Designing Secured Services for Authentication, Authorization, and Accounting of Users

Denys Shevchuk[1], Oleh Harasymchuk[1], Andrii Partyka[1], and Nataliia Korshun[2]

[1] *Lviv Polytechnic National University, 12 Stepan Bandera str., Lviv, 79013, Ukraine*
[2] *Borys Grinchenko Kyiv University, 18/2 Bulvarno-Kudriavska str., Kyiv, 04053, Ukraine*

Abstract

In this paperwork service for authentication, authorization, and user management has been designed and developed. The purpose of this work is to simplify the configuration of user security in web and mobile applications with a ready-made solution. The service can be implemented as a separate server on the Internet—you will be able to use it right after registering a client this method does not require the use of your resources, but it is less secure, especially for applications that keep vulnerable data. Still, the best and the most secure option is to download the source code and run the service on the local network. In this case, it will be only accessible to other applications or microservices in the case of microservice architecture.

Keywords

Authentication, authorization, JWT token, OAuth protocol, OpenID, RBAC, ABAC, encryption.

1. Introduction

With the rapid development of information technology even traditional things such as shopping, going to restaurants, or going to work have become available online. It is enough to have access to the Internet on your mobile phone, computer, or tablet [1]. Then you can make purchases online, read the news on a reliable website, do all the work on a laptop, and even organize a meeting with colleagues, etc. There are plenty of examples of how digitization facilitates many processes in different areas of our lives, but it also challenges the security of customers and service providers [2].

Nowadays, most web and mobile applications require the implementation of user accounting, authentication, and authorization tools [3]. In most cases, this functionality will be the same and when you are developing a new application, you'll need to do this all over again. Based on this, you can automate this process and save money and time on creating something that has already been created more than once [4].

The main goal of this paper is to create a service for the authentication, authorization, and accounting of users [5] to reduce the time for the development of such functionality and simplify the existing application. The solution can be integrated with other applications in two different ways:

- As a third-party service that is located on a separate server that accepts requests via the Internet.
- As an embedded service in your internal network which accepts the requests from the internal intranet network [6].

2. Understanding OpenID and OAuth Protocols

Developers looking to streamline logins or Identity and Access Management (IAM) in their apps have many mechanisms available to them [7]. Two common approaches are OpenID

CPITS-2023-II: Cybersecurity Providing in Information and Telecommunication Systems, October 26, 2023, Kyiv, Ukraine
EMAIL: denys.shevchuk.mkbas.2022@lpnu.ua (D. Shevchuk); garasymchuk@ukr.net (O. Harasymchuk); andrijp14@gmail.com (A. Partyka); n.korshun@kubg.edu.ua (N. Korshun)
ORCID: 0009-0004-6881-6847 (D. Shevchuk); 0000-0002-8742-8872 (O. Harasymchuk); 0000-0003-3037-8373 (A. Partyka); 0000-0003-2908-970X (N. Korshun)

Connect and Open Authorization, which are protocols and standards that work together in similar, complementary ways [8].

OpenID and OAuth provide authentication and authorization, respectively. While OAuth offers a foundation of authorization, it doesn't concern itself with authentication at all. OpenID picks up where OAuth leaves off, adding authentication functionalities to your application [9].

OpenID Connect (OIDC) is an authentication protocol i.e. an identity layer built upon OAuth 2.0 [10]. It allows Relying Parties (RP) such as apps and websites to identify if a user is who they claim to be. It does this by prompting a login from a separate, trusted platform. That second platform is an OIDC Provider or Identity Provider that authenticates the user's identity through a proprietary method that meets industry standards. Upon confirmation of the user's identity, the OIDC Provider generates an ID token for the RP to use, without disclosing the user's credentials to the RP. This allows for seamless authentication for the end user, who doesn't have to create and memorize any additional credentials [11].

Open Authorization (OAuth) is a method for authorizing access between apps. It enables a secondary application to access and perform certain functions within your app. By generating access tokens from the secondary app or website, it offloads the work of authentication and simply grants access.

The way OAuth works is similar to OIDC at a surface level. It allows for communication between two apps or websites. Where it differs from OIDC is in its lack of authentication of the user [12].

