The Indonesian Journal of Computer Science
www.ijcs.net
Volume 14, Issue 2, April 2025
https://doi.org/10.33022/ijcs.v14i2.4854

Perancangan dan Evaluasi Keamanan Modul IAM pada Arsitektur Microservice Menggunakan *Keycloak*

Winayaka Ruhur Sandhya Pamungkas, Rizal Fathoni Aji
Winayaka.ruhur11@ui.ac.id[1], rizal@cs.ui.ac.id[1]
[1]Universitas Indonesia

Informasi Artikel

Diterima : 25 Apr 2025
Direvisi : 29 Apr 2025
Disetujui : 30 Apr 2025

Kata Kunci

Identity and Access Management, Sistem Keamanan, Microservice, Keycloak, NIST SP 800-53

Abstrak

Keamanan dan pengelolaan identitas menjadi tantangan penting dalam sistem berbasis *microservice*. Penelitian ini bertujuan untuk merancang dan mengevaluasi modul Identity and Access Management (IAM) yang aman dan terintegrasi, dengan memanfaatkan teknologi Keycloak serta pendekatan pengamanan berdasarkan kerangka kerja NIST SP 800-53. Studi kasus dilakukan pada suatu organisasi yang sedang melakukan transformasi digital ke arsitektur *microservice*. Sistem dirancang untuk mengakomodasi autentikasi dan otorisasi berbasis peran, atribut, serta *permission*. *Federeted identity* diterapkan dengan protokol CAS, OIDC, dan REST API melalui Service Provider Interfaces (SPI). khusus. Desain kemudian diuji dengan tiga jenis pengujian: *unit testing*, *integration testing*, dan *security testing*. Hasil menunjukkan bahwa sistem berjalan sebagaimana rancangannya tanpa ditemukan celah keamanan yang kritis. Penelitian ini berkontribusi kepada praktik penggunaan IAM yang aman dan fleksibel di lingkungan *microservice*.

Keywords

Identity and Access Management, System Security, Microservice, Keycloak, NIST SP 800-53

Abstract

Identity and security management are relevant concerns in microservice-based systems. The aim of this research is to model and examine a secure and unified Identity and Access Management (IAM) module founded on Keycloak and the NIST SP 800-53 security standard. A case study was conducted in organization that is undergoing digital transformation to a microservice architecture. The system offers authentication and authorization based on roles, attributes, and permissions. Identity federation is achieved via CAS, OIDC, and REST API protocols with custom Service Provider Interfaces (SPI). Testing includes unit testing, integration testing, and security testing. Results show the system functions as designed without show-stopping security vulnerabilities. This study contributes to secure and flexible IAM practices for microservice ecosystems.

A. Pendahuluan

Pada digital era ini, keamanan data sangatlah penting. Dalam sudut pandang pengembangan aplikasi, perlu adanya pembatasan akses terhadap data pada aplikasi. Hal ini dapat diatasi dengan mengimplementasi Identity and Access Management (IAM). IAM merupakan metode terstruktur untuk mengelola pengguna, autentikasi, dan kontrol akses [1].

IAM tergbagi menjadi dua, yaitu *Identity Management* dan *Access Management* [2]. *Identity Management* berfungsi sebagai proses dan aturan untuk membuat, mengelola, dan menghapus identias pengguna [3]. Selain itu, *Identity Management* juga menverifikasi identitas agar dapat mengakses data dengan aman [2]. *Access Management* berfungsi untuk memastikan data yang diakses oleh pengguna tepat dan sesuai dengan identitas dan hak akses yang diberikan [2], [3].

IAM sendiri masih memiliki banyak konsep turunan dari kedua konsep tersebut. Salah satunya adalah autentikasi. Autentukasi merupakan metode untuk melakukan verifikasi identitas pengguna [3]. Dalam melakukan verifikasi identitas pengguna, terdapat konsep yang dinamakan Federeted Identity Management (FIM) yang dapat mengelola dan memverifikasi identitas pengguna dari berbagai sistem [2]. FIM terbagi menjadi 2 bagian, yaitu identity provider dan service provider [4][5]. Identity provider bertanggung jawab dalam membuat, mengelola, dan memverifikasi identias dari pengguna. Sendagkan service provider berfungsi untuk memberikan layanan yang ke pengguna [6].

