Univerzita Hradec Králové

Fakulta informatiky a managementu

BAKALÁŘSKÁ PRÁCE

2022 **Tomáš Valenta**

Univerzita Hradec Králové

Fakulta informatiky a managementu

Optimalizace správy Active Directory za využití Windows PowerShell

Bakalářská práce

Autor: Tomáš Valenta

Studijní obor: Aplikovaná informatika, kombinovaná forma

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

Hradec Králové 29. dubna 2022

Prohlášení

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 29. dubna 2022

_____

podpis

Poděkování

Děkuji vedoucímu bakalářské práce panu Mgr. Josefu Horálkovi, Ph.D. za metodické vedení práce a připomínky při jejím zpracování.

Anotace

Cílem bakalářské práce je navrhnout a implementovat optimalizaci správy uživatelské identity v Active Directory pomocí Windows PowerShell za využití procesů a postupů dle ITIL 2011. Autor v teoretické části popíše všechny procesy ITIL 2011, základy služby Active Directory a nástroje Windows PowerShell použité v praktické části. V praktické části autor navrhne a implementuje několik scriptů, které budou využity jako step-by-step návod pro studenty předmětu OS1.

Klíčová slova:

Active Directory, ITIL 2011, Windows PowerShell

Annotation

Title: Optimize Active Directory via Powershell

The aim of the Bachelor Thesis is to design and implement optimization of user identity management in Active Directory via Windows PowerShell using processes and procedures by ITIL 2011. The author describes in the theoretical part all ITIL 2011 processes, Active Directory basics and Windows Powershell tools used in the practical part. In the practical part the author will design and implement several scripts that will be used as a step-by-step guide for OS1 students.

Keywords:

Active Directory, ITIL 2011, Windows PowerShell

Obsah

1 Úvod

Cílem bakalářské práce je navrhnout a implementovat optimalizaci správy Active Directory pomocí Windows PowerShell za využití procesů dle ITIL 2011.

Autor v první kapitole teoretické části popíše všech 26 procesů z 5ti knih ITIL 2011, v kapitole Active Directory bude popsáno rozdělení na fyzickou a logickou strukturu, objekty Active Directory a skupinové politiky. V poslední kapitole bude vysvětleno, kde najít a jak spustit PowerShell, spouštění scriptů pomocí zásad spuštění, příkaz pro nápovědu, rozdíl oproti předchozím shellům, proměnné, podmínky, cykly a vyjímky.

V praktické části autor pro každé laboratorní cvičení nejprve provede analýzu dané situace, poté navrhe způsob implementace, kterou následně implementuje a popíše. Laboratorní cvičení budou vycházet z procesů ITIL 2011 a budou využity jako step-by-step postup pro studenty předmětu OS1. Pro laboratorní úlohy bude vytvořeno virtuální prostředí v aplikaci VMware Workstation 15 Pro, na kterém bude nainstalováno několik počítačů s operačním systémem Windows 10 a Windows Server 2019, který bude povýšen na doménový řadič a bude na něm nainstalována služba Active Directory. Virtualizace a zprovoznění Active Directory na Windows Serveru není předmětem této práce, jak nainstalovat Active Directory na Windows Server je možné zjistit v bakalářské práci od Filipa Antoše (Antoš 2018 [cit. 2022-04-08]). Kompletní scripty, vytvořené v laboratorních cvičení bude možné nalézt v repozitáři na GitHubu.

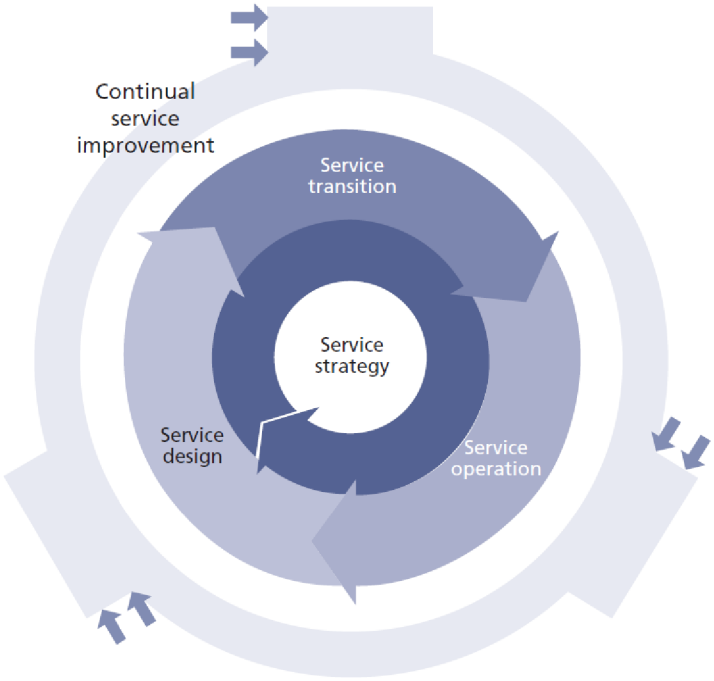
2 ITIL

ITIL je označován, jako rámec, který představuje rozsáhlý a všeobecně dostupný návod pro správu IT služeb.

Ačkoliv je v době psaní práce k dispozici novější verze označována jako ITIL V4 z roku 2019, rozhodl se autor zvolit pro psaní své práce starší verzi označovanou jako ITIL 2011 Edition.

ITIL Edice 2011 se skládá z 5-ti knih tvořících celý životní cyklus služby. V knihách je dohromady popsáno 26 základních procesů, které je možné kombinovat nebo implementovat pouze některé z nich. V této části bude popsáno všech 26 procesů. Procesy, na které se váže uživatelská identita(2.2.7, 2.4.1, 2.4.5) budou využity v laboratorních cvičeních.

Obrázek 1: Životní cyklus služby dle ITIL



Zdroj: (The Cabinet Office 2011d)

2.1 Service Strategy

První kniha ITIL Edice 2011 (The Cabinet Office 2011d) se zabývá Strategií služeb, která funguje jako osa celého životního cyklu služby a ovlivňuje všechny ostatní fáze.

2.1.1 Strategy management for IT services

Strategické řízení pro IT služby je proces, který definuje a udržuje perspektivu, pozici, plán a vzorce chování organizace s ohledem na její služby a správu.

2.1.2 Service portfolio management

Druhým procesem je řízení portfolia služeb, které používá poskytovatel k sledování veškerých investic do služeb prostřednictvím jejich životního cyklu vývoje, dodávky, až k vyřazení služby z nabídky poskytovaných služeb.

2.1.3 Financial management for IT services

Správa financí služeb IT je zodpovědná zajistit odpovídající úroveň financování pro návrh, vývoj a poskytování služeb, které splňují strategii organizace. Zároveň zajišťuje, že se poskytovatel služeb nezaváže ke službám, které není schopen poskytovat.

2.1.4 Demand management

Účelem procesu správy požadavků je pochopit, předvídat a ovlivňovat poptávku zákazníků po službách, aby včas a hospodárně zajistil kapacity pomocí správy kapacit (Capacity Management). Ve spolupráci se zákazníkem by mělo plánování, analýza trendů, dohoda o úrovni služeb (SLA) a řízení snížit nejistotu a výkyvy v poptávce.

2.1.5 Business relationship management

Posledním procesem v knize Service Strategy je správa vztahů s byznysem (BRM). Má dvojí účel, jednak vytvořit a udržovat obchodní vztah mezi poskytovatelem služeb a zákazníkem a také identifikovat potřeby zákazníků a zajistit, aby poskytovatel služeb byl schopen tyto potřeby uspokojit.

2.2 Service Design

Druhá kniha ITIL Edice 2011 (The Cabinet Office 2011b) se zabývá fází Návrhem služby, účelem služby je návrh hodnotových služeb v souladu se strategií. Návrh služby tvoří společně s přechodem a provozem služby vnitřní životní cyklus služby.

2.2.1 Design Coordination

Proces koordinace návrhu má za úkol zajistit splnění cílů a záměrů fáze návrhu služby za pomoci konzistentních a proveditelných požadavků a doporučení pro různé aspekty návrhu.

2.2.2 Service Catalogue Management

Katalog služeb poskytuje a udržuje spolehlivý zdroj konzistentních informací o všech dostupných službách a těch, které jsou připraveny k provozu.

2.2.3 Service Level Management

Ve správě úrovně služeb se zajišťuje, aby všechny současné a plánované IT služby byly dodávány podle dohod. Service Level Agreement (SLA) je dohoda mezi poskytovatelem služeb a zákazníkem. Operation Level Agreement (OLA) je dohoda mezi poskytovatelem a jiným oddělením v rámci jedné organizace.

2.2.4 Availability Management

Správa dostupnosti definuje, analyzuje, plánuje, měří a vylepšuje všechny aspekty dostupnosti IT služeb a zajišťuje, aby služby odpovídaly schválené úrovni dostupnosti.

2.2.5 Capacity Management

Správa kapacit zajišťuje, aby služby IT a infrastruktura splňovaly nákladovou kapacitu a výkonnostní požadavky včasným a efektivním způsobem. Je klíčová pro plánování výkonnostních nároků a pro budoucí růst bussinesu.

2.2.6 IT Service Continuity Management

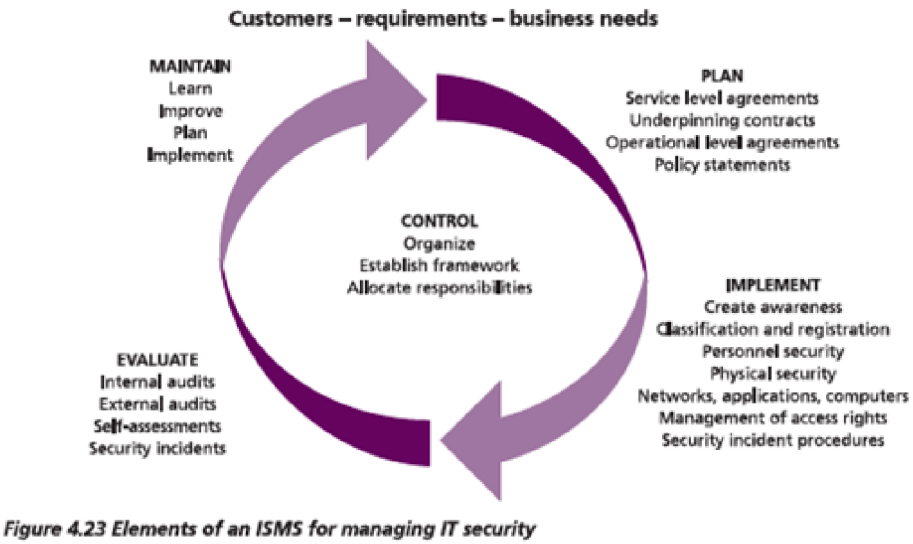
Správa kontinuity služeb je nezbytná pro přežití podniku v krizových situacích, kde řeší a navrhuje scénáře při selhání odolné, nebo vysoce dostupné služby. Cílem je udržet službu v neustálém provozu, potažmo ji dle nastavených priorit obnovit v co nejkratší době.

2.2.7 Information Security Management

Správa bezpečnosti informací má za úkol sladit zabezpečení IT s bezpečností podniku a zajistit, aby důvěrnost, integrita a dostupnost aktiv, informací, dat a IT služeb organizace vždy odpovídala dohodnutým potřebám businessu. Systém pro správu bezpečnosti informací (ISMS) slouží k zavádění a provádění pravidel a cílů, které jsou rozděleni do pěti částí. Tento cyklus je široce používán a je založen na radách a pokynech v mnoho zdrojích včetně ISO / IEC 27001.

- Control
- Plan
- Implement
- Evaluate
- Maintain

Obrázek 2: Prvky ISMS pro správu IT bezpečnosti



Zdroj: (The Cabinet Office 2011b)

2.2.8 Suplier Management

Proces správy dodavatelů zajišťuje dodržování smluv s dodavateli a třetími stranami. V databázi odběratelů by mělo především být hodnocení a kategorizace.

2.3 Service Transition

Třetí kniha ITIL Edice 2011 (The Cabinet Office 2011e) Přechod služby navazuje po návrhu služby a zaměřuje se na přechod služby do provozu, je to tedy fáze životního cyklu, která je zodpovědná za převedení služby z teorie do praxe.

2.3.1 Transition planning and support

Proces plánování a podpory přechodu plánuje přechod služby a koordinuje k tomu potřebné zdroje. K procesu patří i koordinace správy programů a projektů návrhu služby i vývojové činnostní služby.

2.3.2 Change Management

Správa změn řídí životní cyklus všech změn s cílem získat z nich maximální užitek a zároveň minimalizovat nebezpečí výpadků a incidentů poskytované služby způsobené změnami.

2.3.3 Service Asset and Configuration Management

Správa aktiv služeb a konfigurací (SACM) zajišťuje informace o aktivech společnosti, která jsou potřebná pro dodávku služeb. Důležitým aspektem je vytvoření pravidel správy aktiv a konfigurací.