OAuth provides a foundation in authorization, allowing other apps to access yours. And OpenID builds upon that, adding the ability to authenticate user identities. Together, they make all elements of login and IAM much simpler and safer on the user side. Both OIDC and OAuth offer benefits in a wide variety of use cases, and many apps implement both of them [13].

The main difference between OAuth2 and OpenID Connect is that OAuth2 is only concerned with authorization, while OpenID Connect is also concerned with authentication. Authorization means granting access to resources, while authentication means verifying the identity of a user [14].

The problem is that both protocols can be used to accomplish similar tasks, but that doesn't mean they should be used interchangeably. OpenID provides an identity assertion while OAuth is more generic. When a client uses OAuth, a server issues an access token to a third party, the token is used to access a protected resource, and the source validates the token [15]. Notice, that at no point is the identity of the owner of the token verified.

You can think of a token issued by a resource server like it's a ticket to a movie. Nowhere on the ticket does it say any identifying information about an individual, it simply is used as a way of saying I have permission to enter. This means that the issued token may be in the hands of a bad actor or a machine [16].

That is why the service offered below allows you to implement authentication and authorization mechanisms at the same time.

3. Web Service Design for Authentication, Authorization, and Accounting of Users

3.1. Vulnerability Analysis and Authentication Mechanism Design

At a conceptual level, authentication vulnerabilities are among the most understandable, but they can cause some of the most critical damage because of the obvious connection between authentication and security. Most authentication threats are associated with passwords or password-based authentication methods. But broken authentication is also the cause of many vulnerabilities [17]. Broken authentication occurs when the implemented authentication process is corrupted. Unfortunately, this is usually very difficult to investigate in time and can lead to even greater password risks [18–19].

Common broken authentication-related vulnerabilities:

- Vulnerable authentication logic.
- Weak user profile or password recovery process.
- Vulnerable library usage for authentication.
- Unsecured session.
- No blocking and no limit on the number of login attempts.
- Insecure password verification methods.

- Weak password security policies.

The suggested authentication mechanism provides several possibilities at once, which allows for to minimization of the occurrence of typical vulnerabilities and also allows to implementation of the least privileges principle [20].

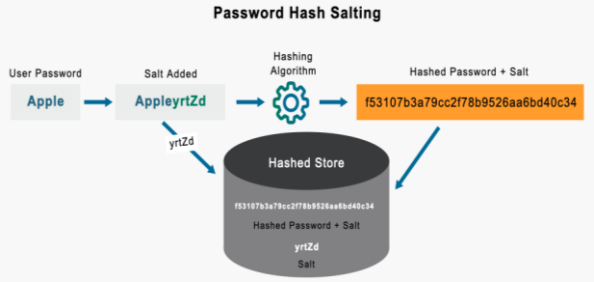

Figure 1: Password hash salting

Client and customer password-based authentication. Basic user authentication is performed using passwords, which are stored as hashes calculated using the reliable SHA-256 hashing algorithm of the SHA-2 family. Also, to counteract the selection of password hashes, each user has a so-called "salt"—a set of random characters, which is used to complicate the definition of hash functions and hashes using the dictionary attack method (Fig. 1). That is if an attacker gets the victim's password and the hash function used, he will not be able to perform a dictionary attack on other users. An additional security protection is the change of the "salt" every time the user changes the password [21].