Dalam mewujudkan konsep tersebut FIM mengaplikasiakn tekonologi Single Sign On (SSO) [7]. Dengan SSO, pengguna hanya satu kali login untuk mengakses berbagai sistem [7]. Penerapan SSO memiliki beberapa keunggulan. Keunggulan yang pertama adalah mempermudah pengguna [7]. pengguna hanya menyimpan data identitasnya di satu tempat saja. Selain itu, pengguna tidak perlu melakukan authorisasi ulang Ketika mengakses aplikasi yang terdaftar sebagai penyedia layanan SSO yang sama [5]. Selain itu, dari segi keamanan, SSO memiliki potensi memperkuat keamanan dalam sistem authentikasi pada website [5]. SSO dapat mengurangi password yang lemah [7]. Pengguna tidak perlu membuat satu akun untuk tiap sistem, melainkan satu akun untuk berbagai sistem. Otomatis pengguna hanya cukup membuat satu password dengan kuat [7]. Selain itu, penerapan SSO juga memudahkan mengelola username dan password serta mengurangi beban kerja dari *help desk* [7].

Penerapan konsep IAM dengan FIM cukup fleksible. IAM dapat implementasi pada berbagai arsitektur, salah satunya microservice. Microservices merupakan arsitektur sistem yang dapat membagi sistem yang kompleks menjadi modular dan dapat berdiri sendiri [8]. Terdapat beberapa keuntungan menggunakan arsitektur *microservices*. Salah satu manfaatnya adalah sistem menjadi lebih mudah untuk dikelola [9]. Selain itu, dari sifatnya yang modular dan dapat berdiri sendiri, hal ini meningkatkan *fault tolerance* dari sistem *microservices* [10]. Hal tersebut juga dapat berpengaruh positif dalam skalabilitas dari sistem *microservices* [10]. Pada saat fase pengembangan, arsitektur ini juga memiliki keunggulan, yaitu fase pengembangan, *testing*, dan *deployment* dapat dilakukan secara modular [10].

Dari beberapa keuntungan diatas, microservice memiliki beberapa kekurangan. Salah satu kekurangan dari microservice adalah Sifat terdistribusi dan karakteristik dari sistem *microservice* membuat keamanan menjadi lebih sulit [10].

Service yang ada pada *microservice* harus dapat menjaga data dan aksesnya terhadap pengguna yang tidak berkenan.

Mengetahui situasi dan kondisi tersebut, Penelitian ini memiliki dua pertanyaan penelitian, yaitu:

- Bagaimana mendesain IAM untuk ke barbagai service dalam microservice?
- Bagaimana mendesain aplikasi IAM yang aman?

Dalam menjawab pertanyaan penelitian diatas, peneliti meneliti dari penelitian sebelumnya. Terdapat beberapa penelitian sebelumnya mengenai desain IAM dan studi kasus yang diberikan pada penelitian sebelumnya. Beberapa penelitian tersebut menitik berakan pada masalah autentikasi, akses ke sumber daya dengan Role Based Access Control (RBAC) dan Attribute Based Access Control (ABAC), dan FIM. Pada penelitian yand dilakukan oleh Rajba, Orzechowski, Rzepka, Szary, Nastaj, dan Cabaj[1], disajikan juga data pengelolaan IAM dari sisi non teknis sepeti administrasi, tatakelola, dan aturan. Selain penelitian tersebut, pada penelian Christie, Bhandar, Nakandala, Marru, Abeysinghe, Pamidighantam, dan Pierce [11] menambahkan *API Gateway* dalam rancangan arsitekturnya. Pada penelitian yang lain dipaparkan hasil kepuasan user yang mencapai 70% setelah mengimplementasikan IAM.

