Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informačních technologiích

Současné bezpečnostní metody k autentizaci a autorizaci uživatele

Bakalářská práce

Autor: Mgr. Long Do

Studijní obor: Aplikovaná informatika

Vedoucí práce: Mgr. Josef Horálek, Ph.D.

Hradec Králové srpen 2022

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedených zdrojů.

V Hradci Králové dne …… …….

podpis autora

Poděkování:

Na tomto místě bych velmi rád poděkoval Mgr. Josefovi Horálkovi, Ph.D. za jeho cenné rady, vstřícnost a trpělivé vedení práce. Dále děkuji kolegům, díky kterým je IT stále fascinující oblastí. V neposlední řadě děkuji Barboře Pohořské za korektury textu, podporu, kterou mi v průběhu vypracování práce poskytla, a za životní inspiraci.

Anotace

Cílem práce je analýza současných bezpečnostních praktik v autentizaci uživatele prostřednictvím open-source poskytovatelů identifikačních služeb a jejich komparace s důrazem na open-source řešení Keycloak. Výstupem bude webová aplikace, jejíž proces ověřování se provádí prostřednictvím serveru Keycloak. Webová frontend aplikace bude vyvíjena ve frameworku SvelteKit a prostřednictvím REST API bude možné provést přihlášení, odhlášení a autorizované vkládání nebo úpravy dat.

Klíčová slova

Keycloak, autorizace, autentizace, protokoly, správce identity, SAML 2.0, OpenID Connect, OAuth 2.0

Annotation

Title: Current security methods for user authentication and authorization

The aim of the work is the analysis of current security practices in user authentication through open-source providers of identification services and their comparison with an emphasis on the open-source Keycloak solution. The output will be a web application whose authentication process is performed through the Keycloak server. The web application will be developed in the SvelteKit framework and through the REST API it will be possible to perform login, logout and authorized data insertion or modification.

Keywords

Keycloak, autorization, authentication, protocols, identity access and management, SAML 2.0, OpenID Connect, OAuth 2.0, Keycloak

Obsah

1 Úvod 1

1.1 Cíl práce a metodologie 1

2 Kybernetická bezpečnost 3

2.1 Kyberprostor a bezpečnost 3

2.1.1 Principy kybernetické bezpečnosti 3

2.2 Kybernetické hrozby a útoky 5

2.2.1 Rizikové prostředí a prevence před útoky 5

3 Emil odposlouchává Alenu a Barboru 8

3.1 Symetrické šifrování, soukromý klíč jako nástroj k šifrování 9

3.1.1 DES (data encryption standard) 9

3.1.2 AES (Advanced Encryption Standard) 10

3.2 Asymetrické šifrování 10

3.2.1 Diffie-Hellman 10

3.2.2 Rivest-Shamir-Adleman 11

3.2.3 Certifikační a podpisové autority 11

4 Správa identit 14

4.1 Vývoj od raných počátků až po současnost 14

4.2 Autentizační a autorizační protokoly 17

4.2.1 Kerberos, LDAP a předehra k moderním protokolům 18

4.2.2 OAuth 2.0 19

4.2.3 OpenID Connect 22

4.2.4 SAML 2.0 24

4.3 Životní cyklus spravování identity 25

4.4 Současní správci identit 29

4.4.1 Keycloak – Keycloak v19 30

4.4.2 Gluu – Gluu v4 30

4.4.3 Ory – v0.10.0 31

5 Praktická část 34

5.1 Svelte + Sveltekit 34

5.2 Typescript 37

5.3 Hasura 38

5.4 GraphQL + Urql 39
5.5 Keycloak 41
5.6 Zpracování a výsledek 41
6 Závěr 42
7 Zdroje 44
7.1 Literární zdroje 44
7.2 Elektronické zdroje 45
8 Přílohy 50

1 Úvod

Ve zprávě o stavu kybernetické bezpečnosti za rok 2021 se uvádí, že za rok 2021 „*se s phishingovými e-maily setkalo 90 % respondentů, se spear-phishingovými e-maily 47 % respondentů a s podvodnými e-maily 84 % respondentů.*“ (NÚKIB, 2022, s. 23) V roce 2021 měl 10% zastoupení v incidentech podvod, do kterého spadá právě phishing, krádež identity nebo neoprávněné využití ICT (NÚKIB, s. 13). Kybernetické hrozby, které jsou klasifikovány jako pokusy o průnik, např. pokus o přihlášení, mají zastoupení v 1% incidentů za minulý rok. Ačkoliv trendy hrozeb nevykazují značný zájem o krádeže identity, nelze si nevšimnout nárůstu aktivit v oblasti vývoje technologií, které posilují úroveň ověřování přístupů do aplikací. Jedná se snad již o „společenskou“ normu, když každý bankovní úkon vyžaduje nějakou formu potvrzení, že žádost o něj pochází skutečně z naší strany?

1.1 Cíl práce a metodologie

Tato práce si klade za cíl zjistit, jaké moderní principy bezpečnosti vztažené na správce identity se dnes uplatňují, a jaké nároky se kladou na open-source služby, které zprostředkovávají autorizační a autentizační procesy.

Pro splnění vytyčeného cíle je nezbytné zodpovědět na tyto otázky:

1. Jaké současné protokoly se převážně využívají v oblasti autentizace a autorizace? Je důvod, proč konkrétní protokoly byly stanoveny normou pro zabezpečení?
2. Co je klíčové pro zabezpečení správců identit? Jak je zabezpečení dosaženo?
3. Které oblasti jsou nejvíce exponované u správců identity?
4. Jaké znaky vykazují moderní open-source správců identity? Na co se zaměřují?

Dále je součástí práce praktický výstup. Ten bude realizován tak, že dojde k vytvoření prototypu aplikace, jež bude k autentizaci využívat token poskytovaný serverem Keycloak a která zajistí zabezpečený přístup k aktivům.

Ve své teoretické části, která se zabývá oblastí správců identity, čerpá tato práce z knihy Yvonne Wilsonové a Abhisheka Hingnikara *Solving Identity management in Modern Application: demystifying OAuth 2.0, OpenID Connect, and SAML 2.0.* Základ týkající se oblasti šifrování, zabezpečení sítí a kyberbezpečnosti nabídly převážně knihy od Karla Burdy *Kryptografie okolo nás*, Libora Dostálka *Velký průvodce protokoly TCP/IP Bezpečnost,* Jana Koloucha *Cybersecurity* a Rity Pužmanové *TCP/IP v kostce.*

Pro analýzu vybraných open-source projektů od firem Ory Corp a Gluu Inc a Keycloaku sloužily jejich oficiální dokumentace. Oficiální dokumentace byla také využita ke studiu jednotlivých nástrojům a jejich funkcionalit, jenž byly využity v praktické části práce. Jedná se konkrétně o dokumentace týkající se nástrojů *GraphQL, Hasura, Heroku, Svelte, Sveltekit, Typescript* a *urql*.

2 Kybernetická bezpečnost

Tato kapitola se věnuje pojmům souvisejícím s kybernetickou bezpečností, autorizací a autentizací. Představují cíle ochrany ve virtuálním světě.

2.1 Kyberprostor a bezpečnost

Pojem kyberprostoru definuje Jan Kolouch (2019, s. 36) jako dostupný virtuální prostor skládající se z informačních a komunikačních technologií, který se řídí protokolem TCP/IP, a společně s počítačovými systémy tvoří celosvětovou počítačovou síť. Oblast kybernetické bezpečnosti lze shrnout do dvou stěžejních bodů – obrana před hrozbami a řešení následků. Důležitou součástí bezpečnosti ve virtuálním světě je způsobilost ochrany před kybernetickými hrozbami či útoky. Metody prevence a obrany se opírají o soubory právních, organizačních, technických a vzdělávacích prostředků. Druhý bod zahrnuje nejen řešení vzniklých škod, ale i dovednost bezprostřední reakce v průběhu napadení. Dle Koloucha (2019, s. 44) je podstatou, aby počítačové systémy, další prvky ICT a s nimi související služby, byly schopné se bránit před hrozbami ve virtuálním světě a zároveň aby byly způsobilé k adekvátním reakcím a řešením následků.

2.1.1 Principy kybernetické bezpečnosti

Paul C. Van Oorschot a Jan Kolouch mají na principy kyberbezpečnosti odlišné pohledy. Van Oorschot (2021, s. 3) shrnuje kyberbezpečnost do šesti bodů, kterými jsou důvěrnost (confidentiality), integrita (integrity), svolení k přístupu (authorization), dostupnost (availability), ověření (authentication) a odpovědnost (accountability). Na druhé straně Kolouch (2019, s. 45) vysvětluje kyberbezpečnost pomocí triád, které mohou působit konkrétněji než přístup Vana Oorschota.

Začněme šestibodovým přehledem Vana Oorschota. Pojem důvěrnost Van Oorschot dokládá na premise, kdy přístup k aktivům má pouze ověřený uživatel. Vlastnost skrytí před veřejnými entitami se netýká pouze dat, která jsou na uložištích zálohována, ale i dat, která jsou v procesu zpracovávání, kdy dochází k jejich přesunu při sdílení dat mezi dvěma stranami. Obvykle jsou tato data šifrována bezpečnostním algoritmem.

Následující princip integrity může být vnímán jako zachování stabilního stavu ať už dat, softwaru či hardwaru. Snahou je udržení konzistence dat, která by neměla být neplánovaně a bez svolení upravována. Předně je také důležité, aby byly učiněny preventivní kroky, které by zabránily poškození dat, které by mohlo přinést nežádoucí změny. Záruku integrity dat zprostředkovávají kódy pro kontrolu chyb v datech (error correcting codes) a kryptografické hašovací funkce (cryptographic checksum).

Rita Pužmanová (2009, s. 519) definuje autorizaci jako určení, které operace a data jsou dostupné. Jednotlivé operace proto podléhají autorizované kontrole, na kterou dohlíží autorizovaná entita, kupříkladu autorizovaný vlastník dat či vybraný autorizovaný server.

Plnění principu dostupnosti se docílí zajištěním spolehlivého zázemí pro hardware a software a jejich ochranu před narušením chodu nebo přetížením. Kolouch (2019, s. 54) je dále popisuje jako jistotu přístupu k informacím, datům nebo systému ve chvíli, kdy je to potřebné. Zajištění dostupnosti je, jak uvádí Kolouch (2019, s. 54-55), docíleno zálohováním dat či správou funkčnosti zálohovacích systémů.

Ověřitelnost je předposledním a neméně důležitým aspektem bezpečnosti cílící na kontrolu kontextu a identity. Podle Pužmanové (2009, s. 517) se jedná se o ověřování a potvrzování entit, tj. komunikujících stran. Ověřováním se dokládá důvěra entitě, která prošla autentizací a může takpracovat s daty či s jinými prostředky, ke kterým získala přístup. Vezmou-li se v úvahu principy integrity, svolení k přístupu (autorizace) a důvěrnosti v kontextu principu ověřitelnosti, je mezi nimi možné najít úzkou propojenost. Autentizací identity se ujišťujeme, že aktiva jsou zpřístupněna ověřené entitě. Plánované vykonávání požadavků ověřované identity pak musí projít autorizací a kontrolou, zda v rozsahu jejích pravomocí může pracovat s daty. Takto se hlídá integrita dat. Data musí být nedotčena a nepoškozena.

Smysl posledního principu, principu odpovědnosti, je založen na evidování aktivit entity zodpovídajících za předešlé prováděné změny. Záznamy se mohou provádět v jakékoli elektronické formě (Van Oorschot, 2021, str. 4).

První trojice CIA[1] z Kolouchova rozdělení (2019, s. 45), zahrnuje pojmy důvěrnost, integrita a dostupnost, které se obecně vztahují ke zpracovávání informací. Ve druhé triádě se Kolouch zaměřuje na problematiku prvků, které se rozdělují podle hlediska lidí, technologií a procesů. Poslední skupinou, podle které lze dodržovat kybernetickou

[1] Akronym vznikl složením počátečních anglických slovíček confidentiality, integrity a availability.

bezpečnost, je životní cyklus samotné kybernetické bezpečnosti. Jednotlivé cyklické etapy se opírají o prevenci, detekci a reakci před hrozbami.

Van Oorschotovy principy je možné brát jako obecný souhrn cílů kybernetické bezpečnosti a naopak Kolouchovy interpretace se zaměřují na konkrétnější kybernetické hrozby, jež autor rozděluje dle vymezených oblastí. Oba přístupy předkládají odlišné pohledy na uchování kybernetické bezpečnosti. Tato práce se na následujících stranách odkazuje na Oorschotovy principy, protože zahrnují obecnější cíle, které lze použít při analýze správců identit. Kolouchova rozdělení se vztahují na další faktory bezpečnosti, které jsou nad rámec této práce (např. technologické zázemí, analýza rizik či realizace opatření proti kybernetickým útokům).

2.2 Kybernetické hrozby a útoky

Dojde-li k narušení nebo poškození počítačového systému nebo sítě, jehož následkem je změna informace, aplikace či samotného systému, tento čin se klasifikuje jako akt směřující ke škodě a dle Koloucha (2019, s. 75) o něm lze hovořit jako o kybernetické hrozbě.

Kolouch (2019, s. 77) předkládá taxonomii kybernetických hrozeb členěných dle cílů hrozby; buď se jedná o útoky na principy důvěrnosti, integrity a dostupnosti, nebo je cílem poškodit prvky kybernetické bezpečnosti. V takovém případě se do ohrožení dostávají lidé, technologie a procesy.

Mezi útoky, které jsou ovlivněné lidským faktorem, se řadí sociální inženýrství, phishing, malware, krádeže a další. Útoky cílící na technologie mohou poškozovat zejména počítačové systémy, servery, síťovou infrastrukturu, operační systémy či jiné aplikace a datová uložiště. Kolouch (2019) dále popisuje útoky na procesy, u nichž uvádí příklady založené na neopodstatněném testování, fungující bezpečnosti i funkcionalitě testování.

2.2.1 Rizikové prostředí a prevence před útoky

S ohledem na tematickou oblast práce, jež se dotýká problematiky autorizace a autentizace, a podkapitoly zaměřující se na kybernetické hrozby, se tato část věnuje útokům, které mohou ohrozit odcizení identity či kompromitovat ochranu digitálních dat.

Dále se zabývá tím, jak se preventivně v prostředí kyberprostoru chránit. Základním pojmem, který je potřeba stanovit je aktivum. Aktivum může být čímkoliv, co má specifickou hodnotu pro osobu, organizaci či stát, a jeho rys není omezen hmotnou nebo nehmotnou podobou (Kolouch, 2019, s. 72).

Článek *Identity is key to stopping these 5 cyber security attacks* (Okta, 2021) se zaměřuje na nejčastější útoky v kyberprostoru ohrožující identitu, jimiž jsou útoky typu phishing, spear phishing, credential stuffing, password spraying a man-in-the-middle.

Cílem phishingu je získání údajů o uživateli klamavými metody, které spočívají ve snižování ostražitosti a vzbuzování důvěry uživatelů (Kolouch, 2016, s. 246). Převážně je phishing využíván v e-mailech. Příjemce takového e-mailu je obelháván odesílatelem, který se vydává za někoho jiného. Příjemce odesílateli uvěří, následkem čehož s ním pak může např. sdílet osobní údaje. Ty pak odesilateli zpřístupní osobní účty příjemce, aniž by si byl příjemce uvědomil důsledek svých činů. Šedivec (2021) tak doporučuje kontrolu odkazů, které příjemce přesměrovávají na neznámé stránky, či ověřování validity e-mailu a jméno odesílatele.