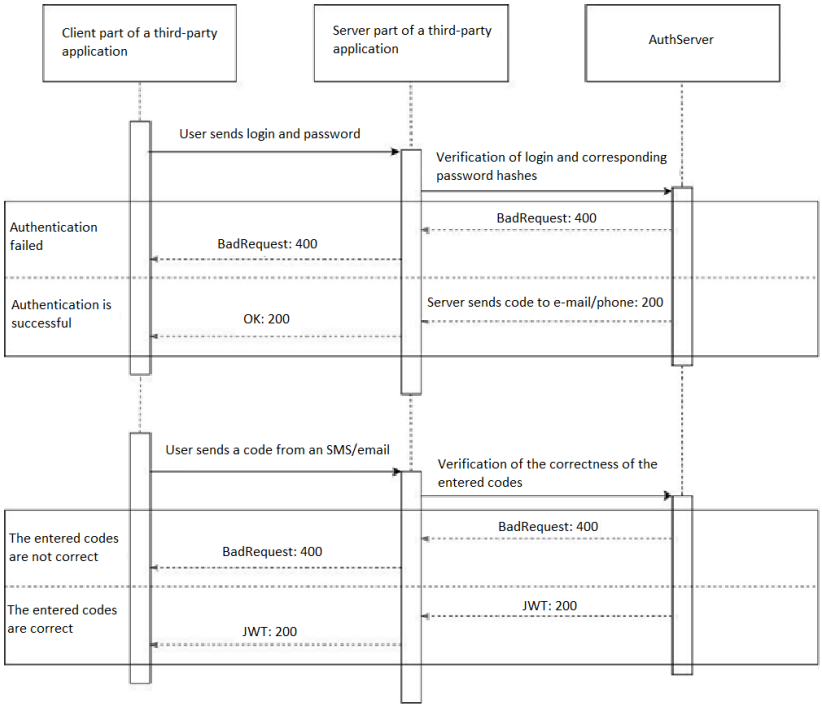

Figure 2: Multi-layer authentication scheme

Multi-layer authentication using email or phone [22]. To implement this authentication mechanism, you must first confirm the user's phone or email by entering a confirmation code that will be sent to the corresponding phone number or email address. Moreover, the code that is sent in different ways is different. It means that the user has 4 options for setting up authentication:

- Using a password.
- Using a password and confirmation with a code via an email address [23].
- Using a password and confirmation with a code via a phone number.

Using a password, confirmation with a code via an email address, and confirmation with a code via a phone number (Fig. 2).

Password reset. To use this feature, you need to have a verified email address or phone number. In this case, a pseudo-random password is generated and sent to a verified phone number or mailbox. It is worth noting that the temporary password is valid only for a short period after which if the password has not been changed the procedure must be repeated (Fig. 3).

Users profile block. This feature is designed to prevent brute-force attacks on users [24].

When trying to go through many user passwords the attacker faces the fact that after several unsuccessful attempts the user's profile is blocked, and he can no longer log in with the usual password. In this case, the user can remove the block only by entering a code that can be sent to a verified email or phone number (Fig. 4).