2.3.4 Release and Deployment Management

Správa releasů a provozního nasazení řídí sestavení služby, její testování a následné nasazení do produkčního prostředí, kde tvoří pro zákazníka hodnotu.

2.3.5 Service Validation and testing

Proces validace a testování služby zajišťuje, aby při nasazení nové, nebo upravené služby do produkčního prostředí byla služba funkční dle požadavků a spolehlivá.

2.3.6 Change Evalution

Vyhodnocení změny je proces, který poskytuje konzistentní a standardizované prostředky a možnosti pro zjišťování výkonosti služby. Skutečný výkon se porovnává s očekávanou výkoností a odchylky se zpracovávají.

2.3.7 Knowledge Management

Správa znalostí zajišťuje, aby během životního cyklu služby byly k dispozici spolehlivé a bezpečné informace, které jsou důležité pro správné vyhodnocení situací a kontextů.

2.4 Service Operation

Čtvrtá kniha ITIL Edice 2011 (The Cabinet Office 2011c) se věnuje popisu fáze životního cyklu po předání služeb do provozu. Provoz služeb má zajišťovat, aby zákazník dostával odpovídající hodnotu a je zodpovědný za provádění služeb, které byly v předchozích fázích naplánovány, navrženy a sestaveny.

2.4.1 Event Management

Hlavní úlohy správy událostí jsou identifikace a analýza událostí, z kterých se mají vyvodit adekvátní opatření. Správa událostí je proto základem pro monitorování systémů.

2.4.2 Incident Management

Správa incidentů zaznamenává, zkoumá, kategorizuje, přiřazuje prioritu a sleduje všechny poruchy služeb, aby bylo možné co nejrychleji obnovit normální provoz služby a minimalizovat tak nepříznivý dopad na provoz, čímž se zajistí zachování dohodnuté úrovně kvality služby.

2.4.3 Request Fulfillment

Plnění požadavků je proces odpovědný za správu životního cyklu požadavků na služby od uživatelů, které nesouvisí s poruchou. Například požadavek pro stěhování výpočetní techniky do jiné kanceláře.

2.4.4 Problem Management

Účelem správy problémů je sledování problému od začátku do konce (identifikace, analýza, dokumentace a odstranění). Dále pak minimalizovat nepříznivý dopad incidentů a problémů na podnikání. K dosažení tohoto cíle se snaží dostat k hlavní příčině incidentů, zdokumentovat je a zahájit akce ke zlepšení, nebo nápravě.

2.4.5 Access Management

Správa přístupů je proces, který autorizovaným uživatelům uděluje právo používat službu a zároveň brání přístupu neautorizovaným uživatelům. Dále řeší konflikty rolí, dostupnost informací a jejich ochranu. V některých organizacích se označuje jako správa práv, nebo správa identit.

2.5 Continual Service Improvement (CSI)

Pátá kniha (The Cabinet Office 2011a) obsahuje jeden proces, který popisuje v sedmi krocích neustále, postupně zlepšující se službu viz 3.

2.5.1 The seven-step improvement process

Zlepšovací proces v sedmi krocích je nedílnou součástí CSI a slučuje základní myšlenky Demingova cyklu (PDCA) s přístupem CSI. Účelem procesu je definice a správa kroků, které jsou potřeba pro identifikaci, definici, shromažďování, přípravy, analýzy, prezentaci a následné implementaci. Proces navíc splňuje model: data, informace, znalost, moudrost (DIKW).

1. Identifikace strategie: Identifikace celkové vize, business potřeby, strategické, taktické a provozní cíle.

2. Definování, co se bude měřit: Přímé propojení se strategickými, taktickými a provozními cíly. Definování, co se může skutečně měřit, provádění diferenční analýzy a dokončení plánu měření.

3. Shromáždění dat: Shromažďování dat vyžaduje monitorování, které nejčastěji probíhá pomocí aplikací, nebo systémů, ale může být provedeno i manuálně.

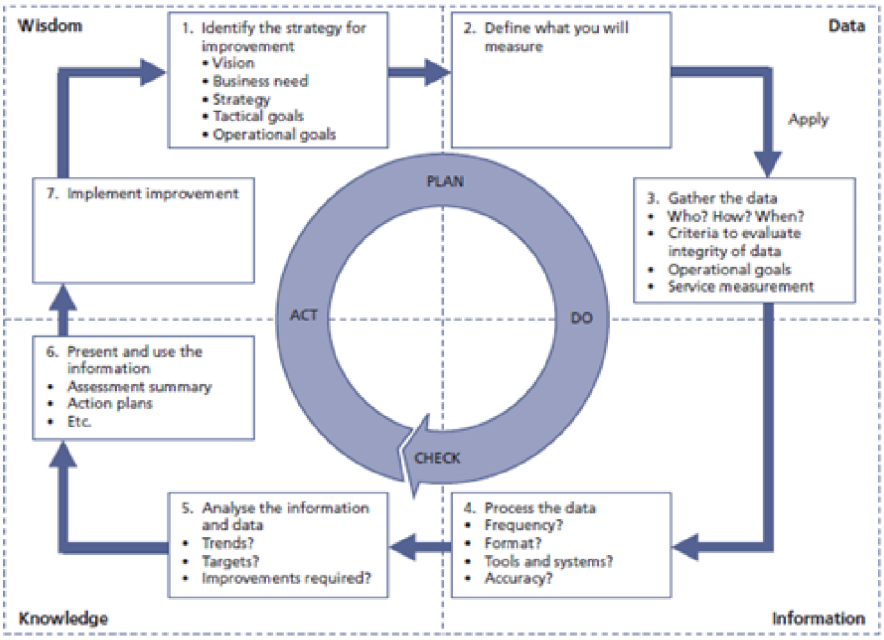
4. Zpracování dat: Zpracovávají se data, která byla shromážděna, aby bylo možné použít ukazatele výkonosti a zkontrolovat tak rozhodující faktory úspěchu.

5. Analýza dat a informací: Zpracovaná data se transformují na informace k identifikaci mezer ve službách, trendů a dopadů na business.

6. Prezentace a využití informací: Prezentování výsledků zainteresovaným stranám (zákazník, IT, dodavatelé).

7. Implementace zlepšení: Dosažené výsledky se využívají k optimalizaci, zlepšení a opravám služeb.

Obrázek 3: Demingův cyklus PDCA



Zdroj: (The Cabinet Office 2011a)

3 Active Directory

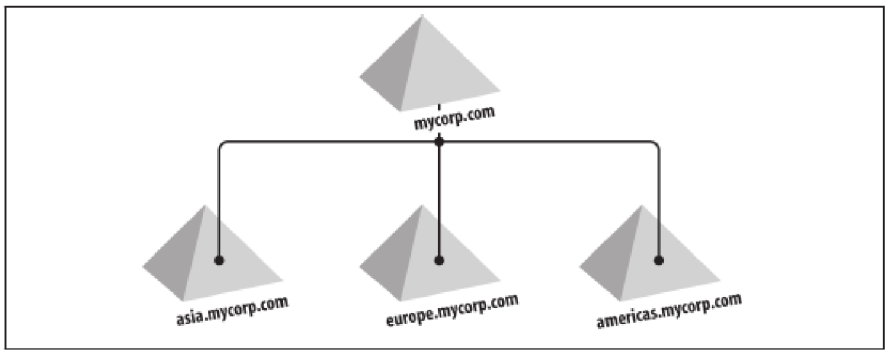
Adresářová služba Active Directory umožňuje správcům efektivně spravovat informace o uživatelích, počítačích, skupinách, tiskárnách, aplikacích a službách z jednoho místa.(Desmond 2013) V této kapitole budou popsány pouze komponenty, které se vztahují k uživatelké identitě a budou následně využity v praktické části bakalřské práce.

3.1 Logické komponenty

Logická struktura v Active Directory obsahuje dva typy objektů. Kontejnerové (containers) a nekontejnerové (non-containers), kterým se také říká listové objekty (leaf object). Mezi kontejnetové patří: les (`forrest`), strom (`tree`), doména (`domain`), organizační jednotka (`organizational unit`). Kontejnerové objekty v hierarchické struktuře mohou obsahovat další objekty, naopak listové objekty žádný další objekt obsahovat nemohou. Do listových objektů patří: uživatel (`user`), počítač (`computer`), tiskárna (`printer`), služba (`service`) (Francis 2019).

- Forrest - Les se může skládat z jedné nebo více domén a doménových stromů, které návzájem sdílí společnou logickou strukturu, schéma a konfiguraci adresářů. Přičemž každá doména má své vlastní charakteristiky, hranice a zdroje.
- Tree - Doménový strom je kolekce domén, hlavní doména se označuje jako `root domain` nebo `parent domain` a je jediná v celém lese, ostatní domény se označují jako `subdomains` nebo `child-domain` viz obrázek 4.
- Domain - Doména se skládá z hierarchické struktury kontejnerů a objektů, `DNS` a `security service`, která ověřuje a autorizuje jakýkoli přístup ke zdrojům pomocí účtů v doméně, popřípadě vztahem důvěryhodnosty (`trust`) s ostatnímy doménami.
- Organizational Unit - Každá doména by měla mít uživatele a počítače zařazeny do organizačních jednotek, které vytvoří organizovaný a strukturovaný adresář. Organizační jednotky mohou rozlišovat například lokalitu, nebo role v organizaci.

Obrázek 4: Doménový strom



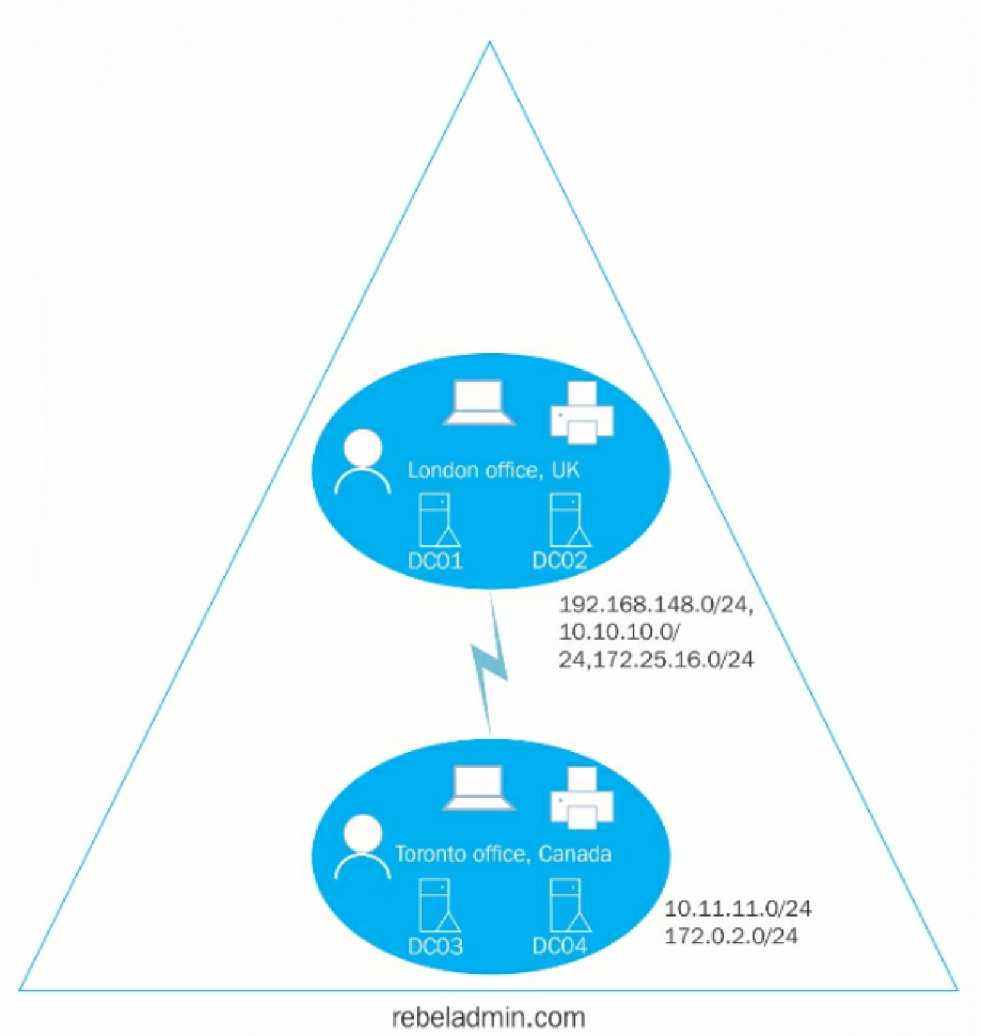
Zdroj: (Desmond 2013)

3.2 Fyzické komponenty

Ve fyzické struktuře jsou: doménový řadič (`domain controller`), globální katalog (`global catalogue`) a `site`. Narozdíl od logické struktury se objekty ve fyzické struktuře nemohou snadno přeskupovat, jelikož by to mohlo určitými způsoby ovlivnit replikaci (Francis 2019).

