Implementation of two-factor user authentication in computer systems

Mihailo D. Tomić[a], *Olivera* M. Radojević[b]

[a] Codebehind doo, Belgrade, Republic of Serbia,
e-mail: mihailo.tomic@codebehind.rs, **corresponding author**,
ORCID iD: https://orcid.org/0009-0003-0581-8776

[b] Serbian Armed Forces, Air Force and Air Defence,
210th Signal Battalion, Belgrade, Republic of Serbia,
e-mail: oliveradojevic@gmail.com,
ORCID iD: https://orcid.org/0009-0000-0977-5396

DOI: https://doi.org/10.5937/vojtehg72-48081

FIELD: computer sciences, IT
ARTICLE TYPE: original scientific paper

Abstract:

Introduction/purpose: The paper explores the implementation of two-factor authentication (2FA) in computer systems, addressing the increasing need for enhanced security. It highlights the vulnerabilities of password-based authentication and emphasizes the advantages of 2FA in mitigating digital threats. The development of the VoiceAuth application, integrating 2FA through a combination of password and voice authentication, serves as a practical illustration.

Methods: The research adopts a three-tier architecture for the VoiceAuth application, encompassing a database, server-side REST API, and client-side single-page application. Speaker verification is employed for voice authentication, analyzing elements like pitch, rhythm, and vocal tract shapes.The paper also discusses possibilities for future upgrades, suggesting enhancements such as real-time voice verification and additional 2FA methods.

Results: The application's implementation involves a detailed breakdown of the REST API architecture, Single Page Applications (SPAs), and the Speaker Verification service.

Conclusion: The research underscores the crucial role of two-factor authentication (2FA) in bolstering the security of computer systems. The VoiceAuth application serves as a practical demonstration, showcasing the successful integration of 2FA through a combination of password and voice authentication. The modular architecture of the application allows for potential upgrades.

Key words: authentication, computer systems, biometrics.

Introduction

With the development of Internet technologies and their role in our everyday lives, the need for security and data protection has become a necessity. In a time when sensitive transactions can be conducted and personal and financial information can be accessed over the Internet or other computer networks, the possibility of secure access to data becomes crucial. Authentication in computer systems represents the confirmation of a user's identity before allowing access to the system or data. This involves verifying and confirming the authenticity of associated information to prevent unauthorized access and protect sensitive data.

Classical computer systems often used password-based authentication methods, but the drawbacks of such methods are becoming more pronounced in modern systems. Passwords are vulnerable to brute-force attacks, as modern high-performance computers can quickly uncover them if they are not sufficiently complex. Users of computer systems often use the same passwords for all systems, so in the case of data leaks from one popular service, an attacker can gain access to all user accounts. For these reasons, the need for an additional layer of security during data and service access is highly emphasized today.

Two-factor authentication provides that extra layer of security by requiring an additional proof of the user's authenticity in addition to the username and password. The significance of two-factor authentication is evident in the fact that major companies like Google and Meta require their users to use two-factor authentication in the future to protect against data breaches.

The concept of authentication

Authentication represents the process of verifying and confirming the identity of a user or system in computers and information systems. This security measure is applied to ensure access only to authorized individuals and to prevent unauthorized access and misuse.

1) Authentication is the most crucial step in the process of gaining access to data, consisting of three main steps:
2) Identification: During this step, the user asserts who they are.
3) Authentication: During this step, the user provides evidence that confirms their identity.
4) Authorization: During this step, it is determined whether the user has the right to access.

In most computer systems, identification and authentication are combined into a single step, where the user sequentially enters a username for identification and a password as an authentication factor. The combination of a username and password is the most common way to prove identity, especially in web environments (Marky et al, 2022).

Vulnerabilities of password-based authentication

Despite the widespread use of passwords as an authentication method, they are by no means secure and have multiple vulnerabilities that can easily lead to identity theft if exploited (Tot et al, 2021). The vulnerabilities of passwords do not lie solely in the authentication method itself when considered in isolation. In theory, a sufficiently long password that combines letters with numbers and other characters poses a significant barrier to any brute-force attack (Bondarchuk et al, 2023).

It is clear that a sufficiently complex password serves as a significant barrier against brute-force attacks, but a small number of passwords fall into this category. The vulnerabilities of passwords lie in their widespread use as well as in users' bad habits. Some weaknesses of passwords in the context of users include:

- Users choose easily memorable passwords such as "12345678," "11111111," or "password." Attackers can guess such passwords manually, and any short passwords are quickly guessed by computers.
- Users do not want to enter passwords with each login and carelessly use the browser's save password options.
- Users use the same passwords for accounts on different services. This practice leads to identity theft, even in the case of very complex passwords. Almost all major platforms have reported data breaches in the past few years.

Recommendation: It is advised to use different passwords, especially for systems authorized to perform financial transactions.

Two-Factor Authentication as a method of protection

Two-factor authentication involves adding another layer of security during identity verification. It commonly combines a password method with another method, but any of the two methods used can be classified into one of the following three groups:

- Something the user knows: This could be a password or some form of code or response to a question.

- Something the user possesses: often a hardware device like a mobile phone or payment card.
- Something the user is: Typically, a biometric data point such as a fingerprint or eye biometrics.

By combining two factors from different groups, the likelihood of unauthorized access is significantly reduced, as the chances are lower that an attacker possesses both factors, especially if they are not in any way related to each other (Chandrakar & Om, 2015).

Common methods of two-factor authentication

Today, various types of two-factor authentication are in use, and some of the decisive factors in their prevalence are speed of use, ease of setup, and security.

1) Authentication Using SMS messages - Two-factor authentication via SMS. After receiving the username and password, the website sends the user a one-time code (OTP) via text message. The user must enter the OTP code into the application to gain access (Yuan, 2013).

