

computers



Article

Enhancing JWT Authentication and Authorization in Web Applications Based on User Behavior History

Ahmet Bucko, Kamer Vishi, Bujar Krasniqi and Blerim Rexha

Special Issue

Innovative Authentication Methods

Edited by





Prof. Dr. Leandros Maglaras, Prof. Dr. Helge Janicke, Dr. Mohamed Amine Ferrag and Dr. Francisco J. Aparicio-Navarro



<https://doi.org/10.3390/computers12040078>

Article

Enhancing JWT Authentication and Authorization in Web Applications Based on User Behavior History

Ahmet Bucko ¹, Kamer Vishi ², Bujar Krasniqi ^{1,*} and Blerim Rexha ¹¹ Faculty of Electrical and Computer Engineering, University of Prishtina, 10000 Prishtina, Kosovo² Department of Informatics, University of Oslo, Gaustadalléen 23B, 0373 Oslo, Norway

* Correspondence: bujar.krasniqi@uni-pr.edu

Abstract: The rapid growth of the web has transformed our daily lives and the need for secure user authentication and authorization has become a crucial aspect of web-based services. JSON Web Tokens (JWT), based on RFC 7519, are widely used as a standard for user authentication and authorization. However, these tokens do not store information about the user's behavior history. To address this issue, this paper presents a solution to enhance the trustworthiness of user authentication in web applications based on their behavior history. The solution considers factors such as the number of password attempts, IP address consistency, and user agent type and assigns a weight or percentage to each. These weights are summed up and stored in the user's account, and updated after each transaction. The proposed approach was implemented using the .NET framework, C# programming language, and PostgreSQL database. The results show that the proposed solution effectively increases the level of trust in user authentication. The paper concludes by highlighting the strengths and limitations of the proposed solution.

Keywords: cybersecurity; user behavior; authentication; authorization; JWT



Citation: Bucko, A.; Vishi, K.; Krasniqi, B.; Rexha, B. Enhancing JWT Authentication and Authorization in Web Applications Based on User Behavior History. *Computers* **2023**, *12*, 78. <https://doi.org/10.3390/computers12040078>

Academic Editors: Leandros Maglaras, Helge Janicke, Mohamed Amine Ferrag and Francisco J. Aparicio-Navarro

Received: 12 February 2023

Revised: 10 April 2023

Accepted: 11 April 2023

Published: 13 April 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The growth of the Internet and technological advancements have impacted various aspects of our lives. Over the past two decades, the use of web-based systems has dramatically increased, revolutionizing the way we access public services, banking services, e-learning, travel booking services, and more. However, the increased demand for these e-services has highlighted the need for improved security measures against impersonation fraud. While many studies have addressed the security of web-based systems, it's important to note that a universal solution, such as "one size fits all", is not adequate for securing a wide range of e-services against fraud [1].

One of the main challenges on the Internet is the trustworthiness of user identity. On the Internet, it is easier to pretend to be someone else. At a time when the number of services offered through the Internet is rapidly growing, and also the number of users of these services has increased in an even more rapid way, which has increased the need for new infrastructure of services of identification, authentication and authorization. Big tech players and the open source community have influenced the development and maintenance of these products, hence each company today tends to use the identification, authentication and authorization process depending on the need of their service.

This paper will propose a secure application for conducting transaction services. (The proposed model repositories on Github, API and template, are located at https://github.com/ahmetbucko/Template_Csharp [accessed on 11 February 2023], whereas the Rewarding service is found at <https://github.com/ahmetbucko/BackgroundWorker> [accessed on 11 February 2023]). The proposed application will utilize JSON Web Tokens (JWT) authentication and authorization techniques and enhance them based on user behavior history. The contributions of this research paper include the development of a

secure and personalized approach to transaction services that considers the behavior of users to enhance the authentication and authorization process. The proposed model is made available as a public repository on Github, along with an API and template. This research paper aims to provide a comprehensive guide for researchers and developers who are interested in improving the security of web-based systems. It will delve into the process of registration, authentication, and authorization, using a dynamic platform that is customized to fit the individual user. The user's level of reliability will be calculated based on their behavior and usage of the platform.

The remainder of this paper is structured as follows: in Section 2, we delve into the background information, focusing on topics such as user identification, authentication, and authorization. In Section 3, we provide a review of related works. Our proposed approach is presented in Section 4, while Section 5 outlines our experimental setup, including an in-depth description of our relational database and the implementation of our method, featuring two mathematical formulas. This section also covers aspects such as the registration and login process, user credibility assessment, challenge generation, and transaction and CRUD operations. Section 6 features a critical review on results, along with an analysis of the performance of our method. Finally, we conclude the paper in Section 7, where we offer our discussion and conclusion, as well as suggestions for future research.

2. Background Information

The Internet has greatly influenced the development of various methods of identification, authentication, and authorization. However, it has also facilitated the emergence of false identities and anonymous activities. This paper focuses on the characteristics of the access control process and discusses the different processes involved in obtaining access. Unauthorized access is a significant risk that poses a serious threat to the digital world. It can lead to devastating consequences such as data breaches, ransomware, and password leaks. In many cases, these security breaches occur due to weak access control mechanisms and inadequate use of modern technologies to protect digital systems. Previous studies have also highlighted the importance of addressing these issues [2].

2.1. Identification

Identification refers to the process of connecting a specific person with a particular identity from among many [3]. This is followed by authentication and authorization. In this context, the following questions arise: 'Is the person who he claims to be?', 'Has this person been here before?' and 'Should this individual be allowed access to our system?' The state-of-the-art definitions for these questions are presented in several research publications, as in [4,5] are presented an approach using smart cards and X.509 certificates to provide user identification and ensure data privacy and integrity.

2.2. Authentication

Authentication is determining whether a user or process is the person or entity it claims to be. It assures that the communicating entity is genuine [6]. The user must provide the credentials required to access the web service during the authentication process. Biometric authentication has gained more attention [7], especially after Google and other identity providers introduced passkey technology. However, at the same time, more attacks are being implemented using biometrics, as presented in [8], using Machine Learning techniques.

Some of the authentication methods, traditionally, can be classified as follows [9]:

- Something we know (PIN, passwords).
- Something we have (smart cards, digital certificates, etc.).
- Something you are (biometric, fingerprints, face, etc.).

The importance of authentication lies in the fact that it validates users or processes that meets the conditions to gain access to resources in sensitive data [3]. This paragraph discusses the benefits of authentication in protecting data and limiting access. Multiple-

factor authentication is recommended to increase security, such as combining passwords with biometric data or smart cards. A new form of authentication gaining popularity is cookie authentication, which uses an HTTP cookie to identify a user to a web service. This is convenient for users, as they don't need to remember their login details every time they visit the website. However, if cookies are not encrypted properly, they can be stolen and pose security risks. SSL usage is recommended to mitigate this risk.

