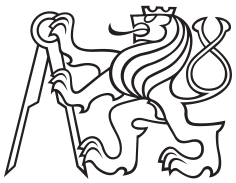


České
vysoké
učení technické
v Praze

Katedra telekomunikační techniky

Návrh architektury pro bezheslové ověřování uživatelů

Adam Olša

Školitel: Ing. Jaroslav Burčík, Ph.D.
Květen 2023

Poděkování

Chtěl bych poděkovat panu Burčíkovy za vedení práce a panu Marcelu Poláčkovy s pomocí naučení ve virtuálním prostředí.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

Podpis autora práce .....

V Praze, 26. května 2023

I declare that this work is all my own work and I have cited all sources I have used in the bibliography.

Prague, May 26, 2023

Abstrakt

V této práci je vysvětleno, že tradiční heslové metody autentizaci nejsou již v dnešní době bezpečné a prezentuje stávající metody bezpečnější bezheslové autentizace jako jsou např.: využití tokenů, USB klíčů nebo různých tvarů jednorázového hesla. V druhé části je uveden příklad implementace metody timed-one-time password na webové aplikaci ve virtuálním prostředí vSphere.

Klíčová slova: architektura, autentizace, bezheslová, php, totp, psql

Školitel: Ing. Jaroslav Burčík, Ph.D.

Abstract

This work explains that traditional password-based authentication methods are not quite safe nowadays and presents available passwordless methods, that are both more safe and convenient. Examples of such methods are: token-based authentication, USB keys or various forms of one-time passwords. The second part shows an example of a webpage application that uses timed-one-time passwords to authenticate it's users, implemented on a vSphere virtual platform.

Keywords: architecture, authentication, passwordless, php ,totp , psql

Title translation: Architecture Design for Passwordless User Authentication

Obsah

Obrázky

Tabulky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Olša** Jméno: **Adam** Osobní číslo: **499022**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra telekomunikační techniky**

Studijní program: **Elektronika a komunikace**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Návrh architektury pro bezheslové ověřování uživatelů

Název bakalářské práce anglicky:

Architecture Design for Passwordless User Authentication

Pokyny pro vypracování:

Vžijte se do role architekta a správce sítě malé firmy. Prostudujte různé přístupy k bezheslovému ověřování uživatelů. Navrhněte vzorovou architekturu a implementujte ji v prostředí virtuální laboratoře. Na této architektuře posléze realizujte vybrané přístupy ověřování uživatelů. Jako součást práce, navrhněte laboratorní úlohu pro studenty, která demonstruje jednotlivé komunikační protokoly a dění v procesu ověřování uživatele.

Seznam doporučené literatury:

[1] Phillip J. Windley, Learning Digital Identity, O'Reilly Media, Inc. 2023, ISBN: 9781098117696
[2] Materiály do stupné na https://www.ismsforum.es/ficheros/descargas/open-reference-architecture-for-security-and.pdf [on-line]

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jaroslav Burčík, Ph.D. katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **14.02.2023** Termín odevzdání bakalářské práce: **26.05.2023**

Platnost zadání bakalářské práce: **22.09.2024**

Ing. Jaroslav Burčík, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.	
Datum převzetí zadání	Podpis studenta

Cíl práce

Cílem práce je vyzkoušet si práci síťového architekta a správnce malé, fiktivní firmy. Prvním cílem je seznámit se s existujícími metodami bezheslové autentizace, jak fungují a jaké mají výhody a nevýhody. Dalším cílem je navrhout vzorovou architekturu firemní sítě a její vytvoření. Posledním cílem je vybrání vhodné autentizační metody na základě rešerše a poté ji implementovat na vytvořenou síť.

Úvod

Tradiční model heslové autentizace je v neustále rozrůstajícím prostředí digitální bezpečnosti, čím dál tím méně bezpečný. Nové, sofistikovanější typy útoků a levně dostupné a výkonnější stroje umožňují velmi rychlé prolomení klasických heslových systémů. Hlavním zaměřením kyberbezpečnostních instutucí v dnešní době je vývoj a hladký přechod k využívání nových, bezpečnejších a přívětivějších bezheslových autentizačních modelů a standardů. V této práci představím hlavní vady hesel a uvedu stávající bezheslové systémy s kterými se dnes můžete setkat. Na závěr budu demonstrovat implementaci vybraného autentizačního systému ve virtuálním prostředí.

Model Autorizace, Autentizace a Záznamování

Více známý jako AAA (Authorization, Authentication, Accounting) je bezpečnostní model, který se stará o přístup k elektronickým
prostředkům a udržuje záznamy o jejich použití [1]. Skládá se ze tří částí:

- Authorization (Autorizace/Oprávnění) - Tato část se zabývá přidělováním příslušných privilegií určitým uživatelům. Tyto privilegia mohou určovat do jakých částí sítě či systému má uživatel přístup, nebo například kdy a kolik akcí může provádět. Změny privilegií jsou většinou prováděny administrátorem, ale mohou být v omezené míře automatizované.
- Authentication (Autentizace/Ověření) - Proces autentizace se zabývá ověřením, že určitý uživatel vlastní správné oprvávnění pro přístup. Autentizace může probíhát na základě mnoha různých kritérií, které jsou dále rozvedeny v kapitole 1.
- Accounting (Záznamování) - Poslední část se zabývá zaznamenáváním čiností uživatelů v sítí či systému. Např.: odkuď, kdy a jak dlouho byl uživatel přihášen nebo jaké stránky naštívil. V bezpečnostím smyslu tato část např. umožnuje objevit kde vznikl nějaký problém.

Použití tohoto modelu omezí přístup neoprávněných uživatelů a zaznamená aktivity oprávněných uživetlů, takže administrátor může vidět jestli nezneužívají své privilegia.

Nedostatky hesel

Koncept hesel je známý již od starodávných civilizací. Záznamy již ze starověkého Řecka popisují způsob využití hesel [2]. Zde se ale zabýváme moderními hesly a jejich funkcí.

Prvním nedostatkem hesel je jejich *abstraktní* charakteristika, tedy že heslo nemá fyzickou formu. Útočník toto může využít a získat heslo pomocí metod jako jsou phishing[1] nebo odposlech[2].

Dalším nedostatkem je jejich přítomnost na serverech. Hesla pro svoji funkci musí být v nějaké formě uložena na straně služby, ke které se uživatel přihlašuje. Přestože hesla jsou většinou uložena v *hash*[3] formě, pokud se k nim útočník dostane může použít metodu *cracking*[4] pro rozšifrování hesel.

Hlavním problémem s hesly je jejich jednoduchost v porovnání s alternativními metodami. Heslo je pouze určitý seznam znaků v určitém pořadí. Toto umožňuje provést *brute force* útoky, které zkouší různá hesla dokud se nedostane ke správnému. V závislosti na délce hesla a počtu různých znaků můžeme vypočítat počet všech kombinací N podle

$$N = Z^d, \tag{1}$$

kde Z je počet možných znaků a d je délka hesla. Pokud dále uvažujeme dostupnou procesorovou soupravu s rychlostí $632 \cdot 10^9$ pokusů za vteřinu [3] můžeme vytvořit tabulku 1. Z tabulky lze vidět, že nějaké hodnoty přesahují hranici roků, což bych považoval za relativně bezpečné. Tato tabulka ale předpokládá rovnoměrné rozložení hesel, tedy že každá kombinace má stejnou pravděpodobnost výskytu. V další části se zaměříme na několik statistik ohledně hesel, které ukazují na to, že hesla rovnoměrně rozložená nejsou.

[1] Různé způsoby vymámení hesla z uživatele, např. přes falešné přihlašovací stránky

[2] Když se útočník připojí ke komunikačnímu médiu a najdou v něm heslo při přenosu

[3] Metoda nevratného šifrování.

[4] Způsob kdy útočník provede vlastní hash pro různé hesla a pak je porovná s obdrženými hashy, aby našel shodu.

Tabulka 1: Rychlost vyzkoušení všech možných kombinací hesel

Délka	Jen čísla	Jen písmena	Písmena a čísla	Písmena, čísla a znaky
4	<1 s	<1 s	<1 s	<1 s
5	<1 s	<1 s	<1 s	<1 s
6	<1 s	<1 s	<1 s	20 s
7	<1 s	2 s	6 s	49 min
8	<1 s	1 min	6 min	5 dní
9	<1 s	1 hod	6 hod	2 roky
10	<1 s	3 dny	15 dní	330 let
11	<1 s	138 dní	3 roky	50000 let
12	2 s	20 let	162 let	$8 \cdot 10^6$ let
13	16 s	1000 let	10000 let	10^9 let
14	3 min	$53 \cdot 10^3$ let	$622 \cdot 10^3$ let	$176 \cdot 10^9$ let
15	26 min	$3 \cdot 10^6$ let	$39 \cdot 10^6$ let	$27 \cdot 10^{12}$ let
16	4 hod	$143 \cdot 10^6$ let	$2 \cdot 10^9$ let	$4 \cdot 10^{15}$ let
17	2 dny	$7 \cdot 10^9$ let	$148 \cdot 10^9$ let	$619 \cdot 10^{15}$ let
18	18 dní	$388 \cdot 10^9$ let	$9 \cdot 10^{12}$ let	$94 \cdot 10^{18}$ let
19	183 dní	$20 \cdot 10^{12}$ let	$570 \cdot 10^{12}$ let	$14 \cdot 10^{21}$ let
20	5 let	10^{15} let	$35 \cdot 10^{15}$ let	$2 \cdot 10^{24}$ let

Ze studií a statistik ohledně reálných uživatelských hesel [4], [5], [6], lze vidět několik zajímavých výsledků. Průměrná délka hesla leží mezi 8 a 11 znaky. Většina hesel obsahuje velká i malá písmena a čísla[5]. Toto znamená, že na prolomení průměrného hesla stačí mezi 6 minutami a 3 roky času. Toto ale není vše, studie [6] odhalila, že většina hesel obsahovala právě jedno velké písmeno, právě na začátku. Dále zjistila, že většina hesel obsahovala právě 2 nebo 4 čísla, právě na konci, tyto čísla ve většině případů reprezentovaly nějaký rok. Pokud heslo obsahovalo speciální znak, tak se většinou vyskytoval úplně na konci za číslicemi. Pokud vezmeme v potaz tyto statistiky tak i zdánlivě bezpečné heslo může být velice nebezpečné. Například prolomení 12-ti cifrového hesla s malými i velkými písmeny a čísly typicky trvá 162 let. Pokud ale heslo následuje výše zmíněné charakteristiky a útočník to předpokládá, doba prolomení se sníží na pouhých 7 hodin.