2) Software Tokens - A recommended alternative to SMS codes involves using software-generated time-based one-time codes (known as TOTP). First, the user needs to download and install a free two-factor authentication app on their mobile phone or computer. They can then use the app with any website supporting this type of authentication. Software tokens eliminate the possibility of interception by hackers, addressing a concern with methods delivered via SMS (Reese et al, 2019).

3) Push Notification Method - The push notification is an authentication request sent to the mobile phone, and the user approves or rejects it with a single touch. Due to its speed of use, Push Notification is considered a two-factor authentication method that prioritizes user convenience.

4) Biometric Methods - In the past, the obstacle to biometrics was the high cost of devices capable of recognizing biometric data. Today, biometric authentication has become commonplace for most people. Most smartphones now have fingerprint or facial recognition capabilities. However, some

biometric methods have emerged but are not widely used, such as voice authentication, gait analysis, and typing habits. Biometric authentication has several drawbacks. All characteristics it relies on are permanent user traits. Another significant problem is that biometric data is extremely sensitive and allows not only user authentication but also person identification. Therefore, the collection and transmission of this data to digital services should be treated with exceptional care. Hence, biometric data is usually used for local authentication and stored and processed on the device to avoid sending it over the network (Kaur & Kumar, 2021).

Development of the VoiceAuth Application

The practical part of this work encompasses the development of a test application that applies the concepts of two-factor user authentication. An application has been created that allows users to authenticate through a combination of password and voice authentication. Since the user's voice has been chosen as the second authentication factor, the application has been named VoiceAuth. The application was developed in a three-tier architecture with a separate data store, server-side, and client-side web application. The database was created using the PostgreSQL object-relational DBMS, allowing data manipulation through enhanced SQL commands. The server-side web application was created as a REST API service and written in the TypeScript programming language, using the NestJs framework that enables the creation of modular web applications. The client-side application was built as a single-page application (SPA) in TypeScript using the Angular framework, which possesses a similar modular architecture to the NestJs framework. The client application is tailored for use on both mobile and desktop devices. For the purpose of voice verification, a service was developed in the Python programming language, communicating with the server application via the HTTP protocol (Zou et al, 2021).

REST API Architecture

The REST API (Representational State Transfer Application Programming Interface) represents a standard way of communication and data exchange between various software applications and systems over the Internet. This technology enables simple and efficient interaction between client applications and servers, providing a means for requesting, updating, and sharing data in a manner consistent with the fundamental principles of web architecture.

In REST API architecture, resources (such as data or functionalities) are represented as URLs, and operations are performed using HTTP methods like GET (retrieve data), POST (send new data), PUT (update existing data), and DELETE (delete data). This allows programmers to easily communicate with servers and perform various actions based on their needs.

REST API architecture is widely applied in various domains, including web applications, mobile applications, and data exchange services, and it facilitates the integration of different software systems in a simple and cost-effective manner. This popular technology provides a modern approach to the development, integration, and communication between web applications.

The complete code written for the REST API of the VoiceAuth application can be found at (Tomić, 2023a).

Single Page Applications (SPAs)

A single-page application (SPA) is a type of web application that runs on a single web page without the need to redirect to new pages when communicating with the server or processing user actions. Instead, SPAs dynamically change the content and structure of the page using asynchronous communication with the server and exchanging data with the server using technologies such as AJAX (asynchronous JavaScript and XML).

In SPA applications, one page typically contains components that are dynamically loaded and displayed, achieving faster responsiveness and less server load. Users can interactively work with the application without the need to refresh the entire page.

This concept has become popular in recent years due to its user-friendliness and quick development. Well-known web frameworks and libraries such as Angular, React, and Vue.js are often used for developing SPA applications.

The complete code written for the SPA VoiceAuth application can be found at (Tomić, 2023b).

Speaker verification

Speaker verification is a biometric authentication method used to confirm the identity of a person based on their voice. This technology is employed to determine whether a person's voice matches a previously registered voice in the system (Zhu et al, 2020).

During speaker verification, the system analyzes various aspects of the voice, such as pitch, rhythm, vocal tract shapes, and other characteristic parameters. These data are compared with a voice sample registered earlier in the system. If the system determines that the analyzed voice matches the registered sample, it is considered a successful verification, and the user is granted access.

There are several types of speaker verification, but the most common distinction is between text-dependent and text-independent methods. In text-dependent verification, the user is required to read words from the screen, and such methods often involve speech recognition in addition to the speaker verification system.

Speaker verification methods commonly utilize machine learning algorithms and neural networks.

The complete code written for the Speaker Verification service in the VoiceAuth application can be found at (Tomić, 2023c).

Authentication implementation techniques in web applications

In the previous chapters, we talked about what authentication is and which authentication methods exist, but we did not explain how authentication mechanisms are implemented in different systems. In this chapter, we will look at the most common techniques for implementing authentication in web applications.

In the previous chapters, it was mentioned that authentication is the process of verifying and confirming the identity of a user using some form of proof, often referred to as credentials. In modern systems, users do not provide their credentials for every action they perform; instead, they provide them to the system during login, and for a certain period, they are considered authorized to work with the system. During this time, the system is responsible for keeping information about the logged-in user, and this is also called maintaining a session. Due to the nature of the HTTP protocol, which does not imply session persistence, there are various techniques in web applications to achieve this, and we will now look at some of them.

Basic HTTP authentication

This is the simplest method of session persistence implementation. The client-side of the application is responsible for storing user credentials in memory and sending them to the server during each HTTP request in the authorization header, usually encoded in Base64 format.

This session persistence method is not suitable for use because the user's credentials are sent with every request, making it susceptible to theft and identity loss.

Session cookie

An HTTP cookie is a block of data created by a web server when a user visits a website and is stored on the user's computer or other device through a web browser.