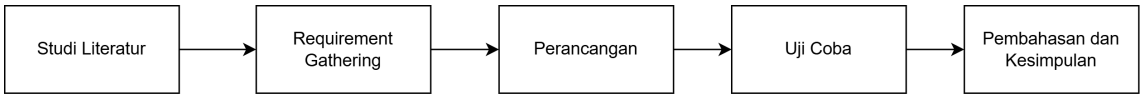
Selain penelitian mengenai desain dan studi kasus dari penerapan IAM, teradapat penelitian yang dapat memperkaya penerapan IAM dan security. Pada penelitian Nahar dan Gill [2], disebutkan terdapat delapan pola dalam IAM, yaitu Identity Manager Pattern, *Identity Federation Pattern, Local Authenticator Pattern*, *Remote Authenticator Pattern*, *Authorisation Pattern*, *Authorisation with Predicate Pattern*, RBAC *Pattern*, dan *Session Pattern*. Adapun penelitian yang dapat menjadi bahan acuan untuk pengamanan dalam konteks IAM memberikan daftar bab dan sub bab dari NIST SP 800-53 yang dapat diikuti [12].

Selain penelitian mengenai IAM, terdapat penelitian mengenai keamanan dari arsitektur *microservices* yang dilakukan adalah "An empirical study of security practices for microservices systems" yang diterbitkan dalam Journal of Systems and Software pada tahun 2023 oleh Ali Rezaei Nasab, Mojtaba Shahin, Seyed Ali Hoseyni Raviz, Peng Liang, Amir Mashmool, dan Valentina Lenarduzzi [10]. Penelitian ini melibatkan studi empiris yang dilakukan oleh para penulis untuk mengenali dan mengevaluasi praktik keamanan yang lazim diterapkan dalam sistem *microservice*. Mereka mengumpulkan data dari berbagai organisasi dan proyek yang menggunakan arsitektur *microservice*, lalu menganalisis strategi keamanan yang digunakan, kendala yang muncul, serta tingkat keberhasilan dari praktik-praktik tersebut. Pada penelitian ini, fokus utama yang dapat diambil dari penelitian ini adalah keamanan praktis untuk autentikasi dan otorisasi.

Penulisan dari penelitian ini terbagi menjadi enam bagian. Tahap pertama akan membahas mengenai latar belakang dari penelitian ini. Selanjutnya, metode penelitian yang digunakan pada penelitian ini dibahas pada tahap kedua. Dilanjutkan pada tahak ketiga yaitu hasil dan pembahasan. Penelitian ini ditutup dengan Kesimpulan dan ucapan terimakasih pada tahap keempat dan kelima.

B. Metode Penelitian

Metode penelitian yang digunakan merupakan studi kasus pada organisasi X. Adapun tahapan dalam penelitian ini dapat dilihat pada gambar 1. Penelitian ini diawali dengan studi literatur yang dilanjutkan dengan pengumpulan *requirements*. Setelah *requirements* dikumpulkan, kemudian penelitian ini dilanjutkan ke perancangan. Setelah itu, hasil dari perancangan itu kami uji coba. Proses dan hasil perancangan dan uji coba tersebut kemudian dituangkan di pembahasan dan disimpulkan pada kesimpulan.



Gambar 1. Metode Penelitian

Studi kasus pada penelitian ini dilakukan disalah satu organisasi yang mengatur pengambilan, pemrosesan, dan penjualan komoditas tertentu. organisasi ini akan membuat aplikasi *microservice* untuk menunjang kebutuhan dan proses bisnis mereka. Sebelumnya, organisasi ini tidak memiliki IAM yang terpusat. Aplikasi yang dimiliki merupakan aplikasi silo yang berdiri sendiri. Proses bisnis yang dimiliki kemudian dibagi ke berbagai modul. Untuk memastikan keamanan akses, autentikasi, dan autorisasi, maka modul IAM dibuat.

