the OP being tested. The test server determines whether the authorization server or OP has passed the conformance test.

The Conformance Suite was not provided as a Docker image. Therefore, the Dockerfile (A in Fig. 6) downloads the source codes of the Conformance Suite and builds them to run the test server, test database, and reverse proxy in each Docker container.

- Test Runner (B in Fig. 6): A script referred to as run-testh. sh invokes the test server and enable it to execute conformance test automatically (as described in Section 3).
- Test Configuration (C in Fig. 6): Configuration files of each security profile conformance test stored in the directory named fapi-conformance-suite-configs (as described in Section 3).
- Specification Entity (D in Fig. 6): An entity required to execute the conformance test of a security profile (as described in Section 3).

There are two specification entities: resource server and consent management server.

- Resource Server (E in Fig. 6): A server defined in OAuth 2.0 for managing resources accessed by a client with an access token. Execution of the conformance tests of FAPI1 Advanced, FAPI-CIBA, Australia CDR, and Brazil Open Banking requires a resource server. Therefore, we created and included a resource server in the conformance test execution platform, although the resource server is irrelevant to a test target such as Keycloak.
- Consent Management Server (F in Fig. 6): A server that manages consent obtained from a user for a client to access their resources on the resource server with an access token. The execution of the conformance test of Brazil Open Banking needs the consent management server. Therefore, we created the consent management server and included it into the conformance test execution platform, although the resource server is irrelevant to a test target such as Keycloak.
- Test Target (G in Fig. 6): An entity against which a conformance test is executed (as described in Section 3).
- Keycloak (H in Fig. 6): Actual test target of the conformance test execution platform.
- Test Target Configuration (I in Fig. 6): A configuration file of Keycloak.

Because the security requirements imposed on the authorization server or OP differ from the security profile, the conformance test execution platform prepared a configuration file that satisfied the requirements of each security profile.

- Target Specific Entity (J in Fig. 6): An entity required for executing a conformance test against a specific test target (as described in Section 3).
- Client Key Hosting Server (K in Fig. 6): A server that provides a public key of a client. Some conformance tests require the client to provide a public key for a digital signa-

ture. There are two ways to provide the public key: registering it to an authorization server or OP or providing it by a separate server, such as the client-key hosting server. The execution of the conformance tests against Keycloak requires the latter server. Therefore, the client-key hosting server was created and included in the conformance test execution platform.

- Authentication Entity Server (L in Fig. 6): A server that performs user authentication in CIBA [17].

The specification of the CIBA does not define the method of authenticating a user. Therefore, the method differs from authorization servers or OPs. Keycloak uses an authentication entity server for user authentication in the CIBA. Execution of a conformance test for FAPI-CIBA against Keycloak requires the server. Therefore, we created and included the authentication entity server in the conformance test execution platform.

- API Gateway (M in Fig. 6): A gateway placed in front of the resource server, authentication entity server and consent management server to perform common processing and route their communication inside the Docker network.
- Load Balancer (N in Fig. 6): A load balancer placed in front of the Conformance Suite, Keycloak and API gateway to perform common processing and route their communication outside the Docker network.
- .env (O in Fig. 6): A file that defines the environment variables for Docker. Environment variables were used to configure the conformance test execution platform.
- docker-compose.yml (P in Fig. 6): A configuration file for Docker Compose that configures all containers.

By using the conformance test execution platform, a developer of an authorization server or OP who executes conformance tests of security profiles and standard specifications defined by the OIDF and uses Keycloak as their test target can execute conformance tests without having to create programs other than Keycloak.

### 4.2 Issue 2: Cost Reduction of New Conformance Test Execution

To resolve Issue 2, we formulated procedures for making the conformance test execution platform to execute a conformance test of a new security profile following the design policies in Section 3.3. The procedures are as follows:

( 1 ) The work related to the test-target dependent part
  (a) Test Target Configuration: A developer creates a Keycloak configuration file for a conformance test of the new security profile and stores it in the keycloak/realms directory (I in Fig. 6).
  (b) Target Specific Entity: When the developer needs to add a new target-specific entity, it must be created as a new container. First, a new directory is created. Subsequently, the Dockerfile is created and stored in the directory (J in Fig. 6). Finally, the container settings are written in docker-compose.yml (P in Fig. 6). When new functionalities are required, the files of existing target-specific entities are modified (J in Fig. 6).

( 2 ) The work related to the test-target independent part
  (a) Specification Entity: When developer adds a new specification entity, it must be created as a new container. First, a new directory is created. Next, the Dockerfile is created and stored in the directory (D in Fig. 6). Finally, the container settings are written in docker-compose.yml (P in Fig. 6). When new functionalities are required, the files of existing specification entities are modified (D in Fig. 6).

( 3 ) Conformance Suite
  (a) Test Configuration: A developer creates a new security profile conformance test configuration file and stores it in fapi-conformance-suite-configs directory (C in Fig. 6).
  (b) Conformance Suite: The developer determines the value of the Docker environment variable TEST_PLAN to call the conformance test configuration file of the new security profile, and modifies the script for automatic test execution (run-tests.sh). (A in Fig. 6).

( 4 ) Test execution: The developer sets the Docker environment variable TEST_PLAN to the value determined in the procedure 3.a to execute the conformance test of the new security profile and sets the Docker environment variable KEYCLOAK_REALM_IMPORT_FILENAME to the name of the Keycloak realm configuration file created and stored in the procedure 1 (O in Fig. 6). Subsequently, the following test execution command is executed:

```
docker-compose -p keycloak-fapi \
-f docker-compose.yml up --build
```

By following the formulated procedures, the developer of an authorization server or OP can execute a conformance test of a new security profile against its test target, such as Keycloak, without any difficulty through trial and error.