Poslední z nedostatků, je samotná podstata, že uživatel si heslo musí pamatovat. Toto by sám o sobě problém nebyl, ale čím je heslo složitější, tím hůře si ho uživatel pamatuje. To vede k tomu, že si uživatel zvolí jednodušší heslo, nebo ho dokonce opakuje pro více služeb. Někteří uživatelé si místo pamatování heslo uloží jinak, např. si ho napíší na papír vedle počítače, což může být velké riziko.

[5]Toto je pravděpodobně způsobeno minimálními požadavky na komplexitu hesla.

Kapitola 1

Stávající metody bezheslové autentizace

Bezheslovovou autentizací je míněna taková autentizace, kde samotný uživatel nemusí pro příhlášení do určité služby zadat heslo. Je ale důležité zmínit, že heslo může být stále využito, pokud ho uživatel nezadává sám, zadá jen jednou nebo ho zadá jinde. Současné metody autentizace dělíme do tří kategorií, dle charekteristiky využitého prvku:

- *Něco co máš* – Znamená vlastnicví nějakého předmětu, typicky autentizátor, specifické zařízení jako je např. mobilní telefon nebo nějaký typ tokenu. Do této metody se také zahrnuje případ kdy se uživatel příhlásí jinak, např. pomocí hesla a služba si pamatuje zařízení a umožní se přihlásit bez autentizace. Tato metoda je považována za nejbezpečnější, jelikož zmocnit se fyzického předmětu je pro útočníka hodně práce a replikace je prakticky nemožná. Rizikem této metody je, že pokud se útočník předmětu zmocní, může mít přístup ke všem spojeným službám.

- *Něco co jsi* – Označuje biometrické prvky, které se používá pro ověření uživatele na autentizačním zařízení. Výhoda těchto metod spočívá v tom, že si uživatel nemusí nic pamatovat a autentizace je většinou velmi rychlá, ale zařízení mohou být relativně drahá. Samotná bezpečnost se liší na základě biometrického prvku a kvalitě zařízení, kde některé jsou jednodušší a některé složitější na prolomení.

- *Něco co znáš* – Tato metody představuje typ autentizace dle nějaké informace kterou uživatel zná. Typickým případem jsou hesla či uričitá gesta. Tento samotný typ autentizace není moc bezpečný a je jednoduché ho prolomit. Bezheslové autentizace s v tomto případě dosahuje buď kombinací s jinou metodou (typicky vlastnictví) nebo vylepšením určitého aspektu, např. že se uživatel přihásí jen jednou na delší dobu, nebo využití druhého, typicky bezpečnějšího, účtu.

1.1 Autentizace pomocí tokenu

Principem této metody je, že uživatel projde jednou autentizací a obdrží tzv. *token*. Token je typicky pseudonáhodná informace, která dokazuje že uživatel prošel autentizací. Tokeny jsou ve většině případů vydány na určitou dobu, která může být určena časem nebo dokud není manuálně ukončena, toto závisí vždy na specifickém systému. Tokeny umožňují uživateli přihlášení do služby, pro kterou byl token udělen, bez další autentizace. Rozlišujeme čtyři typy tokenů [7]:

- Kontaktní – Představují tokeny obsažené v zařízeních, které se připojují přímo do systému. Typickým příkladem je USB klíč.
- Bezkontaktní – Představují tokeny obsažené ve fyzických zařízeních, které komunikují bezdrátově na krátké vzdálenosti. Typickým příkladem jsou přístupové karty.
- Vzdálené – Představuji tokeny obsažené v zařízeních, které nejsou připojeny přímo do systému, ale poskytují informace pro autentizace, např. jednorázová hesla. Typickým příkladem jsou různé aplikace na mobilním telefonu.
- Lokální – Představují tokeny, které jsou přítomny přímo na zařízení na kterém se uživatel přihlašuje. Typickým příkladem jsou cookies.

Použití tokenů pro autentizaci je v dnešní době velmi časté a existuje mnoho systémů které tento způsob využívá.

1.1.1 Single sign-on

Single sign-on (dále jen SSO) je autentizační systém, který umožňuje uživateli provést jen jedno přihlášení na více spojených, ale nezávislých služeb [8]. Hlavními prvky tohoto sytému jsou jednotlivé služby a jeden tzv. *poskytovatel identity* (dále jen ID poskytovatel), což může být externí služba (např. OneLogin [9]), které důvěřují na základě certifikátu či interní, ale oddělená služba (např. CVUTID [10]). Registrace na službě, která využívá SSO znamená registraci u ID poskytovatele a umožňuje příhášení do všech službách, které sdílí tohoto ID poskytovatele. Tato registrace probíhá jako standardní registrace u heslového nebo jiného systému. První přihlášení na tento systém probíhá stejně jako u heslových nebo jiných systémů s jediným rozdílem, tím je že ID poskytovatel udělí uživateli lokální token. Tento token je většinou časově omezen na dobu několika minut až několika hodin, nebo dokud se uživatel odhlásí. Tento token je dále využit na autentizaci uživatele při přhlášení do jakékoliv služby spojené s ID poskytovatelem, který token vydal.

Výhody systému SSO spočívá v menším počtu přihlášení, tzn. rychlejší přihlášení pro uživatele a menší výskyt informací na internetu (při přenosu), což omezuje riziko útoků typu odposlech. Dále v případě využití hesel, uživatel potřebuje jen jedno heslo na více účtů a nemusí si jich pamatovat více, toto

může vést k využití sílnějšího hesla bez vyšší náročnosti na paměť. Toto platí i pro paměť na serverech, tím že indivudální služby si nemusí pamatovat uživatelská hesla. I když tato centralizovaná struktura omezí náklady na jednotlivé služby a jejich administraci, zvyšuje nároky a rizika na centrálního ID poskytovatele. Navíc pokud se naskytne problém s ID poskytovatelem, tak může uživatel být odstřižen od všech spojených služeb.

Problémem systému je, že je plně závislý na bezpečnosti prvního přihlášení uživatele, které velmi často bývá heslové a je tak náchylné na všechny typy útoků jako heslové systémy. To, že uživatel *může* použít silnější heslo neznamená, že ho opravdu použije. Navíc pokud se útočník hesla skutečně zmocní, má přístup ke všem spojeným službám, což ho často činí mnohem horší, než využití individuálních přihlášení.

1.1.2 Cookies

Jeden z nejrozšířenějších způsobů autentizace na webových stránkách je použití HTTP cookies [11]. Uživatel se registruje a přihlašuje pomocí nějaké, typicky heslové, metody. Při přihlášenní uživatel obrdrží lokální token ve formě cookie do prohlížeče [12]. Tento cookie většinou obsahuje uživatelské jméno nebo jiné infromace. V každé další komunikaci se stránkou prohlížeč posílá i tento cookie, který stránka využije pro autentizaci. Cookie tohoto typu se nazývá *trvalý cookie*. Trvalý zde znamená jen rozdíl od tzv. *session cookie*, který trvá jen dokud uživatel nezavře stráku. Trvalý cookie má většinou dobu platnosti po které přestane fungovat. Tyto cookies typicky jsou, ale nemusí být šifrované jak při přenosu, tak v prohlížeči, záleží na nastavení na stránce.

Tento způsob poskytuje jednodušší přihlášení na stránky na které se uživatel již jednou přihlásil. Další výhodou je jednoduchá implementace na webové stránce, která používá přímo funkce webového prohlížeče.

Stejně jako SSO je ale závisý na bezpečnosti prvního příhlášení a je náchylný na stejné útoky. Navíc pokud se útočník dostane k zařízení, může mít přístup na všechny stránky kde uživatel využívá k autentizaci cookies. Málo zabezpečené cookies mohou být odcizeny pomocí tzv. *CSRF* (cross-site request forgery) útoků, což je metoda kde cizí stránka předstírá, že je jiná stránka aby dostala příslušný cookie.

1.1.3 PHP Sessions

PHP Session je pokročilý způsob cookies, kde informace je uložená na serveru stránek namísto prohlížeče [13]. Token, který v tomto případě uživatel obdrží, je cookie který jen identifikuje session, ve které jsou uložené důležité informace. Toto znamená, že narozdíl od cookies, se tyto informace neposílají při každé komunikaci.

Tento způsob je prakticky identický s cookies, ale pokud se útočník v tomto případě zmocní cookies, nenalezne v nich žádné osobní informace.

1.1.4 Přístupové karty

Tento systém využívá párů fyzických zařízení, jmenovitých přístupových karet a tzv. *čteček* [14]. Samotné karty jsou typ bezkontaktního tokenu, které se přiloží, protáhne nebo vsune do čtečky, která dále aktivuje elektronický zámek. Systémy mohou být typu tzv. *Access Control*, to znamená, že čtečky jsou připojeny na nějaký centrální systém, který provádí verifikace karet a ovládá přístup. Čtečka v tomto případě jen čte data z karty a přeposílá je centrálnímu systému. Tento systém je velmi modulární v tom jaké priviliegie se dají nastavit na jednotlivé karty. Dá se nastavit např. jaké dveře otevře, v jaké časy nebo dny a dokonce kolikrát může uživatel vstoupit. Dále umožňuje jednoduchou deaktivaci karet v případě ztráty. Alternativou je systém kde čtečky provádí veškerou činnost bez centrálního systému, tato metoda se nejvíce podobá mechanickým zámkům a klíčům a používá se např. v hotelech. Samotné karty jsou v systému pasivní, tedy nepotřebují napájení což zvyšuje jejich životnost. V závislosti na využité technologii rozlišujeme 3 typy karet:

- Magnetický pruh – Také známy jako protahové, tyto karty nesou informace v magnetickém pruhu, který je složen z tisíců malých proužků. Každý proužek může být polarizován buď severně či jižně, což představuje binární kód. Informace se na kartu zapisují pomocí kódovacího přístroje, který přichází se systémem. Při protažením karty čtečka dokáže přečíst informace obsažené v magnetickém pruhu a udělí či odmítne přístup. Systémy které využívají tento typ karty mohou, ale nemusí být typu Access Control.

  Magnetické karty jsou z pravidla levnější než alternativy. Navíc jelikož informace není přenesena prostorově, nemůže být odchycena potenciálním útočníkem při přenosu. Nevýhodou je, že magnetické pruhy a informace na nich se mohou poškodi, často bývají nespolehlivé a mohou vyžadovat několik protáhnutí pro správné přečtení. Navíc jejich jednoduchá technologie znamená, že jsou jednodušší na replikaci pro útočníka.