A session cookie is a cookie that lasts for the duration of a session. The session begins when the user logs in to work. Session cookies contain information stored in temporary memory or on the server's disk and are deleted after the session ends. A session cookie is tied to a specific server. This means that a session cookie cannot be read or accessed by any machine other than the one that generated it. The corresponding server is the one that stores the web application the user is visiting. The same server also creates the session ID. The session ID is a unique, randomly generated number stored in the session cookie.

A session cookie is suitable for use in applications that use the MVC architecture because most frameworks have built-in support for server sessions. Still, it is not suitable for use in implementing REST API solutions because one of the principles of the REST methodology is not to store session data on the server. This way, servers can scale horizontally, with multiple instances of web servers each able to handle any HTTP request.

JSON Web Token (JWT)

JSON Web Token (JWT) is a standard used to exchange information between two entities, usually a client and a server (Jones et al, 2015). JWTs contain JSON-format objects that carry information to be shared (). Each JWT is signed using hashing functions or asymmetric cryptography algorithms to ensure that the contents (known as JWT claims) cannot be altered by the client or a malicious party. JWT can take the form of JWS (JSON Web Signature) or JWE (JSON Web Encryption). JWS tokens contain public information that anyone can read but whose contents are verified by the server that issues them. JWE uses asymmetric cryptography algorithms to encrypt messages, allowing them to contain private information.

Commonly, JWT has three parts:

- Header, which contains information about the algorithm used for signing or encrypting and information about the token type.
- Payload, the part containing claims that are verified. It also includes the expiration period of the token.
- Signature, in the case of JWS, contains the cryptographic digest of the associated header, claims, and a secret key stored on the server.

JWT is suitable for implementation in the REST API architecture. The token is stored on the client-side of the application (whether it is a SPA, mobile, or desktop application) and is attached to each request in the authorization header. The token is then verified on the server by creating a new message digest and compared with the one in the token's signature. Only the server can issue and verify tokens.

Analysis of the proposed solution

The VoiceAuth application was developed as a test application to review data on employees' working hours and days off in the company (Tomić, 2023d). The data in the application is randomly generated, but in a real system, it would be retrieved from a third-party system, whether it is an embedded check-in system, a time-tracking application for projects like Clockify, or an accounting program. The application distinguishes two types of users: regular users and administrators, who have different rights and different login systems for work.

The best way to represent the structure of the application is by using UML diagrams. This is a way to visualize a software program through a collection of diagrams. Unified Modeling Language (UML) is a standardized modeling language that helps programmers visualize, construct, and document new software systems and drawings. UML is used to create diagrams of static structure based on various engineering practices that have proven successful in creating complex systems. There are different types of UML diagrams used for different software structures. There are 14 official types, with some additional versions that are not officially recognized but are widely used. In this work, two diagrams will be described: the Class Diagram and the Use Case Diagram.

Use Case Diagram

The Use Case Diagram is a representation of the interaction between the user and the system, illustrating the relationship between the user and different use cases in which the user is involved.

The use case diagram can identify different types of users in the system and different use cases.

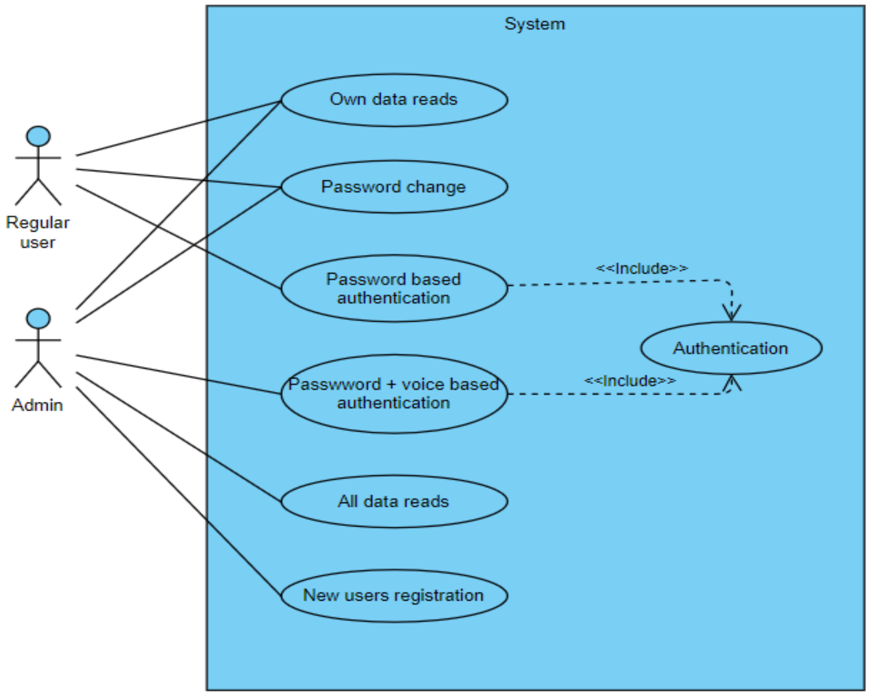


Figure 1 – Use Case Diagram

Fig. 1 shows the use case diagram for the application described in the paper.

The first step is user login. Regular users use only their username and password, while administrators, after logging in with their password, must verify their voice.

Figure 2 shows the login process for regular users. The appearance of the application on a mobile phone is also depicted.