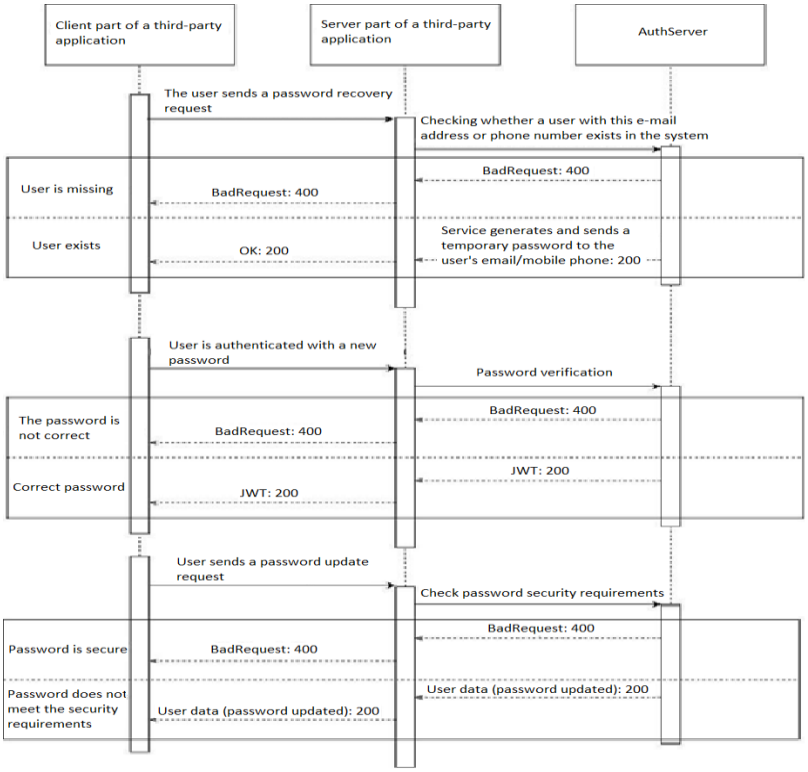

Figure 3: Reset password scheme

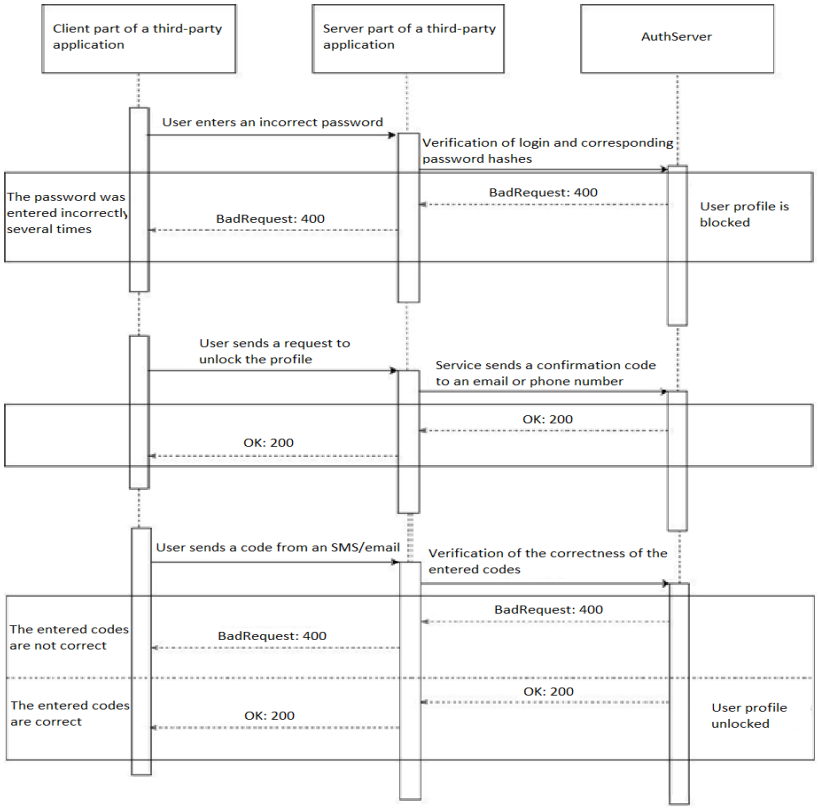

Figure 4: Blocking/unblocking user's profile scheme

Generation of JWT tokens to manage the authentication session described in Fig. 5.

Access token generation. Upon successful authentication, a pair of tokens is sent to the user: an access token and a recovery token. Using the access token, the user can make requests to resources without re-authentication as the token acts as a guarantor. Access tokens are mostly valid for a short time (for example, 15 minutes) so that when stolen, an attacker cannot pretend to be an authenticated user for a long time [25].

Refresh token generation. Also, if necessary, the recovery token can be regenerated. They are stored in the database and are personal for each user. Recovery tokens usually last for a long time, such as one week.

Access token refreshment using the refresh token. Refresh tokens are generated for authenticated users to be able to obtain a new access token after the previous one has expired without having to reenter a password every fifteen minutes to retrieve it [26].