### 4.3 Issue 3: Parallelization

To resolve Issue 3, we considered a mechanism enables automatic execution of conformance tests in parallel.

According to the design policy in Section 3.4, we did not implement a mechanism to execute multiple conformance tests simultaneously on the conformance test execution platform. Instead, we used a GitHub Action to execute them in parallel. This

is because this conformance test execution platform is published in the GitHub repository: hence, the conformance test execution platform can use GitHub Actions by default.

We decided to build a GitHub Actions workflow that executes the following processes:

( 1 ) First, execute each conformance test in parallel as a GitHub Action's job.

( 2 ) After all the jobs have been executed, execute a job that aggregates the results of them and outputs the name of the failed test.

Finally, we refer to the results of Step 2 on the GitHub Actions management console in a browser and checked whether all the conformance tests were passed.

By using the GitHub Actions workflow, a developer of an authorization server or OP can execute multiple conformance tests of security profiles against their test target like Keycloak by using the conformance test execution platform in parallel.

## 5. Evaluation

We evaluated the conformance test execution platform described in Section 3 to confirm whether it resolved the issues described in Section 2.

### 5.1 Issue 1: Automation

The conformance test execution platform was published on GitHub[*20] for easy accessibility. To confirm that the conformance test execution platform resolves Issue 1, we simulated a developer of Keycloak as an authorization server that has the computing environment listed below, used git[*21] to clone the repository of the published conformance test execution platform on the computing environment, and ran the conformance test of Australia CDR against Keycloak.

- Processor: Intel(R) Core(TM) i7-10610U CPU 1.80 GHz / 2.30 GHz
- RAM: 32.0 GB
- System: 64 bit OS, x64 based processor
- OS(Host): Windows 10 Pro version 21H2 build 19044.1826
- OS(Virtual): Ubuntu 20.04 (running on Windows Subsystem for Linux 2)
- Docker Compose version: 1.27.4 build 40524192
- Browser: Google Chrome 103.0.5060.134

As developers of Keycloak, we executed Australia CDR security profile conformance test against Keycloak 23.0.3, using the cloned conformance test execution platform on both computing environment of Windows 10 and Ubuntu 20.04 (running on Windows Subsystem for Linux 2). We confirmed that the conformance test can be executed in both computing environments by reviewing the log of the conformance test, as shown in **Fig. 7**.

```
...
32756: test_runner_1 Run Australia CDR fapi1-advanced tests
...
70128: test_runner_1 2023-12-24 01:02:52 Results for
       fapi1-advanced-final-test-plan
       [client_auth_type=private_key_jwt]
       [fapi_profile=consumerdataright_au]
       [fapi_response_mode=plain_response]
       [fapi_auth_request_method=by_value]
       with configuration
        ../conformance-suite/.gitlab-ci/
           fapi-conformance-suite-configs/fapi-aus-cdr/
           fapi-aus-cdr-private-key-PS256-PS256-automated.json:
...
70169: test_runner_1 2023-12-24 01:02:52 Overall totals:
       ran 40 test modules.
       Conditions: 3306 successes, 0 failures, 0 warnings.
       86.6 seconds
...
70174: test_runner_1 2023-12-24 01:02:52 Results for
       fapi1-advanced-final-test-plan
       [client_auth_type=private_key_jwt]
       [fapi_profile=consumerdataright_au]
       [fapi_response_mode=plain_response]
       [fapi_auth_request_method=pushed]
       with configuration
       ../conformance-suite/.gitlab-ci/
         fapi-conformance-suite-configs/fapi-aus-cdr/
         fapi-aus-cdr-private-key-par-PS256-PS256-automated.json:
...
70299: test_runner_1 2023-12-24 01:02:52 Overall totals:
       ran 53 test modules.
       Conditions: 4664 successes, 0 failures, 0 warnings.
      390.1 seconds
...
```

**Fig. 7** Log of the conformance test (partial excerpt).

The number to the left of the log indicates the line number of the log file. This is an addition to the original log to make it easier to read. Furthermore, logs with long lines were wrapped.

Line 32756 in the log indicates the start of the conformance test for Australia CDR security profile.

The Australia CDR security profile has two test patterns: AU-CDR Adv. OP w/ Private Key and AU-CDR Adv. OP w/ Private Key, PAR.

Line 70128 of the log indicates that the conformance test of the test pattern for AU-CDR Adv. OP w/ Private Key has been completed, and line 70169 of the log indicates that the number of failed tests (shown by failures) is zero, implying that Keycloak passed the conformance test of this test pattern.

Line 70174 of the log indicates that the conformance test of the test pattern for AU-CDR Adv. OP w/ Private Key, PAR has been completed, and line 70299 of the log indicates that the number of failed tests (shown by failures) is zero, implying that Keycloak passed the conformance test of this test pattern.

Considering these points, we confirmed that Keycloak passed the conformance test of Australia CDR, implying that Keycloak complies with the specification of Australia CDR and the conformance test execution platform satisfies requirement (1) Test

[*20] https://github.com/keycloak/kc-sig-fapi/tree/main/conformance-tests-env

[*21] https://git-scm.com/

Automation in Section 2.1.

We executed Australia CDR security profile conformance test against Keycloak 23.0.3 on the same computing environment ten times with both automatically and manually by one of the authors who are familiar with this conformance test. The average completion time of the conformance test executed automatically was 7 minutes 29 seconds with 9 seconds of its standard error while the one executed manually was 17 minutes 19 seconds with 23 seconds of its standard error, which shows that the completion time of the conformance test was reduced by 56.8% by this test automation. Therefore, the conformance test execution platform reduces the time required to complete the conformance test, implying that the conformance test execution platform satisfies requirement (2) Reduced Time of Test Automation in Section 2.1.

