Protection of the frontend and backend using Azure Active Directory authentication

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Bachelor thesis

Degree Program in Computer Applications
spring, 2023

Roman Konstantinov

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Computer Applications Abstract

Author Roman Konstantinov Year 2023

Subject Protection of the frontend and backend using Azure AD

Supervisors Elina Vartiainen

ABSTRACT

The thesis sought to find and implement the methods that Azure Active Directory platform provides to secure web applications. The main point is to find effectiveness of such methods as well as understand advantages and limitations of Azure as a cloud platform and the services it provides.

The thesis was commissioned by Netum Oy which currently is utilizing Azure Active Directory and wants to know what they can do better to secure their applications.

First, the theory part will answer all the raised research questions and the thesis will proceed by introducing development project and implementing security methods that were discussed in the theory part. The primary research method was to look up related topics on the official Microsoft Learn portal, then data were analyzed by relevance and described in the thesis.

The research and development project demonstrates what and how Netum Oy can implement security methods for their application according to their specific needs.

Keywords Azure Web Applications Security Methods

Pages 35 pages and appendices 01 pages

Glossary

Azure AD	Azure Active Directory
JWT	JSON Web Token
SSO	Single sign-on
MSAL	Microsoft Authentication Library
URL	Uniform Resource Locator
VPN	Virtual Private Network
API	Application Programming Interface
SPA	Single-page application
DOM	Document Object Model
PHS	Password hash synchronization
PTA	Pass-through authentication

Contents

Annexes

Annex 1 Material management plan

1 Introduction

The thesis will dive into methods of securing both front & back -end of web applications using Azure Active Directory (Azure AD) authentication. It will explore different practices of securing web apps, their advantages, and disadvantages as well as advantages and limitations of Azure AD. This work is intended for individuals, companies, universities and for whomever is interested in improving their web applications using Azure AD.

The work will intentionally leave out detailed technical information about programming and installation, instead focusing on providing a conceptual understanding of Azure AD authentication and its use cases.

The purpose of the practical part is to demonstrate different methods of security as well as producing the highest possible level of security. For this part, a small web application will be developed which will include a React with TypeScript frontend and Node.js with Express.js utilizing MongoDB. The thesis theme was suggested by Netum OY which is interested in securing their systems using Azure AD.

Research questions:

- How effective is Azure Active Directory authentication for securing front & back -end of web applications?
- What are the key advantages and limitations of using Azure Active Directory authentication compared to other authentication methods?
- What are the best practices for implementing and managing Azure Active Directory authentication to ensure a high level of security?

2 Azure and Azure Active Directory

This chapter will uncover the history behind Azure and Azure Active Directory creation, will tell the difficulties and barriers on the way of becoming one of the best cloud platforms on the current market.

2.1 Azure

Azure is a cloud platform developed by Microsoft. Firstly, announced on October 28, 2008, as a cloud computing operating system for Businesses and Developers without additional coding. With the original name “Windows Azure” it was presented as a response in competition with Amazon EC2 and Google App Engine. Roosevelt Abandy (2022) summarizes early Windows Azure accurately: “Windows Azure was built from as an extension of the Windows NT which was the beginning of Microsoft Cloud Platform as a Service (PaaS).” (Roosevelt Abandy, 2022)

2.1.1 The First-Generation Services

The First iteration of Windows Azure included limited functional, but it started formation of the Azure cloud service as we know today. The first version allowed developers to run the ASP.NET web applications and APIs, as well as running long processes with no User Interface. (Roosevelt Abandy, 2022)

Windows azure became available in the early year of 2010. At the time of release, Windows azure supported Java, PHP, micro related services, .NET Framework 4 with the support of Microsoft SQL Server. (Roosevelt Abandy, 2022)

2.1.2 The Second-Generation Services

Open-Source Software became more popular among the developers and with that Linux virtual machines, MySQL, PHP, and Apache became more used. Amazon`s “Amazon EC2” was growing rapidly which made Microsoft re-strategies Windows Azure. After a reform of their strategy, Microsoft was able to revolutionaries the Windows Azure, firstly by renaming it to “Microsoft Azure” and making Microsoft Azure the best place to run Linux which meant the transformation

from bottom to top from PaaS (Platform as a Service) to IaaS (Infrastructure as a Service). (Roosevelt Abandy, 2022)

2.1.3 The Third-Generation Services

When the era of big data, analytics, and Internet of Things (IoT) came, Amazon were pushing their EMR. Amazon EMR became one of the first “Big Data” cloud platform which allowed developers to scale up their data processing jobs. Amazon EMR provided use of machine learning applications with the use of Apache Spark, Apache Hive, and Presto and introduced interactive SQL queries. While Amazon were developing “Amazon EMR”, Google pushed BigQuery as the data warehouse into the cloud. (Roosevelt Abandy, 2022)

Microsoft partnered with “Hortonworks” to enable Azure HDInsight and a managed Apache Hadoop service in the Microsoft cloud. To provide end-to-end Big Data and analytics on Azure, Microsoft launched Azure Data Lake Store and Azure Data Lake Analytics and later acquired “Revolution Analytics” to make R language native to Azure data platform. (Roosevelt Abandy, 2022)

Microsoft saw the potential behind Internet of Things (IoT) and started making huge investment in it, which on the long run helped Microsoft to be one of the first cloud provider to offer end-to-end connection powered by Event Hub, IoT Hub, Stream Analytics, SQL Database and Power BI. (Roosevelt Abandy, 2022)

2.1.4 The Fourth-Generation Service

The Fourth iteration of Azure finally brought Microsoft leaderships against other cloud service providers, but this was only the start of competition between cloud service providers for the customers. Microsoft started investing very early into Machine Learning (ML) and Artificial Intelligence (AI) which made Microsoft among the first cloud providers offering an easy-to-use Machine Learning visual designer named Azure ML Studio” Azure ML Studio supported deep learning models, NVIDIA GPU (Graphics processing unit), Intel FPGA (Field-programmable gate array), enhanced pipelines, Machine Learning Operations and drag and drop designers for training neural networks. (Roosevelt Abandy, 2022)

Thanks to partnerships with big giants like Intel, Nvidia and Qualcomm and investment into databases, Big Data, AI and IoT made Microsoft Azure platform set the standards for running compute, storage, and analytics at the edge. (Roosevelt Abandy, 2022)

2.1.5 The present and Beyond

With the adoption of Kubernetes, Microsoft was forced to take advantage of the trend and launched the “Azure Arc” which allows customers to work with Virtual Machines (VMs), physical or on-premises machines and other workloads from a single control panel. (Roosevelt Abandy, 2022)

Currently, Microsoft Azure offers over 600 services and provides Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) technologies. (Roosevelt Abandy, 2022)

2.2 The history of Azure Active Directory

Before diving into Azure Active Directory Security capabilities, the history of Azure Active Directory formation is a must knowledge to understand the topic.

2.2.1 Windows Active Directory

Windows Active Directory is a predecessor to Microsoft Azure Active Directory released in 2000 with the Windows 2000 Server edition.