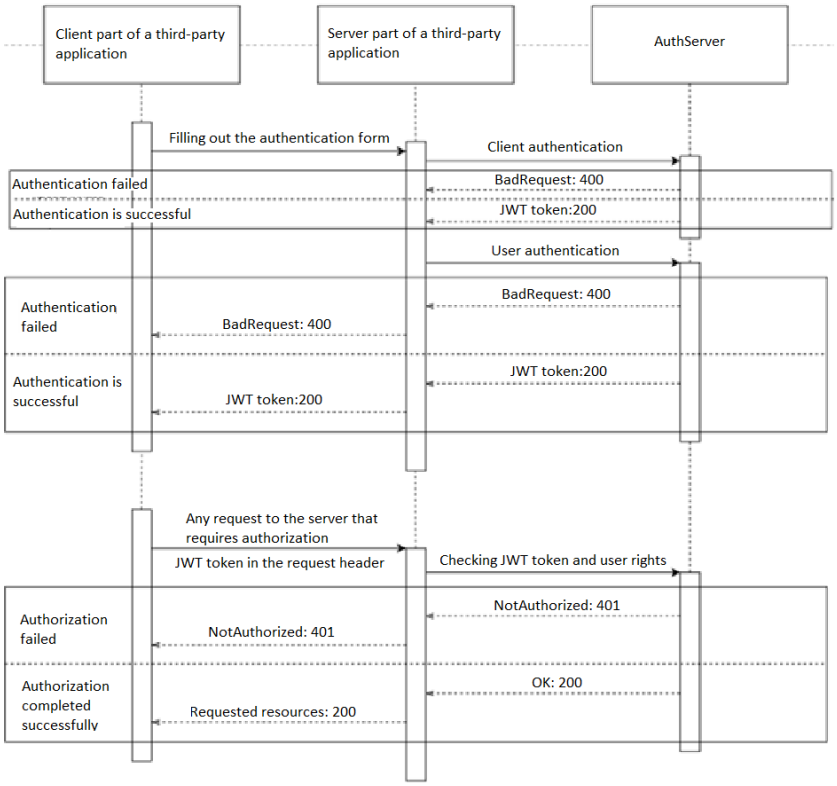

Figure 5: JWT-token generation scheme

3.2. Selection and Design of the Authorization Mechanism

Role-Based Access Control (RBAC). The service provides an opportunity to perform authorization using roles. That is, the user has a set of roles that were given to him, and the client can create his requirements for certain actions by comparing the requested roles for authorization of the action with the roles that the user has. Upon successful verification, the user will be authorized to perform a specific action [27].

Attribute Based Access Control (ABAC). Also, the service provides an opportunity to perform authorization using attributes [28]. The user has a set of attributes that were given to him and the client can create his requirements for certain actions by comparing the requested attributes to authorize the action with the attributes that the user has. Upon successful verification, the user will be authorized to perform a specific action [29].

Mixed authorization. This type of authorization expands the set of possible options for granting rights to users and allows better implementation of the least privileges rule [30]. That is, do not give the user extra rights that can lead to unpredictable consequences of the program's operation. At the same time, the check is carried out separately for roles and separately for

attributes. This division allows for a better delimitation of authorization because, for example, not everyone who can delete a user is an administrator, but every administrator should be able to delete a user.

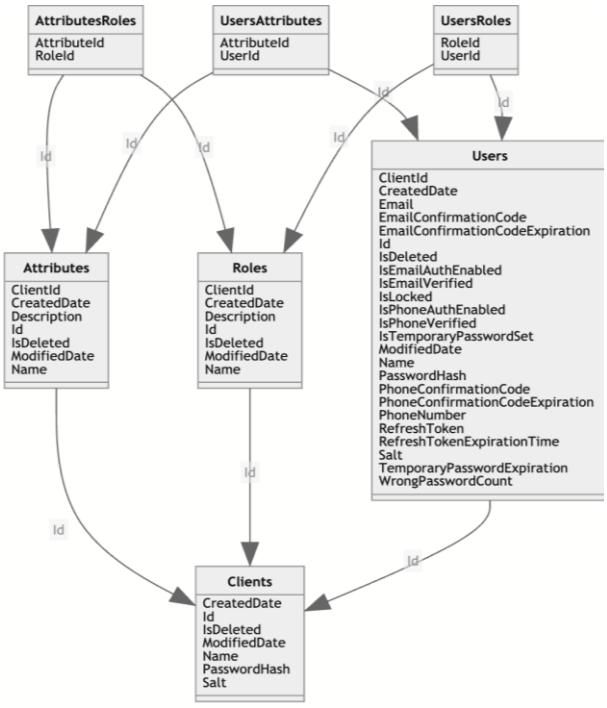

Figure 6: Database tables and relationships among them

Also, a database was designed (Fig. 6) in which all the necessary data for a service to be able to function is stored [31], namely:

1. Users—table for storing users' information (name, email, phone number, hashed password, etc.).
2. Roles—table for storing role information that can be created in a system to be able to implement an RBAC authorization.
3. Attributes—table for storing attribute information that can be created in a system to be able to implement an ABAC authorization.
4. Clients—table for storing client information.
5. UsersRoles—table for implementing many-to-many relationships between User and Role tables.
6. UsersAttributes—table for implementing many-to-many relationships between User and Attribute tables.
7. AttributesRoles—table for implementing many-to-many relationships between Attribute and Role tables [32].

During the development of the application, a code-first approach was used to create the database in which the SQL code is generated automatically based on the server language code. In this case, C# 10 was used in conjunction with the Entity Framework Core data access tool [33].

