Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Autentizace a autorizace v moderních systémech s vysokou bezpečností

Bc. Lukáš Vydržel

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Lukáš Vydržel

Informatika

Název práce

Autentizace a autorizace v moderních systémech s vysokou bezpečností

Název anglicky

Authentication and authorization in modern high security systems

---

Cíle práce

Cílem práce je vytvoření návrhu pro architekty, analytiky a vývojáře systémů s požadovanou vysokou úrovní zabezpečení. Tento návrh bude ověřen testovací aplikací a jejím zhodnocením.
Dílčím cílem je analýza použitelných technologií za účelem implementace uživatelské autentizace a autorizace v systémech, kde je vyžadována vysoká úroveň zabezpečení (například bankovní aplikace).

Metodika

Na základě nastudování předepsané literatury budou vybrány technologie vhodné pro zabezpečení aplikace. Při výběru musí být rozhodováno na základě funkčních požadavků a na základě bezpečnostních požadavků (s přihlédnutím na existující bezpečnostní hrozby a předepsané právní normy).

Následně bude implementována demo aplikace na které bude konkrétní technologický stack a konfigurace předvedena. Jednotlivé části aplikace budou v práci detailně popsány.

Doporučený rozsah práce

60-80s.

Klíčová slova

Autorizace, Autentizace, Bezpečnost, OpenID Connect, SAML, OAuth

---

Doporučené zdroje informací

ANDERSON, Ross. 2020. Security engineering: a guide to building dependable distributed systems. 3. vyd.. Indianapolis: Wiley. 1186 s. ISBN 978-1119642787.

GARBIS, J. 2021. Zero Trust Security: An Enterprise Guide. 1. vyd. Apres. 324 s. ISBN 978-1484267011.

GILMAN, E. 2017. Zero Trust Networks: Building Secure Systems in Untrusted Networks. 1. vyd. O'Reilly Media. 240 s. ISBN 978-1491962190.

SMEJKAL, Vladimír, Tomáš SOKOL a Jindřich KODL. 2019. Bezpečnost informačních systémů podle zákona o kybernetické bezpečnosti. 1. vyd. Plzeň: Vydavatelství a nakladatelství Aleš Čeněk. ISBN 978-80-7380-765-8.

WILSON, Y., HINGNIKAR, A. 2019. Solving Identity Management in Modern Applications: Demystifying OAuth 2.0, OpenID Connect, and SAML 2.0. 1. vyd. Appress. 337 s. ISBN 978-1484250945.

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

Ing. Martin Havránek, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 14. 7. 2022

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 28. 11. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 12. 11. 2023

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Autentizace a autorizace v moderních systémech s vysokou bezpečností" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 29. 11. 2023 _____

Poděkování

Rád(a) bych touto cestou poděkoval Ing. Martinu Havránkovi Ph.D. za odborné konzultace, poskytované cenné rady, kterými přispěl k vytvoření této diplomové práce.

Děkuji také mé manželce Ing. Tereze Vydrželové za všeobecnou podporu a pomoc při testování demo aplikace vytvořené v rámci této diplomové práce.

Autentizace a autorizace v moderních systémech s vysokou bezpečností

Abstrakt

Diplomová práce je zaměřena na oblast autentizace a autorizace ve velkých podnikových systémech, kde je kladen důraz na bezpečnost. Jedním z úkolů této práce bude zmapovat používané bezpečnostní metody, bezpečnostní technologie a bezpečnostní protokoly. Avšak hlavním smyslem a účelem této práce bude příprava teoreticky podloženého návrhu autentizačního a autorizačního mechanismu vhodného pro dnešní typy aplikací. Součástí návrhu musí být také soupis doporučených bezpečnostních technologií a protokolů vhodných pro vybranou konkrétní architekturu systému. Architektura systému by měla vycházet z aktuálních trendů, a tudíž bude třeba provést průzkum v této oblasti. Oblast bezpečnosti by měla být také podložena legislativními požadavky a doporučeními v případě systémů, kde je to vyžadováno. Návrh by měl sloužit jako vzor pro architekty, analytiky a vývojáře pracující na přípravě a implementaci systémů, kde je vyžadována vysoká úroveň zabezpečení. Díky tomu mohou ušetřit velké množství času.

Funkce navrženého řešení se musí demonstrovat a ověřit na vlastní implementaci v rámci vhodně zvoleného programovacího jazyka. Bezpečnost navrženého a implementovaného řešení se musí ověřit pomocí dostupných nástrojů nebo metod penetračního testování aplikací.

Klíčová slova: Identifikace uživatele, Autorizace uživatele, Autentizace uživatele, Kybernetická bezpečnost, Bezpečnostní hrozby, Bezpečnostní standardy, Návrh a vývoj aplikací

Authentication and authorization in modern high security systems

Abstract

This Diploma Thesis focuses on the area of authentication and authorization in large enterprise systems, with an emphasis on security. One of the tasks of this Diploma Thesis will be to map the security methods, security technologies, and security protocols currently in use. However, the main target is to prepare of a theoretically substantiated design of an authentication and authorization mechanism suitable for current types of applications. The result design must also include a list of recommended security technologies and protocols suitable for the selected specific system architecture. This system architecture should be based on current trends, requiring research in this area. The security domain should also be supported by legislative requirements and recommendations for systems where it is required. The proposal should serve as a template for architects, analysts, and developers working on the preparation and implementation of systems where a high level of security is demanded, thus saving a considerable amount of time. The functionality of the proposed solution must be demonstrated and verified through its own implementation within a carefully chosen programming language. The security of the proposed and implemented solution must be verified using available tools or methods of application penetration testing.

Keywords: User identification, User authorization, User authentication, Cyber security, Security threats, Security standards, Application design and development

Obsah

1 Úvod

Diplomová práce „Autentizace a autorizace v moderních systémech s vysokou bezpečností“ se zabývá problematikou bezpečnosti dnešních velkých podnikových systémů. Primárně je zaměřená na oblast autentizace a autorizace uživatelů. Vhodnými systémy jsou systémy, které jsou provozovány v rámci podniků, které vyžadují nebo je jim nařizována vysoká úroveň zabezpečení. Může se například jednat o státní instituce, zdravotnické zařízení nebo bankovní ústavy.

Problematika v této oblasti je velice rozsáhlá a pokud chce podnik implementovat bezpečnostní mechanismy do svého systému, tak to znamená, že je třeba nastudovat velké množství zdrojů. Jedná se o zdroje technické (technologie, standardy, architektura, dostupné knihovny a frameworky), zdroje legislativní (zákony, směrnice a doporučení) a zdroje komunitní z oblasti bezpečnosti (současné bezpečnostní hrozby a jejich obrana). Tato práce by měla pomoci podnikům a jejich pracovníkům v uvedené oblasti snížením nákladů díky omezení času procházením všech výše uvedených zdrojů a rovnou jim nabídne návrh a vzorovou implementaci ověřenou funkčně i bezpečně.

Autor si uvedené téma zvolil záměrně, protože pracuje jako vývojář aplikací v jedné z českých bank a s tématem bezpečnosti aplikací se setkává velice často. Z důvodu bezpečnosti a mlčenlivosti nejsou v této práci popsány přesné bezpečnostní metody a implementace se kterými se autor v práci setkává. Nicméně jeho praktické zkušenosti přispěly k dosažení požadovaného cíle práce.

Struktura práce je rozdělená na teoretickou část a praktickou část. Teoretická část na začátku obsahuje úvod do bezpečnosti z pohledu identifikace, autentizace a autorizace uživatelů v aplikacích z běžné dostupných zdrojů internetu a literatury. Následuje základní popis využívaných technologií a bezpečnostních protokolů používaných k výše uvedené problematice. Převážně se jedná o RFC standardy a jejich implementace. Z velké části je práce zaměřená na REST API a s ním používaný JSON formát dat spolu s JSON Web Token (JWT) k bezpečnému předávání dat skrze internet. Dále se v teoretické části autor věnuje oblasti architektury dnešních podnikových systémů, kde jsou zmíněny dvě základní architektury. Jedná se o architekturu monolitickou a mikroservisní. Uvažovaná architektura podnikového systému je důležitým vstupem pro dosažení výsledného návrhu. Následuje část, která se věnuje legislativní oblasti. Bylo třeba ověřit, zda jsou některé subjekty v ČR regulovány z pohledu bezpečnosti a případně o jaké subjekty se jedná. Zákony v ČR na tuto

problematiku myslí a obsahují směrnice, které definují subjekty poskytující takzvané základní služby, které jsou součástí kritické infrastruktury a je pro ně definována sada povinností a doporučení. Zdrojem obsahu těchto směrnic jsou směrnice definované již v rámci EU s tím, že jsou většinou přizpůsobeny potřebám a legislativě konkrétního státu. V ČR existuje také Národní úřad pro kybernetickou bezpečnost (NÚKB), který dohlíží na dodržování bezpečnostních pravidel vybraných subjektů a prověřuje vzniklé bezpečnostní incidenty. Zároveň vydává přehlednou příručku bezpečnostních doporučení a nastavení pro správce systémů, která jim má napomoci ke zvýšení bezpečnosti jejich systémů. Na konci teoretické části je zmíněna nezisková organizace Open Web Application Security Project (OWASP) založená za účelem zlepšení bezpečnosti aplikací a webových služeb. V rámci penetračního testování aplikací se využívá jejich udržovaný seznam nejčastějších bezpečnostních hrozeb v oblasti webových aplikací (OWASP Top 10).

V praktické části se autor z počátku věnuje vhodnému výběru technologií pro navržení referenčního řešení. Rozhodujícím faktorem byly informace získané z vybraných statistik a průzkumů dostupných na internetu. Následuje sestavení referenční architektury systému a k ní vhodného bezpečnostního mechanismu pro identifikaci, autentizaci a autorizaci uživatele. Bezpečnostní mechanismus vychází z informací, nařízení a doporučení obsažených v teoretické části této práce. Výsledný návrh byl následně přenesen do konkrétní implementace, která byla vytvořena za účelem potvrzení funkčnosti a bezpečnosti tohoto návrhu. Implementace není plnohodnotná aplikace, pouze obsahuje významné části podnikového systému, které implementují bezpečnostní mechanismy a zároveň poskytují data využívající tyto bezpečnostní mechanismy ke svému zabezpečení. Funkční část demo aplikace byla ověřena manuálním testováním předem vytvořených testovacích scénářů. Bezpečnost návrhu a demo aplikace byla ověřena na základě zhodnocení ošetření jednotlivých bezpečnostních hrozeb OWASP Top 10.

Obsah teoretické části této práce a následně vytvořený návrh včetně vzorové implementace může sloužit jako zdroj informací a inspirací pro zabezpečení aplikací. Je určen pro odborníky věnující se návrhu a vývoji aplikací v podnicích, které vyžadují vysokou úroveň zabezpečení jejich systémů.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je vytvoření návrhu pro architekty, analytiky a vývojáře systémů s požadovanou vysokou úrovní zabezpečení. Tento návrh bude ověřen testovací aplikací a jejím zhodnocením.

Dílčím cílem je analýza použitelných technologií za účelem implementace uživatelské autentizace a autorizace v systémech, kde je vyžadována vysoká úroveň zabezpečení (například bankovní aplikace).

2.2 Metodika

Na základě nastudování předepsané literatury budou vybrány technologie vhodné pro zabezpečení aplikace. Při výběru musí být rozhodováno na základě funkčních požadavků a na základě bezpečnostních požadavků (s přihlédnutím na existující bezpečnostní hrozby a předepsané právní normy).

Následně bude implementována demo aplikace, na které bude konkrétní technologický stack a konfigurace předvedena. Jednotlivé části aplikace budou v práci detailně popsány.

3 Teoretická východiska

3.1 Autentizace, autorizace a identifikace

Pro správné navržení a sestavení přihlašovacích mechanismů v aplikacích dnešních i minulých, je třeba pochopit správně význam pojmů identifikace, autentizace a autorizace, které označují 3 základní procesy v této problematice. Tyto procesy jsou dominantní částí bezpečnostní oblasti většiny aplikací.

Za pojmem identifikace se skrývá proces, kdy systém určuje identitu daného subjektu či osoby. Řekněme, že známe emailovou adresu uživatele (například na základě přihlašovacího jména) a my na základě našich registrů nebo registrů třetích stran si k ní přiřadíme konkrétní osobu. Tím jsme docílili toho, že dokážeme říct, že do aplikace se nám přihlásil uživatel s konkrétním jménem a příjmením, aniž by nám ho uživatel v rámci přihlášení musel sdělit. (1)

Pojmy autentizace a autorizace spolu úzce souvisí, ale každý má zcela jiný význam a občas jsou chybně zaměňovány. Autentizace neboli ověření, že uživatel, který se do aplikace přihlašuje je opravdu osoba, za kterou se vydává prostřednictvím přihlašovacího identifikátoru. V procesu identifikace se již předpokládá, že získaný identifikační údaj (například emailová adresa) je skutečně údaj spojený s uživatelem, který se do naší aplikace přihlašuje a lze jej napárovat na správnou identitu. A k tomu existuje velké množství autentizačních mechanismů a ty nejznámější včetně jejich detailů jsou uvedeny v jednotlivých podkapitolách autentizace. (1) (2)

Autorizace je proces, který následuje po procesu autentifikace, kdy systém důvěřuje identitě uživatele, protože bez identity není koho autorizovat. Většina aplikací obsahuje velké množství funkcí a velké množství dat. Ve většině případech není vhodné vše zpřístupnit všem, a proto existuje proces autorizace, kdy systém na základě definovaných mechanismů dokáže určit, jestli daný uživatel má nebo nemá přístup k dané funkcionalitě nebo datům. Ty nejznámější mechanismy jsou uvedeny v jednotlivých podkapitolách autorizace. (1) (3)

3.1.1 Způsoby autentizace

Existuje velké množství autentizačních metod, které se odlišují složitostí a bezpečností. V době výkonných počítačů, které mohou být využity v kybernetických útocích musí být dostatečná úroveň zabezpečení. Toho můžeme dosáhnout větší komplexitou autentizačních metod. Tomu nahrává pokročilá elektronika dnešní doby (chytré mobilní telefony, chytré hodinky, kamery s okamžitým zpracováním obrazu atd.), která otvírá nové možnosti v oblasti autentizačních metod.

Nejjednodušší způsob autentizace je tzv. jednofaktorové ověření (označované jako **SFA – Single Factor Authetication**). Bývá implementované pomocí přihlašovacího jména a hesla. Přístup je tedy podmíněn znalostí hesla a někdy i jména. Aby byl tento způsob dostatečně bezpečný musí být kladen velký důraz na samotné heslo a nakládání s ním. Obecně doporučovaný seznam pravidel týkající se hesla podle OWASP (5):

- Heslo musí být minimálně 12 znaků dlouhé, neboť kratší jsou považována za slabá.
- Heslo může být maximálně 64 znaků dlouhé z důvodu překročení délky vstupu hashovacích algoritmů a možné útoku **Long password denial of service**.
- Znaková sada hesla by neměla být omezována.
- Dlouhá hesla by neměla být na pozadí zkracována.
- Uživateli by měla aplikace ukazovat sílu hesla a zamítat již prolomená hesla.
- Měla by být zajištěna rotace hesla a mít tedy pro něj časové omezení platnosti.
- Uživatel musí mít možnost si heslo změnit a při změně musí zadat aktuální heslo.
- Uživateli by nemělo být povoleno používat jako nové heslo některé z historických hesel.
- Pole pro zadání hesla by mělo mít maskované znaky při zadávání, aby nemohlo být čteno druhou osobou.
- Pole pro zadání hesla by nemělo podporovat schránku pro kopírování a vkládání.
- Obsah hesla by neměl být spojován s osobou (jméno, datum narození atd.)
- Hesla neukládáme v čisté podobě, ale pouze hash, aby v případě úniku dat nedošlo k jejich odhalení.
- Každé heslo by mělo být obohaceno náhodných řetězcem před hashováním k zamezení detekce stejných a známých hesel.

Většina výše uvedených pravidel může způsobit jiné bezpečnostní riziko, které se pojí se složitým heslem. Uživatelé si složitá hesla hůře pamatují, a to může vést k tomu, že si je někde zapíšou v čitelné podobě a zde může dojít k jejich úniku. Z tohoto důvodu by se nemělo zacházet do extrémů a hledat kompromis mezi bezpečným a dobře zapamatovatelným heslem. Totéž může způsobit i kladený důraz na uživatele, aby používali různá hesla v různých aplikacích. Uživatelé si mohou usnadnit práci s více hesly díky bezpečným uložištím hesel (fyzických či aplikačních přímo na zařízení a dnes i online), kde jsou všechna hesla chráněna a šifrována jedním hlavním heslem, které si jediné musí uživatel pamatovat. V rámci jedné společnosti, kde jsou uživatelé nuceni pracovat ve více aplikacích se využívá funkce jednotného přihlašování (označované jako **SSO – Single Sign On**). Díky tomu uživatel se přihlásí jednou a do všech dalších aplikací je přihlášen automaticky. K zajištění mnohem lepšího zabezpečení a zároveň nesnížení komfortu se využívá více faktorových ověření (označované jako **MFA – Multiple Factor Authentication**). Většinou se využívá dvou faktorové ověření, krajně i tří faktorové. V následujících kapitolách jsou uvedeny další způsoby ověření uživatele, které jsou pro více faktorové ověření využívány. (4) (5)

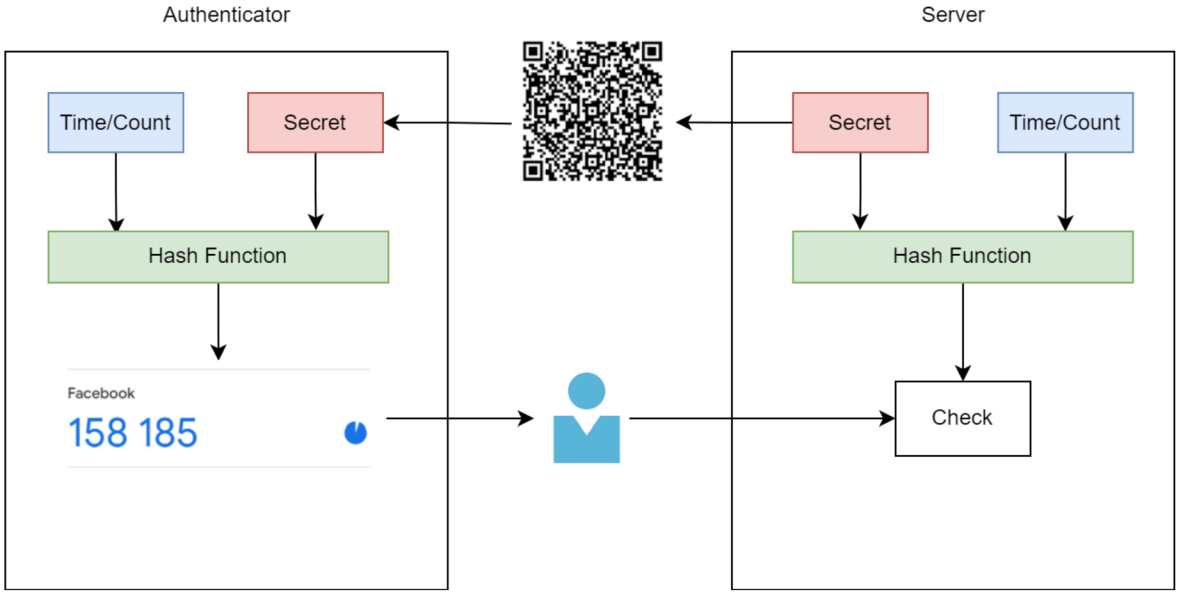
3.1.1.1 Ověření pomocí OTP – One Time Password

Jedná se o metodu, která je používána k posílení bezpečnosti autentizace uživatelů. Tato metoda zahrnuje generování jednorázového hesla, které platí pouze po omezenou dobu nebo pro jedno konkrétní přihlášení. Existuje několik způsobů, jak generovat heslo. To může zahrnovat generování na základě časových hodnot (například **TOTP – Time Based One Time Password**) nebo na základě náhodných událostí (například **HOTP – HMAC based One Time Password**). Oba způsoby jsou znázorněny na obrázku 1.

Princip spočívá v tom, že existuje hardwarové zařízení (mobilní telefon, klíčenka a další), které obsahuje softwarový generátor s uloženým tajným klíčem (shared secret), které si vzájemně vyměnily a uložily tento generátor s autentizačním serverem. Ke generování se používá hashovací funkce jejímž výsledkem je jednoduše čitelný řetězec. Pro snadné použití je doporučeno používat číslo o 6 číslicích. Dle metody do hashovací vstupuje další prvek, který se dynamicky mění. Buď se jedná o časovou značku nebo hodnotu počítadla použitých vygenerovaných hesel. Oboje zařídí, že se nám v čase hesla mění a není třeba požívat online spojení mezi generátorem a autentizačním serverem. Využití časové značky má nevýhodu, že je třeba použitá hesla si ukládat, aby nebylo možné opakovaného použití v rámci daného

časového okna, které funguje jako rezerva pro případnou časovou odchylku generátoru od autentizačního serveru. Podobně i algoritmus s počítadlem musí vyzkoušet několik hodnot dopředně pro případnou odchylku způsobenou nevyužitím vygenerovaných hesel. (6) (7) (8)

Obrázek 1 – Proces fungování HOTP/TOTP



Zdroj: https://dz2cdn1.dzone.com/storage/temp/16700045-hotp.png (2023)

3.1.1.2 Biometrické ověření

Biometrické ověření uživatelů je založeno na unikátní tělesné vlastnosti člověka, kterou získal během svého narození nebo během života a její charakteristiku lze elektronicky změřit. Existuje několik typů biometrických ověření uživatelů, které jsou založené na unikátních vlastnostech jedince. Jedinečné vlastnosti člověka byly předmětem výzkumů a statistik. Dnes jsou běžně využívány v kapesních elektronických zařízeních (chytré telefony, chytré hodinky, laptopy atd.) a bezpečnostních zařízeních chránící majetek osob.
Zde je seznam nejběžnějších metod biometrického ověření:

- **Rozpoznávání obličeje** – tento typ biometrického ověření využívá digitální obraz obličeje uživatele a porovnává ho s předem uloženým obrazem, aby potvrdil totožnost uživatele
- **Rozpoznávání otisku prstu** – tento typ biometrického ověření porovná otisk prstu uživatele s předem uloženým otiskem prstu, aby potvrdil totožnost uživatele

- **Rozpoznávání hlasu** – tento typ biometrického ověření využívá nahrávky hlasu uživatele a porovnává ji s předem uloženou nahrávkou, aby potvrdil totožnost uživatele.
- **Rozpoznávání duhovky** – tento typ biometrického ověření získá digitální obraz duhovky uživatele a porovná ho s předem uloženým obrazem, aby potvrdil totožnost uživatele
- **Rozpoznávání žilního obrazu** – tento typ biometrického ověření získá digitální obraz žilní struktury uživatele a porovná ho s předem uloženým obrazem, aby potvrdil totožnost uživatele (tato metoda není příliš využívána)
- **Rozpoznávání chůze** – tento typ biometrického ověření získává data o chůzi uživatele a porovnává je s předem uloženými daty, aby potvrdil totožnost uživatele (tato metoda je určena spíše k detektivním činnostem, identifikaci osob na videích) (9)

3.1.1.3 Mobilní telefon jako autentizační metoda

Jeden ze způsobů implementace vícefaktorové autentizace je použití mobilního telefonu, u kterého máme ze strany uživatele potvrzené vlastnictví. Například při zavedení jednorázových hesel (OTP) můžeme uživateli posílat hesla prostřednictvím textových zpráv (**SMS – Short Message Service**). Uživatel si také může na chytrý mobilní telefon nainstalovanou některou z běžně dostupných aplikací sloužící jako generátory jednorázových hesel. Tento způsob ověření uživatele využívají i bankovní instituce v ČR v rámci podpisu smluv či autorizaci platebních příkazů. (10)

Zákon ČR to definuje jako silné ověření uživatele v rámci **§ 223 zák. č. 370/2017 Sb.** a jeho znění s vymezením pravidel je následující:

„§ 223 *Silné ověření uživatele*

(1) Osoba oprávněná poskytovat platební služby použije silné ověření uživatele, jestliže plátce

a) přistupuje ke svému platebnímu účtu prostřednictvím internetu,

b) dává platební příkaz k elektronické platební transakci,

c) provádí jiný úkon, který je spojen s rizikem podvodného jednání v oblasti platebního styku, zneužitím platebního prostředku nebo informací o platebním účtu, nebo

d) požaduje informace o platebním účtu prostřednictvím poskytovatele služby informování o platebním účtu.