Even though Windows AD and Azure AD are both IAM (Identity & Access Management) systems, they are quite different. Azure AD is an upgraded version of Windows AD which provides Identity as a Service (IaaS). IaaS makes sure that the one logging in are who they say they are. (Michael Buckbee, 2022) (Table 1)

Table 1 Key differences between Windows AD and Azure AD (Michael Buckbee, 2022)

	Azure AD	Windows AD
Communication	Representational State Transfer (REST) APIs	Lightweight Directory Access Protocol (LDAP)
Authentication	Cloud-based protocols	Kerberos and New Technology LAN Manager (NTLM)
Network Organization	Flat structure of users and groups	Organizational units, domains, and forests
Devices	Mobile device management	No mobile device management
Desktops	Windows desktops can join with Microsoft Intune	Desktops are governed by Group Policy (GPOs)
Servers	Uses Domain Services to manage servers	Managed by GPOs or other on-premises server management system

2.2.2 Azure Active Directory

Azure Active Directory is the lates solution from Microsoft that provides cloud-based identity and access management solutions. Azure AD is heavily integrated into Microsoft products, for example, if the user uses Microsoft Teams, they are accessing it through Azure AD authentication via OAuth. (LogicMonitor, n.d)

Many companies in the world uses hybrid model for work, though it is convenient it creates new security concerns. Each team member must access resources for their work despite their location.

Azure AD offers organization identity-based authentication, that way each member of the team has the right privileges. Azure Active Directory provide a huge variate of benefits including but not limiting to simplified access to multiple applications using one login, easy reset password procedure, multi-factor authorization and Conditional Access manager, guest invite support, real-time monitoring, and analytics. The platform is constantly developing, new features are added and existing ones updating regularly. (LogicMonitor, n.d)

3 Azure Active Directory Security

Azure is a big cloud computing service that supports numerous programming languages, operating systems, frameworks, databases, and different devices. It supports Linux containers that can be run in Docker, build apps with JavaScript, Python, .NET, PHP, Java and Node.js as well as back-end apps for IOS, Android and Windows devices. Azure supports the same technologies that millions of people in the IT industry use every day. All that needs to be secured somehow, this is where Azure, especially shines. Azure AD offers six built-in functional areas: Operations, Applications, Storage, Networking, Compute, and Identity. (Microsoft Learn, 2022 -a)

There are only few clouds computing provides that includes a wide variety of security methods like Identity Protection, Conditional Access, Multi-factor authentication and more but also a consistent application platform and IaaS, which is what making Microsoft Azure the best choice for the businesses and individuals. Azure AD is the latest security management system developed by Microsoft, which includes a single identity for each user of your organizations, Single sign-on access to applications, rule-based Multi-Factor Authentication, and secure remote access to on-premises web applications. (Microsoft Learn, 2023 -b)

Azure AD provides access to security monitors, alerts, and machine learning reports, you can view anomaly reports – risk-based sign in events, application errors – shows usage date of your applications, error reports – shows errors which happens when inviting guests to your application, user-specific reports – shows sign-in activity of a specific user and activity logs – shows a records of all activities, changes, request made by users withing last 24 hours, 7 days or last 30 days. (Microsoft Learn, 2023 -b)

Though Azure has vast variation of security functionalities, this thesis will cover only parts from the vast variation of security methods available in Azure Cloud Services.

3.1 Data security and encryption

Since all the processes happen in the cloud, protecting and encrypting the data is one of the most important tasks for every business. Azure recommends building your data encryptions around two states: at-rest and in-transit.

At-rest – is a symmetric encryption which is used to quickly encrypt and decrypt copious amounts of data. At-rest makes sure that the data which is stored on a physical drive is secured, so if someone would be able to steal the hard drive from the server, they will not be able to access the data stored on it. Hacking such an encryption will require a lot of time and resources which in the end may result in corrupted data, for this reason it is highly recommended to use at-rest, especially if working in a security industry or government organization. To access data encrypted by Rest, data encryption keys and key encryption need to be provided, in Azure environment they all stored in Azure Key Vault. (Microsoft Learn, 2023 -c)

In-transit – the data is always moving from one location to another, so it is recommended to always use SSL or TLS protocols when moving the data. For an additional layer of security, you may also use a VPN. (Microsoft Learn, 2022 -b)

3.2 Azure Key Vault

In real-time application, it is not a new situation when it is needed to store API keys, passwords, certificates, or any other keys securely, that is where Azure Key Vault comes in especially handy. Vaults and managed hardware security modules pools are the two containers supported by Azure Key Vault. (Microsoft Learn, 2022 -d)

Before accessing the Key Vault, authentication needs to be completed, general recommendation from Microsoft is to use Managed identities for Azure resources which managed when deploying application to App Services and assigning deployed application to have access to Key Vault. To ensure the best security of your Key Vault, it is possible to limit which IP addresses have access to Key Vault, setup Virtual Network services endpoints, configure firewall or using Azure Private Link Service. When Key Vault is created, it connects to current Azure AD tenant and that way it ensures

that to access the Vault, all users must be in the said tenant and have right access level. (Microsoft Learn, 2022 -d)

Azure Key Vault supports logging, which enables administrators to monitor what vaults were accessed, by whom and when. Since all Vaults are secured, it takes time, usually around 10 minutes, to access Vault monitoring data but it is also possible to combine Azure Monitor logs to show Key Vault logs. (Microsoft Learn, 2022 -d)

3.3 End-to-end security in Azure

Azure provides a wide variety of security tools and capabilities which overall help create secure solutions in Azure. Confidentiality, integrity, availability of customer data with transparent accountability, which is all why Azure is on the top regarding security. (Microsoft Learn, 2022 -e)

Thesis focuses more on Azure AD security which includes Microsoft cloud-based identity and access management service, Conditional Access, Domain Services, Privileged Identity Management, Multi-factor authentication and Azure AD Identity Protection. (Microsoft Learn, 2022 -e)

3.3.1 Conditional Access

Conditional access is a way for Azure AD to make decisions, simply put it is an if-then statement. For example, if the user wants to access one of the services, then the user needs to complete specified actions, e.g., sign in using multi-factor authentication. Conditional access gives administrations powers to keep their organization secure by applying access controls. (Microsoft Learn, 2022 -f)

Conditional access works by bringing together signals like IP location information or risky sign in and then decides to allow access or not or what steps needs to be done before granting the access. (Microsoft Learn, 2022 -f)

3.3.2 Domain Services

For legacy applications that do not support modern authentication methods, Azure provides Domain Services to help resolve that issue. Domain Services provides domain join, group policy, lightweight directory access protocol, Kerberos/NTLM authentication. (Microsoft Learn, 2022 -g)

It works by deploying two windows server domain controllers into selected Azure region, which is known as a replica set. All the configuration is done by the Azure platform including backups and encryption. By performing a one-way synchronization with Azure AD, managed domain can access a set of users and groups. (Microsoft Learn, 2022 -g)

3.3.3 Privileged Identity Management

Privileged Identity Management provides tools to manage, control, and monitor access to vital resources of an organization. It is used to minimize unauthorized access to secure information by providing time-based and approval-based role activation. PIM allows to receive notification when privileged roles activates, show justification of users activations, setup mandatory multi-factor authentication. PIM managed only by users who is part of the Privileged Role Administrator or Global Administrator groups. It also enables a feature to invite guests/contributors and provide them with access to the organization resources. (Microsoft Learn, 2022 -h)

3.3.4 Multi-factor authentication

Multi-factor authentication or 2FA is an additional step of sign in process which require the user to enter a code from the SMS, Authentication App (e.g., Google Authenticator) or a Biometrical authentication (fingerprint, face id). 2FA adds an additional layer of security to the application and protects it from weak, exposed, or hacked passwords. Azure AD Multi-Factor Authentication provides a variety of forms of verification like Microsoft Authenticator, SMS, Voice call, OATH software token as well as a self-service for a password reset in one step. (Microsoft Learn, 2022 -i)