- RFID karty – RFID (Radio-Frequency IDentification) karty využívají RF a NFC[1] technologii pro přenos informací. Samotné karty obsahují tři důležité komponenty - Jednoduchou anténu, kondenzátor a integrovaný obvod. Informace o ID uživatele je obsažena v obvodu. Informace se ukládají do paměti integrovaného obvodu pomocí RFID zapisovače. Čtečky tohoto typu karty vysílají stálé rádiové pole. Při přiložení se na kartu indukuje elektrický proud ze zmíněného pole. Tento proud napájí obvod v kartě, která začne vysílat ID uživatele. Čtečka tuto informaci předá centrálnímu systému, který udělí či odmítné přístup. RFID systému jsou výhradně typu Access Control.

  Jelikož se informace přenáší přes rádiové pole, přenášená informace může být v určitých případech odposlechuta.

[1] Near-Field-Communication.

- Wiegand karty – Karty tohoto využívají tzv. *Wiegandův jev*, což je nelineární magnetický jev, který se vyskytuje na specifickém typu drátu [15]. Karty se aktivují protažením čtečkou podobně jako karty s magnetickým pruhem. Wiegandovy karty se již moc nepoužívají jelikož informace na kartě je nastavena při výrobě a nedá se nijak změnit, což je činní nevhodné pro praktické využití. Tato charakteristika ale měla i své výhody, hlavně velmi dlouhá životnost a vysoká odolnost proti útočníkům, kteří by chtěli kartu přepsat.

Obecnou výhodou karetních systémů je vysoká modulárnost privilegií, možnost deaktivace jednotlivých karet, pohodlí uživatele, jednoduché záznamy přístupu a jednoduchá instalace. Hlavní výhodou je nižší cena oproti alternativním systémům (mimo mechanických zámků).

Nevýhodou je hlavně relativně nízká bezpečnost oproti alternativám, většinu karet je jednoduché a levné replikovat či přepsat. Navíc cena jednotlivých karet je např. oproti mechanickým klíčům velmi vysoká, toto v kombinaci s jenoduchou ztrátou či poškození karty může vést k velkým nákladům.

1.1.5 HMAC a time-based one time password

HMAC[2]-based one-time password (dále jen HOTP) je algoritmus založen na tajném klíči a inkrementálním čítači uloženém v autentizátoru [16]. Time-based one-time password (dále jen TOTP) je algoritmus který opět využívá sdílený tajný klíč, ale místo čítáče využívá synchorizovaný systémový čas. Token je v tomto systému právě tajný klíč a je typu vzdálený. Autentizátor může v obou případech představovat unikátní fyzické zařízení nebo aplikaci na nezávislém zařízení. Principem je vytváření unikátních jendorázových hesel na základě sdíleného tajného klíče a hodnoty v čítači nebo hodin. U systému HOTP se při každé generaci hesla inkrementuje čítač, což zajistí unikátnost každého hesla. U systému TOTP se použitelné heslo mění po určitých časových úsecích, typicky 30-60 vteřin. Toto heslo je dále hashováno pro omezení rizika odposlechu. Systém tohoto typu sice může být, ale většinou není použit jako hlavní autentizační systém. Velmi často se používá jako druhý faktor pro heslovou autentizaci.

Jednorázová charakteristika podstatně omezí riziko útoků typu odposlech a brute force, jelikož se hesla stále mění. Systém TOTP je jednodušší na implementaci, ale časová závislost může způsobit problémy pokud se server a zařízení desynchronizují. Systém HOTP je více náchylný na útoky typu brute force, jelikož se heslo mění méně často. Toto vedlo některé systémy k přidání časového komponentu, čímž se vytváří jistá kombinace HOTP a TOTP.

Problém nastane pokud se útočník dostane k tajnému klíči, který je přítomen jak na serveru služby tak na zařízení uživatele, který dále umožní neomezený přístup. Navíc pokud je registrace software autentizátoru odposlechnuta, útočník může mít přistup k tajnému klíči. Toto naopak není

[2]hash-based message authentication code.

problém u fyzických autentizátorů, které mají klíče zabudované. I když jsou systémy relativně přívětivé, zadávání hesel z jiného zařízení může být někdy otravné, hlavně v případě TOTP kde se musí heslo zadat v omezeném časovém okně.

1.1.6 USB klíč

USB klíče jsou USB zařízení nesoucí token, který identifikuje uživatele, jemuž bylo zařízení přiděleno [17]. USB klíče jsou typicky připojeného typu, ale novější USB klíče mohou zároveň fungovat i bezkontaktně na stejné technologii jako RFID karty. Tyto zařízení bývají často zabezpečeny druhým prvkem přímo na zařízení, např. skener otisku prstu, který dokazuje identitu vlastníka. Většina USB klíčů využívá kryptografii veřejných klíčů (popsána na začátku sekce 1.2) Během přihlašování uživatel namísto zadání hesla připojí USB klíč a autentizace proběhne automaticky. USB klíče mohou být typicky použity pro více účtů, kde pro každý účet vytvoří nový token. Tokenem je v tomto případě soukromý klíč uživatele.

Použití fyzického předmětu a kryptografii veřejných klíčů prakticky eliminuje riziko útoků typu odposlech, phishing a brute force a pokročilé protokoly jako např. FIDO U2F [18] omezí riziko MITM[3] útoků. Navíc jsou často ry-chlejší a jednodušší na použití než hesla. Zařízení je relativně malé a není problém ho připnout na klíče nebo nosit v peněžence.

Fyzická podoba USB klíčů je ale i nedostatkem, jelikož je nutné ho stále nosit u sebe, což může vést ke ztrátě či odcizení. Ztráta takového zařízení může často vést ke ztrátě celého účtu, pokud uživatel nevlastní záložní. Dalším problémem může být samotná závislost na USB technologii, která není adekvátně zapezpečena pro použití v důležitých sítích. Firmware USB zařízení je relativně jednoduché přepsat pro přenos různých malwarů. USB zařízení navíc může provádět akce před tím než je zařízení schopno provést jeho kontrolu, což může dále infikovat počítače, které budou infikovat další USB zařízení. Malware se tak může v síti, která využívá mnoho USB zařízení, velmi rychle šířit [19]. Dalším problémem může být relativně malé možnosti využití, jelikož mnoho (hlavně menších) služeb tyto zařízení nepodporuje. Jednotlivá zařízení jsou mohou být relativně drahé.

1.2 Autentizace založena na certifikátu

Tato metoda využívá kryptografii veřejných klíčů, jinak nazýváno asymetrická kryptografie [20]. Tato metoda využívá k šifrování a rozšifrování 2 různé klíče - veřejný a soukromý. Pří vysílání se data zašifrují veřejným klíčem příjemce, tyto data se poté dají rozšifrovat jen soukromým klíčem příjemce, který má jen on. Uživatel v tomto případě drží svůj soukromý klíč a digitální certifikát, který mu vydá Certifikační Autorita (dále jen CA). CA může být buď nějaký interní nebo externí systém. Tento certifikát se ve své podstatě nedá změnit, při jakékoliv změně by se poškodil podpis CA, což by certifikát znehodnotilo.

[3] Man-in-the-middle

Toto dává systému velký stupeň bezpečnosti. Certifikáty od pokročilých CA je prakticky nemožné padělat. Systém je tak v celku velmi odolný proti MITM, phishing a odposlech útokům.

Narozdíl od tokenů, které se přímo používají k autentizaci, certifikáty obsahují informace o uživateli, které jsou potřeba k autentizaci. Tyto informace mohou zahrnovat:

- Veřejný klíč uživatele
- Jméno zařízení či uživatele
- Jméno CA, která ho vydala
- Podpis CA
- Datum začátku a vypršení platnosti
- Verzi certifkačních dat
- Sériové číslo

Během autentizace server pošle uživateli tzv. *nonce*, což jsou nějaká náhodná data. Zařízení uživatele tento nonce zašifruje pomocí soukromého klíče a výsledek odešle společně s certifikátem zpět serveru. Server dále zkontroluje zda jsou informace na certifikátu platné a rozšifruje data pomocí přijatého veřejného klíče. Pokud se vyslaná a přijatá data shodují tak autentizace proběhla úspěšně.

Systém umožňuje relativně vysokou kontrolu nad jednotlivými certifikáty, které se dají jednodušše vydat, prodloužit či zrušit. Narozdíl od dalších systémů, certifikáty mohou být vydány nejen osobám, ale i jednotlivým zařízením jako jsou servery, PC nebo cokoli spadající pod IoT. Dokud je soukromý klíč udržen tajný, systém je velmi bezpečný.

Nevýhodou je hlavně relativně vysoká cena oproti alternativám, toto platí jak pro zavedení systému, tak jeho trvalý provoz. Pokud je odcizen privátní klíč uživatele, útočník je schopen se za něj vydávat a autentizační systém to nedokáže rozeznat. Navíc pokud útočník prolomí bezpečnost CA může vytvářet platné certifikáty pro všechny spojené služby [21].

1.3 Autentizace přes 3. stranu

Tato metoda zahrnuje způsoby, kde autentizace probíhá přes existující účty či zařízení patřící jiné, nezávislé službě. Účty na službě která tuto metodu využívá jsou tak plně závislé na účtech přes které probíhá autentizace. Již zmíněná metoda SSO 1.1.1, kde je využito externího ID poskytovatele je případem této metody. Tato metoda je velmi rozšířená, kvůli jednoduché implemetaci a úspoře na vlastních autentizačních systémech, ale autentizace se tak stává závislá na cizím systému, kterému musí důvěřovat.

1.3.1 Využití účtu 3. strany

Velmi častou metodou autentizace pro menší služby je využití bezpečnějších a častých účtů na jiných službách (typicky jsou to účty na sociálních médiích jako Facebook a Twitter, nebo jiné jako např. Google), namísto implementace vlastních autentizačních systémů. Uživatel si tak nemusí vytvářen nový účet a vymýšlet nové heslo, což je činní mnohem přívětivější.

Problém ale může nastat pokud uživatel nevlastní žádný z vyžadovaných účtů a je tak donucen si vytvořit nový, poteciálně nežádoucí, účet. Některé služby tomuto zabraňují tak, že tuto metodu nabízí jen jako alternativní ke konvenční registraci. Dále hrozí, že pokud uživatel ztratí nebo jsou mu odcizeny údaje k danému účtu, může ztratit všechny spojené účty.

1.3.2 Jednorázové heslo přes SMS

Dalším z častých řešení je využití unikátnosti SIM karet k identifikaci uživatele pomocí tzv. SMS OTP (SMS one-time-password) [22]. Tato metoda bývá nejčastěji používána jako drůhý faktor autentizace, ale dá se využít i samostatně. Uživatel může ke svému účtu registrovat telefonní číslo, na které mu dále budou chodit SMS s jednorázovými hesly. Při autentizaci je uživateli odeslána SMS s jednorázovým heslem, které musí dále přepsat zpět do prohlížeče.