Spear fishing se podobá útoku phishing. Rozdíl spočívá v tom, že útok je zaměřen na specifický subjekt, ať už na jednotlivce nebo organizaci. Prevence vůči takovému útoku se shoduje s phishingem (Šedivec, 2021). Okta (2021) řadí tyto útoky do metod sociálního inženýrství, které staví na zvědavosti, strachu či odměně. Upozorňuje na komplexní úroveň manipulací prostřednictvím e-mailové zprávy.

Mezi útoky, jejichž doménou je napadení slabých hesel, patří credential stuffing (sběr přihlašovacích údajů). Credential stuffing spočívá v tom, že se nejdříve z různých zdrojů, ať už prostřednictvím dark webu či ze stránek, které již byly prolomeny, shromáždí velké množství přihlašovací údajů, které se následně využijí k útoku. Jeho silnou vlastností je možnost automatizovaného útoku.

Password spraying (heslový rozprašovač) lze přiřadit podle Okta (2021) k útokům hrubé síly. Koncept útoku je takový, že útočník určí běžné heslo, které splňuje pravidla vybraných domén. S daným heslem se pak zkouší přihlásit. Útočníci využívají pravděpodobnost skutečnosti, že mnoho uživatelů používá k zabezpečení svých účtů velice slabá a snadno odhadnutelná hesla, např. 1234. Snížením pokusů o přihlášení na jeden pokus jsou tyto útoky pak obtížně detekovatelné a na první pohled nemusí být správcovi domény jasné, že se jednalo o útok. Ač se může zdát, že útoku typu password spraying se

nepodaří prolomit účet, opak bývá pravdou. Okta (2021) např. upozorňuje na informaci, že heslo „password1“ bylo využito více než 2 milión krát. Šedivec (2021) jako prevenci proti útokům na hesla doporučuje využívání vícefaktorové autentizace a nastavení silného hesla. Heslo by mělo být jedinečné a nemělo by se shodovat v jiné registrované službě. Další prevencí je pravidelná aktualizace hesla.

Útoky směřované na komunikaci mezi dvěma stranami se nazývá man-in-the-middle (muž uprostřed). Jejich cílem je narušit základní principy kybernetické bezpečnosti jako důvěrnost a integritu dat. Při těchto útocích tedy dochází k poškozování sdílených dat mezi komunikujícími stranami. Lze rozlišovat útoky, které se odehrávají na zařízení nebo mimo něj. Útok obecně spočívá v tom, že napodobuje určité známé prostředí. Kupříkladu útočník napodobí přístup k veřejné wifi a nic netušící uživatel se k ní připojí v kavárně a zadá do ní své přihlašovací údaje. I přes šifrování se k těmto datům může útočník dostat a získat případný přístup do aplikace (Okta, 2021). Prevencí, jak doporučuje Šedivec (2021), je aktualizovat operační systém, kontrolovat přístup k zařízení a obezřetněji hlídat odkazy stránek.

Okta (2021) také prezentuje koncept princip zero trust (nulová důvěra), jehož jádrem je přístup „nikdy nedůvěřuj, ale prověřuj“. Koncept se v praxi užívá tak, že identitám nebo požadavkům se nedůvěřuje, dokud se neověří validita nebo se neautorizuje žádost.

3 Emil odposlouchává Alenu a Barboru[2]

Následující kapitola se konkrétněji zaměří na oblast zabezpečení přenosového systému, např. počítačové sítě či internetu. Zabezpečení přenášených údajů, které jsou sdíleny pouze mezi ověřenými účastníky v kyberprostoru, se zajišťuje šifrovacími metodami.

Výklad je opřen o Karla Burdu (Burda, 2019) a Ritu Pužmanovou (Pužmanová, 2009). Vychází z jejich výkladů o šifrovacích metodách a o procesu zabezpečení komunikace v kyberprostoru. Pohled do problematiky také doplňuje Libor Dostálek (Dostálek et al, 2003) především se svým přehledem o certifikační autoritě a Simon Singh (Singh, 2003), který přináší historický kontext pozadí vývoje Diffie-Hellmanova algoritmu.

Současné šifry se zaměřují na šifrování obsahu zprávy a zároveň na ověřování odesílatele. Simon Singh (2003, s. 232) píše, že sebesložitější šifrování je založeno na dvou základních operacích, na substituci a transpozici. Jak je zřejmé z názvu, metodou substituce je nahrazení znaků sdílené zprávy jinými. U transpozice se mění umístění znaků ve zprávě. Věda, jež problematiku zabezpečení obsahu a ověření autorství sdílené zprávy zkoumá, je nazývána kryptografie. Slovy Burdy (s. 19, 2019) je kryptografie vědou, která „zkoumá matematické metody utajování obsahu i prokazování původu přenášených zpráv.“ Zprávu můžeme vnímat jako aktivum.

Šifrováním zabezpečujeme komunikaci. Dělí se na asymetrické a symetrické šifrování. V obou metodách hraje důležitou roli funkce klíče. U symetrického šifrování se sdílí mezi komunikujícími stranami jeden stejný klíč, který je nazýván soukromý (private key). Je charakteristický tím, že jedním klíčem se šifruje a dešifruje zároveň. Pužmanová (2009, s. 507) vysvětluje, že symetrický způsob šifrování se využívá v procesu autentizace či v přenosových systémech. Naopak asymetrické šifrování pracuje s dvěma klíči, jeden je veřejným klíčem a druhý je privátním klíčem. Účastnící komunikace asymetrického šifrování si určují vlastní soukromý klíč, který s nikým, ani mezi sebou, nesdílí.

[2] Název kapitoly byl inspirován aktéry z problematiky tzv. distribuce klíčů. Při vysvětlování distribuce klíčů se v zahraniční literatuře ustálili tři protagonisté se jmény Alice, Bob a Eve (Singh, 2003, s. 243). V práci pojmenované postavy jsou využity při výkladu vztahů mezi útočníkem a těmi, kteří chtějí ochránit své soukromí.

K druhému klíči mají přístup bez výjimky všichni. Každý z klíčů zastává buď roli šifrovacího nebo dešifrovacího klíče.

Kritickým aspektem při ustavování klíčů je jejich distribuce mezi komunikujícími stranami. Distribuce klíčů se týká obou typů šifrování. Singh (2003, s. 244) píše, že výměna klíčů bude vždy pro kryptografii důležitá, avšak nejedná se o problém, který by neměl řešení.

3.1 Symetrické šifrování, soukromý klíč jako nástroj k šifrování

Symetrické šifrování je typické tím, že pomocí jednoho klíče, kterému se říká soukromý klíč, se šifruje a zároveň dešifruje zpráva. Pokud známe jeden z klíčů, lze druhý odvodit velmi snadno, jak tvrdí Burda (2019, s. 21). Tento způsob šifrování se užívá nejvíce v oblasti zpracovávání velkého množství dat. Výrazným kladem je rychlost zpracování a negativem je složité předávání klíče adresátovi (Burda, 2019). Z důvodu provádění rychlých a jednoduchých výpočtů je délka klíčů krátká, jak popisuje Pužmanová (2009, s. 508). Autorka také zdůrazňuje, že je potřeba zvýšeného zabezpečení klíče při přenosu v kyberprostoru, proto se soukromý klíč mění.

3.1.1 DES (data encryption standard)

Příkladem symetrického šifrování je algoritmus DES, který od roku 1976 sloužil jako norma bezpečného šifrování. Za vývojem stála americká firma IBM a tento standard byl využíván bezpečnostní agenturou americké vlády NSA. DES je řazen mezi tzv. blokové šifry[3], které jsou charakteristické šifrovaným blokem, jehož délka je určena násobkem délky písmen abecedy. Blok algoritmu DES má 8 bytů, délka klíče v algoritmu DES má 64 bitů. Dostálek píše (2003, s. 215), že časem bylo potřeba navýšit délku klíče pro větší bezpečnost. Proto vznikaly další algoritmy jako 3DES, IDEA či Blowish. 3DES byl vylepšený algoritmus DES a vyznačoval se větší délkou 168bitového klíče.

Zabezpečení algoritmem DES nebylo dostačující a měl slabiny vůči útokům hrubou silou. Např. Deep Crack je dostupný nástroj, který umí DES právě prolomit (Simplelearn,

[3] Existují také proudové šifry, jež jsou založené na zpracovávání se slovy. Délka je rovna jednomu písmenu a využívají se při znakově orientovaných přenosech informací (Dostálek, 2003, s. 215).

2022). Užívání DES skončilo v roce 2002, kdy byl nahrazen novým standardem AES (advanced description standard).

3.1.2 AES (Advanced Encryption Standard)

Nástupcem DES se stal AES, jenž je také znám jako Rijndaelův algoritmus, a využívá klíče o délkách 128 bitů, 192 bitů nebo 256 bitů. Za jeho vývojem stál tým vědců z vlámské Katolické univerzity v Lovani. Algoritmus je založen na substitučně-permutační síti. Oproti DES je rychlejší a bezpečnější (Simplelearn, 2022). Algoritmus se využívá v oblasti zabezpečení bezdrátových sítí, procesorů, prohlížení webových stránek či šifrování souborů.

3.2 Asymetrické šifrování

Asymetrické šifrování, také známé jako šifrování veřejným klíčem, pracuje s dvojicí klíčů, jedná se o soukromý klíč a veřejný klíč. Je přesným opakem symetrického šifrování. Každý z klíčů zastupuje jednu roli v procesu šifrování a dešifrování. Vlastní soukromý klíč si vygeneruje každá strana zvlášť a tím také každá strana zvlášť zodpovídá za jeho bezpečné uchovávání. Při každé nové relaci se vygenerují nové klíče. Burda (2019, str. 21) s Van Oorschotem (2021, s. 214) sdílí názor, že určit hodnotu soukromého klíče pomocí veřejného klíče je velmi obtížný až nemožný úkol.

Velkou výhodou je možnost distribuovat veřejný klíč nezabezpečeným kanálem, jak píše Pužmanová (2009, s. 510). Toto šifrování je narozdíl od symetrické metody pomalejší a je využíváno pro šifrování menšího objemu dat. Převážně se využívá pro šifrování klíče, který je potřeba sdílet mezi komunikujícími stranami.

3.2.1 Diffie-Hellman

V roce 1976 představili pánové Whitfield Diffie, Martin Hellman a Ralph Merkel standard pro šifrování prostřednictvím veřejného klíče, který odhalil způsob bezpečné distribuce šifrovacího klíče. Problematika předávání klíčů má tři cesty řešení, jednou z nich je předávání právě prostřednictvím Diffie-Hellmanova algoritmu. „Bezpečnost naprosté

většiny moderních kryptosystémů je budována na pesimistickém předpokladu, že útočník ví o daném kryptosystému úplně vše, kromě tajného, resp. soukromého klíče," tak se vyjadřuje k obecné problematice sdílení Burda (2019, str. 30).

Podle Pužmanové (2009, s. 511) se Diffi-Hellmanův algoritmus „zakládá na NP-složitosti výpočtu diskrétních logaritmů v poli konečné délky". Výhradně se používá pro šifrování distribučního klíče, jenž je použit jako nástroj k šifrování dat. Komunikující strany se domluví na společném klíči, díky němuž se zabezpečí obsah zpráv, který si strany sdílejí. Bohužel tento typ asymetrického šifrování může čelit kybernetické hrozbě typu man-in-the-middle. Ta využívá slabiny, že si účastníci komunikace vzájemně nezkontrolují identitu. Hrozba nastane v okamžiku, kdy v průběhu komunikace jedna strana získá podvržený klíč, ale bude si myslet, že se jedná o správný klíč druhé strany (Pužmanová, 2009, tamtéž).

3.2.2 Rivest-Shamir-Adleman

Další velmi silný algoritmus pochází od trojice Ronald Rivest, Adi Shamir a Leonard Adleman. Podle nich byl algoritmus pojmenován. Často je využíván ve sféře elektronické korespondence, digitálních podpisů nebo budování privátních sítí. Slouží nejen k šifrování komunikace, ale i k autentizaci. Je znám také v oblasti zabezpečení webu pomocí Secure Sockets Layer. „RSA je založen na NP-složitosti faktorizace velkých celých čísel" (Pužmanová, 2009, s. 511). Metoda RSA vyžaduje klíč minimálně přes 100 číslic dlouhých a jeho výše bezpečnosti se zlepšuje dle délky klíče. Čím delší je řetězec klíče, tím je spolehlivější zabezpečenější.

3.2.3 Certifikační a podpisové autority

Nástroj veřejného klíče nalézáme také v digitálních certifikátech, které slouží k identifikaci uživatele, a v digitálních podpisech, kde je klíč úzce navázán na zprávu odesílatele. Digitální certifikáty ukládají citlivé údaje o jeho držiteli a jsou vydávány certifikační autoritou (CA). CA zaručuje důvěrnost údajů mezi účastníky. Certifikáty jsou

omezeny časem, proto je důležité je pravidelně kontrolovat. S certifikátem je spojen pojem infrastruktura veřejného klíče (PKI)[4].

PKI značí souhrn protokolů, které jsou vzájemně provázané a dbají na normu využívání šifrovacích klíčů a certifikátu (Dostálek et al., 2003, s. 223). Certifikát se skládá z dat profilu držitele, veřejného klíče, identifikace a podpisu vydavatele, vlastního identifikačního čísla, platnosti a rozsahu využití. Využíváním certifikátu preventivně lze zabránit útokům typu man-in-the-middle. Možná kybernetická hrozby je uvedena na následujícím příkladě. Existují dvě uživatelky Alena a Barbora, které chtějí navázat komunikaci mezi sebou, a pro zabezpečení jejich přenosového kanálu si vygenerují dvojici soukromého a veřejného klíče. Do jejich komunikace bez jejich vědomí vstoupí agresor Emil, jehož cílem bude podvrhnout veřejný klíč a to tak, že jej zamění při přenosu. Vymění svůj klíč za Alenin veřejný klíč. Barbora, aniž by tušila, že nedostala původně adresovaný veřejný klíč od Aleny, zašifruje svoji zprávu věnovanou Aleně šifrovacím klíčem od Emila a zprávu odešle Aleně. Zprávu však zachytí Emil a dešifruje její obsah.

Aby se takovému druhu útoku zabránilo, využije se certifikát, který je vystaven důvěryhodnou autoritou. Součástí žádosti, která je podepsána soukromým klíčem uživatele, jsou např. data o uživateli a jeho veřejný klíč. Certifikační autorita na základě žádosti zpracovává věrohodnost uživatele a až po validaci identity, vydá certifikát, jenž podepíše vlastním soukromým klíčem, který si autorita vygenerovala spolu s veřejným klíčem. Žadatel přijme certifikát, který odešle adresátovi. Ten si prověří autentičnost a shledá-li, že se nejedná o podvrh, využije veřejný klíč odesílatele z certifikátu. V navazujícím příkladu si Alena pomocí certifikátu ověřila totožnost odesílatele zprávy Barbory. Veřejným klíčem z certifikátu zašifruje text a odešle Barboře, která si původní zprávu dešifruje pomocí vlastního soukromého klíče.