3.3.5 Hybrid identity

In today's world, applications are becoming increasingly a combination between front-end and back-end which indicates that the end-user needs to utilize both resources. Azure Hybrid identity allows to communicate with all the resources by creating and using a single identity. It is possible thanks to provisioning and synchronizations, provisioning responsible for creating, updating, and deleting the object on predefined conditions while synchronization responsible for keeping identity information up to date with the cloud services. (Microsoft Learn, 2023 -k)

3.3.6 Password hash synchronization

Password hash synchronization is a sign-in method used in Azure AD to achieve hybrid identity. PHS uses Azure AD Connect which synchronizes a hash of user's password from an on-premises AD instance to a cloud-based Azure AD instance, which allows for users to use only one password for every application, and it results in increased user productivity. PHS includes leaked credential detection, which was achieved working alongside dark web researchers and law enforcement. (Microsoft Learn, 2023 -l)

3.3.7 Pass-through authentication

Pass-through authentication is another Azure AD sign-in method. PTA as well as PHS allows to use only one password to access all application, while PHS allows only to access on-premises AD instances, PTA allows to access on-premises and cloud-based applications. Otherwise, it provides the same functionality as PHS and more, one of the key features, which will be used in practical part of this thesis, is Seamless single sign-on. (Microsoft Learn, 2023 -m)

3.3.8 Federation

When two businesses start to work together, they need a way to securely access common resources, federation for the rescue. Federation is a list of businesses that trust each other, the allowed access may be different but usually it is an authentication with authorization. Administrations could also setup a stricter access control. Federation also allows to setup PHS as a backup sign-in method. (Microsoft Learn, 2023 -n)

3.4 Microsoft identity platform

To start building applications in Azure Active Directory platform, it is needed to understand identity platform concept. Microsoft identity platform helps build applications which will use Azure authentication and allow users and customers to easily sign in using their personal Microsoft credentials. It provides tools to give control access to businesses APIs and utilize Microsoft Graph API. (Microsoft Learn, 2023 -o)

Microsoft identity platform includes several components:

- OAuth 2.0 and OpenID Connect standard-compliant authentication – allows authentication of work or school accounts, personal Microsoft accounts (Skype, Xbox, Outlook) and social or local accounts.
- Open-source libraries – one of which is Microsoft Authentication Library that will be used in the practical part of this thesis.
- Application management portal – enables registration and configuration of our applications.
- Application configuration API and PowerShell – for configuring our application through Microsoft Graph API for the purpose of automating DevOps tasks (Pipelines, Git repositories, Releases.)
- Developer content – includes documentation, QuickStart guides and general guides.

The huge advantage of Microsoft identity platform is flexibility and included modern technologies such as password less authentication, step-up authentication, and Conditional Access.
Identity platform supports various application types from SPA, protected web APIs, daemon app to desktop and mobile applications. (Microsoft Learn, 2023 -o)

3.5 Identity Protection

Identity protection is a system designed by Microsoft which is based on years of research acquired from organizations that use Azure Active Directory, personal Microsoft Accounts, and Xbox gaming and it is all done for the sole purpose of protecting users. Three crucial tasks are made possible for businesses by utilizing identity protection: automation detection and remediation of identity-based risks, risk investigation and exporting risk detection data to other tools. By analyzing trillions

of signals per day, identity protection helps to spot risks and protect customers from them. (Microsoft Learn, 2022 -p)

It detects risk like atypical travel, password spray, leaked credentials, or unfamiliar sign-in properties, and then triggers remediation, additional layer of security that the user would need to complete before accessing the resource (e.g., password reset, multi-factor authentication).
All detections are stored in the Azure platform and allow administrators to review them and take additional actions. (Microsoft Learn, 2022 -p)

Identity Protection filters all risk by tiers: low, medium, and high, though Microsoft does not provide any details of how risks calculated it brings more clarity to what is wrong. As already mentioned, risk data can be exported to other tools which will help to further study the data. (Microsoft Learn, 2022 -p)

3.6 Risk-based access policies

Azure AD provides two risk conditions “Sign-in risk” and “User risk.” For dealing with these types of risk, administrator creates Conditional Access depending on the risk level and selects what method would be used to control the access. Whenever the user attempts to sign-in, Identity Protections scans and analyzes a vast number of signals and then proceeds to calculate the risk level, if risk was detected Identity Protection sends a risk level to Conditional Access, which then manages it and applies security policies. (Microsoft Learn, 2022 -q)

Risk-based policy also includes automatic risk remediation, which takes off the risk level from their sign-in when the user completes the mandatory sign in steps. Automatic risk remediation helps to reduce the burdens of manual work for administrators. (Microsoft Learn, 2022 -q)

Identity Protection can be used alone but Microsoft recommends using it with Conditional Access since it ensures high security of your application. (Microsoft Learn, 2022 -q)

3.7 Securing PaaS web and mobile applications

Azure provides a platform-as-a-service portal called “App Service” which allows organizations to host their web and mobile applications in the Azure environment. Authentication in App Services happens by utilizing Azure Active Directory and OAuth 2.0 service. Access to App Services can be restricted by user role or by IP address and it also includes Azure Key Vault. (Microsoft Learn, 2022 -r)

3.8 Microsoft identity platform basics

This chapter will cover the basics of identity platform and access the questions: What is Authentication and Authorization? What are Security tokens? and What is Microsoft Authentication Library? This is crucial for understanding the practical part of the thesis.

3.8.1 Authentication and Authorization

It might seem that these two things are the same but that is not true. Authentication checks credentials and verifies if the user that logs in is the one who he says he is, it verifies the identity of the person or device by utilizing OpenID Connect. Authorization on the other hand, grants permissions to authenticated users by utilizing OAuth 2.0. (Microsoft Learn, 2022 -t)

3.8.2 Security Tokens

Security tokens are a set of different tokens that includes access tokens, refresh token and ID token which allows end-user to utilize or get access to various resources, e.g., Sending request to API. (Microsoft Learn, 2022 -u)

Access token – as part of the OAuth 2.0, an authorization server issues a token that contains information about both the user, and the resource. Refresh token – since access tokens are short-lived (default is 1 hour), authorization server generates a refresh token which can be used to retrieve new access token. ID token – issued as part of the OpenID Connect and can be used with or instead of access token. ID tokens used to authenticate the users. (Microsoft Learn, 2022 -u)

3.8.3 Microsoft Authentication Library

Microsoft Authentication Library is a JavaScript package developed by Microsoft. MSAL enables developers to easily setup and implement Azure authentication into projects, it works both on front and back -end and allows to acquire Security Tokens as well as making calls to Microsoft Graph API. It requires a bit of setup including creating of Authentication Configuration file in the root of the project and registering application into tenant through “App Registration” service. (Microsoft Learn, 2022 -v; Azure AD GitHub 2021.)