Systém je velmi rozšířený jelikož prakticky každý vlastní mobilní telefon a jeho použití je velmi snadné, i když přepisování kódu z telefonu do prohlížeče může být otravné. Největší výhodou této metody je využití fyzického zařízení, kterého je složitější se zmocnit.

Závislost na SMS ale může být velkým bezpečnostním hazardem. SMS jako takové nejsou šifrované, což je dělá velmi náchylné na útoky typu odposlech. Útočník může také využít tzv. *SIM swap*, což je metoda kdy útočník shromažďí informace o uživateli (např. pomocí phisingu) a přesvědčí poskytovatele mobilních služeb aby migroval telefonní číslo na SIM kartu útočníka, čímž získá přístup ke zprávám uživatele [23]. Oproti podobné e-mail OTP metodě je SMS OTP relativně drahé, jelikož jednotlivé zprávy jsou mnohem dražší než e-maily.

1.3.3 Jednorázové heslo přes e-mail

Tato metoda je velmi podobná metodě SMS OTP, ale zprávy přichází na e-mail uživatele namísto telefonu [22]. Tato metoda je často využívána jen jednou během registrace účtu na e-mail a často slouží jen jako záložní pokud uživatel např. zapomene heslo svého účtu. Dá se ale využít úplně samostatně s tím, že uživatel musí využít svého e-mail účtu při každém přihlášení. Obdržený e-mail může obsahovat buď náhodně generovaný kód, který uživatel zkopíruje na stránku služby nebo může obsahovat odkaz, který uživatele automaticky přihlásí.

Výhody této metody spočívají v jednoduché implementaci a použití a nízké ceně jednotlivých e-mailů. Bezpečnost účtů na službách využívající tuto

metodu jsou vázené s bezpečností jednotlivých e-mailových schránek, což může být výhoda i nevýhoda.

E-maily jako takové nejsou v základní formě nijak šifrované a mohou být odposlechuty podobně jako SMS [24]. Uživatelé, kteří používají tuto metodu jsou více náchylní na phishing odkazy, jelikož jsou zvyklí bezmyslně klikat na odkazy které jim přijou na e-mail. Velké využití a závislost na e-mailových účtech je činní hlavními cíly pro útočníky, kteří jejich prolomením mohou získat přístup k velkému množství účtů a informací.

1.4 Biometrické systémy

Tyto systémy využívají pro identifikaci uživatele biometriku [25]. Biometrika je unikátní údaj osoby založena na fyzických prvcích nebo chování uživatele. Existuje mnoho různých biometrik, kterých lze k identifikaci použít, nejčastější z nich jsou otisk prstu, hlas, obličej nebo DNA. Tyto systémy využívají pokročilé skenery a algoritmy k převedení biometrických prvků do číselné podoby, která se poté dá uložit a porovnávat. Rozlišujeme dva hlavní typy systémů:

- Dálkové – V tomto případě uživatel využívá vzdáleného zařízení na ověření své biometriky. Naproti drůhému systému, zde informace o uživateli nemusí opouštět zařízení uživatele a omezují tak riziko, že budou odcizeny. Většina dnešních telefonů ma zabudované biometrické sensory a není tak potřeba pořizovat nové zařízení. Tyto sensory jsou ale často menší kvality a dají se jedodušeji obelhat.

Výhodami biometrických systémů je rychlé, jednoduché a pohodlné použití pro uživatele, který si nemusí nic pamatovat, jen přiloží prst nebo se podívá do skeneru. Samotná bezpečnost systémů závisí na využité biometrice a kvalitě sensoru. Většina kvalitních systémů je extrémně složitá na prolomení nebo obelhání, ale jednoduché systémy jde rychle obejít.

Biometrické systémy ale nejsou tak bezpečné, jak se mohou ze začátku zdát. I když je velmi složité obržet skutečné biometrické prvky uživatele, útočník může využít jiné metody na obelhání samotného sensoru. Prvním je tzv. *prezentační* útok, kde útočník předloží zfalšované biometriky (např. fotku skutečného uživatele na skener obličeje). Pokročilé systémy jsou schopné tyto útoky identifikovat, ale nikdy nejsou perfektní. Dalším způsobem je použití tzv. *master printu*. Tato metoda je použitelná proti systémům, které využívají částečnou schodu. Tento master print je uměle vytvořený otisk, který obsahuje mnoho častých prvků, které se vyskytují ve skutečných otiscích. Tento otisk se poté dá použít pro přístup na velké množství účtů. Jednou z největších nevýhodou je jejich velmi vysoká cena, která je opět závislá na využitých sensorech.

Kapitola 2

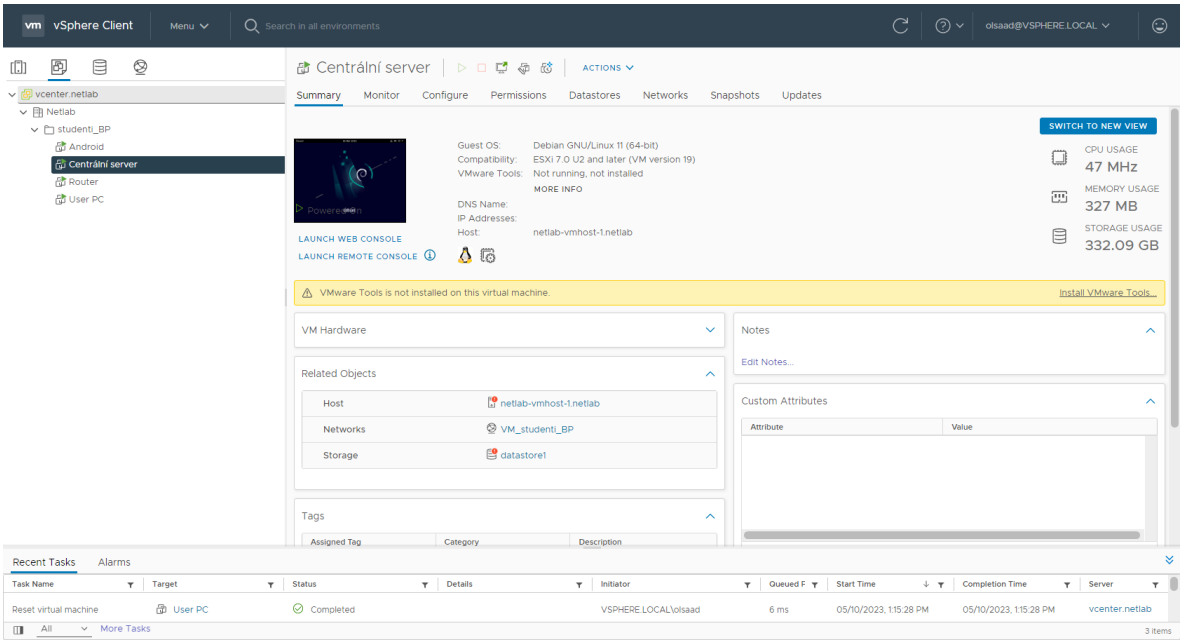
Vybraná metoda vypracování

Na základě rešerše jsem se rozhodl využít metodu HOTP/TOTP 1.1.5, protože dle mého názoru poskytuje velmi vysoký stupeň bezpečnosti oproti ceně implementace a provozu. Ikdyž tato metoda může být méně pohodlná pro uživatele, jelikož musí použít druhé zařízení, zdá se mi jako ideální řešení pro malou firmu.

Autentizaci jsem se rozhodl implementovat na webovou stránku, kterou by tato fiktivní firma využívala pro většinu svých potřeb. Uživatelé (zaměstanci) se poté přihlašují právě přes tuto webovou stránku pomocí dvou faktorové autentizace (klasické heslo + OTP). Pro generaci OTP kódů v této práci využítvám na mobiním zařízení OpenSource aplikaci FreeOTP [26], ale daly by se využít i další aplikace které využívají standardů pro TOTP a HOTP, jako je například Google Authenticator [27].

2.1 Virtuální prostředí VMware

Praktickou stranu práce jsem se, na návrh pana vedoucího, rozhodl dělat ve virtuálním prostředím. To hlavně protože mi to umožní vytvořit a pracovat s virtuálními stroji (Virtual Machine, dále jen VM) podle potřeby bez nutnosti pořízení fyzických strojů, ale také mám s virtualizací minimální zkušenosti a toto je dobrá příležitost se přiučit něčeho nového. K práci mi byl poskytnut přístup do školní virtualizační platformy VMware vSphere. VMware vSphere je cloudová virtualizační platforma. Prakticky je to jeden fyzický server, který dokáže na sobě dokáže vytvářet, propojit a spravovat mnoho VM. Všechny tyto funkce jsou prezentovány v grafickém prostředí, ve kterém jsem byl ze začátku velmi zmatený, ale po nějaké době jsem se v něm začal relativně dobře orientovat. Na obrázku 2.1 můžete vidět jak vypadá klient vSphere.



Obrázek 2.1: Okno webového klienta vSphere

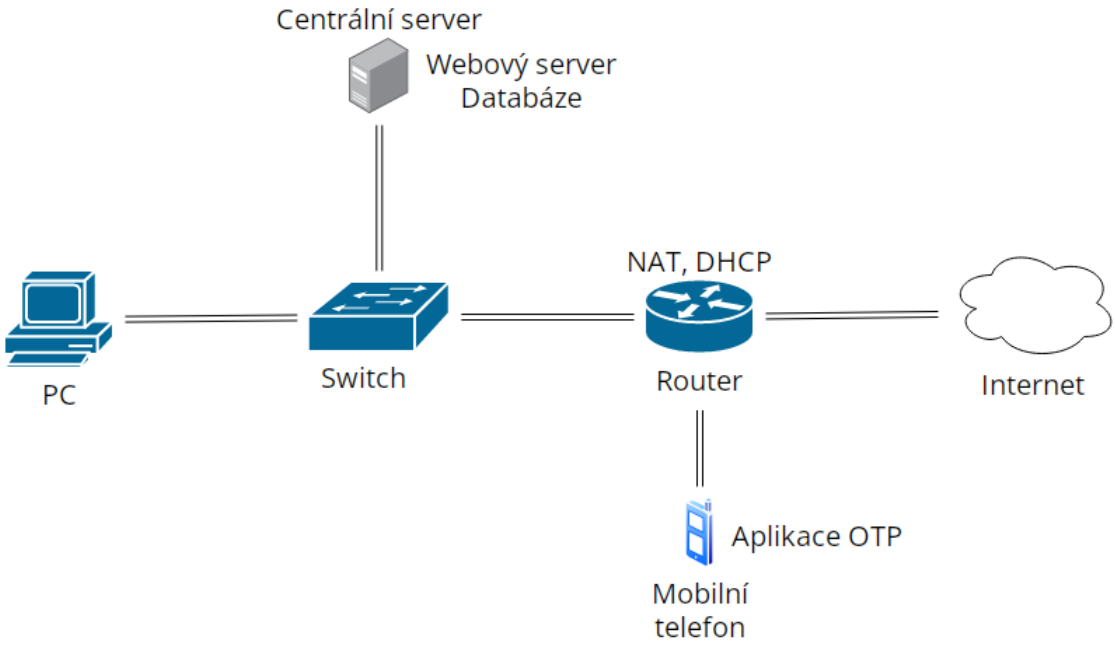
Vytváření VM ve vSphere je relativně jeduduché, dále je zjednodušeně vysvětlen proces:

- 1. Nejprve vyberete výpočetní stroj (server), v mém případě byl jen jeden.
- 2. Dále vyberete cloudové uložiště, v mém případě bylo opět dostupné jen jedno.
- 3. Dalším krokem je zvolit vybraný OS, prakticky se to provádí výběrem instalačního ISO obrazu z uložiště.
- 4. Výběr samotného hardwaru, tedy zvolení počtu jader, velikosti paměti a RAM a různé dalši možnosti jako třeba síťové karty.