2.3. Authorization

Authorization is a security technique for determining a user's privileges or suitability to perform specific tasks on a system [3]. Authorization is the last chain after identification and authentication, which completes the process of granting access. Authorization also plays a role at the level of user or process access, or the authorization displays as many web service resources as the user has permission to access, or the level of access that the user or process possesses in this case.

The importance of authorization lies in the fact that usability limitations are imposed on the user, which have the following effects:

- Ensures that users cannot access an account that is not theirs.
- Prevents visitors and employees from accessing secure areas.
- Ensures that all features are not available for free accounts.
- Ensures that internal accounts only have access to the information they require [3].

As correctly noted by [10] authorization is about a more detailed set of constraints on access to various system resources, while the result of the authentication process is a binary decision, true for granting and false for rejecting access.

2.3.1. Authorization Strategies

- RBAC (Role-Based Access Control): is an authorization strategy that is directly related to the role, but not to the user, where the role is a collection of permissions that are assigned to the role for a specified or unspecified time and the role is assigned to a group of people.
- ReBAC (Relationship-Based Access Control): an interesting type of code as it tries to answer the question "Does this user have enough of a relationship with this object or action that they can access it?", where the user may be part of a group of users with its own resource management. Furthermore, some links can be prioritized to be filled for the user to gain access.
- ABAC (Attribute-Based Access Control): it is the simplest strategy, it is assigned to only one user, and it is checked whether the user in question has permission to access that resource. Moreover the management is more defined for each user where privileges must either be added or removed.

2.3.2. Authorization Types

Basic authorization is one of the oldest authorizations that is used in old web platforms. It is known as one of the first types of authorization at the level of the communication of the web platforms with the user via the web. Its usage exposes a vulnerability, since it contains the username and password encoded with the base64 protocol.

The Authorization Protocol used in the header of the request made to the server as:

```
Authorization: Basic dXNlcjpwYXNzd29yZA==
```

where the part after Basic is the user and password separated by colons (:), that is, user:password. Let us present the case where there is no SSL or the application X.509 certificate expires. Then we have a case that enables the complete web traffic to be captured with tools such as Wireshark, where a simple filter can capture the web traffic. Even the Base64 conversion to plain text is performed automatically, without any deeper knowledge.

Authorization: Basic user:password

The authorization procedure in HTTP communication is not secure as it is in plain text and can be read by any third party. It is not recommended for use in applications with sensitive data, but if used over a secure connection, it is simple as it requires the user to provide a username and password for identification.

Bearer authorization is a newer, more secure method of authentication for HTTP and HTTPS communication. It uses security tokens, specifically JSON Web Tokens (JWT), instead of transmitting the user and password with each request. After successful authentication using the username and password, an encrypted key is generated and validated by the platform's database. The platform then generates a JWT token, which can be used for subsequent requests. This process enhances security by eliminating the need to transmit sensitive information with each request, improving the user experience while protecting sensitive data [11]. The bearer authentication process involves the following steps:

1. The client makes an initial request to the server without authentication.
2. The server responds with a 401 response, indicating that authentication is required, and includes a new case value (i.e. a nonce).
3. In the next request, the client includes an Authorization: Digest header, which includes the user's credentials, the protected space identifier, and a response value that is generated through the hash function (hash(nonce, user, space, URI, password)).
4. The server then looks up the user in its database and calculates the hash function using the user's password. If the values match, the request is deemed authenticated.

An example of the returned signed JWT token:

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.e..

And decoded in JSON format is presented in Listing 1.

Listing 1. Decoded token.

```
//Header:
{
  "alg": "HS256",
  "typ": "JWT"
}
//Payload:
{
  "unique_name": "john.doe@example.com",
  "nbf": 1665335411,
  "exp": 1665335710,
  "iat": 1665335411,
  "iss": "login_fiek",
  "aud": "login_fiek"
}
//Signature:
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
)
```

Bearer authorization is commonly used by open source platforms for authentication and automation purposes. Some platforms even support tokens that never expire, making it easier for systems to use them. However, tokens with an expiration date are typically utilized by users for work-related purposes.

Digest Authorization

Digest authorization is an advancement of basic authorization. It addresses the security concerns of basic authorization, where both the user and password are sent in

the authorization header, making it an insecure and unreliable method of authentication. Instead, digest authorization uses the user identifier, protected space identifier, and the URL of the request to generate a hash, typically using Message Digest (MD5) technology. This provides a more secure way of authentication by adding extra layers of protection to the password [12].

Digest authorization is considered secure against replay attacks due to the use of the MD5 hashing algorithm. Each request generates a unique random number, known as a nonce, which is combined with other parameters, such as the user identifier, protected space identifier, and the URL of the request. This combination results in a hash that is used to authenticate the request.

2.3.3. Authorization Protocols

The purpose of an authentication protocol is to verify the identity of a user or a software system that is accessing a platform. This is accomplished by the receiving party, such as a server, confirming the identity of the connecting party. Network authentication is a widely used security mechanism in computer systems, and various methods of authentication are employed for this purpose [13,14]. Below we will discuss several commonly used authentication protocols:

Single Sign-On (SSO) is an authentication method that allows users to securely access multiple applications and websites using a single set of credentials. This method is based on a trust relationship established between the service provider (an application) and an identity provider.

Furthermore, the trust relationship between the identity provider and the service provider is established through the exchange of a certificate. The certificate serves as a sign that the identity information being sent from the identity provider to the service provider is from a trusted source. In Single Sign-On (SSO) processes, this identity information is presented in the form of tokens that carry identifying details of the user, such as an email address or username [15,16].

The Security Assertion Markup Language (SAML) Framework is a widely used authentication method that leverages Extensible Markup Language (XML) for exchanging authentication data between the user and the service provider that manages the authorization process. SAML works in conjunction with an identity provider to securely authenticate users [17].

The Single Sign On (SSO) system poses a challenge for password management as users need to remember multiple passwords for various platforms. This leads to security issues as users may opt for simpler passwords, making sensitive data vulnerable. To address this, the Security Assertion Markup Language (SAML) framework allows users to access multiple services with one password, authenticated through an XML file signed with the XML signature scheme. The IAM service request grants users access to different platforms, reducing the need for managing multiple passwords. The SAML framework simplifies the authentication process and enhances security by validating users' certificates to determine if they are authorized to use a web service. By implementing a single sign-on solution, the authentication process is simplified, and security is improved. [18].