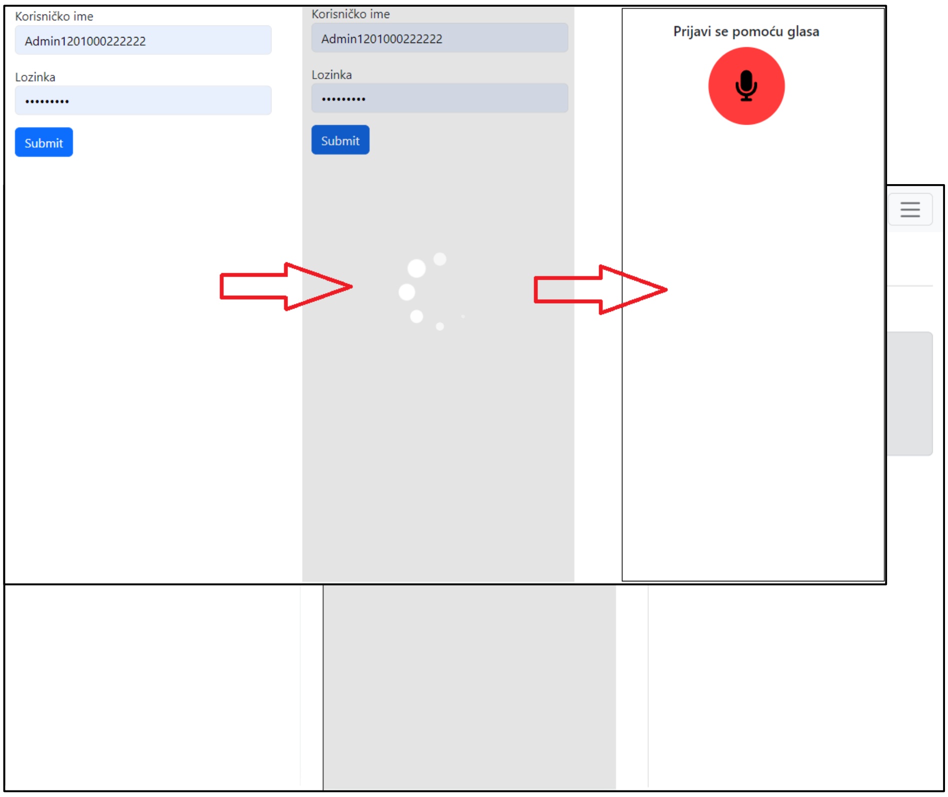


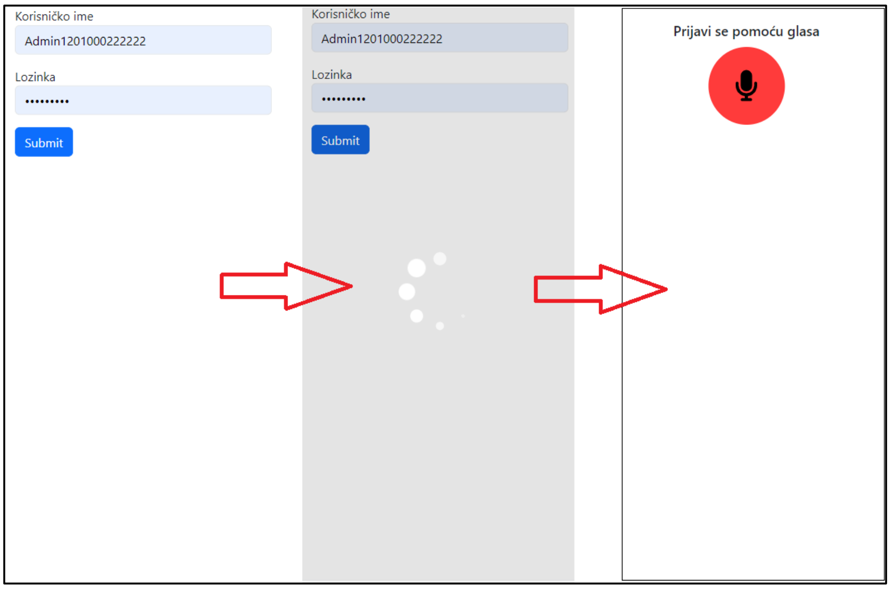
Figure 2 – Regular user login

Administrators log in to the application first with a username and password, and then they verify their voice. Voice verification is not possible if authentication has not been successful.

As mentioned earlier, voice verification is done using a service developed in the Python programming language, and the Speechbrain package is used.

The machine learning models provided by this package are suitable because voice verification only requires one recording for comparison, achieving an accuracy of over 80 percent.

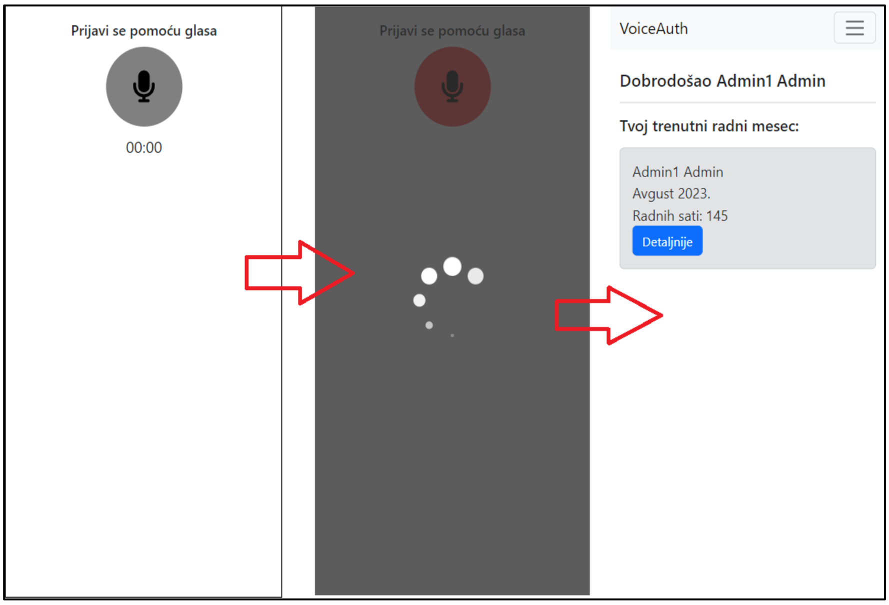
The authentication flow for administrators is shown in Figures 3 and 4.