3.9 Single Page Applications

The secret behind Single Page Applications is in the naming, it is an application that rewrites the content of the page rather than rendering a whole new page from another file. Thesis front-end part of the project is a Single Page Applications but before starting to write code, necessary prerequisites need to be configured in the Azure Active Directory. This chapter will explain and cover what configurations need to be done and why. (Katie Lawson, 2022)

3.9.1 App registration

The first step required to start using Azure Active Directory in any of the applications is App Registration. App registration provides tools to manage and control access to applications, set up authentication points, configure access token and what data token will return, manage groups and roles and much more. (Microsoft Learn, 2022 -w)

To register application, simply navigate to Azure portal, open Azure Active Directory settings, find “App registration” and at the top click “New registration” button. Mandatory information needs to be filled in before completing the registration, after that click “Register” and application registration will be completed and prepared for the next step. (Microsoft Learn, 2022 -w)

3.9.2 Authentication

Before anyone would be able to sign into application using their Microsoft credentials authentication setup required. To set up authentication, navigate to App Registration settings,

click on “Authentication” menu, select “Add platform” and select app platform, in case of this thesis it is SPA. Redirect URLs are required to be configured, such URL will redirect the user after they complete the authentication. Usually, the Redirect URL is the main page of the application, but it all depends on business needs. Authentication configuration completed and the next step is token configuration. (Microsoft Learn, 2022 -x)

3.9.3 Token Configuration

Token configuration is used to configure what additional information will be included with the token. It is needed for security purposes, for example receiving IDs of groups that the user belongs to, which helps to identify users group and what permissions users will have in the app. (Microsoft Learn, 2022 -x)

In case of this thesis, the token will be configured to include users group IDs but only the groups that assigned to the application. (Microsoft Learn, 2022 -x)

3.10 Authentication methods

Azure authentication in App registrations supports up to 3 different ways of authenticating end-users: (Microsoft Learn, 2022 -y)

1. Redirect – after pressing the sign in or sign out button, the user will be redirected to the Microsoft authentication page. (Microsoft Learn, 2022 -y)
2. Popup – after pressing the sign in or sign out button, popup window will be shown to users with the opened Microsoft authentication page. (Microsoft Learn, 2022 -y)
3. SSO – if a used browser supports third party cookies and the user has already signed into Azure services, then SSO will automatically sign in user to the application without any user interactions. (Microsoft Learn, 2022 -y)

4 Aim and purpose of the development work

The thesis is trying to answer the raised questions about Azure Active Directory security by uncovering the history behind the Azure and Azure AD, describing various security methods using simple language for the purpose of everyone who is interested in using Azure in their business or personal project to understand what methods of securing application they can use.

To showcase on practice some of the basics way of securing front-end and back-end web applications, the development of project was chosen as a method. Development project utilizes only those security methods that the thesis described.

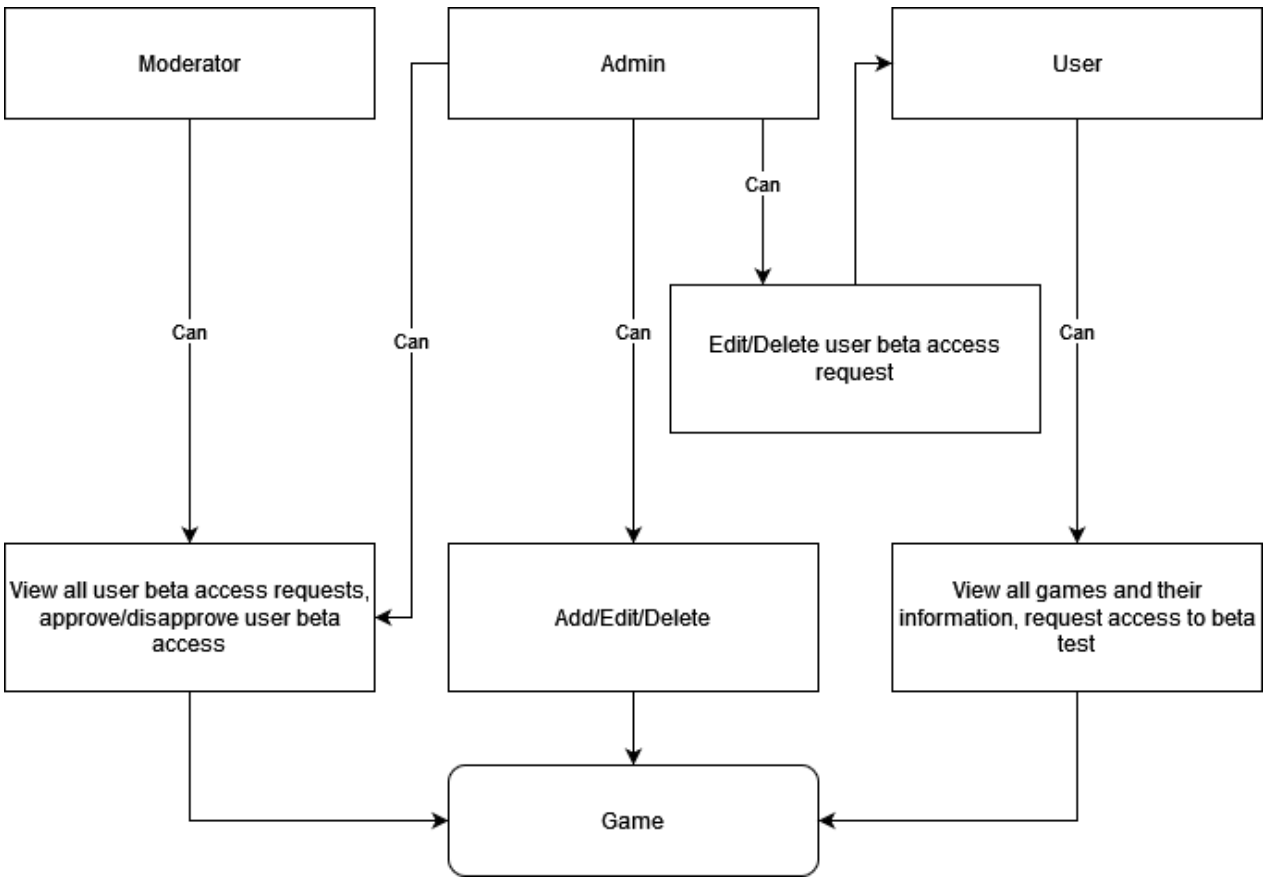
The waterfall project model was used to develop the project, firstly the idea was described, then prerequisite and the design of the front-end and back-end was created which led by developing and testing the final version of the project.

5 Video Game Beta Access project

To demonstrate protection of the web application using Azure Active Directory a simple Single-Page Application will be written. It will include both the back end and the front-end of the application, with Azure AD setup, MSAL setup and methods of protection.

The idea of application is a game development studio asked to create a system that will allow their user-base easily view information about their latest games and be able to request an access to the beta test of the game. Also, they required to create a control panel for admin and moderator. The admin would be able to add new games, edit game, delete game, view all users' registrations, approve, or disapprove user access to the game, edit or delete user, while moderator would be able to view all user registrations and approve or disapprove user beta access. (Figure 1)

Figure 1 API users possibilities scheme



5.1 Prerequisite

Both the front-end and back-end will be using Node.js 18. The front-end would be written on React framework with TypeScript and utilizing React-Router for handling routing, MSAL for handling Azure AD Authentication and other packages for UI elements. The back end would be written on Node.js using Express.js, MongoDB for noSQL database, JSON web token for handling Azure token verification and other helpful packages for handling API.

5.2 Common solutions

Before diving deeply into writing front end and back end, a description of common solutions is required. Both front end and back end will be utilizing JSON Web token and Environmental variables during their work.