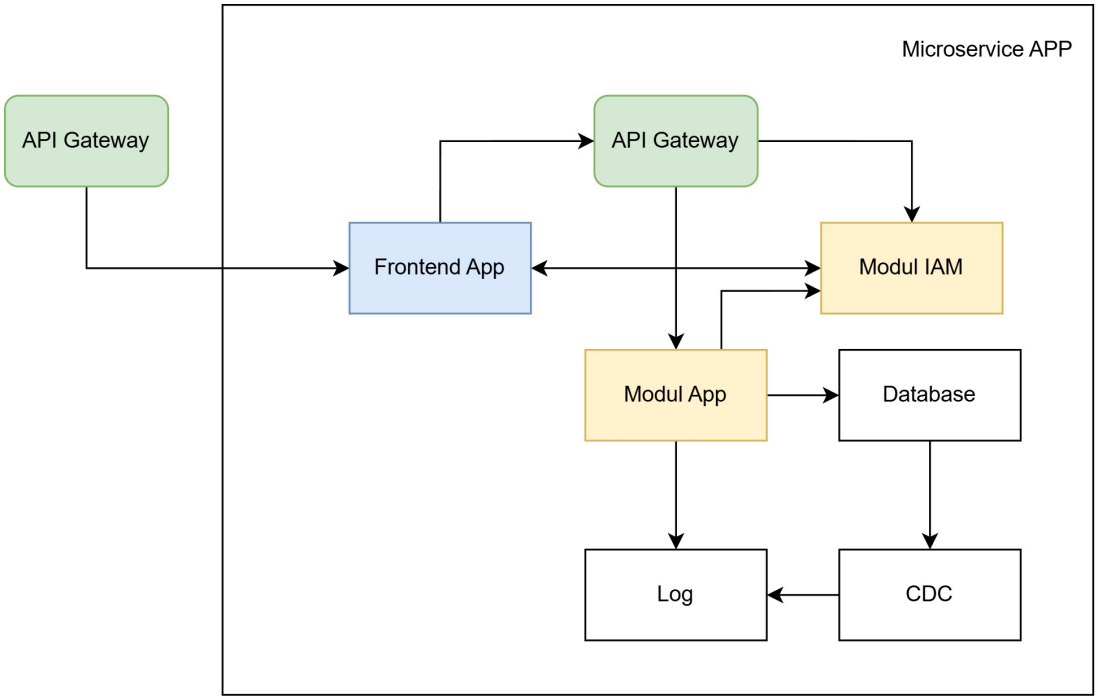
Organisasi ini memiliki dampak kerugian yang sangat besar apabila terjadi peretasan pada data. Oleh karena itu, aspek keamanan sangat diperhatikan pada penelitian ini. Peneliti telah mengidentifikasi potensi celah keamanan yang mungkin dan dipetakan dengan peringkat sepuluh teratas OWASP dan NIST SP 800-53.

C. Hasil dan Pembahasan

Terdapat tiga sub topik yang dapat dibahas pada pembahasan penelitian ini. Desain keamanan sistem, desain sistem IAM, dan hasil pengujian menjadi sub topik yang akan dibahas. Sub topik tersebut yang kemudian dapat menjawab pertanyaan penelitian.

1. Desain Keamanan Sistem

Pada Gambar 2 dapat dilihat desain arsitektur dari penelitian ini. Ada beberapa teknologi yang digunakan pada sistem di penelitian ini. Teknologi tersebut adalah Laravel untuk *backend*, Vue js untuk *frontend*, Kafka sebagai *message broker*, Debezium untuk *Change Data Capture* (CDC), Elastic Search untuk *logging*, dan Keycloak untuk IAM. Pada modul IAM, Keycloak yang digunakan dibungkus lagi dengan *frontend* dan *backend* yang sama dengan modul lain.