Prostředí vShpere dále umožňuje vytváření sítí pro VM. Všechny zařízení v jedné síti jsou přímo připojeny do automaticky vytvořeného virtuálního switche. V případě, že by jsme chtěli propojit více sítí, je potřeba manuálně vytvořit virtuální router.

S vytvořenými VM se poté zachází velmi jednoduše. V prostředí vSphere se vybere zařízení a spustí se webová konzole, přes kterou se dá pracovat jako na fyzickém zařízení.

2.2 Návrh architektury



Obrázek 2.2: Schéma architektury

Na obrázku 2.2 můžete vidět schéma vzorové architektury, kterou jsem pro práci vytvořil. Dále jsou vysvětleny a popsány jeho části:

- Switch - Switch je síťové zařízení, které pracuje na 2. vrstvě OSI modelu [28]. Tedy přeposílá rámce na základě fyzických adres připojených zařízení. Switch jako takový není ve virtualním prostředí samotné zařízení, ale je přímo implementován v systému a poskytuje spojení pro všechy zařízení, které patří do jeho sítě. V této práci jsem s ním tedy přímo nepracoval, ale využil jsem ho tak, že jsem všechna ostatní zařízení přidal do jeho sítě.
- Router - Router je síťové zařízení, které pracuje na 3. vrstvě OSI modelu [28]. Router směruje informace na základě IP adres, ale provádí také další činosti, jako například překlad adres a přidělování IP adres zařízením pomocí systému DHCP. V mé síti je router realizován přes VM na kterém běží OS pfSense 2.6.0. pfSense je operační systém, který umožňuje stroji fungovat jako router. Toto řešení jsem si vybral jelikož je velmi jednoduché a pro potřeby práce mi úplně vystačí.
- PC - PC zde reprezentuje osobní počítače zaměstnanců firmy. V sítí je pro potřeby práce implementován jen jeden, ale v reálné sítí by jich bylo mnoho. V této práci potřebuji PC jen na přístup k webové stránce, proto jsem zvolil OS Windows 10.

- Mobilní telefon - Jak jsem zmínil na začátku kapitoly, mobilní telefon v tomto systému představuje druhý faktor autentizace. Mobilní telefon jsem nakonec v konečném systému neimplementoval virtuálně, to z důvodu že to nebylo nutné. K generaci TOTP kódů v aplikaci FreeOTP stačí nastkenovat QR kód přímo z obrazovky, byl jsem tak schopen k testovaní využít vlastní fyzický telefon. Telefon tedy není přímo připojen k síti, ale musí mít přístup k obrazovce, na které se generuje QR kód.

- Internet - Položka internet v tomto schématu jen reprezentuje připojení k internetu přes router.

- Centrální server - Nejdůležitělší část architektury. Na serveru běží jak webová aplikace, tak autentizace, ale také všechny využité databáze. Pro potřeby práce jsem zvolil OS Linux Debian11 s grafickým rozhraním GNOME, jelikož s ním mám dobré zkušenosti a umím s ním pracovat. Webový server běží přes aplikaci Apache2 a pro databázi jsem si zvolil PostgreSQL (dále jen PSQL), opět protože s nimi mám zkušenosti.

2.3 Návrh webové aplikace

Jak jsem zmínil na začátku kapitoly 2, autentizační systém jsem se rozhodl implementovat na webovou aplikaci. S vytvářením webových aplikací mám již zkušenosti a v dnešní době jsou webové stránky extrémě populárním médiem, což je dle mého názoru tvoří ideálním kandidátem pro demostraci autentizačního systému. Pro vytvoření grafického designu jsem bych mohl využít různé editory, ale pro demostraci znalosti HTML a CSS jsem se rozhodl stránku vytvořit manuálně. Při výběru programovacího jazyku pro kódování aplikace jsem se rozhodoval mezi PHP a Python. Při vytváření webových stránek jsem se setkal a pracoval s oběmi jazyky, ale měl jsem lepší zkušenosti s jazykem PHP, který jsem se proto rozhodl využít pro tuto práci. Webovou aplikaci koncipuji s následujícími funkcemi:

- Registrace - Systém bude schopen registrovat nové uživatele do databáze. Jelikož práci vytvářím pro malou, fiktivní firmu, rozhodl jsem se, že nové uživatele bude registrovat správce sítě manuálně. Správce využije příslušnou stránku a vyplní formulář s informacemi nového uživatele. Tyto infromace jsou uživatelské jméno, heslo a opakované heslo. Ano správce zadává prvotní heslo uživatele, ale ten bude mít možnost si ho později změnit.

- Příhlášení - Registrovaní uživatelé budou schopni se do systému příhlásit skrz tuto webovou stránku pomocí uživatelského jména, hesla a TOTP. Postup příhlášení bude: Uživatel zadá své jméno a heslo a stiskne tlačítko. Systém porovná zadané informace s databází a buď přihlásí uživatele či odmítne. Uživatel je v této fázi přihlášen, ale stále nebude mít žádné pravomoce, jelikož neprošel 2. faktorem autentizace. V okénku přihlašovacího okna se objeví nová kolonka, do které se vkládá TOTP kód.

Uživatel využije svého mobilního zařízení a vloží vygenerovaný TOTP kód. Systém dále ověří platnost kódu a buď uživatele autentizuje a přidá mu jeho pravomoce, nebo ho odmítne.

- Vytváření informací pro TOTP autentizaci - Systém musí být schopen přesunout informace nutné pro generování TOTP kódu do příslušné aplikace na mobilním zařízení. Správce sítě bude schopen, přes tuto webovou aplikaci, vytvořit QR kódy, které budou obsahovat nutné informace. Příslušný uživatel poté tento QR kód naskenuje do své aplikace. Administrátor na stejné stránce uloží identické informace do databáze pro budoucí ověření kódů. Uložené informace se budou dát změnit v případě, že uživatel bude potřebovat nové.

- Různé pravomoce - Uživatelé mají ve svém záznámu v databázi také tzv. *urověň pravomoce*, nebo jen pravomoce. Tento údaj může být využit k umožnění přístupu k určitým stránkám či zdrojům uživatelům se specifickými pravomocemi.

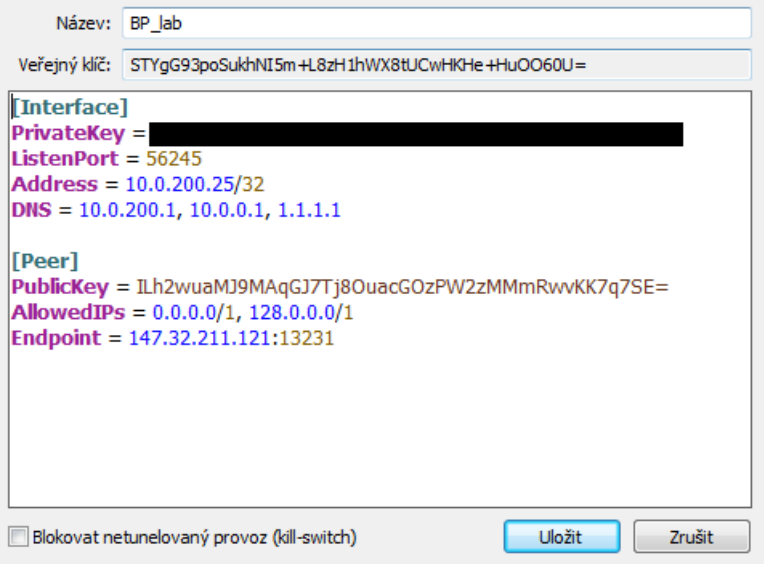
- SSO přes celou aplikaci - Aplikace bude využívat druhou metodu bezheslové autentizace pro přenos autentizace na všechny stránky. Toto znamená, že uživatel se bude moci přihlásit jen jednou a pokračovat na všechny stránky v aplikaci bez nutnosti další autentizace. Pro tuto funkci budu využívat nativní funkci PHP a to PHP Session 1.1.3. Systém bude nastaven tak, že si bude pamatovat údaje se kterými se uživatel úspešně přihlásil po dobu 30 minut. Pokud upline 30 minut bez toho aby se uživatel přihlásil nebo změnil stránku, session bude zahozena a uživatel se bude muset přilhásit znovu.