(2) Dává-li uživatel platební příkaz prostřednictvím internetu nebo prostřednictvím elektronického zařízení, které lze použít k dálkové komunikaci, nebo dává-li platební příkaz nepřímo, osoba oprávněná poskytovat platební služby použije silné ověření uživatele, které zahrnuje jednorázové prvky propojující platební transakci s přesnou částkou a určitým příjemcem.

(3) Silným ověřením uživatele se pro účely tohoto zákona rozumí ověření, které je založeno na použití alespoň 2 z těchto prvků:

a) údaje, který je znám pouze uživateli,

b) věci, kterou má uživatel ve své moci,

c) biometrických údajů uživatele.

(4) Prvky podle odstavce 3 musí být vzájemně nezávislé a prolomení jednoho prvku nesmí ovlivnit spolehlivost prvků ostatních. Postup ověření musí zabránit zneužití prvků, které jsou k ověření používány.

(5) Způsob silného ověření uživatele podle odstavců 1 a 2 stanoví přímo použitelný předpis Evropské unie, kterým se provádí čl. 98 směrnice Evropského parlamentu a Rady (EU) 2015/236613).

(6) Ustanovení odstavců 1 a 2 se nepoužijí v případech, které stanoví přímo použitelný předpis Evropské unie, kterým se provádí čl. 98 směrnice Evropského parlamentu a Rady (EU) 2015/236613).“ (12)

3.1.2 Autentizační technologie a protokoly

3.1.2.1 JWT – JSON Web Token

JSON Web Token (dále jen JWT) je standard definovaný v RFC7519 používaný ke standardizovanému předávání informací mezi oddělenými aplikacemi (klient / server), nikoliv však k velkým datovým objemům. Samotné JWT je pouze standard pro formát a strukturu dat v JSON formátu a nemá definované standardy zaměřené na bezpečnost. Bezpečnostní standardy jsou definované samostatně, a to JSON Web Encryption pro účely šifrování obsahu a JSON Web Signature pro účely ověření autenticity a konzistence obsahu, že data nebyla cestou změněna. Oba standardy jsou zpracovány v dalších kapitolách. (13)

JWT na obrázku 2 se skládá ze tří textových částí oddělených tečkou (dvě povinné a třetí nepovinná), kde každá část je ve skutečnosti samostatný strukturovaný JSON objekt zakódovaný pomocí Base64 algoritmu.

```
HEADER(hlavička).PAYLOAD(tělo).SIGNATURE(podpis)
```

Hlavička obsahuje metadata o tokenu, jako jsou například typ tokenu a použitý šifrovací algoritmus. Může obsahovat i další klíče podle potřeby, které jsou uvedeny také v JSON formátu. (14)

Typicky se skládá z následujících claimů (klíč+hodnota):

- **alg** (algorithm) - určuje použitý šifrovací algoritmus pro podepsání tokenu
- **typ** (type) - udává typ tokenu, například jwt

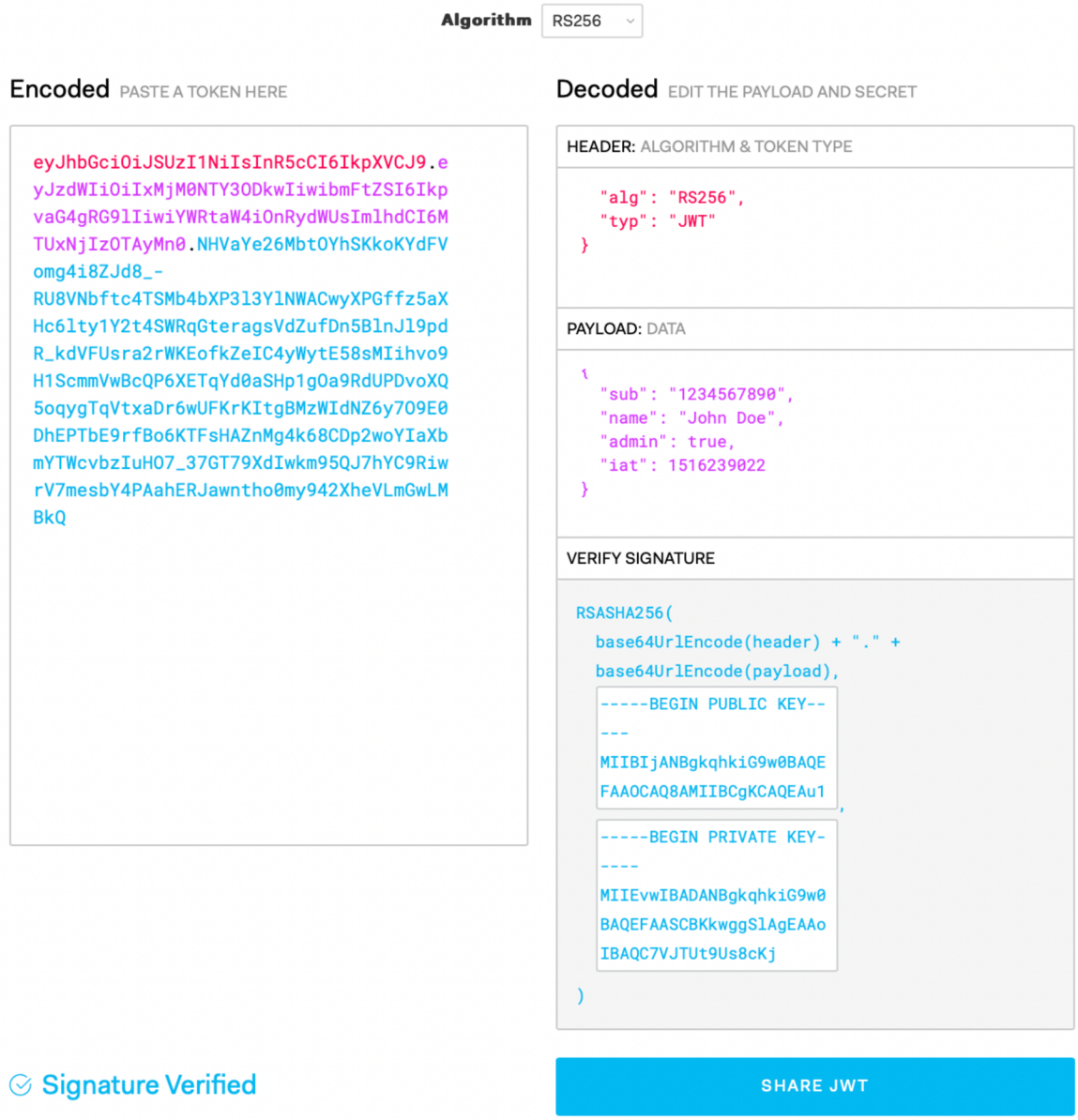
Tělo (data) obsahuje užitečné informace o uživateli nebo o tokenu. Podobně jako hlavička, tělo může obsahovat další klíče podle potřeb dané aplikace, které jsou uvedeny také v JSON formátu. (14)

Typicky se skládá z následujících claimů (klíč+hodnota):

- **iss** (issuer) - identifikátor vydavatele tokenu
- **sub** (subject) - identifikátor subjektu, například login uživatele
- **exp** (expiration time) - datum a čas, kdy token expiruje
- **nbf** (not before) - datum a čas, kdy token se stává platným
- **iat** (issued at) - datum a čas, kdy byl token vydán
- **jti** (jwt identifier) - unikátní identifikátor tokenu

Podpis zajišťuje integritu a autenticitu tokenu. Na základě zvoleného algoritmu je použito symetrické nebo asymetrické šifrování pro vytvoření podpisové části. Do hashovací funkce vstupuje hlavička (Base64 kódovaná) a tělo (Base64 kódované) spojené tečkou a výstup je použit jako podpis. V případě symetrického šifrování (algoritmus HS256) musí být vytvořeno tajemství (secret), kterým je token podepsán a následně pro ověření podpisu musí být tajemství bezpečným způsobem sdíleno s konzumentem tokenu. Právě bezpečné sdílení a doporučená rotace tohoto tajemství je nevýhodou tohoto způsobu podepisování, a proto je doporučeno využít asymetrického šifrování (algoritmus RS256). Vydavatel tokenu vlastní dvojici klíčů, kde jeden je privátní a ten použije k podpisu tokenu a tento klíč nesmí a ani nemusí s nikým sdílet. Druhý klíč je veřejný a ten sdílí se všemi svými konzumenty tokenu, kteří si díky němu mohou podpis tokenu ověřit. Detaily a standardy procesu podepisování jsou definovány v rámci JSON Web Signature, který je popsán v další kapitole. (13) (14)

Obrázek 2 – Ukázka dekódovaného JWT



Zdroj: https://jwt.io (2023)

3.1.2.2 JWE – JSON Web Encryption

JSON Web Encryption (dále jen JWE) je standard definovaný v RFC7516 pro zabezpečení přenosu dat v JSON formátu pomocí šifrování obsahu a zároveň poskytuje způsob, jak tento řetězec dešifrovat a získat původní data. JWE používá algoritmy šifrování, jako je například AES nebo RSA, k zabezpečení dat. Při použití JWE jsou data nejprve zašifrována pomocí náhodného klíče a pak je tento klíč zašifrován pomocí klíče určeného pro přenos dat. Výsledkem je JSON řetězec obsahující šifrovaná data, zašifrovaný náhodný klíč a další metadata potřebné pro dešifrování dat. V praxi se často JWE a JWT používají

společně jako zabezpečený a standardizovaný způsob, jak přenášet sensitivní autentizační a autorizační informace mezi aplikacemi (klient / server), které využívají komunikaci například pomocí REST API rozhraní přes nezabezpečenou síť nebo chtějí data utajit přes systémy střední vrstvy. (13)

JWE se skládá z pěti textových částí oddělených tečkou, kde každá část je zakódovaná pomocí Base64 algoritmu.

HEADER(hlavička)**.ENCRYPTED_KEY**(zašifrovaný_klíč)**.IV**(inicializační_vektor)**.CIPHERTEXT**(zašifrovaný_obsah)**.AUTHETICATION_TAG**(autentizační_tag)

Hlavička obsahuje metadata o tokenu, jako jsou například algoritmus pro šifrování klíče či algoritmus pro vygenerování šifrovacího klíče a jeho vlastností.

Typicky se skládá z následujících claimů (klíč+hodnota):

- **alg** (algorithm) - určuje algoritmus pro šifrování klíče
- **enc** (encryption algorithm) – určuje algoritmus k zašifrování obsahu
- **zip** (compression algorithm) - určuje algoritmus pro kompresi obsahu
- **jku** (jwk set url) – URL adresa s veřejným klíčem ve formátu JWK, který byl použit k zašifrování v případě asymetrické šifry
- **jwk** (json web token) – obsahuje JWK, který byl použit k zašifrování
- **kid** (key id) – identifikátor klíče použitého k šifrování
- **x5u** (x.509 url) – URL adresa s veřejným klíčem ve formátu PEM, který byl použit k zašifrování v případě asymetrické šifry
- **x5c** (x.509 certificate chain) – cesta k vydavateli certifikátu veřejného klíče, který byl použit k zašifrování v případě asymetrické šifry
- **x5t** (x.509 certificate sha-1 thumbprint) – otisk certifikátu veřejného klíče, který byl použit k zašifrování v případě asymetrické šifry
- **typ** (type) – udává typ tokenu, například jwt
- **cty** (content type) – určuje formát šifrovaného obsahu
- **crit** (critical) – udává rozšíření, které je nutné brát v potaz pro ověření platnosti, například exp, který klade důraz na expiraci (15)

3.1.2.3 JWS – JSON Web Signature

JSON Web Signature (dále jen JWS) je standard definovaný v RFC7515 pro elektronické podepisování dat v JSON formátu. Používá se ve spojení s JSON Web Token, aby bylo možné ověřit autenticitu a konzistenci dat, která je součástí daného tokenu. (13)

Přidává další claims (klíč+hodnota), které obsahují informace o elektronickém podpisu.

- **alg** (algorithm) - určuje algoritmus pro šifrování klíče
- **enc** (encryption algorithm) – určuje algoritmus k zašifrování obsahu
- **zip** (compression algorithm) - určuje algoritmus pro kompresi obsahu
- **jku** (jwk set url) – URL adresa s veřejným klíčem ve formátu JWK, který byl použit k zašifrování v případě asymetrické šifry
- **jwk** (json web token) – obsahuje JWK, který byl použit k zašifrování
- **kid** (key id) – identifikátor klíče použitého k šifrování
- **x5u** (x.509 url) – URL adresa s veřejným klíčem ve formátu PEM, který byl použit k zašifrování v případě asymetrické šifry
- **x5c** (x.509 certificate chain) – cesta k vydavateli certifikátu veřejného klíče, který byl použit k zašifrování v případě asymetrické šifry
- **x5t** (x.509 certificate sha-1 thumbprint) – otisk certifikátu veřejného klíče, který byl použit k zašifrování v případě asymetrické šifry
- **typ** (type) – udává typ tokenu, například jwt
- **cty** (content type) – určuje formát šifrovaného obsahu
- **crit** (critical) – udává rozšíření, které je nutné brát v potaz pro ověření platnosti, například exp, který klade důraz na expiraci (16)

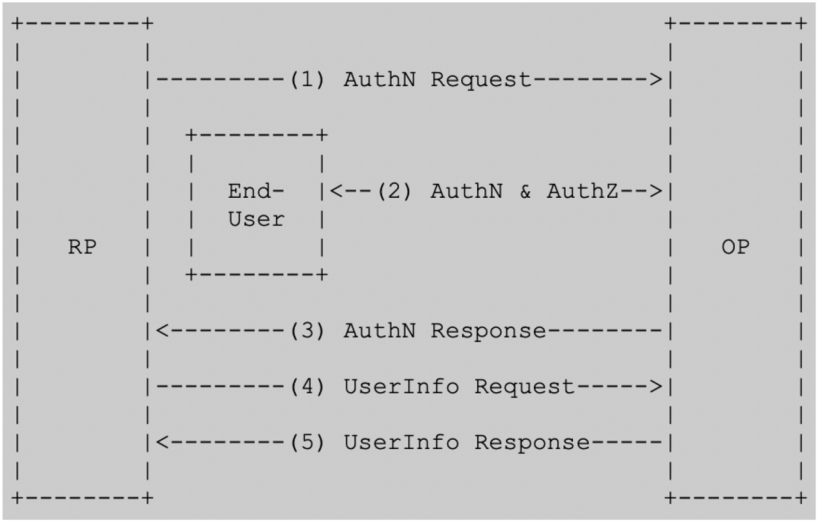
3.1.2.4 JWK – JSON Web Key

JSON Web Key (dále jen JWK) je standard definovaný v RFC7517, který definuje formát pro reprezentaci klíčů používaných v protokolech JSON Web Signature a JSON Web Encryption. JWK poskytuje standardizovaný způsob pro reprezentaci klíčů v JSON formátu, což umožňuje snadné přenosy a výměnu klíčů mezi oddělenými systémy. (13) (17)

3.1.2.5 OpenID Connect (OIDC)

OpenID Connect je otevřený standard pro autentizaci a autorizaci, který je postaven na frameworku OAuth 2.0 (autorizační protokol zmíněný v další kapitole). Je standardizován a řízen organizací OpenID Foundation. Umožňuje federovanou autentifikaci uživatelů třetím stranám / aplikacím. Informace o uživateli jsou bezpečně předávány prostřednictvím JSON Web Tokenu, který může být digitálně podepsán ověřovacím serverem pomocí asymetrické šifry, v případech, kdy máme dvě nezávislé aplikace a sdílení veřejného klíče certifikátu ověřovacího serveru je jediný bezpečný přístup. Zároveň je možný způsob ověření konzistence a autenticity dat pomocí předem dohodnutého klíče s využitím symetrického šifrování. (11) (18)

Obrázek 3 – Základní sekvenční diagram OIDC



Zdroj: https://openid.net/specs/openid-connect-basic-1_0.html (2023)

Hlavním rozšířením OpenID Connect protokolu je ID token, který je možný získat od autentizačního serveru spolu se standardním přístupovým tokenem (access token). ID token je standardní JSON Web Token (JWT), který obsahuje sadu volně definovaných claimů (atributů) uživatele, které představují jeho identitu. Jednoduše si to můžeme představit jako digitální občanský průkaz přihlášeného uživatele. ID token musí být také digitálně podepsaný, aby byl důvěryhodný pro klientskou aplikaci. Na rozdíl od přístupového tokenu, který se využívá k autentizaci na volaném API, tak ID token zůstává na straně klientské aplikace pro informativní účely. Protokol je navržen tak, aby byl rozšiřitelný a umožňoval přidání dalších vlastností a funkčností pro specifické potřeby. Existuje mnoho implementací

a knihoven, které usnadňují integraci tohoto protokolu do implementovaných aplikací a služeb. (11)

Následuje ukázka úspěšného autorizačního průchodu a jeho požadavků i odpovědí mezi klientskou aplikací a autentizačním serverem (Authorization Code Flow). (18)

1. Začínáme odesláním autorizačního požadavku na autentizační server.

```
GET /authorize?
  response_type=code
  &scope=openid%20profile%20email
  &client_id=s6BhdRkqt3
  &state=af0ifjsldkj
  &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb HTTP/1.1
Host: server.example.com
```

2. Následně je uživatel autentizován některou z podporovaných metod (přesměrování na stránku k zadání jména a hesla, Kerberos SSO (Single Sign On) přihlášení na Windows a jiné). Součástí může být i vícefaktorové ověření.
3. Po úspěšném ověření uživatele je odeslána odpověď obsahující přesměrování (HTTP status 302) zpět do aplikace (redirect_uri) s code a state atributem.

```
HTTP/1.1 302 Found
Location: https://client.example.org/cb?
  code=SplxlOBeZQQYbYS6WxSbIA
  &state=af0ifjsldkj
```

4. Klientská aplikace si nyní za pomocí získaného code atributu požádá o přístupový token a ID token.

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
  &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

5. V rámci úspěšné odpovědi autentizačního serveru získá klientská aplikace ID token, přístupový token a obnovovací token (refresh token).

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
 "access_token": "SlAV32hkKG",
 "token_type": "Bearer",
 "refresh_token": "8xLOxBtZp8",
 "expires_in": 3600,
 "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc
   yI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5
   NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAibi0wUzZ
   fV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEyODA5Nz
   AKfQ.ggW8hZ1EuVLuxNuuIJKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6q
   Jp6IcmD3HP99Obi1PRs-cwh3LO-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJ
   NqeGpe-gccMg4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJ1zZwgjxqGByKHiOtX7Tpd
   QyHE5lcMiKPXfEIQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJbOEoRoS
   K5hoDalrcvRYLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVVk4
   XUVrWOLrLl0nx7RkKU8NXNHq-rvKMzqg"
}
```

3.1.2.6 Security Assertion Markup Language (SAML) 2.0

SAML standard verze 2.0 je otevřený protokol vydaný v roce 2005 organizací OASIS pro zabezpečenou výměnu autentizačních a autorizačních dat mezi dvěma bezpečnostními entitami.

- Identity Provider (IDP) – systém zodpovědný za ověření identity uživatelů
- Service Provider (SP) – aplikace, která vyžaduje autentizaci uživatele (19)

SAML protokol je založen na XML formátu zpráv a je založen na výměně SAML tvrzení (Assertion). Toto tvrzení je XML dokument vytvořený na straně IDP, který zapouzdřuje určitá data. Každé tvrzení má následující strukturu.

- ID tvrzení
- Subjekt
- Podmínky pro ověření tvrzení
- Přídavné informace
- Vydavatele tvrzení a jeho podpis

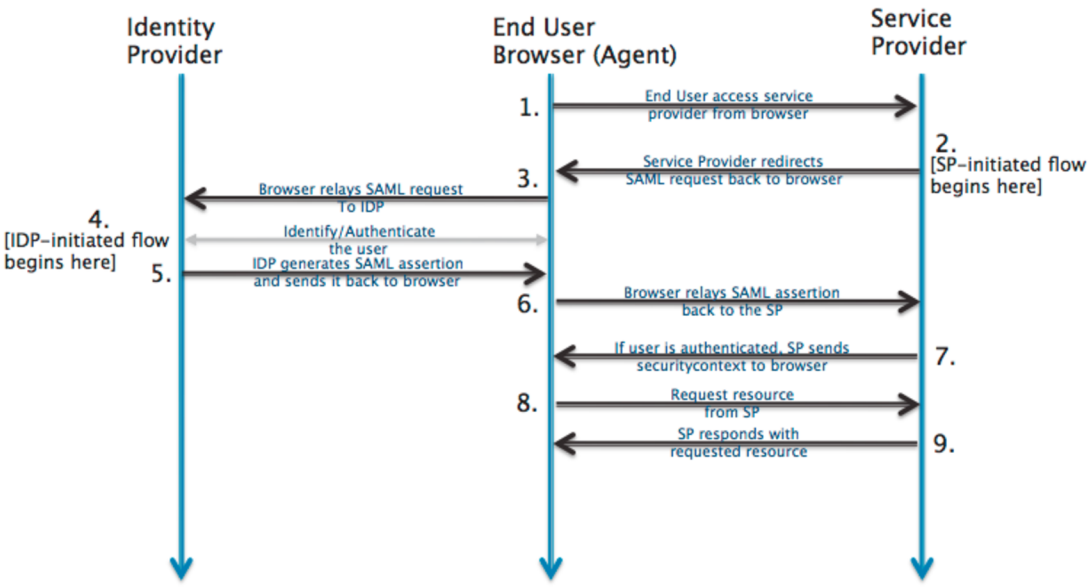
Existují 3 základní typy tvrzení:

- Autentizace – informace o autentizovaném subjektu
- Autorizace – povolení, že subjekt může přistupovat k určitým zdrojům
- Atribut – atributy svázané se subjektem (20)

SAML protokol umožňuje použití několika mechanismů pro přenos tvrzení mezi IDP a SP. Tyto mechanismy se nazývají vazby (bindings) a mohou realizovat přesměrování (redirect) nebo využit protokol HTTP a metodu POST. SAML zahrnuje metadatový záznam (metadata) pro umožnění interoperability mezi IDP a SP, který obsahuje informace o konfiguraci veřejných entit (například veřejný klíč pro elektronický podpis). Tyto metadatové záznamy jsou často publikovány na volně dostupné adrese. V porovnání s OIDC je SAML více uzavřený a méně rozšiřitelný. Zároveň je komplikovanější na implementaci vzhledem k použitému XML formátu dat a obsáhlosti. (19) (20)

Na obrázku 4 je pomocí sekvenčního diagramu znázorněn základní autentizační a autorizační proces za pomocí SAML

Obrázek 4 – Základní sekvenční diagram SAML 2.0



Zdroj: https://developer.okta.com/img/saml/saml_guidance_saml_flow.png (2023)

Autentizační a autorizační proces začíná odesláním požadavku na IDP ze strany SP na který klientská aplikace zaslala požadavek o přístup k požadované službě nebo datům, které podléhají ověření uživatele a jeho přístupových práv. Ukázka takového požadavku je obrázku 5.

Obrázek 5 – Náhled na autentizační požadavek SAML 2.0

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
ID="ONELOGIN_809707f0030a5d00620c9d9df97f627afe9dcc24"
Version="2.0"
ProviderName="SP test"
IssueInstant="2014-07-16T23:52:45Z"
Destination="http://idp.example.com/SSOService.php"
ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
AssertionConsumerServiceURL="http://sp.example.com/demo1/index.php?acs">
<saml:Issuer>http://sp.example.com/demo1/metadata.php</saml:Issuer>
<samlp:NameIDPolicy Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress"
AllowCreate="true" />
<samlp:RequestedAuthnContext Comparison="exact">
<saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport</saml:AuthnContextClassRef>
</samlp:RequestedAuthnContext>
</samlp:AuthnRequest>
```