Veřejný klíč ve formě digitálního podpisu v dokumentu slouží k autentizaci autora zprávy. Cílem tohoto klíče není zpracovat celou zprávu do šifry, ale pouze šifruje dílčí část s pomocí jednosměrné hašovací funkce SHA-1, která produkuje hodnotu o délce 128 bitů, nebo prostřednictvím algoritmu MD5, jehož výsledek dosahuje délky 160 bitů. MDA (message digest algorithm) spadá mezi sekvenční algoritmy. Jeho vyprodukované šifře se říká otisk, který dosahuje délky 128 bitů. Přestože samotný otisk neodhalí obsah zprávy,

[4] Neboli norma public key infrastructure, které jsou definovány v RFC 3279 a 5280.

dokládá, zda během přenosu zprávy, nedošlo k manipulaci s obsahem. Výstupy algoritmů jsou jedinečné, nedochází tak k záměnám zpráv. (Pužmanová, 2009, s. 512)

Algoritmická norma SHS (secure hash standard) vytváří výstupy o délce 160 bitů a její proces zpracování může být ve srovnání s některými algoritmy MDA pomalejší, zato její šifrování, jak může být zjevné z délky bitů, bývá výrazně bezpečnější. Algoritmus slouží převážně pro ověření autenticity poslané zprávy. Jeho ověřovací metoda je postavena na srovnání vygenerovaných řetězců odesílatelovy zprávy a zprávy adresátovi. Zpráva se nejdříve zahešuje. Zahešovaná zpráva se pak šifruje soukromým klíčem. (Pužmanová, 2009, s. 512)

Mluvíme-li v kontextu zabezpečení zprávy, respektive o elektronickém podpisu, elektronické obálce[5] nebo autentizaci dat, nelze opomenout normu CMS[6]. Proces zpracování zabezpečení obsahu zprávy probíhá tak, že jsou pokaždé definovány dvojice prvků, označení druhu dat (contentType) a samotná data (content). Dvojice prochází dvěma koly zpracování dle CMS. Nejdříve do procesu CMS vstoupí nezašifrovaná dvojice, ta je následně signaturována digitálním podpisem. Příznaky u dvojic se změní na podepsaný prvek A a podepsaný prvek B. Druhá vlna zpracování označí tyto prvky za zašifrované, které jsou opatřeny digitálním podpisem (Dostálek et al., 2003, s. 289).

V *Minimálních požadavcích kryptografických algoritmů* vydaných Národním úřadem pro kybernetickou a informační bezpečností (2022) se doporučují symetrické algoritmy typu AES s využitím klíčů o délkách 128, 192 a 256 bitů. Dále se doporučení týká ukončení využívání 3DES algoritmů s určenou délkou o 112 bitech, které mají v roce 2023 dosloužit. NÚKIB schválil Diffie-Helmanův algoritmus pro procesy, ve kterých se stanovují klíče a jejich šifrování s konkrétními parametry, kde délka klíčů je 3072 a více. Naopak algoritmus s délkou 2048 se již bude v roce 2023 řadit mezi dosluhující.

S příchodem kvantových počítačů, jejichž výpočetní zdatnost dle předpovědí značně předčí současné vyvíjené počítačové systémy (Sedlák, 2021), pravděpodobně dojde k oslabení některých šifrovacích metod, neboť k jejich prolomení by mohlo docházet rychleji (Kolouch, 2019, s. 485).

[5] Dostálek (2003, s. 289) označuje elektronickou obálku to, co je zašifrované.
[6] Dříve znám pod protokolem PKCS#7, současná norma CMS (Cryptographic Message Standard) je definována v RFC 3369 (Dostálek et al., 2003, s. 288).

4 Správa identit

Nadcházející kapitola si klade za cíl popsat historický vývoj správců identit. Věnuje se problematice autorizace a autentizace v souvislostech technologického vývoje. Interpretace postupného zdokonalování správců se opírají o Wilsonovou (2019) a jsou obohaceny o další články související s tématem.

4.1 Vývoj od raných počátků až po současnost

Na pomyslném počátku ověřování identit aplikace neměly implementovány vlastní metody autorizace a autentizace. Sdílení dat mezi více aplikacemi také bylo nezvyklé, proto si každá aplikace uchovávala přihlašovací údaje do své vlastní databáze. Jedna z prvních forem autentizace se datuje k roku 1961, kdy byl na MIT[7] s ohledem na potřeby spravování přístupu ke specifickým aktivům vyvinut systém pro sdílení dat. Ten byl ale limitován tehdejšími technologickými parametry. Každý přihlášený uživatel měl nastaven míru využití CPU a paměti na tamějších MIT počítačích řady IBM 709. Omezení přístupu bylo čistě z praktických důvodů a týkalo se technického zázemí. Můžeme však najít určitou podobnost s dnešními standardy řízených přístupů, které s omezením přístupů pracují, byť již v jiném kontextu. Ranými pionýry systémů řízeného přístupu, u kterých se ustálil standard autentizace, byly protokoly Kerberos a LDAP (Gori, 2021).

Metodika raných správců řízeného přístupu pak mohla narazit na významný problém otázky zpracování většího množství dat. V okamžiku, kdy aplikací bylo více a kdy bylo ke každé aplikaci nutné projít procesem ověření identity, tedy kontrolou shody mezi heslem a přihlašovacím údajem, stala se situace z hlediska bezpečnostní práce s přihlašovacími údaji dlouhodobě neudržitelnou. Problém se dotýkal hlavně velkých organizací, kterým se takové řešení záhy vymstilo a které doplácely na nevýhody izolovaných procesů ověřování, s nimiž se doposud pracovalo. Problémy spočívaly v nesynchronizovaných datech mezi jednotlivými aplikacemi. Za této situace byl také kladen velký tlak na uživatele, aby si pamatovali rozdílné přihlašovací údaje vždy pro

[7] Massachusettský technologický institut (Massachusetts Institute of Technology), jedná se o americkou univerzitu známou pro svoje technologické a mezioborové zázemí. S MIT jsou spojovány např.významné osobnosti fyziky Richard Feynman či informatiky Richard Stallman.

každou aplikaci zvlášť. Uživatelé se tak mohli uchylovat ke špatným návykům, při nichž využívali stejné přihlašovací údaje do více aplikací. Tím však ohrožovali bezpečnost vlastních osobních údajů. Takový způsob správy ověřování identit, při němž docházelo k decentralizaci údajů a kdy nebylo jednoduché získat kompletní přehled o uživatelích, ztěžoval možnosti konzistentního zpracování osobních údajů. Barker (2020) upozorňuje, že izolováním dat od širší infrastruktury vede k nadbytečným úkonům, které ztíží procesy ověřování a znemožňují dynamičtější práci s účty.

Postupně se tak přecházelo k nutnému propojení adresářských služeb s databázemi schraňujícími přihlašovací údaje uživatelů. Pomalu se měnil přístup ke spravování identit a údajů o uživatelích. Přecházelo se na validaci údajů prostřednictvím databáze a k uchovávání údajů na jednom místě v jedné databázi. Centralizace dat doznala výrazných změn, které převyšovaly svými kvalitami předešlý izolovaný přístup k aplikacím. Mariani (2022) klade důraz na upuštění od izolovaných dat a vyzdvihuje centralizaci dat jako jednu z charakteristik moderního uchovávání dat. K dalším výhodám uchování dat na jednom místě patřilo omezení chybovosti v datech. Sjednocení uživatelských přihlašovacích dat tak bylo přirozenější z hlediska jejich spravování. Přineslo to ulehčení směrem k celostním úpravám dat o uživatelích. Data jsou tak méně náchylná k chybám či duplicitám. Ovšem i novější přístupy s sebou nesly určité nevýhody. Ty se pak staly klíčovými pro další změny a posunuly vývoj o píď dál. Jmenovitě se jednalo o stanovení délky doby, po kterou bude uživatel v aplikaci přihlášen, tzv. session. Také bylo nutné vyřešit sdílení přihlašovacích údajů mezi jednotlivými aplikacemi.

S příchodem raných verzí jednotného přihlašování již hrálo roli přesměrování na servery, na kterých se samostatně ověřovaly identity přihlašovaného. Už nebylo třeba se znovu ověřovat, pokud byl uživatel již dříve serverem autentizován. Usnadnila se tím implementace procesu ověřování, neboť bylo možné konfiguraci provádět pouze na jednom místě, a k samotným heslům měl přístup jen server, jenž zodpovídal za jejich ověřování. Tím se vyhnulo sdílení hesel mezi aplikacemi. Nevýhody ale s sebou nesly i tyto rané SSO[8] servery. Ty spočívaly ve využívání tehdejších cookies, které omezovaly přihlašování pouze na jednu doménu. To šlo proti zájmům filosofie aplikací, které se začínaly vyvíjet podle modelu SaaS[9]a od ustálených metod se odkláněly. Aplikace tak

[8] Single-sign-on, čili jednotné přihlašování. Metoda, kdy se přes autentizační autoritu ověříme pouze jednou, a do aplikací, které sdílejí stejnou metodu ověřování, nemusí proces znovu opakovat.

[9] Software as a Service, doslova software jako služba, je taková aplikace, ke které se přihlásíme kupříkladu pomocí cloudu. Ustálenou metodu je míněna např. instalace softwaru na cílové zařízení, na kterém se pak se

primárně nemusely být instalovány na počítačích nebo na serverech, ale byly již připravené k využívání prostřednictvím cloudových služeb a webového rozhraní, k nimž vedly přístupy z různých domén. Správa se díky tomuto novátorskému modelu v několika ohledech zjednodušila a prosazování cloudových služeb přineslo do vývoje aplikací revoluci (IBM, 2020).

Microsoft přinášívlastní pohled na SaaS a razí model S+S. Model lze brát jako kombinaci obou přístupů, tedy „tradičního" pojetí s moderním přístupem „software jako služba". S+S model, tj. služba plus software, počítá s instalací aplikace na cílové řízení a také s využíváním softwaru jako služby. Jedná se o důležitou kombinaci, která maximalizuje možnosti SaaS (Laird, 2008).

Nový přístup k aplikacím SaaS ovlivnil vývoj správců identit. Vlna zájmu o využívání aplikací uložených na cloudových serverech vzrostla, jak píše Wilsonová (2019, s.22). Bohužel do těchto externích služeb nebylo možné implementovat tehdejší způsob jednotného přihlášení a vracela se opět nutnost pamatovat si pro každou cloudovou aplikaci unikátní přihlašovací údaje. V roce 2005 byl však vytvořen nový protokolový standard, který zajistil jednotné přihlášení a federativní identitu. Ty pak bylo možné aplikovat napříč různými doménami. Byl jím právě SAML 2.0.

SAML 2.0 přinesl samostatné ověření údajů mimo aplikaci, velkou výhodou také bylo umožnění federativní identity. Proces přihlášení se skládal ze dvou kroků. Z autentizace a z autorizace. Když se do aplikace nebylo možné přihlásit s federativní identitou, musely se kroky přihlášení vyřešit v samostatné aplikaci. Aplikace také musela mít samostatný proces oprávněného přístupu k přihlašovacím údajům, dále bylo potřeba myslet na případy zapomenutého hesla či jeho obnovy. Nastávala rizika opakovaných hesel, která jsou náchylná pro odcizení či uhádnutí útočníkem (EDUID, 2022). Federativní identita zvyšuje bezpečnost. Zastřešuje bezpečnou správu hesel a proces ověření. Je složena ze dvou poskytovatelů. Prvním je poskytovatel služeb. Jedná se o rozhraní, s nimž spravujeme konkrétní službu. Např. webové rozhraní pro e-mail nebo kalendář. Dalším je poskytovatel identit, který zodpovídá za spravování databáze, které ukládají data o uživatelích. Příklad webu, který využívá federativní identity je GitLab. Je možné si založit v GitLabu účet prostřednictvím třetích stran jako Google, Github či Bitbucket.

---

softwarem pracuje. Příkladem takové SaaS aplikace jsou e-mailové služby. U nich má uživatel zřízený účet, díky němuž se do e-mailové aplikace přihlašuje. Výhodou SaaS je také práce s velkým objemem dat.

Veškeré interní práce týkající se procesu autentizace mohl provádět poskytovatel identity a zavádění podmínek pro nastavení hesla nebo multifaktorového ověřování se řešilo na jednom konkrétním místě.

Wilsonová píše, že SAML je obtížný pro implementaci a že z důvodu velkého množství scénářů[10] je možné udělat více chyb při implementaci (Wilson, et al., 2019, s. 23). Protokol SAML 2.0 nebyl tedy finálním řešením. Bylo potřeba stanovit takový standard, který by také řešil autorizační procesy, které SAML 2.0 neřešil.

Protokoly OAuth 2.0, OpenID Connect spolu se SAML 2.0 jsou výraznými standardy v kontextu spravování identit a v oblasti autorizace a autentizace. OAuth 2.0 vnesl do oblasti autorizace výrazný prvek bezpečného volání API. OAuth 2.0 v kombinaci s OpenID Connect, protokolem, nabízí jednotné přihlášení do služeb. Wilsonová (2019, s. 26) poznamenává že tyto tři protokoly jsou nedílnou součástí moderní aplikace.

Správa identit se rychle rozvíjela. Její rozvoj se dotýkal oblastí práce s velkým objemem dat, způsobu sdílení citlivých údajů s dalšími stranami. S nastupujícími změnami v charakteristikách spravování identity lze nalézt stěžejní principy, které jsou klíčovými pro další rozvoj bezpečné práce se zpracováním identity. Těmito principy jsou:

1. soudržnost a způsob práce s daty uživatelů,
2. důslednost práce při sdílení přihlašovacích údajů, jejich skrytí před dalšími stranami v procesu autorizace či autentizace
3. dynamika procesu při přihlašování a sjednocování účtů pro přihlášení.

4.2 Autentizační a autorizační protokoly

Tato podkapitola se zabývá základní charakteristikou vybraných protokolů, především klade důraz na normy využívání, scénáře postupů a termíny, které se váží k funkcionalitě a hlavním principům protokolu. Výběr byl zohledněn vůči Keycloaku, do něhož je většina protokolů implementována. Podkapitola rozšiřuje protokoly o další současné bezpečnostní normy, které řídí proces autorizace nebo autentizace.

[10] Špatné konfigurace SAML se nevyhnul Microsoft. V roce 2016 se v programu Office 365 našla chyba, která umožňovala obcházet proces ověření. (Mimoso, 2016).

4.2.1 Kerberos, LDAP a předehra k moderním protokolům

Kerberos představoval interní projekt MIT, se kterým byla veřejnost seznámena až v 80. letech, jeho nejvyšší verze je z roku 1993. Gori (2021) jej označuje za předchůdce SSL/TLS. V období 90. let a počátku nultých let významně zatěžovaly CPU kryptografické operace, kvůli nimž spojení pomocí TLS nemohla fungovat (Gori, 2021). Tento problém byl vyřešen až s příchodem protokolu Kerberos. Kerberos slouží jako nástroj pro autentizaci požadavků, jehož prostřednictvím lze distribuovat výpočetní prostředí a jehož metoda šifrování je založena na algoritmu DES. Proces ověřování probíhá prostřednictvím metod výměny tiketů, díky kterým není nutné posílat heslo přes síť. Vlastní autentizační server spravuje databázi uživatelů a jejich tajných klíčů (Pužmanová, 2009, s. 509).

LDAP je binární protokol, jenž spravuje přístupy uživatelů na adresářovém serveru, do něhož je též začleněn proces ověřování totožnosti uživatele pomocí hesla. Aktiva nejsou členěna do tabulek, jejich datový konstrukt se spíše podobá schématu klíče a hodnoty (key - value). Takové schéma nalézáme třeba u objektu ve formátu JSON. LDAP byl v roce 1997 vydán Mezinárodní telekomunikační unií[11].

Protokoly Kerberos a LDAP svého času zaznamenaly velký zájem a nebylo systému, který neměl implementován LDAP plugin řešící autentizaci. S nastupujícími technologiemi v souvislosti s mobilními aplikacemi a API nestačily protokoly Kerberos a LDAP udržet krok s překotným vývojem a narazily na problém s renovováním přístupového systému a se spravováním identit. Zlomový okamžik nastal v roce 2000, kdy byly předloženy základy k architektuře REST, které se dodnes využívají. Problémy, které vyplynuly z technologického vývoje, se týkaly procesu ověřování, který doposud využíval k ověření pouze heslo (Gori, 2021).