Gambar 2. Desain Jaringan *Microservice*

Berdasarkan desain rancangan yang dibuat pada penelitian ini, jalur keluar masuk API dikelola oleh *API gateway*. API yang dikembangkan terbagi manjadi dua dibedakan dari URLnya. API dengan URL "/internal" merupakan api yang hanya dapat diakses pada lingkungan organisasi. Jika API tidak memiliki URL tersebut maka URL tersebut dapat diakses melalui publik. Agar pengguna dapat mengakses dan menggunakan layanan yang ada, pengguna harus terlebih *login*. Terdapat tiga jenis pengguna yang dapat *login* ke sistem ini. Yang pertama adalah pengguna internal. Pengguna internal merupakan pengguna internal organisasi. Pengguna internal dapat *login* menggunakan SSO kepegawaian yang dimiliki oleh organisasi. Tipe pengguna yang kedua adalah pengguna organisasi eksternal. Tipe pengguna ini akan melakukan *login* dengan REST API. Terakhir adalah tipe pengguna eksternal. Tipe ini dapat *login* melalui formulir *login* yang ada.

Implementasi modul IAM di *deploy* menggunakan Kubernetes. Kubernetes memiliki fitur untuk melakukan replikasi. Dengan demikian, jika transaksi yang sedang berlangsung sedang banyak, Kubernetes dapat menambahkan sumber daya sehingga memiliki sumber daya yang cukup. Ketersediaan data menjadi terjamin dengan hal ini.

Dengan penerapan NIST SP 800-53 dan informasi dari penelitian sebelumnya penelitian ini dapat menjawab pertanyaan penelitian mengenai desain aplikasi IAM yang aman. Untuk mengetahui NIST apa saja yang perlu digunakan, peneliti mengacu pada peringkat sepuluh OWASP dalam keamanan aplikasi web. Dalam rangka menjaga keamanan sistem, peneliti mencoba memetakan OWASP ke NIST SP 800-53 seperti pada tabel 1.

Seperti yang dapat dilihat pada tabel 1, ada beberapa NIST yang digunakan. Dalam menangani kasus kesalahan dalam konfigurasi, peneliti mengimplementasikan *configuration management* (CM). Peneliti mengimplementasikan *environtment* untuk *development* dan *testing* dalam

pengembangan untuk pengecekan konfigurasi. Setelah itu, peneliti juga mengimplementasi *Access control* (AC) untuk mencegah masalah *broken access control*. Beberapa poin yang digunakan pada AC adalah *account maangement, access enforcement, least privilege, Disable account,* dan *security and privacy attribut.* Untuk menghadapi masalah *Security Logging and Monitoring Failures*, peneliti mengambil solusi dari System & Information Integrity (SI) dan Audit & Accountability (AU). Terakhir, permasalahan dengan login dan authentikasi dapat dicegah dengan mengimplementasikan *Identification & Authentication* (IA) pada NIST.

Tabel 1. Pemetaan Antara Masalah, OWASP, dan NIST

No	Celah Keamanan	*Kategori OWASP*	NIST SP 800-53
1	Pengaturan *public* dan *private* client	Security Misconfiguration	Configuration Management (CM)
2	Pengaturan dari permission antar client, terutama untuk *client* modul IAM	Broken Access Control	Access control (AC)
3	Pengaturan antara objek dan *permission*	Broken Access Control	Access control (AC)
4	Pengaturan pemisahan URL internal dan publik	Security Misconfiguration	Configuration Management (CM)
5	Pencurian JWT token	Insecure Design	System & Information Integrity (SI)
6	Environment variable yang diimplementasi secara *hardcoded*	Security Misconfiguration	Configuration Management (CM)
7	Upload berkas dengan format yang tidak sesuaiatau berkas yang membahayakan	Security Misconfiguration	Configuration Management (CM)
8	Pengguna dengan peran tertentu bisa mengakses data untuk peran lain	Broken Access Control	Access control (AC)
9	Pengguna dapat mengakses data organisasi lain yang bukan haknya	Broken Access Control	Access control (AC)
10	Transaksi yang terjadi pada sistem tidak tercatat	Security Logging and Monitoring Failures	Audit & Accountability (AU), System & Information Integrity
11	Kegagalan dalam *login*	Identification & Authentication Failures	Identification & Authentication (IA)

2. Desain Sistem IAM

Setelah mendesain keamanan dari aplikasi, peneliti melanjutkan untuk mendesain IAM. Setidaknya dalam desain terdapat komponen seperti autentikasi, otorisasi, dan IAM. Desain ini dibuat dengan konsiderasi proses bisnis dari tempat studi kasus dan hasil desain keamanan sistem.