The entity relationship diagram is presented below (Fig. 7):

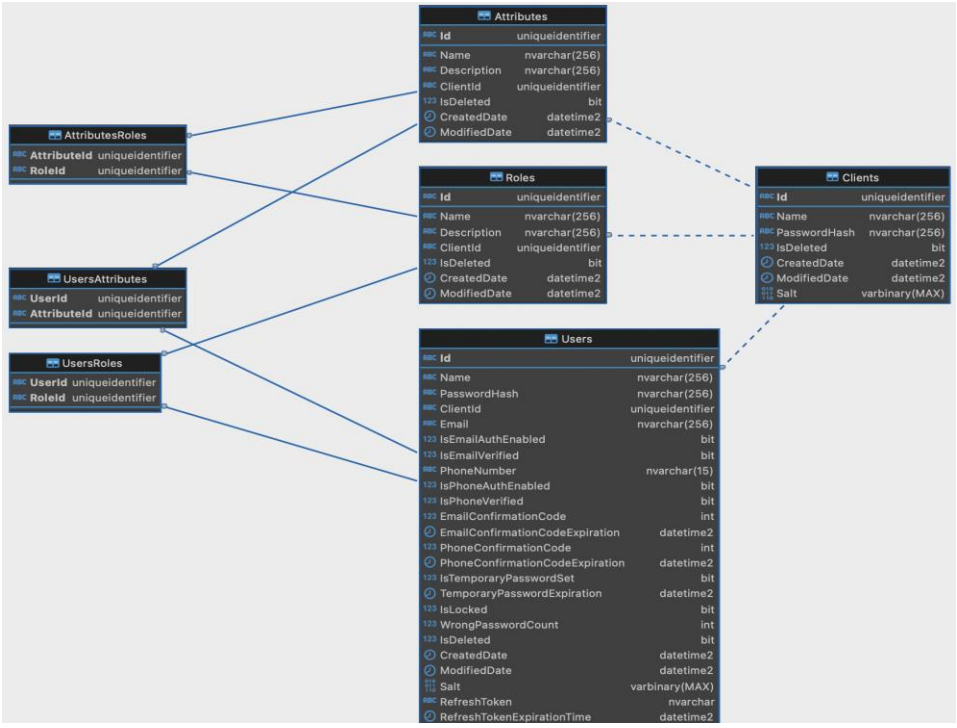

Figure 7: Entity Relation database diagram

4. Service Features Description

Password-based client and user authentication. In the first stage, it is checked whether the user exists at all and, if so, whether his profile is blocked and whether the temporary password is valid if it was enabled. Then their hashes are checked. The password entered by the user is hashed using the same hash function (SHA-256) and using the same "salt" that is stored in the database and is unique per each user. If more than 5 unsuccessful login attempts are made the user profile will be blocked.

Confirmation code generation. The confirmation code that will be sent via e-mail or SMS is selected as follows: a pseudo-random number between 100000 and 999999 is generated. That is any six-digit number. These are pseudo-random numbers that are an important element in solving cybersecurity problems [34].

Password reset. When choosing the password recovery option, the service generates a new secure pseudo-random 20-character password. A pseudo-random sequence of bytes is also generated which acts as a "salt" for hashing the password.

JWT-token generation. JWT tokens are generated using the asymmetric RSA encryption algorithm, the key size is 256 bits. Generation sequence:

- The private key is extracted from the settings and an RSA object is created.
- A new signature is created for the JWT token.
- The token itself is created and signed with the corresponding key.

JWT-token validation. To validate the token service validates the following: the lifetime of the token, the signature key, and the issuer.

Refresh token generation. When a recovery token is created, a pseudo-random sequence of bits is generated, which is stored in the database and sent to the user in response to successful authentication.

Obtaining a new access token using a recovery token. The recovery token can be regenerated by sending a corresponding request. Then, with its help, you can update the access token without filling out the authentication form again. You just need to send a request to update the access token, usually after its expiration date, by filling out the appropriate form.

Processing of JWT token in HTTP request. Before any request that requires the user to be authenticated, a check is made to see if the token is present in the request header and if it is, the data from the token is retrieved and stored in a temporary storage for the request duration.

When RBAC is performed the required roles for authorization and the user roles are compared.

When ABAC is performed the required attributes for authorization and the user attributes are compared.

Authorization attribute. The authorization attribute checks whether the request sent to the service is authorized, i.e. it checks whether the JWT token was successfully verified and the client/user data were placed in the temporary storage [35].