Protokol OpenID jako první zavedl možnost přihlášení pomocí třetích stran za využití interního nebo externího ověřovače identity. Tento způsob přihlášení zároveň umožňoval omezit využívání množiny identit napříč různými doménami. Protokol se rychle vyvíjel díky příspěvkům od komunity programátorů či korporací. Gori (2021) uvádí organizace JanRain a NetMesh, které přispěly zpracováním protokolu do několika programovacích jazyků, respektive formátem XRDS, jenž se posléze využíval v LDAPu. S přibývajícími nadšenci rostla komunita kolem OpenID a v roce 2007 byla založena nadace OpenID

[11] International Telecom Union je agentura spravována OSN a jejich náplní jsou informační a telekomunikační technologie.

Foundation, jejímž cílem je šířit povědomí o protokolu a zlepšovat jeho technologie a standardy. Nadace je stále aktivní a nadále své služby poskytuje. Přestože OpenID ve své době standardizoval proces ověřovaní, průlom směrem k současnému způsobu autorizace a autentizace přinesly až následující protokoly.

Cílem následující části není obsáhnutí tematického okruhu zabezpečení identity v celé své šíři, nýbrž tímto výběrem protokolů, respektive protokoly OpenID Connect, OAuth 2.0 a SAML 2.0, popsat stěžejní standardy v oblasti autorizace a autentizace, které se ustálily v souvislosti služeb zabývajících se touto problematikou.

4.2.2 OAuth 2.0

OAuth 2.0 (OA) spadá do rodiny autorizačních protokolů a jeho hlavním účelem je umožnění aplikaci volání API třetích stran prostřednictvím tokenů, aniž by musela sdílet citlivé údaje o uživatelovi s třetí stranou. Dále standard zpřístupňuje jen vybraný rozsah pravomocí v omezeném čase (Wilson, et al., 2019).

K protokolu se vážou čtyři role definující zodpovědnost při autorizaci a za volání API. Jsou jimi role zdrojového serveru (resource server), na kterém jsou uložená aktiva, tj. jakákoli forma cennosti, a jeho vlastníka (resource owner), který má ke zdroji přístup. Dalším je klient, respektive aplikace, která jedná v zájmu vlastníka, a nakonec role autorizačního serveru (authorization server), k němuž má důvěru zdrojový server. Autorizační server ověřuje vlastníka nebo aplikaci a bere zodpovědnost za povolení žádostí při volání API (Wilson, et al., 2019).

Protokol zpřesňuje dva typy aplikací – utajovanou (confidential client) a veřejnou (public client). Odlišují se ve způsobu, jakým ukládají přístupová data, jak se ověřují data vůči autorizačnímu serveru či jiné ověřovací autoritě. Rozdílné je také místo, kde se spouští samotná aplikace. V případě utajované aplikace, je aplikace nasazena na zabezpečeném serveru, na němž je možné bezpečně ukládat citlivé údaje pro autorizaci. V opačném případě se aplikace spouští v uživatelském prostředí (např. mobilní telefon) nebo v prohlížeči. V případě takového spuštění však práce s přístupovými údaji není zabezpečená, a nejsou ani zajištěny prostředky pro komunikaci s autorizačním serverem (Wilson, et al., 2019, s. 56). Uvedené typy aplikací lze dále dělit na tři základní profily (jejichž charakteristika se prolíná, důsledkem čehož hranice profilů není v praxi pevně

stanovaná): webová aplikace (web application), aplikace založená na klientovi (user agent-based app) a nativní aplikace (native application). Webová aplikace je taková aplikace, která je nasazena v zabezpečeném prostředí vlastního serveru, jehož správu zajišťuje vnitřní logika (backend) a na kterém jsou ukládány ověřovací údaje. Aplikace založena na klientu běží převážně v prohlížeči uživatele. U profilu nativní aplikace je předpokladem, že aplikace je nainstalovaná nebo je součástí nějakého přístroje, např. mobilního telefonu, osobního počítače (Wilson, et al., 2019).

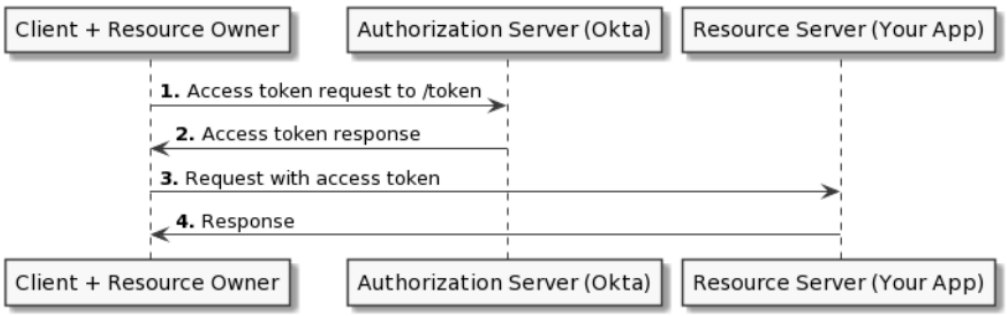
Dalšími pojmy vázanými na funkci protokolu jsou tokeny a autorizační kód (authorization code). Tokeny slouží buď k tomu, aby bylo možné vstoupit do aplikace prodloužení povolení aplikace (access token) či k prodloužení délky přihlášení (refresh token). Autorizační kód (authorization code) slouží k získání tokenů pro přístup nebo pro obnovení API relace (Wilson, et al., 2019, s. 57). Unikátní kód odlišuje čtyři druhy udělení autorizace:

1. udělení autorizačního kódu (authorization code grant),
2. bezpodmínečné udělení (implicint grant),
3. udělení autorizace vlastníkem zdroje prostřednictvím ověřeného hesla (resource owner password credentials grant),
4. udělení prostřednictvím přihlašovacích údajů (client credentials grant).

Na základě vybraného autorizačního typu může být odlišen proces autorizace od procesu udělování tokenu. Wilsonová (2019, s. 74) nedoporučuje využívání bezpodmínečného udělení nebo udělení autorizace vlastníkem zdroje prostřednictvím ověřeného hesla. V prvním případě může dojít ke kompromitování tokenů. Ve druhém případě bude mít aplikace přístup k přihlašovacím údajům, respektive heslu, což je nežádoucí z hlediska bezpečnosti. Dále Wilsonová (2019, tamtéž) upozorňuje také na udělení autorizace vlastníkem prostřednictvím přihlašovacích údajů, které je vhodné užívat v případě, kdy součástí aplikace je i zdroj, kvůli kterému se volá API. Proces autorizace u protokolu se podobá typu udělení autorizačního kódu (další varianty se odlišují způsobem přístupu stanovených rolí k přihlašovacím údajům a k jejich kompromitaci mimo zabezpečený okruh).

Proces autorizace, který je komunikací mezi aplikací a autorizačním serverem, je tvořen ze dvou požadavků. Důležitým prvkem je obdržení přístupového tokenu. První žádost je založena na přesměrování na autorizační koncový bod (endpoint), jehož

prostřednictvím si uživatel vyžádá autorizaci API volání přes povolení vlastníka. To v praxi znamená, že prohlížeč se přesměruje webovou adresu (endpoint), na kterém sídlí autorizační server. Uživatel musí projít procesem autentizace, kterým tak potvrdí souhlas s vyžádaným požadavkem, tj. aplikace získává autorizační kód. Poté autorizační server přesměruje prohlížeč zpět na adresu aplikace. Aplikace je nyní vybavena autorizačním kódem a může vyslat druhý požadavek autorizačnímu serveru. Z další webové adresy (koncový bod) získá přístupový token. Autorizační server přidělí aplikaci token, s jehož pomocí pak může aplikace využívat API volání.



[Obr. 1 Proces autorizace – služba Okta[12]]

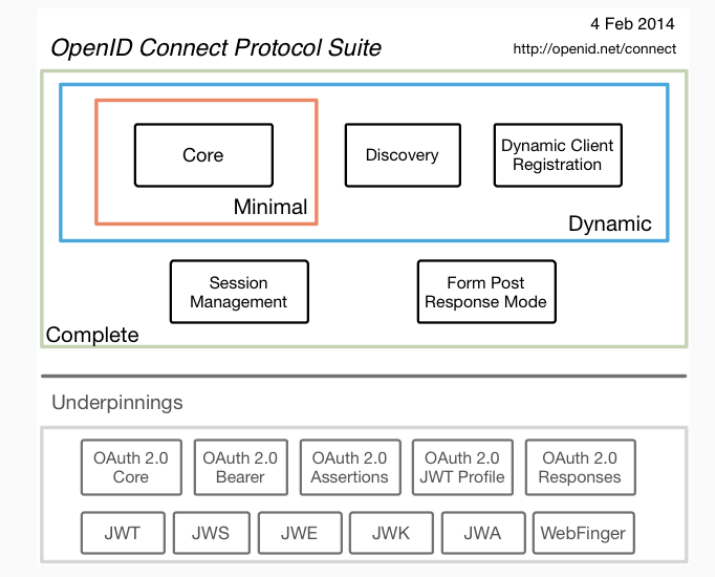
Wilsononová (2019, s. 59) dále uvádí možnost posílit zabezpečení při autorizaci tak, že se požadavek o autorizační kód rozšíří o PKCE, tzv. „pixi“ (Proof Key for Code Exchange). Tímto rozšířením pak dojde k zabránění zneužití unikátního kódu neověřenou třetí stranou. Pixi funguje tak, že si aplikace vytvoří náhodný šifrovaný řetězec o takové délce, která ztíží potencionálnímu útočníku jeho prolomení. Délka se doporučuje mezi 43 až 128 znaky, řetězec se skládá z písmen, čísel a speciálních znaků (Wilson, et al., 2019, s. 63). Tento určený řetězec se v rozšíření „pixi“, v tzv. kódovém potvrzení (code_verifier), zašle spolu s unikátním kódem a s kódovou výzvou (code_challenge) jako autorizační požadavek autorizačnímu serveru. Při vyžádání o autoriza5.ci požadavku si autorizační server kódové potvrzení zahešuje pomocí vybrané metody (code_challenge_method) a zakóduje do base64. Server nově vzniklý řetězec pak srovná s kódovou výzvou, kterou přijal v požadavku aplikace. Pokud se kódové výzvy aplikace a serveru shodnou, autorizační server požadavek povolí. Tím končí životní cyklus PKCE a začne znovu ve chvíli, kdy aplikace vyšle nový požadavek na autorizaci.

[12] Okta DEveloper [online]. 2022 [cit. 2022-07-22]. Dostupné z: https://developer.okta.com/docs/concepts/oauth-openid/#resource-owner-password-flow

OA je využíván především při autorizaci požadavků volání API a lze jej zabezpečit o rozšíření PKCE, snižuje možnost odchycení unikátního kódu neověřenou entitou. Klíčové dále je, že prostřednictvím protokolu se pracuje s tokeny, které umožňují přístup k aktivům. V současnosti se od roku 2012 dosud užívá verze 2.0. Samotnou autentizaci zprostředkovává protokol OpenID Connect, který je vrchní vrstvou protokolu OAuth 2.0.

4.2.3 OpenID Connect

OpenID Connect je třetí iterací protokolu OpenID a je koncepčně podobný protokolu OA a jeho hlavním účelem řídit proces autentizace. OIDC a OA jsou tak neodmyslitelnými partnery řešícími průběhy autentizace a autorizace. OIDC dále povoluje možnost jednotného přihlášení (Single sign-on). Ověření uživatele v rukou OIDC probíhá také přes autorizační server. Aplikace po úspěšném dokončení ověření dostává autorizační kód, přístupový token a zároveň ID token. ID token schraňuje informace o uživateli, které lze také získat z koncového uživatelského bodu (UserInfo endpoint). Dalším podstatným rozdílem je, že v OIDC je požadavek rozšířen o rozsah (scope) pravomocí uživatele. Stanoveným rozsahem se volaný požadavek ověřeného uživatele omezí pouze na vybrané dílčí části, které byly povoleny. Pro pochopení funkčnosti protokolu jsou dále popisovány pojmy, které se vážou na protokol OIDC.



[Obr. 2 OpenID Connect a OAuth – diagram[13]]

[13] OpenID [online]. 2022 [cit. 2022-07-22]. Dostupné z: https://openid.net/connect/

Protokol užívá rozdílné pojmy týkající se rolí a nastavení, avšak jsou koncepčně stále podobné s těmi stanovenými v OA. V OIDC vystupují tři aktéři. Uživatel (end user) je ten, který se nechává ověřit. Může se jednat o osobu či program, který podává žádost o ověření. Autorizační server má v OIDC jiný termín, je nazýván tzv. poskytovatelem OpenID (OpenID Provider), jeho role je založena na sdílení informace ověřeného uživatele a hlášení o stavu ověření, které odesílá aplikaci. Posledním pojmem je spoléhající strana (relaying party), kterou zastává role klienta OA. Cílem spoléhající strany je informovat poskytovatele OpenID o skutečnostech týkajících se ověřování uživatele. Typy klientů se shodují s charakteristikou v OA a nijak se v OIDC neodlišují. I v OIDC se tak dělí klienti na veřejné a utajované, a mohou se dále profilovat (Wilson, et al., 2019).

OIDC pracuje jako OA s autorizačním, tj. unikátním, kódem, a tokeny. Tokeny rozlišuje na ty, které odpovídají za přístup a za obnovu délky přístupu. Dále rozlišuje i tokeny, které nejsou součástí OA specifikace, jedná se o tzv. ID token řešící primárně stav ověřeného uživatele. ID token využívá poskytovatel OpenID k tomu, aby předal aplikaci informaci týkající se procesu ověření. ID token je kódován ve formátu JSON Web Token (JWT).

```
{
 "iss": "https://server.example.com",
 "sub": "24400320",
 "aud": "s6BhdRkqt3",
 "nonce": "n-0S6_WzA2Mj",
 "exp": 1311281970,
 "iat": 1311280970,
 "auth_time": 1311280969,
 "acr": "urn:mace:incommon:iap:silver"
}
```

[Obr. 3 ID token – JWT formát[14]]

Jedná se o JSON[15] objekt, který slouží jako sdílený informativní článek mezi dvěma stranami, skládající se ze záhlaví (header section), dat (payload section) a podpisu (signature section). Záhlaví obsahuje informace týkající se typu objektu a využití algoritmu pro podpis. Pro šifrování se využívají buďto algoritmy HS256 (HMAC s SHA256) nebo RS256 (RSA podpis s SHA256). Samotný podpis zaručuje, že data

[14] OpenId [online]. 2022 [cit. 2022-07-22]. Dostupné z: https://openid.net/specs/openid-connect-core-1_0.html

[15] Javascript Object Notation – jedná se o způsob zápisu, kde objekt je členěn dle klíče (key) a hodnoty (value).

nebyla kompromitována (integrita data). Data obsahují informace o uživateli a stavu ověření a podpis obsahuje digitální podpis složený z ID tokenu a utajovaného klíče, který je znám pouze poskytovateli OpenID (Wilson, et al., 2019, s. 81). Wilson (2019, s. 81) dodává, že je možné JWT dále šifrovat pomocí JSON Web Encryption[16] a tím i JWT více zabezpečit.

Základní proces ověřování uživatele kopíruje kroky autorizačního požadavku OA. Součástí procesu je přesměrovávání klienta, např. webový prohlížeč, na koncový bod, tj. webovou adresu, kde se daný uživatel autentizuje u poskytovatele OpenID.