- Domain Controller - Doménový řadič je počítač, ve kterém je operační systém Windows Server a je na něm nainstalována role `Active Directory Domain Services`. Každá doména může mít více doménových řadičů, což je i doporučováno z důvodu replikace dat.
- Global Catalogue - Globální katalog může být pouze na serveru, který je zároveň doménový řadič a slouží k uchovávání kopií objektů z celého stromu či lesa.
- Site - Site definuje fyzickou topologii sítě, slouží k propojení doménových řadičů, pokud jsou v různých lokalitách viz obrázek 5.

Obrázek 5: Doménové řadiče v různých lokalitách



Zdroj: (Francis 2019)

3.3 Objekty Active Directory

Osoba v reálném světě je velice podobná objektu `user`. Tak jako osoba má své jméno, příjmení, druh povolání, ty samé atributy může mít i objekt `user`. Je pravděpodobné, že může nastat situace, kdy budou mít dvě osoby stejné jméno, příjmení i druh povolání, poté není možné osoby podle těchto atributů rozeznat. Abychom tyto osoby mohly přesně identifikovat, je zapotřebí jednožnačný identifikátor, například: rodné číslo, číslo občanského průkazu. V Active Directory mezi jednoznačné identifikátory patří:

(Francis 2019)

- `globally unique identifier (GUID)` - Globálně jedinečný identifikátor je 128 bitové číslo, které je použito pro každý objekt v Active Directory pod atributem `ObjectGUID`.
- `security identifier (SID)` - Hodnota `SID` je pro objekt jedinečná pouze v rámci domény, pokud dojde k migraci objektu do jiné domény, vytvoří se pro daný objekt nová hodnota `SID`.
- `distinguished name (DN)` - Rozlišující názvev zobrazuje úplnou cestu objektu v adreáři a je složen ze tří atributů:
  - `organizationalUnitName(OU)`
  - `domainComponent(DC)`
  - `commonName(CN)`

  Například uživatel Josef Novák, který je v organizační jednotce `admins` v doméně `fim.cz`, výsledný `DN` by vypadal takto: `CN=Novák Jan,OU=admins,DC=fim,DC=cz`.
- `relative distinguished name (RDN)` - Relativní rozlišující název je jedinečný v rámci svého nadřazeného kontajneru. To znamená, že v doméně může být více uživatelů, kteří mají `CN=Novák Jan`, ale nesmí být ve společném kontajneru, nebo organizační jednotce.

3.4 Group Policy Objects (GPO)

Group Policy sloužící k centrální správě počitáčů a uživatelů v Active Directory, mají dvě hlavní části. První z nich `Computer Configuration`, která se týká nastavení počítače, který je připojený do domény. Politika upravuje registry v `HKEY_LOCAL_MACHINE`, spouští se při startu počítače a nezáleží, jaký uživatel je přihlášený. Naopak `User Configuration` se vztahuje na uživatele, upravuje registry v `HKEY_CURRENT_USER`, spuští se při přihlášení uživatele a nezáleží na jakém počítači se uživatel hlásí (Bouška 2010 [cit. 2022-04-08]).

Ačkoliv Group Policy Objects úzce souvisí s uživatelskou identitou a procesy ITIL ISM 2.2.7 a Access Management 2.4.5, nebude tomuto tématu věnována velká pozornost. V praxi se totiž GPO jednou nasadí a není zapotřebí nad nimi provádět žádné hromadné operace, které by bylo potřeba optimalizovat pomocí PowerShellu. Autor vidí přidanou hodnotu GPO v Powershellu pouze u freelancerů, kteří mohou mít pod správou několik menších firem.

4 Windows PowerShell

V této kapitole budou popsány pouze funkcionality využité v laboratorních cvičení. Detailní popis jazyka lze nalést přímo v dokumentaci na stránkách Microsoftu, nebo v publikacích českého autora Patrika Maliny, který o Powershellu ve své knize píše: *"vezměte to nejlepší z Perlu,Pythonu, jazyka C a Javy, zjednodušte to pro potřeby správců a zabudujte do shellu."* (Malina 2007)

Novější verze operačních systému Windows (od Windows Vista a Windows Server 2008) mají PowerShell defaultně nainstalovaný, proto není potřeba nic instalovat. PowerShell v prostředí Windows 10 je možné nalézt a spustit hned několika způsoby. První možností je přes lupu na hlavním panelu a zadáním výrazu `powershell`, další možností je přes klávesovou zkratku `WIN + R` a zadání `powershell`, případně `powershel_ise`. Aplikace `powershell.exe` je obdobná aplikaci `cmd.exe`, slouží převážně ke spuštění jednodušších scriptů, namísto aplikace `powersel_ise.exe` slouží k psaní, úpravě a editaci složitějších scriptů.

4.1 Execution Policy

Spuštění scriptů se řídí zásadami spuštění (`Execution Policy`), ve výchozím nastavení je zásada nastavena na `Restricted`. Pro zjištění, která zásada je právě nastavena slouží příkaz `Get-ExecutionPolicy`. Zásadu je možné změnit pomocí příkazu `Set-ExecutionPolicy`, například: `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned`. (Payette 2007)

- Restricted - Výchozí nastavení, spouštění scriptů je zakázáno.
- AllSigned - Umožňuje spouštět pouze podepsané scripty od důvěryhodného vydavatele, bez ohledu na to, zda je script stažen z internetu, nebo vytvořen lokálně na počítači.
- RemoteSigned - Scripty stažené z internetu musí být důvěryhodně podápsány, jedná se o výchozí nastavení na Windows Serverech.

- Unrestricted - Lze spouštět všechny scripty, pouze u scriptů stažených z internetu vyskočí varování.
- Bypass - Lze spuštět všechny scripty, bez varování.

4.2 Get-Help

Příkazy, neboli comandlety se v PowerShellu skládají ze slovesa, pomlčky a podstatného jména v jednotném čísle. Například: `Get-Date`. Slovesa by měla být volena ze seznamu povolených sloves, jejich kompletní výpis je možné získat příkazem `Get-Verb`. Aby si administrátor nemusel pamatovat, jak konkrétní příkaz funguje, existuje v PowerShellu příkaz `Get-Help`, který zobrazí dokumentaci k danému příkazu. Například `Get-Help Get-ADUser -Full`.

4.3 Proměnné

Základní proměnné se zapisují pomocí znaku `$` a přiřazením hodnoty, PowerShell automaticky pozná o jaký datový typ se jedná a přiřadí ho k dané proměnné. Pokud bychom chtěli zamezit pozdějšímu přetypování, je lepší uvést datový typ už při deklaraci proměnné `[int] $cislo = 10`.

4.4 Roura

Roura (`pipeline`) je série příkazů oddělená svislou čarou `|`. Každý příkaz v rouře příjme objekt včetně jeho atributů a metod z předchozího příkazu, na rozdíl od předchozích shellů (bash, comand promnt), které rourou předávají pouze textový řezězec. (Payette 2007) Například příkaz `Get-ADUser -Filter* | format table` vezme objekt každého uživatele, pošle ho rourou a následně zformátuje do tabulky.

4.5 Podmínky a cykly

Obdobně jako většina proceduálních programovacích jazyků, i ve Windows PowerShell je možné najít řadu podmínek a cyklů například: `if, switch, while, foreach`. Na rozdíl od ostatních programovacích jazyků PowerShell používá jiný zápis operátorů, níže jsou vypsány ty nejpoužívanější.

- `-eq` - rovná se
- `-ne` - nerovná se
- `-gt` - větší než
- `-ge` - větší nebo rovno
- `-lt` - menší než
- `-le` - menší nebo rovno
- `-and` - a zároveň
- `-or` - nebo

if

Nejzákladnější podmínkou je podmínka `if`, která porovnává, zda je výraz v závorce pravda, či nepravda. Pokud je výraz pravdivý, provede se akce uvnitř složených závorek, jinak se provede akce uvnitř závorek za `else`. Podmínka může být ještě rozšířena o výraz `elseif`, který může být opakován vícekrát. Například 1, kde se porovnává, zda je proměnná x větší než 100, pokud bude menší, porovná se, zda je větší než 50, pokud i to nebude pravda, vypíše se: Číslo není příliš velké.

```
if ($x -gt 100) {
    "It's greater than one hundred"
} elseif ($x -gt 50) {
    "It's greater than 50"
} else {
    "It's not very big."
```

```
}
```

Ukázka kódu 1: Podmínka if; **Zdroj:** (Payette 2007)

switch

Než používat ve scriptu několikrát `elseif`, je vhodnější využít podmínku `switch`. Podmínka `switch` porovnává, zda proměnná uvitř závorky odpovídá daným podmínkám. Pokud odpovídá nějaké podmínce, provede se akce v bloku podmínky. Pokud neodpovídá žádné podmínce, provede se blok `deafult`. Jelikož `switch` vyhodnocuje všechny podmínky, i v případě, kdy byla nějaká podmínka splněna, je zapotřebí za každou podmínku uvádět `break`.

```
$temperature = 20
switch($temperature) {
    { $_ -lt 32 } { "Below Freezing"; break }
    32 { "Exactly Freezing"; break }
    { $_ -le 50 } { "Cold"; break }
    { $_ -le 70 } { "Warm"; break }
    default { "Hot" }
}
```

Ukázka kódu 2: Podmínka switch; **Zdroj:** (Holmes 2013)

while

Cyklus `while` bude stále dokola provádět akci uvnitř bloku, dokud nebude výraz uvnitř závorky platit.

```
$val = 0
while($val -ne 3) {
    $val++
    write-host "The number is $val"
}
```

Ukázka kódu 3: Cyklus while; **Zdroj:** (Payette 2007)

foraech

Cyklus `foreach` slouží k prohledání pole, nebo kolekce. V úkázce 4 cyklus `foreach` projde postupně celé pole a vždy vypíše aktuální hodnotu.

```
$numbers = 1,2,3,4,5
foreach($number in $numbers) {
    "Value is:" + $number
}
```

Ukázka kódu 4: Cyklus foreach; **Zdroj:** (Autor)

4.6 Vyjímky

Při spuštění scriptu se může stát, že script skončí na nějaké chybě a neprovede se až do konce. Patrik Malina ve své knize, Jak vyzrát na Microsoft Windows PowerShell 2.0, rozděluje chyby do dvou situací. V první situaci se jedná o syntaktickou chybu, kterou je snadné opravit, neboť jí PowerShell hned označí podtržením. Naopak druhá situace je o něco komplikovanější, jelikož chyby nastávají pouze za určitých okolností a nejsou vždy snadno předvídatelné. V těchto situacích je žádoucí script ošetřit pomocí *Try-Cath-Finally* 5, který je známý z ostatních programovacích jazyků. V bloku *Try* se vykoná ta část scriptu, ve kterém se mohou vyskytovat chyby, jedná se o takzvaný pokus. Blok *Catch* je určen k odchycení chyby a případné reakci na chybový stav. Blok *Finally* provede nasledné operace, ať už byla chyba odchycena, či nikoliv (Malina 2010).

```
try {
    1 / 0
}
catch [DivideByZeroException] {
    "Don't divide by zero: $_"
}
finally {
"Script that will be executed even if errors occur in the try
   statement"
```

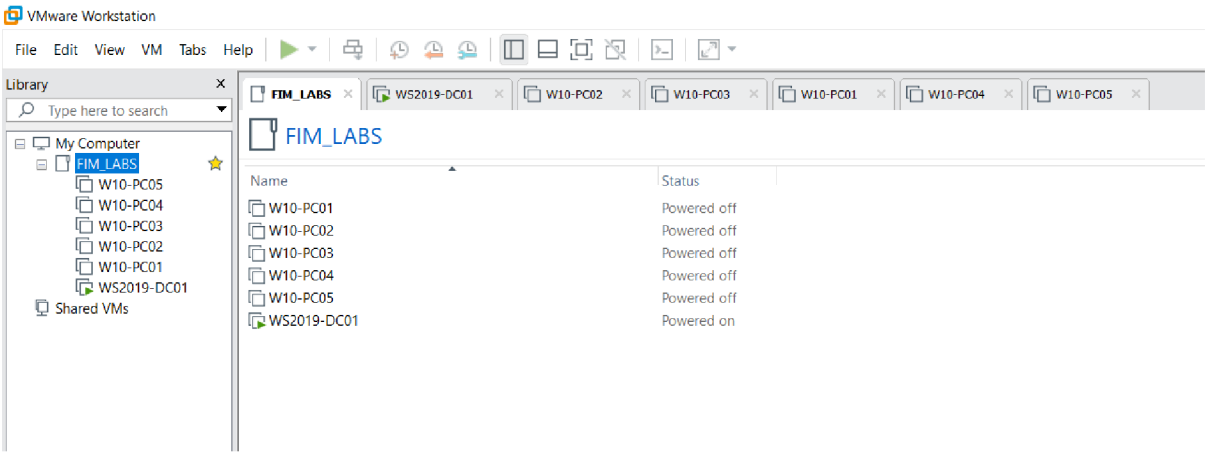
```
}
```

Ukázka kódu 5: Try-Catch-Finally; **Zdroj:** (Holmes 2013)

5 Praktická část práce

Pro laboratorní úlohy je nejprve potřeba vytvořit virtuální prostředí, které bude vytvořeno pomocí licencovaného softwaru VMware® Workstation 15 Pro. Software pro virtualizaci je možné nahradit Virtual Boxem od firmy Oracle, který je zdarma pro domácí i komerční použití. Ve virtuálním prostředí bude nainstalován Windows Server 2019 Standard Evalution a několik Windows 10 Enterprise viz obrázek 6. Scripty budou psány v jazyce Powershell ve verzi 5.1, pro zjištění verze Powershellu je možné použít příkaz `$PSVersionTable`.