The existing Keycloak's CI includes six integration tests[*22] called Base IT (1) to (6) and these tests are executed in parallel. We investigated ten runs of the existing Keycloak's CI and found that the maximum completion time was 35 minutes 19 seconds with 57 seconds of its standard error, which is greater than the average completion time of the conformance test executed automatically in 7 minutes 29 seconds with 9 seconds of its standard error. Therefore, the conformance test execution platform satisfies requirement (3) Test Applicability to CI in 2.1.

### 5.2 Issue 2: Cost Reduction of New Conformance Test Execution

To confirm that the conformance test execution platform resolves Issue 2, we simulated a developer of Keycloak an authorization server that need to execute the conformance test of UK Open Banking that Keycloak 20 newly supports, and executed the conformance test against Keycloak 23.0.3, following the procedures formulated in Section 4.2.

( 1 ) Work related to test-target dependent part

(a) Test Target Configuration: We created Keycloak's configuration file realm-fapi-uk-ob.json for UK Open Banking and stored it in a keycloak/realms directory (A in **Fig. 8**).

(b) Target Specific Entity: We needed to do nothing.

( 2 ) The work related to test-target independent part

(a) Specification Entity: The conformance test of UK Open Banking requires the resource server to operate in accordance with the UK Open Banking API Profile [18]. Therefore, we added a new functionality to perform processing according to this profile in the source code of the resource server (resource-server/main.go) (B in Fig. 8).

( 3 ) Conformance Suite

(a) Test Configuration: UK Open Banking has two test patterns: UK-OB Adv. OP w/ MTLS and UK-OB Adv. OP w/ Private Key. Therefore, we created fapi-uk-ob-mtls-PS256-PS256-automated.json and fapi-uk-ob-private-key-PS256-PS256- automated.json as configuration files for each test pattern, and stored them in fapi-conformance-suite-configs directory (C in Fig. 8).

(b) Conformance Suite: We modified the script for automatic test execution (run-tests.sh) (D in Fig. 8) to read the newly added configuration files for each test pattern and run the conformance tests for each test pattern if the value of the Docker environment variable TEST_PLAN was set to --fapi-uk-ob-all.

( 4 ) Test execution: We executed the conformance tests of each test pattern automatically by setting the value of the Docker environment variable TEST_PLAN to --fapi-uk-ob-all (set up in Step 3.2) and setting KEYCLOAK_REALM_IMPORT_FILENAME to realm-fapi-uk-ob.json (created in Step 1.1).

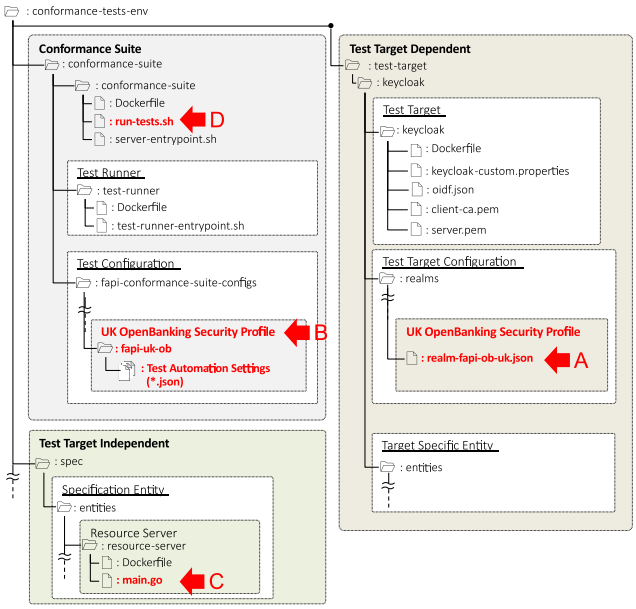


**Fig. 8** Added/modified files to the conformance test execution platform.

By reviewing the log of the conformance test shown in **Fig. 9**, we confirmed that Keycloak passed the UK Open Banking conformance test.

The number to the left of the log indicates the line number of the log file. This is an addition to the original log to make it easier to read. Furthermore, logs with long lines were wrapped.

Line 32693 of the log indicates the conformance test of the UK Open Banking security profile has started.

[*22] https://github.com/keycloak/keycloak/blob/23.0.3/.github/workflows/ci.yml#L86-L123

```
...
32693: test_runner_1 Run UK OpenBanking fapi1-advanced tests
...
47828: test_runner_1 2023-12-24 00:57:47 Results for
       fapi1-advanced-final-test-plan
       [client_auth_type=private_key_jwt]
       [fapi_profile=openbanking_uk]
       [fapi_response_mode=plain_response]
       [fapi_auth_request_method=by_value]
       with configuration
       ../conformance-suite/.gitlab-ci/
          fapi-conformance-suite-configs/fapi-uk-ob/
          fapi-uk-ob-private-key-PS256-PS256-automated.json:
...
47874: test_runner_1 2023-12-24 00:57:47 Overall totals:
       ran 42 test modules.
       Conditions: 4183 successes, 0 failures, 1 warnings.
       87.9 seconds
...
47880: test_runner_1 2023-12-24 00:57:47 Results for
       fapi1-advanced-final-test-plan
       [client_auth_type=mtls]
       [fapi_profile=openbanking_uk]
       [fapi_response_mode=plain_response]
       [fapi_auth_request_method=by_value]
       with configuration
       ../conformance-suite/.gitlab-ci/
          fapi-conformance-suite-configs/fapi-uk-ob/
          fapi-uk-ob-mtls-PS256-PS256-automated.json:
...
47919: test_runner_1 2023-12-24 00:57:47 Overall totals:
       ran 36 test modules.
       Conditions: 3410 successes, 0 failures, 1 warnings.
       67.7 seconds
...
```

**Fig. 9** Log of the conformance test (partial excerpt).

The UK Open Banking security profile has two test patterns: UK-OB Adv. OP w/ Private Key and UK-OB Adv. OP w/ MTLS.