Agar pengguna dapat mengakses dan menggunakan layanan yang ada, pengguna harus terlebih *login*. Terdapat tiga jenis pengguna yang akan mengakses sistem ini. Yang pertama adalah pengguna internal. Pengguna internal merupakan pengguna internal organisasi. Pengguna internal dapat pengguna menggunakan SSO kepegawaian yang dimiliki oleh organisasi. Tipe pengguna yang kedua adalah pengguna organisasi eksternal. Tipe pengguna ini akan melakukan *login* dengan REST API. Terakhir adalah tipe pengguna eksternal. Tipe ini dapat *login* melalui modul IAM yang dikembangkan.

Proyek ini menggunakan *login* SSO dengan teknologi Keycloak. SSO Keycloak memiliki layanan untuk menunjang FIM yaitu dengan *Identity Providers* dan *User Federeted*. *Identity providers* akan langsung mengarahkan pengguna untuk *login* melalui SSO yang sudah ada. SSO yang sudah ada selain bisa milik organisasi lain

bisa juga dengan sosial media seperti Google, Facebook, Github, dan lain-lain. *Identity providers* secara standar dapat dihubungkan menggunakan protokol SAML dan OIDC. Layanan *user federated*, merupakan layanan yang akan terhubung langsung ke sebuah *database* dan *directory*. Salah satu contoh penggunaannya adalah dengan menghubungkan ke Lightweight Directory Access Protocol (LDAP). Selain itu, Keycloak memberikan ruang untuk melakukan penyesuaian dengan SPI. Dengan SPI, proyek ini dapat menambahkan layanan untuk *login* SSO dengan aplikasi lain melalui protokol CAS dan REST API.

Dalam pengecekan autentikasi, izin, dan hak akses, pengguna yang sudah melakukan autentikasi akan mendapatkan JWT token. Token ini kemudian akan dicek pada berbagai modul untuk pengecekan izin dan hak akses. Pengecekan izin dan hak akses pengguna dilakukan dengan berbagai cara. Metode pertama yaitu menggunakan pengecekan berdasarkan peran atau RBAC. Peran yang dimiliki pengguna diberikan berdasarkan akses modul yang dimiliki oleh pengguna. Metode yang kedua adalah pengecekan berdasarkan atribut yang dimiliki oleh pengguna. Pengguna internal memiliki atribut kode organisasi yang dapat memberikan hak akses dan izin spesial berdasarkan kode organisasi tersebut. pengguna eksternal memiliki atribut tipe yang digunakan untuk pengecekan hak akses pada pengguna eksternal. Metode lainnya adalah pengecekan berdasarkan *permission*. *Permission* akan ditentukan dari objek yang teridentifikasi dari tiap modul. Dari objek-objek tersebut kemudian didefinisikan aksi yang diperbolehkan yang kemudian diberikan ke pengguna. Adapun metode lainnya merupakan gabungan dari kombinasi dari metode-metode sebelumnya.

Dalam melakukan implementasi, modul IAM menggunakan Keycloak versi 24. Teknologi ini menjawab berbagai kebutuhan untuk mengembangkan IAM yang baik, seperti pengelolaan identitas pengguna, pengelolaan peran dan atribut, pengelolaan kredensial, pengelolaan token JWT dan lain-lain. Pada studi kasus, terdapat kebutuhan untuk *login* dengan dua sumber *login* berbeda. Sumber pertama menggunakan protokol Central Authentication Service (CAS). Sayangnya Keycloak tidak menyediakan *Identity Provider* bawaan untuk koneksi ke protokol CAS. Oleh karena itu, peneliti perlu membuat Service Provider Interfaces (SPI) untuk penghubung protokol CAS ke Keycloak. Begitu juga dengan sumber *login* ke dua yang menggunakan REST API. Peneliti kemudian membungkus layanan REST API tersebut ke protokol OIDC yang bisa dapat terhubung ke Identity Provider Keycloak.