Zdroj: https://www.samltool.com/generic_sso_res.php (2023)

Odpověď IDP po úspěšné autentizaci uživatele může být rozdílná dle konfigurace výše uvedeného požadavku. Zde je uvedeno 8 příkladů konfigurací odpovědí a jedné vyobrazené na obrázku 6. (21)

- Nepodepsaná SAML odpověď s nepodepsaným tvrzením

- Nepodepsaná SAML odpověď s podepsaným tvrzením
- Podepsaná SAML odpověď s nepodepsaným tvrzením
- Podepsaná SAML odpověď s podepsaným tvrzením
- Nepodepsaná SAML odpověď se zašifrovaným tvrzením
- Nepodepsaná SAML odpověď se zašifrovaným podepsaným tvrzením
- Podepsaná SAML odpověď se zašifrovaným tvrzením
- Podepsaná SAML odpověď se zašifrovaným podepsaným tvrzením

Obrázek 6 – Ukázka nepodepsané SAML 2.0 odpovědi s podepsaným tvrzením

```
<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" ID="_8e8dc5f69a98cc4c1f
<saml:Issuer>http://idp.example.com/metadata.php</saml:Issuer>
<samlp:Status>
<samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
</samlp:Status>
<saml:Assertion xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://www.w3.org/2001/XMLSchema" ID="pfx11b65bfc-4ce0-f362-1
<saml:Issuer>http://idp.example.com/metadata.php</saml:Issuer><ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo><ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
  <ds:Reference URI="#pfx11b65bfc-4ce0-f362-1ff2-c14d152cb6d0"><ds:Transforms><ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enve
<ds:KeyInfo><ds:X509Data><ds:X509Certificate>MIICajCCAdOgAwIBAgIBADANBgkqhkiG9w0BAQ0FADBSMQswCQYDVQQGEwJ1czETMBEGA1UECAwKQ2FsaWZvcm5pYTEVMBMG
<saml:Subject>
<saml:NameID SPNameQualifier="http://sp.example.com/demo1/metadata.php" Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">_ce3d294
<saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
<saml:SubjectConfirmationData NotOnOrAfter="2024-01-18T06:21:48Z" Recipient="http://sp.example.com/demo1/index.php?acs" InResponseTo="ONELOGI
</saml:SubjectConfirmation>
</saml:Subject>
<saml:Conditions NotBefore="2014-07-17T01:01:18Z" NotOnOrAfter="2024-01-18T06:21:48Z">
<saml:AudienceRestriction>
<saml:Audience>http://sp.example.com/demo1/metadata.php</saml:Audience>
</saml:AudienceRestriction>
</saml:Conditions>
<saml:AuthnStatement AuthnInstant="2014-07-17T01:01:48Z" SessionNotOnOrAfter="2024-07-17T09:01:48Z" SessionIndex="_be9967abd904ddcae3c0eb4189
<saml:AuthnContext>
<saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:Password</saml:AuthnContextClassRef>
</saml:AuthnContext>
</saml:AuthnStatement>
<saml:AttributeStatement>
<saml:Attribute Name="uid" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
<saml:AttributeValue xsi:type="xs:string">test</saml:AttributeValue>
</saml:Attribute>
<saml:Attribute Name="mail" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
<saml:AttributeValue xsi:type="xs:string">test@example.com</saml:AttributeValue>
</saml:Attribute>
<saml:Attribute Name="eduPersonAffiliation" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
<saml:AttributeValue xsi:type="xs:string">users</saml:AttributeValue>
<saml:AttributeValue xsi:type="xs:string">examplerole1</saml:AttributeValue>
</saml:Attribute>
</saml:AttributeStatement>
</saml:Assertion>
</samlp:Response>
```

Zdroj: https://www.samltool.com/generic_sso_res.php (2023)

3.1.3 Autorizace

Autorizace je proces, jehož výsledkem je informace, která obsahuje informace, co uživatel smí provádět po úspěšné autentizaci. Tento proces zahrnuje udělení nebo odepření přístupu k určitým zdrojům, datům nebo funkcím v rámci systému nebo aplikace. Autorizace je klíčovým prvkem zabezpečení a ochrany dat a pomáhá organizacím omezit přístup k citlivým informacím a chránit zdroje (služby) před neoprávněným použitím. (3)

Přístup k určitým zdrojům je většinou řízen pomocí rolí. Role jsou pojmenované sady oprávnění, které jsou přiřazeny uživatelům nebo skupinám uživatelů, kteří mají danou roli přiřazenou. Díky tomu je snazší správa oprávnění, protože můžete přiřadit oprávnění k určitému zdroji skrze roli skupině uživatelů namísto každému zvlášť. S pojmem autorizace také souvisí oprávnění (permission). Jsou to pravidla nebo pravomoci, které určují, co může uživatel provádět. Například oprávnění v rámci souborového systému, kde máme sadu oprávnění pro čtení, zápis a spouštění souborů. Podobně se může jednat o pravomoci v rámci databáze (select, update, insert, delete). Kontrola přístupu (access control) se vztahuje k mechanismům a pravidlům, která určují, kdo může přistupovat k určitým zdrojům nebo datům a za jakých podmínek. To může zahrnovat pravidla založená na uživatelích, rolích, čase, umístění a dalších faktorech. Pro zajištění řízené autorizace v rámci organizací, jsou definovány bezpečnostní politiky (security policy). Bezpečnostní politika obsahuje pravidla a směrnice, které určují, jakým způsobem se autorizace provádí v organizaci. Zahrnuje definici rolí, oprávnění a způsoby, jakými jsou oprávnění přidělována a spravována. (22)

Podobně jako v oblasti autentizace existují i pro implementaci autorizace online zdrojů a služeb vydané a udržované standardy. Tyto standardy umožňují bezpečnou a jednotnou implementaci v provozovaných systémech organizace. Jeden z nejznámějších standardů je OAuth 2.0 protokol, který je základem pro autentizační standard OIDC. (11)

3.1.3.1 Open Authorization (OAuth) verze 2.0

OAuth protokol verze 2.0 je otevřený standard pro autorizaci klientů třeti-stranových aplikací k omezenému přístupu ke službách skrze online API. První verze OAuth standardu byla touto verzí kompletně vytlačena a je zpětně nekompatibilní. OAuth 2.0 více odpovídá současným aplikacím (webové i mobilní), je mnohem jednodušší na použití, je rozšiřitelná pro více scénářů použití, využívá jednoduchých tokenů různého typu a významu a poskytuje podporu pro více faktorovou autentizaci. (24) (25)

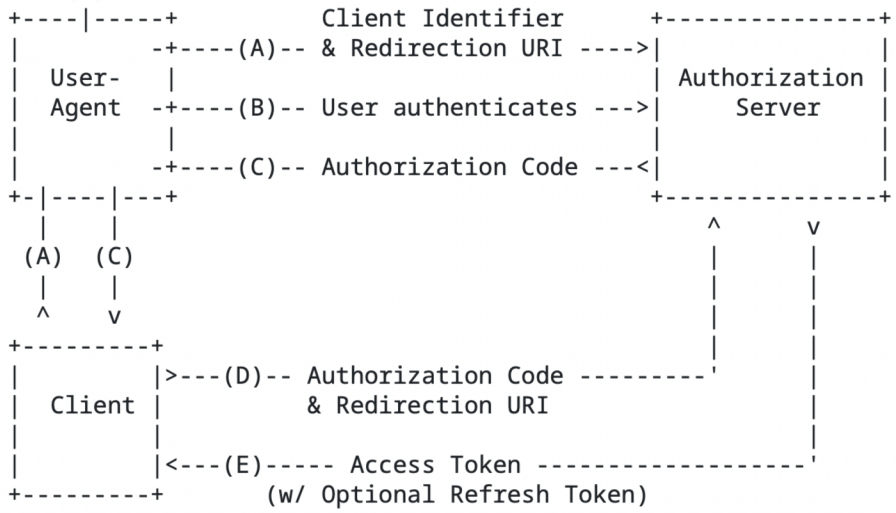
Standard definuje 4 role, které se v komunikaci vyskytují.

- **Client** (uživatel co požaduje přístup ke zdroji)
- **Resource owner** (vlastník zdroje)
- **Authorization server** (autorizační server, který vydává přístupové tokeny)
- **Resource server** (server poskytující zdroje)

Standard definuje několik typů průchodů (authorization flow) a je třeba vybrat vhodný dle typu cílové aplikace a požadované úrovně zabezpečení. Níže jsou uvedeny ty nejzákladnější.

Authorization Code Grant – průchod na obrázku 7 při kterém klientská aplikace získá v rámci HTTP přesměrování (status odpovědi 302) současně přístupový token (access token) i obnovovací token (refresh token). Pro zvýšení bezpečnosti se v rámci přesměrování předává pouze autorizační kód (code), který je použitelný pouze k získání výše uvedených tokenů již na pozadí aplikace. Protože průchod obsahuje HTTP přesměrování je vhodný pro webové aplikace. (26)

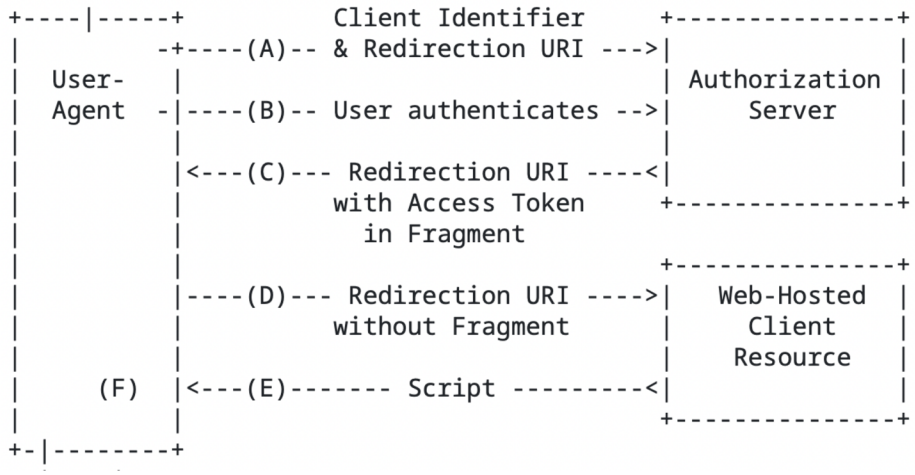
Obrázek 7 – Sekvenční diagram pro OAuth 2.0 Authorization Code Grant



Zdroj: https://datatracker.ietf.org/doc/html/rfc6749 (2023)

Implicit Grant – průchod na obrázku 8 je vhodný, pokud klientská aplikace nemá možnost ukládat důvěrné informace, jako je autorizační kód (code). Podobně jako Authorization Code Grant používá HTTP přesměrování a je vhodný pro webové aplikace. Nicméně má nižší úroveň zabezpečení, a proto by měl být před použitím pečlivě zvážen. Je důležité zajistit, že komunikace mezi klientem a autorizačním serverem je šifrovaná pomocí HTTPS. Přístupový token (access token) je získán v rámci prvního přesměrování a není možné získat obnovovací token (refresh token). (26)

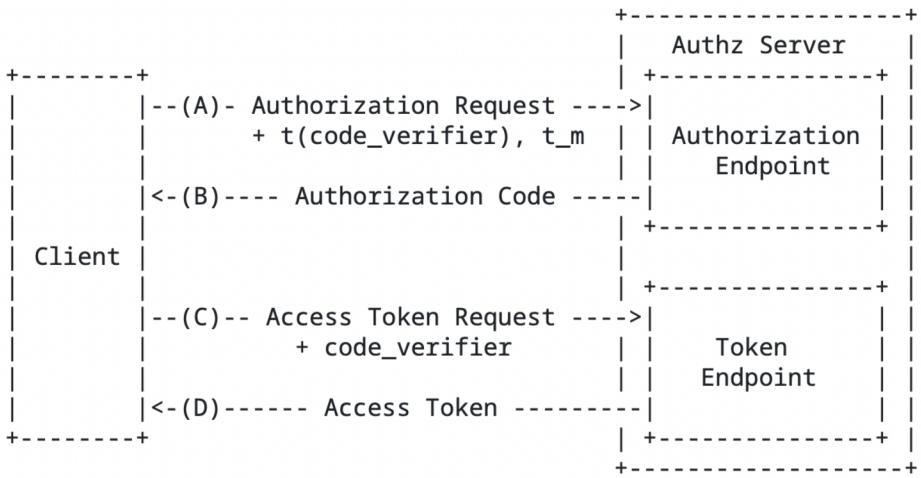
Obrázek 8 – Sekvenční diagram pro OAuth 2.0 Implicit Grant



Zdroj: https://datatracker.ietf.org/doc/html/rfc6749 (2023)

Proof Key for Code Exchange (PKCE) – průchod na obrázku 9, který je rozšířením výše zmíněného průchodu Authorization Code Grant v rámci původního standardu OAuth 2.0 pro zvýšení zabezpečení na zařízeních, kde není možné bezpečné ukládání důvěrných informací, aby nedošlo k jejich ukradení útočníkem. (27)

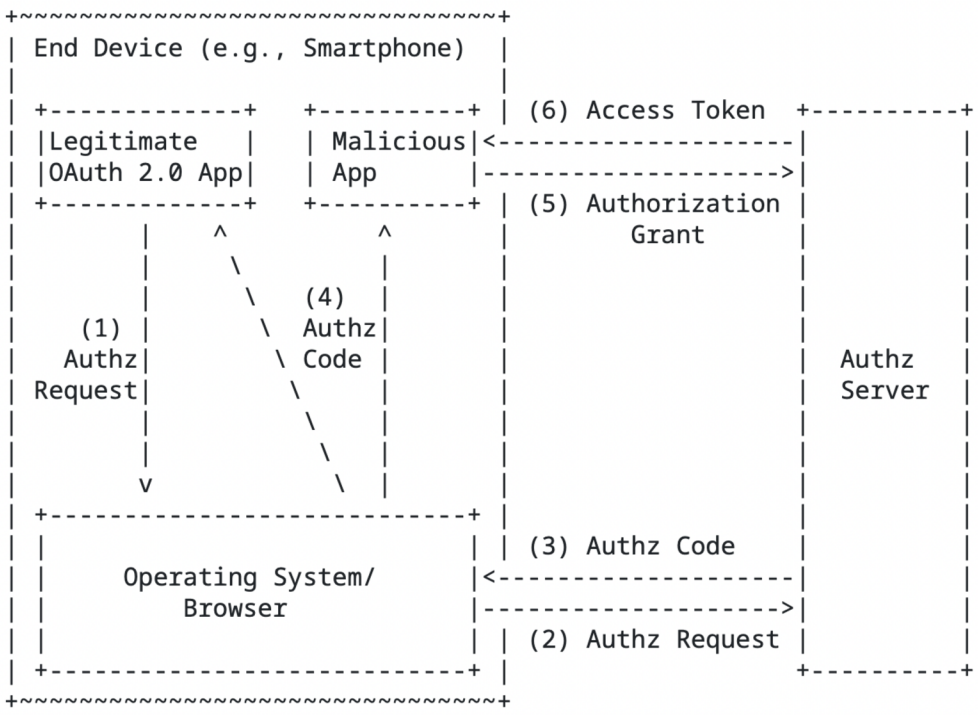
Obrázek 9 – Sekvenční diagram pro OAuth 2.0 PKCE



Zdroj: https://datatracker.ietf.org/doc/html/rfc7636 (2023)

Rozšířením je, že klientská aplikace generuje náhodný řetezec (code_verifier) a generuje jeho hash (code_challenge). Hash a hashovací funkci (například SHA-256) odesílá v rámci autorizačního požadavku a autorizační server si tyto údaje ukládá. Náhodný řetězec je pak odesílán při výměně autorizačního kódu (code) za přístupový token (access token). Autorizační server si generuje vlastní hash pomocí uložené hashovací funkce a získaného náhodného řetězce. Následně validuje vygenerovaný hash s uloženým hashem. Byl navržen k ochraně před útokem, který by mohl ohrozit utajení autorizačního kódu (code), což by mohlo vést ke kompromitaci bezpečnosti aplikace. PKCE je zejména důležitý, pokud používáte Authorization Code Grant v aplikacích, kde není možné ukládat důvěrné informace. Útočníkova aplikace běžící ve stejném zařízení by se mohla dostat k autorizačnímu kódu v případě, kdy si jej aplikace nezabezpečeně ukládá v rámci operačního systému nebo prohlížeče. (27)

Obrázek 10 – Ukázka zranitelnost OAuth 2.0 Authorization Code Grant



Zdroj: https://datatracker.ietf.org/doc/html/rfc7636 (2023)

3.2 Architektura dnešních aplikací

Nelze najít jednotnou univerzální architekturu aktuálně moderního systému, protože systémy se mohou lišit dle použitých technologií, systému provozu, velikostí systému, integrovanými systémy či jinými důležitými vlastnostmi. Můžeme však definovat charakteristiky moderního systému, které by daná architektura měla splňovat. (28)

3.2.1 Charakteristika moderní aplikační architektury

Cloud native – aplikace by měla být navrhována tak, že je možné ji nasazovat a provozovat v cloudu (Google Cloud, Amazon AWS, Microsoft Azure a jiné) a byla integrovatelná na cloudové služby (datové zdroje, monitorovací nástroje, analytické nástroje a jiné).

Škálovatelnost – aplikace nebo její moduly by měly umožňovat snadné vertikální škálování (navyšování HW prostředků pro dosažení vyššího výkonu) a horizontální škálování (přidávání dalších instancí stejné služby nebo modulu pro dosažení vyššího výkonu). Tato vlastnost je velice důležitá v cloudovém prostředí, kde aplikace či její moduly jsou provozovány v kontejnerech (Docker, Kubernetes orchestration).

Nezávislost na platformě – výběr technologií, programovacích jazyků a způsob provozu aplikace by měl být nezávislý na platformě operačního systému, kde bude aplikace provozována (Unix, MacOS, Windows a jiné). Tuto vlastnost například splňují kontejnerové aplikace, které jsou zabaleny včetně vlastního běhového prostředí do spustitelného obrazu (Docker image) a ten spouštíme jednotně v rámci kontejneru (Docker container).

Modulárnost – aplikace se skládá ze vzájemně nezávislých modulů namísto monolitické architektury, kdy je celá aplikace v rámci jednoho modulu. Moduly jsou mezi sebou závislé pouze funkčně. Díky tomu můžeme moduly jednodušeji udržovat a případně nahrazovat.

API rozhraní – aplikace nebo její moduly by měly poskytovat data či služby skrze vlastní API rozhraní (REST, GraphQL a jiné). Totéž platí i pro konzumovaná data a služby. Díky tomu můžeme moduly nahrazovat za nové bez dopadu na konzumenty, jelikož máme jasně definovaný kontrakt v rámci API rozhraní. I aplikace samotná by měla mít oddělený frontend a backend aplikace a propojený skrze API rozhraní (technologická nezávislost, aplikace třetích stran mohou konzumovat naše služby v rámci svého frontendu).

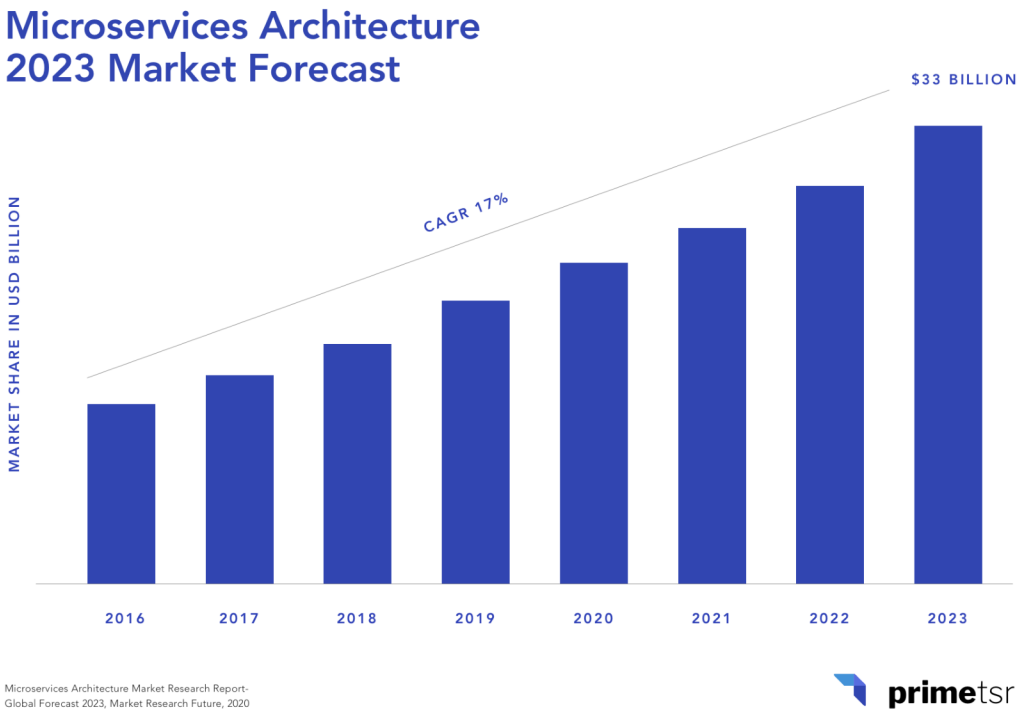
Kontinuální integrace a nasazení (CI/CD) – aplikaci či jednotlivé moduly by měly podporovat automatizovaný proces sestavení a doručování, který umožňuje rychlé a spolehlivé nasazování změn do produkce.