Korisničko ime
Admin1201000222222
Lozinka
•••••••••

Korisničko ime
Admin1201000222222
Lozinka
•••••••••

Figure 3 – Administrator login, part 1



Dobrodošao Admin1 Admin

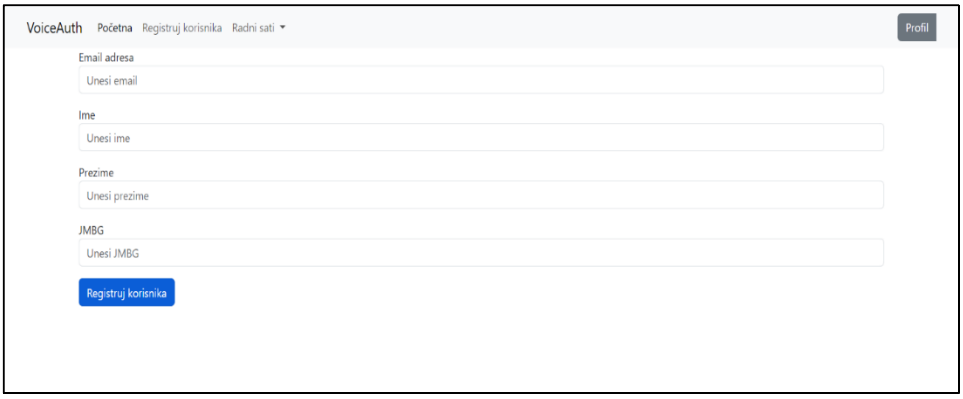
Tvoj trenutni radni mesec:

Admin1 Admin
Avgust 2023.
Radnih sati: 145

Figure 4 – Administrator login, part 2

After the administrator logs in with a username and password, they will be redirected to the voice verification page. The dependency between the two login steps is achieved by creating a temporary JSON Web Token, which serves as confirmation that the administrator has successfully completed password verification.

In the application, the registration of new users is performed by administrators, and in this way, only regular users can be registered. The registration of new users is done through an email address entered by the administrator. The administrator enters the user's information, including the first name, last name, ID number (JMBG), and the email address to which the user will receive a message for further registration, as shown in Figure 5.



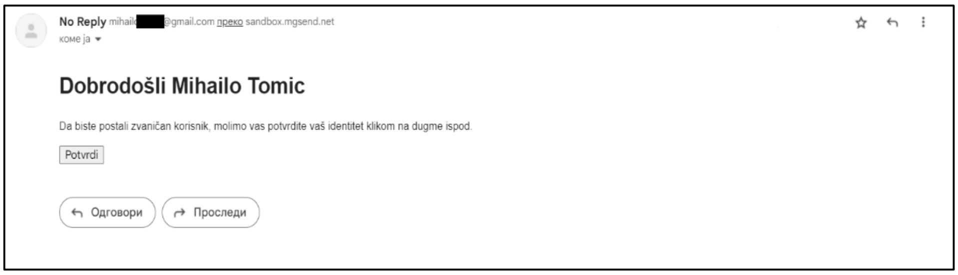
Email adresa
Unesi email

Ime
Unesi ime

Prezime
Unesi prezime

JMBG
Unesi JMBG

Figure 5 – New user registration



No Reply mihail[redacted]@gmail.com преко sandbox.mgsend.net
коме ја

Dobrodošli Mihailo Tomic

Da biste postali zvaničan korisnik, molimo vas potvrdite vaš identitet klikom na dugme ispod.

Potvrdi

Figure 6 – Confirmation email

After this, an invitation with a link arrives at the email address, which can be used for further registration and creating a password for future logins, as shown in Figure 6.

Class diagram

In order for the above to be realized, the implementation of the application has a certain number of classes in the TypeScript programming language, which are implemented in the manner shown in the class diagram. The best way to understand and overview the entire code is by displaying class diagrams. The class diagram allows us to map the system structures by showing different classes, attributes, operations, and relationships between objects. In the diagram, a class is represented by a rectangle. Each rectangle is divided vertically into three parts. The upper part has the class name. The second and third sections provide details about the class's operations, behavior, and attributes. Figure 7 shows the class diagram for the REST API application described in this paper because it contains the majority of the business and domain logic of the system, without the presentation layer.