Line 47828 of the log indicates that the test pattern for UK-OB Adv. OP w/ Private Key has been completed, and line 47874 of the log indicates that the number of failed tests (shown by failures) is zero, implying that Keycloak passed the conformance test of this test pattern.

Line 47880 of the log indicates that the test pattern for UK-OB Adv. OP w/ MTLS has been completed, and line 47919 of the log indicates that the number of failed tests (shown by failures) is zero, implying that Keycloak passed the conformance test of this test pattern.

Considering these points, we confirmed that Keycloak passed the conformance test of UK Open Banking, implying that Keycloak complies with the specification of UK Open Banking and the conformance test execution platform satisfies requirement (1) Execute New Security Profile's Conformance Test of Section 2.2.

To execute UK OpenBanking security profile, a developer needs to create Resource Server (written in Go, E in Fig. 6), API Gateway (written in Lua, M in Fig. 6), and Client Key Hosting Server (written in Go and shell script, K in Fig. 6) whose lines of code are 169, 217, and 459, respectively. Therefore, the developer needs to create them whose lines of code is about 845 in total if they do not use the conformance test execution platform.

On the other hand, if the developer uses the conformance test execution platform to execute UK OpenBanking security profile, they need to modify Resource Server to meet the requirements of UK OpenBanking security profile, which cost them to modify 121 lines of code[*23]. Therefore, lines of code of programs the developer needs to write was reduced by 85.7% by using the conformance test execution platform, which implies that the conformance test execution platform satisfies requirement (2) Reduced Effort of New Conformance Test of Section 2.2.

### 5.3 Issue 3: Parallelization

As described in Section 4.3, we did not implement a mechanism to execute multiple conformance tests in parallel on the conformance test execution platform used in this study. Instead, we used the GitHub Actions[*24] workflow.

We performed conformance tests on the GitHub Actions workflow definition file run-conformance-tests.yml, which defines nine jobs for executing the conformance tests of each of the nine security profiles and standard specifications, and one job that aggregates the results of these conformance test executions (**Fig. 10**).

The number to the left of the log indicates the line number of the log file. This is an addition to the original log to enhance readability. Furthermore, logs with long lines were wrapped.

In line 6, a job that executes the conformance test is referred to as the name run-conformance-test. As shown in line 11, by using Matrix Strategy, we save the effort of writing each job to execute conformance tests of the nine security profiles and standard specifications individually. By setting the value of the profile in line 12 to the value of the Docker environment variables TEST_PLAN and KEYCLOAK_REALM_IMPORT_FILENAME, a job in which a conformance test of the security profile or standard specification is indicated by this value is automatically generated. (**Table 3**).

Starting from line 50, a job that aggregates the execution results of the nine conformance tests is referred to as the evaluate-test-results. As shown in line 51, this job begins its execution after all nine jobs were completed. This job reads the execution result log for each conformance test, prints the name of the test that failed to pass, as shown in line 88, and makes it available for download from the browser as an artifact, as shown in lines 89 to 92.

This workflow has been published as a branch of one of the

[*23] https://github.com/keycloak/kc-sig-fapi/pull/346/files#diff-5c29cf98a783fab59889e0c8d2c879d2c2a7a3c871abdd7c275ce671ab02675e

[*24] https://docs.github.com/en/actions

```
...
 5: jobs:
 6:  run-conformance-test:
 7:   runs-on: ubuntu-latest
 8:    continue-on-error: true
 9:    strategy:
10:     fail-fast: false
11:     matrix:
12:      profile: [fapi-aus-cdr, fapi-ciba, fapi-uk-ob, fapi1-advanced,
                   fapi2-ms-id2, fapi2-sp-id2, ob-br-fapi1-advanced,
                   oidcc, oidcc-logout]
...
16:    steps:
...
21:     env:
...
28:      TEST_PLAN: --${{ matrix.profile }}-all
...
33:      KEYCLOAK_REALM_IMPORT_FILENAME:
           realm-${{ matrix.profile }}.json
...
50: evaluate-test-results:
51:  needs: [run-conformance-test]
...
56:  steps:
57:   - uses: actions/download-artifact@v2
58:     with:
59:      path: matrix-job-outputs
60:   - uses: actions/github-script@v4
61:     with:
62:      script: |
...
88:       fs.writeFileSync("${{ github.workspace }}/
          all-tests-outputs.txt", JSON.stringify(outputs))
89:   - uses: actions/upload-artifact@v2
90:     with:
91:       name: all-tests-outputs
92:       path: ${{ github.workspace }}/all-tests-outputs.txt
```

**Fig. 10** Workflow definition file (partial excerpt).

**Table 3** GitHub Action's jobs created by Matrix Profile.

| Matrix Profile | Job | Specification of Conformance Test |
|---|---|---|
| fapi-aus-cdr | run-conformance-test (fapi-aus-cdr) | Australia CDR |
| fapi-ciba | run-conformance-test (fapi-ciba) | FAPI-CIBA |
| fapi-uk-ob | run-conformance-test (fapi-uk-ob) | UK Open Banking |
| fapi1 -advanced | run-conformance-test (fapi1-advanced) | FAPI1 Advanced |
| fapi2-ms-id2 | run-conformance-test (fapi2-ms-id2) | FAPI2 MS |
| fapi2-sp-id2 | run-conformance-test (fapi2-sp-id2) | FAPI2 SP |
| ob-br-fapi1 -advanced | run-conformance-test (ob-br-fapi1-advanced) | Brazil Open Banking |
| oidcc | run-conformance-test (oidcc) | OIDC |
| oidcc-logout | run-conformance-test (oidcc-logout) | OIDC for Logout |

authors' GitHub repository[*25].