Automatizované testování – tato vlastnost souvisí s CI/CD procesem, který můžeme rozšířit o spouštění automatizovaných testů v rámci sestavení aplikace (jednotkové testy) nebo v rámci doručování (integrační testování). Díky tomu urychlíme nasazování změn do produkce (vynechání manuálního testování) a zároveň zvýšíme stabilitu. (30)

3.2.2 Mikroservisní architektura (microservices)

Posledních pár let nejvíce využívaný architektonický styl v návrhu aplikací je rozdělení aplikace do mikroslužeb. Organizace na základě nalezeného průzkumu investují nemalé peníze na přechod z monolitických systémů na mikroservisní architekturu a předpokládá se další růst. (31)

Obrázek 11 – Odhady investic do mikroservisní architektury

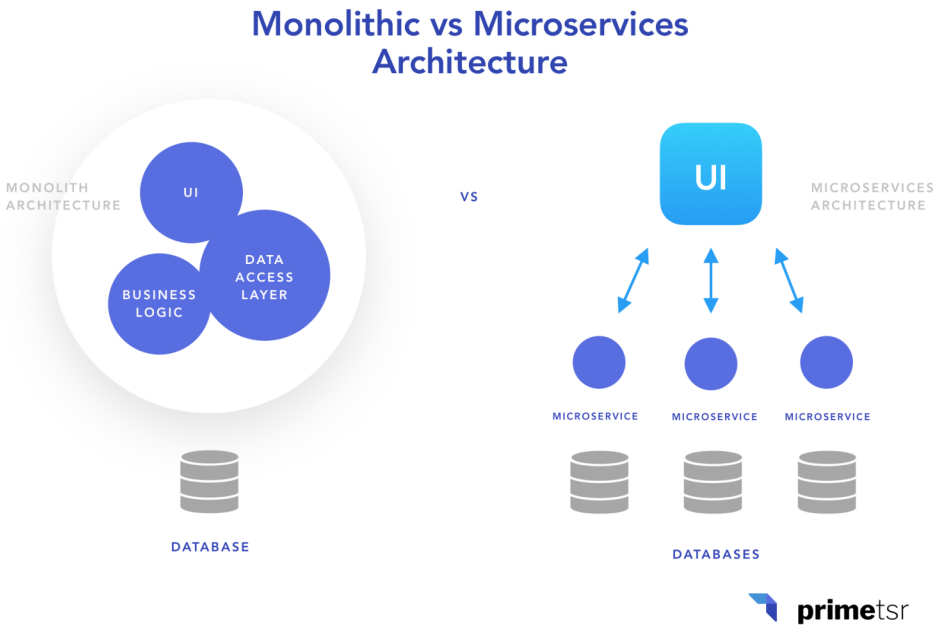


Zdroj: https://primetsr.com/wp-content/uploads/2020/03/Microservices-Market-Size.png (2023)

Tato architektura se vyznačuje přístupem k návrhu a vývoji softwarových aplikací, ve kterém jsou aplikace rozděleny na malé a nezávislé moduly, nazývané mikroslužby. Každá

mikroslužba má jasně definovanou oblast (business entita) na kterou jsou její služby orientované. Pro okolní svět má jasně definované rozhraní (REST API) a v případě úprav je třeba myslet na zpětnou kompatibilitu. Tyto mikroslužby mohou být vyvíjeny, nasazovány a škálovány nezávisle na sobě. Díky tomu se na vývoji a provozu jedné aplikace může účastnit i více vývojových týmů. Mikroslužby mohou konzumovat služby cizích aplikací nebo komunikovat mezi sebou v rámci jedné aplikace. Mezi modulová komunikace může vést ke zvýšení komplexity aplikace a navýšení latencí na jednotlivých službách. Tato architektura je vhodná zejména pro velké a komplexní aplikace, které potřebují rychle reagovat na změny a škálovatelnost. V případě, kdy komplexní aplikace je monolitická, tak je velice náročný proces jejího rozšiřování či obnovování některých částí, neboť vždy pracujeme a upravuje aplikaci jako celek. Srovnání těchto dvou typů architektur ukazuje obrázek 12. (29) (33)

Obrázek 12 – Rozdíl mezi monolitickou a mikroservisní architekturou



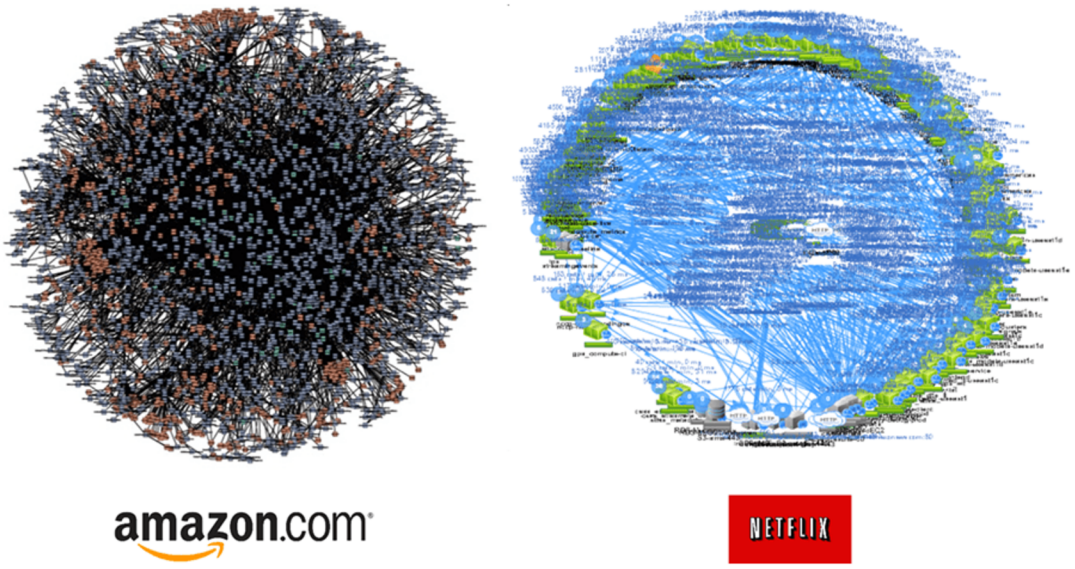
Zdroj: https://primetsr.com/wp-content/uploads/2020/03/micro-vs-monolith.png (2023)

Mikroservisní architektura nemusí být vždy nejlepším řešením a v případě špatného návrhu či nedostatečné maturity vývojového a provozního týmu může problémy prohloubit. Citace z odborného článku: *„Provozování silně orientovaného mikroservis systému bude klást větší nároky na zdroje, dokumentaci, logování, monitoring, řízení změn, a tedy i*

vyzrálost odpovídajících procesů, a nástrojů umožňujících tuto automatizaci, a je otázka, zda se díky lepší možnosti škálování podaří skutečně dosáhnout očekávaných úspor.“ (32)

Celosvětově známé firmy jako Netflix, eBay, Amazon, Twitter, PayPal a další používají mikroservisní architekturu, jak ukazuje obrázek 13. (31)

Obrázek 13 – Ukázka množství mikroslužeb Amazonu a Netflixu



Zdroj: https://primetsr.com/wp-content/uploads/2020/03/amazon_netflix.png (2023)

Základní výhody a nevýhody mikroservisní architektury jsou přehledně uvedeny v tabulce 1. (32) (33)

Tabulka 1 – Srovnání výhod a nevýhod mikroservisní architektury

Výhody	**Nevýhody**
Jednodušší vývoj (menší moduly)	Složitější provoz (mnoho modulů)
Jednodušší modernizace aplikace	Latence (mezimodulová komunikace)
Znovu použitelnost modulů	Řešení distribuovaných transakcí
Škálovatelnost modulů	Náročnější na hardware
Jednoduší integrace vývojáře	
Není jeden bod selhání	
Rozdělení dat pro rychlejší škálovatelnost	

Zdroj: Vlastní zpracování (2023)

3.3 Kybernetická bezpečnost

3.3.1 Legislativa ČR a EU

V ČR upravuje práva a povinnosti osob a působnost a pravomoci orgánů veřejné moci v oblasti kybernetické bezpečnosti ***zákon 181/2014 Sb. o kybernetické bezpečnosti a o změně souvisejících zákonů (zákon o kybernetické bezpečnosti)***, který byl ještě v průběhu upraven změnami 104/2017 Sb., 183/2017 Sb., 205/2017 Sb., 35/2018 Sb., 12/2020 Sb., 111/2019 Sb., 12/2020 Sb., 261/2021 Sb. Tento zákon zapracovává příslušný předpis Evropské unie ***SMĚRNICE EVROPSKÉHO PARLAMENTU A RADY (EU) 2016/1148 ze dne 6. července 2016 o opatřeních k zajištění vysoké společné úrovně bezpečnosti sítí a informačních systémů v Unii*** a navazuje na předpis Evropské unie ***NAŘÍZENÍ EVROPSKÉHO PARLAMENTU A RADY (EU) 2019/881 ze dne 17. dubna 2019 o agentuře ENISA („Agentuře Evropské unie pro kybernetickou bezpečnost“), o certifikaci kybernetické bezpečnosti informačních a komunikačních technologií a o zrušení nařízení (EU) č. 526/2013 („akt o kybernetické bezpečnosti“)*** a upravuje zajišťování bezpečnosti sítí elektronických komunikací a informačních systémů. (35) (36) (37)

Zákon a směrnice definují nařízení v oblasti kybernetické bezpečnosti pro poskytovatele základních služeb (kritické infrastruktury). Poskytovatel základní služby může být veřejný nebo soukromý subjekt. Kritéria pro určení provozovatele základních služeb jsou následující.

- Subjekt poskytuje službu, která je základní z hlediska zachování kritických společenských nebo ekonomických činností.
- Poskytování dotyčné služby je závislé na sítích a informačních systémech a incident by vedl k významnému narušení poskytování této služby. (36)

Tabulka 2 – Subjekty poskytující základní službu

Odvětví	Pododvětví	Druh subjektu
Energetika	Elektřina	Elektroenergetické podniky ve smyslu čl. 2 bodu 35 směrnice Evropského parlamentu a Rady 2009/72/ES, které zastávají funkci „dodávky“ ve smyslu čl. 2 bodu 19 uvedené směrnice
		Provozovatelé distribuční soustavy ve smyslu čl. 2 bodu 6 směrnice 2009/72/ES
		Provozovatelé přenosové soustavy ve smyslu čl. 2 bodu 4 směrnice 2009/72/ES

	Ropa	Provozovatelé ropovodů
		Provozovatelé zařízení na těžbu, rafinaci a zpracování ropy a skladovacích a přenosových zařízení
	Zemní plyn	Dodavatelské podniky ve smyslu čl. 2 bodu 8 směrnice Evropského parlamentu a Rady 2009/73/ES
		Provozovatelé distribuční soustavy ve smyslu čl. 2 bodu 6 směrnice 2009/73/ES
		Provozovatelé přepravní soustavy ve smyslu čl. 2 bodu 4 směrnice 2009/73/ES
		Provozovatelé skladovacího zařízení ve smyslu čl. 2 bodu 10 směrnice 2009/73/ES
		Provozovatelé zařízení LNG ve smyslu čl. 2 bodu 12 směrnice 2009/73/ES
		Plynárenské podniky ve smyslu čl. 2 bodu 1 směrnice 2009/73/ES
		Provozovatelé zařízení na rafinaci a zpracování zemního plynu
Doprava	Letecká doprava	Letečtí dopravci ve smyslu čl. 3 bodu 4 nařízení Evropského parlamentu a Rady (ES) č. 300/2008
		Řídící orgány letiště ve smyslu čl. 2 bodu 2 směrnice Evropského parlamentu a Rady 2009/12/ES, letiště ve smyslu čl. 2 bodu 1 uvedené směrnice, včetně hlavních letišť uvedených v příloze II, části 2 nařízení Evropského parlamentu a Rady (EU) č. 1315/2013; a subjekty provozující pomocná zařízení v rámci letišť
		Provozovatelé kontroly řízení provozu poskytující služby řízení letového provozu ve smyslu čl. 2 bodu 1 nařízení Evropského parlamentu a Rady (ES) č. 549/2004
	Železniční doprava	Provozovatelé infrastruktury ve smyslu čl. 3 bodu 2 směrnice Evropského parlamentu a Rady 2012/34/EU
		Železniční podniky ve smyslu čl. 3 bodu 1 směrnice 2012/34/EU, včetně provozovatelů zařízení služeb ve smyslu čl. 3 bodu 12 směrnice 2012/34/EU
	Vodní doprava	Společnosti vnitrozemské, námořní a pobřežní osobní a nákladní vodní dopravy, jak jsou vymezeny pro námořní dopravu v příloze I nařízení Evropského parlamentu a Rady (ES) č. 725/2004, kromě jednotlivých plavidel provozovaných těmito podniky
		Řídící orgány přístavů ve smyslu čl. 3 bodu 1 směrnice Evropského parlamentu a Rady 2005/65/ES, včetně jejich přístavních zařízení ve smyslu čl. 2 bodu 11

		Nařízení (ES) č. 725/2004; a subjekty provozující díla a zařízení v rámci přístavů
		Provozovatelé služeb lodní dopravě ve smyslu čl. 3 písm. o) směrnice Evropského parlamentu a Rady 2002/59/ES
	Silniční doprava	Silniční orgány ve smyslu čl. 2 bodu 12 nařízení Komise v přenesené pravomoci (EU) 2015/962 odpovědné za kontrolu řízení provozu
		Provozovatelé inteligentních dopravních systémů ve smyslu čl. 4 bodu 1 směrnice Evropského parlamentu a Rady 2010/40/EU
Bankovnictví		Úvěrové instituce ve smyslu čl. 4 bodu 1 nařízení Evropského parlamentu a Rady (EU) č. 575/2013
Infrastruktura finančních trhů		Provozovatelé obchodních systémů ve smyslu čl. 4 bodu 24 směrnice Evropského parlamentu a Rady 2014/65/EU
		Ústřední protistrany ve smyslu čl. 2 bodu 1 nařízení Evropského parlamentu a Rady (EU) č. 648/2012
Zdravotnictví	Zdravotnická zařízení	Poskytovatelé zdravotní péče ve smyslu čl. 3 písm. g) směrnice Evropského parlamentu a Rady 2011/24/EU
Dodávky a rozvody pitné vody		Dodavatelé a distributoři „vody určené k lidské spotřebě“ ve smyslu čl. 2 bodu 1 písm. a) směrnice Rady 98/83/ES, avšak kromě distributorů, pro něž je distribuce vody určené k lidské spotřebě pouze částí jejich obecné činnosti spočívající v distribuci komodit a zboží, která není považována za základní službu
Digitální infrastruktura		Výměnné uzly internetu (IXP)
		Poskytovatelé služeb systému doménových jmen (DNS)
		Registry internetových domén nejvyšší úrovně (TLD)

Zdroj: https://eur-lex.europa.eu/legal-content/CS/TXT/HTML/?uri=CELEX:32016L1148 (2023)

Poskytovatelé (subjekty) základních služeb definování dle tabulky 2 musí mít předepsaná a zavedená následující opatření, které může Úřad kontroly vyžadovat při kontrole.

- Bezpečnostními opatření
  - Organizační
  - Technická
- Organizačními opatření

- Řízení bezpečnosti informací
- Řízení rizik
- Bezpečnostní politika
- Organizační bezpečnost
- Stanovení bezpečnostních požadavků pro dodavatele
- Řízení aktiv
- Bezpečnost lidských zdrojů
- Řízení provozu a komunikací
- Řízení přístupu osob
- Akvizice, vývoj a údržba
- Zvládání kybernetických bezpečnostních událostí a kybernetických bezpečnostních incidentů
- Řízení kontinuity činností a mít kontrolu a audit

- Technická opatření
  - Fyzická bezpečnost
  - Nástroj pro ochranu integrity komunikačních sítí
  - Nástroj pro ověřování identity uživatelů
  - Nástroj pro řízení přístupových oprávnění
  - Nástroj pro ochranu před škodlivým kódem
  - Nástroj pro zaznamenávání činnosti informačního nebo komunikačního systému, jeho uživatelů a administrátorů
  - Nástroj pro detekci kybernetických bezpečnostních událostí
  - Nástroj pro sběr a vyhodnocení kybernetických bezpečnostních událostí
  - Aplikační bezpečnost
  - Kryptografické prostředky
  - Nástroj pro zajišťování úrovně dostupnosti informací
  - Bezpečnost průmyslových a řídicích systémů (35)

3.3.2 Bezpečnostní doporučení NÚKIB pro administrátory

NÚKIB vydal přehlednou několika stránkovou příručku, která má pomoci všem organizacím ke zlepšení své bezpečnosti. Doporučení jsou rozdělaná do několika kategorií.

Oblast autentizace a autorizace se objevuje v každé kategorii vypsaných do jednotlivých podkapitol s doslovnou citací vydaného doporučení. (39)

3.3.2.1 Infrastruktura

Kategorie zabývající se infrastrukturním vybavením organizace má následující doporučení.

- Zavedení segmentace sítě na menší celky a segregace uživatelských oprávnění napříč uživateli.
- Blokování škodlivých IP adres již na úrovní gateway pomocí blacklistů.
- Využití systémů IDS/IPS k detekci a prevenci průniku díky používané heuristické analýze k identifikování anomálního provozu v rámci sítě.
- Sledování síťového provozu pomocí sond (komunikace do internetu, komunikace mezi servery a další)
- Uchování síťového provozu alespoň na 12 měsíců pro případné forenzní zkoumání. V případě kritické informační infrastruktury (KII) a u informačních systémů základní služby (PZS) podle zákona o kybernetické bezpečnosti a návazných vyhlášek je minimální lhůta 18 měsíců.
- Kontrola příchozí pošty pomocí mechanismů Sender ID, SPF (Sender Policy Framework), DKIM (Domain Keys Identified Mail) a DMARC (Domain based Message Authentication, Reporting and Conformance).
- Použití šifrovaného spojení mezi poštovními servery (TLS) k zajištění důvěrnosti poštovních zpráv.
- Zavedení automatizované dynamické analýzy obsahu emailů a navštěvovaných webů.
- Na firewallu povolit jen žádoucí služby a standardní provoz.
- Kontrola důvěryhodnosti a platnosti používaných klíčů a certifikátů.
- Zavedení whitelistu webových domén (povolené domény) je účinnější než blacklist (zakázané domény).
- Volba jednoduchých vlastních domén, aby byla zřetelnější záměna písmen ve phishingových emailech.
- Zavedení ANTI.DDoS technologie nejlépe ve spolupráci s poskytovatelem internetového připojení.

- Vypracování DRP (Disaster Recovery Plan) pro trénování a následnou rychlou obnovu systému při jejich selhání nebo napadení. (39)

3.3.2.2 Správa účtů

V oblasti uživatelských účtu jsou dána následující doporučení.

- Centrální správa všech uživatelských účtů (jeden účet a heslo v rámci všech aplikací organizace) včetně jednotné politiky s výchozím odebráním rozšířených práv například pro instalaci software nebo spouštění skriptů.
- Vynucování vícefaktorové autentizace, zejména při přístupu k citlivým informacím.
- Oddělené administrátorské účty ke správě serverů od běžných účtů k přístupu například do kancelářských aplikací či elektronické pošty.
- Každý administrátor by měl mít vlastní účet a nevyužívat sdílených účtů.
- Silné zabezpečení lokálních účtů na serveru například na Windows pomocí LAPS (Local Administrator Password Solution).
- Vynucení používání silných hesel s definovanou délkou, platností, omezení používaných stejných nebo slovníkových hesel včetně kontroly kompromitace používaných hesel s následným vynucením jejich změny.
- Zavedení pravidelných kontrol účtů a jejich oprávnění. (39)

3.3.2.3 Stanice a servery

V rámci počítačových stanic a serverů v organizaci jsou následující doporučení.

- Udržování aktuálního operačního systému a instalovaného software s včasným aplikováním všech vydaných bezpečnostních záplat.
- Nepoužívání softwarových produktů, které již nemají aktivní podporu ze strany dodavatele.
- Ověřování identity aplikací a souborů před spuštěním ke kterému můžeme využít v prostředí Windows například Device Guard, AppLocker, či SRP (Zásady omezení softwaru).
- Hardening konfigurace uživatelských aplikací, což znamená, že v rámci aplikací povolujeme jen funkcionality, které jsou nutné pro práci a

nebezpečné funkcionality zakazujeme (například Flash v prohlížeči či makra v MS Office).

- Používání obecných preventivních mechanismů, které můžou systém ochránit například před zero-day zranitelnostmi (DEP – Data Execution Prevention) či SELinux v linuxových systémech.
- Používání IDS/IPS systémů, které umí detekovat anomální chování (injekce kódu do cizích procesů, změny chráněných klíčů, zachytávání kláves, načítání neznámých ovladačů a další).
- Centralizované a časově synchronizované logování síťových událostí.
- Pravidelné zálohování důležitých a citlivých dat.
- Zavedení pro pracovní stanice a servery SOE (Standard Operating Environment), které zajišťuje standardizovanou konfiguraci s vypnutím nežádoucích funkcí.
- Omezení přímého přístupu na internet z pracovních stanic.
- Používání bezpečnostního softwaru (firewall, antivirový program či omezovače spouštění nebezpečných aplikací).
- Zavedení pravidelného šifrování disků.
- Využívání TPM (Trusted Platform Module) pro generování a uložení hesel a kryptografických klíčů.
- Nastavení unikátního hesla pro UEFI/BIOS každé pracovní stanice.
- Vynucení secure bootu pomocí boot manageru s definovaným pořadím a heslem zabezpečenou konfigurací.
- Zavedení ochrany před útoky na uložená hesla pomocí hashování (Argon2, bcrypt, scrypt, PBKDF2) či strojových zkoušení (CAPTCHA).
- Pro správu serverů využívat SSH s přihlašováním pomocí klíče, a nikoliv pomocí hesla.
- Hardening konfigurace serverových aplikací jako jsou databáze, CRM systém, HR systém, ERP systém či účetní systémy k omezení nežádoucích funkcionalit.
- Kontrola přenosných médií včetně vedení evidence povolených USV zařízení.