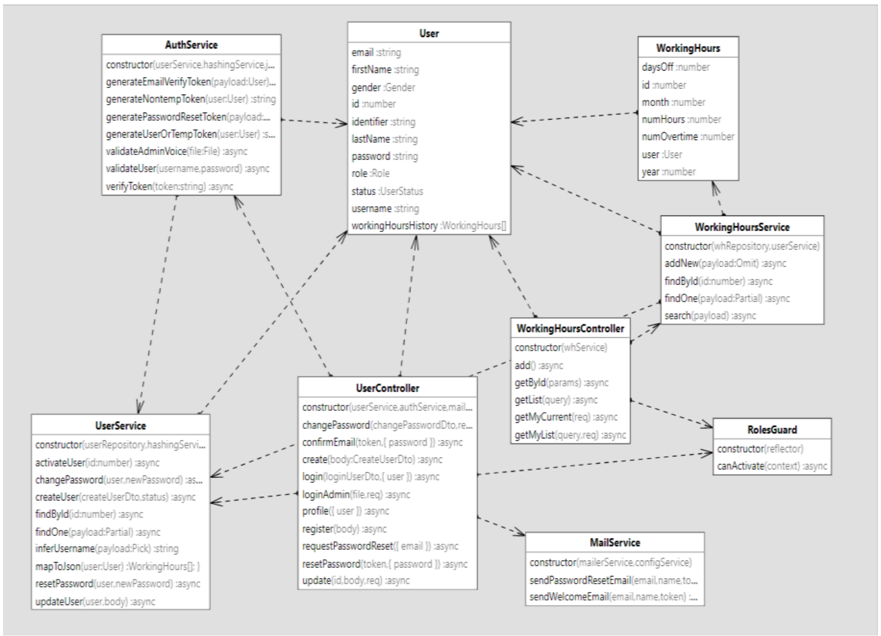

AuthService
constructor(userService,hashingService,j...
generateEmailVerifyToken(payload:User)...
generateNontempToken(user:User) :string
generatePasswordResetToken(payload:...
generateUserOrTempToken(user:User) :s...
validateAdminVoice(file:File) :async
validateUser(username,password) :async
verifyToken(token:string) :async
User
email :string
firstName :string
gender :Gender
id :number
identifier :string
lastName :string
password :string
role :Role
status :UserStatus
username :string
workingHoursHistory :WorkingHours[]
WorkingHours
daysOff :number
id :number
month :number
numHours :number
numOvertime :number
user :User
year :number
WorkingHoursService
constructor(whRepository,userService)
addNew(payload:Omit) :async
findById(id:number) :async
findOne(payload:Partial) :async
search(payload) :async
WorkingHoursController
constructor(whService)
add() :async
getById(params) :async
getList(query) :async
getMyCurrent(req) :async
getMyList(query,req) :async
UserController
constructor(userService,authService,mail...
changePassword(changePasswordDto,re...
confirmEmail(token,{ password }) :async
create(body:CreateUserDto) :async
login(loginUserDto,{ user }) :async
loginAdmin(file,req) :async
profile({ user }) :async
register(body) :async
requestPasswordReset({ email }) :async
resetPassword(token,{ password }) :async
update(id,body,req) :async
UserService
constructor(userRepository,hashingServi...
activateUser(id:number) :async
changePassword(user,newPassword) :as...
createUser(createUserDto,status) :async
findById(id:number) :async
findOne(payload:Partial) :async
inferUsername(payload:Pick) :string
mapToJson(user:User) :WorkingHours[]; }
resetPassword(user,newPassword) :async
updateUser(user,body) :async
RolesGuard
constructor(reflector)
canActivate(context) :async
MailService
constructor(mailerService,configService)
sendPasswordResetEmail(email,name,to...
sendWelcomeEmail(email,name,token) :...

Figure 7 - Class diagram

The class diagram shown in the figure contains the most basic classes that represent the key components of the entire application, namely: UserController, WorkingHoursController, UserService, WorkingHoursService, User, WorkingHours, AuthService, MailService, and RolesGuard. The explanations of the main classes are given below:

- UserController and WorkingHoursController: These are controller classes defining the inputs to the system and operations provided by the created API. Each method in these classes corresponds to a URL resource that allows data manipulation in the system.

- UserService and WorkingHoursService: These are the core service classes containing the main business logic of the application. They define methods that, using an Object-Relational Mapper (ORM) tool, manipulate data in the database. These service classes are intended to be used by controller classes but can also be utilized in other service classes.

- AuthService: This auxiliary service class handles user authentication. Besides the basic authentication role, it also incorporates two-factor authentication and token generation.

- MailService: This auxiliary service class is responsible for sending messages via email. It utilizes the SMTP protocol to send messages to users, with messages playing a role in registering new users and changing passwords.

- User and WorkingHours: These are data model classes defined using an ORM tool. The attributes of these classes are used for generating tables and fields in the database.

- RolesGuard: This class plays a role in user authorization when accessing resources. It provides a generalized solution for authorization in the application. With the specific syntax for the programming language, it allows adding user authorization based on their role in the system (regular user or administrator) with just one line of code for each resource (represented as methods in the controller classes).

Analysis of the proposed solution

As mentioned in the previous chapters, the majority of the application is written in the TypeScript programming language, along with a voice verification service written in the Python programming language. The text below provides the basic information about the software tools and packages used in the development of the application.

NestJs

NestJS is a modern, flexible, and scalable framework for developing server-side applications in the TypeScript programming language. This framework provides a contemporary way of structuring applications using a modular concept and views the application as a set of interconnected modules.

NestJS relies on fundamental concepts from object-oriented programming as well as advanced concepts such as Inversion of Control (IoC) and Dependency Injection (DI), enabling better code organization and testing. This architecture encourages the development of applications that are easy to maintain and test.

With support for synchronous and asynchronous code, NestJS allows the development of high-performance applications. Additionally, within NestJS, there is access to a wide range of modules and packages that facilitate the implementation of various functionalities, including authentication, database management, routing, and many others.

TypeORM

TypeORM is an object-relational mapping (ORM) library for JavaScript and TypeScript programming languages. This tool allows developers to easily communicate with relational databases using objects and classes.

TypeORM provides an intuitive way to define and model databases using object-oriented programming concepts. Working with databases and tables is achieved through classes and annotations, reducing the need for SQL queries and direct table interactions.

Angular

Angular is a popular framework for developing web applications. Developed by Google, this framework enables programmers to build dynamic and interactive web applications using HTML, CSS, and the TypeScript language.

Angular uses the concept of components, where different parts of the application are developed as a set of blocks that can be reused multiple times. This approach facilitates the organization and maintenance of code, as well as the development of various functionalities. Angular supports modular code organization and the use of object-oriented programming principles in the development of the presentation layer of web applications.

Speechbrain

SpeechBrain is an open-source Python framework for speech processing that covers various aspects of speech and language analysis. This framework allows researchers and programmers to develop applications and models for speech processing, speech recognition, speech synthesis, emotional analysis, and many other applications in the field of natural language processing (NLP).

SpeechBrain provides a straightforward way to build and experiment with different models and algorithms for speech analysis. It offers a dataset, preprocessing tools, model-building modules, and the ability to test and fine-tune models.