5.2.1 JSON Web token

JSON Web token is a digitally signed key that is used to securely transfer information between places. Essentially, it is a JSON object that includes information, for example the user data. There are two ways to sign the token, by using HMAC algorithm which will mean that the token is signed by the secret or using a public/private key pair with Rivest-Shamir-Adleman cryptosystem or Elliptic Curve Digital Signature Algorithm. (JWT, n.d.)

Commonly, JSON web tokens are used for authorization, when the user logins, for example to Azure environment, JWT is generated and is required to be provided for every secured action like accessing protecting API route or resource. Typically, JWT looks like this xxxxx.yyyyy.zzzzz where X is a header, Y is a payload and Z is a signature. (JWT, n.d.)

In this thesis, front-end will be sending to back-end an Azure JWT, which will be retrieved from Azure AD authorization endpoints, for every protecting route, which then will be checked if it is valid Azure JWT and if the user that provided JWT belongs either to admin or moderator group.

5.2.2 Environment Variables

Generally, it is a bad idea to store APIs secret keys or database credentials in the code, if you are working with the public code and forgot to delete sensitive information from the code, someone with the ill intensions could create a lot of problems, like a huge bill for AWS features that you were not even using. Or if you want to have the ability to change specific values on runtime, without editing the code of the project. Here where environment variables come in handy. (Microsoft Learn, 2022 -z)

Environment variables stores the data on the level of the operation system, meaning no one without access to the operating system will be able to retrieve the value of the environment variable. Environment variables will be used in both front-end and back end for storing sensitive information like MongoDB connection string, Admin and moderator groups ID and Azure "App Registration" sensitive IDs. (Microsoft Learn, 2022 -z)

5.3 Front-end Setup

Before starting to write the code for thesis front-end, necessary installations and configurations need to be made. Front-end will be written on React framework with TypeScript, to initialize react project with typescript, Node.js needs to be installed and using Windows Command Prompt (Command 1) needs to be executed.

Command 1 Create React App with Typescript

```
npx create-react-app app-name --template typescript
```

Front-end will utilize several npm packages, to install npm packages in the root folder of the project Command 2 needs to be executed.

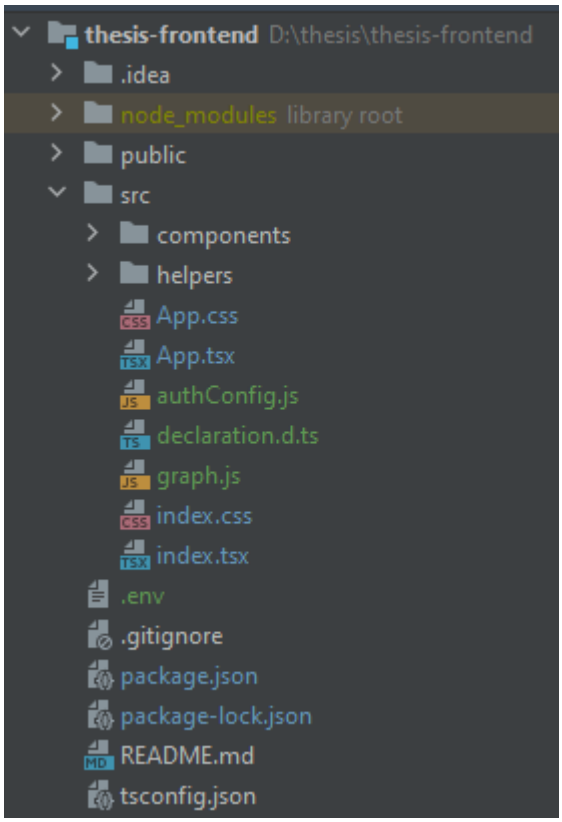
Command 2 Installs npm packages from the npm manager

```
npm install package-name
```

These packages will be utilized "MSAL" for handling Azure authentication, "moment" for handling date types, "react-icons" for a huge library of royalty free icons and "react-router-dom" for handling front-end routes.

After installing all the necessary packages, project file structure will be created. (Figure 2)

Figure 2 Front-end file structure



Project files will be in the src folder which will be consisted of components folder for storing all the components, “helpers” folder for storing all the help files like APIEndpoints.js, helperFunctions.js, main project file is App.tsx where all the routes will be located and the index.tsx which will render front-end to browser DOM.

For storing sensitive information like moderator and admin group IDs and App Registrations IDs, environment variable file will be created for both the front-end and back end. (Program code 1)

Program code 1 Front-end Environment Variable file

```
REACT_APP_API_LINK="http://localhost:8080"
REACT_APP_APP_ID="ef00df72-5e3b-4ef6-8ab5-7e3460622cb1"
REACT_APP_TENANT_ID="e76a6c0f-ad54-4e40-a5c8-8e1ffd91bcb2"
REACT_APP_ADMIN_GROUP_ID="ca7e64a4-22b3-4fba-baa4-0fd8b2603337"
REACT_APP_MODERATOR_GROUP_ID="601a2403-5a90-4337-bbee-715d2ca3a626"
REACT_APP_ADMIN_ID="8f782dd2-2e3c-4551-93fe-a721056c6c81"
```

For handling Azure authentication, azureConfig.js will be created.

Program code 2 Auth Configuration Setup

```
export const msalConfig =
{
    auth: {
        clientId: process.env.REACT_APP_APP_ID,
        authority:
`https://login.microsoftonline.com/${process.env.REACT_APP_TENANT_ID}`, //
This is a URL (e.g. https://login.microsoftonline.com/{your tenant ID})
        redirectUri: "http://localhost:3000/",
    },
    cache: {
        cacheLocation: "sessionStorage", // This configures where your
cache will be stored
        storeAuthStateInCookie: false, // Set this to "true" if you are
having issues on IE11 or Edge
    }
};

// Add scopes here for ID token to be used at Microsoft identity platform
endpoints.
export const loginRequest =
{
    scopes: ["User.Read"]
};
```

authConfig.js will help guide MSAL functions to the proper endpoint, handle App Registration authorization and Security tokens receiving. (Program code 2)

5.4 Front-End

Thesis front-end is a React application, the whole idea behind such applications is to have different components each representing one part of the UI and corresponding functionality, because of that application built for practical demonstration have different components from Navigation Bar component to User Beta Access Request components. Thesis will cover only those components and their part that highlights working with Azure AD and security.

5.4.1 Authorization

There are two implemented methods that manage authorization in thesis front-end: SSO and through Sign in button.