- Omezení přístupu k SMB (Server Message Block) a NETBIOS na pracovních stanicích a serverech.
- Vynucení VPN připojení všech zařízení, které se připojují mimo síť organizace, jinak omezte síťovou komunikaci ze stanice na minimum.
- Zajištění fyzické bezpečnosti IT hardwaru. (39)

3.3.3 Open Web Application Security Project (OWASP)

OWASP je nezisková organizace založená za účelem zlepšení bezpečnosti aplikací a webových služeb. Zaměřuje se na identifikaci, výzkum a prevenci reálných bezpečnostních rizik a hrozeb. Organizace zahrnuje dobrovolníky, výzkumníky, profesionály a vývojáře z celého světa, kteří spolupracují na vytváření nástrojů, dokumentace a směrnic pro zlepšení bezpečnosti. Nabízí také vzdělávací materiály a tréninkové kurzy zaměřené na bezpečnostní aspekty návrhu a vývoje aplikací. Pořádá konference a workshopy, které umožňují odborníkům v oblasti bezpečnosti a vývojářům se setkávat, sdílet znalosti a diskutovat o aktuálních bezpečnostních tématech. V rámci svých webových stránek poskytuje portál pro výměnu informací a spolupráci mezi vývojáři a zveřejňuje na něm seznam nejrizikovějších bezpečnostních chyb. OWASP dále poskytuje mnoho nástrojů zaměřených na bezpečnost webových aplikací a softwarového vývoje, které mohou být zdarma využity komunitou. Celkově je OWASP klíčovým hráčem v oblasti zlepšení bezpečnosti aplikací a zvyšování povědomí o bezpečnostních hrozbách a rizicích v této oblasti. Jejich práce pomáhá organizacím a jednotlivcům vyvíjet bezpečné aplikace a chránit se před potenciálními bezpečnostními útoky. (5)

OWASP pravidelně aktualizuje seznam 10 nejrizikovějších bezpečnostních chyb ve webových aplikacích (**OWASP Top 10 Vulnerabilities**), který lze použít jako referenční bod pro vývojáře a bezpečnostní odborníky. Seznam těchto zranitelností lze použít jako základní penetrační test aplikace. (38)

A01:2021 Broken Access Control (narušená autentizace) – chyby v autentizačních mechanismech, které mohou umožnit útočníkovi neoprávněný přístup k datům.

- Porušení zásady minimálního oprávnění nebo odepření přístupu ve výchozím nastavení služby. Namísto toho jsou služby dostupné všem ve výchozím stavu.

- Možnost obejít bezpečnostní ověření úpravou parametrů, vnitřního stavu aplikace, HTML stránky nebo úpravou požadavků odesílaných API rozhraní.
- Možnost manuální změny identifikace uživatele k získání cizích oprávnění.
- Na API rozhraní ke čtení dat chybí bezpečnostní ověření pro zbylé http metody POST, PUT, PATCH a DELETE.
- Možnost manuální změny role uživatele k získání vyššího oprávnění.
- Možnost manipulace s metadaty aplikace (JWT, Cookies, Storage) k získání vyššího oprávnění nebo změny identity.
- Nenastavené CORS umožňuje přístup k API z cizích zdrojů.
- Možnost přímého přístupu neautorizovanému uživateli na chráněné stránky, pokud chráníme stránky pouze skrytím odkazů. (38)

A02:2021 Cryptographic Failures (chyby kolem šifrování) – chybějící nebo nesprávné mechanismy šifrování mohou způsobit únik citlivých údajů v rámci přenosu.

- Data jsou přenášena v surové podobě (HTTP, SMTP, FTP).
- Použití zastaralých nebo slabých šifrovacích algoritmů.
- Použití zastaralých nebo slabých hashovacích metod.
- Chybí vynucení HTTPS přenosu a existuje podpora HTTP.
- Aplikace nevaliduje certifikát serveru.
- Používají se výchozí šifrovací klíče nebo nechybí pravidelná obnova či rotace.
- Detailní chybové zprávy jsou zneužitelné.
- Inicializační vektory pro šifrovací algoritmy jsou vynechány.
- Nevhodné algoritmy pro generování náhodných řetězců používaných v šifrovacích a hashovacích funkcích. (38)

A03:2021 Injection (injekce) – je způsob kdy se útočník dostane k chráněným datům nebo provede neautorizovanou operací prostřednictvím neošetřeného vstupu do aplikace.

- Uživatelská vstupní data nejsou validována, filtrována a čištěna.
- Dynamické dotazy nebo neparametrizované volání nejsou escapovány dle kontextu volání a jsou použity přímo.
- Vstupní data jsou přímo použita nebo zřetězena v rámci SQL příkazů.

- Vstupní data jsou přímo použita v rámci vyhledávacích parametrů v objektově relačním mapování (ORM). (38)

A04:2021 Insecure Design (nebezpečnostní návrh) – v případě, kdy na bezpečnost není myšleno již při návrhu aplikace jsou v rámci vývoje zanášeny bezpečnostní zranitelnosti a v rámci testování chybí potřebné testy. Důležité je zapracovávat bezpečnostní mechanismy již do návrhu aplikace, nepoužívat knihovny se známými zranitelnostmi, psát jednotkové a integrační testy, které ověřují zabezpečení. (38)

A05:2021 Security Misconfiguration (nevhodná bezpečnostní konfigurace) – nevhodná konfigurace aplikace může zvýšit bezpečnostní riziko, případně ho přímo způsobit, pokud se jedná o konfiguraci spojenou s bezpečností.

- Nedostatečná revize a omezování konfigurace cloudových služeb a povolujeme i nepotřebné funkce pro běh aplikace.
- Na serveru jsou nainstalovány nepotřebné služby, povolené nepotřebné porty, neomezený přístup k internetu, nepoužívané účty či oprávnění.
- Nastavení detailního výpisu chyb odhaluje detaily aplikace.
- Konfigurace zabezpečení aplikačního serveru, aplikačního frameworku (Struts, Spring, ASP.NET), knihoven nebo databázích není na bezpečných hodnotách.
- Server neposílá bezpečnostní hlavičky nebo neobsahují bezpečné hodnoty (například nastavení CORS – Cross Origin Request Site). (38)

A06:2021 Vulnerable and Outdated Components (zranitelné a zastaralé komponenty) – bezpečnostní riziko způsobené neaktualizovaným systémem nebo jeho součástí.

- Neznalost verzí všech komponent systému včetně tranzitivních znemožňuje určit jejich aktuálnost.
- Použité softwarové části systému jsou zranitelné, nepodporované nebo zastaralé. To zahrnuje OS, webový/aplikační server, systém správy databází (DBMS), aplikace, rozhraní API a všechny komponenty, runtime prostředí a knihovny.

- Správce systému pravidelně nevyhledává zjištěné zranitelnosti v rámci komunit nebo dodavatelů.
- Odložené instalace aktualizací a bezpečnostních záplat.
- Neotestovaná komptabilita aktualizovaných nebo opravených knihoven. (38)

A07:2021 Identification and Authentication Failures (chyby v autentizaci) – bezpečnostní chyby spojené s identifikací, autentifikací a autorizací uživatele v aplikaci.

- Umožnění automatizovaného přihlašování umožní útočníkovi hromadné zkoušení účtů a hesel.
- Povolení slabých a obecně známých hesel nebo zanechání aktivních výchozích účtů (admin/admin).
- Slabé nebo neúčinné ověření při obnovování zapomenutých hesel (například odpovědi založené na znalostech)
- Uložená hesla v surové podobě nebo použitá slabá hashovací funkce.
- Viditelný identifikátor přihlášené relace (session ID) přímo v URL.
- Používání stejných identifikátorů přihlášené relace (session ID) v rámci opakovaných přihlášení.
- Nedeaktivování všech relací v rámci odhlášení uživatele. (38)

A08:2021 Software and Data Integrity Failures (chyby integrity softwaru a dat) – Tato zranitelnost se týká kódu a infrastruktury, které se nechrání proti integritnímu narušení. Příkladem toho je situace, kdy aplikace spoléhá na pluginy, knihovny nebo moduly z nedůvěryhodných zdrojů, úložišť a sítí pro doručování obsahu (CDN). Nezabezpečený kanál CI/CD může představovat potenciál pro neoprávněný přístup, škodlivý kód nebo kompromitaci systému. V dnešní době mnoho aplikací obsahuje funkci automatických aktualizací, kdy se aktualizace stahují bez dostatečného ověření integrity a aplikují se na dříve důvěryhodnou aplikaci. Útočníci by potenciálně mohli nahrát své vlastní aktualizace, které budou distribuovány a spuštěny ve všech instalacích. Dalším příkladem je situace, kdy jsou objekty nebo data zakódována nebo serializována do struktury, kterou útočník vidí a kterou může upravit, je zranitelná vůči nezabezpečené deserializaci. (38)

A09:2021 Security Logging and Monitoring Failures (chyby bezpečnostního logování a monitorování) – bez dostatečného logování a monitorování nemohou být odhaleny bezpečnostní rizika a napadení systému.

- Chybí logování událostí jako jsou úspěšná a neúspěšná přihlášení nebo nestandardních operací.
- Varování a chyby neobsahují žádné nebo nedostatečné či nejasné informace.
- Aplikační logy a audit volání na API rozhraní nejsou sledovány pro podezřelou aktivitu.
- Logy se ukládají pouze lokálně.
- Nevhodné nastavení prahových hodnot pro automatizované varování v monitorování aplikace.
- Neaplikováno penetrační testování a skenování pomocí nástrojů dynamického testování zabezpečení aplikací (například OWASP ZAP).
- Monitorování nemůže detekovat, eskalovat nebo upozorňovat na aktivní útoky v reálném čase nebo téměř v reálném čase. (38)

A10:2021 Server Side Request Forgery (SSRF, podvržení požadavku na straně serveru) – jedná se o typ útoků, u kterých útočník odesílá podvržené požadavky ze serveru zranitelné webové aplikace. Cílem je přístup k interním systémům, které nejsou přístupné z vnější sítě. SSRF může mít vážné důsledky, včetně získání neoprávněného přístupu k interním síťovým prostředím nebo službám, jako jsou databáze nebo úložiště dat.

Příkladem útoku může být manipulace s adresou URL: Útočník odesílá HTTP žádost z cílového serveru, ale může ovlivnit adresu URL v takovém rozsahu, že server provede požadavek na něco jiného, než bylo zamýšleno.

Druhým příkladem může být zpracování obrázků: Některé služby mohou umožňovat uživatelům nahrávat obrázky a poté generovat náhledy. Pokud tyto služby nejsou správně ošetřeny (například chybí bezpečnostní skenování nahrávaných dat), může útočník využít SSRF k provedení požadavků na interní služby. (38)

4 Vlastní práce

Tato kapitola je již věnována konkrétnímu návrhu autentizačního a autorizačního mechanismu vhodného pro dnešní moderní podnikové systémy splňující veškeré bezpečnostní nařízení a doporučení. Součástí je analytický návrh řešení za pomoci standardních nástrojů včetně dodržení dokumentačních standardů k udržení čitelné formy pro většinu technických analytiků, jako vstup pro vlastní návrh či úpravu stávajícího systému. Následuje realizace demo aplikace, kde je v minimalistické formě s pomocí standardně používaných technologií vyvinuta autentizační a autorizační aplikace, zabezpečená aplikace poskytující data a aplikace, která data konzumuje. Tím je dokázána realizovatelnost a funkčnost celého návrhu. Zároveň řešení může sloužit jako vstup vývojářům, kteří se mohou inspirovat, jak danou problematiku s danou technologií řešit. V následující kapitole bude provedeno ověření a zhodnocení bezpečnosti návrhu a realizovaného MVP (Minimum Viable Product, neboli nejmenší udržitelný produkt) za pomocí běžných nástrojů a teoretické části této práce.

4.1 Technologie referenčního řešení

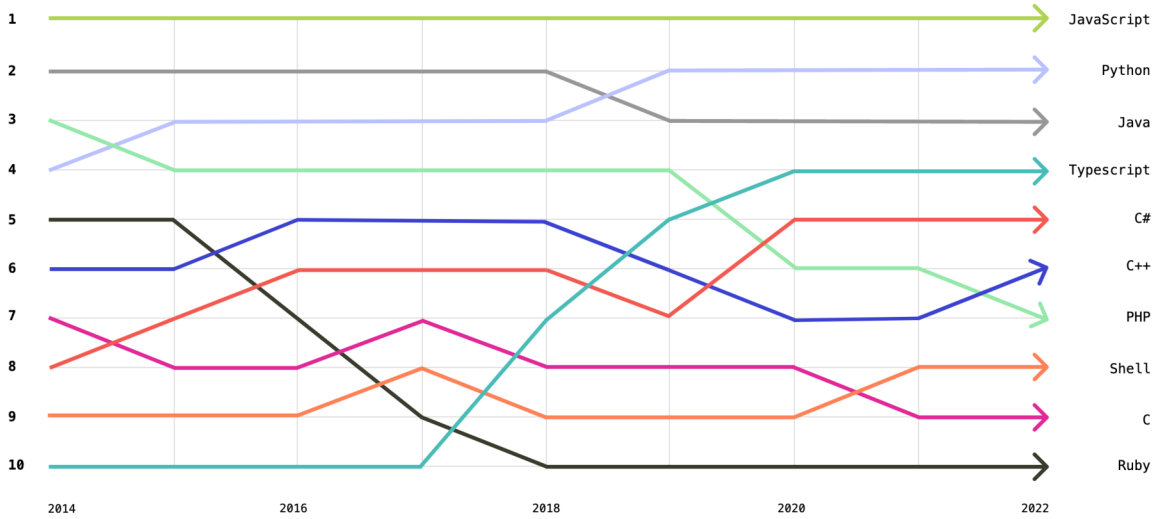
K vytvoření využívaného a bezpečného řešení autentizace a autorizace v dnešních moderních systémech je třeba zajistit vhodný výběr technologií. Při výběru technologií by měla být rozhodující podpora ze strany vývoje, bezpečnost dané technologie a možnosti integrace do vybraného systému dle jeho architektury. Důležitým vstupem při rozhodování jsou technické informace k jednotlivým technologiím v teoretické části této práce. Při nerozhodném výsledku nebo ujištění správného výběru je také možné využít otevřených zdrojů poskytujících statická data o využívanosti nebo oblíbenosti při vyhledávání technických termínů a slovních spojení. Jedním z takových zdrojů je Google Trends.

4.1.1 Programovací jazyk

Výsledné řešení má být ověřenou implementací demo aplikace u které je nutné zvolit vhodný programovací jazyk. Demo aplikace byla rozdělena do dvou částí (modulů). Jedná část frontendová (na straně uživatele) a druhá část backendová (na straně serveru). Pro obě části bylo nutné zvolit programovací jazyk nebo jazyky podle toho, kde bude kód vykonáván. Kód frontendového modulu se vykonává u klienta v prohlížeči, zatímco u backendového modulu se kód vykonává na serveru. Autor práce se rozhodoval na základě

svých zkušeností s využívanými programovacími jazyky a zvolil takové se kterými má největší teoretické i praktické zkušenosti. Pro frontendovou část byl použit Javascript s HTML a pro backendovou část Java. Správný výběr programovacích jazyků byl i částečně potvrzen statistikami jednoho z větších veřejných repozitářů kódu (GitHub) na obrázku 14, kde se ukázaly jako nejpoužívanější v rámci repozitářů, které si uživatelé na serveru vytvořili.

Obrázek 14 – Využití programovacích jazyků na Githubu

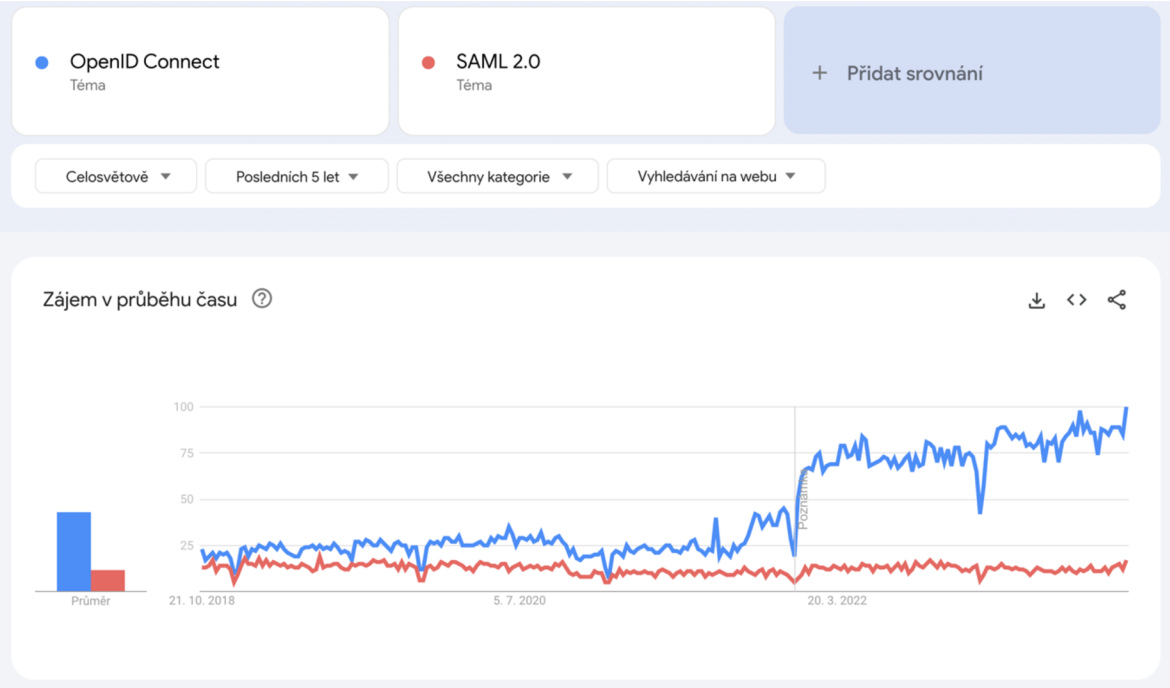


Zdroj: https://octoverse.github.com/2022/top-programming-languages (2023)

4.1.2 Autentizační protokol

V teoretické části práce je zmíněno několik autentizačních protokolů. V rámci MVP aplikace bylo záměrem zvolit aktuálně nejvíce používanou. Za tímto účelem byl proveden průzkum internetu. Předpokladem bylo že vývojáři a analytici často hledají informace k využívaným technologiím na internetu. Proto byla opět využita služba Google Trends. Celosvětové srovnání OpenID Connect protokolu a SAML 2.0 protokolu na obrázku 15 ukazuje, že OpenID Connect v posledních letech jasně převyšuje nad SAML 2.0. Důvod může být oblíbenost jednoduchého textového formátu JSON namísto staršího XML formátu. OIDC je také vhodnější pro dnešní typy SPA (Single Page Application) aplikací, které většinou komunikují s backendovou částí aplikace pomocí REST API rozhraní nebo dnešní nativní mobilní aplikace, které používají stejný typ komunikace.

Obrázek 15 – Srovnání vyhledávání termínů OIDC a SAML 2.0



Zdroj: https://trends.google.com/trends/explore?date=today%205-y&q=%2Fg%2F11bc6h8qw4,%2Fm%2F02q69hf&hl=cs (2023)

4.1.3 Autorizační protokol

Na základě výběru autentizačního protokolu OpenID Connect je vlastně rozhodnuto i pro autorizační protokol. OpenID Connect je autentizační nástavba nad autorizačním protokolem OAuth 2.0. Protokol umožňuje federativní autorizaci nad oddělenými systémy a je proto vhodný pro velké podnikové systémy. Poskytuje pouze přístupové tokeny, které mají omezenou platnost a rozsah oprávnění. Zároveň je navržen tak, že je rozšiřitelný což umožňuje přidávat další vlastnosti a rozšíření pro specifické potřeby a scénáře podnikových administrátorů. Je jednoduchý na implementaci pro vývojáře a umožňuje používat tokeny ve formátu JSON Web Tokens (JWT), které usnadňují práci s autentizačními a autorizačními tokeny a také jsou rozšiřitelné o další vlastní atributy dle potřeb jednotlivých podnikových systémů. Bezpečnost a důvěryhodnost JWT tokenů je zajištěna elektronickým podpisem nebo šifrováním pro skrývání citlivých informací v obsahu tokenu. Podpora OAuth 2.0 protokolu v rámci jednotlivých programovacích jazyků je velice široká díky velkému množství knihoven.

4.1.4 Knihovny a protokoly

Na základě předchozích výběrů, provedených průzkumů, teoretické části této práce a praktických zkušenosti autora této práce v oblasti vývoje podnikových aplikací byla připravena přehledová tabulka 3 zvolených technologií (knihoven).

Tabulka 3 – Přehled referenčních technologií

Oblast / funkce systému	Technologie / knihovna
Autentizační a autorizační protokol	OpenID Connect (OIDC)
API rozhraní	RESTful API
Programovací jazyky	Java, Javascript, HTML, CSS
Frameworky a knihovny na frontendu	JOSE (digitální podpis a šifrování JWT) jQuery (práce s DOM a REST volání) Bootstrap (grafický framework)
Frameworky a knihovny na backendu	Spring MVC (implementace REST API) Spring Security (zabezpečení API) Spring Data (ORM framework) Spring boot (aplikační server) Auth0 (JWT, JKS) Argon 2 (hashování dat), Bouncycastle (šifrování) Samstevens (TOTP) PostgreSQL (relační databáze)

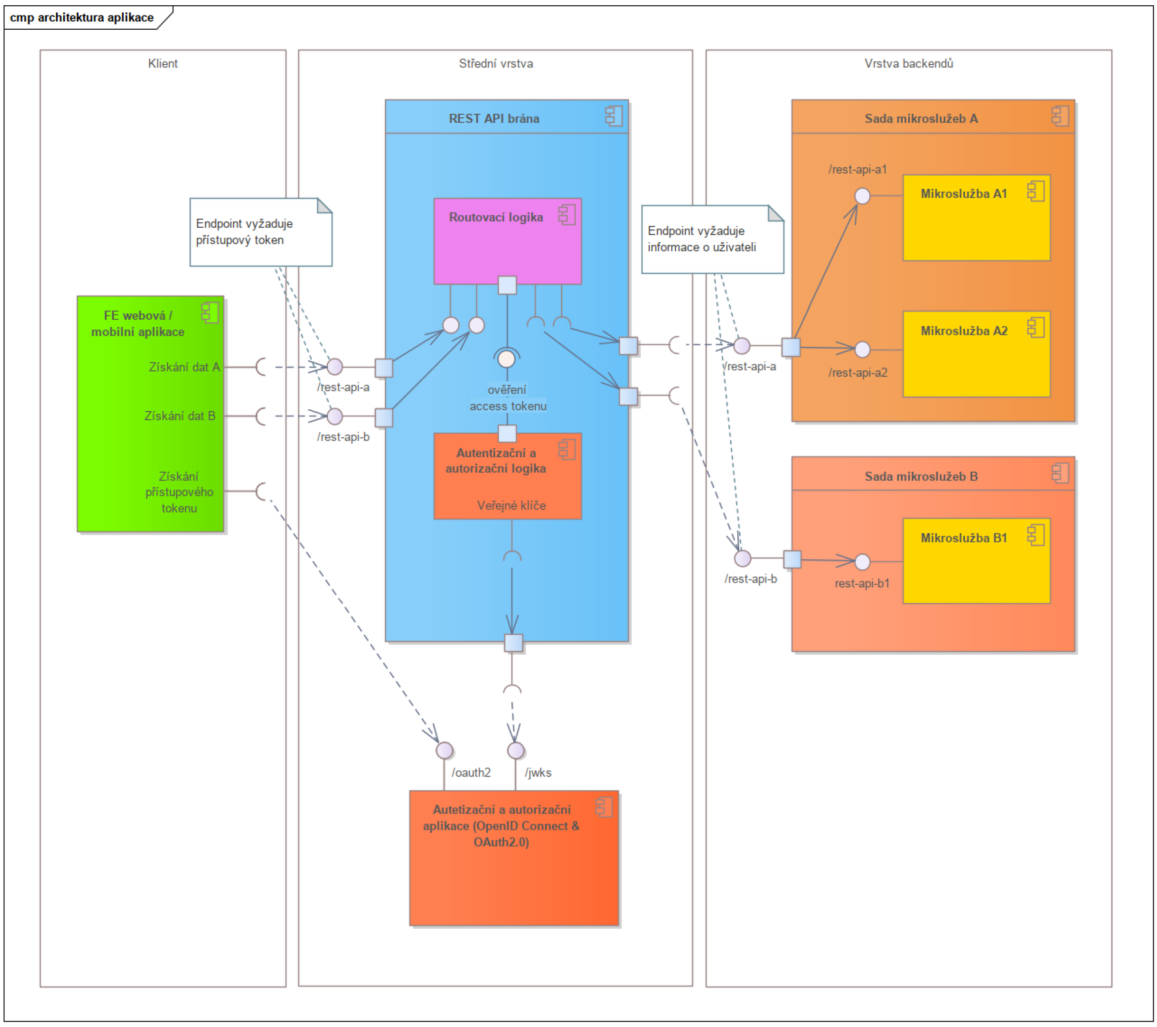
Zdroj: Vlastní zpracování (2023)

4.2 Návrh referenčního řešení

4.2.1 Referenční architektura systému

V rámci návrhu celého řešení bylo vycházeno z aktuálních trendů v oblasti architektur velkých korporátních systémů, kde se vyžaduje flexibilita, modularita a vysoká úroveň zabezpečení. Pro lepší představu je na obrázku znázorněna v minimalistickém provedení architektura orientovaná na mikroslužby s REST API rozhraním, která aktuálním trendům odpovídá. Jedná se o architekturu, kterou lze rozdělit do několika vrstev. Každá vrstva má svůj funkční význam a úroveň zabezpečení. Pro jednodušší představu uvažuji tři hlavní vrstvy (klientská vrstva, střední vrstva a vrstva backendů), ale reálně jich může být i více. Jednalo by se o podmnožiny uvažovaných vrstev nebo opakování stejných vrstev.