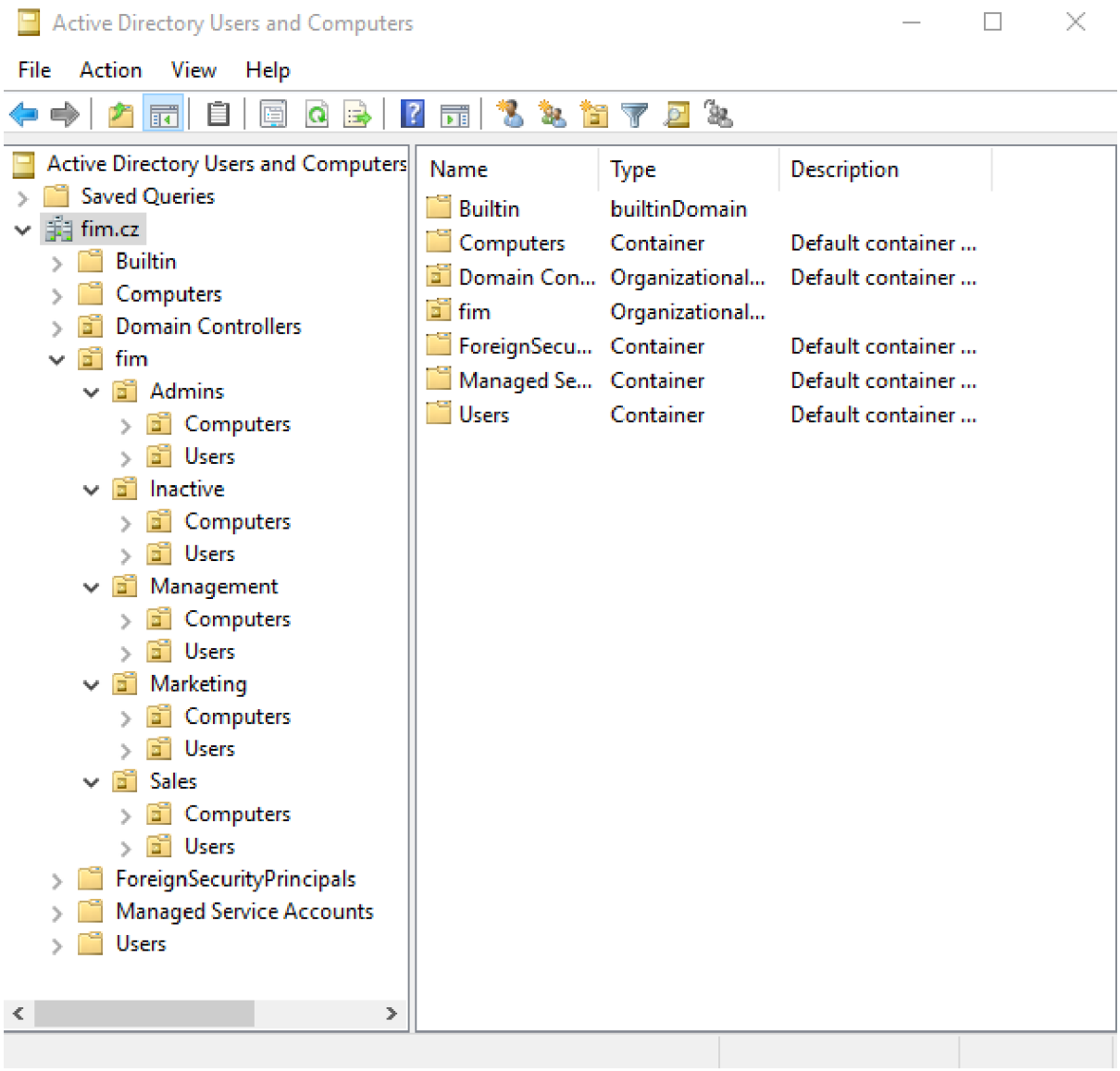
Obrázek 6: Virtuální prostředí ve VMware® Workstation



Zdroj: (Autor)

Na Windows Serveru bude nainstalována služba `Active Directory Domain Services` s doménou `fim.cz`. Server bude povýšen na `Domain Controller` a `Global Catolog`. Po instalaci bude vytvořena doménová struktura za pomocí organizačních jednotek, které budou rozděleny dle oddělení viz obrázek 7. Dříve bylo zapotřebí ještě instalovat Powershell a importovat modul pro Active Directory, u novějších serverů již toto není potřeba.

Obrázek 7: Struktura domény FIM



Zdroj: (Autor)

5.1 LAB 1 - Uživatelská identita v AD

Uživatelská identita v Active Directory začíná nástupem zaměstnance do firmy, kdy je potřeba zaměstnanci vytvořit účet v AD pro přístup k doméně. Každý zaměstnanec ve firmě nastupuje na určitou pozici, proto je důležité tyto pozice oddělit v doménové struktuře pomocí organizačních jednotek a skupin. Nástup zaměstanců do firmy je rutinní záležitost, která může nastávat každý měsíc. Proto je vhodné, aby vše probíhalo automaticky a administrátor byl pouze informován.

Návrh úlohy

1. Vytvoření uživatele v doméně.
2. Přesunutí uživatele do patřičné organizační jednotky dle oddělení.
3. Přidání uživatele do oprávněné skupiny dle oddělení.
4. Vytvoření a export logu.
5. Odeslání logu emailem na IT oddělení.
6. Vytvoření úlohy pro opakování každý první den v měsíci.

Vytváření, přesouvání do OU a přidávání uživatelů do skupin zabere spousty času, zejména když se jedná například o desítky zaměstnanců. V praxi to funguje tak, že je zaměstnanec při nástupu do firmy vytvořen v HR systému, ze kterého se importují data a celý proces proběhne za pomocí PowerShellu automaticky. Pro laboratorní úlohu není potřeba žádný HR systém, data o zaměstnancích budou naimportována z CSV souboru. Parametrů pro vytvoření uživatele existuje zdaleka více, než je na obrázku 8, úplný seznam je možný zjistit příkazem `Get-Help New-ADUser -full`.

Obrázek 8: LAB 1 - Data o zaměstnancích

FirstName	Lastname	Department
Kayley	Decker	Sales
Pharrell	Adam	Marketing
Naomi	Connolly	Management
Britney	Alison	Admins
Valentino	Murphy	Sales
Naomi	Connolly	Admins

Zdroj: (Autor)

V první řade je zapotřebí nadefinovat proměnné. Do proměnných `$import` a `$export` se uloží cesta v adresáři pro import, respektive export souboru. Název souboru bude dynamický, závislý na proměnné `$today` z důvodu importování a exportování souboru podle datumu. Proměnné pod komentářem email slouží k odeslání logu emailem na IT oddělení, ke kterému dojde na konci scpritu. Proměnné `$login` a `$password` jsou z důvodu bezpečnosti uloženy v textových souborech, heslo je navíc úplně skryto pomocí příkazu `ConvertTo-SecureString -AsPlainText`. Přihlašovací údaje (`$credentials`) jsou potom vytvořeny pomocí nové instance `PSCredential($login, $password)`. Následně se do proměnných uloží organizační jednotky a skupiny podle oddělení. Nakonec se do proměnné `$users` uloží údaje o zaměstnancích pomocí příkazu `Import-CSV` s parametrem `-Delimiter ";"`, pomocí kterého se slova v CSV souboru rozdělí podle středníku.

```
#Set variables
$domain = (Get-ADDomain).DNSRoot
$date = (Get-Date)
$today = $date.ToString("yyyy-MM-dd")
$import = "C:\import\new-users\new-users_" + $today + ".csv"
$export = "C:\export\new-users\new-users_" + $today + ".txt"

#Email
$From = "security@fim.cz"
$To = "tomas1valenta@gmail.com"
$Attachment = $export
$Subject = "New users in AD"
```

```
$Body = "<h2>Monthly report</h2>"
$SMTPServer = "smtp-relay.sendinblue.com"
$SMTPPort = "587"
$login = Get-Content C:\Import\smtp-login.txt
$password = Get-Content C:\Import\smtp-password.txt |
   ConvertTo-SecureString -AsPlainText -Force
$credentials = New-Object System.Management.Automation.
   PSCredential($login, $password)

#OU
$OUAdmins = "OU=Users,OU=Admins,OU=fim,DC=fim,DC=cz"
$OUManagement = "OU=Users,OU=Management,OU=fim,DC=fim,DC=cz"
$OUMarketing = "OU=Users,OU=Marketing,OU=fim,DC=fim,DC=cz"
$OUSales = "OU=Users,OU=Sales,OU=fim,DC=fim,DC=cz"

#Group
$GAdmins = "GRP_Admins"
$GManagement = "GRP_Management"
$GMarketing = "GRP_Marketing"
$GSales = "GRP_Sales"

#Import CSV file
$users = Import-Csv $import -Delimiter ";"
```

Ukázka kódu 6: Import CSV a deklarace proměných; **Zdroj:** (Autor)

V ukázce 7 je zobrazen cyklus `foreach`, který končí až na konci celého scriptu. Celé jádro cyklu proběhne pro každého uživatele zvlášť, nejprve se vytvoří proměnné, do kterých se uloží data z CSV souboru, poté pomocí oddělení (`department`) a příkazu `switch` se nastaví příslušná organizační jednotka a skupina. Proměnná `$ADUsers` uvnitř cyklu `foreach` představuje performační hrozbu, jelikož pro každého uživatele v CSV souboru bude znovu procházet celou databázi uživatelů. Script ale bude spuštěn pouze jednou měsíčně v noci, navíc pokud by v daném CSV souboru byly dva zaměstnanci se stejným jménem a příjmením, je zapotřebí mít stále nejnovější data, proto je možné toto performační riziko příjmout.

```
#LOOP
foreach ($user in $users) {

    $ADUsers = Get-ADUser -Filter *
    $firstName = $user.firstname
    $lastName = $user.lastname
    $userName = "$($firstName[0])$lastName".ToLower()
    $initials = "$($firstName[0])$($lastName[0])"
    $upn = "$userName@$domain"
    $department = $user.department
    $displayName = "$firstname $lastname"

    #Condition for OU and Group
    switch ($department) {
        "Admins"         {$OU = $OUAdmins
                          $GRP = $GAdmins}
        "Management"     {$OU = $OUManagement
                          $GRP = $GManagement}
        "Marketing"      {$OU = $OUMarketing
                          $GRP = $GMarketing}
        "Sales"          {$OU = $OUSales
                          $GRP = $GSales}
    }
```

Ukázka kódu 7: Foreach cyklus; **Zdroj:** (Autor)

Zejména ve větších společnostech se může stát, že existují dva zaměstnanci se stejným jménem, z toho důvodu je zapotřebí tuto skutečnost ověřit a řešit. Cyklus `while` nejprve ověří, zda `$username` ze souboru CSV již existuje v doméně. Pokud ano, jádro cyklu proběhne a přiřadí za `$username` hodnotu z indexu `$i`. Před cyklem je zapotřebí si nadefinovat pomocné proměnné, pokud bychom tak neučinili, tak by po prvním proběhnutí byl například `username1`, ale při dalším `username11`, namísto `username2`. Může nastat situace, kdy nastoupí najednou dva zaměstnanci se stejným jménem a příjmením na stejné oddělení, ačkoliv je tato situace málo pravděpodobná, je vhodné tuto situaci vyřešit. Jelikož musí být `distinguished name` v Active Directory unikátní, musí se danému uživateli přičíst k příjmení číslo dle seznamu.

```
#If username exist, change username
$defaultName = $userName
$defaultLastName = $lastName
$distinguishedName = "CN=" + $firstName + " " + $lastName + "
," + $OU
$i = 1


while ($ADUsers.SamAccountName -eq $userName) {

    $userName = $defaultName + [string]$i
    $upn = "$userName@$domain"

    if ($ADUsers.DistinguishedName -eq $distinguishedName) {

        $lastName = $defaultLastName + [string]$i
    }
        $i++
}
```

Ukázka kódu 8: Podmínka pro tvorbu loginu; **Zdroj:** (Autor)