The two goals of the SAML framework are:

1. To allow users who are authenticated to gain access to all the service provider's applications without having to re-authenticate for each one.
2. To provide access to new applications using existing user accounts.

In the SAML framework, there are three key participants:

1. User Application
2. Identity Provider - responsible for user session management
3. Service Provider - responsible for accepting authentication assertions to grant or deny access to services.

Another authorization protocol is **OpenID Connect**, also known as OpenID. This is an additional layer added to the OAuth 2.0 authentication framework. It is a specific implementation of OAuth 2.0 where the identity provider managing the authorization server also holds the protected resource, which in this case is the user data that the application intends to access. OpenID enables supported parties to both verify the end-user's identity and retrieve user information using the Representational State Transfer (REST) API [19].

The processes involved in OpenID include:

- The resource owner who holds the user information that the application intends to access.
- The relying party who needs access to the end user's protected information.
- The OpenID provider, which acts as both an OAuth 2.0 authorization server and resource server, and holds the user information and grants access.

In essence, **OAuth** is a set of rules and standards that govern secure, third-party authorization for access to resources across servers and services that are not physically connected. This exchange of information can be carried out without actually sharing initial credentials, such as passwords, cards, or biometric data [7].

The OAuth 1.0 framework, being the first version of the OAuth framework, presents a more complex process compared to its successor, OAuth 2.0. The OAuth 1.0 framework comprises of two tokens, namely the request token and the access token. In contrast, the OAuth 2.0 framework simplifies the authorization procedure by presenting a more streamlined process, as will be discussed in the next section [20].

Moreover, the OAuth framework operates as a third-party system and facilitates the access to protected resources. The process begins with a token request, where the user provides their user key (Consumer Key) and password (Consumer Secret) to request authorization. Upon successful validation, a request token is returned. The user is then redirected to exchange the request token for an access token, which is validated and returned once the process is successful. With the access token, the user is able to consume services and resources from the platform.

The OAuth1.0 protocol utilizes two types of tokens for authorization:

- **Request Token:** This token includes the consumer key and the consumer secret that are utilized to request service authorization from the platform.
- **Access Token:** This token grants access to the resources of the platform and is obtained after successful validation of the Request Token.

On the other side, **OAuth 2.0**, the second version of the OAuth framework, builds upon the foundation laid by its predecessor, OAuth 1.0. This version aims to simplify the authorization process by eliminating the request token and consolidating it into a single access token, without compromising security. Despite this simplification, the widespread use of OAuth 2.0 serves as evidence that its security remains robust. In [21], the authors propose a trust-establishing approach among various parties, including the client, resource servers, and authorization server.

The authentication procedure in the OAuth 2.0 framework involves the following steps:

1. The user requests access to the platform resource by providing their unique username and password, along with the URI where the request will be made.
2. The authorization server verifies the client's authentication and checks if the requested purposes are permitted.
3. The resource owner interacts with the authorization server to grant access.
4. The authorization server redirects the client with either an authorization code or an access code, depending on the permission type. In some cases, a refresh token may also be provided.
5. With the access token, the client can request access to the resource from the resource server. (Reference: What Is OAuth 2.0 and What Does It Do for You?, n.d.)

Types of tokens inside OAuth 2.0 are:

- **Access Tokens** in OAuth 2.0 have a similar functionality as those in OAuth 1.0, serving as a string that represents authorized access to resources. This string is often meaningless to the end user and there are several types of Access Tokens, each with unique properties and use cases. OAuth 2.0 Access Tokens can be represented in several forms including:
 - **JSON Web Tokens (JWT)**, encrypted tokens, and reference tokens. JWT Tokens possess the advantage of being able to be verified without the presence of an authorization server, but are not capable of being revoked.
 - **Encrypted Tokens** prioritize data confidentiality and require introspection for validation by the resource server.
 - **Reference Tokens** serve as a reference to the data stored in the authorization server records and offer the advantage of data confidentiality and revocation capabilities, but may require each reading from the database to retrieve the claims held by the token.
- **Refresh Tokens:** In OAuth, the access token is issued by the authorization server in response to an authentication request for accessing the resource, but it is subject to expiration. When it expires, the user must re-authenticate with the authorization server to receive a new access token. Refresh tokens address this issue by enabling the extension of the access token's lifetime through periodic refreshing, using the refresh token. This allows for reduced token expiration time, enhancing security while still maintaining seamless access to the resource.

In Table 1 one can find the differences between the two OAuth frameworks.

Table 1. The key differences between OAuth 1.0 and OAuth 2.0. Adapted from [22].

OAuth 1.0	OAuth 2.0
Does not require HTTPS communication	Requires HTTPS communication
Requires a digital signature to sign the OAuth message	Does not require a digital signature as it relies on SSL communication
Handles only web-based applications	Can also handle non-web-based applications
Can be inflexible and challenging to implement	Is flexible and easily implementable for third parties
Uses the unique consumer secret to sign all requests to the authentication server	Includes a client secret in each request
Provides higher security through the use of digital signatures	Is considered less secure compared to OAuth 1.0 and is based on bearer tokens.

3. Related Work

User authentication and authorization are critical aspects of web-based services and the use of JSON Web Tokens (JWT) for this purpose has become widely adopted. However, the trustworthiness of user authentication in web applications can be compromised by various factors, such as password attacks, session hijacking, impersonation etc. To address these challenges, researchers and practitioners have proposed various approaches and frameworks to enhance the security of user authentication and authorization.

This section reviews the relevant literature on this topic, including studies on user behavior analysis, authentication protocols and security frameworks. We compare them to the proposed solution presented in this paper. Specifically, we discuss the strengths and limitations of these approaches and how they differ from our proposed solution, which aims to enhance the trustworthiness of user authentication based on the user's behavior history.

Zamfiroiu et al. [23] proposed a secure learning management system that employs user behavior for authentication and access control. Furthermore, the authors demonstrated

that the proposed system could authenticate users and grant them access to resources based on their behavior patterns with high accuracy, demonstrating the effectiveness of user behavior as an authentication mechanism.

Saleh et al. [24] conducted a systematic survey on identity authentication methods for online exams, examining various techniques and technologies, their strengths, weaknesses and effectiveness. The authors found that while various identity authentication methods were available for online exams, there needed to be more than a one-size-fits-all solution. A combination of techniques may be required to ensure exam integrity.

Catalin et al. [25] developed a user behavior profiling method for social media applications that uses clustering algorithms and semantic analysis to group users and identify their behavioral patterns. To address the issues, the authors found that their user behavior profiling method was able to effectively group users based on their behavioral patterns, which could help identify potential threats and improve security in social media applications.