Obrázek 16 – Komponentní digram zjednodušené architektury systému



Zdroj: Vlastní zpracování (2023)

Nejnižší vrstvou je vrstva backendů, která je běhové prostředí pro jednotlivé moduly vystavující přes REST API rozhraní funkce a data pro klienty či jiné moduly v rámci mezimodulové komunikace. Tyto moduly dle specifik architektury orientované na mikroslužby musí být malé a pokrývat určitou business část aplikace. Zabezpečení na této úrovni je minimální a víceméně je řešeno samotným rozhraním modulu, které je dané business specifikací. Například pokud rozhraní vyžaduje identifikaci uživatele, který službu využívá, tak musí být vždy předáno (v rámci REST API většinou hlavičkou) jinak je požadavek ve validační logice zamítnut. Nízkou úroveň zabezpečení si můžeme dovolit díky tomu, že vrstva je síťově oddělena od vnějšího světa a přístup k ní mají pouze komponenty, které již zabezpečeny jsou nebo samotné zabezpečení přímo implementují.

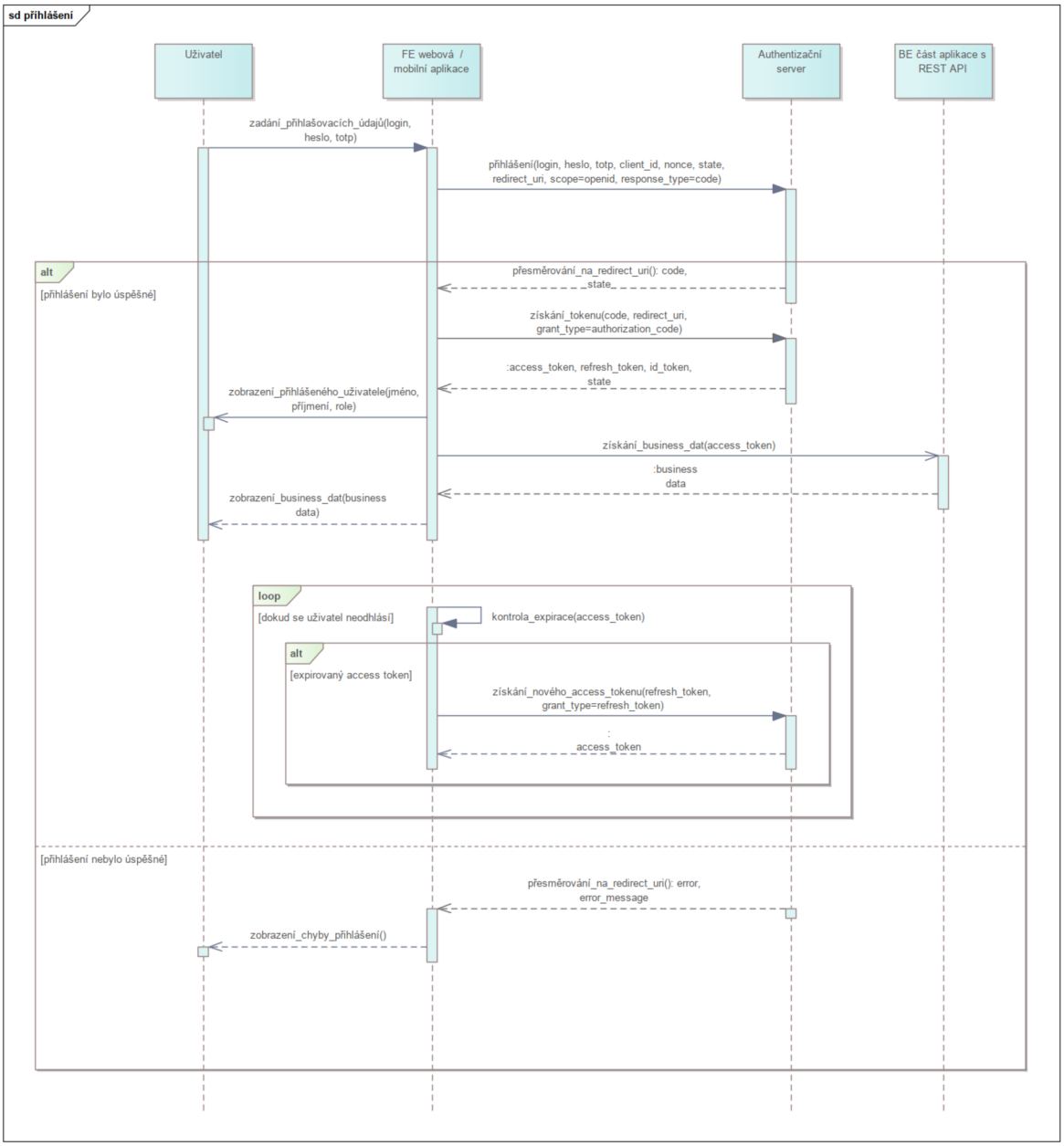
Na vrstvu backendů přímo navazuje střední vrstva, která již není běhové prostředí pro backendové moduly, nýbrž pro provozní komponenty. Mezi takové provozní komponenty spadá například REST/SOAP API brána, ESB (Enterprise Service Bus), poskytovatel identit, autorizační a autentizační aplikace, logovací a monitorovací aplikace. Na obrázku 16 je znázorněno zapojení REST API brány spolu s autentizační a autorizační aplikace, neboť spolu úzce spolupracují. Hlavní funkcí brány je vystavení jednotlivých API mikroslužeb na vrstvě backendů pro klientskou vrstvu. Vystavené API může zahrnovat složitější routovací logiku (například přes více API na backendu), logovací a monitorovací nástroje a v neposlední řadě také zabezpečovací mechanismy. V tomto případě klient vkládá do autorizační hlavičky svůj přístupový token, který REST API brána dekóduje, ověřuje elektronický podpis a předává definované informace o uživateli z tokenu do požadavku na backendové API. Z této vrstvy jsou zabezpečené i nezabezpečené API síťově dostupné již pro klienty buď v rámci intranetu nebo internetu.

Poslední naznačenou vrstvou je klientská. Tato vrstva obsahuje aplikace ve formě tenkého klienta. Tenký klient znamená, že aplikace neobsahuje datovou vrstvu a vyžaduje ke svému běhu další služby. Může se jednat o mobilní aplikace či webové aplikace. Primárním účelem tenkého klienta je realizovat vizualizační a komunikační článek mezi uživatelem a vnitřním systémy v rámci autentizačních procesů i business části aplikace.

4.2.2 Autentizační a autorizační logika

Na základě navržené referenční architektury systému a předepsaného fungování OpenID Connect protokolu je připravený proces přihlášení uživatele a následného získání dat popisující sekvenční diagram na obrázku 17.

Obrázek 17 – Sekvenční diagram přihlášení a načtení dat



Zdroj: Vlastní zpracování (2023)

4.2.3 Registrační logika

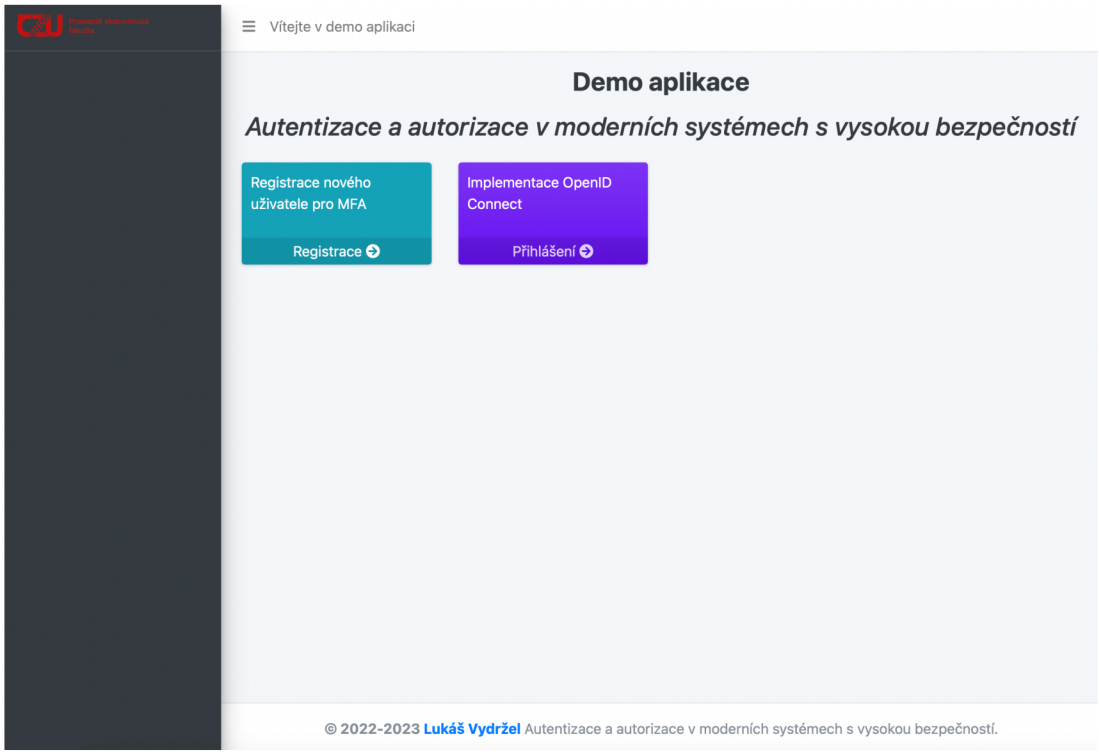
V oblasti registrace uživatele není důležitý samotný proces, jak je uživatel do systému zaregistrován. Může být registrován správcem systému v rámci organizačního Active Directory nebo například samotným uživatelem a uložením do relační. Možností způsobů zavedení uživatele do systému je nepřeberné množství. Z pohledu bezpečnosti jsou důležité následující pravidla, které jsou zároveň referenčním řešením:

- Pravidla pro způsob ověření uživatele
    - Vyžadováno unikátní přihlašovací jméno
    - Vyžadováno heslo
    - Vyžadována vícefaktorová autentizace (TOTP, HOTP, biometrie, …)
- Pravidla pro heslo
    - Minimální délka 12 znaků
    - Kombinace malých a velkých znaků, číslic a speciálních znaků
    - Zamezení slovníkových hesel
    - Omezená platnost hesla
    - Omezení opakování hesel a číselných sekvencí
    - Uložení do databáze v hashované podobě za pomocí soli (5) (40)

4.3 Demo aplikace

Backendová část demo aplikace je napsaná v jazyce Java s využitím Spring frameworku. Frontendová část pomocí jazyků Javascript, HTML a CSS s využitím knihoven jQuery, JOSE a Bootstrap. Zdrojové kódy jsou v příloze A.

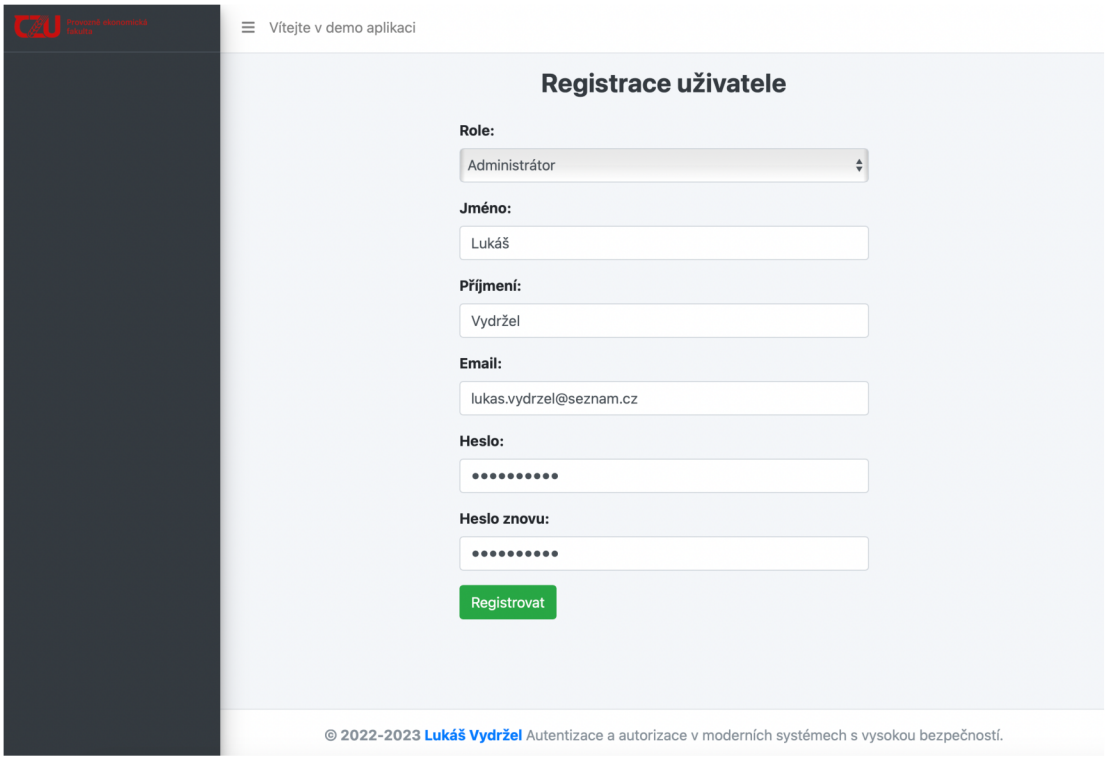
Obrázek 18 – Domovská stránka demo aplikace



Zdroj: Vlastní zpracování (2023)

Grafické rozhraní na domovské stránce na obrázku 18 obsahuje hlavní menu formou prokliknutelných dlaždic. Přes toto menu je možné přejít na proces registrace uživatele nebo na proces přihlášení uživatele. V procesu přihlášení uživatele je také načtení zabezpečených informací o uživateli s využitím OpenID Connect protokolu.

Obrázek 19 – Registrační formulář demo aplikace



Zdroj: Vlastní zpracování (2023)

Registrace uživatele začíná registračním formulářem na obrázku 19. Pro potřeby identifikace uživatele je třeba vyplnit jméno a příjmení. Pro potřeby autentizace je třeba vyplnit e-mail sloužící jako přihlašovací jméno a heslo s opakováním pro zamezení překlepů. A pro potřeby autorizace se volí role uživatele (uživatel nebo administrátor). Registrační formulář obsahuje validace k udržení konzistence dat, formátu dat a bezpečnostních doporučení.

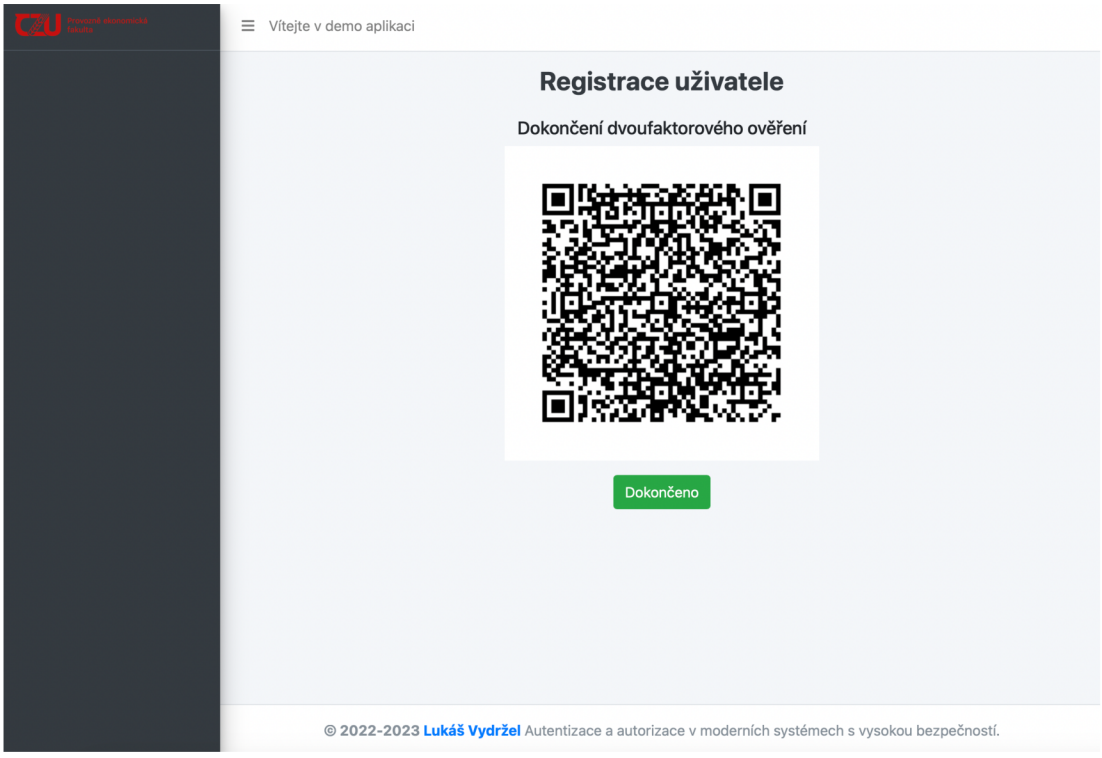
Tlačítko registrovat vyvolá následující asynchronní HTTP požadavek na demo backend (zápis pomocí curl):

```
curl 'http://localhost:8080/api/idp/register' \
  -H 'Content-Type: application/json;charset=UTF-8' \
  -H 'Origin: http://localhost:8080' \
```

```
  --data-raw
'{"role":"ADMIN","firstname":"Lukáš","lastname":"Vydržel","login":"lukas.
vydrzel@seznam.cz","password":"***********"}'
```

Backendová část demo aplikace představující IDP (Identity Provider) provede validaci požadavku podobně jako frontend, ale navíc i vyhodnotí případnou duplicitu na přihlašovacím jméně. Následně se vypočítá hash z hesla a unikátního řetězce (soli) za pomocí Argon 2 algoritmu. Všechny data se následně ukládají do Postgres databáze. Následně probíhá příprava podpory vícefaktorové autentizace pomocí TOTP (Time-based One Time Password). Vygeneruje se tajemství (secret) pro TOTP, které se následně ukládá do databáze. Za pomocí knihovny Samstevens se vygeneruje QR kód pro snadné přidání registračních údajů do autentizačních aplikací implementující podporu TOTP protokolu (například TOTP Authenticator od společnosti Binaryboot). Výsledný QR kód je zakódován do formátu Base64 a odeslán v odpovědi na registrační požadavek.

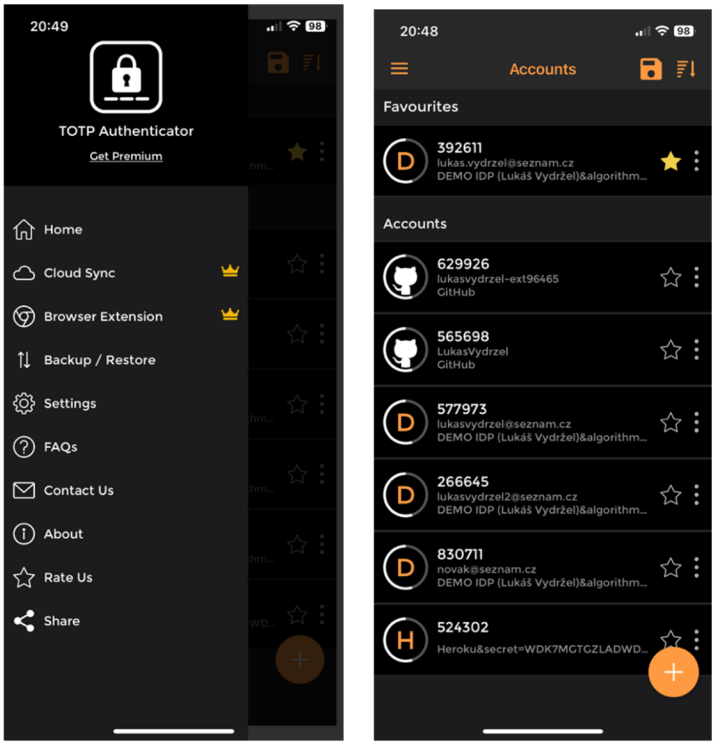
Obrázek 20 – Obrazovka s QR kódem pro registraci TOTP v demo aplikaci



Zdroj: Vlastní zpracování (2023)

Na obrázku 20 je obrazovka s výše uvedeným vyobrazeným QR kódem pro načtení autentizačních informací do zvolené vícefaktorové aplikace.