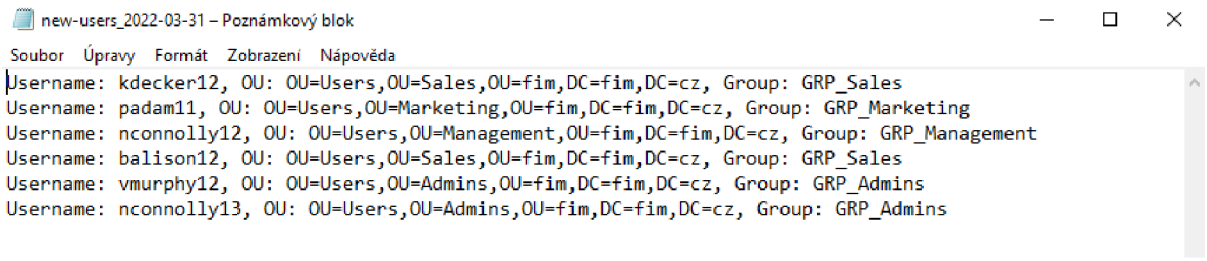
Příkazem `New-ADUser` a zadáním parametrů se vytvoří uživatel v doméně. Za zmínku stojí parametr `-AccountPasswordAtLogon`, který nemá z důvodu bezpečnosti hodnotu přes proměnnou. Heslo je napsáno odděleně v souboru `default-password.txt`, přes příkaz `Get-Content` vloženo a pomocí příkazu `ConvertTo-secureString` zašifrováno. Na závěr je uživatel přidán do patřičné skupiny příkazem `Add-ADGroupMember`, navíc je pomocí příkazů `Write-Output` a `Add-Content` přidán záznam do logu. Výsledný log, který je následně odeslán emailem na IT oddělení pomocí příkazu `Send-MailMessage` viz 10, je možný vidět na obrázku 9.

```
#Create new user
New-ADUser `
-SamAccountName $userName `
-UserPrincipalName $upn `
-Name "$firstName $lastName" `
-GivenName $firstName `
-Surname $lastName `
-Initials $initials `
-Enabled $True `
-DisplayName $displayName `
-Path $OU `
-EmailAddress $upn `
-Department $department `
-AccountPassword (Get-Content C:\Import\default-password.txt
| `
ConvertTo-secureString -AsPlainText -Force) `
-ChangePasswordAtLogon $True

#Add user to group
Add-ADGroupMember $GRP $userName

Write-Output "Username: $userName, OU: $OU, Group: $GRP" |
Add-Content $export
```

```
}
```

Ukázka kódu 9: Vytvoření uživatele; **Zdroj:** (Autor)

Obrázek 9: LAB 1 - Výsledný log



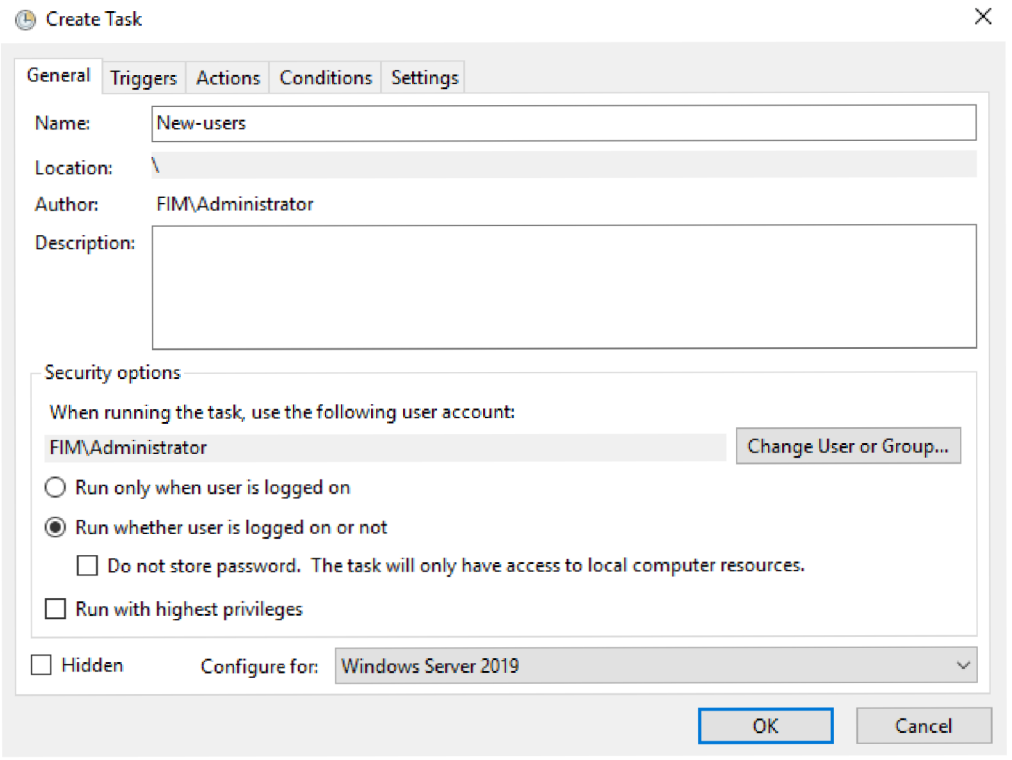
Zdroj: (Autor)

```
#Send Email
Send-MailMessage -From $From -To $To -Subject $Subject -Body
$Body -SmtpServer $SMTPServer -Usessl -Port $SMTPPort -
Credential $credentials -Attachments $export -Priority Normal
```

Ukázka kódu 10: Odeslání emailu]; **Zdroj:** (Autor)

Aby celý script nemusel administátor spouštět každý měsíc, je vhodné pro tento script vytvořit plánovač úloh, který bude daný script spouštět automaticky a opakovaně, podle pravidel, které se nastaví. Plánovač úloh je možný spustit klávesovou zkratkou `WIN + R` a zadání `taskschd.msc`. Při kliknutí na `Create task...` se otevře okno viz. obrázek 10. V sekci general se nastaví Name: New-users, přepne se radio button na Run whether user is logged on or not a nastaví se Configuration for: Windows Server 2019.

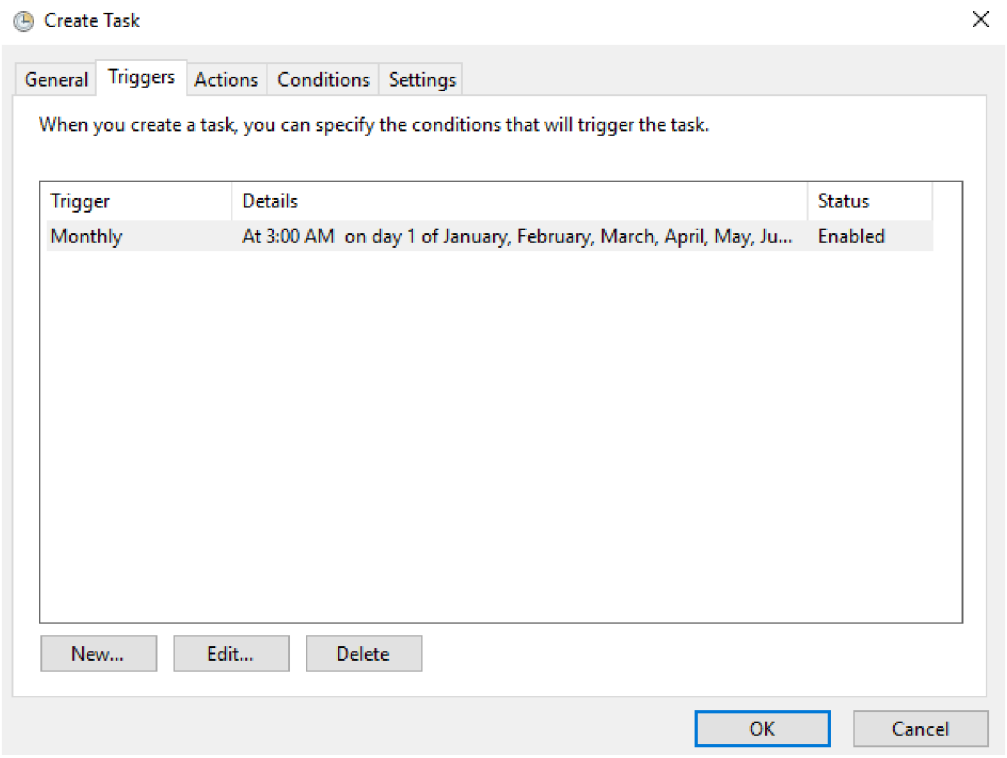
Obrázek 10: LAB 1 - Task - general



Zdroj: (Autor)

V sekci Triggers na obrázku 11 se nastaví spouštěč, v této úloze je to každého prvního v měsíci v 03:00 ráno.

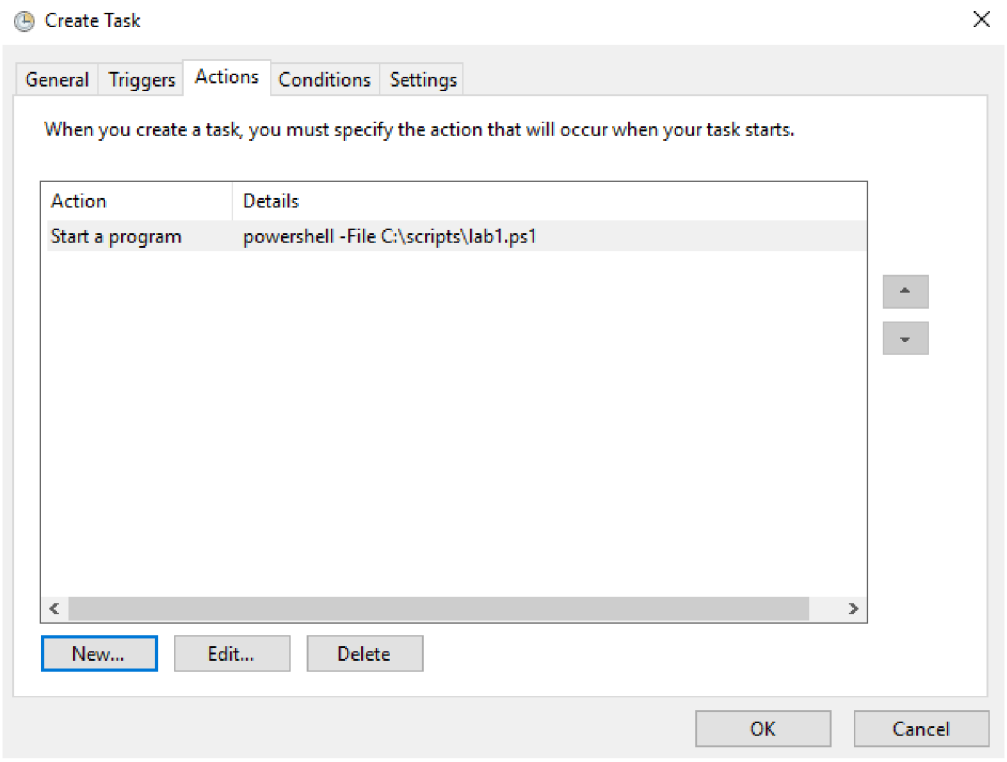
Obrázek 11: LAB 1 - Task - triggers



Zdroj: (Autor)

Nakonec se v sekci Action na obrázku 12 nastaví, jaký script se bude spouštět. Po kliknutí na tlačítko OK bude ještě zapotřebí zadat administrátorské heslo a plánovač úloh bude nastaven.

Obrázek 12: LAB 1 - Task - actions



Zdroj: (Autor)

5.2 LAB 2 - Zdroj zamčení uživatele

Z důvodu bezpečnosti by měla být v každé doméně nastavena politika hesel, která se nastavuje přes GPO3.4. V politice se nastavuje složitost hesla, změna hesla po určitém časovém intervalu a v případě několika chybných pokusů zadání hesla zamčení uživatele. Uživatel poté musí počkat určitou dobu, která je nastavena také politikou, než ho AD znovu odemkne. V případě, že se uživatel potřebuje do domény přihlásit hned, musí kontaktovat administrátora, který ho může odemknout okamžitě. Zamčení uživatele se většinou nestane při špatně zadaném hesle od uživatele, ale z důvodu uložených hesel v aplikacích, které jsou propojené s AD. Poté, při změně hesla, může při procesu „na pozadí“ dojít k opakovanému zadaní chybného hesla a tím se uživatelský účet uzamkne. Je proto vhodné napsat script, který prozkoumá Event log a zobrazí nám zdroj zamčení daného uživatele.