Yu et al. [26] proposed a method for constructing user profiles by analyzing their behavior history, which involves modeling user interests, preferences, and activities using data mining techniques. The authors indicated that analyzing user behavior history was an effective method for constructing user profiles, which could be used to personalize recommendations and improve user experience in various applications.

Sandhu et al. [27] introduced role-based access control models, which provide a mechanism for regulating access to resources based on users' roles and privileges. The authors demonstrated that role-based access control models were effective in regulating access to resources based on users' roles and could be used to improve security and access control in various organizations.

Sandhu et al. [28] presented the NIST model for role-based access control, which aims to standardize the implementation of role-based access control across different organizations and systems. Moreover, the authors discovered that the NIST model for role-based access control could be used to standardize the implementation of role-based access control across different systems and organizations, improving interoperability and security.

In their paper Caruccio et al. [29] discuss the growing importance of end-user development (EUD), specifically in the context of web-based applications (EUDWeb). It highlights the need for better support in implementing access control, which is often considered complex. To address this, the authors propose an EUDWeb framework and tool that incorporates access control mechanisms in web applications. By building on a previous mockup-based EUDWeb approach and introducing visual assistance for role-based access control policies, they make it easier for end-users to develop web applications with access control functionalities. The usability of the framework is evaluated through a user study, demonstrating its effectiveness for a diverse group of end-users.

Thomas and Sandhu [30] introduced the task-based authorization controls (TBAC) approach as an active approach for authorization management, differing from traditional subject-object models. TBAC enabled just-in-time permission control and introduced authorization-steps for managing permissions. The authors collaborated with other universities to explore visual languages for policy expression and investigated issues related to authorization delegation and revocation.

Giordano and Polese [31] presented the Visual Computer Managed Security (Vicoms) framework, designed to help programmers implement access control in Java applications. Vicoms provides a transparent method for managing security aspects in enterprise-level applications, including legacy ones. The framework has been integrated into the Eclipse open-source development environment and tested through several case studies, one of which is detailed in the article.

Zhang et al. [32] presented a tool that generates verified eXtensible Access Control Markup Language (XACML) scripts from access control system descriptions written in a simple but expressive language proposed in a previous publication. While XACML is a standard language used in e-business, policy files written in it can be difficult to read and analyze directly. The proposed tool allows for the generation of XACML scripts for access

control systems that can be formally verified to be implementing the relevant policies. This enables algorithmic verification of access control systems against appropriately formalized policies, making it easier to ensure that the system is functioning as intended.

Heydon et al. [33] in this paper authors argue that traditional methods of specifying security policies using text-based languages are difficult to read and comprehend, so they use Miro, which provides a graphical interface that allows users to define security policies based on user roles and permissions. The paper also provides an overview of the Miro system and describes how it is used to specify security policies and also discuss the benefits of using it, including readability and easiness of use.

4. The Proposed Approach

After conducting several tests, the proposed method for initial credibility determination was established based on the numerical values that demonstrated the best performance, as presented in Section 5.2.3, specifically Section 5.2.3.1. However, upon initial result evaluations, it was determined that hard-coded numerical boundaries were not intuitive and lacked standardization. As a result, a second method was developed that normalized the boundaries using percentage values as presented in Section 5.2.3.2 and derived from the initial method. This approach provides a more understandable and standard way of calculating user credibility within the application.

The application employs two mathematical formulas, aimed to provide credible access to users while maintaining the security of the platform. The methods are designed to ensure that users who have met certain behavioral requirements are deemed trustworthy and are granted easy access to the platform. The others will be challenged via email to determine their true identity. The purpose of these calculations is to facilitate credible user access without compromising platform security.

After conducting various analyses, the implementation of authentication methods in various companies and corporations has highlighted the need for a more credible technique, where the user himself determines the level of credibility based on his behavior and interaction with the platform. The rapid growth of the Internet has also brought with it an increase in online threats and the restriction of certain resources for specific users has led to the development of various authentication techniques. To enhance security, we are now using JWT-based platforms, but with a dynamic approach to granting access to resources based on credibility. Our platform has introduced two methods for measuring the credibility of users during the authentication process to access resources. This allows secure transactions to be carried out via our application.

5. Experimental Setup

For the development of this platform, the .NET Version 6.0 framework is used, which is based on the C# programming language. For API development, is used the REST technology and PostgreSQL as database engine. The application runs as a service and returns JSON responses. The app is represented in SWAGGER with OpenAPI (API and template repository is available on Github: https://github.com/ahmetbucko/Template_Csharp [accessed on 11 February 2023]) configuration as illustrated in the Figure 1.

Entity Framework Core is also used, for the implementation and handling of the code-first approach for creation, updating and deletion of tables or fields of the database, which is supported by migrations inside this framework.

5.1. Relational Database

Figure 2 illustrates the relationship of the main tables in our developed platform, developed in PostgreSQL and represented in Barker's notation [2,34]. The main tables of our proposed framework include: User, Customer, and Transaction, where the User and Customer tables are related to each other through a one-to-one relationship since the User table holds the personal data and the Customer table holds the user state data. The Customer table is related to Transactions table via a one-to-many relationship. The Transactions table

holds the user transaction data and manages points earned by updating the user's points earned in the Customer table. The data that are deleted from the REST API are actually not deleted, but only not displayed to the user, using a simplified middleware of the Repository pattern for data management. Finally, we have the table EFMigrationsHistory which is the table generated by the application of Entity Framework managing the changes in our application and maintaining the database through the code where, apart from that, the code also knows the status of the database to manage the data inside it.

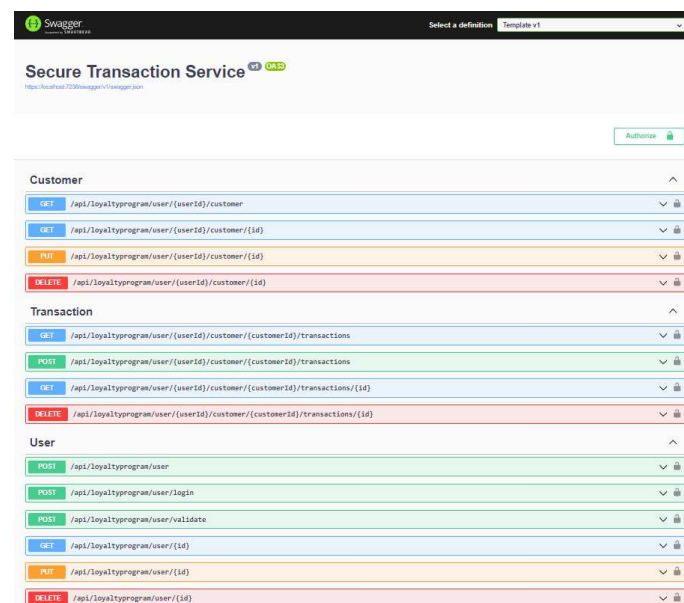


Figure 1. SWAGGER API structure.