Kapitola 3

Postup práce

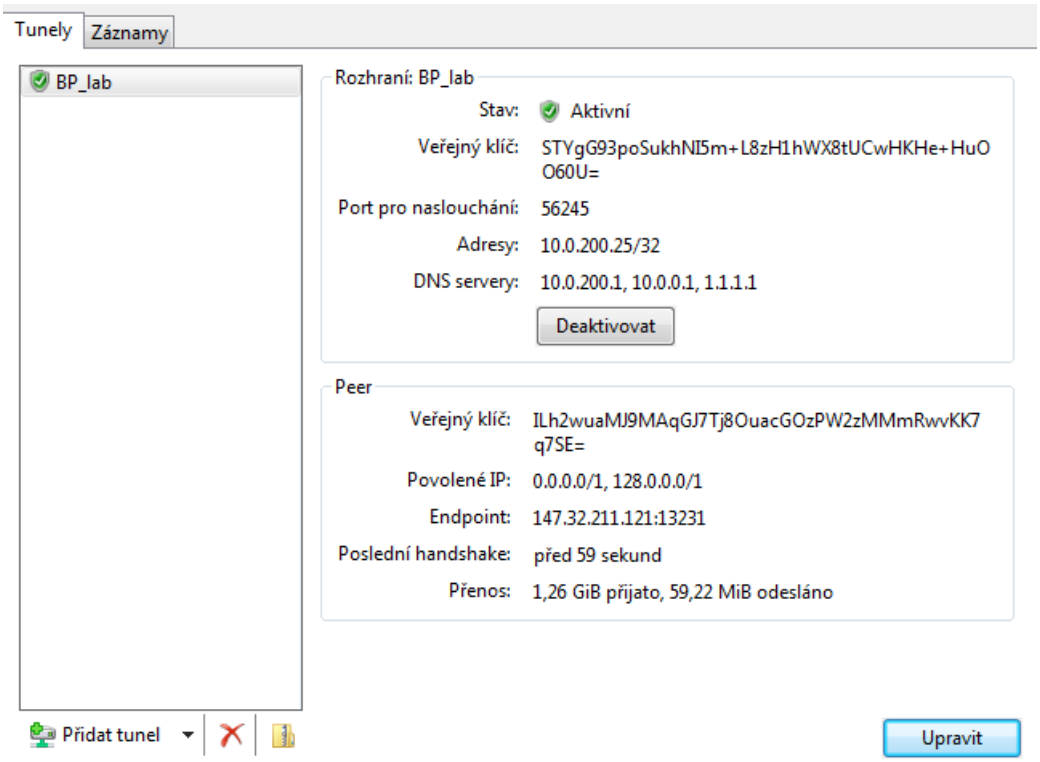
3.1 Připojení k virtuálnímu prostředí vSphere

Jak bylo zmíněno v sekci 2.1, pro potřeby práce mi byl udělen přístup do školní virtualizační platformy vSphere. Obdržel jsem instrukce na připojení od správce sítě, pana Marcela Poláčka. Pro připojení k platformě je nutno využít VPN, proto jsem musel stáhnout a využít aplikaci WireGuard, která posyktuje velmi jednoduchý, ale velmi bezpečný OpenSource VPN protokol, využívající nejmodernějších kryptografických metod[29].



Obrázek 3.1: Okno vytvoření tunelu (soukromý klíč odstraněn)

Tuto aplikaci jsem používal poprvé, ale velmi rychle jsem se v ní zorientoval. Na základě poskytnutých údajů jsem vytvořil tunel a podařilo se mi připojit k úvodní stránce, ale nedařilo se mi přejít na další stránku. Nakonec se ukázalo, že problém byl s DNS překladem na straně serveru, který správce rychle opravil. Dále jsem byl schopen se úspěšně přihlásit do systému. Na obrázku 3.1 je vidět vytvoření tunelu a na obrázku 3.2 je vidět okno WireGuard při připojení.

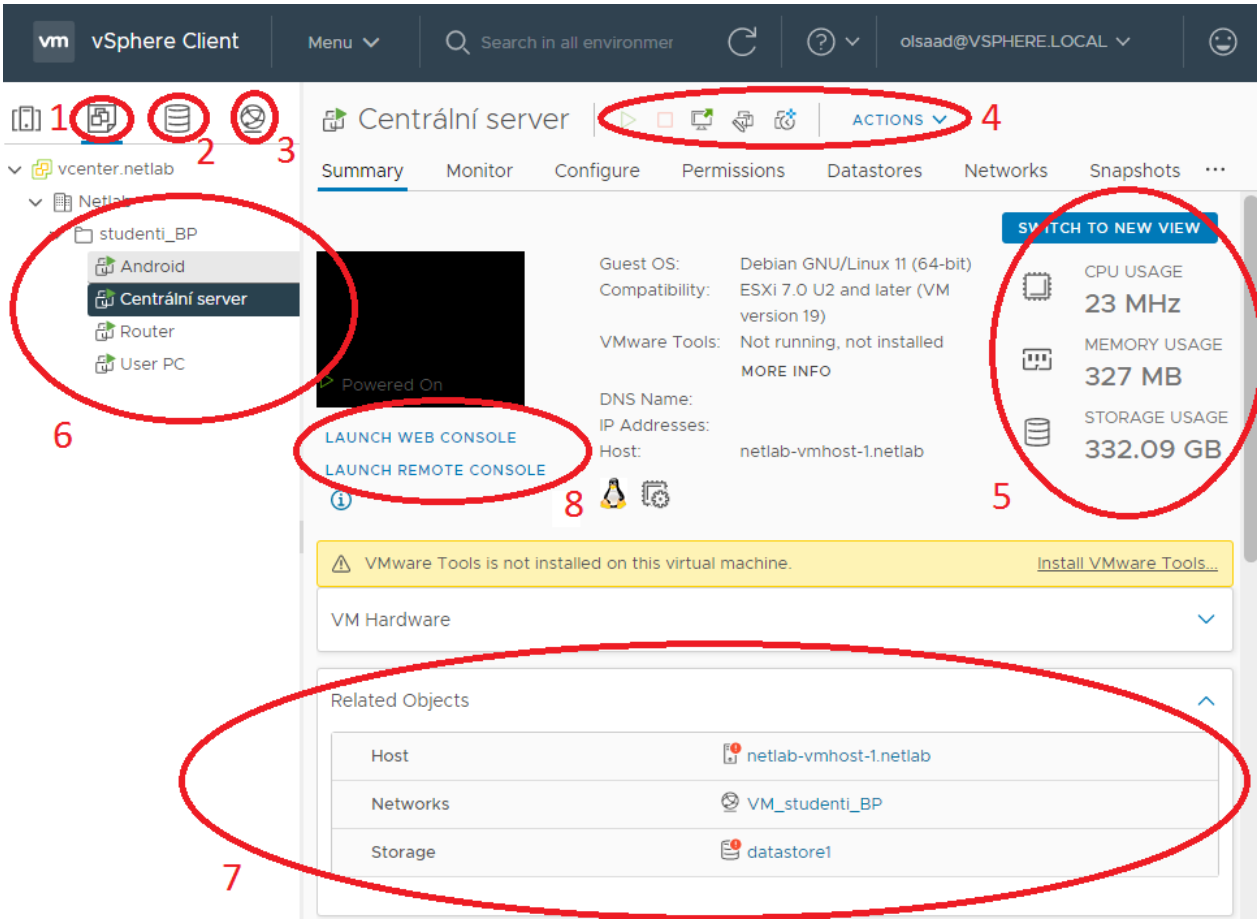


Obrázek 3.2: Okno WireGuard

3.2 Práce ve virtuálním prostředí

Když jsem se poprvé rozhlédl v rozhraní virtuálního prostředí tak jsem byl velmi zmaten a trvalo mi nějakou dobu než jsem se v rozhraní zorientoval. Vůbec nepomohlo, že systém neobsahoval žádný návod, a všechny části nebyly pořádně popsané. Na internetu jsem sice našel nějaké užitečné informace, ale všechny návody které jsem našel měly jiné rozhraní, než na které jsem se koukal já. Chci tady tedy prezentovat krátký orientační návod pro právě toto rozhraní.

Na obrázku 3.3 jsem vyznačil části, které jsem používal nejvíce a zdají se mi jako nejdůležitější. Následuje vysvětlení:



Obrázek 3.3: Referenční okno vSphere

- V záložce č. 1 najdete všechny vaše VM, rozdělené do složek, např. v kroužku č. 6 je vidět moje záložka *studenti_BP* a mé čtyři VM. V této záložce se také vytvářejí nové VM, stistkutím právého tlačítka na složku a vybráním *New Virtual Machine*.
- V záložce č. 2 najdete seznam dostupných uložišť. Vždy při vytvoření VM vybíráte do jakého uložiště bude zapadat, z vybraného uložiště nové

VM získávají ISO obrazy pro instalaci příslušných OS a berou si z něj dostupnou paměť. V této práci jsem musel na svém osobním PC stáhnou potřebné ISO obrazy a dále je nahrát na toto uložiště, abych je potom mohl využít na nové VM.

- V záložce č. 3 jsou vidět dostupné virtuální sítě, které jsou potřeba aby vytvořené VM dokázaly mezi sebou komunikovat. Tuto záložku jsem příliš nepoužíval, ale chápu jak může být důležitá pro rozsáhlejší sítě.
- Když je vybrána nějaká VM, máte přístup k liště č. 4. Tato lišta se používá k ovládání VM, umožňuje např. vypnout či zapnout stroj, nebo změnit jeho hardware.
- V části č. 5 je vidět současné využití hardwaru. Důležité pro kontrolu jestli VM není přežížený.
- V okně č. 7 jsou vidět relevantní objekty s kterými je VM propojena. Např. zde můžete vidět že tato VM je připojena k síti *VM_studenti_BP*, nebo že je přiřazen do uložiště *datastore1*.
- Tlačítka v sekci č. 8 jsou využita pro otevření konzole vybrané VM, což umožňuje na něm pracovat jako na normálním zařízením.

3.3 Vytváření VM

Prvním krokem práce bylo vytvoření jednotlivých VM ve vSphere.

3.3.1 Router

První zařízení které jsem vytvořil byl router, to proto aby se usnadnila instalace pro zbytek zařízení. Na router nejsou velké hardwarové požadavky, proto jsem zvolil 2 jádra, 2GB RAM a 10GB paměť. Všechna zařízení mají po vyvoření jen 1 síťový adaptér a pokud potřebujete druhý, jako já zde, musíte ho přidat manuálně v nastavení. Postup instalace OS pfSense byl velmi jednoduchý, po nastartování stroje jsem v příkazové řádce nastavil LAN a WAN porty a zapnul DHCP distribuci do LAN sítě.

3.3.2 PC

Jelikož PC je zde využit jen pro připojení k serveru přes webové rozhraní, tak na něj nejsou velké hardwarové požadavky. Zvolil jsem proto 4 jádra, 8GB RAM a 48GB paměti, bez dalších přídavků. Instalace OS Windows 10 je obecně velmi jednoduchá, a zde tomu nebylo jinak.

3.3.3 Centrální server

Server má, jakožto nejdůležitější část sítě, největší požadavky na hardware. Zvolil jsem pro něj 8 jáder, 32GB RAM a 300GB paměti. S výkonem serveru jsem při práci neměl problém a nezabral jsem jím tolik zdrojů virtualizační platformy.

3.4 Příprava systému

V přípravě na implementaci autentizačního systému jsem musel na serveru připravit několik systémů a aplikací. Jako první jsem musel stáhnout a nainstalovat aplikaci pro provoz webového serveru. K této funkci jsem vybral Apache2, což je nejrozšířenější OpenSource HTTP server [30]. S Apache jsem měl již zkušenosti a jeho nastavnení a zprovoznění nebyl velký problém.