Program code 3 SSO authorization

```
useEffect(() =>
{
	instance.ssoSilent(loginRequest)
		.catch(err =>
		{
			console.log(err);
		});
}, []);
```

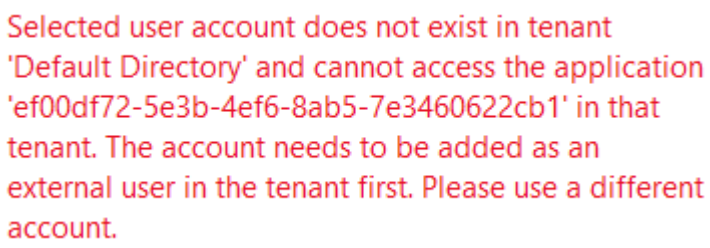
SSO tries to silently login the user to application if the already logged in to Azure. It works by looking for the cache azure login details that stored in the session storage of the user browser. It is possible to configure a fallback if the SSO fails and requires user interaction but for this thesis it will not allow users, that do not belong to Azure AD tenant, to access the application. (Program code 3)

Program code 4 Sign in button component

```
import React from "react";
import { useMsal } from "@azure/msal-react";
import { loginRequest } from "../authConfig";

export const SignInButton = () =>
{
    const { instance } = useMsal();

    const handleLogin = () =>
    {
        instance.loginRedirect(loginRequest).catch(e =>
        {
            console.log(e);
        });
    }

    return (
        <button onClick={() => handleLogin()}>Sign in</button>
    );
}
```

When the user press the Sign In button (Program code 4), page redirects to the Microsoft Azure authentication where user required to enter their Microsoft account credentials, after that Azure checks if the user that is trying to login currently have access to the tenant of the application, if user have such access, the application redirects back to main page and user is successfully logged in, if not then corresponding message appears. (Figure 3)

Figure 3 Azure Authentication error message



After the user is successfully logged in, application making request to Microsoft Graph API to retrieve the security tokens silently, which then stores the idToken and AccessToken into corresponding variables. If silent approach was unsuccessful, then browser window pop-ups where the user required to complete asked steps to acquire the tokens. (Program code 5)

Program code 5 RequestAccessToken function that retrieves Security Tokens from Graph API

```
export const RequestAccessToken = (accounts: AccountInfo[], instance:
IPublicClientApplication, setGraphAccessToken:
Dispatch<SetStateAction<string>>, setJwtToken:
Dispatch<SetStateAction<string>>): void =>
{
    const request =
        {
            ...loginRequest,
            account: accounts[0]
        };

    // Silently acquires an access token which is then attached to a
request for Microsoft Graph data
    instance.acquireTokenSilent(request).then((response) =>
    {
        setJwtToken(response.idToken);
        setGraphAccessToken(response.accessToken);
    }).catch((e) =>
    {
        instance.acquireTokenPopup(request).then((response) =>
        {
            setJwtToken(response.idToken);
            setGraphAccessToken(response.accessToken);
        });
    });
}
```

When tokens acquired, groups IDs that the user belongs to being retrieved from the idTokenClaims which is possible because of Token Configuration. (Program code 6)

Program code 6 Retrieving groups IDs

```
if(accounts[0].idTokenClaims !== undefined &&
Array.isArray(accounts[0].idTokenClaims.groups))
{
    setGroupsID(accounts[0].idTokenClaims.groups)
} else
{
    setGroupsID([accounts[0].localAccountId]);
}
```

5.4.2 Custom checkers

For checking the logged in user group, custom checkers were written. Custom checkers, for example IsAdmin, checks whether the user is belonging to admin groups by checking all user groups id and trying to find a match for the admin group id which stored in Environment variable. (Program code 7)

Program code 7 isAdming custom checker

```
export const isAdmin = (groupsID: string[]): boolean =>
{
    return groupsID.some(id => id ===
process.env.REACT_APP_ADMIN_GROUP_ID);
}
```

5.4.3 Control panel

For viewing and managing all the user, user requests and games, control panel component was created. Only user who are logged in and belongs to either admin or moderator groups can see the link and access the control panel, it is possible thanks to Azure in-build Authenticated/Unauthenticated template components and custom checkers. (Program code 8)

Program code 8 Security of control panel route

```
App.tsx component:

<Route path="/control-panel/:option" element={
    <>
        <AuthenticatedTemplate>
            {(isAdmin(groupsID) || isModerator(groupsID) ||
isMainAdmin(groupsID)) ?
                <ControlPanel />
                : <p>You are not allowed to access this page.</p>}
        </AuthenticatedTemplate>
        <UnauthenticatedTemplate>
            <p>You are not signed in! Please sign in.</p>
        </UnauthenticatedTemplate>
```

```
    </>
} />
```

NavBar component:

```
<div className={styles.navigation_controls}>
    <a href="/">Games</a>
    {(isModerator(groupsID) || isAdmin(groupsID) || isMainAdmin(groupsID))
&& <a href="/control-panel/">Control Panel</a>}
</div>
```

Control panel includes two components GameList and UserList, they are both renders, corresponding to their name, lists and functionalities.

For retrieving data from the back end, async/await functions with fetch utilized, whenever an admin or a moderator requesting access to a protected route, along with necessary information for the route, authorization header passed. Bearer Authentication used to authorize the user and pass Azure JWT token, which then decoding on the back end and based on result, outputs the data that will be utilized in the front-end. (Program code 9)

Program code 9 addGame async function

```
export const addGame = async (jwtToken: string, name: string, description:
string, betaDate: Date) =>
{
    try
    {
        return await fetch(`${process.env.REACT_APP_API_LINK}/game/`,
            {
                method: "POST",
                headers: {'Authorization': `Bearer ${jwtToken}`, 'Content-
Type': 'application/json'},
                body: JSON.stringify({
                    "name": name.trim().replace(/\s\s+/g, ' '),
                    "description": description.trim().replace(/\s\s+/g, '
'),
                    "betaDate": betaDate
                })
            })
            .then(response => response.json());
    }
    catch (err)
    {
        console.log(err);
    }
}
```

5.5 Back-end Setup

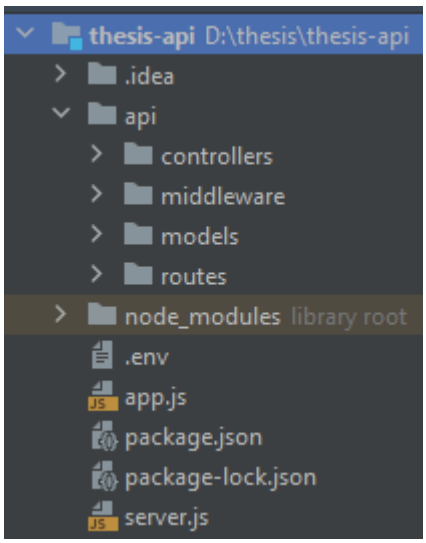
Before we start writing code for our back end, the project needs to be initialized and a few helpful packages should be installed. To initialize our back end, we should create a folder with the appropriate name, navigate to the newly created folder and using the Windows Command Prompt Command 3 needs to be executed.

Command 3 Creates Node.js project

```
npm init
```

Executing Command 2 we will install several npm packages: “azure-ad-jwt-lite” and “jsonwebtoken” which will help us to validate Azure issued JWT tokens, “body-parser” and “cors” which will handle the security of the routes, “dotenv” which will handle the Environment variables, “Express.js” framework on which API will be build, “mongo-sanitize” for security against SQL injections, “mongoose” which will help us to work with powerful noSQL database MongoDB where all the user and games data will be stored, and “nodemon” to ease our work with the development environment. After installing the needed packages, we will need to create a back-end file structure.

Figure 4 Back-end file structure



“api” contains “controllers” folder which contains all the functions for the routes, “middleware” folder which contains JWT token and admin or moderator checker functions, “models” folder

which contains all the schemas for the database tables, and “routes” folder which contains all the routes of the api. The main folder will contain “app.js” file, “server.js” file and “.env” file.
With all the preparations done, we can start writing a secured back end. (Figure 4)

5.6 Back-End

In this section, back-end functionality will be shown and explain along with showcasing security measures that were implemented to secure back-end routes.