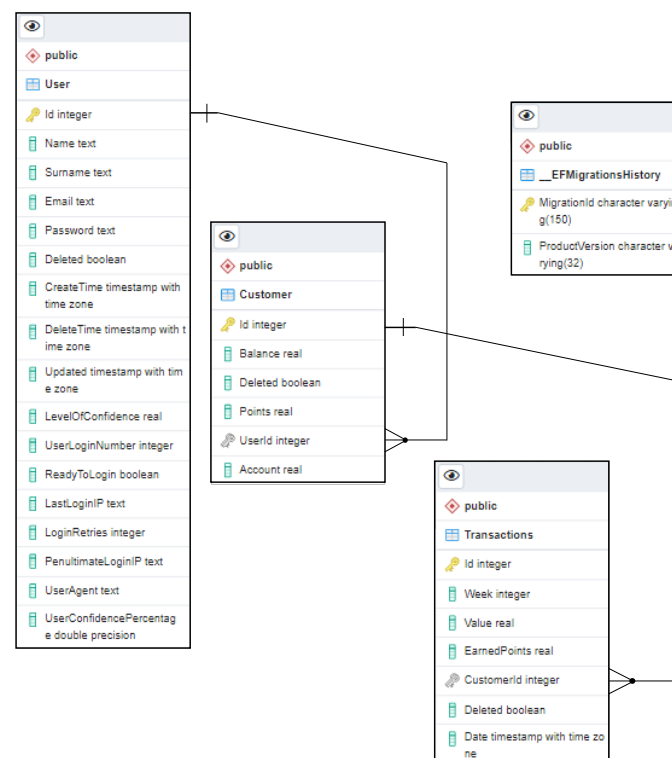


Figure 2. Enhanced Entity-Relationship Diagram (ERD) of the proposed model.

5.2. Proposed Method and Its Implementation

To access the platform, users must first register by providing their personal information. Upon successful login and validation, users can proceed to access the platform through authorization. Our paper focuses on the dynamic assignment of confidence levels by the user and the ease of access to personal resources based on their behavior while using the platform.

Once logged in, users can add funds to their balance, conduct transactions, and earn points for each transaction based on its value. Earned points will be added to the user's balance at the end of the week if certain transactions and terms are met. The user can also view transaction history, as presented in Figure 4 in Section 5.2.5, view the list of transactions with the corresponding points earned, and monitor the status of transactions. These and other functionalities will be further explained in subsequent chapters.

5.2.1. Registration

To gain access in our platform, one must first complete the registration process, and to be signed into our platform some data must be provided such as name, last name, and email, as presented on the Listing 2.

Listing 2. JSON request and response for the registration process.

```
Endpoint: /api/v1/loyaltyprogram/user
Method: POST
//JSON Request Body:
{
  "name": "string",
  "surname": "string",
  "email": "string",
  "password": "string",
  "confirmPassword": "string"
}

//JSON Response:
{
  "id": 0,
  "name": "string",
  "surname": "string",
  "email": "string"
}
```

5.2.2. Log In

The login to our platform to perform secure transactions happens dynamically and depends on the behavior of the user and how he tries to gain access by determining the level of credibility of himself and how our platform is currently calculating credibility, utilizing points, or employing a percentage of credibility, which will be discussed in more detail in next sections and also presented on the Listing 3.

5.2.3. User Credibility

Within the platform, we have two types of validation or access to the platform's resources since it is an internal process and is treated as two developed methodologies. The developed platform allows us to switch from one validation method to another without much trouble, where it is only designated as a switch from which the appSettings are read during startup and use of the platform for credibility calculations. These two types of validation methods are utilizing earned points and the percentage of the user's credibility which is earned through behaviors while logging into the platform.

Listing 3. JSON request and response for the log in process.

```

Endpoint:/api/v1/loyaltyprogram/user/login
Method:POST
//JSON Request Body:
{
    "email": "string",
    "password": "string"
}

//JSON Response is dynamic:
/*If the user is credible or reliable the user will get the access
   token instanlty:*/
{
    "jwtToken":{
        "id": 0,
        "authorizationToken": "string"
    }
}
/*item if the user is not credible the response in json would be:*/
"userChallengeNumbers":{
    "userId": 0,
    "number1": 0,
    "number2": 0,
    "number3": 0
}

```

The calculation of the user credibility is carried out according to three criteria: number of retries, IP of login, and User-Agent. The following shows how we calculate using these two methods in our platform.

5.2.3.1. Login by Means of Earned Points

- Number of Retries:
 - 0 retries: 0.7 points
 - 1 retry: 0.4 points
 - 2 retries: 0.2 points
 - 3 retries: 0 points and resets all your previous trust level to 0
- IP of Login:
 - Same IP as last login: 0.2 points
 - Different IP login: 0 points
 - 2 different IP logins: −0.5 points (minus)
- User-Agent:
 - Same browser: 0.2 points
 - Different browser: 0 points

The above explanation is mathematically represented in Equation (1):

$$\begin{aligned}
 & \left\{ \left[(0 \rightarrow 0.7) \parallel (1 \rightarrow 0.4) \parallel (2 \rightarrow 0.2) \parallel (> 3 \rightarrow 0 \text{ and reset}) \right]_{\text{retrials}} + \right. \\
 & \left[(\text{sameAsLast} \rightarrow 0.2) \parallel (\text{differentIP} \rightarrow 0) \parallel (\text{twoLastDifferentIP} \rightarrow (-0.5)) \right]_{\text{IP}} + \\
 & \left. \left[(\text{sameAsLast} \rightarrow 0.2) \parallel (\text{different} \rightarrow 0) \right]_{\text{userAgent}} \right\} \geq 3.5
 \end{aligned} \tag{1}$$

If the user gathers more than 3.5 points, he will be granted access directly without the challenge to gain access to the platform.