The following tools were used to send messages to the phone and email:

- For sending SMS the service was integrated with the Twilio [36].
- for sending emails the service was integrated with the SendGrid [37].

In both tools, a service user profile was created and configured and service integration was configured using the NuGet package.

5. Conclusion

The selection of the type of the correct authentication and authorization method depends on the task we're performing. It's appropriate when we're working on a new system deployment to test some alternative mechanisms and see which one leads to better protection from malicious attacks.

OAuth and OIDC are both important protocols for developers to understand, with OAuth providing authorization and OIDC providing authentication and identity management. By understanding the differences between these two protocols, developers can make informed decisions on how to go about building authentication and integrating with third-party services.

The paper considers the issue of creating a secure service for the authentication and authorization of users, which can be implemented as an independent or built-in

service. In the first case, the service can be placed on a separate secure server with limited access, to which requests are sent via the Internet. The second option of using this service is in an internal controlled network, requests to which are sent exclusively in the internal intranet network.

The service provides various options for the authentication mechanism which allows you to implement various scenarios depending on security requirements. The service also provides protection against brute-force attacks, the ability to reset the password, and restore the blocked user's profile.

Both the RBAC and ABAC mechanisms were used in the design of the service for the authorization process. The ability to use mixed authorization allows you to expand the capabilities of RBAC authorization, which is usually suitable for implementing simple business rules, with the ABAC approach, which supports data filtering and rules with dynamic parameters.

References

[1] M. Repetto, A. Carrega G. Lamanna, An Architecture to Manage Security Services for Cloud Applications, 4th Int. Conf. Comput. Commun. Secur. (2019) 1–8.

[2] A. Bissada, A. Olmsted, Mobile Multi-Factor Authentication, 12th Int. Conf. Internet Technol. Secur. Trans. (2017) 210–211. doi: 10.23919/ICITST.2017.8356383.

[3] Y. Sadykov, et al., Technology of Location Hiding by Spoofing the Mobile Operator IP Address, in: IEEE International Conference on Information and Telecommunication Technologies and Radio Electronics (2021) 22–25. doi: 10.1109/UkrMiCo52950.2021.9716700

[4] G. Grieco, et al., Authentication and Authorization in Cyber-Security Frameworks: a Novel Approach for Securing Digital Service Chains, IEEE 8th Int. Conf. Netw. Softwarization (2022) 468–473. doi: 10.1109/NetSoft54395.2022.9844030.

[5] A. Froehlich, What are the most common digital authentication methods? (2021). URL: https://www.techtarget.com/searchsecurity/answer/What-are-the-most-common-digital-authentication-methods?

[6] R. Sheldon, S. Shea, BYOI (Bring Your Own Identity) (2023). URL: https://www.techtarget.com/searchsecurity/definition/BYOI-bring-your-own-identity

[7] V. Grechaninov, et al., Decentralized Access Demarcation System Construction in Situational Center Network, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems II, vol. 3188, no. 2 (2022) 197–206.

[8] S. Yevseiev, et al., Modeling of Security Systems for Critical Infrastructure Facilities, TECHNOLOGY CENTER, (2022). doi: 10.15587/978-617-7319-57-2.

[9] Authentication vs. Authorization (2023). URL: https://auth0.com/docs/get-started/identity-fundamentals/authentication-and-authorization

[10] M. TajDini, et al., Brainwave-based Authentication using Features Fusion, Comput. Secur. 129, no. 103198 (2023) 1–11. doi:10.1016/j.cose.2023.103198

[11] What is the Difference Between Authentication and Authorization? (2023). URL: https://www.sailpoint.com/identity-library/difference-between-authentication-and-authorization/

[12] J. Coutinho, OIDC vs OAuth (2023). URL: https://supertokens.com/blog/oauth-vs-oidc

[13] Y. Li, et al., A Lightweight Identity-Based Authentication Protocol, IEEE Int. Conf. Signal Processing Commun. Comput. (2013) 1–4. doi: 10.1109/ICSPCC.2013.6664134.

[14] D. Kelley, Federate and Secure Identities With Enterprise BYOI (2021). URL: https://www.techtarget.com/searchsecurity/tip/Federate-and-secure-identities-with-enterprise-BYOI

[15] S.-H. Kim, S.-H. Kim, General Authentication Scheme in User-Centric IdM, 18th Int. Conf. Adv. Commun. Technol. (2016) 737–740, doi: 10.1109/ICACT.2016.7423540.