Back end of practical part of this thesis is a RESTful API written on Node.js, Express.js and using MongoDB as database for storing information. The main file of the API is called “app.js” which consists of different API settings. At the top of the file is located all the necessary imports (Program code 10)

Program code 10 app.js necessary imports

```
const express = require('express'); //initializing express
const app = express(); //for better usability, express functions stored in
app   variable
const bodyParser = require('body-parser');//initializing body-parser for
routes encoding
const mongoose = require('mongoose');//initializing mongoose for working
with mongoDB

// All the routes of API is stored in a corresponding files for the
usability and readability purposes.
const userRoutes = require("./api/routes/user");
const gameRoutes = require("./api/routes/game");

// initializing dotenv to work with the Environment variables.
require('dotenv').config();
```

Next step is to connect to MongoDB database, encode our routes and allow make API routes visible (Program code 11)

Program code 11 Connection MongoDB, routes encoding and making them reachable

```
mongoose.connect(process.env.MANGO_DB_LINK); //By using mongoose and
dotenv, connection to MongoDB happens.
app.use(bodyParser.urlencoded({extended: false})); //encoding api routes
app.use(bodyParser.json({})); // telling to our api, that all the data that
it will receive or give is in json format
```

```
//making api routes, previously imported, visible
app.use('/user', userRoutes);
app.use('/game', gameRoutes);
```

That concludes app.js file and API settings setup.

5.6.1 Schemas

MongoDB Schemas is a JSON objects that tells database how the record will look like in the document. Schemas supports different types of data from basic like string, date or Boolean to more complex like BSON. (MongoDB, n.d)

Thesis back-end have two schemas that represents a user and a game, both schemas reference each other in some part to help connect them.

Program code 12 Game Schema

```
const mongoose = require('mongoose');

const gameScheme = mongoose.Schema({
    _id: mongoose.Schema.Types.ObjectId,
    name: {type: String, required: true},
    description: {type: String, required: true},
    betaDate: {type: Date, required: true},
    approvedUsers: [{type: mongoose.Schema.Types.ObjectId, ref: "User",
default: []}]
});

module.exports = mongoose.model('Game', gameScheme);
```

The Game Schema Reference “User” schema which tells MongoDB that approvedUsers array stores user IDs, it is needed, since MongoDB has a restriction of record size, but to be able to receive all the approved user information, reference is made. (Program code 12)

Program code 13 User Schema

```
const mongoose = require('mongoose');

const userScheme = mongoose.Schema({
    _id: mongoose.Schema.Types.ObjectId,
    firstName: {type: String, required: true},
    lastName: {type: String, required: true},
    email: {type: String, required: true},
    gameID: {type: mongoose.Schema.Types.ObjectId, ref: "Game", required:
```

```
true},
    approved: {type: Boolean, default: false}
});

module.exports = mongoose.model('User', userScheme);
```

Same logic applies for the User Schema, but this time in references “Game” Schema and tells MongoDB that the “gameID” field stores game ID. (Program code 13)

5.6.2 Routes

Routes in the API is an endpoint which represents a response to a user action. Each route always represents one of these methods GET, POST, PATCH, DELETE, PUT or SEND. The route consists of its method, route path, for example “/user”, optional parameter “Middleware” and the functions that will be executed upon route call. (Radhey Shyam, 2018)

Since, thesis back end have two schemas it also has two routes file, the user and the game, each routes file consists of getting all record, getting one record by id, creating a record, updating a record, and deleting a record, user routes also include the PATCH routes for approving or disapproving beta access. (Program code 14)

Program code 14 Part of the user routes file

```
// Get all users
router.get('/', [checkAuth, isModeratorOrAdminOrMainAdmin],
UserController.get_all_users);

// Add a user
router.post('/', UserController.add_user);


// Update a user by id
router.patch('/:id', [checkAuth, isAdminOrMainAdmin],
UserController.update_user_by_id);

// Approve user beta access by userid
router.patch('/approve/:id', [checkAuth, isModeratorOrAdminOrMainAdmin],
UserController.user_beta_access_status_by_id);
```

5.6.3 Middleware’s

In the previous chapter, optional parameter middleware was mentioned. Usually, middleware’s is a security check functions that needs to be passed successfully before accessing the route. By

examining Program Code 5, after route path a few middleware's are called, checkAuth and isModeratorOrAdmin.

Program code 15 Check Auth middleware

```
const {verifyAzureToken} = require("azure-ad-jwt-lite");

module.exports = async (req, res, next) =>
{
    try
    {
        const token = req.headers.authorization.split(' ')[1];
        const decoded = await verifyAzureToken(token,
            {
                useCache: false
            });
        res.groups = decoded.groups ? decoded.groups : [decoded.oid];

        await next();
    } catch (error)
    {
        return res.status(401).json(
            {
                error: "Auth failed"
            });
    }
};
```

Check auth middleware verifies that receiving with the Bearer Authentication Azure JWT token is authentic and retrieving necessary IDs for next middleware. (Program code 15)

Program code 16 isModeratorOrAdmin middleware

```
module.exports = (req, res, next) =>
{
    try
    {
        const groups = res.groups;
        if(groups.some(id => id === process.env.MODERATOR_GROUP_ID) ||
groups.some(id => id === process.env.ADMIN_GROUP_ID))
        {
            return next();
        }

        throw new Error();
    } catch (error) {
        return res.status(401).json(
            {
                error: "You are not allowed to access this resource."
            });
    }
};
```

isModeratorOrAdmin middleware receives groups ID from the previous middleware check auth which then compares if any of the received IDs corresponds to moderator or admin group. (Program code 16)

Using middleware in API routes we can ensure, that only the right user will get access to the requested resource.

6 Results

All raised researched questions were answered fully with detailed explanations and with providing additional information and explanations were needed. Development project was completed successfully, and it served the role to provide a detailed instructions on how to implement Azure authentication and how to secure front and back -end of Web Applications. Feedback from the interim seminar was positive, the main key words of feedback were: “easy to read and understand such a complex topic”. All the critiques and suggestions were reviewed, and necessary changes were implemented into the final version of the thesis.

Summarizing the answers to the research questions:

- How effective is Azure Active Directory authentication for securing front & back -end of web applications?

Provided security methods of Azure and Azure Active Directory, such as Conditional Access, Multi-factor authentication, Single sign on, Identity protection, Risk-based access policies allows to ensure high security in both front and back -end of web, mobile and other applications.

- What are the key advantages and limitations of using Azure Active Directory authentication compared to other authentication methods?

Key advantages of Azure AD are single credentials for all services, easy implementation to existing or legacy projects, high selection of security methods, from multi-factor authentication to role access policies and collaborators invitation to the projects, while also ensuring strict access policies to the organization resources.

Key Limitations of Azure AD are cost, while Azure provides a big package of free services for small enterprises, when grow happens, the cost of using services is also rise and centralization on Microsoft products, since Azure is Microsoft cloud platform, it is heavily promoting use of Microsoft products, which maybe not suitable for everyone

- What are the best practices for implementing and managing Azure Active Directory authentication to ensure a high level of security?

To maintain the high level of security, use of Conditional access, identity protection with risk-based access policies is mandatory, using MSAL library it is easy to implement Azure AD authentication to the project, but most security features need to be setup in Azure Active Directory services rather than in code, though it is advised to at least writes custom checkers for different situations (e.g., Render only those part of the UI that the user have access to).