Poskytovatel obdrží žádost o rozsahu pravomocí uživatele a dále uživatele prověřuje. Pokud uživatel projde ověřovacím procesem, poskytovatel OpenID za vybraných podmínek povolí uživateli přístup. Až v ten okamžik se klient přesměruje zpět na původní stránky, kde je již vybaven autorizačním kódem. Posléze klient vyšle další požadavek, jenž nyní obsahuje autorizační kód, poskytovali OpenID, kterým aplikace žádá o token. Poskytovatel OpenID vrátí aplikaci ID a přístupový token (případně obnovovací pro prodloužení přístupu). Následně se uživatel již může přihlásit, byl ověřen.

Nadcházející část se bude týkat protokolu SAML. Tento protokol má společné jednotné přihlášení s moderním OIDC protokolem.

4.2.4 SAML 2.0

SAML (Security Assertion Markup Language) je založen na XML frameworku[17] a byl navržen pro zabezpečenou výměnu informací mezi podnikatelskými partnery (Wilson, et al., 2019, s. 100). Jeho výhodou je, že umožňuje jednotné přihlášení z více domén, které protokoly OA a OIDC nenabízí, a federativní přihlášení.

Protokol definuje několik aktérů, mezi nimi i subjekt, který může projít procesem ověření. Subjektem může být samotný uživatel nebo software. SAML dále určuje tzv. SAML profil, který stanovuje, jak využívat zprávy v kontextu SAML v případech jednotného přihlašování v rámci firem. Dalšími aktéry v protokolu jsou zprostředkovatel identity (Identity provider) a poskytovatel služby (Service Provider). Zprostředkovatel

[16] JSON Web Encryption, jedná se o šifrovaný JSON objekt.
[17] Je to framework, ve kterém se vytváří XML. XML je obecný značkovací jazyk.

identity slouží jako nástroj pro ověření a zároveň pro vydávání bezpečnostní zprávy o uživateli, tzv. SAML prosazování (SAML Assertion). Role poskytovatele služby spočívá v tom, že posílá ověřovací údaje zprostředkovateli identity a zároveň má přístup k datům prostřednictvím zprávy. Pro komunikaci mezi poskytovatelem služby a zprostředkovatelem identity slouží SAML prosazování (SAML Assertion). Touto specifikou si poskytovatelé sdílí bezpečnostní zprávu o subjektu. Pro vzájemnou důvěru mezi aktéry ověření je potřeba mezi nimi nastavit vztah, ve kterém jsou oba aktéři ověřeni. Tato důvěra mezi zprostředkovatelem identity a poskytovatelem služby musí být nastavena dříve, než bude možné, aby aktéři začaly odesílat požadavky. Informace o vzájemné důvěře mezi těmito dvěma aktéry jsou známy jako metadata. Metadata obsahují URL adresu a certifikáty, jimiž je stanoven validovaný vztah (Wilson, et al., 2019, s. 101).

Poskytovatel služby si dále nastaví způsob šifrování ověřovacího požadavku (request encryption) a Wilsononová (2019, s. 108) zároveň doporučuje k požadavku dále přiřadit podpisový algoritmus (request signing). Naopak u zprostředkovatele identity se nastavují očekávané odpovědi na požadavek, opět se stanoví algoritmy pro šifrování a podpis (response signing, response encryption).

Proces ověření začíná v okamžiku, kdy si uživatel jako subjekt podá žádost o vstup do aplikace, kterou v tomto kontextu vnímejme jako poskytovatele služby. Aplikace si vyžádá po zprostředkovali identity ověření uživatele. Zprostředkovatel uživatele ověří a zašle zprávu ve formě SAML prosazování (SAML Assertion) poskytovateli služby. Ve zprávě, kterou Wilson (2019, s. 101) popisuje jako bezpečnostní token, jsou popsána data o ověřeném uživateli. Samotná role zprostředkovatele se může v některých případech změnit na roli poskytovatele služby, to nastane ve chvíli, kdy se proces ověřování přesouvá na externí službu.

4.3 Životní cyklus spravování identity

V rámci práce je definován termín identity dle Wilsonové (2019), která jej obecně popisuje jako množinu rysů, které jsou spojeny s konkrétní osobou či entitou ve vymezeném kontextu. Údaje o identitě konkrétní osoby jsou kupříkladu jméno, adresa či věk. Entitu lze naopak vnímat v rovině nějakého nástroje, buď ve formě aplikace,

mobilního zařízení či bota[18], jehož identita je složena kupříkladu z IP adresy, modelu či sériového čísla. Identita je tak nedílnou součástí procesu autorizace a autentizace a je v této souvislosti přimknutá k uživatelskému účtu, jehož pomocí může jeho vlastník provádět vybrané akce v prostředí konkrétní aplikace. Každá identita může mít více než jeden účet, každý účet je naopak vždy vázán na jednu konkrétní identitu a nemůže se tedy stát, aby specifický účet vlastnilo více identit.

Správce identity je tedy nástrojem, díky němuž lze zpracovávat množinu identit, u nichž je potřeba řídit ověřování identity, odsouhlasení požadavků a vedení účtů. Wilsonová (2019, str. 10) klade důraz na klíčový rys, kterým je ochrana před nepovolenými vstupy. Tyto nástroje jsou dle ní (Wilson, et al., 2019) neopomenutelnou součástí zabezpečení našeho soukromí.

Po představení, čím je správce identit a identita samotná, se kapitola dále zaměřuje na cyklus spravování identity ve správci identit. Celý cyklus dle Wilsonové (2019, s. 11) se dělí na 10 následujících částí.

Registrace (provisioning)

Vše začíná tvorbou účtu s unikátní identifikačním znakem, který je propojen s konkrétní a nezaměnitelnou identitou registrovaného uživatele. Nový účet je buď zcela nově vytvořený nebo se může jednat o již vytvořený účet, jenž se využije v novém prostředí, např. při změně banky či operátora.

Schvalování požadavků (authorization)

V průběhu vytváření účtu se udělí rozsah dostupných akcí, s nimiž by mohl daný uživatel pracovat, a přístup k neveřejným datům, pro které by mělo být uděleno rozpětí povoleného čtení či manipulace s daty. Např. pravomoci mezi účty běžného a správcovského účtu ve Windows 10 Home se odlišují v možnostech instalací nebo užívání určitých programů, např. správce může instalovat všechny možné aplikace, běžný uživatel může instalovat určený výběr. Udělování rozsahu pravomocí by měla být na zodpovědnosti vybrané autority či správce daného systému.

[18] Je myšlen počítačový program.

Ověření identity (autentizace)

V zájmu ochrany aktiv je potřeba, aby identita byla řádně ověřena a spárována s konkrétními přihlašovacími údaji. Ověřením tedy uživatel prokáže, že má vstupní právo na účet. Wilsonová upřesňuje (2019, s. 12) obsah přihlašovacích údajů tak, že se jedná o prostředky, které uživatel zná, vlastní a kterými je. Tím se míní to, že identitu lze ověřit heslem, kterou zná pouze vlastník účtu. Dále se lze identifikovat nástrojem, který vlastní uživatel účtu, kterým lze vygenerovat nějaký kód. Kupříkladu po potvrzení bankovního převodu, dostane uživatel SMS zprávu s kódem na své mobilní číslo, které pak vpíše do aplikace pro potvrzení své identity. Poslední metoda autentizace může být ve formě biometrického údaje, což je fyzická identifikace vlastníka účtu. Identifikace se provádí pomocí otisku prstů či fotky uživatele. Příklad využití těchto tří metod pro identifikaci uživatele lze nalézt např. u tzv. zesíleného zabezpečeného ověřování. Jedná se o vícefázové ověření, také jako multifaktorové (multi-factor authentication), a zesílené ověření (step-up authentication).

Vícefázové ověřování se skládá z několika kroků pro ověřování uživatele. Běžným způsobem je dvoufázové ověření. Spočívá v kontrole hesla a potvrzení unikátního kódu, který dostaneme prostřednictvím mobilního telefonu. Zesílené ověření, také jako dodatečné zesílení ověření, nastane ve chvílích, kdy uživatel vyšle požadavek, který musí také potvrdit opětovným ověřením. Příkladem budiž uživatel, který je přihlášen na bankovním účtu a požaduje změnu nastavení na svém účtu, potvrzení o změně provede vybraným ověřením. V tomto případě uživatel získá jednorázové heslo od banky, kterým pak v aplikaci ověří svou identitu.

Wilsonová (2019, s. 15) tak shledává ověření ve formě hesla a přihlašovacího jména jako slabší formu ověřování s ohledem na to, že ověření sleduje pouze jeden faktor kontroly. Součástí autentizace je také standard jednotného přihlášení, jehož předpokladem je ověření u stejného poskytovatele ověřovací služby. Jednotné přihlášení zprostředkovává možnost projít procesem ověřování pouze jednou a není potřeba opětovně zadávat přihlašovací údaje. Po uplynutí stanoveného času je potřeba myslet i na odhlašování.

Vynucení rozsahu přístupu (Access Policy Enforcement)

Po ověření a přihlášení uživatele je důležitým zabezpečením schvalování požadavků. Kontroluje se, zda se požadavky shodují s rozsahem pravomocí nastavených u účtu. Pokud tomu tak bude, požadavek se autorizuje a žádost se provede.

Délka přihlášení (session)

Bezpečnost přístupu ke konkrétní aplikaci zajišťuje také vynucování opakovaného ověřování zda se jedná o přihlášeného uživatele K ověřování dochází vždy po určité, předem stanovené, době po přihlášení uživatele do aplikace. Tímto neustálým ověřování dochází k zamezení možnosti využívání účtu neautorizovaným a neověřeným uživatelem. Stanovení délky, kdy uživatel zůstává přihlášen je důležitým prvkem bezpečné ochrany účtu.

Odhlášení

Způsoby odhlašování se dělí na ty, které jsou iniciované uživatelem, a ty, jenž jsou způsobené skončením povolené délky přihlášení. Odlišují se zejména ve způsobu rozvržení procesu odhlašování. Odhlášení při ukončení povolené délky přihlášení je způsobené tím, že se s účtem dál nepracuje. Dochází k němu samovolně uplynutím stanovené doby. Ve druhém případě je odhlášení vědomou volbou uživatele, který musí úkon odhlášení přímo provést. V takovém případě je odhlášení platné ihned po provedení úkonu. Zároveň, má-li uživatel přihlášen účet přes jednotné přihlášení, odhlášením z jedné aplikace se automaticky odhlásí i z aplikací ostatních. V obou případech je po odhlášení nutné znovu projít procesem přihlašování.

Spravování a obnova účtu

Uživatel by měl mít ve správě identity, tj. v aplikaci, možnost úpravy vybraných údajů. Obvykle se nejčastěji jedná o možnost resetování hesla či změny osobních údajů. V průběhu těchto změn by mělo také docházet ke kontrole identity. Wilsonová (2019, s. 16) uvádí jako příklad změnu hesla, k níž může dojít v případě ztráty nebo resetování hesla. V takovém případě je třeba nechat si alternativním způsobem potvrdit vybrané úkoly.

Smazání účtu (deprovisioning)

Smazání účtu patří mezi základní možnosti správy účtu. Jeho smyslem je účet zcela uzavřít takovým způsobem, kdy je účet deaktivován a nelze jej dále užívat. Údaje týkající se účtu jsou však archivovány pro další využití.

4.4 Současní správci identit

V následující podkapitole je srovnávána trojice současných open-source správců identity. Ke srovnání byly vybrány projekty od firem Gluu (Gluu Inc, 2022) a Ory (Ory Corp, 2022), jež jsou srovnávány s nejnovější verzí Keycloaku (Red Hat, 2022). Konkrétní výběr Gluu a Ory byly vybrány z důvodů, že se jejich projekty těší velké oblibě u komunity vývojářů a jejich repozitáře jsou často aktualizovány.

Častá aktualizace, počet aktualizací a četnost vydávání nových verzí projektů značí, že se jedná o živý projekt, který je aktivně spravován a který se neustále vyvíjí. V repozitáři Keycloaku (Red hat, 2022) se lze v záložce Milestones dočíst, že se v dohledné době plánuje vydání verze 20. Projekt má na kontě přes 1 000 vyřešených issues a letos byly dosud od nového roku vydány čtyři nové verze. V repozitáři Gluu jsou u projektů oxTrust či oxAuth (Gluu Inc, 2022) vypsány plánované dílčí verze a projekty se nevyznačují tak vysokou míru aktivity jako je tomu u Keycloaku. Na druhou stranu lze i v repozitáři Gluu sledovat aktivitu oprav a počet vyřešených issues je srovnatelný s Keycloakem. Ory je z této trojice nejmladším projektem. Zatím nebyla vydána plnohodnotná verze 1, dostupná je ale verze 0.10. Projekt Ory má však aktivní základnu vývojářů a konkrétně u projektu kratos vyřešených přes 700 issues, výhledově se jedná o nadějný projekt.

Stěžejním důvodem pro jejich výběr byla také přehledná a dostupná dokumentace.

Vybraní správci identit jsou srovnáváni na základě bodů:

1. Jaké standardy využívají v rámci autentizace nebo autorizace
2. Co obsahují repozitáře jednotlivých projektů
3. Možnost instatní aplikování nástroje
4. Dostupnost API

4.4.1 Keycloak – Keycloak v19

Keycloak je tzv. open-source správce identity a přístupu (Identity and Access Management) nabízející služby jednotného přihlášení, federativní identity a REST API. Vývoj Keycloaku začal v roce 2014 a zaměřoval se na zabezpečení aplikací pro vývojáře. Jeho vývoj byl převážně zaštítěn firmou Red Hat, která následně v roce 2016 projekt opustila a zaměřila se na komerční produkt RH-SSO řešící jednotné přihlašování. Keycloak letos dosáhl verze 18 díky silné podpoře komunity vývojářů. Thorgersen (2021) píše, že jej lze použít pro menší projekty s malým počtem uživatelů či pro projekty větších korporací s milióny uživateli. Nabízí možnost stylování přihlašovací stránky, nastavení silného ověřování a velký záběr připravených programovacích toků.

Keycloak má kupříkladu vlastní řešení pro obnovu hesla, podněcování uživatelů k aktualizaci hesla nebo k potvrzení smluvních podmínek. Součástí Keycloaku jsou standardy pro ověřování a ukládání citlivých údajů o uživatelích. Aplikace, do nichž bude začleněna funkcionalita Keycloaku, si nebudou na své straně ukládat hesla. Bezpečnostní token získají při procesu ověřování od Keycloaku, díky němuž se uživatelé přihlašují nebo posílají autorizované požadavky.

Součástí keycloaku je také jednotné přihlášení, které nenutí uživatele pokaždé se do různých aplikací zvlášť přihlašovat, a spravování délky přihlášení. Služby Keycloaku jsou založené na současných protokolech OAuth 2.0, OpenID Connect a SAML 2.0. Keycloak také obsahuje vlastní databázi, kterou lze rovnou využívat při ukládání uživatelů. Keycloak umožňuje také zapojení adresových služeb jako Active Directory nebo LDAP. Silnou stránkou je i možnost nastavit Keycloak jako poskytovatele identity, případně je možné začlenit přihlášení prostřednictvím služeb třetích stran, tj. jsou dostupné služby federativní identity. Umožnuje také nastavení tokenu.