[16] H. Singhal and A. Kar, Information Security Concerns in Digital Services: Literature Review and a Multi-Stakeholder Approach, Int. Conf. Adv.

Comput. Commun. Inf. (2015) 901–906. doi:10.1109/icacci.2015.7275725.
[17] S. Shah, Most Common Authorization Vulnerabilities (2022). URL: https://goteleport.com/blog/authorization-vulnerabilities/
[18] H. Hulak, et al. Formation of Requirements for the Electronic Record-Book in Guaranteed Information Systems of Distance Learning, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, CPITS 2021, vol. 2923 (2021) 137–142.
[19] B. Bebeshko, et al., Application of Game Theory, Fuzzy Logic and Neural Networks for Assessing Risks and Forecasting Rates of Digital Currency, Journal of Theoretical and Applied Information Technology 100(24) (2022) 7390–7404.
[20] V. Susukailo, I. Opirskyy, S. Vasylyshyn, Analysis of the Attack Vectors Used by Threat Actors During the Pandemic, IEEE 15th Int. Scientific Tech. Conf. Comput. Sci. Inf. Technol. (2020) 261–264. doi: 10.1109/csit49958.2020.9321897.
[21] D. Sutkowski, P. Czernicki, The Benefits of Self-Sovereign Identity Authentication using Blockchain (2022). URL: https://softwaremind.com/blog/the-benefits-of-self-sovereign-identity-authentication-using-blockchain/
[22] A. Hassan, M. Emam, Additional Authentication and Authorization using Registered Email-ID for Cloud Computing, Int. J. Soft Comput. Eng. 3(2) (2013) 110–113.
[23] P.-L. Chen, J.-H. Yang, C.-I. Lin, ID-Based User Authentication Scheme for Cloud Computing, J. Electron. Sci. Technol. 11(2) (2013) 221–224.
[24] I. Gordin, A. Graur, A. Potorac, Two-Factor Authentication Framework for Private Cloud, 23rd Int. Conf. Syst. Theory Control Comput. (2019) 255–259, doi: 10.1109/ICSTCC.2019.8885460.
[25] G. Zhao, et al., Asynchronous Challenge-Response Authentication Solution Based on Smart Card in Cloud Environment, 2nd Int. Conf. Inf. Sci. Control Eng. (2015) 156–159. doi: 10.1109/ICISCE.2015.42.
[26] H. Dinesha, V. Agrawal, Multi-Level Authentication Technique for Accessing Cloud Services, Int. Conf. Comput. Commun. Appl. (2012) 1–4. doi: 10.1109/ICCCA.2012.6179130.
[27] HTTP Authorization, (2023). URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Authorization
[28] Y. Wang, J. Liu, An Attribute-Based Statistic Model for Privacy Impact Assessment, Int. Conf. Collaboration Technol. Syst. (2016) 619–621. doi: 10.1109/cts.2016.0117.
[29] N. Dan, et al., Attribute Based Access Control (Abac)-Based Cross-Domain Access Control in Service-Oriented Architecture (soa), Int. Conf. Comput. Sci. Serv. Syst. (2012)1405–1408.
[30] A. Gupta, A. Sharma, Authentication & Authorization, Int. J. Eng. Res. Technol. 5(3) (2017).
[31] SQL Server Technical Documentation (2023). URL: https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15
[32] V. Buriachok, et al., Invasion Detection Model using Two-Stage Criterion of Detection of Network Anomalies, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, vol. 2746 (2020) 23–32.
[33] C# Documentation (2023). URL: https://learn.microsoft.com/en-us/dotnet/csharp/
[34] V. Maksymovych, Combined Pseudo-Random Sequence Generator for Cybersecurity. Sensors 22(24) (2022). doi: 10.3390/s22249700.
[35] R. Housley, B. Aboba, Guidance for Authentication, Authorization, and Accounting (AAA) Key Management (2007). URL: https://datatracker.ietf.org/doc/html/rfc4962
[36] Twilio, Programmable Messaging API Overview (2023). URL: https://www.twilio.com/docs/sms/api
[37] Sendgrid Knowledge Center (2023). URL: https://docs.sendgrid.com/