Dalam pengimplementasian RBAC kita menggunakan fitur yang ada dari keycloak yaitu *client role*. *Client role* merupakan yang terdefinisikan di tiap modul. Pengguna yang memiliki wewenang untuk akan diberikan peran yang bersangkutan. Untuk implementasi ABAC, ada beberapa atribut dari yang menjadi rujukan untuk pemberian akses. Untuk pengguna internal organisasi atribut yang digunakan untuk pengecekan akses adalah kode organisasi. Sedangkan untuk pengguna eksternal atribut yang digunakan untuk ABAC adalah nomor perusahaan dan kategori perusahaannya. Dalam implementasi pemberian akses dengan *permission*, peneliti melekatkan *permission* tersebut ke sebuah peran.

3. Pengujian Sistem

Terdapat tiga jenis pengujian yang dilakukan pada penelitian ini. Tiga jenis testing tersebut adalah *unit testing*, *integration testing*, dan *security testing*. *Unit*

testing dilakukan untuk pengujian unit terkecil dari sistem seperti tombol dan *form*. Pengujian selanjutnya adalah *integration testing*. IAM adalah sistem yang digunakan oleh seluruh sistem *miscroservice* lainnya. Oleh karena itu perlu dilakukan *integration testing*. Selain itu adalah *security testing*. Pengujian ini dilakukan untuk melihat kerentanan dari sistem terutama *broken access control*.

Unit testing dan *integration testing* dilakukan secara manual dengan menjalankan skenario-skenario yang telah ditentukan. Hasil dari pengujian ini menunjukkan bahwa seluruh fungsi berjalan sesuai harapan, tidak ditemukan kendala berarti seperti error atau kegagalan komunikasi antar sistem. Sementara itu, pengujian keamanan dilakukan dalam tiga tahap. Tahap pertama menggunakan fitur *scanning* otomatis dari Burp Suite untuk mendeteksi potensi kerentanan secara general dan hasilnya tidak ditemukan isu yang signifikan. Tahap kedua adalah simulasi serangan man-in-the-middle menggunakan Burp Suite untuk memantau lalu lintas data dan memastikan tidak ada informasi sensitif yang bocor. Tahap ketiga dilakukan dengan membuat kode Python khusus untuk menguji potensi *broken access control* pada beberapa endpoint. Berdasarkan hasil dari ketiga tahap pengujian keamanan tersebut, tidak ditemukan celah keamanan yang dapat dimanfaatkan untuk mengakses data secara tidak sah.

Tabel 2. Hasil Pengujian Desain

Jenis Pengujian	Jumlah Skenario / Endpoint	*Metode Pengujian*	Hasil Pengujian	Keterangan
Unit Testing	105	Manual	104 Lulus, 1 Tidak Lulus	1 gagal dengan tingkat urgensi rendah
Integration Testing	3	Manual	3 Lulus	Semua skenario integrasi berhasil
Security Testing	-	Otomatis (Burp Suite)	Tidak ditemukan celah keamanan	Pengecekkan kerentanan
Security Testing	82 *Endpoint*	Otomatis (*Script* Python)	Semua Aman	Pengujian Broken Access Control dengan skrip kustom Python

D. Kesimpulan

Pada penelitian ini, peneliti mendesain bagaimana IAM dapat digunakan pada aplikasi *microservice* dan bagaimana keamanannya. Peneliti telah mendesain modul IAM dari masalah jaringan, implementasi, FIM, *authentikasi*, dan *authorization*. Adapun dengan keamanannya peneliti telah mengidentifikasi beberapa kemungkinan kerentanan dalam aplikasi dan memetakannya ke sepuluh teratas OWASP. Dari hasil pemetaan tersebut peneliti mencari solusinya dengan memetakan ke kerangka kerja NIST SP 800-53. Hasil dari desain tersebut kemudian dilakukan uji coba dengan 3 jenis pengujian, yaitu: *unit testing, integration testing,* dan *security testing*. Dari hasil uji coba tersebut tidak ada hasil yang signifikan yang dapat membuat desain tidak dapat berjalan maupun memiliki celah keamanan yang berarti.