The results of executing this workflow are shown in the GitHub's console (**Fig. 11**).

The workflow took 1 hour 30 minutes and 57 seconds to complete. Among the nine conformance tests, there were some tests that cannot be automatically executed due to the nature of the tests, and the tests could not complete after waiting for an execution timeout. Due to this execution timeout wait, it took a long time for the workflow to complete. These tests we found were written on the manual this conformance test execution platform in the GitHub repository and we added notes showing that these tests must be executed manually[*26].

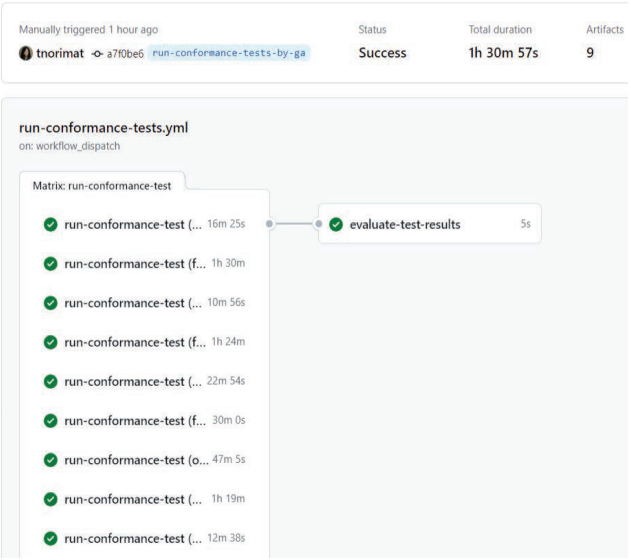


**Fig. 11** Parallel execution of nine conformance tests and their test results.

Considering the result of the workflow, we confirmed that all nine conformance tests were executed in parallel, implying that the conformance test execution platform satisfies requirement (1) Test Parallelization of Section 2.3.

We ran the workflow ten times and found the average completion time of the workflow and calculated the average completion time of each nine conformance tests. Moreover, we estimated the completion time if all nine conformance tests are executed in tandem by summing completion time of each nine conformance tests and averaging them.

The average completion time of the workflow for test parallelization was 79 minutes 35 seconds with 14 seconds of its standard error. The estimation of the completion time if all nine conformance tests are executed in tandem was 211 minutes and 24 seconds with 20 seconds of standard error. The former was shorter than the latter and decreased by 62.4% compared with the latter, which means that the completion time of nine conformance tests were reduced by 62.4% by this test parallelization. Therefore, the conformance test execution platform satisfies requirement (2) Reduced Time of Test Parallelization of Section 2.3.

We compared the average completion time of the workflow

[*25] https://github.com/tnorimat/keycloak-fapi/blob/test-env-baseline/.github/workflows/run-conformance-tests.yml

[*26] https://github.com/keycloak/kc-sig-fapi/tree/main/conformance-tests-env#not-passed-tests-automatically

with the average completion time of Keycloak's CI calculated in Section 5.1. The former was 79 minutes 35 seconds with 14 seconds of its standard error while the latter was 79 minutes 12 seconds with 380 seconds of its standard error. Therefore, the average completion time of the workflow was almost the same as the average completion time of Keycloak's CI, which implies that the conformance test execution platform satisfies requirement (3) Test Applicability to CI of Section 2.3.

## 6. Discussion

Based on the evaluation results presented in Section 5, we confirmed that the conformance test execution platform resolved the three issues presented in Section 2.

In Section 5.1, we cloned the conformance test execution platform published in the GitHub repository, and confirmed that it is possible to execute a conformance test using the conformance test execution platform on both OSes of Windows and Linux (Ubuntu). We also confirmed that the completion time of the conformance test was reduced by 56.8% by this test automation.

In Section 5.2, we attempt to perform a new conformance test for UK Open Banking security profile and confirm that it is possible by following the procedures formulated in Section 4.2, without having to spend time on trial and error. We also confirmed that lines of code of programs the developer needs to write without the conformance test execution platform was reduced by 85.7% by using the conformance test execution platform.

In Section 5.3, we confirmed that the nine conformance test types can be automatically executed in parallel using the conformance test execution platform with GitHub Actions. We also confirmed that the completion time of nine conformance tests was reduced by 62.4% by this test parallelization.

### 6.1 Applicability to an Authorization Server or OP other than Keycloak

According to the design policies in Section 3, the conformance test execution platform is divided into one that does not depend on the test target (test-target independent) and one that depends on the test target (test-target dependent). Therefore, we believe that it is possible to perform a conformance test by following the following five procedures.

- Adding a Test Target: We must prepare a Dockerfile that runs the test target in a container under the test target directory (I in Fig. 6).
- Adding Test Target Configuration: We must prepare a configuration file for the test target for each security profile under the test target directory (G in Fig. 6).
- Adding a Target Specific Entity: We must prepare a Dockerfile and other files for an entity required by the test target (J in Fig. 6).
- Changing Test Configuration: We must change each configuration file for automatic conformance test execution in conformance-suite/fapi-conformance-suite-config directory such that an authorization server or OP other than Keycloak is the test target (C in Fig. 6).
- Starting the Test Target and Test Specific Entity in a container: We must modify the docker-compose.yml file, such that the test target and test-specific entity software run in containers and connect to the Docker network (P in Fig. 6).