Dále jsem připravil databáze a tabulky, které potřebuji pro ukládání uživatelských a autentizačních údajů. Vytvořil jsem jednu databázi pro své potřeby s názvem *db_bp*. V této databázy jsem dále vytvořil dvě tabulky potřebné pro autentizaci uživatelů. První z nich je tabulka *users*, která obsahuje 6 sloupců:

- *username* - Sloupec username obsahuje uživatelské jméno, jeho datový typ je varchar(255) a také slouží jako primární klíč tabulky.
- *perms* - Kolonka perms obsahuje číslo, které reprezentuje pravomoce daného uživatele, jeho datový typ je smallint. Např. číslo 10 je využito

pro správce. Díky této kolonce může firma mít větší kontrolu nad tím kam má určitý uživatel(zaměstnanec) přístup.

- *password* - Položka password obsahuje hashované heslo uživatele. Heslo má podobu, která může vypadat takto: 2*y*10$Wn2S3T1h/KmlrPrMwQ/U/eYKCz.DjflpgKkbFLk9fMh8qj92Ju4FK. Jeho datový typ je varchar(127)
- *salt* - Sloupec salt je typu bytea a obsahuje náhodně generované hexadecimální číslo, které se využívá při hashování hesla pro zvýšení bezpečnosti.
- *created* - V tomto sloupci je číslo reprezentující čas založení účtu v podobě UNIX Timestamp[1]. Jeho datový typ je bigint.
- *last_login* - V posledním sloupci je obsažen čas posledního přihlášení uživatele, opět ve formátu UNIX Timestamp. Jeho datový typ je také bigint. Tato položka umožňuje accounting.

Druhá tabulka s názvem *auth_data* obsahuje informace potřebné k provedení autentizaci přes TOTP. Tato tabulka také obsahuje 6 sloupců:

- *username* - Stejné jako v tabulce users.
- *algorithm* - Algorithm říká jaký algoritmus byl použit pro generaci tajného klíče, např. "SHA1", "SHA256"nebo další. Jeho datový typ je varchar(31).
- *method* - Tato položka udává jestli se k autentizaci používá TOTP či HOTP (systém zatím podporuje jen TOTP, ale existuje pro možnost rozšíření). Datový typ je varchar(7).
- *key* - Sloupec key obsahuje tajný klíč autentizačního tokenu. Jeho datový typ je varchar(255) jelikož klíče jsou typicky 32bytové v hexadecimální formě, ale mohou být kratší i delší.
- *digits* - Kolonka digits udává jak dlouhé budou generované OTP, typicky bývá 6 či 8. Jeho datový typ je varchar(3).
- *time* - Ve sloupci time je pro metodu TOTP uložena doba mezi změnou OTP. Pro metodu HOTP by v této kolonce byl inkrementační čítač. Datový typ je varchar(7).

[1] Množství vteřin od 1.1.1970 UTC

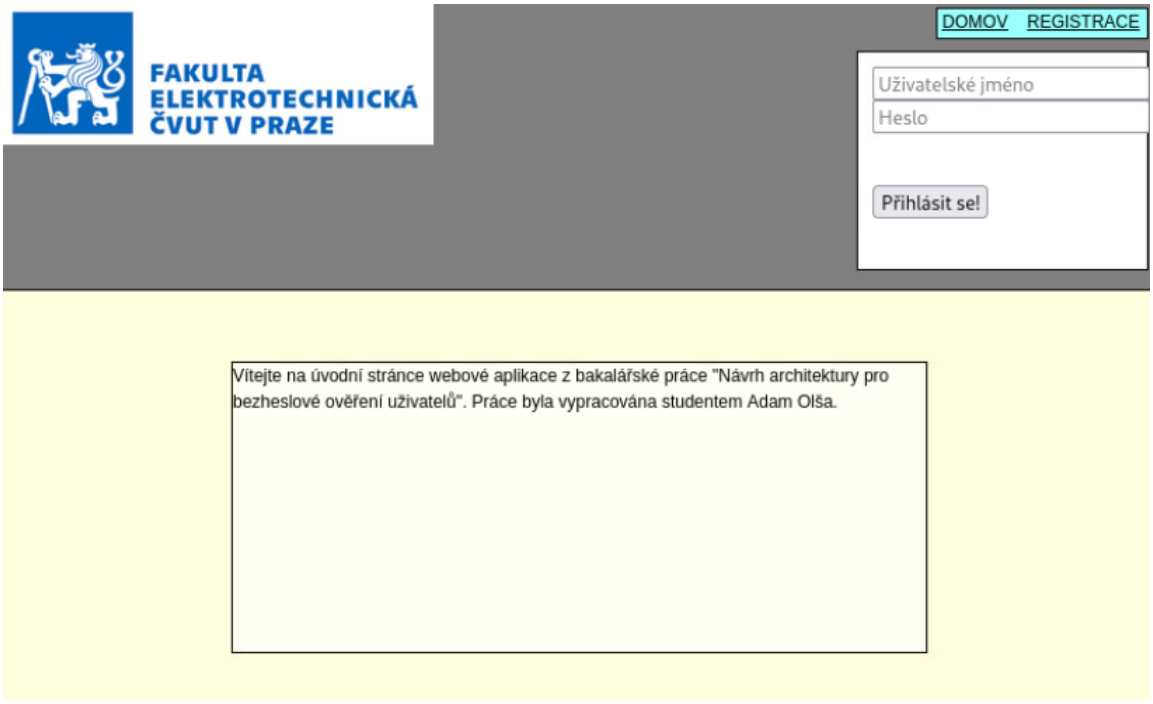
3.5 Vytvoření webové aplikace

Poslední částí práce bylo vytvoření samotné webové aplikace. Pro vytvoření celé aplikace jsem využil několika programovacích jazyků. Těmito jazyky jsou:

- HTML - HyperText Markup Language je standartní jazyk pro vytváření grafického rozložení webových stránek. V této práci je tento jazyk využit pro vytvoření jednotlivých objektů v aplikace (textové bloky, odkazy, obrázky.
- CSS - Cascading Style Sheets je přídavný modul pro HTML, který umožňuje snažší a rozsáhlejší stylizaci vzhledu a vlastností HTML objektů jako např.: Velikost, pozice, barva a další. Všechny využíté styly jsou sepsány ve zdrojovém souboru *style.css* (odkaz na přílohu). Na tento soubor se poté každá stránka odkazuje pomocí příkazu *<link href="style.css" rel="stylesheet">*.
- PHP[2] - Hypertext Preprocessor je jazyk velmi podobný programovacímu jazyku C, ale přizpůsoben pro webové stránky. Využívá se pro vytvoření dynamckých programů právě na webových stránkách. Všechen PHP kód je prováděn na stránce serveru nikoly na straně uživatele jako HTML nebo JavaScript. V této práci provádím většinu funkcí aplikace pomocí právě tohoto jazyku. Nějaké z fukncí jsou např.: registrace, přihlašování nebo práce s databází. Pro práci s databází jsem používal knihovnu PHP: PSQL, která obsahuje funkce na přímou práci s PSQL knihovnou. Tyto funkce umožňují provadět tzv. *Sanitized input*, což prakticky znamená, že komunikace s databází je ochráněna před tzv. *SQL injection* útoky. Tyto typy útoků se snaží využít neochráněných vstupů do databáze vložením vlastních příkazů, čímž mohou zobrazit, změnit či smazat data v ní.
- JS - JavaScript je poslední z využitých jazyků. JavaScript se odlišuje od ostatních jazyků v tom, že umožňuje změnu prvků webové stránky v reálném čase (tj. bez komunikace se serverem). Tento jazyk jsem využil pro vytváření QR kódů na stránce *qr_code.php*.

S pomocí všech těcho jazyků jsem vytvořil všechny části celé aplikace. Prvním krokem bylo navrhnout jednoduché ale přehledné grafické rozhraní stránky. S využitím HTML a CSS jsem vytvořil návrh stránky, který můžete vidět na obrázku 3.4. Součástí každé stránky je logo fakulty v levém horním rohu, jednoduché menu v pravém horním rohu a okno přihlášení hned pod ním. Ve spodní části rozhraní se poté nachází tělíčko stránky, které se mění podle současné stránky.

[2]Zkratka PHP je pozůstalá z původního názvu "Personal Home Page"



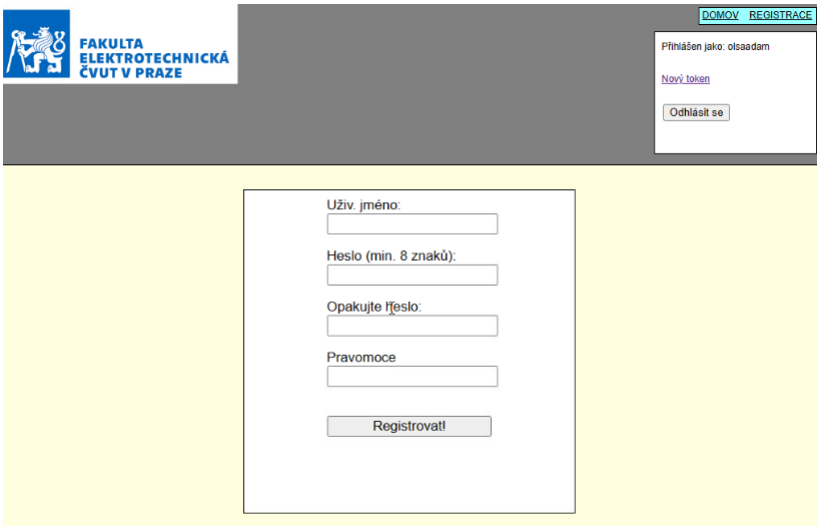
Obrázek 3.4: Úvodní stránka

3.5.1 Vytvoření jednotlivých stránek

Celá aplikace se složena z několika jednotlivých stránek, z nichž každá provádí nějakou funkci, ale nějaké části kódu jsou sdílené přes všechny stránky. Tyto části se zabívají hlavně přihlašováním a PHP Session. Uživatel je schopen se přihlásit na jakékoliv stránce a po úspěšném přihlášení se informace o jeho přihlášení uloží do jeho PHP Session. Při načtením nové stránky se kontroluje pokud uživatel má platnou PHP Session. Pokud ano tak je uživatel automaticky přihlášen bez dalšího dotázání. V další části následuje seznam všech vytvořených stránek a jejich funkce v aplikaci:

- *index.php* - Index je standardní název pro úvodní stránku, jelikož je na ni uživatel automaticky přesměrován pokud přímo neposkytne název stránky. Tato stránka není nijak zajímavá, jediná věc která ji osamucuje od ostatních je úvodní textové okno. Tuto stránku jsem vytvořil jako první a dále jsem ji používal jako šablonu pro ostatní.
- *test.php* - Tato stránka je identická kopie index.php, ale při autentizaci vypisuje informace, s kterými systém pracuje. Tato stránka bude sloužit v lab. úloze, kde bude možné prohlédnout do systému generace TOTP kódů.
- *registrace.php* - Stránka registrace působí jako rozhraní pro správce, kde může vytvářet nové účty zaměstnanců. Vytváření účtů spočívá ve vyplnění předloženého formuláře, který obsahuje kolonky pro uživatelské jméno,

heslo a pravomoce. Pokud byly vyplněny platné informace, uživatel je registrován a jeho informace jsou uloženy v databázi. Na obrázku 3.5 můžete vidět jak tato stránka vypadá.