4.4.2 Gluu – Gluu v4

Původní myšlenkou Gluu serveru bylo zjednodušení jednotného přihlášení a nabídnutí alternativy těžící z open-source konceptu. Jeho duchovním otcem je Mike Schwartz, který ranou verzi projektu představil v roce 2009 v Mnichově. Tehdejší forma Gluu serveru byla

založena na projektech SunOpenSSO a OpenDS[19], které pracovaly s méně zabezpečenými možnostmi jednotného přihlášení a propojení s adresářovými službami. S přicházejícími novými verzemi byly tyto protokoly opuštěny a implementovaly se moderní přístupy zabezpečení autorizace a autentizace. Ve svém repertoáru tak Gluu těží z protokolů SAML 2.0, OpenID Connect a OAuth 2.0. Oproti Keycloaku podporuje také protokoly CIAM[20], CAS[21], který řeší jednotné přihlašování pomocí tiketového systému, a FIDO[22], díky němuž se uživateli nabízí ověření pomocí mobilu či jiného nástroje bez využití zašifrovaného hesla.

Nástroje, se kterými pracuje Keycloak, jsou také součástí Gluu, tedy jednotné přihlášení i odhlášení, REST API, dvoufaktorové ověřování a federativní identita.

Zatímco u Keycloaku převažuje ověřování prostřednictvím hesla, Gluu umožňuje přihlášení bez hesla, respektive pomocí vybraného přístroje, např. mobilu. Přestože se oba projekty dosud vyvíjí a mají dostatečnou aktivní komunitní podporu, Gluu server má výhodu v tom, že na jeho aktualizace dohlíží samotná firma Gluu (2020).

4.4.3 Ory – v0.10.0

Ory je poměrně zajímavý open-source projekt skládající se z několika nástrojů – Kratos, Hydra, Keto, Oathkeeper - ale pro správu identity v rozsahu Gluu serveru nebo Keycloaku je potřeba tyto nástroje mezi sebou kombinovat, případně je doplnit o jinou adekvátní externí službou. Vybrané aplikace od Ory jsou schopny zajistit proces ověření. Pokud je nutné spravování uživatelských dat a mít plný rozsah využití standardu OIDC a OA 2.0, doporučuje Rekkas (2022) spíše využití služeb Keycloaku.

[19] SunOpenSSO je autentizační nástroj a Open DS je adresářová služba.
[20] CIA(Customer Identity and Access) je protokol, který je založen na IDaaS (Identity-as-a-Service) a spočívá v tom, že spravování identity řeší celkově externí služba (Poza, 2021). Staví na čtyřech stěžejních pilířích dle Pozy (2021), kterýmijsou škálovatelnost, jednotné přihlášení, centralizace spravování uživatelů a více faktorovém ověřování. Poza dále dodává (2021), že se odlišuje od ostatních SSO protokolů tím, že jeho prostřednictvím je možné rozlišovat nespočetně mnoho typů uživatelů, které je potřeba oddělit na základě úrovně zabezpečení a způsobu spravování účtu.
[21] Protokol CAS, tak jako u protokolů OIDC či OA, má stanovenou entitu – CAS server – vůči které se uživatelé ověřují. Prostřednictvím tiketového systému získává tiket, který se ukládá do prohlížeče, díky němuž se lze přihlásit „pouze jednou“.
[22] FIDO protokoly umožňují silnější ověřování identity, využívají párové klíče, tj. veřejný a soukromý klíč, respektive se jedná o asymetrické šifrování. Princip zesíleného ověřování využívá hardwarové klíče, např. mobily, na kterých lze různými akcemi – od vložení PIN kódu po potvrzením hlasem – ověřit identitu (FIDO ALIANCE).

Samotný projekt má zpracovanou bohatou dokumentaci a předkládá ukázky jak jednotlivé Ory programy implementovat do vlastních aplikací. Dle dokumentace Ory mají jednotlivé dílčí softwarové služby implementované současné standardy a disponují řadou známých protokolů řešících autorizace a autentizaci. Součástí jsou jmenovitě OAuth 2.0/2.1, OpenID Connect, FIDO2 U2F a Zero Trust Networking. Součástí je také přehled koncových bodů pro REST API, jejich struktura volání a odpovědi. (Ory Corp)

Bezpečnostní model, jak je uvedeno na stránkách dokumentace (Ory Corp), zpracovává vlastní řešení týkající se tokenů, cookies či délky sezení a zaručuje bezpečnost i před kybernetickými hrozbami typu Cross-site scripting (XSS) nebo Cross-site request forgery (CSRF). Článek od Rekkase (2022) a dokumentace kladou důraz na zvážení využití protokolů OIDC a OAuth 2.0. Předešlé ověřovací služby ve formě Gluu serveru či Keycloaku vyzdvihují na svých domovských stránkách právě tyto protokoly. Ory Corp do jisté doporučuje využívání protokolů jen pro specifické účely, byť jeho služba Ory Hydra má do sebe protokoly OIDC a OAuth 2.0 implementovány. V dokumentaci (Ory Corp, 2022) se dále akcentuje nutnost využití OIDC a OAuth 2.0 jen v rámci některých konkrétních případů. Byť se jedná o názor v rozporu s tím, co tvrdí Wilsonová (Wilson, 2019), která právě využívání těchto protokolů vnímá jako standard zabezpečení aplikace, lze tento postoj Ory interpretovat kladně. Lze jej vnímat jako snahu implementovat pouze nutné nástroje, jež plnohodnotně odpovídají rozsahům potřeb vyvíjené aplikace.

Silným jádrem Ory je práce s HTTP cookies. Cookies, tj. sušenky[23], jsou nezašifrovaným textovým řetězcem s možným nastavením délky platnosti, který je uložen ve webovém prohlížeči na počítači a kterým lze identifikovat totožnost uživatele. Tato data jsou ve velikosti několika KB a mohou být prostřednictvím webového prohlížeče mazána či může být povoleno jejich ukládání na počítači. (Pužmanová, 2009, s. 431) V dokumentaci ORY (Ory Corp, 2022) se lze dočíst, že aplikace ORY využívají cookies pro ukládání délky stavů přihlášení. Cookies se nastavují dle úrovně zabezpečení:

1. „secure“: cookies se využívá pro přenos obsahu přes připojení HTTPS[24] a omezuje kybernetické útoky typu man-in-the-middle,

[23] Od roku 2022 jsou provozovatelé webových stránek povinni dokládat seznam všech využívaných cookies a souhlas se zpracováním cookies, respektive s ukládáním specifických informací o uživatelích.
[24] Jedná se zabezpečené připojení prostřednictvím HTTPS serveru, který vyžaduje autentizaci připojujícího uživatele. Existují dvě implementace, které definují RFC dokumenty RFC-2812 a RFC-2817. Buď musí uživatel mít již ověřený a validní certifikát, nebo ve druhém případě si při připojení na stránku uživatel

2. httpOnly: cookies není dostupná, je to prevence proti útokům typu XSS[25],

3. samSite=Strict: cookies si lze zavolat jen ze stejné domény, je to prevence proti útokům typu CSRF[26].

Každá úroveň je dostupná pouze za vymezených podmínek a brání proti útokům typu XSS či CSRF. Tak jako Keycloak i projekt Ory zohledňuje problematiku CORS.

[Obr. 4 CORS chyba[27]]

zažádá o povýšení protokolu, což jinými slovy znamená, že dojde k přechodu na připojení TLS. (Dostálek et al, 2003, s. 414-415)

[25] Cross Site Scripting, napadení webové stránky. Útočník využije napadenou stránku pro další vektory útoků (KirstenS, 2022).

[26] Cross Site Request Forgery, jedná se útok, při kterém je uživatel nucen provést neplánovaný úkon na stránce, na které se ověřil (KirstenS, 2022).

[27] flaviocopes [online]. 2022 [cit. 2022-07-22]. Dostupné z: https://flaviocopes.com/cors/

5 Praktická část

V praktické části jsou popsány konkrétní nástroje, které byly využity při implementaci a vývoji aplikace. Jedná se o prototyp aplikace, jehož primárním účelem je schraňování informací o klientech. V aplikaci lze přidávat, upravovat a mazat jednotlivé klienty prostřednictvím uživatelského účtu. Pro přístup do aplikace je potřeba se přihlásit a mít konkrétní přístupová povolení. Cílem bylo prostřednictvím aplikace poukázat na bezpečnostní prvky související s principy kybernetické bezpečnosti, které je potřeba zohlednit při využívání externí služby poskytující správu řízeného přístupu, tj. správce identity. Vývoj byl zaměřen na frontend část aplikace. Primárním nástrojem, který byl v projektu implementován, je server Keycloak. Ten zajišťuje proces ověřování uživatele a zároveň je správcem identity. Server Keycloak a Hasura cloud byly nasazeny prostřednictvím cloudové služby Heroku. Výběr dalších nástrojů a důvody jejich výběru a účel budou popsány v kapitole.

5.1 Svelte + Sveltekit

Svelte je frontend framework a Sveltekit je jeho nadstavba, která je stále ve vývoji a nedosáhl zatím verze 1. Sveltekit je framework, který zajišťuje komplexnější formu přesměrování a je do vštěpen moderní principy dnešních webových aplikací, které řeší server side rendering či static page rendering.

Dá se připodobnit Svelte byl vydán v roce 2018. Jeho duchovním otcem je Rich Harris. Silnou stránkou Svelte oproti jiným frontend knihovnám jako React či Angular je reaktivita dílčích komponent, které velmi svižně a dynamicky mění svůj obsah. Nad názvem[28] se lze pozastavit a zvážit, zda framework je opravdu z pohledu kódu „štíhlý“. Aplikace ve Svelte je složena z komponent, které obsahují HTML, CSS a Javascript kódy. Jednotlivé části kódu se ničím neodlišují, kód je psán klasicky. Tyto komponenty pak Svelte zpracovává do dílčích Javascript modulů. Stěžejním momentem je ale způsob, lépe řečeno, moment, kdy se kompiluje balíček kódu pro zobrazení ve webovém prohlížeči. Ke kompilaci dochází hned na serveru. Klient ve formě webového prohlížeče dostane již

[28] Svelte znamená štíhlý.

zpracovanou kompilaci a nemusí balíček dál zpracovávat. Jedná se o jeden z klíčových rozdílů, se kterým přišel Rich Harris.

```
<script lang="ts">
    import {
        Column,
        Content, Grid,
        Header, HeaderGlobalAction,
        HeaderNav,Row
    } from "carbon-components-svelte";
    import UserAvatarFilledAlt from "carbon-icons-svelte/lib/UserAvatarFilledAlt.svelte";
    import OdhlasenyUzivatelModal from "./modaly/OdhlasenyUzivatelModal.svelte";

    let jeOdhlasen = false;
</script>

<Header company="BP-CM" platformName="Keycloak" href="/">
    <HeaderNav>
            <HeaderGlobalAction aria-label="Odhlaseni" icon={UserAvatarFilledAlt} on:click={() => jeOdhlasen = true}/>
    </HeaderNav>
</Header>

<Content>
    <Grid>
        <Row>
            <Column>
                <slot/>
            </Column>
        </Row>
    </Grid>
</Content>

<OdhlasenyUzivatelModal bind:jeOdhlasen/>

<style>
    :global(.bx--header) {
        display: flex;
        justify-content: space-between;
    }
</style>
```

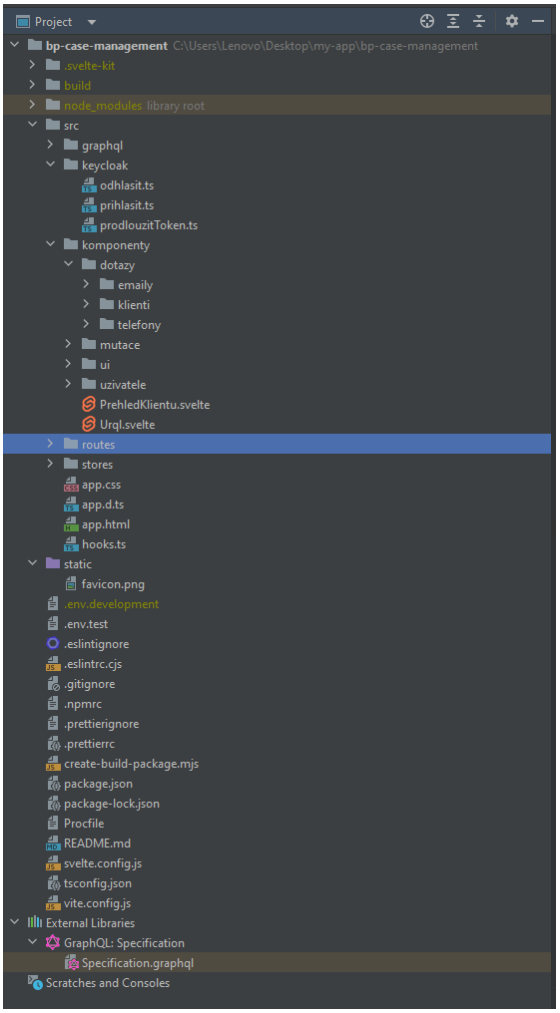
[Obr. 5 Příklad komponenty UIShell ve frameworku Sveltekit]

Projekt byl vytvářen dle vzorové struktury, kterou určuje Svelte[29]. V hlavním zdrojovém souboru *src* je rozdělen do složek, které se zaměřují na určitou část. Složka graphql obsahuje konfigurační soubory pro nastavení nástroje GraphQl. Další složkou je *keycloak*, která se skládá z funkcí týkající se přihlašování, odhlašování a prodloužení sezení. Složky *komponenty*, *routes* a *stores* jsou klasické složky Svelte. Komponenty se skládají z dílčích částí. Aplikace je složena z pětice základních komponent: modal, formulář, záhlaví, přehled uživatelů a Urql. Modaly jsou vyskakující okna, ve kterých lze upravit, smazat nebo přidat klienta. UIShell komponenta zobrazuje záhlaví stránky a zpřístupňuje tlačítko pro odhlášení. V komponentě *PrehledUzivatelu* se zobrazuje data

[29] Každá komponenta ve frameworku Svelte má příponu „.svelte“.

z databáze, které lze editovat. Aplikace dále odlišuje dva stavy, přihlášení s tokenem. K tomu slouží komponenta Urql, která zastřešuje funkce pro posílání požadavků a prodlužování délky přihlášení.

Každá Svelte komponenta se skládá alespoň ze tří částí. První část se týká kódu Typescript[30], který je psán ve script tagu. Do něho se vpisují funkce a proměnné, které řeší funkcionalitu komponenty. Druhá část se skládá z HTML kódu[31] či z vložených dalších Svelte komponent, které byly vytvořeny. Poslední část kódu, která může být součástí, je stylování, které se píše do style tagu. V projektu se stylování pomocí CSS buď píše takto samostatně či se píšou jako inline CSS. Převážně se využívaly komponenty z knihovny carbon-component-svelte.



[Obr. 6 Struktura projektu]

[30] Bude vysvětleno v následující podkapitole. Jedná se o Javascript, který je obohacen o typování vstupních a výstupních proměnných.
[31] Píše se HTML kód, která se nijak neodlišuje. Dají se vkládat tagy pro paragrafy, záhlaví či pro obrázky. Svelte každý z HTML tagů dokáže správně zpracovat.

5.2 Typescript

Typescript je rozšířením jazyka Javascript o statické typování, třídy a rozhraní. Na rozdíl od Javascriptu umožňuje datové typování, které může preventivně zabránit případným chybám. Kvapil (2018) popisuje, že v programování rozlišujeme dva způsoby typového systému. Jedná se o dynamický a statický typový systém. U dynamického typového systému není třeba deklarovat typ. Samotná typy si daný programovací jazyk určí vnitřně sám, proto se může stát, že do jedné proměnné můžeme vkládat jednou číslo, podruhé text. Příkladem takového programovacího jazyka je právě Javascript. Na druhou stranu statický typovací systém přísně dodržuje definování datových typů, které se u proměnné v průběhu konání funkce nemění (Kvapil, 2018). Výhodou je, že před spuštěním souboru/funkce dochází k datové kontrole. Ve chvíli, kdy typy neprojdou kontrolou, samotný soubor se nespustí. Právě díky Typescriptu jsou schématické vlastnosti objektu a jeho typy dobře viditelné.