### 6.2 Applicability to Quality Assurance of Keycloak

The conformance test execution platform can also be used as part of Keycloak regression testing. Every time a new version of Keycloak was released, we executed conformance tests as part of Keycloak regression testing to ensure that the new version was compliant with the security profiles supported by previous versions. The results are published on the OAuth SIG's website[*27].

## 7. Related Work

To execute conformance tests of specifications in Table 2, a conformance test execution platform needs to do the task of generating and sending HTTP request from a browser to an authorization or OP, and receiving and reading a HTTP response from the authorization or OP to the browser.

We investigated works for automatically and flexibly executing several types of conformance tests or vulnerability tests to find the methods of doing the task. Next, we compared the methods by the works with the one by the developed conformance test execution platform in this research.

### 7.1 Works for Automatic Execution of Conformance Tests and Vulnerability Test

OAuch, a system that semi-automatically executes conformance tests for OAuth 2.0, and its related security specifications against an authorization server, was developed in [19]. OAuth was published as OSS[*28] and Web Services[*29]. OAuch simulates a browser and client, and automatically sends a request to an authorization server being tested and determines whether its response meets the requirements of OAuth 2.0 and its related security specifications. OAuch requires its user to manually input data into the browser while [11] that is included in the confor-

[*27] https://github.com/keycloak/kc-sig-fapi?tab=readme-ov-file#passed-conformance-tests-per-keycloak-version
[*28] https://github.com/DistriNet/OAuch
[*29] https://oauch.io

mance test execution platform developed in this study can perform this operation automatically.

Whether an RP that uses Facebook, Google, or PayPal as an IdP has vulnerabilities was studied in [20]. The contents of the messages exchanged between the RP and IdP were automatically analyzed, and the results of the analysis were manually investigated to determine whether the RP had vulnerabilities.

SSOScan, a system that automatically executes a vulnerability diagnosis for an RP using Facebook as the IdP, was developed in [21]. SSOScan was published as OSS[*30] and Web Services[*31]. SSOScan creates an RP as the attacker and registers it on Facebook. Using an attacker's RP, SSOScan automatically determines whether the RP being tested has known vulnerabilities.

OAuthTester, a system that automatically executes vulnerability diagnosis for an RP using an IdP supporting an OAuth 2.0-based protocol, such as Facebook, was developed in [22]. OAuthTester uses a customized browser plugin to intercept messages exchanged between the RP and IdP. By investigating the contents of the messages, OAuthTester automatically constructs a finite-state machine describing the RP and IdP, and determines whether the RP has known and unknown vulnerabilities.

WPSE, a Chrome browser plugin that automatically executes vulnerability diagnoses for an RP that uses an IdP supporting an OAuth 2.0-based protocol or OIDC, was developed and released as OSS[*32] in [23]. The WPSE not only determines whether the RP has vulnerabilities but also prevents attacks that exploit vulnerabilities.

The vulnerabilities of an RP that uses an IdP supporting an OAuth 2.0-based protocol, such as Facebook, or OIDC, such as Google, were studied in [24], [25], [26]. By investigating the contents of the messages exchanged between the RP and the IdP, the study automatically determined whether the RP has unknown vulnerabilities that could lead to spoofing.

### 7.2 Comparing the Works with the Conformance Test Execution Platform

**Selenium WebDriver.** For doing the task of generating and sending HTTP request from a browser to an authorization or OP, and receiving and reading a HTTP response from the authorization or OP to the browser, the developed conformance test execution platform uses Selenium WebDriver[*33] and runs scripts to handle a browser automatically. The method for the task is appropriate for executing conformance tests in CI because the method can run tests automatically and it does not require additional setup and configuration for browser so it does not increase completion time of executing conformance tests in CI.

**Manual.** In [19], the task was done by handling a browser by a tester manually. This method for the task is not appropriate for executing conformance tests in CI because the method dose not execute conformance tests in CI automatically without human intervention.

**Proxy and plugin.** In [20], [24], the task was done by a plugin installed in a browser and a proxy installed a machine on which the browser run. This method for the task is not appropriate for executing conformance tests in CI because the method need to install and set up the plugin to the browser, and install and set up the proxy on the machine on which the browser runs whenever executing conformance tests, which cause additional overhead and increase completion time of executing conformance tests in CI. Moreover, a tester needs to modify the plugin and proxy if the tester executes conformance tests for a new specification, which requires additional amount of man-hours.

**Plugin.** In [22], [23], [25], [26], the task was done by a plugin installed in a browser. This method for the task is not appropriate for executing conformance tests in CI because the method need to install and set up a plugin to the browser whenever when executing conformance tests, which causes additional overhead and increases completion time of executing conformance tests in CI. Moreover, a tester needs to modify the plugin if the tester executes conformance tests for a new specification, which requires additional amount of man-hours.

**Web scraping.** In [21], the task was done by web scraping. This method for the task is not appropriate for executing conformance tests in CI because the method dose not execute conformance tests because a tester need to develop scraping scripts if the tester execute conformance tests for a new specification or executing conformance test against a new authorization server or OP, which requires additional amount of man-hours.

## 8. Conclusion

Many authorization servers or OP software products support multiple FAPI security profiles developed by the OIDF and Open Banking security profiles that are based on FAPI security profiles. However, creating programs other than the authorization server or OP software to execute their conformance tests, providing support for execution of a new conformance test if required by a new security profile for their product, and automatically executing multiple conformance tests is man-hour intensive.

To resolve these issues, we designed and developed a conformance test execution platform together with the OSS community OAuth SIG, assuming Keycloak as a specific test target for con-

*30 https://github.com/Treeeater/vulCheckerFirefox
*31 http://ssoscan.org/
*32 https://sites.google.com/site/wpseproject/
*33 https://www.selenium.dev/documentation/webdriver/

formance tests. By evaluating the platform, we confirmed that these issues were resolved.