Overall, the results are satisfying, researched questions answered, and project development was a success. In the future, the research of Cloud Applications protections is possible and advised.

7 Summary

While reading the thesis, I found that all the research questions were answered thoroughly and necessary information, examples and deep explanation of additional aspects were provided to make given answers as easy understandable as possible.

Working on this thesis greatly improved my knowledge and understanding of Azure and Azure Active Directory in particular, which was of great help during my current internship, since the topic and the thesis research questions comes from the company where I am undergoing my internship, Netum Oy, which heavily relies on Azure services in their everyday work. It was also fun learning different security ways that Microsoft provides, trying to experiment with them and understand how to implement them into thesis development project, though it was not easy as it sounds or looks like but, in the end, results were achieved, research questions were answered and I acquired great knowledge, which I am hoping will help me not only now but in my future careers.

It is wort noting that Microsoft constantly improving Azure platform, and the platform provides not only the security methods but also Cloud solutions, Cosmos Database solution, which also supports MongoDB, Virtual Network solutions and much more. The thesis covered only the tip of the iceberg called Azure and, in the future, it could be broadened to include Cloud Applications security, Docker Security, Mobile Applications security and situation-based Web Application security, which were not cover by the thesis.

Overall, working on thesis was a rewarding experience, it proved to be both interesting, helpful and at the same time fun. I was able to get valuable insight on how Azure works and what it provides, and I will continue exploring this brave new world of Azure.

8 References

Roosevelt Abandy. (2022, August 24). The History of Microsoft Azure. https://techcommunity.microsoft.com/t5/educator-developer-blog/the-history-of-microsoft-azure/ba-p/3574204

Michael Buckbee. (2022, January 19). What is Azure Active Directory? A Complete Overview https://www.varonis.com/blog/azure-active-directory

LogicMonitor. (n.d). What is Azure Active Directory? https://www.logicmonitor.com/blog/what-is-azure-active-directory

Microsoft Learn. (2022, November 16. -a). Introduction to Azure security https://learn.microsoft.com/en-us/azure/security/fundamentals/overview

Microsoft Learn. (2023, January 22. -b). Azure security technical capabilities https://learn.microsoft.com/en-us/azure/security/fundamentals/technical-capabilities

Microsoft Learn. (2022, November 15 -c). Azure Data Encryption at rest https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-atrest

Microsoft Learn. (2022, December 18. -d). Azure Key Vault basic concepts https://learn.microsoft.com/en-us/azure/key-vault/general/basic-concepts

Microsoft Learn. (2023, March 17. -e). End-to-end security in Azure https://learn.microsoft.com/en-us/azure/security/fundamentals/end-to-end

Microsoft Learn. (2023, February 28. -f). What is Conditional Access? https://learn.microsoft.com/en-us/azure/active-directory/conditional-access/overview

Microsoft Learn. (2023, April 4. -g). What is Azure Active Directory Domain Services? https://learn.microsoft.com/en-us/azure/active-directory-domain-services/overview

Microsoft Learn. (2023, March 10. -h). What is Azure AD Privileged Identity Management? https://learn.microsoft.com/en-us/azure/active-directory/privileged-identity-management/pim-configure

Microsoft Learn. (2023, March 15. -i). How it works: Azure AD Multi-Factor Authentication https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-mfa-howitworks

Microsoft Learn. (2023, April 5. -k). What is hybrid identity with Azure Active Directory? https://learn.microsoft.com/en-us/azure/active-directory/hybrid/whatis-hybrid-identity

Microsoft Learn. (2023, April 5. -l) What is password has synchronization with Azure AD? https://learn.microsoft.com/en-us/azure/active-directory/hybrid/connect/whatis-phs

Microsoft Learn. (2023, April 5. -m) User sign-in with Azure Active Directory Pass-through Authentication https://learn.microsoft.com/en-us/azure/active-directory/hybrid/connect/how-to-connect-pta

Microsoft Learn (2023, April 5. -n) What is federation with Azure AD? https://learn.microsoft.com/en-us/azure/active-directory/hybrid/connect/whatis-fed

Microsoft Learn (2023, May 5. -o) What is the Microsoft identity platform? https://learn.microsoft.com/en-us/azure/active-directory/develop/v2-overview

Microsoft Learn. (2023, March 8. -p). What is Identity Protection https://learn.microsoft.com/en-us/azure/active-directory/identity-protection/overview-identity-protection

Microsoft Learn. (2022, November 16. -q). Risk-based access policies https://learn.microsoft.com/en-us/azure/active-directory/identity-protection/concept-identity-protection-policies

Microsoft Learn. (2022, August 25. -r). Best practices for securing PaaS web and mobile applications using Azure App Service https://learn.microsoft.com/en-us/azure/security/fundamentals/paas-applications-using-app-services

Microsoft Learn. (2023, January 31. -s). What is the Microsoft identity platform? https://learn.microsoft.com/en-us/azure/active-directory/develop/v2-overview

Microsoft Learn. (2023, January 1. -t) Authentication vs. authorization https://learn.microsoft.com/en-us/azure/active-directory/develop/authentication-vs-authorization

Microsoft Learn. (2023, January 8. -u) Security tokens https://learn.microsoft.com/en-us/azure/active-directory/develop/security-tokens

Microsoft Learn. (2022, October 12. -v) Overview of the Microsoft Authentication Library (MSAL) https://learn.microsoft.com/en-us/azure/active-directory/develop/msal-overview

Azure AD GitHub. (2021, September 7) microsoft-authentication-library-for-js https://github.com/AzureAD/microsoft-authentication-library-for-js

Microsoft Learn. (2023, April 5. -w) Tutorial: Register a Single-page application with the Microsoft identity platform https://learn.microsoft.com/en-us/azure/active-directory/develop/single-page-app-tutorial-01-register-app

Microsoft Learn. (2023, April 5. -x) Tutorial: Prepare a Single-page application for authentication

https://learn.microsoft.com/en-us/azure/active-directory/develop/single-page-app-tutorial-02-prepare-spa?tabs=visual-studio

Microsoft Learn. (2023, April 5. -y) Tutorial: Create components for sign in and sign out in a React single page app https://learn.microsoft.com/en-us/azure/active-directory/develop/single-page-app-tutorial-03-sign-in-users?tabs=visual-studio

JWT. (n.d.) Introduction to JSON Web Tokens https://jwt.io/introduction

Microsoft Learn. (2022, November 30. -z) about_Environment_Variables https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_environment_variables?view=powershell-7.3

MongoDB. (n.d.) Schemas https://www.mongodb.com/docs/atlas/app-services/schemas/

Radhey Shyam. (2018, August 27). Node JS API Route https://medium.com/@radheyg11/node-js-api-route-89f4497e7131

Katie Lawson. (2022, September 16). What Are Single Page Applications and why Do People Like Them So Much? https://www.bloomreach.com/en/blog/2018/what-is-a-single-page-application?spz=learn_orig

Annex 1: Material management plan

During the development project, a diary is kept, in which technical information about the project is collected. This information is analyzed for the thesis implementation of theory topics into development project. The diary is stored on drive C of the author's computer and is regularly backed up to personal Google Drive. The diary is kept at station C and Google Drive for at least one year after the completion of the thesis.

The completed development project is stored on drive D of the author's computer and is regularly backed up to personal Google Drive. The project is kept at station D and Google Drive for at least one year after the completion of the thesis.