Possibilities of upgrading the proposed solution

This test application provides only the basic scope of functionality and represents a demonstration of using modern biometric authentication techniques in the Internet and other network systems. The application itself does not provide utility value but can serve as a good foundation for building a more complex system and its use through the internet or private computer networks. The application can be upgraded as a time-tracking system by introducing integrations with external time-tracking systems, providing necessary legal and accounting information within the company. Additionally, the modular nature of the developed solution allows for efficient transformation of the application into a microservices architecture, enabling its use as a standalone service for user authentication and authorization.

Further upgrades to the application would depend on the needs of the real system and the features it would offer, but some possible functionalities that could enhance the application's significance include:

- Enabling the use of voice verification dependent on spoken words by adding speech recognition algorithms to the verification service. This would allow real-time voice verification based on the uttered words, significantly improving the system's security.

- Providing other types of two-factor authentication, contributing to user experience.
- Enabling two-factor authentication for all types of users.

From a technical perspective, the quality of the already developed solution can be enhanced by allowing access to the application in a test environment via a web server for users to test application functionalities from their computers. Simplifying the installation of the application on servers could be achieved by introducing DevOps work methodology and automating the installation process through integration with version control systems like GitHub, used during solution development, and CI/CD (Continuous Integration & Continuous Deployment/Delivery) systems such as Jenkins and GitHub Actions.

Conclusions

Modern information technologies and information systems provide us with increasing possibilities day by day, penetrating into more and more areas of our lives. Internet users are becoming owners of new accounts on various systems and services every day. With the increase in accounts and the processing power of modern computers, there is a growing occurrence of attacks and digital identity theft. Two-factor authentication is one solution to this problem, offering users a higher level of security when accessing their accounts and becoming a standard in complex services.

This paper explores the possibility of implementing two-factor authentication for users in an information system. The ultimate goal was to demonstrate that two-factor authentication is not a burden to users and can be used easily and efficiently in computer systems. Additionally, the paper aims to show that biometric methods can be successfully used as one of the factors.

References

Bondarchuk, A.P., Onysko, A.I., Otrokh, S.I. & Shevchuk, D.O. 2023. Two-factor user authentication system using facial recognition. *Telecommunication and Information Technologies*, 3, pp.79-84 (in Ukrainian). Available at: https://doi.org/10.31673/2412-4338.2023.039699.

Chandrakar, P. & Om, H. 2015. RSA Based Two-factor Remote User Authentication Scheme with User Anonymity. *Procedia Computer Science*, 70, pp.318-324. Available at: https://doi.org/10.1016/j.procs.2015.10.023.

Jones, M., Bradley, J. & Sakimura, N. 2015. JSON Web Token (JWT), Request for Comments:7519. *Internet Engineering Task Force (IETF)* [online]. Available at: https://datatracker.ietf.org/doc/html/rfc7519 [Accessed: 4. October 2023]. ISSN: 2070-1721.

Kaur, D. & Kumar, D. 2021. Cryptanalysis and improvement of a two-factor user authentication scheme for smart home. *Journal of Information Security and Applications*, 58, art.number:102787. Available at: https://doi.org/10.1016/j.jisa.2021.102787.

Marky, K., Ragozin, K., Chernyshov, G., Matviienko, A. & Schmitz, M. 2022. “Nah, it’s just annoying!” A Deep Dive into User Perceptions of Two-Factor Authentication. *ACM Transactions on Computer-Human Interaction*, 29(5), art.number:43, pp.1-32. Available at: https://doi.org/10.1145/3503514.

Reese, K., Smith, T., Dutson, J., Armknecht, J., Cameron, J. & Seamons, K. 2019. A Usability Study of Five Two-Factor Authentication Methods. In: *Proceedings of the Fifteenth Symposium on Usable Privacy and Security*, Santa Clara, CA, USA, August 12-13 [online]. Available at: https://www.usenix.org/conference/soups2019/presentation/reese [Accessed: 4. October 2023].

Tot, I., Trikoš, M., Bajčetić, J., Lalović, K. & Bogićević, D. 2021. Software Platform for Learning about Brain Wave Acquisition and Analysis. *Acta Polytechnica Hungarica*, 18(3), pp.147-162. Available at: https://doi.org/10.12700/APH.18.3.2021.3.8.

Tomić, M. 2023a. Mihailotomi / voice-auth-api. *Github.com*, e954845 [online]. Available at: https://github.com/mihailotomi/voice-auth-api [Accessed: 4. October 2023].

Tomić, M. 2023b. Mihailotomi / voice-auth-ui. *Github.com*, da4f656 [online]. Available at: https://github.com/mihailotomi/voice-auth-ui [Accessed: 4. October 2023].

Tomić, M. 2023c. Mihailotomi / voice-auth-verification. *Github.com*, 1da74e5 [online]. Available at: https://github.com/mihailotomi/voice-auth-verification [Accessed: 4. October 2023].

Tomić, M. 2023d. *Jedan pristup implementaciji dvofaktorske autentikacije u računarskim sistemima*. BS thesis. Belgrade, Serbia: University of Defence (in Serbian).

Yuan, J.-J. 2013. An enhanced two-factor user authentication in wireless sensor networks. *Telecommunication Systems*, 55, pp.105-113. Available at: https://doi.org/10.1007/s11235-013-9755-5.

Zhu, H., Jin, W., Xiao, M., Murali, S. & Li, M. 2020. BlinKey: A Two-Factor User Authentication Method for Virtual Reality Devices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(4), art.number:164, pp.1-29. Available at: https://doi.org/10.1145/3432217.

Zou, S., Cao, Q., Wang, C., Huang, Z. & Xu, G. 2021. A Robust Two-Factor User Authentication Scheme-Based ECC for Smart Home in IoT. *IEEE Systems Journal*, 16(3), pp.4938-4949. Available at: https://doi.org/10.1109/JSYST.2021.3127438.