Návrh úlohy

1. Vytvoření formuláře.
2. Přidání komponent do formuláře.
3. Vytvoření ovládání tláčítek.

V první řadě je zapotřebí načíst dvě .NET třídy `System.Windows.Forms` a `System.Drawing` pomocí příkazu `Add-Type` s patametrem `-AssemblyName`. Následně se do proměnné `$form` uloží nová instance `Form`, které se poté definuje velikost, text a barva pozadí, viz 11.

```
# Init PowerShell Gui
Add-Type -AssemblyName System.Windows.Forms
Add-Type -AssemblyName System.Drawing

# Create a new form
$Form                    = New-Object system.Windows.Forms.Form
$Form.ClientSize         = '500,400'
$Form.text               = "Zdroj zamceni uzivatele"
```

```
$Form.BackColor            = "#ffffff"
```

Ukázka kódu 11: Vytvoření formuláře; Zdroj (Autor)

Do formuláře je zapotřebí přidat ostatní prvky. Nejpre ve vytvoří hlávní popisek a popisek pro textbox vytvořením instance `Label`, popiskům se nastaví text, výška, šířka a pozice pomocí instance `Point`. Textbox se vytvoří pomocí instance `TextBox`, kde se nastaví, aby nebyl víceřádkový `multiline = $false`, metoda `focus()` znamená, že při otevření formuláře bude umísten kurzor přímo v Textboxu, takže Administrátor může hned začít psát login uživatele. Dále se vytvoří tři tlačítka pomocí instance `Button`. Tlačítka Search a Unlock mají nastavenou událost pomocí `add_Click`, která bude řešena později, navíc tlačíko Unlock bude při spuštění formuláře skryto (`$btnUnlock.Visible = $false`). Tlačítko Cancel má logiku nastavenou rovnou. Pomocí `DialogResult`, který volá hodnotu `Cancel`. Poslední prvek je `ListView`, který je rozdělen do tří sloupců: Uzivatel, Zdroj, Datum. Na závěr se všechny prvky přiřadí do formuláře.

```
#Label
$lbl = New-Object System.Windows.Forms.Label
$lbl.Text = "Zadej login uzivatele"
$lbl.AutoSize = $true
$lbl.Width = 25
$lbl.Height = 10
$lbl.Location = New-Object System.Drawing.Point(20,15)

#Label login
$lblLogin = New-Object system.Windows.Forms.Label
$lblLogin.text = "Login:"
$lblLogin.AutoSize = $true
$lblLogin.width = 25
$lblLogin.height = 20
$lblLogin.location = New-Object System.Drawing.Point(20,40)

#Textbox
$txtLogin = New-Object system.Windows.Forms.TextBox
$txtLogin.multiline = $false
```

```
$txtLogin.width = 300
$txtLogin.height = 20
$txtLogin.location = New-Object System.Drawing.Point(80,40)
$txtLogin.Focus()

# Search button
$btnSearch = New-Object System.Windows.Forms.Button
$btnSearch.BackColor = "#00AFF0"
$btnSearch.Text = "Search"
$btnSearch.Width = 90
$btnSearch.Height = 30
$btnSearch.Location = New-Object System.Drawing.Point(390,40)
$btnSearch.add_Click($btnSearch_Click)

#Unlock button
$btnUnlock = New-Object System.Windows.Forms.Button
$btnUnlock.BackColor = "#00AFF0"
$btnUnlock.Text = "Unlock"
$btnUnlock.Width = 90
$btnUnlock.Height = 30
$btnUnlock.Location = New-Object System.Drawing.Point(100,300)
$btnUnlock.Visible = $false
$btnUnlock.add_Click($btnUnlock_Click)

#Listview
$list = New-Object System.Windows.Forms.ListView
$list.View = 'Details'
$list.Width = 460
$list.Height = 200
$list.Location = New-Object System.Drawing.Point(20,80)
$list.AutoResizeColumns(2)
$list.Columns.Count = 3
$list.Columns.Add("Uzivatel")
$list.Columns.Add("Zdroj")
```

```
$list.Columns.Add("Datum")

#Cancel button
$btnCancel = New-Object System.Windows.Forms.Button
$btnCancel.BackColor = "#ffffff"
$btnCancel.Text = "Cancel"
$btnCancel.Width = 90
$btnCancel.Height = 30
$btnCancel.Location = New-Object System.Drawing.Point(260,300)
$btnCancel.DialogResult = [System.Windows.Forms.DialogResult]::
   Cancel
$Form.CancelButton = $btnCancel

$Form.Controls.AddRange(@($lbl,$lblLogin,$txtLogin,$btnSearch,$
   btnUnlock,$list,$btnCancel))
```

Ukázka kódu 12: Přidání komponent do formuláře; Zdroj (Autor)

Aby bylo možné formulář ovládat je zapotřebí napsat logiku pro tlačítka. Při kliknutí na tlačítko Search se nejprve ověří, zda je uživatel, který byl zadán na vstupu, v doméně. Pokud byl uživatel nalezen, provede se blok `try`, který nejprve definuje proměnné pro doménový řadič, event ID a datum 30 minut zpětně. Do proměnné `$event` se uloží událost, která má následující kritéria: událost se stala na doménovém řadiči (`-ComputerName $dc`), `LogName = "Security"`, `ProviderName = "Microsoft-Windows-Security-Auditing"`, událost se stala maximálně 30 minut dozadu (`StartTime = $date`) - více minut není potřeba protože je nastavena politika hesel, která uživatele po 30ti minutách automaticky odemkne, ID události je 4740, které znamená, že uživatel byl uzamknut a nakonec upřesnění, že jde o zaměstnance zadaného na vstupu (`data = $txtLogin.Text`). Hodnoty z proměnné `$event` se zobrazí na výstupu vytvořením nové intance `ListViewItem`. Do itemů se nejprve uloží hodnota z `Properties` s indexem 0, kde je uložený uživatel, dále hodnota z `Properties` s indexem 0, kde je uložen zdroj zamčení a následně datum a čas zamčení z `TimeCreated`, který se ještě musí převést na textový řetězec pomocí metody `ToString()`, jelikož `ListViewItem` nepodporuje `Date` formát. Na závěr se zob-

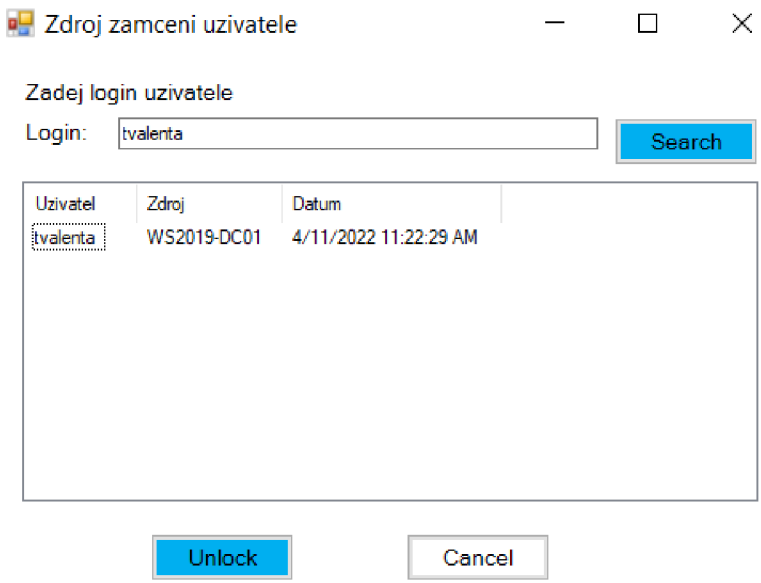
razí tlačítko pro odemčení `$btnUnlock.Visible = $true`. Vyhledaného, zamčeného uživatele je možné vidět na obrázku 13.

```
$btnSearch_Click = {

    $existingUser = Get-ADUser -Filter {SamAccountName -eq $
    txtLogin.Text}

    if ($existingUser) {

        try {
            $dc = (Get-ADDomain).PDCEmulator
            $eventID = 4740
            $date = (Get-Date).AddMinutes(-30)

            $event = Get-WinEvent -ComputerName $dc -
    FilterHashtable @{
                LogName = "Security"
                ProviderName = "
    Microsoft-Windows-Security-Auditing"
                StartTime = $date
                Id = $eventID
                data = $txtLogin.Text
            }

            $item = New-Object System.Windows.Forms.ListViewItem(
    $event.Properties[0].Value)
            $item.SubItems.Add($event.Properties[1].Value)
            $item.SubItems.Add($event.TimeCreated.ToString())
            $list.Items.Add($item)
            $btnUnlock.Visible = $true

        } catch {

            [System.Windows.Forms.MessageBox]::Show("Akce se
```

```
    nezdarila", "Error")

        }

     } else {

        [System.Windows.Forms.MessageBox]::Show("Uzivatel
    nenalezen", "Warning")

     }
}
```

Ukázka kódu 13: LAB 2 - Tlačítko search

Obrázek 13: LAB 2 - Nalezený záznam



Zdroj: (Autor)

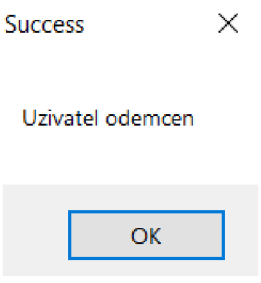
Tlačítko Unlock, které se zobrazí až pokud je nalezen záznam o zamčení daného uživatele, provede v bloku `try` odemčení uživatele, následně administrátora notifikuje dialogovým oknem o zdařilé akci viz. obrázek 14. Dojde k vymazání hodnot z textboxu pomocí `$txtLogin.Clear()` a listview `$list.Items.Clear($item)`. Následně se kurzor

zobrazí v texboxu pomocí `$txtLogin.Focus()`. Pokud by v bloku `try` došlo k chybě, provede se blok `catch`, který zobrazí dialogové okno o nezdařilé akci viz. obrázek 15.

```
$btnUnlock_Click = {

    try {

        Unlock-ADAccount -Identity $txtLogin.Text
        [System.Windows.Forms.MessageBox]::Show("Uzivatel odemcen
   ", "Success")
        $txtLogin.Clear()
        $list.Items.Clear($item)
        $list.Refresh()
        $txtLogin.Focus()

    } catch {

        [System.Windows.Forms.MessageBox]::Show("Akce se
   nezdarila", "Error")
    }
}
```

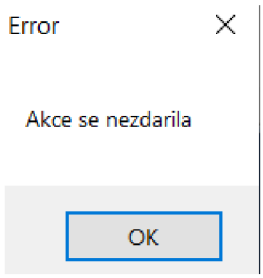
Ukázka kódu 14: LAB 2 - Tlačítko unlock

Obrázek 14: LAB 2 - Odemčení uživatele



Zdroj: (Autor)

Obrázek 15: LAB 2 - Akce se nezdařila



Zdroj: (Autor)

Na závěr se formulář zobrazí zavoláním metody *Show.Dialog()* viz. 15.

```
# Display the form
[void]$Form.ShowDialog()
```

Ukázka kódu 15: LAB 2 - Spuštění aplikace

5.3 LAB 3 - Zablokování neaktivního uživatele / PC

V rámci bezpečnosti celé firmy jsou nastaveny politiky GPO3.4, může se ale stát, že některý počítač, aniž by byl používaný, zůstane připojený k doméně. To samé se může stát u uživatelů, například při nástupu zaměstnaňkyně na mateřskou dovolenou. Její identita ve firmě zůstává, zůstává tedy i její identita v AD, ale není schopna si měnit heslo dle politiky v GPO. Je proto vhodné napsat script, který prohledá všechny neaktivní uživatele a PC. Jejich účty budou následně zablokovány a přesunuty do organizační jednotky k tomu určené.

Návrh úlohy

1. Vyhledání neaktivních uživatelů a počítačů.
2. Zablokování neaktivních uživatelů a počítačů.
3. Přesunutí neaktivních uživatelů a počítačů do OU Inactive.
4. Vytvoření a export logu.
5. Odeslání logu emailem na IT oddělení.
6. Vytvoření úlohy pro opakování každou neděli.