V projektu je Typescript využit pro psaní funkcí, které se týkají přihlašování, odhlašování a prodlužování tokenu. Tyto jednotlivé funkce využívají funkce fetch pro volání REST API, které nabízí server Keycloak. Dále je tu funkce pro stažení schématu, které se využívá pro nástroj Hasura. Další soubory jsou buď vygenerované soubory, které slouží pro načtení dat prostřednictvím nástroje GraphQL / Urql a končí na příponu „gql.ts“, nebo se jedná o soubory, které souvisí s funkcemi Sveltekit, např. „hooks.ts“ a „app.d.ts“.

Přiložená ukázka demonstruje využití typování při definování typu chyby.

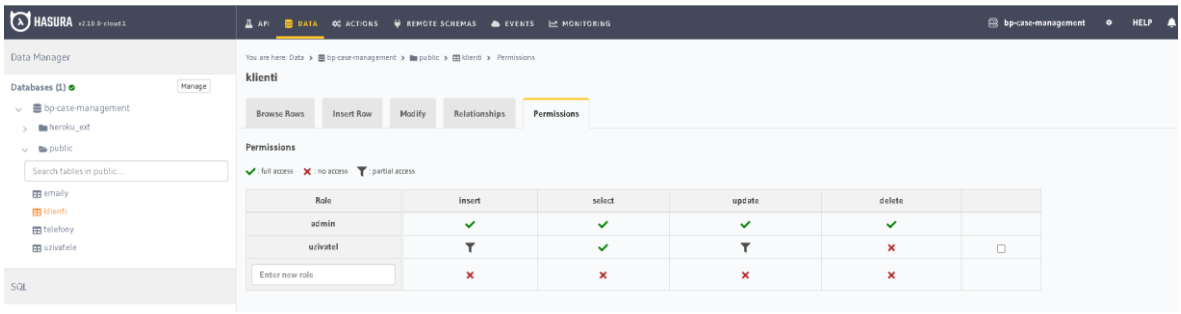
```
let odpoved;
try {
    odpoved = await fetch(logoutUrl, init: {
        method: "POST",
        body: new URLSearchParams({
            client_id: clientID,
            refresh_token: refreshToken
        }))
    })

    if (odpoved.ok && typeof window !== "undefined") {
        localStorage.removeItem(nazevTokenu);
        jePrihlasenStore.set(false);
        await goto( url: "/odhlaseni");
    }
} catch (e) {
    if (e instanceof Error) {
        console.error("Chyba", e)
    } else {
        console.error("Neznámá chyba.", e)
    }
}
```

[Obr 7. e instanceof Error]

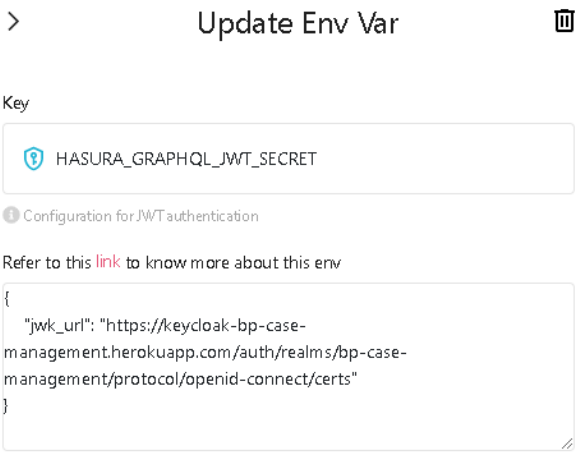
5.3 Hasura

Hasura je open-source rozhraní, které nabízí REST API a GraphQl API pro přístup k datům do databáze (Hasura, 2020). Právě prostřednictvím Hasury můžeme posílat GraphQL dotazy, které vrátí data z databáze. Ve webovém rozhraní Hasury lze pak dále nastavovat jednotlivé role či autorizační vlastnosti pro každý volaný dotaz. Konkrétně v projektu je možné zobrazit data a upravovat je pouze v roli „uživatel“.



[Obr 8. Datový manažer Hasury – konzole]

Služba Hasury také nabízí nastavení proměnných týkající se JWT tokenu, které bylo potřeba propojit se službou Keycloaku Pomocí této proměnné Hasura může zkontrolovat, zda token, který je součástí hlavičky požadavku obsahuje atribut týkající se požadované role „uzivatel“.



[Obr. 9 Nastavení proměnné pro spárování Keycloaku s Hasurou]

5.4 GraphQL + Urql

GraphlQL se řadí mezi dotazovací jazyky pro API a byl veřejně přístupný od roku 2015 (Lardinois, 2018). Jazyk byl vyvíjen firmou Facebook a je zaštíten GraphQL Foundation. Jeho největší výhodou je, že prostřednictvím API rozhraní po poslání požadavku, dorazí přesně data, která byla v dotazu vypsána. Rozlišuje tři typy dotazů: query, subscription a mutation. V projektu se využívají query po zobrazení dat, pro jednotlivé úpravy dat se využily požadavky typu mutation. Subscription slouží pro zobrazení dat v reálném čase. Pro dotazy typu mutation je výhodné také typovat přijímané proměnné a vyhnout se tak dalším možným chybám.

```
mutation pridatKlienta (
    $krestni: String!
    $prijmeni: String!
    $rokNarozeni: numeric!
    $narodnost: String!
    $telefonniCislo: String!
    $email: String!
) {
    insert_klienti_one(object: {
        krestni: $krestni
        prijmeni: $prijmeni
        rokNarozeni: $rokNarozeni
        narodnost: $narodnost
        telefony: {data: {telefonniCislo: $telefonniCislo}}
        emaily: {data: {email: $email}}
    }) {
        id
    }
}
```

[Obr 10. mutation – přidání klienta]

```
query klienti {
    klienti {
        id
        krestni
        prijmeni
        telefony {
            telefonniCislo
        }
        emaily {
            email
        }
        narodnost
        rokNarozeni
        createdAt
        updatedAt
    }
}
```

[Obr. 11 query – zobrazení klientů]

Urql je GraphQL klient, který je možné také propojit s prostředím frameworku Svelte. Pro možné volání ve frontendu aplikace je potřeba nastavit kontext, k tomu slouží Svelte komponenta Urql, ve které se volá Urql metoda initContextClient. Samotný kontext klienta je možné dál nastavovat. V projektu se nastavuje metoda authExchange, která zpracovává proces autentizace a autorizování požadavků. Metoda addAuthToOperation přidává k GrahpQL dotazům hlavičku s uživatelovým tokenem. Pokud se blíží expirace tokenu, tak v metodě getAuth se zavolá funkce pro prodloužení tokenu. (Formidable, 2022)

```
let upravenyRokNarozeni;

const kontextKlienta = getContextClient();

function upravitRokNarozeni() {
    const unsuscribe = mutationStore({
        client: kontextKlienta,
        query: upravitRokNarozeniDocument,
        variables: {
            id,
            rokNarozeni: Number(upravenyRokNarozeni),
        },
        context: {
            fetchOptions: {
                headers: {
                    "x-hasura-role": "uzivatel"
                }
            }
        }
    }).subscribe(async (odpoved) => {
        if (!odpoved.error) {
            odpoved.data?.update_klienti_by_pk;
        }
    });
    unsuscribe();
}
```

[Obr 12. Příklad volané mutace a získání kontextu]

5.5 Keycloak

Způsob přihlášení, které nabízí server Keycloak nebyl v projektu využit. Namísto toho se využilo rozhraní REST API. Bylo tak možné získat tokeny, které umožnily vyřešit vlastní způsob přihlášení, odhlášení a prodloužení tokenu. Tokeny se ukládají do místního uložiště v prohlížeči pod názvem „keycloak-token“. Textový soubor si ukládá access token, refresh token a termín expirace tokenu.


```
  "https://hasura.io/jwt/claims": {
    "x-hasura-default-role": "uzivatel",
    "x-hasura-user-id": "97f8a566-ff5d-45ad-ad24-
38f7ee9a8f16",
    "x-hasura-allowed-roles": [
      "uzivatel"
    ]
  },
  "name": "Long Tester",
  "preferred_username": "tester",
  "given_name": "Long",
  "family_name": "Tester",
  "email": "long.do@email.cz"
}
```


[Obr. 13 Úsek z tokenu - access token]

5.6 Zpracování a výsledek

Praktická část práce se věnuje využití služby Keycloak ve webové aplikaci. Cílem práce bylo využití služby autentizace bez zatížení na detaily týkající se vizuálního ztvárnění aplikace. Prostřednictvím Keycloaku bylo možné získat přístupový token, se kterým se dále pracovalo. Token byl uložen do prohlížeče a mohl tak být sdílen mezi jednotlivými dílčími částmi aplikace. Token byl využit při vkládání do hlaviček požadavků pomocí klienta Urql. Token dále obsahoval parametr, díky kterému byl nástroj Hasura schopný rozeznat, zda je oprávněn určité části dat zpřístupnit a umožnit jejich manipulaci.

Tímto došlo rozkrytí důležité vlastnosti. Je praktické, pokud je možné jednotlivé služby propojit prostřednictvím REST API či přídavných knihoven. Bohužel je obtížné najít správnou kombinaci bezpečnostního zajištění, aby nedošlo ke kompromitaci dat. Během vývoje bylo čerpáno z doporučení, které poskytovaly dokumentace jednotlivých služeb. Předně se ukázalo, že ne všechny bezpečnostní kroky je schopen zajišťovat frontend, ale pro zajištění dostatečné bezpečnosti je třeba jeho spolupráce s backendem. Ten zpracovává proměnné, které se týkají hesel umožňujících přístup do databází či dalších faktorů, které by neměly být rozkryté ve frontendu.

6 Závěr

V této bakalářské práci byly zmapovány dnešní standardy v oblasti autentizace a autorizace. Ukázalo se, že velká část autentizačních aplikací využívá současné protokoly jako jsou SAML 2.0, OpenID Connect či OAuth 2.0. Všechny tyto protokoly jsou implementovány v Keycloaku. Na oficiálních stránkách Keycloaku lze také sledovat časté aktualizace týkající se rozšiřování o další možné certifikované protokoly. Důležité tedy je, zda je jejich implementace správně naprogramována či nikoliv. Případně zda má vybraná služba dostatečné zázemí pro kvalitní rozvoj. Keycloak je částečně podporován firmou Red Hat, ale k jeho výraznému rozvoji přispívá open-source komunita.

Hlavními znaky moderního open-source správce identity jsou implementace současných protokolů, které řeší převážně způsob autorizace a autentizace. Ve vybraných službách včetně Keycloaku převažovaly protokoly OpenID Connect a OAuth 2.0. Wilsonová tyto protokoly také označuje za standardní výbavu správce identity (Wilson, et. al 2019, s. 26). Klíčové pro protokol OA je to, že poskytuje autorizaci pro API. Protokol OIDC je významný kvůli poskytování autentizačních služeb a je podporou pro OA. Protokol SAML 2.0 je důležitý pro poskytování federativní identity (Wilson, et. al, 2019, s. 27). Ve vývoji správci identity se ukázalo, jak důležitá je pro spravování identity kvalitní práce s daty. Seskupení dat na jedno místo pomáhá nejen odbourat nekonzistenci v datech, ale umožňuje kvalitní a dynamickou správou uživatelských údajů.

Keycloak byl srovnán také s dalšími open-source projekty poskytujícími autentizační či autorizační služby. Ukázalo se, že Keycloak je ve své pozici open source projektů jedinečný, protože má silnou základnu komunity, která dbá o jeho vývoj. Dále jeho nasazení neklade nároky na kapacitu hardwaru[32]. Vybrané projekty od firmy Ory Corp jsou stále ve fázi vývoji, avšak také poskytují autorizační služby. Jejich projekt Ory Hydra má implementován protokoly OpenID Connect a OAuth 2. Naneštěstí společnost Ory Corp neposkytuje jednu souhrnnou službu, která by zaštítila podobný rozsah možností jako Keycloak. Není to ani jejich cílem a sami v dokumentaci ke svým projektům odkazují na Keycloak v případě, že uživatel vyhledává komplexnější řešení. Na druhé straně se nacházejí projekty od firmy Gluu Inc., které jsou daleko ambicióznější a jejichž tvůrci

[32] Keycloak (Red Hat, 2022) zabírá 1 GB paměti, zatímco Gluu zabírá 40 GB (Gluu, inc, 2022).

tvrdí, že jejich služby předčí možnosti Keycloaku. Nabízejí velkou škálu standardů, které přesahují rozsah Keycloaku.

Až čas ukáže, zda projekt Keycloak vydrží a bude se nadále držet ve špičce open source správců identit jako dosud. Nyní je však velmi snadné sledovat, jak se jeho vývoj neustále žene vpřed.

Praktická část sloužila k vývoji prototypu, do nějž byly implementovány služby Keycloaku s dalšími externími službami., např. s cloudy či rozhraní API. Došlo ke kombinaci služeb pro nasazení do produkce (Heroku), nastavení rozhraní API pro přístup k databázi (Hasura, GraphQL API) a samotné nastavení Keycloaku, tak, aby korespondoval s frontendem aplikace.

7 Zdroje

7.1 Literární zdroje

ANDERSON, Ross. *Security engineering: a guide to building dependable distributed systems*. Second edition. Indianapolis: Wiley, [2020]. ISBN 978-0470068526.

BURDA, Karel. *Kryptografie okolo nás*. Praha: CZ.NIC, z.s. p.o., 2019. CZ.NIC. ISBN 978-80-88168-49-2.

DOSTÁLEK, Libor a kolektiv. *Velký průvodce protokoly TCP/IP Bezpečnost*. Praha: Computer Press, 2003. CZ.NIC. ISBN 80-7226-849-X.

HARDJONO, Thomas a Scott CANTOR. OASIS Security Services (SAML) TC. *OASIS OPEN* [online]. USA: OASIS, 2005 [cit. 2022-07-10]. Dostupné z: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

KOLOUCH, Jan. *CyberCrime*. Praha: CZ.NIC, z. s. p. o., 2016. ISBN 978-80-88168-18-8.

KOLOUCH, Jan a Pavel BAŠTA. *CyberSecurity*. Praha: CZ.NIC, z.s.p.o., 2019. CZ.NIC. ISBN 978-80-88168-34-8.

PUŽMANOVÁ, Rita. *TCP/IP v kostce*. 2., upr. a rozš. vyd. České Budějovice: Kopp, 2009. ISBN 978-80-7232-388-3.

SAKIMURA, Nat, John BRADLEY, Michael Blair JONES, Breno DE MEDEIROS a Chuck MORTIMORE. OpenID Connect Core 1.0 incorporating errata set 1. *OpenID* [online]. Kalifornie, USA: OpenID Foundation, 2014, 8. 11. 2014 [cit. 2022-07-10]. Dostupné z: https://openid.net/specs/openid-connect-core-1_0.html

THORGERSEN, Stian a Pedro Igor SILVA. *Keycloak - identity and access management for modern applications: harness the power of Keycloak, OpenID Connect, and OAuth 2.0 protocols to secure applications*. Birmingham: Packt, 2021. ISBN 978-1800562493.

VAN OORSCHOT, Paul C. *Computer security and the internet: tools and jewels from malware to bitcoin*. Second edition. Cham, Switzerland: Springer, [2021]. Information security and cryptography. ISBN 978-3-030-83411-1.