We implemented FAPI security profiles and Open Banking security profiles based on the FAPI security profiles in Keycloak. Using our conformance test execution platform, we confirmed that Keycloak passed the conformance tests of these security profiles, suggesting that it complies with the specifications of the security profiles.

In addition, based on the conformance test results obtained from our platform, members of the Keycloak community other than the authors obtained certification from the OIDF, demonstrating that Keycloak complies with the specifications of the security profiles. Currently, Keycloak can be used as an OIDF-certified OP/Authorization server because it is an OSS.

For the quality assurance of Keycloak, owe executed conformance tests against Keycloak by using the platform as part of regression tests every time a new version of Keycloak was released. Consequently, we confirmed that the new version of Keycloak still complies with security profiles already supported by the existing version of Keycloak. To ensure easy accessibility, we have published these conformance test results and the conformance test execution platform developed in this study as a Keycloak sub-project in the OAuth SIG's GitHub repository.

In the future, we will improve the workflow of the GitHub Actions described in Section 4.3 and integrate the conformance test execution platform with Keycloak's CI/CD pipeline. This is to ensure that the new version of Keycloak can be released in compliance with security profiles and standards already supported by the existing version of Keycloak.

**Acknowledgments** The initial foundation of the conformance test execution platform was created by Hiroyuki Wada of Nomura Research Institute, Brendan Rothwell and Daniel Huffer of Integral in Australia, and Arun Ganesh Alagappan of Maverick Systems in India. Subsequently, Keycloak maintainer Marek Posolda of Red Hat in the Czech Republic, Keycloak maintainer Thomas Darimont of codecentric AG in Germany, Raphael Abreu of Red Hat in Brazil, and Shane Boulden of Red Hat in Australia have contributed to improving the platform. We would like to express our gratitude to them.

## References

[1] OpenID Foundation: FAPI Working Group - Specifications (online), available from 〈https://openid.net/wg/fapi/specifications〉 (accessed 2023–12–30).

[2] IETF: RFC 6749 The OAuth 2.0 Authorization Framework (online), available from 〈https://datatracker.ietf.org/doc/html/rfc6749〉 (accessed 2023–12–30).

[3] OpenID Foundation: OpenID Connect Core 1.0 incorporating errata set 2 (online), available from 〈https://openid.net/specs/openid-connect-core-1_0.html〉 (accessed 2023–12–30).

[4] The Open Banking Implementation Entity: Open Banking Security Profiles (online), available from 〈https://standards.openbanking.org.uk/security-profiles〉 (accessed 2023–12–30).

[5] OpenID Foundation: Financial-grade API - Part 2: Read and Write API Security Profile (online), available from 〈https://openid.net/specs/openid-financial-api-part-2-wd-06.html〉 (accessed 2023–12–30).

[6] The Data Standards Body: Consumer Data Right Security Profile (online), available from 〈https://consumerdatastandardsaustralia.github.io/standards/#security-profile〉 (accessed 2023–12–30).

[7] OpenID Foundation: Financial-grade API Security Profile 1.0 - Part 2: Advanced (online), available from 〈https://openid.net/specs/openid-financial-api-part-2-1_0.html〉 (accessed 2023–12–30).

[8] Banco Central do Brasil: Open Finance Brasil Financial-grade API Security Profile 1.0 Implementers Draft 3 (online), available from 〈https://openfinancebrasil.atlassian.net/wiki/spaces/DraftOF/pages/76283925/EN+Open+Finance+Brasil+Financial-grade+API+Security+Profile+1.0+Implementers+Draft+3〉 (accessed 2023–12–30).

[9] OpenID Foundation: Financial-grade API Security Profile 1.0 - Part 1: Baseline (online), available from 〈https://openid.net/specs/openid-financial-api-part-1-1_0.html〉 (accessed 2023–12–30).

[10] Banfico, Ltd.: IBM API Connect in Saudi Open Banking Implementation (online), available from 〈https://www.banfico.com/ibm-api-connect-in-saudi-open-banking〉 (accessed 2023–12–30).

[11] OpenID Foundation: About the Conformance Suite (online), available from 〈https://openid.net/certification/about-conformance-suite〉 (accessed 2023–12–30).

[12] OpenID Foundation: OpenID Certification (online), available from 〈https://openid.net/certification〉 (accessed 2023–12–30).

[13] Norimatsu, T. and Nakamura, Y.: Flexible Way for Realizing OAuth 2.0 Based Security Profiles on Keycloak, *Proc. Computer Security Symposium 2020*, Online, IPSJ, pp.853–858 (2020). (in Japanese).

[14] Norimatsu, T. and Nakamura, Y.: Flexible Way for Realizing OAuth 2.0 Based Security Profiles on Keycloak (part 2), *Proc. Computer Security Symposium 2021*, Online, IPSJ, pp.639–646 (2021). (in Japanese).

[15] Norimatsu, T., Nakamura, Y. and Yamauchi, T.: Flexible Method for Supporting OAuth 2.0 Based Security Profiles in Keycloak, *Lecture Notes in Informatics (LNI) Proc. Open Identity Summit 2022*, Lyngby, Denmark, Gesellschaft für Informatik e.V., Vol.P-325, pp.87–98, DOI: 〈https://doi.org/10.18420/OID2022_07〉 (2022).

[16] Norimatsu, T., Nakamura, Y. and Yamauchi, T.: Policy-Based Method for Applying OAuth 2.0-Based Security Profiles, *IEICE Transactions on Information and Systems*, Vol.E106-D, No.9, pp.1364–1379, DOI: 〈https://doi.org/10.1587/transinf.2022icp0004〉 (2023).