Obrázek 21 – Ukázka mobilní aplikace TOTP Autheticator



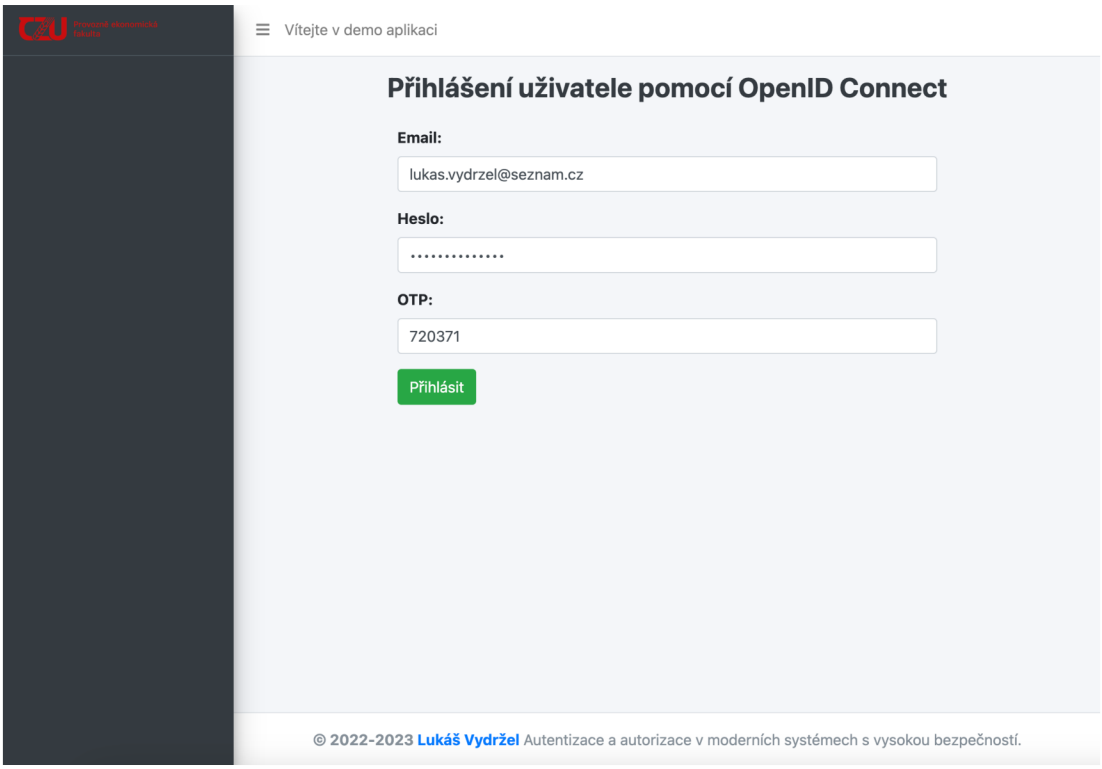
Zdroj: Vlastní zpracování (2023)

Na obrázku 21 je ukázka zaregistrovaného výše uvedeného QR kódu v aplikaci TOTP Authenticator od společnosti Binaryboot, která umožňuje následné generování časově tvořeného jednorázového hesla. Aplikace umožňuje uložit i více přihlašovacích účtů včetně jednotlivých aplikací. Vždy po zapnutí aplikace je vyobrazeno časově omezené heslo, které je třeba zaslat v požadavku na přihlášení spolu s přihlašovacím jménem a heslem. Někdy se uživatelé mohou setkat s řešením, že zaslání tohoto jednorázového hesla je požadováno až v dalším kroku. Nicméně vzhledem k tomu, že v demo aplikaci je jednorázové heslo povinné a je to jediná vícefaktorová metoda, tak je vyžadováno hned v prvním kroku. Díky tomu je zamezeno robotickému zkoušení hesel.

Přihlašovací požadavek ve formátu curl vypadá následovně:

```
curl 'http://localhost:8080/api/idp/oidc/login' \
  -H 'Content-Type: application/json;charset=UTF-8' \
  -H 'Origin: http://localhost:8080' \
  --data-raw
'{"login":"lukas.vydrzel@seznam.cz","password":"************","totp":"530
745"}'
```

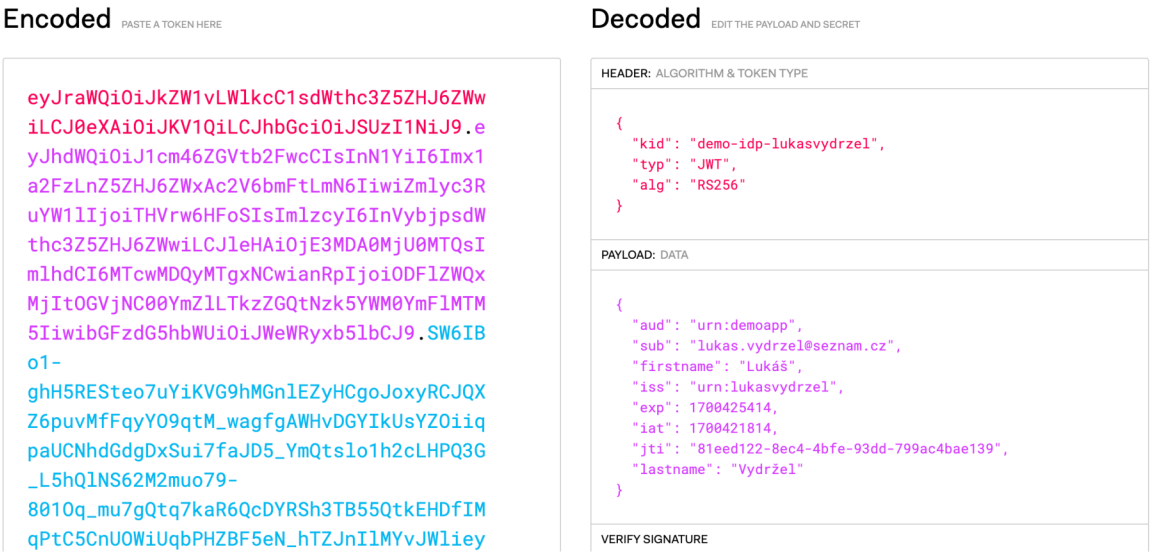
Obrázek 22 – Přihlašovací formulář demo aplikace



Zdroj: Vlastní zpracování (2023)

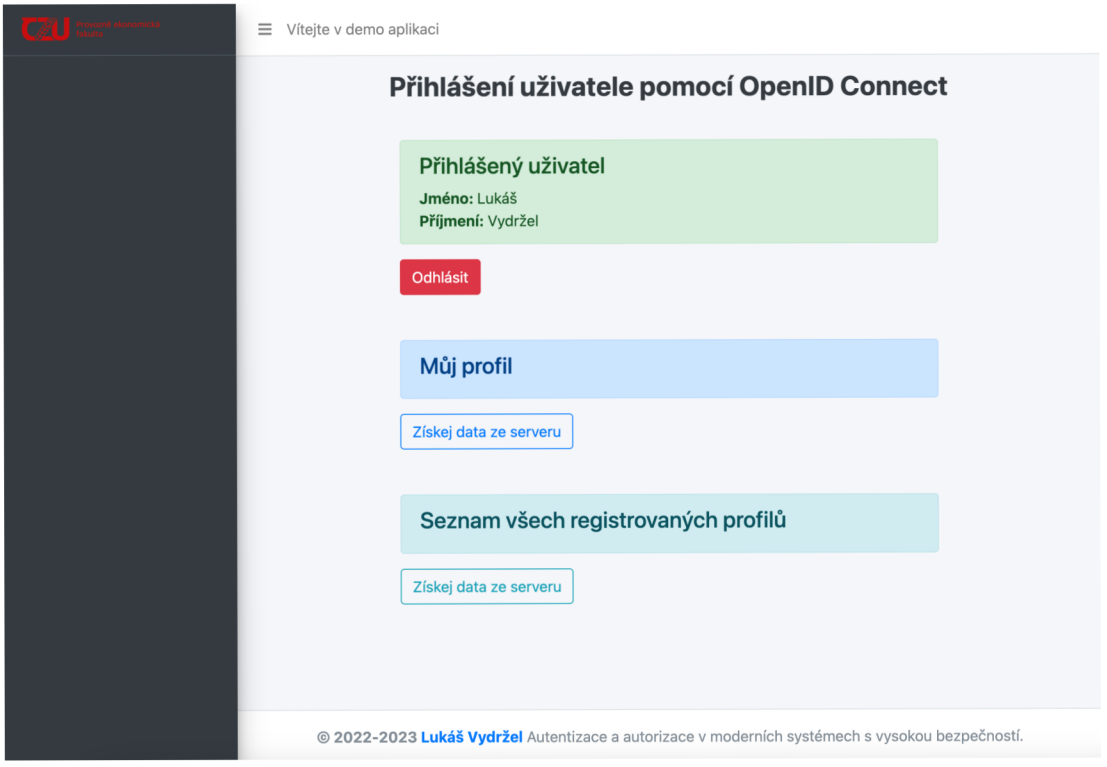
Na obrázku 22 je k přihlašovacímu požadavku vyobrazena obrazovka s přihlašovacím formulářem. V odpovědi na úspěšné přihlášení vrátí IDP přístupový token (access token) a identity token (ID token). Identity token na obrázku 23 je určen primárně k identifikaci uživatele a demo aplikace z něj zobrazí jméno a příjmení přihlášeného uživatele.

Obrázek 23 – Dekódovaný identity token (JWT) z demo aplikace



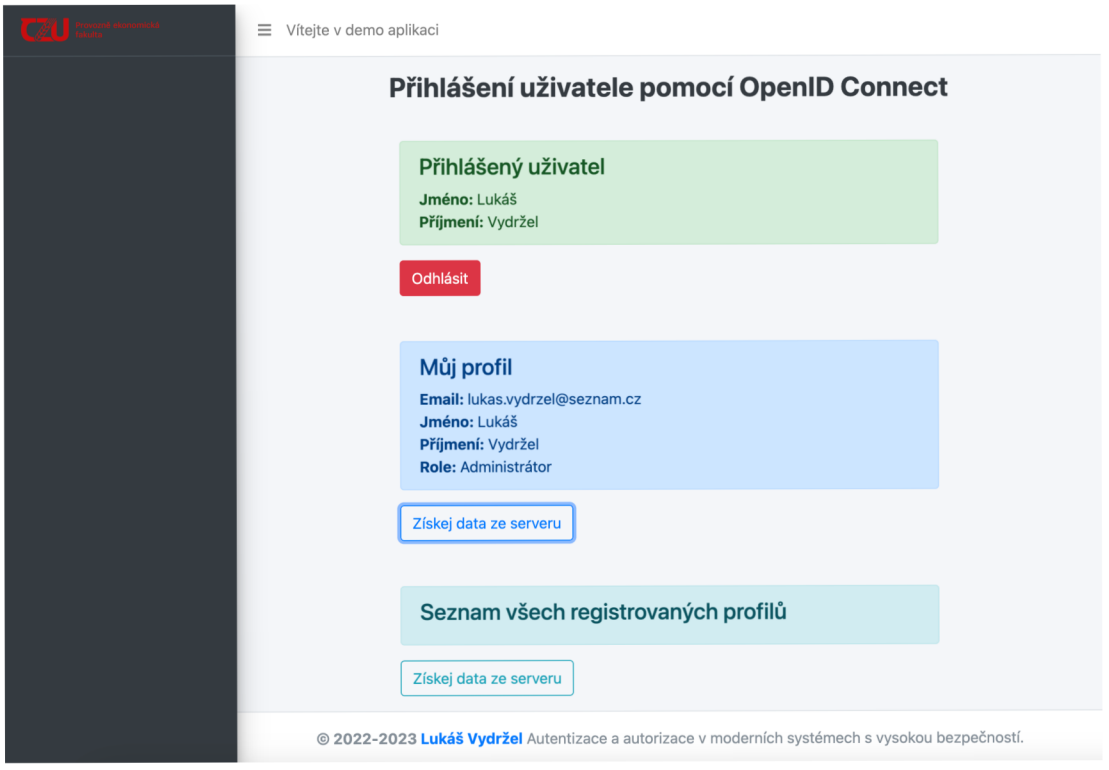
Zdroj: Vlastní zpracování (2023)

Obrázek 24 – Výchozí stránka přihlášeného uživatele



Zdroj: Vlastní zpracování (2023)

Obrázek 25 – Stránka s načteným profilem přihlášeného uživatele



Zdroj: Vlastní zpracování (2023)

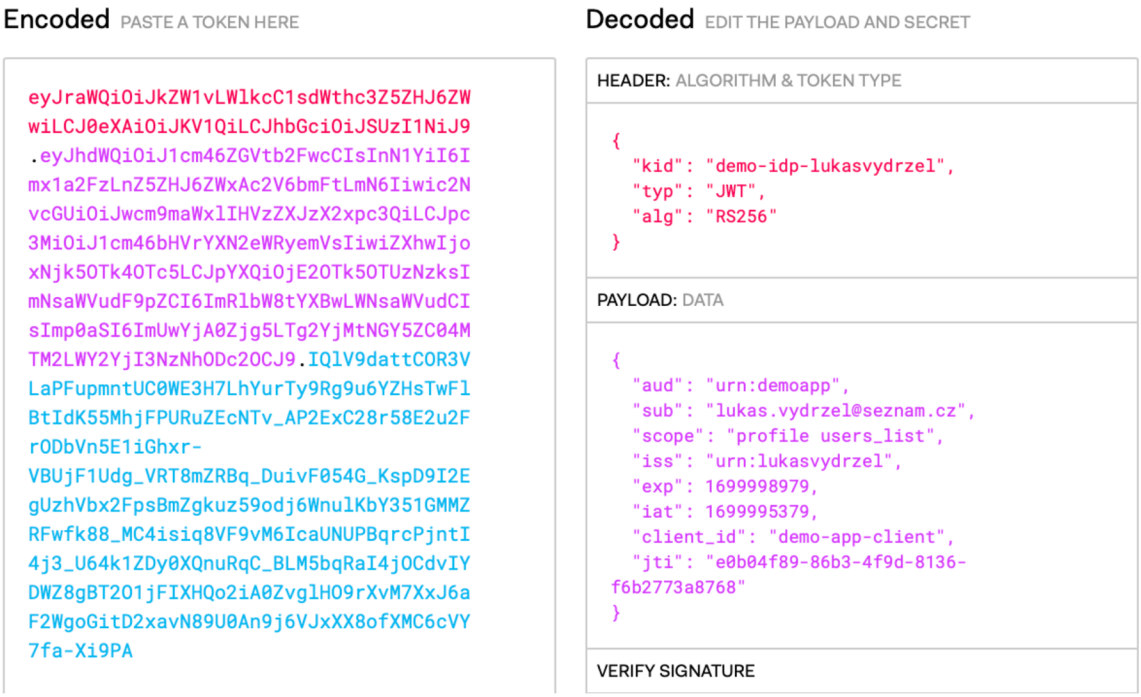
Na obrázku 24 je vyobrazena výchozí stránka po úspěšném přihlášení uživatele. Jsou zde vypsány identifikační údaje uživatele z identity tokenu (ID token) a možnost odhlášení. V rámci odpovědi od IDP frontendová část demo aplikace také získala přístupový token (access token), který je nyní možné využít na volání datových API. Demo aplikace umožňuje zavolat REST API pro získání profilu uživatele. Toto API je zabezpečeno způsobem, že je jej možné zavolat pouze s platným přístupovým tokenem. Získaná data se vždy vztahují k přihlášenému uživateli a není třeba další zabezpečení na úrovní rolí. Načtená data jsou vyobrazena na obrázku 25.

Požadavek ve formátu curl na získání profilu přihlášeného uživatele vypadá následovně:

```
curl 'http://localhost:8080/api/demo-app/my-profile' \
  -H 'Authorization: Bearer
eyJraWQiOiJkZW1vLWlkcC1sdWthc3Z5ZHJ6ZWwiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1N
iJ9.eyJhdWQiOiJ1cm46ZGVtb2FwcCIsInN1YiI6Imx1a2FzLnZ5ZHJ6ZWxAc2V6bmFtLmN6I
iwic2NvcGUiOiJwcm9maWxlIHVzZXJzX2xpc3QiLCJpc3MiOiJ1cm46bHVrYXN2eWRyemVsIi
wiZXhwIjoxNjk5OTk4OTc5LCJpYXQiOjE2OTk5OTUzNzksImNsaWVudF9pZCI6ImRlbW8tYXB
wLWNsaWVudCIsImp0aSI6ImUwYjA0Zjg5LTg2YjMtNGY5ZC04MTM2LWY2YjI3NzNhODc2OCJ9
.IQlV9dattCOR3VLaPFupmntUC0WE3H7LhYurTy9Rg9u6YZHsTwFlBtIdK55MhjFPURuZEcNT
v_AP2ExC28r58E2u2FrODbVn5E1iGhxr-
VBUjF1Udg_VRT8mZRBq_DuivF054G_KspD9I2EgUzhVbx2FpsBmZgkuz59odj6WnulKbY351G
MMZRFwfk88_MC4isiq8VF9vM6IcaUNUPBqrcPjntI4j3_U64k1ZDy0XQnuRqC_BLM5bqRaI4j
OCdvIYDWZ8gBT2O1jFIXHQo2iA0ZvglHO9rXvM7XxJ6aF2WgoGitD2xavN89U0An9j6VJxXX8
ofXMC6cVY7fa-Xi9PA'
```

Další možností v rámci frontendové části demo aplikace je možné zavolat REST API pro získání seznamu všech registrovaných uživatelů. Toto API je také zabezpečeno způsobem, že je jej možné zavolat pouze s platným přístupovým tokenem. Ale na rozdíl od profilu uživatele je zde implementovaná rozdílná autorizace uživatele a je kontrolována přidělená role přihlášeného uživatele. Pouze administrátor může získat seznam všech registrovaných uživatelů. Role uživatele není přímo obsažena v přístupovém tokenu, ale namísto ní se používá JWT claim scope (scp). Scope může být například seznam oprávnění nebo povolených možností uživatele (s vydaným přístupovým tokenem) oddělených mezerou.

Obrázek 26 – Dekódovaný přístupový JWT z demo aplikace



Zdroj: Vlastní zpracování (2023)

V dekódovaném přístupovém tokenu (JWT) na obrázku 26 jsou vidět dvě oprávnění. První oprávnění (scope) je **profile**, který získávají všichni uživatelé v demo aplikaci, aby mohli zavolat REST API poskytující profil přihlášeného uživatele. Druhým oprávněním je **users_list**, který získávají všichni uživatelé s rolí administrátora. Tento scope je nutný pro úspěšně zavolání REST API pro získání profilů všech registrovaných uživatelů. Zrušení přímé vazby rolí na autorizační proces v rámci API zabezpečení umožní budoucí variabilitu a nezávislost, jak při úpravách rolí uživatelů, tak i při úpravě seznamů oprávněních na API (scopes).

Důležité je také zmínit, že tento přístupový token je vždy v rámci zabezpečení na všech REST API demo aplikace validován, že není vypršený a je platně podepsán certifikátem důvěryhodného a povoleného vydavatele tokenů (demo IDP).

5 Zhodnocení výsledků

5.1 Funkční testy demo aplikace

Funkční testování probíhalo manuálně v rámci předem naplánovaných testovacích scénářích definujících vstupní podmínky, jednotlivé kroky testu a očekávaný výsledek (tabulky 4-9). Cílem testu bylo ověřit funkčnost naimplementovaného referenčního procesu identifikace, autentizace a autorizace v rámci demo aplikace.

Tabulka 4 – Detaily testovacího scénáře TS1

Atribut scénáře	**Hodnota**
Název	Registrace uživatele s podporou dvoufaktorové autentizace
Vstupní podmínky	Nový uživatel, nainstalovaná aplikace pro TOTP.
Testovací kroky	Zobrazení registračního formuláře. Vyplnění jména, příjmení, emailu, role, hesla včetně opakování. Zobrazení QR kódu pro registraci TOTP. Registrace TOTP ve vlastní aplikaci. Zobrazení potvrzení úspěšné registrace uživatele.
Očekávaný výsledek	Uživatel úspěšně uložen do databáze.
Stav testu	OK

Zdroj: Vlastní zpracování (2023)

Tabulka 5 – Detaily testovacího scénáře TS2

Atribut scénáře	**Hodnota**
Název	Přihlášení registrovaného uživatele a načtení dat
Vstupní podmínky	Uživatel je registrovaný včetně TOTP.
Testovací kroky	Zobrazení přihlašovací stránky. Vyplnění emailu, hesla a TOTP. Odeslání přihlašovacího formuláře.
Očekávaný výsledek	Zobrazení stránky s daty o uživateli.
Stav testu	OK

Zdroj: Vlastní zpracování (2023)

Tabulka 6 – Detaily testovacího scénáře TS3

Atribut scénáře	Hodnota
Název	Validace registračního formuláře
Vstupní podmínky	-
Testovací kroky	Kontrola povinných polí. Kontrola veškerých omezení a doporučení v rámci volby hesla. Zamezení duplicitních účtů.
Očekávaný výsledek	Každý krok dostává smysluplnou chybovou hlášku a nedovolí v procesu pokračovat.
Stav testu	OK

Zdroj: Vlastní zpracování (2023)

Tabulka 7 – Detaily testovacího scénáře TS4

Atribut scénáře	Hodnota
Název	Validace přihlašovacího formuláře
Vstupní podmínky	-
Testovací kroky	Kontrola povinných polí. Kontrola zadání nesprávného hesla. Kontrola zadání nesprávného TOTP.
Očekávaný výsledek	Každý krok dostává smysluplnou chybovou hlášku a nedovolí v procesu pokračovat.
Stav testu	OK

Zdroj: Vlastní zpracování (2023)

Tabulka 8 – Detaily testovacího scénáře TS5

Atribut scénáře	Hodnota
Název	Uložení dat o uživateli do databáze
Vstupní podmínky	Kontrolovaný uživatel je úspěšně registrovaný.
Testovací kroky	Kontrola řádku v databázi s daty vybraného uživatele.
Očekávaný výsledek	Všechna povinná data jsou o uživateli vyplněna a heslo je v hashované formě.
Stav testu	OK

Zdroj: Vlastní zpracování (2023)

Tabulka 9 – Detaily testovacího scénáře TS6

Atribut scénáře	Hodnota
Název	Validace datového REST API
Vstupní podmínky	-
Testovací kroky	Volání REST API bez přístupového tokenu. Volání REST API se zastaralým přístupovým tokenem. Volání REST API s validním přístupovým tokenem cizí aplikace. Volání REST API endpointu omezeného rolí s validním přístupovým tokenem, ale bez potřebné role. Volání REST API s podvržených přístupovým tokenem.
Očekávaný výsledek	Každé volání REST API dostává smysluplný chybový kód a neposkytne v odpovědi data.
Stav testu	OK

Zdroj: Vlastní zpracování (2023)

5.2 Bezpečnostní testy demo aplikace (referenčního řešení)

Demo aplikace byla vytvořena za účelem demonstrace procesů, protokolů a ošetření určitých bezpečnostních rizik, a proto není nasazena a provozována na konkrétním serveru. Jedná se o omezení pro použití automatizovaných nástrojů penetračního testování. Proto pro ověření bezpečnosti připraveného návrhu a implementované demo aplikace bylo zapotřebí manuální metody. Autor práce se rozhodl využít seznam 10 nejrizikovějších bezpečnostních zranitelností ve webových aplikacích (**OWASP Top 10 Vulnerabilities**). Pro každou zranitelnost bylo ověřeno, zdali je pro výstupy této práce relevantní a zda má proti této zranitelnosti dostatečnou ochranu či ji úplně eliminuje (tabulky 10-19). (38)

Tabulka 10 – Ověření A01:2021 Broken Access Control

Zdroj zranitelnosti	Stav	Komentář
Porušení zásady minimálního oprávnění nebo odepření přístupu ve výchozím nastavení služby. Namísto toho jsou služby dostupné všem ve výchozím stavu.	OK	Výchozí zabezpečení u všech API služeb.