5.2.3.2. Login by Percentage of Credibility

- Number of Retries:
 - 0 retries: 70%
 - 1 retry: 40%
 - 2 retries: 20%
 - 3 retries: no percentage and returns all confidence level to 0%
- IP of Login:
 - Same IP as last login: 20%
 - Different IP login: no percentage
 - 2 different IP logins: restore confidence level to 0%
- User-Agent:
 - Same browser: 10%
 - Different browser: no percentage

If the percentage of users in the platform at the time of login is below 80%, then the percentage will be recalculated and a challenge will be generated by asking the user through email for the required value.

The percentage-base calculation is represented in Equation (2):

$$\left\{ \left[(0 \rightarrow 70\%) || (1 \rightarrow 40\%) || (2 \rightarrow 20\%) || (> 3 \rightarrow 0\% \text{ and reset}) \right]_{\text{retrials}} + \left[(\text{sameAsLast} \rightarrow 20\%) || (\text{differentIP} \rightarrow 0\%) || (\text{twoLastDifferentIP} \rightarrow 0\%) \right]_{\text{IP}} + \left[(\text{sameAsLast} \rightarrow 10\%) || (\text{different} \rightarrow 0\%) \right]_{\text{userAgent}} \right\} > 80\% \quad (2)$$

5.2.4. User Not Credible—Challenge Generation

Based on our credibility formulas, if the user credibility does not meet the criteria then a challenge will be generated, where our challenge foresees that the user's views contains three numbers that will appear, to be chosen by the user, the one sent to the user's e-mail is the correct one that needs to be chosen by the user in order to access the transaction server. Referring to the Figure 3, the user receives the number via registered e-mail.

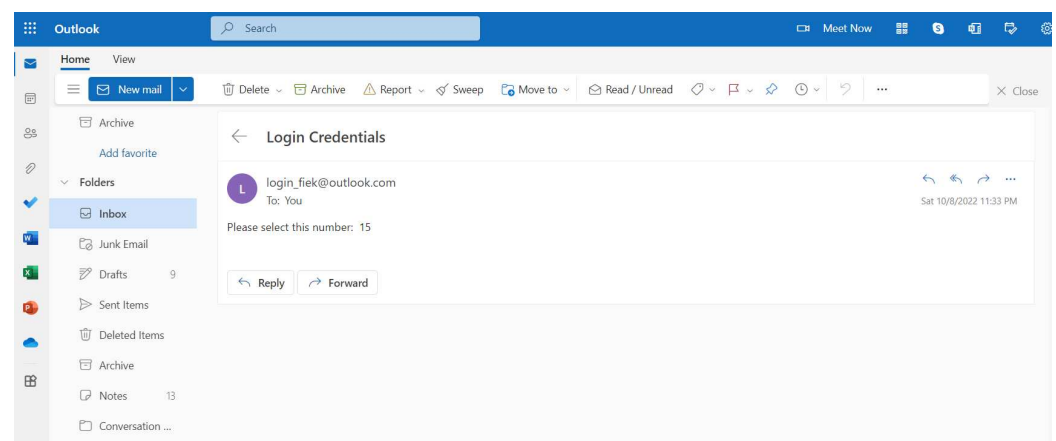


Figure 3. Email with the requested value.

By selecting the value which has been sent through e-mail in these endpoints and presented in Listing 4, the user will be granted with an access token valid for 5 min.

Listing 4. JSON request and response for acquiring token process.

```

Endpoint: /api/v1/loyaltyprogram/user/validate
Method: POST
// JSON Request Body:
{
    "userId": 0,
    "mailedValue": 0
}

// JSON Response:
{
    "id": 0,
    "authorizationToken":
        "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ..."
}

```

5.2.5. Transaction and CRUD Operations

Upon successful login, the user has access to four transaction service endpoints. These endpoints include the ability to initiate a secure transaction through a POST request, retrieve information on a specific transaction through a GET request using a unique *TransactionId*, view a list of transactions with pagination through another GET request, and delete a transaction through a DELETE request, which is only available to privileged users. The endpoint definitions for the transaction service can be found in Figure 4.

Transaction	
GET	/api/loyaltyprogram/user/{userId}/customer/{customerId}/transactions
POST	/api/loyaltyprogram/user/{userId}/customer/{customerId}/transactions
GET	/api/loyaltyprogram/user/{userId}/customer/{customerId}/transactions/{id}
DELETE	/api/loyaltyprogram/user/{userId}/customer/{customerId}/transactions/{id}

Figure 4. Transaction CRUD operations.**5.2.6. Winning Bonuses at the End of the Week**

Earning points at the end of the week is a process in itself completely separated from the API application which is exposed, this part has been treated as a background worker (Rewarding service is available on Github: <https://github.com/ahmetbucko/BackgroundWorker> [accessed on 11 February 2023]) which is a worker that is subscribed to the database and works in the background whom checks it every day if the criteria for date has been made, in our case whether it is the end of the week in particular Sunday which checks for each user if certain conditions have been met in order to gain the bonuses for their transactions for the last week. In the following section are presented the rules for each transaction bonuses and how they are being treated:

- Each earned point can be returned as a cent or EUR 0.01
- Each transaction value spent on a transaction will give the customer:
 - 1 point for each value up to EUR 5000 transaction value
 - 2 points for each value from EUR 5001 to EUR 7500 transaction value
 - 3 points for each value over EUR 7501 transaction value

After the calculation is finished and the point calculation worker starts, it must fulfill some conditions that the user must have before being ready to earn certain points:

- A user will lose all points if no transaction has been made in the last 5 weeks.
- The user has spent at least EUR 500 that week.

- At least one transaction exists for that user on each day of the week.

6. Critical Review on Results

During the research performed on this paper and the accompanying experimental setup, the process of granting access to the user was described. To gain access to the platform, the user must first complete registration and then undergo authentication. The authentication process is based on the user's level of trust or credibility, which is determined by their behavior on the platform, such as the number of login attempts, location, and device used. The measurement of credibility is performed using two distinct methodologies: point-based credibility measurement and percentage-based reliability measurement, which are discussed in greater detail elsewhere in the paper.

While both methods serve the same purpose of measuring user credibility, they differ in their approach. As indicated by their names, the primary difference between them is that the first method, point-based credibility measurement, assesses credibility through the awarding or deduction of points based on the user's behavior during authentication. The second method, percentage-based reliability measurement, is a normalized version of the first method that expresses credibility as a percentage. However, the differences between the two methods extend beyond the way in which credibility is measured. They also differ in the way users are penalized and the points earned, as discussed in more detail in subsequent chapters that delve into these methodologies.

The normalization method, expressed as a percentage, was derived from the point-based method of determining credibility to simplify the authentication process without compromising the security of the mathematical formula. The point-based method required the user to undergo multiple authentication steps to reach a sufficient level of trust for access without challenges. In the event of a major mistake, such as repeatedly entering incorrect credentials, the user's credibility would be reset to zero and the process of earning points would start from the beginning, which can be frustrating for the user.