[17] OpenID Foundation: OpenID Connect Client-Initiated Backchannel Authentication Flow - Core 1.0 (online), available from 〈https://openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0-final.html〉 (accessed 2023–12–30).

[18] The Open Banking Implementation Entity: The Open Banking API Specifications: Account and Transaction API Profile - v3.1.10 (online), available from 〈https://openbankinguk.github.io/read-write-api-site3/v3.1.10/profiles/account-and-transaction-api-profile〉 (accessed 2023–12–30).

[19] Philippaerts, P., Preuveneers, D. and Joosen, W.: OAuch: Exploring Security Compliance in the OAuth 2.0 Ecosystem,

*Proc. the 25th International Symposium on Research in Attacks, Intrusions and Defense*, pp.460–481 (2022).

[20] Wang, R, Chen, S. and Wang, X.: Signing Me onto Your Accounts through Facebook and Google: A Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services, *Proc. 2012 IEEE Symposium on Security and Privacy*, pp.365–379 (2012).

[21] Zhou, Y. and Evans, D.: SSOScan: Automated Testing of Web Applications for Single Sign-On Vulnerabilities, *Proc. the 23rd USENIX Security Symposium (USENIX Security 14)*, pp.495–510 (2014).

[22] Yang, R., Li, G., et al.: Model-based Security Testing: An Empirical Study on OAuth 2.0 Implementations, *Proc. the 11th ACM on Asia Conference on Computer and Communications Security*, pp.651–662 (2016).

[23] Calzavara, S., Focardi, R., et al.: WPSE: Fortifying Web Protocols via Browser-Side Security Monitoring, *Proc. the 27th USENIX Conference on Security Symposium*, pp.1493–1510 (2018).

[24] Wang, H., Zhan, Y., et al.: The Achilles heel of OAuth: a multi-platform study of OAuth-based authentication, *Proc. the 32nd Annual Conference on Computer Security Applications*, pp.167–176 (2016).

[25] Sun, S. and Beznosov, K.: The devil is in the (implementation) details: an empirical analysis of OAuth SSO systems, *Proc. the 2012 ACM Conference on Computer and Communications Security*, pp.378–390 (2012).

[26] Bai, G., Lei, J., et al.: AUTHSCAN: Automatic Extraction of Web Authentication Protocols from Implementations, *Proc. the 20th Annual Network and Distributed System Security Symposium (NDSS)*, (2013).

## Appendix

### A.1 Evidence of Our Contribution to the Conformance Test Execution Platform as Keycloak's Sub-project

The following list includes the pull-requests sent by us and merged to the conformance test execution platform as Keycloak's sub-project.

https://github.com/keycloak/kc-sig-fapi/pull/35, 36, 37, 38, 112, 123, 142, 150, 162, 170, 187, 188, 191, 199, 204, 213, 223, 224, 229, 230, 237, 241, 249, 258, 259, 260, 261, 262, 263, 264, 267, 270, 273, 275, 278, 280, 282, 284, 286, 293, 295, 299, 300, 303, 306, 307, 310, 312, 313, 314, 315, 328, 336, 338, 344, 346, 352, 354, 356, 364, 370, 377, 381, 384, 388, 395, 398, 412, 414, 416, 428, 431, 434, 436, 437, 442, 445, 458, 459, 461, 475, 477, 479, 482.

### A.2 Data Source of Keycloak's CI Run Results used for the Evaluation

The following list includes the web pages for Keycloak's CI run results used to be evaluate the conformance test execution platform in Section 5.1 and 5.3. These CI runs had been executed when pull-request to Keycloak had been sent before Keycloak 23.0.3 was released, which is the version of Keycloak used for the evaluation of the conformance test execution platform in Section 5.

https://github.com/keycloak/keycloak/actions/runs/7132499576, 7139728398, 7032155981, 7074907017, 7139832428, 7046221921, 7122667381, 7098487569, 7060027388, 7038104417.

### A.3 Data Source of Conformance Tests Execution on the GitHub Actions Workflow used for the Evaluation

The following list includes the results of conformance tests execution on the GitHub Actions workflow. The results of conformance tests execution were used to evaluate the conformance test execution platform in Section 5.3.

https://github.com/tnorimat/keycloak-fapi/actions/runs/9728965818, 9729388199, 9729784644, 9730283579, 9730726378, 9731181993, 9731638566, 9732617966, 9734024826, 9734738933.

**Takashi Norimatsu** received his B.S. degree in science from National Institution for Academic Degrees and Quality Enhancement of Higher Education and M.S. degree in mathematical engineering from University of Tsukuba in 2001. He has been working for Hitachi, Ltd. since 2001, and is also studying at Okayama University to obtain a Ph.D. degree. He is a member of IPSJ.

**Yuichi Nakamura** received his B.S. and M.S. degrees in physics from University of Tokyo in 1999 and 2001, M.S. degree in computer science from The George Washington University in 2006, and Ph.D. degree in computer science from Okayama University in 2016. He worked for Hitachi Solutions over 2001–2015 and has been working for Hitachi, Ltd. since 2016. He is a member of IPSJ.

**Toshihiro Yamauchi** received B.E., M.E. and Ph.D. degrees in computer science from Kyushu University, Japan in 1998, 2000 and 2002, respectively. In 2001, he became a Research Fellow of the Japan Society for the Promotion of Science. In 2002, he became a Research Associate in Faculty of Information Science and Electrical Engineering at Kyushu University. He has served as associate professor of Graduate School of Natural Science and Technology at Okayama University since 2005, and has been serving as professor of Graduate School of Natural Science and Technology at Okayama University since 2021. His research interests include operating systems and computer security. He is a member of IPSJ, IEICE, ACM, USENIX, and IEEE.