Implementación de la autenticación del usuario a través de dos factores en sistemas informáticos

Mihailo D. Tomić[a], *Olivera* M. Radojević[b]

[a] Codebehind doo, Belgrado, República de Serbia, **autor de correspondencia**

[b] Fuerzas Armadas de Serbia, Fuerza Aérea y Defensa Aérea,
210.º Batallón de Señales, Belgrado, República de Serbia

CAMPO: ciencias de computación, TI
TIPO DEL ARTÍCULO: artículo científico original

Resumen:

Introducción/objetivo: El artículo explora la implementación de la autenticación de dos factores (2FA) en sistemas informáticos, abordando la creciente necesidad de una mayor seguridad. Destaca las vulnerabilidades de la autenticación basada en contraseñas y enfatiza las ventajas de 2FA para mitigar las amenazas digitales. El desarrollo de la aplicación VoiceAuth, que integra 2FA mediante una combinación de contraseña y autenticación de voz, sirve como ejemplo práctico.

Métodos: La investigación adopta una arquitectura de tres niveles para la aplicación VoiceAuth, que abarca una base de datos, una API REST del lado del servidor y una aplicación de una sola página del lado del cliente. La verificación del orador se emplea para la autenticación de voz, analizando elementos como el tono, el ritmo y las formas del tracto vocal. El documento también analiza posibilidades para futuras actualizaciones, sugiriendo mejoras como la verificación de voz en tiempo real y métodos 2FA adicionales.

Resultados: La implementación de la aplicación implica un desglose detallado de la arquitectura API REST, las aplicaciones de página única (SPA) y el servicio de verificación de oradores.

Conclusión: La investigación subraya el papel crucial de la autenticación de dos factores (2FA) para reforzar la seguridad de los sistemas informáticos. La aplicación VoiceAuth sirve como demostración práctica, mostrando la integración exitosa de 2FA a través de una combinación de contraseña y autenticación de voz. La arquitectura modular de la aplicación permite posibles actualizaciones.

Palabras claves: autenticación, sistemas informáticos, biometría.

Внедрение двухфакторной аутентификации пользователя в компьютерных системах

Михаило Д. Томич[а], *Оливера* М. Радоевич[б]

[а] Codebehind doo, г. Белград, Республика Сербия, **корреспондент**

[б] Вооруженные силы Республики Сербия, Военная авиация и
Противовоздушная оборона, 210 батальон связи,
г. Белград, Республика Сербия

РУБРИКА ГРНТИ: 28.17.33 Компьютерное моделирование реальности. Виртуальная реальность
ВИД СТАТЬИ: оригинальная научная статья

Резюме:

Введение/цель: В данной статье исследуется внедрение двухфакторной аутентификации (2FA) в компьютерных системах, подчеркивая растущую потребность в усилении безопасности. В статье освещаются уязвимость аутентификации на основании пароля и подчеркиваются преимущества 2FA в снижении цифровых угроз. В качестве практической иллюстрации представлена разработка приложения VoiceAuth, интегрирующего 2FA с помощью комбинации пароля и голосовой аутентификации.

Методы: В данном исследовании используется трехуровневая архитектура для приложения VoiceAuth, включающая базу данных, серверный REST API и пользовательское приложение. Для голосовой аутентификации используется проверка говорящего, анализирующая такие элементы, как: высота тона, ритм, тембр и такт. В статье также обсуждаются возможности будущих обновлений, в том числе проверка голоса в режиме реального времени и дополнительные методы 2FA.

Результаты: Внедрение приложения включает в себя детальную описание архитектуры REST API, одностраничных приложений (SPA) и службы аудиопроверки.

Выводы: В исследовании подчеркивается решающая роль двухфакторной аутентификации (2FA) в повышении безопасности компьютерных систем. Приложение VoiceAuth демонстрирует на практике успешную интеграцию 2FA посредством комбинации пароля и голосовой аутентификации. Модульная архитектура приложения допускает возможность его модернизации.

Ключевые слова: аутентификация, компьютерные системы, биометрия.

Имплементација двофакторске аутентификације у рачунарским системима

Михаило Д. Томић[а], *Оливера* М. Радојевић[б]

[а] Codebehind doo, Београд, Република Србија, **аутор за преписку**

[б] Војска Србије, Ратно вазухопловство и противваздухопловна одбрана, 210. батаљон везе, Београд, Република Србија

ОБЛАСТ: рачунарске науке, информационе технологије
КАТЕГОРИЈА (ТИП) ЧЛАНКА: оригинални научни рад

Сажетак:

Увод/циљ: У раду се истражује имплементација двофакторске аутентификације (2FA) у рачунарским системима и наглашава растућа потреба да се унапреди безбедност. Истиче се угроженост аутентификације на основу лозинке и наглашавају предности 2FA у смањивању дигиталних претњи. Развој апликације VoiceAuth, која интегрише 2FA кроз комбинацију лозинке и аутентификације гласом, служи као практичан приказ.

Методе: Истраживање користи трослојну архитектуру за развој апликације VoiceAuth, обухватајући базу података, REST API на страни сервера и клијентску апликацију. За аутентификацију гласом користи се верификација говорника, анализирајући елементе као што су тон, ритам и облици вокалног такта. У раду се, такође, разматрају могућности за будућа унапређења; предлажу се додаци као што су верификација гласа у реалном времену и додатне методе 2FA.

Резултати: Имплементација апликације укључује детаљан опис архитектуре REST API-а, клијентске апликације и сервиса за верификацију говорника.

Закључак: Улога двофакторске аутентификације (2FA) у јачању безбедности рачунарских система је кључна. Апликација VoiceAuth служи као практична демонстрација, приказујући успешну интеграцију 2FA кроз комбинацију лозинке и аутентификације гласом. Модуларна архитектура апликације оставља простор за потенцијална унапређења.

Кључне речи: аутентификација, рачунарски системи, биометрија.

Paper received on: 05.10.2023.
Manuscript corrections submitted on: 03.03.2024.
Paper accepted for publishing on: 04.03.2024.