E. Referensi

[1] P. Rajba, N. Orzechowski, K. Rzepka, P. Szary, D. Nastaj, and K. Cabaj, "Identity and Access Management Architecture in the SILVANUS Project," *ACM International Conference Proceeding Series*, 2024, doi: 10.1145/3664476.3670935.

[2] K. Nahar and A. Q. Gill, "Integrated identity and access management metamodel and pattern system for secure enterprise architecture," *Data Knowl Eng*, vol. 140, no. June 2021, p. 102038, 2022, doi: 10.1016/j.datak.2022.102038.

[3] L. Tsoodol, M. Tuvdendorj, J. Rentsendorj, and O. E. Namsrai, "Implementing Unified Identity and Access Management System: Case Study for the National University of Mongolia," in *Proceeding - 2024 IEEE 9th International Conference on Data Science in Cyberspace, DSC 2024*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 821–824. doi: 10.1109/DSC63484.2024.00124.

[4] A. Badirova, S. Dabbaghi, F. F. Moghaddam, P. Wieder, and R. Yahyapour, "A Survey on Identity and Access Management for Cross-Domain Dynamic Users: Issues, Solutions, and Challenges," *IEEE Access*, vol. 11, no. May, pp. 61660–61679, 2023, doi: 10.1109/ACCESS.2023.3279492.

[5] F. Alaca and P. C. V. Oorschot, "Comparative Analysis and Framework Evaluating Web Single Sign-on Systems," *ACM Comput Surv*, vol. 53, no. 5, Sep. 2020, doi: 10.1145/3409452.

[6] A. Mahnamfar, K. Bicakci, and Y. Uzunay, "ROSTAM: A passwordless web single sign-on solution mitigating server breaches and integrating credential manager and federated identity systems," *Comput Secur*, vol. 139, no. December 2023, p. 103739, 2024, doi: 10.1016/j.cose.2024.103739.

[7] N. Shaikh, K. Kasat, and S. Jadhav, "Secured Authentication by Single Sign On (SSO): A Big Picture," in *3rd IEEE 2022 International Conference on Computing, Communication, and Intelligent Systems, ICCCIS 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 951–955. doi: 10.1109/ICCCIS56430.2022.10037708.

[8] M. S. Saravanan and N. Gogineni, "A New framework for microservices with single sign-on, security assertion markup language and openid connect," in *Proceedings - 2024 3rd International Conference on Sentiment Analysis and Deep Learning, ICSADL 2024*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 539–543. doi: 10.1109/ICSADL61749.2024.00094.

[9] M. Matias, E. Ferreira, N. Mateus-Coelho, O. Ribeiro, and L. Ferreira, "Evaluating Effectiveness and Security in Microservices Architecture," *Procedia Comput Sci*, vol. 237, pp. 626–636, 2024, doi: 10.1016/j.procs.2024.05.148.

[10] A. Rezaei Nasab, M. Shahin, S. A. Hoseyni Raviz, P. Liang, A. Mashmool, and V. Lenarduzzi, "An empirical study of security practices for microservices systems," *Journal of Systems and Software*, vol. 198, p. 111563, 2023, doi: 10.1016/j.jss.2022.111563.

[11] M. A. Christie *et al.*, "Managing authentication and authorization in distributed science gateway middleware," *Future Generation Computer Systems*, vol. 111, pp. 780–785, 2020, doi: 10.1016/j.future.2019.07.018.

[12] D. Pöhn, S. Seeber, T. Hanauer, J. A. Ziegler, and D. Schmitz, "Towards Improving Identity and Access Management with the IdMSecMan Process Framework," *ACM International Conference Proceeding Series*, 2021, doi: 10.1145/3465481.3470055.