Možnost obejít bezpečnostní ověření úpravou parametrů, vnitřního stavu aplikace, HTML stránky nebo úpravou požadavků odesílaných API rozhraní.	OK	Přístupový token je digitálně podepsaný.
Možnost manuální změny identifikace uživatele k získání cizích oprávnění.	OK	ID token je digitálně podepsaný.
Na API rozhraní ke čtení dat chybí bezpečnostní ověření pro zbylé HTTP metody POST, PUT, PATCH a DELETE.	OK	Výchozí zabezpečení u všech API služeb a jejich HTTP metod.
Možnost manuální změny role uživatele k získání vyššího oprávnění.	OK	Přístupový token je digitálně podepsaný.
Možnost manipulace s metadaty aplikace (JWT, Cookies, Storage) k získání vyššího oprávnění nebo změny identity.	OK	U přístupového a ID tokenu je kontrolován digitální podpis.
Nenastavené CORS umožňuje přístup k API z cizích zdrojů.	OK	Nastavení CORS omezeno.
Možnost přímého přístupu neautorizovanému uživateli na chráněné stránky, pokud chráníme stránky pouze skrytím odkazů.	OK	Chráněno na úrovni API služeb.

Zdroj: Vlastní zpracování (2023)

Tabulka 11 – Ověření A02:2021 Cryptographic Failures

Zdroj zranitelnosti	**Stav**	**Komentář**
Data jsou přenášena v surové podobě (HTTP, SMTP, FTP).	N/A	Nastavení serveru.
Použití zastaralých nebo slabých šifrovacích algoritmů.	OK	RSA512
Použití zastaralých nebo slabých hashovacích metod.	OK	Použit Argon 2.
Chybí vynucení HTTPS přenosu a existuje podpora HTTP.	N/A	Nastavení serveru.
Používají se výchozí šifrovací klíče nebo nechybí pravidelná obnova či rotace.	N/A	Nastavení serveru.
Detailní chybové zprávy jsou zneužitelné.	OK	Pouze chybový kód.

Inicializační vektory pro šifrovací algoritmy jsou vynechány.	N/A	Nevyužito.
Nevhodné algoritmy pro generování náhodných řetězců používaných v šifrovacích a hashovacích funkcích.	OK	Použit Argon 2.

Zdroj: Vlastní zpracování (2023)

Tabulka 12 – Ověření A03:2021 Injection

Zdroj zranitelnosti	Stav	Komentář
Uživatelská vstupní data nejsou validována, filtrována a čištěna.	OK	Validace a čištění na FE i API službách.
Dynamické dotazy nebo neparametrizované volání nejsou escapovány dle kontextu volání a jsou použity přímo.	N/A	Použit ORM framework.
Vstupní data jsou přímo použita nebo zřetězeny v rámci SQL příkazů.	N/A	Použit ORM framework.
Vstupní data jsou přímo použita v rámci vyhledávacích parametrů v objektově relačním mapování (ORM).	OK	Spring Data escapuje vstupní data.

Zdroj: Vlastní zpracování (2023)

Tabulka 13 – Ověření A04:2021 Insecure Design

Zdroj zranitelnosti	Stav	Komentář
V návrhu, vývoji a testovaní aplikace není brán zřetel na bezpečnost.	OK	Orientováno na bezpečnost.

Zdroj: Vlastní zpracování (2023)

Tabulka 14 – Ověření A05:2021 Security Misconfiguration

Zdroj zranitelnosti	Stav	Komentář
Nedostatečná revize a omezování konfigurace cloudových služeb a povolujeme i nepotřebné funkce pro běh aplikace.	N/A	Není provozováno v cloudu.

Na serveru jsou nainstalovány nepotřebné služby, povolené nepotřebné porty, neomezený přístup k internetu, nepoužívané účty či oprávnění.	N/A	Není provozováno na serveru.
Nastavení detailního výpisu chyb odhaluje detaily aplikace.	OK	Pouze chybové kódy.
Konfigurace zabezpečení aplikačního serveru, aplikačního frameworku (Struts, Spring, ASP.NET), knihoven nebo databázích není na bezpečných hodnotách.	OK	Konfigurace orientovaná na bezpečnost.
Server neposílá bezpečnostní hlavičky nebo neobsahují bezpečné hodnoty (například nastavení CORS – Cross Origin Request Site).	OK	Použití CORS a autorizační hlavičky.

Zdroj: Vlastní zpracování (2023)

Tabulka 15 – Ověření A06:2021 Vulnerable and Outdated Components

Zdroj zranitelnosti	**Stav**	**Komentář**
Neznalost verzí všech komponent systému včetně tranzitivních znemožňuje určit jejich aktuálnost.	OK	Verze známé.
Použité softwarové části systému jsou zranitelné, nepodporované nebo zastaralé. To zahrnuje OS, webový/aplikační server, systém správy databází (DBMS), aplikace, rozhraní API a všechny komponenty, runtime prostředí a knihovny.	OK	Použity aktuální verze bez známých kritických zranitelností.
Správce systému pravidelně nevyhledává zjištěné zranitelnosti v rámci komunit nebo dodavatelů.	N/A	Není provozováno.
Odložené instalace aktualizací a bezpečnostních záplat.	N/A	Není provozováno.
Neotestovaná komptabilita aktualizovaných nebo opravených knihoven.	OK	Testováno.

Zdroj: Vlastní zpracování (2023)

Tabulka 16 – Ověření A07:2021 Identification and Authentication Failures

Zdroj zranitelnosti	Stav	Komentář
Umožnění automatizovaného přihlašování umožní útočníkovi hromadné zkoušení účtů a hesel.	OK	Dvoufaktorové ověření.
Povolení slabých a obecně známých hesel nebo zanechání aktivních výchozích účtů (admin/admin).	OK	Definována pravidla pro hesla.
Slabé nebo neúčinné ověření při obnovování zapomenutých hesel (například odpovědi založené na znalostech).	N/A	Není podpora této funkce.
Uložená hesla v surové podobě nebo použitá slabá hashovací funkce.	OK	Použit Argon 2.
Viditelný identifikátor přihlášené relace (session ID) přímo v URL.	N/A	Bezstavové přihlášení.
Používání stejných identifikátorů přihlášené relace (session ID) v rámci opakovaných přihlášení.	N/A	Bezstavové přihlášení.
Nedeaktivování všech relací v rámci odhlášení uživatele.	N/A	Bezstavové přihlášení.

Zdroj: Vlastní zpracování (2023)

Tabulka 17 – Ověření A08:2021 Software and Data Integrity Failures

Zdroj zranitelnosti	Stav	Komentář
Možnost narušení integrity aplikace její aktualizací skrze použité knihovny.	OK	Využití důvěryhodných zdrojů.

Zdroj: Vlastní zpracování (2023)

Tabulka 18 – Ověření A09:2021 Security Logging and Monitoring Failures

Zdroj zranitelnosti	Stav	Komentář
Chybí logování událostí jako jsou úspěšná a neúspěšná přihlášení nebo nestandardních operací.	OK	Logováno v aplikačním logu.
Varování a chyby neobsahují žádné nebo nedostatečné či nejasné informace.	OK	Logy jsou dostačující.

Aplikační logy a audit volání na API rozhraní nejsou sledovány pro podezřelou aktivitu.	N/A	Není provozováno.
Logy se ukládají pouze lokálně.	N/A	Není provozováno.
Nevhodné nastavení prahových hodnot pro automatizované varování v monitorování aplikace.	N/A	Není provozováno.
Neaplikováno penetrační testování a skenování pomocí nástrojů dynamického testování zabezpečení aplikací (například OWASP ZAP).	N/A	Není provozováno.
Monitorování nemůže detekovat, eskalovat nebo upozorňovat na aktivní útoky v reálném čase nebo téměř v reálném čase.	N/A	Není provozováno.

Zdroj: Vlastní zpracování (2023)

Tabulka 19 – Ověření A10:2021 Server Side Request Forgery

Zdroj zranitelnosti	**Stav**	**Komentář**
Změna volaných adres aplikací na straně serveru.	OK	Konfigurace je statická na serveru.
Infikování serveru nahráváním obrázků a jiných souborů.	N/A	Není API služba s tímto účelem.

Zdroj: Vlastní zpracování (2023)

6 Závěr

Cílem této diplomové práce bylo vytvoření návrhu systému s požadovanou vysokou úrovní zabezpečení. Návrh měl být určen pro architekty, analytiky a vývojáře těchto systémů. Následně měl být návrh ověřen realizací v rámci testovací demo aplikace a jejím zhodnocením (primárně bezpečnostní stránky). Dílčím cílem byla analýza použitelných technologií za účelem implementace uživatelské autentizace a autorizace v systémech, kde je vyžadována takto vysoká úroveň zabezpečení.

Ke splnění cíle bylo třeba nejdříve nastudovat obecnou teorii kolem identifikace, autentizace a autorizace uživatelů. V rámci této oblasti připravil autor v teoretické části práce soupis definic pojmů a soupis možných autentizačních metod. Následovalo zmapování a příprava přehledného soupisu technologií, standardů a protokolů používaných při návrhu a implementaci bezpečnostních mechanismů orientovaných do výše uvedené oblasti. Standardy a protokoly byly v jednotlivých kapitolách popsány, aby bylo jasné jejich fungování a využití.

Jak se ukázalo z dosavadní teoretické přípravy a praktických zkušeností autora práce ze stejné oblasti vývoje, tak výše zmiňované technologie jsou závislé a limitované dle použité architektury systému, do kterého mají být implementovány. Proto bylo třeba zmapovat i oblast architektury dnešních systému. V teoretické části práce jsou uvedeny základní architektury systémů, jejich rozdíly, výhody a nevýhody. V oblasti architektury se ukázalo, že aktuálním favoritem je mikroservisní architektura, která postupně vytlačuje původní monolitickou architekturu velkých podnikových systémů.

Další oblastí v teoretické přípravě bylo prozkoumání legislativních požadavků na podnikové systémy, od kterých je vyžadována vyšší úroveň zabezpečení. Důležité bylo ujasnit si, kterých subjektů se tyto zvýšené požadavky na bezpečnost týkají. Na úrovní Evropské Unie jsou vydané směrnice, které následně přebírají jednotlivé členské státy a upravují dle lokálních specifik a požadavků do vlastních směrnic. Vyšší úroveň zabezpečení se týká subjektů poskytujících v rámci státu základní služby. Detailní definice subjektů a jednotlivá důležitá nařízení jsou uvedeny v rámci teoretické části práce. V České republice funguje Národní úřad pro kybernetickou bezpečnost, který dodržování jednotlivých směrnic kontroluje a vydává velké množství bezpečnostních doporučení pro výše uvedené subjekty a jejich správce systémů.

Existují i otevřené komunitní organizace, které se zabývají bezpečnostní problematikou systémů a provádí analýzu aktuálních bezpečnostních hrozeb a doporučení pro jejich zamezení. Jednou z velkých a známých organizací je OWASP (Open Web Application Security Project), jejíž výstupy jsou uvedeny v teoretické části práce.

Následně ze získaných teoretických základů autor práce začal připravovat návrh bezpečnostního mechanismu pro identifikaci, autentizaci a autorizaci uživatelů v konkrétní architektuře systému, která byla součásti přípravy. Protože některé části bylo možné řešit více standardy či technologiemi, tak bylo třeba provést průzkum z běžně dostupných zdrojů na internetu, které standardy a technologie jsou v době vzniku této práce trendem. Ze získaných informací a praktických zkušeností autora práce byl připraven návrh popsaný převážně pomocí modelovacího jazyka UML (Unified Modeling Language). K jednotlivým částem návrhu byl přidán soupis doporučení a odůvodnění na základě informací z teoretické části práce.

Ke zhodnocení funkční stránky návrhu v rámci implementované demo aplikace bylo zapotřebí definovat jednotlivé testovací scénáře. V rámci testovacích scénářů byly pokryty všechny funkční požadavky na demo aplikaci. Každý scénář obsahoval vstupní podmínky, testovací případy a očekávaný výsledek. V případě, kdy jeden z testovacích scénářů nebyl úspěšný, tak byla provedena oprava v rámci implementace a následně testování opakováno včetně regresního testování i ostatních scénářů k ověření, že v rámci opravy nedošlo k rozbití jiné funkcionality.

K úplnému splnění cíle práce bylo zapotřebí naimplementovat funkční demo aplikaci na základě připraveného návrhu. Autor práce využil programovacích jazyků, se kterými má největší teoretické i praktické zkušenosti. Zároveň se i z průzkumů této práce ukázaly jako aktuálně nejvíce používané. Důvodem průzkumů bylo, aby demo aplikace mohla sloužit jako vzor pro největší množství vývojářů z dané oblasti. Demo aplikace je složená ze tří pomyslných částí. Jedna část představuje implementaci identity provideru (IDP), druhá část představuje modul se zabezpečenými mikroslužbami a třetí část představuje frontendového konzumenta těchto služeb a představuje grafické rozhraní celé demo aplikace. Použitými programovacími jazyky jsou Java a Javascript a moduly komunikují skrze REST API rozhraní. Demo aplikace pro svůj běh používá integrovaný aplikační server a v rámci veškerých dosavadních aktivit byla provozována pouze lokálně na stanici autora práce.

Ke zhodnocení bezpečnostních vlastností celého návrhu a vytvořené demo aplikace se nabízelo použití některého automatizovaného nástroje jako je například nástroj ZAP

vytvořený komunitou OWASP. Nicméně, aby testování přineslo objektivní výsledek, bylo by třeba aplikaci provozovat již na cílovém serveru, což nebylo v rozsahu této práce realizováno. Autor se rozhodl využít seznamu 10 nejčastějších zranitelností udržovaný také komunitou OWASP (OWAPS Top 10) k posouzení bezpečnostních vlastností implementovaného řešení. Pro každou zranitelnost bylo ověřeno, zdali je pro výstupy této práce relevantní a zda má proti této zranitelnosti dostatečnou ochranu či ji úplně eliminuje. Výsledkem byl soupis slovních zhodnocení k jednotlivým zranitelnostem a příčinám. Tento způsob penetračního testování autorovi umožnil i ověřit části návrhu, které nebyly přímo implementovány v demo aplikaci.

Hlavní přínos práce je rozdělen do dvou oblastí. Jednou z oblastí je oblast teoretická, kde práce poskytuje důležité znalosti z oblasti autentizace a autorizace v dnešních systémech v přehledné formě souhrnně z několika technických zdrojů. Protože je práce orientovaná na systémy s vysokou úrovní zabezpečení jsou důležitou součástí teoretické části i legislativní informace, které jsou přehledně zmapovány z oficiálních legislativních zdrojů. Druhou oblastí je oblast praktická, kde výstupy práce mohou využít vývojáři při implementaci podobných systémů. Jednotlivé části implementované demo aplikace mohou být použity v jiných systémech nebo se vývojáři mohou inspirovat použitými technologiemi. Zároveň má přínos i v oblasti penetračního testování aplikací, protože může sloužit jako vzor pro ověření bezpečnosti jiných systémů.

Do budoucna může být zdrojový kód demo aplikace poskytován v rámci autorova veřejného GitHube repozitáře. Tento kód může být dále rozvíjen, rozšiřován o ochrany a doporučení v rámci aktuálních bezpečnostních hrozeb. Dále by bylo vhodné implementaci rozšířit o sadu automatizovaných testů k automatickému regresnímu testovaní všech funkcionalit v případě nové verze aplikace.

7 Seznam použitých zdrojů

(1) ANDERSON, Ross. 2020. *Security engineering: a guide to building dependable distributed systems.* 3. vyd. Indianapolis: Wiley. 1186 s. ISBN 978-1119642787.

(2) *Authetication definition* [online]. [cit. 2023-02-17]. Dostupné z: https://www.techtarget.com/searchsecurity/definition/authentication

(3) *Authorization Cheat Sheet* [online]. [cit. 2023-02-17]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html

(4) CHILDERS, Dave. *State of the Auth* [PDF]. Duo Labs, 2021 [cit. 2023-02-17]. Dostupné z: https://duo.com/assets/ebooks/state-of-the-auth-2021.pdf

(5) *Application Security Verification Standard* [PDF]. Verze 4.0.3 The OWASP Foundation, 2021 [cit. 2023-02-17]. Dostupné z: https://github.com/OWASP/ASVS/raw/v4.0.3/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0.3-en.pdf

(6) *Single factor OTP* [online]. [cit. 2023-02-17]. Dostupné z: https://pages.nist.gov/800-63-3/sp800-63b.html#singlefactorOTP

(7) *RFC 4226 – HTOP* [online]. [cit. 2023-02-17]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc4226

(8) *RFC 6238 – TOTP* [online]. [cit. 2023-02-17]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc6238

(9) A Review Of Authentication Methods. INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH. 2016, 2016(11), 246-249. ISSN 2277-8616.

(10) *Smlouvu potvrdíte jako heslo do banky, startuje můj podpis* [online]. [cit. 2023-02-17]. Dostupné z: https://www.penize.cz/mobilni-bankovnictvi/419932-smlouvu-potvrdite-jako-heslo-do-banky-startuje-mujpodpis

(11) WILSON, Y., HINGNIKAR, A. 2019. *Solving Identity Management in Modern Applications: Demystifying OAuth 2.0, OpenID Connect, and SAML 2.0*. 1. vyd. Appress. 337 s. ISBN 978-1484250945.

(12) *Zákon o platebním styku č. 370/2017 Sb.* [online]. [cit. 2023-02-17]. Dostupné z: https://www.zakonyprolidi.cz/cs/2017-370

(13) PEYROTT, Sebastián. *JWT Handbook* [PDF]. Verze 0.14.1 Auth0 Inc., 2018 [cit. 2023-02-17]. Dostupné z: https://auth0.com/resources/ebooks/jwt-handbook

(14) *RFC 7519 – JWT* [online]. [cit. 2023-02-17]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc7519

(15) *RFC 7516 – JWE* [online]. [cit. 2023-02-17]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc7516

(16) *RFC 7515 – JWS* [online]. [cit. 2023-02-17]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc7515

(17) *RFC 7517 – JWK* [online]. [cit. 2023-02-17]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc7517

(18) *OpenID Connect Basic* [online]. [cit. 2023-02-17]. Dostupné z: https://openid.net/specs/openid-connect-basic-1_0.html

(19) *SAML Core 2.0 Standard* [PDF]. OASIS 2005 Open, 2005 [cit. 2023-02-17]. Dostupné z: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

(20) *RFC 7522 – SAML 2.0* [online]. [cit. 2023-02-17]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc7522

(21) *SAML Tools Web* [online]. [cit. 2023-02-17]. Dostupné z: https://www.samltool.com/generic_sso_res.php

(22) *RFC 4949 – Internet Security Glossary* [online]. [cit. 2023-02-17]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc4949

(23) Authentication vs Authorization [online]. [cit. 2023-02-17]. Dostupné z: https://learn.microsoft.com/en-us/entra/identity-platform/authentication-vs-authorization

(24) *OAuth 1.0 vs OAuth 2.0* [online]. [cit. 2023-02-17]. Dostupné z: https://medium.com/identity-beyond-borders/oauth-1-0-vs-oauth-2-0-e36f8924a835

(25) *RFC 5849 – OAuth 1.0* [online]. [cit. 2023-02-17]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc5849

(26) *RFC 6749 – OAuth 2.0* [online]. [cit. 2023-02-17]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc6749

(27) *RFC 7636 – Proof Key for Code Exchange by OAuth Public Clients* [online]. [cit. 2023-02-17]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc7636

(28) Royal, P. 2023. *Building Modern Business Applications: Reactive Cloud Architecture for Java, Spring, and PostgreSQL*. 1. vyd. Sherman Oaks. 185 s. ISBN 978-1-4842-8991-4.

(29) GILMAN, E. 2017. *Zero Trust Networks: Building Secure Systems in Untrusted Networks*. 1. vyd. O'Reilly Media. 240 s. ISBN 978-1491962190.

(30) *Modern Web Applications Characteristics* [online]. [cit. 2023-02-17]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/modern-web-applications-characteristics

(31) *Monolith To Microservices* [online]. [cit. 2023-02-17]. Dostupné z: https://www.onec1.com/thought-leadership/monolith-to-microservices

(32) *Mikroslužby a rizika s nimi spojená* [online]. [cit. 2023-02-17]. Dostupné z: https://www.cleverandsmart.cz/mikrosluzby-a-rizika-s-nimi-spojena

(33) *Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation* [online]. [cit. 2023-02-17]. Dostupné z: https://ieeexplore.ieee.org/abstract/document/9717259

(34) *Národní úřad pro kybernetickou bezpečnost* [online]. [cit. 2023-02-17]. Dostupné z: https://www.nukib.cz/cs/kyberneticka-bezpecnost/regulace-a-kontrola/legislativa

(35) *Vyhláška o bezpečnostních opatřeních, kybernetických bezpečnostních incidentech, reaktivních opatřeních, náležitostech podání v oblasti kybernetické bezpečnosti a likvidaci dat (vyhláška o kybernetické bezpečnosti)* [online]. [cit. 2023-02-17]. Dostupné z: https://www.zakonyprolidi.cz/cs/2018-82

(36) *SMĚRNICE EVROPSKÉHO PARLAMENTU A RADY (EU) 2016/1148* [online]. [cit. 2023-02-17]. Dostupné z: https://eur-lex.europa.eu/legal-content/CS/TXT/HTML/?uri=CELEX:32016L1148&from=EN#d1e39-1-1

(37) *NAŘÍZENÍ EVROPSKÉHO PARLAMENTU A RADY (EU) 2019/881* [online]. [cit. 2023-02-17]. Dostupné z: https://eur-lex.europa.eu/legal-content/CS/TXT/HTML/?uri=CELEX:32019R0881&qid=1699518886247#d1e40-15-1

(38) OWASP Top 10 Vulnerabilities [online]. The OWASP Foundation, 2021 [cit. 2023-02-17]. Dostupné z: https://github.com/OWASP/Top10/blob/master/2021/docs/index.md

(39) *BEZPEČNOSTNÍ DOPORUČENÍ NÚKIB PRO ADMINISTRÁTORY 4.0* [PDF]. NÚKIB, 2020 [cit. 2023-02-17]. Dostupné z: https://www.nukib.cz/download/publikace/doporuceni/Doporuceni_admin_4.0_brozura_modra.pdf

(40) *MINIMÁLNÍ BEZPEČNOSTNÍ STANDARD* [PDF]. NÚKIB, 2020 [cit. 2023-02-17]. Dostupné z: https://www.nukib.cz/download/publikace/podpurne_materialy/minimalni-bezpecnostni-standard_v1.2.pdf

8 Seznam obrázků, tabulek a zkratek

8.1 Seznam obrázků

8.2 Seznam tabulek

8.3 Seznam použitých zkratek

API	Application Programing Interface
CORS	Cross Origin Resource Sharing
CSS	Cascade Style Sheet
HTOP	HMAC based One Time Pasword
HTML	Hypertext Markup Language
IDP	Identity Provider
OIDC	OpenID Connect
ORM	Object Relation Mapping
OTP	One Time Password
TOTP	Time based One Time Password

Přílohy

Příloha A Zdrojové kódy demo aplikace