The normalization method provides a more streamlined way of determining the user's credibility by considering factors such as entering the correct credentials on the first attempt, using the same IP and browser as the previous session, and fulfilling ideal authentication conditions. This method is easier to attain, but it can also be lost more easily through small mistakes such as accessing the platform from different locations or using different browsers. In contrast, the point-based method is more forgiving of small mistakes and only results in a loss of credibility for major changes in behavior or repeated errors.

7. Discussion and Conclusions

In this research paper, a novel approach to secure authentication was implemented. Rather than relying on traditional two-factor authentication methods, the application utilizes two mathematical calculations to continuously check and update the user's credibility or reliability. The user sets their own level of credibility based on their behavior while using the platform, and access is granted or restricted accordingly.

The focus of this research paper is on the development of two new mathematical calculators. The first calculator employs a point-based system and a percentage of credibility, allowing the user to earn or lose credibility through their behavior. The initial calculator used in the application was found to be insufficient as it took a significant amount of time for the user to gain credibility. As a result, a new authentication method was developed, derived from the point-based system and expressed as a percentage, which simplifies the process of gaining credibility. Initial testing results show that the percentage-based method makes the way of gaining the level of credibility more intuitive and easy for the user. It is also a more efficient way to calculate by the application itself, where the user gains/penalizes its behavioral state more dynamically.

Similarities exist between the suggested method for increasing user authentication in web applications based on behavior history and the Reinforcement Learning (RL) paradigm. In RL, an agent learns to make decisions by performing actions in a given environment to

accumulate the greatest reward over time. The agent's actions and subsequent rewards or punishments comprise a feedback loop that influences its future choices.

Similarly, our system promotes or punishes user behavior depending on parameters such as the number of failed password tries, the consistency of the IP address, and the user agent type. By accumulating these incentives and penalties over time, the system is able to create a more precise degree of user authentication trust. Our technique successfully adapts to users' behaviors, like an RL agent exploring and exploiting its surroundings to maximize its decision-making process.

To further support this relationship, we have researched meaningful reinforcement learning (RL) literature, such as Sutton and Barto's "Reinforcement Learning: An Introduction," Ref. [35] which explains the field's fundamental ideas. Our technique might benefit from adopting specific RL ideas into its design and execution, given these observations. For instance, the exploration-exploitation trade-off, a central concept in RL, might be considered when calculating the right weight or percentage to apply to various user activities, enhancing our trustworthiness judgment.

By understanding the core elements of RL and how they apply to real-world examples, we can better appreciate the connection between our proposed method for enhancing user authentication and the RL paradigm. As we continue to develop our approach, incorporating RL principles and techniques can potentially lead to more robust and adaptive solutions for securing user authentication in web applications.

In conclusion, the limitations of the proposed approach are: mobile users due to frequent changes of their IP addresses are punished by the proposed formula, as well as usage of a different browser being discouraged by the actual approach. Moreover, users continually updating their devices, automatic updates of the browsers, and even changing from one device to another device will penalize the user's behavioral state. The application is built to support other methods of calculating the users' credibility, not only the two above-mentioned methods, because it uses dependency injection. Essentially, this means only the method interface needs to be injected and the new method will be triggered inside the credibility calculator of the user. The application lacks a logging technology of the users' behavioral and application status. Currently, the state of user credibility is stored in the database. For the real-world scenario the best practice is to use a logging technology such as Elastic Search. The logs of the users' behavior will be sent to Elastic. Logged data will be retrieved and will calculate the users' current state of credibility when a user requests access. These constraints could serve as improvements for future work.

Author Contributions: Conceptualization, B.R.; Methodology, K.V.; Software, A.B.; Validation, B.K.; Formal analysis, A.B.; Data curation, A.B.; Writing—original draft, A.B., K.V., B.K. and B.R.; Writing—review & editing, B.K.; Supervision, B.R. All authors have read and agreed to the published version of the manuscript.

Funding: Ministry of Education, Science, Technology and Innovation, Government of Kosovo with Decision no. 2-814 dt. 15 June 2021 has funded this research.

Data Availability Statement: All data were presented in the main text.

Acknowledgments: Authors would like to thank you The Ministry of Education Science, Technology and Innovation of Kosovo, the Department of Computer Engineering from University of Prishtina and the Department of Informatics at the University of Oslo for support and cooperation.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Beaudin, S.; Levy, Y.; Parrish, J.; Danet, T. An empirical study of authentication methods to secure e-learning system activities against impersonation fraud. *Online J. Appl. Knowl. Manag.* **2016**, *4*, 42–61. [CrossRef]
2. Hitchman, S. The Details of Conceptual Modelling Notations are Important—A Comparison of Relationship Normative Language. *Commun. Assoc. Inf. Syst.* **2002**, *9*, 10. [CrossRef]
3. Imageware. Identification, Authentication, Authorization—What's the Difference. 2023. Available online: <https://imageware.io/identification-authentication-authorization-difference/> (accessed on 16 January 2023).