Obrázek 3.5: Stránka registrace

- *qr_code.php* - Tato stránka je posledním komponentem aplikace. Její hlavní částí je skript napsaný v JS, který je schopen vytvářet QR kódy . Tento skript jsem převzal ze zdrojového kódu, který vyrvořili autoři FreeOTP, ze stránky https://github.com/freeotp/freeotp.github.io. Vlastní práce v na této stránce je přispusobení do mého rozhraní a přidání tlačítka pro uložení nutných informací do databáze. Na obrázku 3.6 ukázána demostrace vytvoření takového QR kódu.

Obrázek 3.6: Okno generace QR kódu

3.6 Proces bezheslové autentizace

Hlavní podstatou této práce bylo vytvořit systém, který je schopen bezheslové autentizace uživatele. Bezheslové autentizace je v aplikaci dosaženo kombinací dvou metod.

První a hlavní z nich je zvolená metoda TOTP, která je implementována jako 2. faktor k tradičnímu heslu. První fází autentizace je velmi standardní vložení uživatelského jména a hesla. Pokud tato fáze proběhne úspěšně, uživatel je přihlášen do systému, ale stále není autentizován[3]. Uživatel který je přihlášen, ale neautentizován nemá žádné pravomoce. Aby byl uživatel autentizován, musí vložit platný TOTP kód do kolonky, která se objeví po přihlášení. Tyto TOTP kódy jsou generovány v libovolné TOTP aplikaci na mobilním zařízení. Vložené kódy jsou poté ověřeny na straně serveru. K ověření kódů využívám volně dostupnou PHP knihovnu PHP Gangsta pod licencí BSD [31]. S využitím této knihovny jsem měl menší problémy, hlavně protože mi dlouhou dobu generovala nesprávné kódy. Řešení tohoto problému my dalo hodně práce a čtení ve zdrojovém kódu knihovny. Nakonec se mi podařilo zjistit, že knihovna neumí pracovat s malými písmeny v klíči a předpokládá, že jsou vkládány jen velké. Po jednoduché úpravě vlastního kódu, který s knihovnou zachází se mi podařilo vygenerovat platné kódy.

Systém momentálně podporuje jen následující nastavení autentizačních kódů: algoritmus SHA1, počet znaků 6, metoda TOTP, interval 30s, ale je vytvořen tak, že by se v budoucnu daly imlementovat i ostatní možnosti nabízené na stránce qr_code.php.

Druhou metodou autentizace je využití PHP Session, pro udržení autentizace na krátkou dobu. Implementace této metody znamená, že se uživatel nemusí přihlašovat při načtení nové stránky v aplikaci, nebo pokud ze stránky odejte a pak se vrátí.

3.7 Laboratorní úloha

V rámci práce jsem také umožnil vzdálený přístup na PC ve virtuální síti, takže si tento typ autentizace může vyzkoušet každý. V přílohách jsem také zahrnul krátký návod či lab. úlohu k přípojení k PC a přihlášení na webovou stránku. V ukázkové úloze si studenti mohou prohlédout jaké informace jsou potřeba pro vytvoření a ověření TOTP kódů. Pro vzdálený přístup jsem využil aplikace TeamViewer, která umožňuje stálý a neomezený přístup k zařízení na kterém je spuštěn. Zájemci si musí vytvořit účet TeamVeiwer a buď stáhnout aplikaci na vlastní počítač nebo využít jejich webové aplikace pro připojení.

[3] *Autentizován* zde používám jakožto rozlišný termín od *přihlášen*, pro rozlišení jestli uživatel prošel 2. faktorem.

Závěr

V práci bylo vysvětleno, že klasická hesla nejsou v dnešní době dostatečně bezpečná, jsou náchylná na velké množství útoků nebo jiných metod odcizení. Dále v ní bylo popsáno velké množství stávajících metod bezheslové autentizace a jak přesně zvyšují bezpečnost nebo zlehčují život oproti heslům. V poslední části bylo předvedeno, jak může takový systém využívající bezheslových metod vapadat. Systém bezheslové autentizace byl úspěšne navržen a implementován a je plně funkční. Systém je schopen registrovat nové uživatele a uživatelé se dokážou spolehlivě přihlásit pomocí hesla a svého mobilního zařízení.

Příloha A

Bibliografie

[1] Fortinet. *What is Authentication, Authorization, and Accounting (AAA)?* URL: https://www.fortinet.com/resources/cyberglossary/aaa-security. Dne 4.5.2023.

[2] Polybius. *The Histories*. Kansas City: Digireads.com, 2009. ISBN: 978-1420934236.

[3] Brad Woodward. *High-Power Hash Cracking with NPK*. 2021. URL: https://www.coalfire.com/the-coalfire-blog/high-power-hash-cracking-with-npk. Dne 20.11.2022.

[4] Jonathan Lampe. *Beyond password length and complexity*. 2014. URL: https://resources.infosecinstitute.com/topic/beyond-password-length-complexity/. Dne 20.11.2022.

[5] Justina Alexandra Sava. *Average number of characters for a password in the United States in 2021*. 2022. URL: https://www.statista.com/statistics/1305713/average-character-length-of-a-password-us/. Dne 20.11.2022.

[6] Xinyi HUANG a Gaopeng JIAN WANG Ding; Haibo CHENG; Ping WANG. "Zipf's Law in Passwords". In: *IEEE Transactions on Information Forensics and Security* 12 (11 2017), s. 2776–2791. DOI: 10.1109/TIFS.2017.2721359. Dne 20.11.2022.

[7] Aviad Mizrachi. *Understanding Token-Based Authentication: A Detailed Review*. 2021. URL: https://frontegg.com/blog/token-based-authentication. Dne 14.1.2023.

[8] OneLogin. *How Does Single Sign-On Work?* 2022. URL: https://www.onelogin.com/learn/how-single-sign-on-works. Dne 13.1.2023.

[9] OneLogin. *OneLogin*. 2022. URL: https://www.onelogin.com/. Dne 13.1.2023.

[10] ČVUT. *Federace CVUTID*. 2022. URL: https://ist.cvut.cz/pojmy/federace-cvutid/. Dne 13.1.2023.

[11] MDN Web Docs. *Using HTTP cookies*. 2022. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies. Dne 13.1.2023.

[12] Chameera Dulanga. *Web Authentication: Cookies vs. Tokens*. 2021. URL: https://blog.bitsrc.io/web-authentication-cookies-vs-tokens-8e47d5a96d34. Dne 13.1.2023.

[13] W3Schools. *PHP Sessions*. URL: https://www.w3schools.com/php/php_sessions.asp. Dne 21.1.2023.

[14] Kisi. *Keycards for Access Control Systems*. URL: https://www.getkisi.com/keycard-access-systems. Dne 14.1.2023.

[15] Toby Best. *What is the Wiegand Effect?* 2021. URL: https://www.linkedin.com/pulse/what-wiegand-effect-toby-best. Dne 14.1.2023.

[16] OneLogin. *What's the Difference Between OTP, TOTP and HOTP?* URL: https://www.onelogin.com/learn/otp-totp-hotp. Dne 14.1.2023.

[17] Advantage Services. *What are security keys and what are the pros and cons to using them?* 2020. URL: https://advantageservices.net/what-are-security-keys-and-what-are-the-pros-and-cons-to-using-them/. Dne 14.1.2023.

[18] FIDO Alliance. *Universal 2nd Factor (U2F) Oveview*. URL: https://fidoalliance.org/specs/u2f-specs-master/fido-u2f-overview.html. Dne 21.5.2023.

[19] SecSign. *Why USB Authentication Keys and Tokens are a Bad Idea*. 2015. URL: https://www.secsign.com/usb-authentication-keys-tokens-bad-idea/. Dne 14.1.2023.

[20] Yubico. *What is Certificate-Based Authentication?* URL: https://www.yubico.com/resources/glossary/what-is-certificate-based-authentication/. Dne 15.1.2023.

[21] Dovell Bonnett. *CERTIFICATE AUTHENTICATION IS VULNERABLE*. 2015. URL: https://www.govloop.com/community/blog/certificate-authentication-vulnerable/. Dne 15.1.2023.

[22] Henry Cazalet. *What is SMS OTP? A simple guide for 2022*. 2022. URL: https://thesmsworks.co.uk/blog/sms-otp/. Dne 15.1.2023.

[23] Henry Cazalet. *Is SMS encrypted?* 2022. URL: https://thesmsworks.co.uk/blog/is-sms-encrypted/. Dne 15.1.2023.

[24] James Rose. *How secure is email? Hint: not secure enough*. 2022. URL: https://contentsnare.com/how-secure-is-email/. Dne 15.1.2023.

[25] OneLogin. *Biometric Authentication, the Good, the Bad, and the Ugly*. URL: https://www.onelogin.com/learn/biometric-authentication. Dne 21.1.2023.

[26] Red Hat Software. *FreeOT Two-Factor Authentication*. URL: https://freeotp.github.io/. Dne 18.5.2023.

[27] Matouš Vondrák Martin Doležal. *Co je to Google Authenticator a jak funguje? Návod pro rok 2023*. URL: https://finex.cz/co-je-to-google-authenticator-a-jak-funguje-navod/. Dne 18.5.2023.

[28] Imperva. *What Is the OSI Model*. URL: https://www.imperva.com/learn/application-security/osi-model/. Dne 12.5.2023.

[29] Jason A. Donenfeld. *WireGuard: Next Generation Kernel Network Tunnel*. URL: https://www.wireguard.com/papers/wireguard.pdf. Dne 12.5.2023.

[30] Apache Software Foundation. *APACHE HTTP SERVER PROJECT*. URL: https://httpd.apache.org/. Dne 18.5.2023.

[31] @PHPGangsta Michael Kliewe a contributors. *Google Authenticator PHP class*. URL: https://github.com/PHPGangsta/GoogleAuthenticator. Dne 22.5.2023.