Nejpre je zapotřebí definovat proměnné. Do proměnné `$lastWeek` se pomocí příkazu `Get-Date` a metody `AddDays(-7)` uloží datum přesně o sedm dní zpětně. V proměnné `$export` je uložena cesta k exportovanému logu. Následuje definice proměnných pro odesílání emailu obdobně jako v 5.1. Jelikož se uživatelé a počítače budou přesouvat do jiné organizační jednotky, je zapotřebí pro ně vytvořit přoměnné. `$OUInactiveUsers` pro uživatele a `$OUInactiveComputers` pro počítače. Na závěr ukázky 16 je nutné vyhledat neaktivní uživatele a počítače. To se provede pomocí příkazu `Get-ADUser` pro uživatele, respektivě `Get-ADComputer` pro počítače. Pomocí parametru `-Filter` se určí, aby se vyhledali pouze ti uživatelé / počítače, které mají poslední přihlášení delší než jeden týden a zároveň nemají zablokovaný účet. To je z toho důvodu, aby se nevyhledaly i účty, které jsou již zablokovány a přesunuty do organizační jednotky Inactive.

```
#Set variables
$lastWeek = (Get-Date).AddDays(-7)
$today = (Get-Date).ToString("yyyy-MM-dd")
$export = "C:\export\inactive-accounts\inactive-accounts_" + $
   today + ".txt"

#Email
$From = "security@fim.cz"
$To = "tomas1valenta@gmail.com"
$Attachment = $export
$Subject = "Inactive users and computers"
$Body = "<h2>Weekly report</h2>"
$SMTPServer = "smtp-relay.sendinblue.com"
$SMTPPort = "587"
$login = Get-Content C:\Import\smtp-login.txt
$password = Get-Content C:\Import\smtp-password.txt |
   ConvertTo-SecureString -AsPlainText -Force
$credentials = New-Object System.Management.Automation.
   PSCredential($login, $password)

#OU
$OUInactiveUsers = "OU=Users,OU=Inactive,OU=fim,DC=fim,DC=cz"
$OUInactiveComputers = "OU=Computers,OU=Inactive,OU=fim,DC=fim,DC
   =cz"

#Inactive accounts
$inactiveUsers = Get-ADUser -Filter {LastLogon -le $lastWeek -and
    Enabled -eq $true }
$inactiveComputers = Get-ADComputer -Filter {LastLogon -le $
   lastWeek -and Enabled -eq $true }
```

Ukázka kódu 16: LAB 3 - Deklarace proměných

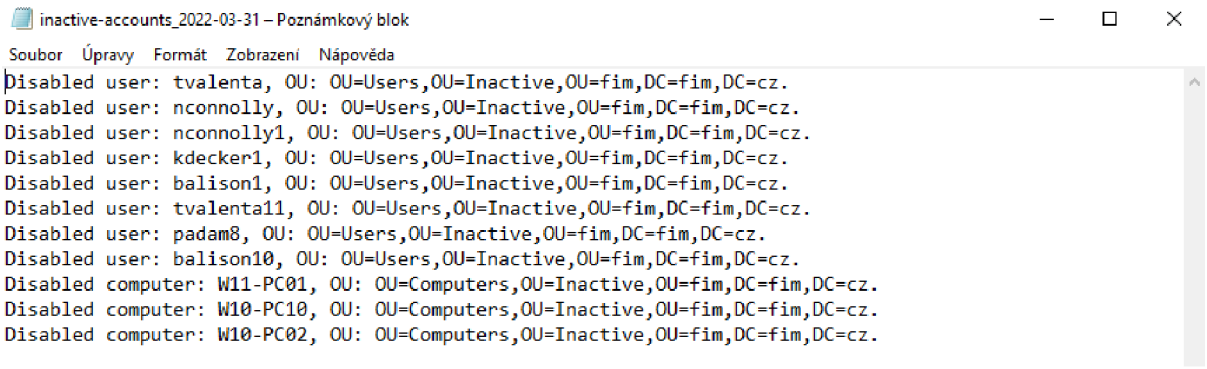
V následující ukázce 17 jsou dva `foreach` cykly. První cyklus pro každého neaktivního uživatele provede zablokování pomocí příkazu `Disable-ADAccount` a příkazem

`Move-ADObject` přesune uživatele do organizační jednotky pro neaktivní uživatele. Následně o tom vypíše informaci a přidá ji do logu. Druhý cyklus provede ty samé operace akorát pro všechny neaktivní počítače.

```
foreach ($user in $inactiveUsers) {

    Disable-ADAccount -Identity $user
    Move-ADObject -Identity $user -TargetPath $OUInactiveUsers
    $user = $user.SamAccountName

    Write-Output "Disabled user: $user, OU: $OUInactiveUsers." |
   Add-Content $export

}

foreach ($computer in $inactiveComputers) {

    Disable-ADAccount -Identity $computer
    Move-ADObject -Identity $computer -TargetPath $
   OUInactiveComputers
    $computer = $computer.Name

    Write-Output "Disabled computer: $computer, OU:
   $OUInactiveComputers." | Add-Content $export
```

Ukázka kódu 17: LAB 3 - Cyklus

Obrázek 16: LAB 3 - Výsledný log



Zdroj: (Autor)

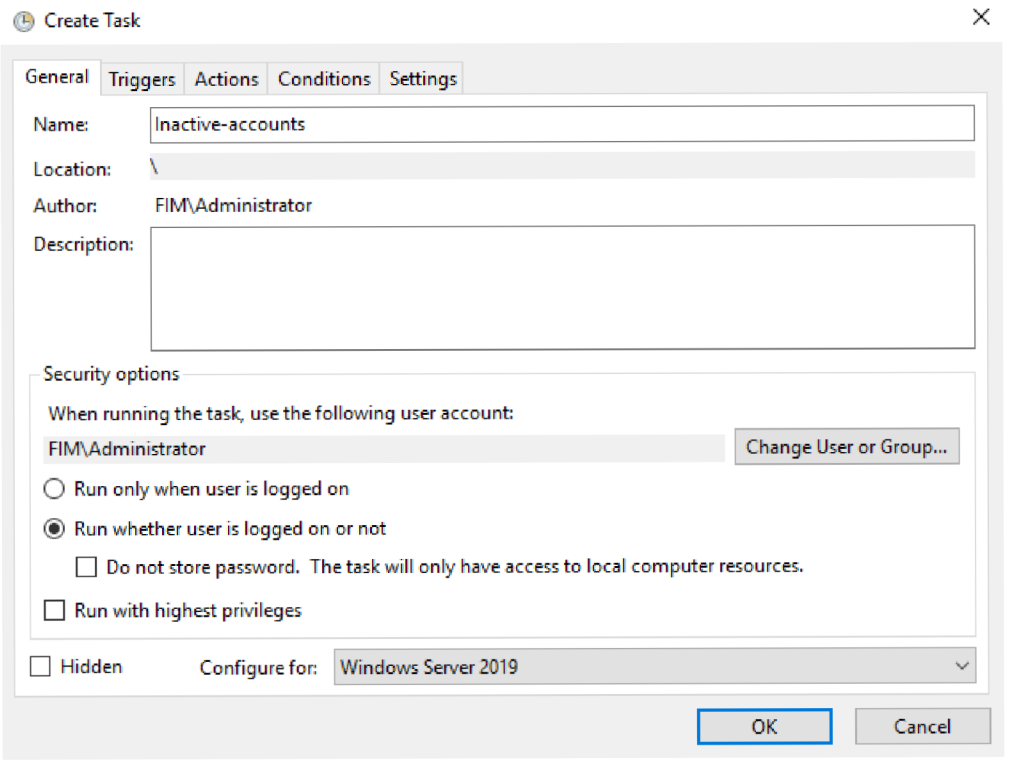
Na závěr scriptu se pomocí příkazu `Send-MailMessage` provede odeslání emailu na IT oddělení s vysokou prioritou. Vysoká priorita je z toho důvodu, aby administrátoři hned v pondělí ráno otevřeli log, který je možný vidět na obrázku 16 a začali řešit, z jakého důvodu se konkrétní účty staly neaktivní.

```
#Send Email
Send-MailMessage -From $From -To $To -Subject $Subject -Body $
    Body -SmtpServer $SMTPServer -Usessl -Port $SMTPPort -
    Credential $credentials -Attachments $export -Priority High
```

Ukázka kódu 18: LAB 3 - Odeslání emailu

Jelikož script hledá účty, které se do domény nepřipojily více jak týden, je také zapotřebí, aby se tento script spouštěl automaticky, v tomto případě na týdenní bázi. To lze provést opět přes plánovač úloh. V sekci general se nastaví Name: Inactive-users, přepne se radio button na Run whether user is logged on or not a nastaví se Configuration for: Windows Server 2019.

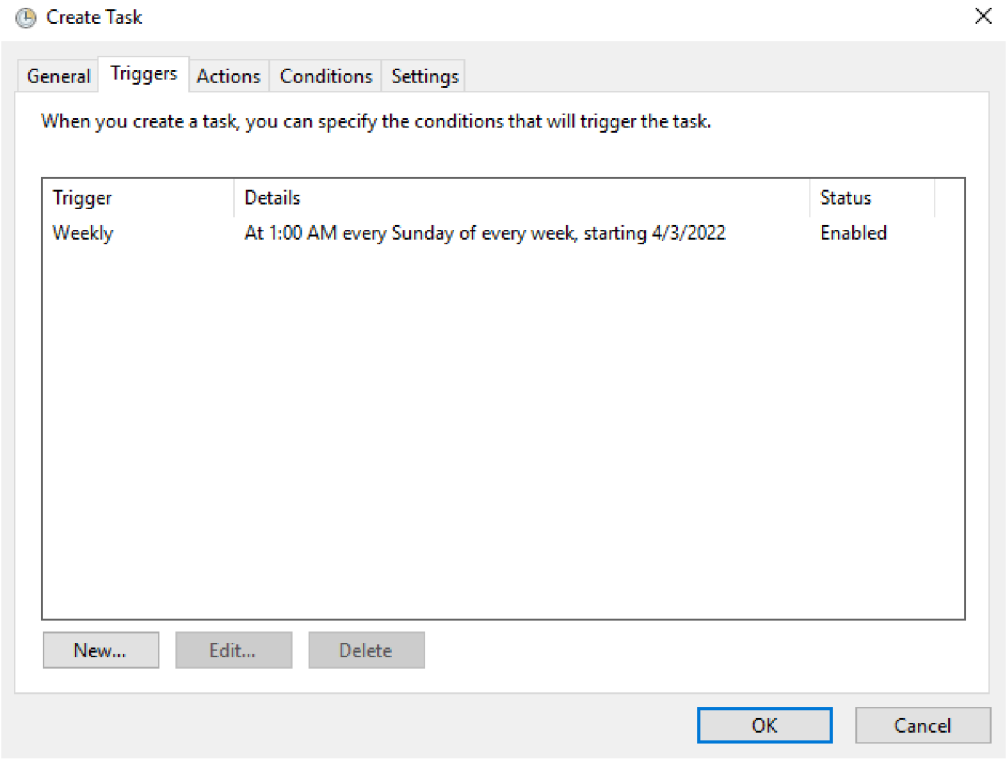
Obrázek 17: LAB 3 - Task - general



Zdroj: (Autor)

V sekci Triggers na obrázku 18 se nastaví spouštěč, v této úloze je to každou neděli v 01:00 ráno.

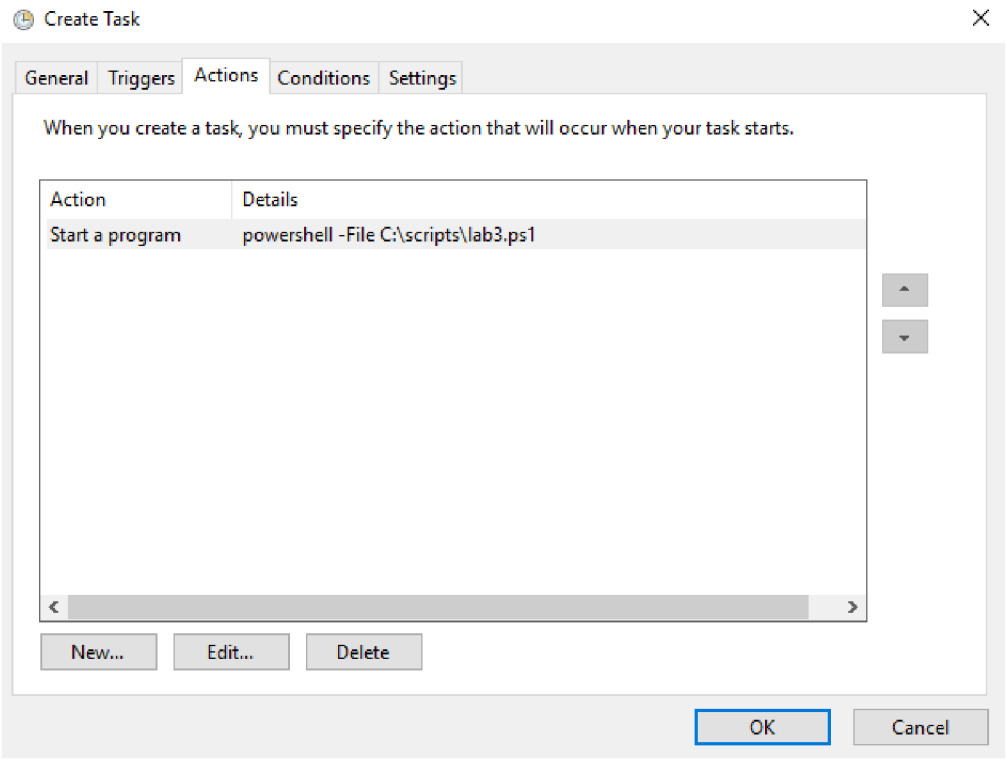
Obrázek 18: LAB 3 - Task - triggers



Zdroj: (Autor)

Nakonec se v sekci Action na obrázku 12 nastaví, jaký script se bude spouštět. Po kliknutí na tlačítko OK bude ještě zapotřebí zadat administrátorské heslo a plánovač úloh bude nastaven.

Obrázek 19: LAB 3 - Task - actions



Zdroj: (Autor)

5.4 LAB 4 - Změna pozice zaměstnance ve firmě

V korporátních společnostech se často stává, že je fluktuace zaměstnanců velmi vysoká. Navíc může docházet u zaměstnanců ke kariérnímu růstu a to i mimo jejich oddělení. Další případ může nastat, když se žena vdá a převezme příjmení po manželovi. V rámci ITIL 2.2.7 a 2.4.5 je proto zapotřebí tyto zkutečnosti každý měsíc hlídat.

Návrh úlohy

1. Načtení změn ze souboru.
2. Provedení změn.
3. Vytvoření a export logu.
4. Odeslání logu emailem na IT oddělení.
5. Vytvoření úlohy pro opakování každý první den v měsíci.