WILSON, Yvonne a Abhishek HINGNIKAR. *Solving Identity Management in Modern Applications: Demystifying OAuth 2.0, OpenID Connect, and SAML 2.0*. New York City: Apress Berkeley, CA, 2019, XXVI, 311. ISBN 978-1-4842-5095-2. Dostupné z: https://doi.org/10.1007/978-1-4842-5095-2

7.2 Elektronické zdroje

BARKER, Peter. Identity Silos: shining a light on the problem of shadow identities. *Technative* [online]. 2020, 7. 5. [cit. 2022-07-12]. Dostupné z: https://technative.io/identity-silos-shining-a-light-on-the-problem-of-shadow-identities/

EDUID. Co je federace identit? [online]. Praha: Cesnet, 2022 [cit. 2022-07-25]. Dostupné z: https://www.eduid.cz/cs/about

Compare Okta vs Keycloak vs Gluu. *Saasworthy* [online]. Delhi: 91Digital Web Private Limited, 2022 [cit. 2022-07-25]. Dostupné z: https://www.eduid.cz/cs/about

FIDO ALLIANCE. How FIDO Works. *Fido alliance* [online]. Beaverton, OR: Fido alliance [cit. 2022-07-25]. Dostupné z: https://fidoalliance.org/how-fido-works/

FORMIDABLE. Authentication. [online]. Seattle, WA: Formidable [cit. 2022-07-25]. Dostupné z: https://formidable.com/open-source/urql/docs/advanced/authentication/

GAŠPARIK, Petr. Co je to identity management? – seriál o IdM část 1. [online]. 2022, 3. 15. [cit. 2022-07-12]. Dostupné z: https://ami.cz/novinka/co-je-to-identity-management-serial-o-idm-cast-1/

GLUU Inc. Gluu documentation [online]. 2020, 7. 5. [cit. 2022-07-12]. Dostupné z: https://gluu.org/docs/

GLUU Inc. Gluu [online]. 2022. [cit. 2022-07-22]. Dostupné z: https://github.com/GluuFederation

GLUU Inc. Gluu vs. Keycloak. *Gluu* [online]. Austin, TX: Gluu, 2020 [cit. 2022-07-25]. Dostupné z: https://gluu.org/gluu-vs-keycloak/

GORI, Massimiliano. History of Open Source Identity management (part 1). Canonical Ltd. [online]. 2021, 11. 8. [cit. 2022-07-13]. Dostupné z: https://ubuntu.com/blog/history-of-open-source-identity-management-part-1

GORI, Massimiliano. History of Open Source Identity management (part 2). Canonical Ltd. [online]. 2021, 11. 22. [cit. 2022-07-13]. Dostupné z: https://ubuntu.com/blog/history-of-open-source-identity-management-part-2

HASURA. What is Hasura? [online]. 2020 [cit. 2022-07-13]. Dostupné z: https://hasura.io/blog/what-is-hasura-ce3b5c6e80e8/

IBM. Top 5 Advantages of Software as a Service (SaaS). *IBM* [online]. 2020, 18 September 2020 [cit. 2022-07-12]. Dostupné z: https://www.ibm.com/cloud/blog/top-5-advantages-of-software-as-a-service?mhsrc=ibmsearch_a&mhq=saas

KirstenS. Top 5 Advantages of Software as a Service (SaaS). *The OWASP® Foundation* [online]. [cit. 2022-07-12]. Dostupné z: https://owasp.org/www-community/attacks/xss/

KirstenS. Cross Site Request Forgery (CSRF). *The OWASP® Foundation* [online]. [cit. 2022-07-12]. Dostupné z: https://owasp.org/www-community/attacks/csrf

KVAPIL, Jiří. Lekce 1 - Úvod do TypeScriptu [online]. 2018 [cit. 2022-07-12]. Dostupné z: https://www.itnetwork.cz/javascript/typescript/uvod-do-typescriptu

LAIRD, Peter. How Oracle, IBM, SAP, Microsoft, and Intuit are Responding to the SaaS Revolution. *LAIRD ONDEMAND: PETER LAIRD COVERS THE ENTERPRISE SOFTWARE SPACE, FOCUSING ON TOPICS SUCH AS SAAS, PAAS, CLOUD COMPUTING, MASHUPS, PORTALS, AND JAVA DEVELOPMENT.* [online]. 2008, 5. 6. [cit. 2022-07-12]. Dostupné z: https://peterlaird.blogspot.com/2008/06/how-oracle-ibm-sap-microsoft-and-intuit.html

LANE, Kin. Intro to APIs: What Is an API?. *Postman - blog* [online]. San Francisco: Postman, 2020, 5.10. [cit. 2022-07-17]. Dostupné z: https://blog.postman.com/intro-to-apis-what-is-an-api/

LARDINOIS, Frederic. Facebook's GraphQL gets its own open-source foundation [online]. Sunnyvale, CA: Yahoo, 2018 [cit. 2022-07-17]. Dostupné z: https://techcrunch.com/2018/11/06/facebooks-graphql-gets-its-own-open-source-foundation/

MARIANI, Dave. The Centralized Data Repository: Breaking Down Data Silos. *Atscale* [online]. 2022, 8. 3. [cit. 2022-07-12]. Dostupné z: https://www.atscale.com/blog/the-centralized-data-repository-breaking-down-data-silos/

MIMOSO, Michael. Office 365 Vulnerability Exposed Any Federated Account. *Threatpost: The First Stop For Security News* [online]. 2016, 28. 4. [cit.

2022-07-12]. Dostupné z: https://threatpost.com/office-365-vulnerability-exposed-any-federated-account/117716/

NÚKIB. Minimální požadavky na kryptografické algoritmy. *Doporučení kryptografické ochrany v oblasti kybernetické bezpečnosti* [online]. Brno: Národní úřad pro kybernetickou a informační bezpečnost, 2022 [cit. 2022-07-21]. Dostupné z: https://www.nukib.cz/download/publikace/podpurne_materialy/Kryptograficke_prostredky_doporuceni_v2.0.pdf

NÚKIB. Zpráva o stavu kybernetické bezpečnosti za rok 2021. [online]. Brno: Národní úřad pro kybernetickou a informační bezpečnost, 2022 [cit. 2022-07-21]. Dostupné z: https://www.nukib.cz/download/publikace/podpurne_materialy/Kryptograficke_prostredky_doporuceni_v2.0.pdf

ORY CORP. Documentation. *Ory* [online]. Doylestown, PA: Ory Corp, 2020 [cit. 2022-07-25]. Dostupné z: https://www.ory.sh/docs/welcome

ODBOR EGOVERNMENTU. Přehled kvalifikovaných poskytovatelů certifikačních služeb a jejich kvalifikovaných služeb [online]. Praha: Ministerstvo vnitra České republiky, 2016 [cit. 2022-07-25]. Dostupné z: https://www.mvcr.cz/clanek/prehled-kvalifikovanych-poskytovatelu-certifikacnich-sluzeb-a-jejich-kvalifikovanych-sluzeb.aspx

OKTA. Identity is Key to Stopping These 5 Cyber Security Attacks. *Okta* [online]. San Francisco, California: Okta, 2021 [cit. 2022-07-18]. Dostupné z: https://www.okta.com/resources/whitepaper/identity-is-key-to-stopping-these-5-cyber-security-attacks/

ORY CORP. Ory Documentation [online]. Mnichov: Ory Corp, 2022 [cit. 2022-07-22]. Dostupné z: https://www.ory.sh/docs/welcome

ORY CORP. Ory [online]. Mnichov: Ory Corp, 2022 [cit. 2022-07-22]. Dostupné z: https://github.com/ory

RED HAT. Documentation 1í.0.1 [online]. Connecticut: Red Hat, 2022 [cit. 2022-07-22]. Dostupné z: https://www.keycloak.org/documentation

RED HAT. Keycloak [online]. Connecticut: Red Hat, 2022 [cit. 2022-07-22]. Dostupné z: https://github.com/keycloak

SIMPLELEARN. What Is DES (Data Encryption Standard)? DES Algorithm and Operation. *Simplelearn* [online]. San Francisco, California: Si7.3mplelearn,

2022 [cit. 2022-01-08]. Dostupné z: https://www.simplilearn.com/what-is-des-article

SIMPLELEARN What Is AES Encryption and How Does It Work? *Simplelearn* [online]. San Francisco, California: Si7.4mplelearn, 2022 [cit. 2022-01-08]. Dostupné z: https://www.simplilearn.com/tutorials/cryptography-tutorial/aes-encryption

SINGH, Simon. *Kniha kódů a šifer: tajná komunikace od starého Egypta po kvantovou kryptografii*. Praha. Praha: Dokořán, 2003. Aliter (Argo: Dokořán): Dokořán): Dokořán). ISBN 80-865-6918-7.

SYSTEMS ENGINEERING. IT Security Cloud Services Industries About Culture The Evolution of Identity and Access Management (IAM). *Systems Engineering* [online]. Portland, Maine: Systems Engineering, 2021, 23.4. [cit. 2022-07-16]. Dostupné z: https://blog.systemsengineering.com/blog/the-evolution-of-identity-and-access-management-iam

POZA, Diego. What is CIAM?: The essential tools for managing user identity and how they impact your bottom line. *Auth0* [online]. Bellevue, WA: Auth0, 2021, 8. 6. [cit. 2022-07-25]. Dostupné z: https://auth0.com/blog/what-is-ciam/

SEDLÁK, Jan. Kvantová nadvláda ještě nepřišla, ale je blízko. Kvantové počítače IBM testují i Češi. [online]. Internet Info, s.r.o.: Lupa.cz, 2022 [cit. 2022-07-21]. Dostupné z: https://www.lupa.cz/clanky/kvantova-nadvlada-jeste-neprisla-ale-je-blizko-kvantove-pocitace-ibm-testuji-i-cesi/

ŠEDIVEC, Tomáš. Bezpečnost elektronické identity. *Architektura eGovernmentu ČR* [online]. Praha: Ministerstvo vnitra České republiky, 2021 [cit. 2022-07-18]. Dostupné z: https://archi.gov.cz/znalostni_baze:bezpecnost_identity

REKKAS, Aeneas. Why you probably do not need OAth2 / OpenID Connect. [online]. Doylestown, PA: Ory Corp, 2022 [cit. 2022-07-18]. Dostupné z: https://www.ory.sh/oauth2-openid-connect-do-you-need-use-cases-examples/

RED HAT. When will be keycloak be supported by Red Hat?. *Red Hat: Customer porta*l [online]. Raleigh, NC: Red Hat, 2016 [cit. 2022-07-25]. Dostupné z: https://access.redhat.com/solutions/1129963

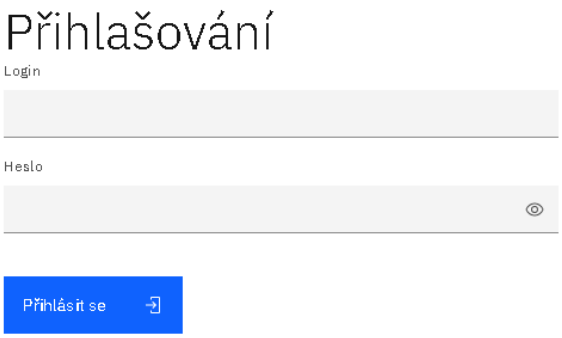
ROOT.CZ_. Kam směřuje cloud? Rozhovor s Filipem Špačkem z Master Internet [online]. Praha: Internet Info, s.r.o., 2022 [cit. 2022-07-25]. Dostupné z:

https://www.root.cz/pr-clanky/kam-smeruje-cloud-rozhovor-s-filipem-spackem-z-master-internet/

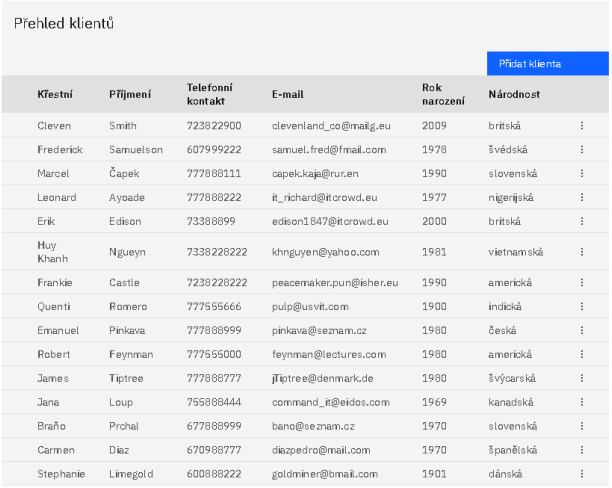
UNIVERSITY OF CONNECTICUT. The CAS Protocol for Application Owners. *INFORMATION TECHNOLOGY SERVICES: Identity & Access Management* [online]. Connecticut: University of Connecticut [cit. 2022-07-25]. Dostupné z: https://iam.uconn.edu/the-cas-protocol-for-application-owners/

ÚOOÚ. Často kladené otázky ohledně souhlasu s cookies uděleného prostřednictvím tzv. cookie lišty. [online]. Praha: Úřad pro ochranu osobních údajů, 2022 [cit. 2022-07-27]. Dostupné z: https://www.uoou.cz/casto-kladene-otazky-ohledne-souhlasu-s-cookies-udeleneho-prostrednictvim-tzv-cookie-listy/ds-6912/archiv=1&p1=2619

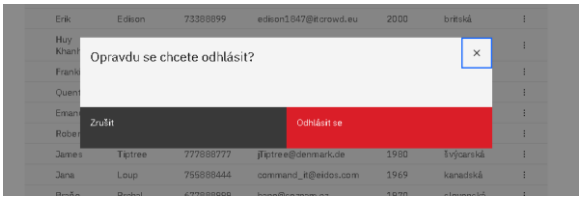
8 Přílohy



[1. přihlašovací stránka]



[2. přehled uživatelů]



[3. Modal - odhlášení]

[4. Odhlášení - oznámení]

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor:	**Mgr. Long Do**
Studium:	I1800483
Studijní program:	B1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název bakalářské práce:	**Současné bezpečnostní metody k autentizaci a autorizaci uživatele**
Název bakalářské práce AJ:	Current security methods for user authentication and authorization

Cíl, metody, literatura, předpoklady:

Cílem práce je analýza současných bezpečnostních praktik v autentizaci uživatele prostřednictvím open-source poskytovatelů identifikačních služeb a jejich komparace s důrazem na open-source řešení Keycloak. Výstupem bude webová aplikace, jejíž proces AAA se provádí prostřednictvím serveru Keycloak. Webová aplikace bude vyvíjena ve frameworku SvelteKit a prostřednictvím REST API bude možné provést registraci a autorizované vkládání nebo úpravy dat.

Documentation 16.1.0. *Keycloak - Open Source Identity and Access Management* [online]. Raleigh, Severní Karolína, USA: Red Hat, 2021 [cit. 2022-01-11]. Dostupné z: https://www.keycloak.org/documentation

BURDA, Karel. *Kryptografie okolo nás*. Praha: CZ.NIC, z.s.p.o., 2019. CZ.NIC. ISBN 978-80-88168-52-2.

VAN OORSCHOT, Paul. *Computer Security and the Internet: Tools and Jewels from Malware to Bitcoin*. 2. New York City: Springer, Cham, 2021, XXIX, 446. Information Security and Cryptography. ISBN 978-3-030-83411-1. Dostupné z: doi:https://doi.org/10.1007/978-3-030-83411-1

Zadávající pracoviště:	Katedra informačních technologií, Fakulta informatiky a managementu
Vedoucí práce:	Mgr. Josef Horálek, Ph.D.
Oponent:	Ing. Lubomír Almer, Ph.D.
Datum zadání závěrečné práce:	17.1.2022