4. Rexha, B.; Lajqi, H.; Limani, M. Implementing data security in student lifecycle management system at the University of Prishtina. *Trans. Inf. Sci. Appl.* **2010**, *7*, 965–974. [\[CrossRef\]](#)
5. Alangot, B.; Szalachowski, P.; Dinh, T.T.A.; Meftah, S.; Gana, J.I.; Aung, K.M.M.; Li, Z. Decentralized Identity Authentication with Auditability and Privacy. *Algorithms* **2023**, *16*, 4. [\[CrossRef\]](#)
6. Abhishek, K.; Roshan, S.; Kumar, P.; Ranjan, R. A Comprehensive Study on Multifactor Authentication Schemes. In *Advances in Computing and Information Technology; Advances in Intelligent Systems and Computing*; Meghanathan, N., Nagamalai, D., Chaki, N., Eds.; Springer: Berlin, Heidelberg, 2013; Volume 177, pp. 561–568. [\[CrossRef\]](#)
7. Vishi, K. Security and Privacy in User Authentication: Aspects of Fusion, Machine Learning, and Privacy in Biometric Authentication. Ph.D. Thesis, Department of Informatics, the Faculty of Mathematics and Natural Sciences, University of Oslo, Oslo, Norway, 2023.
8. Musa, A.; Vishi, K.; Rexha, B. Attack Analysis of Face Recognition Authentication Systems Using Fast Gradient Sign Method. *Appl. Artif. Intell. Int. J.* **2021**, *35*, 1346–1360. [\[CrossRef\]](#)
9. Lal, N.A.; Prasad, S.; Farik, M. A review of authentication methods. *Int. J. Sci. Technol. Res.* **2016**, *5*, 246–249.
10. Stamp, M. *Information Security—Principles and Practice*, 2nd ed.; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2011; ISBN 978-0-470-62639-9.
11. Kornienko, D.V.; Mishina, S.V.; Shcherbatykh, S.V.; Melnikov, M.O. Principles of securing RESTful API web services developed with python frameworks. *J. Phys. Conf. Ser.* **2021**, *2094*, 032016. [\[CrossRef\]](#)
12. Franks, J.; Hallam-Baker, P.; Hostetler, J.; Lawrence, S.; Leach, P.; Luotonen, A.; Stewart, L. RFC 2617—HTTP Authentication: Basic and Digest Access Authentication; Technical report; Internet Engineering Task Force (IETF): Fremont, CA, USA, 1999. [\[CrossRef\]](#)
13. Okta. Authentication Protocols 101: Definition, Types, and When to Use. 2022. Available online: <https://www.okta.com/identity-101/authentication-protocols/> (accessed on 14 January 2023).
14. Mohammad, A.; Al-Refai, H.; Alawneh, A.A. User Authentication and Authorization Framework in IoT Protocols. *Computers* **2022**, *11*, 147. [\[CrossRef\]](#)
15. Onelogin. How Does Single Sign-On Work? Available online: <https://www.onelogin.com/learn/how-single-sign-on-works/> (accessed on 10 January 2023).
16. Sharif, A.; Ranzi, M.; Carbone, R.; Sciarretta, G.; Marino, F.A.; Ranise, S. The eIDAS Regulation: A Survey of Technological Trends for European Electronic Identity Schemes. *Appl. Sci.* **2022**, *12*, 12679. [\[CrossRef\]](#)
17. Onelogin. SAML Explained in Plain English. Available online: <https://www.onelogin.com/learn/saml/> (accessed on 10 January 2023).
18. Hughes, J.; Maler, E. Security assertion markup language (saml) v2.0 technical overview. *OASIS SSTC Work. Draft Sstc-Saml* **2005**, *13*, 1–51
19. ForgeRock. OpenID Connect 1.0 Guide. Available online: <https://backstage.forgerock.com/docs/am/5.5/AM-5.5-Oidc1-Guide.pdf/> (accessed on 29 December 2022).
20. Hammer-Lahav, E. RFC 5849—The OAuth 1.0 Protocol; Technical report; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2010. [\[CrossRef\]](#)
21. Gashi, E.; Rexha, B.; Rexhepi, A. Trust establishment between OAuth 2.0 resource servers using claims-based authorization. *Electron. Gov. Int. J.* **2021**, *17*, 3. [\[CrossRef\]](#)
22. Halder, S. How OAuth Boosts API Security and Access Management. 2021. Available online: <https://nordicapis.com/how-oauth-boosts-api-security-and-access-management/> (accessed on 5 January 2023).
23. Zamfiroiu, A.; Constantinescu, D.; Zurini, M.; Toma, C. Secure Learning Management System Based on User Behavior. *Appl. Sci.* **2020**, *10*, 7730. [\[CrossRef\]](#)
24. Saleh, G.; Tharwat, G.; Gamalel-Din, S. A Systematic Survey on Examinees Identity Authentication in Online Distant Exams. *J. Al-Azhar Univ. Eng. Sect.* **2023**, *18*, 129–151. [\[CrossRef\]](#)
25. Catalin, B.; Alin, Z.; Madaline, Z.; Bogdan, I. User Behavior Profiling in Social Media Applications. *Econ. Comput. Econ. Cybern. Stud. Res.* **2019**, *53*, 21–38. [\[CrossRef\]](#)
26. Yu, C.; Yang, Y.; Wei, Z.; Junyi, S. Analyzing User Behavior History for constructing user profile. In Proceedings of the 2008 IEEE International Symposium on IT in Medicine and Education, Xiamen, China, 12–14 December 2008. [\[CrossRef\]](#)
27. Sandhu, R.; Coyne, E.; Feinstein, H.; Youman, C. Role-based access control models. *Computer* **1996**, *29*, 38–47. [\[CrossRef\]](#)
28. Sandhu, R.; Ferraiolo, D.; Kuhn, R. The NIST Model for Role-Based Access Control: Towards a Unified Standard. In Proceedings of the Fifth ACM Workshop on Role-Based Access Control, RBAC '00, Berlin, Germany, 26–28 July 2000; Association for Computing Machinery: New York, NY, USA, 2000; pp. 47–63. [\[CrossRef\]](#)
29. Caruccio, L.; Deufemia, V.; D'Souza, C.; Ginige, A.; Polese, G. A Tool Supporting End-User Development of Access Control in Web Applications. *Int. J. Softw. Eng. Knowl. Eng.* **2015**, *25*, 307–331. [\[CrossRef\]](#)
30. Thomas, R.K.; Sandhu, R.S. Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented authorization management. In *IFIP Advances in Information and Communication Technology, Proceedings of the Database Security XI, Lake Tahoe, CA, USA, 10–13 August 1997*; Lin, T.Y., Qian, S., Eds.; Springer: Boston, MA, USA, 1997; pp. 166–181. [\[CrossRef\]](#)
31. Giordano, M.; Polese, G. Visual Computer-Managed Security: A Framework for Developing Access Control in Enterprise Applications. *IEEE Softw.* **2013**, *30*, 62–69. [\[CrossRef\]](#)

32. Zhang, N.; Ryan, M.; Guelev, D.P. Synthesising Verified Access Control Systems in XACML. In Proceedings of the 2004 ACM Workshop on Formal Methods in Security Engineering, FMSE '04, Washington, DC, USA, 29 October 2004; Association for Computing Machinery: New York, NY, USA, 2004; pp. 56–65. [[CrossRef](#)]
33. Heydon, A.; Maimone, M.; Tygar, J.; Wing, J.; Zaremski, A. Miro: Visual specification of security. *IEEE Trans. Softw. Eng.* **1990**, *16*, 1185–1197. [[CrossRef](#)]
34. Barker, R. *CASE Method: Entity Relationship Modelling*; Number v.1 in CASE method; Addison-Wesley: Boston, MA, USA, 1990; ISBN 9780201416961.
35. Sutton, R.; Barto, A. *Reinforcement Learning, Second Edition: An Introduction*; Adaptive Computation and Machine Learning series; MIT Press: Cambridge, MA, USA, 2018; ISBN 0262039249.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.