Obrázek 20: LAB 4 - CSV pro import

Username	Firstname	Lastname	Department	EndDate
jnovak	Josef	Novak	Admins	
tvalenta	Tomáš	Valenta	Management	
nconnolly	Naomi	Osaka	Marketing	
balison1	Britney	Alison	Sales	03/31/2022

Zdroj: (Autor)

Začátek scriptu je obdobný, jako v laboratorní úloze 5.1. Do proměnných `$import` a `$export` se uloží cesta v adresáři pro import, respektive export souboru, proměnné pod komentářem email slouží k odeslání logu emailem. Následně se do proměnných uloží organizační jednotky a skupiny a do proměnné `$users` se uloží údaje o zaměstnancích pomocí příkazu `Import-CSV` viz 19.

```
#Set variables
$date = (Get-Date)
$today = $date.ToString("yyyy-MM-dd")
$import = "C:\import\sync-users\sync-users_" + $today + ".csv"
$export = "C:\export\sync-users\sync-users_" + $today + ".txt"

#Email
$From = "sync-users@fim.cz"
$To = "it.department@fim.cz"
$Attachment = $export
$Subject = "Sync users in AD"
$Body = "<h2>Monthly report</h2>"
$SMTPServer = "smtp-relay.sendinblue.com"
$SMTPPort = "587"
$login = Get-Content C:\Import\smtp-login.txt
$password = Get-Content C:\Import\smtp-password.txt |
   ConvertTo-SecureString -AsPlainText -Force
$credentials = New-Object System.Management.Automation.
   PSCredential($login, $password)

#OU
$OUAdmins = "OU=Users,OU=Admins,OU=fim,DC=fim,DC=cz"
$OUManagement = "OU=Users,OU=Management,OU=fim,DC=fim,DC=cz"
$OUMarketing = "OU=Users,OU=Marketing,OU=fim,DC=fim,DC=cz"
$OUSales = "OU=Users,OU=Sales,OU=fim,DC=fim,DC=cz"

#Group
$GAdmins = "GRP_Admins"
$GManagement = "GRP_Management"
$GMarketing = "GRP_Marketing"
$GSales = "GRP_Sales"

#Import CSV file
```

```
$users = Import-Csv $import -Delimiter ";"
```

Ukázka kódu 19: Deklarace proměných a import CSV; Zdroj: (Autor)

Na začátku cyklu v ukázce dojde k deklaraci proměnných z CSV souboru, hodnota do proměnných `$OU` a `$GRP` se uloží podle oddělení pomocí příkazu `switch`. Do proměnné `$existingUser` se uloží celý objekt uživatele, pokud se `$userName` shoduje s nějakým `SamAccountName` v AD.

```
#LOOP
foreach ($user in $users) {
    $firstName = $user.firstname
    $lastName = $user.lastname
    $userName = $user.username
    $upn = "$userName@$domain"
    $department = $user.department
    $expirationDate = $user.enddate
    $existingUser = Get-ADUser -Filter {SamAccountName -eq $
   userName}

    #Condition for OU and Group
    switch ($department) {
        "Admins"      {$OU = $OUAdmins
                       $GRP = $GAdmins}
        "Management" {$OU = $OUManagement
                       $GRP = $GManagement}
        "Marketing"  {$OU = $OUMarketing
                       $GRP = $GMarketing}
        "Sales"       {$OU = $OUSales
                       $GRP = $GSales}
    }
```

Ukázka kódu 20: Začátek cyklu; Zdroj: (Autor)

Pokud uživatel exituje, přiřadí se do proměnné `$SAN SamAccountName` a definuje se stávající oddělení (`$currentDepartment`) a stávající příjmení (`$currentSurname`).

```
 if ($existingUser) {

     $SAN = $existingUser.SamAccountName
     $currentDepartment = (Get-ADUser -Identity $SAN -
Properties Department).Department
     $currentSurname = $existingUser.Surname
```

Ukázka kódu 21: Podmínka: existující uživatel; Zdroj: (Autor)

Pokud není stávající oddělení shodné s oddělením importovaným z CSV souboru, provede se blok podmínky. Stávající skupina (`$currentGroup`) se definuje až uvnitř této podmínky, jelikož se váže na oddělení a bylo by zbytečné jí definovat o úroveň výš. V bloku `try` dojde k přesunutí uživatele pomocí příkazu `Move-ADObject`, příkazem `Remove-ADGroupMember` dojde k odebrání uživatele ze stávající skupiny, nesmí se zapomenout nastavit parametr `-Confirm` na hodnotu `False`, aby script nevyžadoval interakci od uživatele, jelikož má script běžet automaticky. Příkazem `Add-ADGroupMember` se uživatel přidá do nové skupiny a pomocí příkazu `Set-AdUser` se uživateli přepíše oddělení. Nakonec se přes příkaz `Write-Output` vypíší změny, které se pomocí roury `|` a příkazu `Add-Content` přidají do logu. Pokud v bloku `try` nastane nějaká chyba, provede se blok `catch`, který pouze vypíše chybu a přidá jí do logu.

```
     if($currentDepartment -ne $department) {

         $currentGroup = "GRP_" + $currentDepartment

         try {

             Move-ADObject -Identity $existingUser -TargetPath
 $OU
             Remove-ADGroupMember -Identity $currentGroup -
Members $existingUser -Confirm:$False
             Add-ADGroupMember $GRP $existingUser
             Set-ADUser $SAN -Department $department
             Write-Output "Username: $SAN, OU: $OU, Group:
$GRP, Department: $department." | Add-Content $export
```

```
            } catch {

                Write-Output "Username: $SAN -> error in
department." | Add-Content $export
            }
        }
```

Ukázka kódu 22: Podmínka: změna oddělení; Zdroj: (Autor)

Pokud není stávající příjmení shodné s příjmením z CSV souboru, vytvoří se proměnná `$initials` do které se uloží počáteční písmena z jména a příjmení a promměnná `$displayName`, do které se uloží jméno a příjmení. V bloku `try` se příkazem `Set-ADUser` přepíší hodnoty příjmení `-Surname`, iniciály `-Initials` a zobrazované jméno `-DisplayName`. Aby došlo i ke změně `distinguishedname`, musí se použít příkaz `Rename-ADObject`, ve kterém se do parametru `-NewName` uloží hodnota z `$displayName`. Na závěr se vypíší změny a přidají se do logu. Pokud v bloku `try` nastane chyba, například může nastat situace, kdy najednou budou dvě stejná jména a příjmení v jedné oragizační jednotce, tím pádem nebude `distinguishedname` unikátní. V takovém případě se provede blok `catch`, který pouze vypíše chybu a přidá jí do logu. Administrátor poté bude muset najít příčinu chyby a opravit jí ručně, pokud by se jednalo o chybu uvedenou v příkladu, musel by například k příjmení připsat číslo, aby byl `distinguishedname` unikátní. Uživateli se změní příjmení a iniciály, ale uživatelské jméno zůstane ponecháno, je to z toho důvodu, že už může mít pro dané `UPN` vytvořenou emailovou schránku a další přístupy do ostatních aplikací.

```
        if($currentSurname -ne $lastName) {

            $initials = "$($firstName[0])$($lastName[0])"
            $displayName = "$firstname $lastname"

            try {

                Set-ADUser $SAN -Surname $lastName -Initials $
initials -DisplayName $displayName
                Rename-ADObject -Identity $existingUser -NewName
```

```
$displayName
            Write-Output "Username: $SAN, Display Name:
$displayName, Initials: $initials" | Add-Content $export

        } catch {

            Write-Output "Username: $SAN -> error in surname.
" | Add-Content $export
        }
    }
```

Ukázka kódu 23: Podmínka: změna příjmení; Zdroj: (Autor)

Pokud má uživatel v CSV soboru zapsaný datum ukončení pracovního poměru, dojde k nastavení expirace účtu pomocí příkazu `Set-ADUser` a parametru `-AccountExpirationDate` k datu uvedeném v CSV souboru. Administrátor poté nemusí v den odchodu zaměstnance jeho účet ručně zablokovat. Následně se vypíše informace o nastaveném datumu a přidá se do logu.

```
    if(-not [string]::isNullOrEmpty($expirationDate)) {

        Set-ADUser $SAN -AccountExpirationDate $
expirationDate
        Write-Output "Username: $SAN, ExpirationDate:
$expirationDate." | Add-Content $export
    }
```

Ukázka kódu 24: Podmínka: ukončení zaměstnaneckého poměru; Zdroj: (Autor)

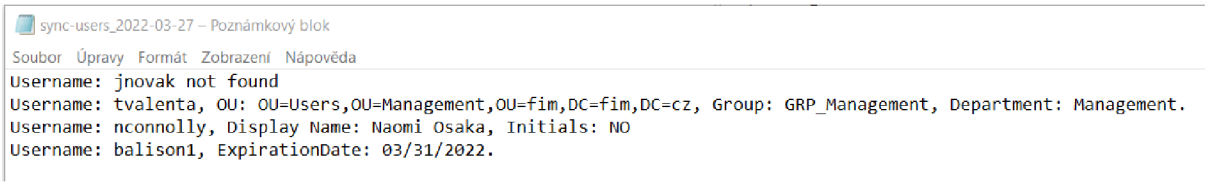
Pokud uživatel v CSV souboru není shodný z žádným uživatelem v doméně, vypíše se tato informace do logu.

```
 } else {

    Write-Output "Username: $userName not found" |
Add-Content $export
 }
```

Ukázka kódu 25: Pokud uživatel neexistuje; Zdroj: (Autor)

Na obrázku 21 je výsledný log. Uživatel jnovak nebyl nalezen, uživatel tvalenta byl přeřazen na oddělení managementu, uživatelce nconnolly se změnilo příjmení na Osaka a uživatel baliston1 ukončí pracovní poměr ve firmě k 31.3.2022.

Obrázek 21: LAB 4 - Výsledný log



Zdroj: (Autor)

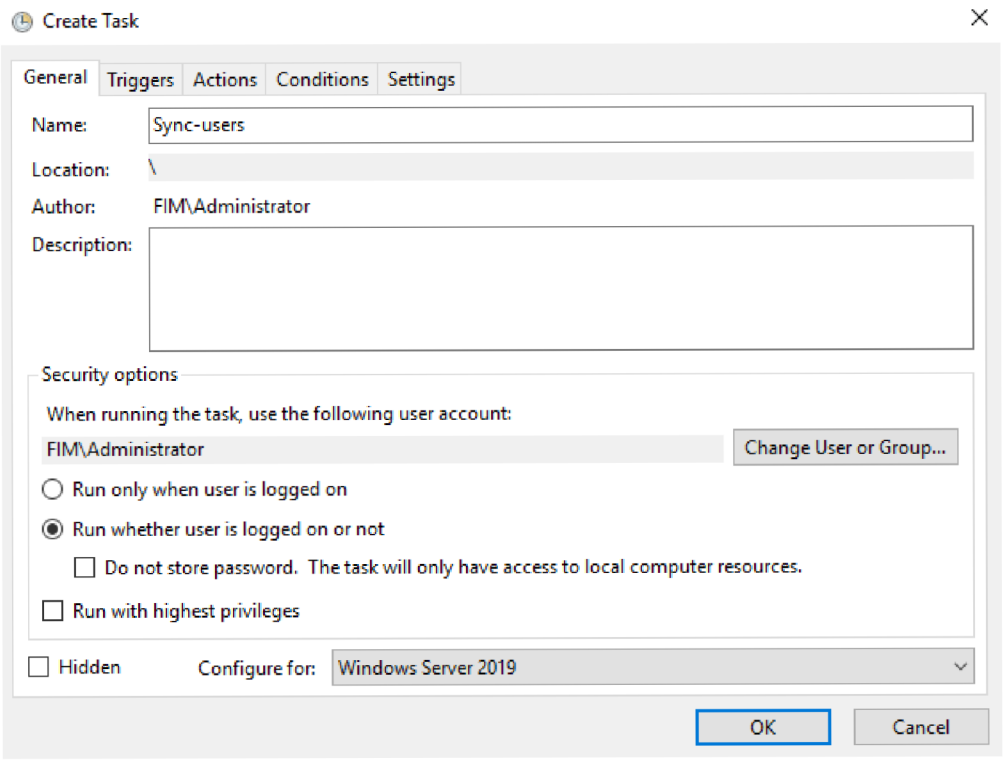
Na závěr scriptu se vysledný blok odešle emailem na IT oddělení příkazem `Send-MailMessage`.

```
#Send Email
Send-MailMessage -From $From -To $To -Subject $Subject -Body $
    Body -SmtpServer $SMTPServer -Usessl -Port $SMTPPort -
    Credential $credentials -Attachments $export -Priority Normal
```

Ukázka kódu 26: Odeslání emailu; Zdroj: (Autor)

Obdobně jako v LABech 5.1 a 5.3, je zapotřebí tento script spouštět automaticky přes plánovač úloh. V sekci general se nastaví Name: Sync-users, přepne se radio button na Run whether user is logged on or not a nastaví se Configuration for: Windows Server 2019.

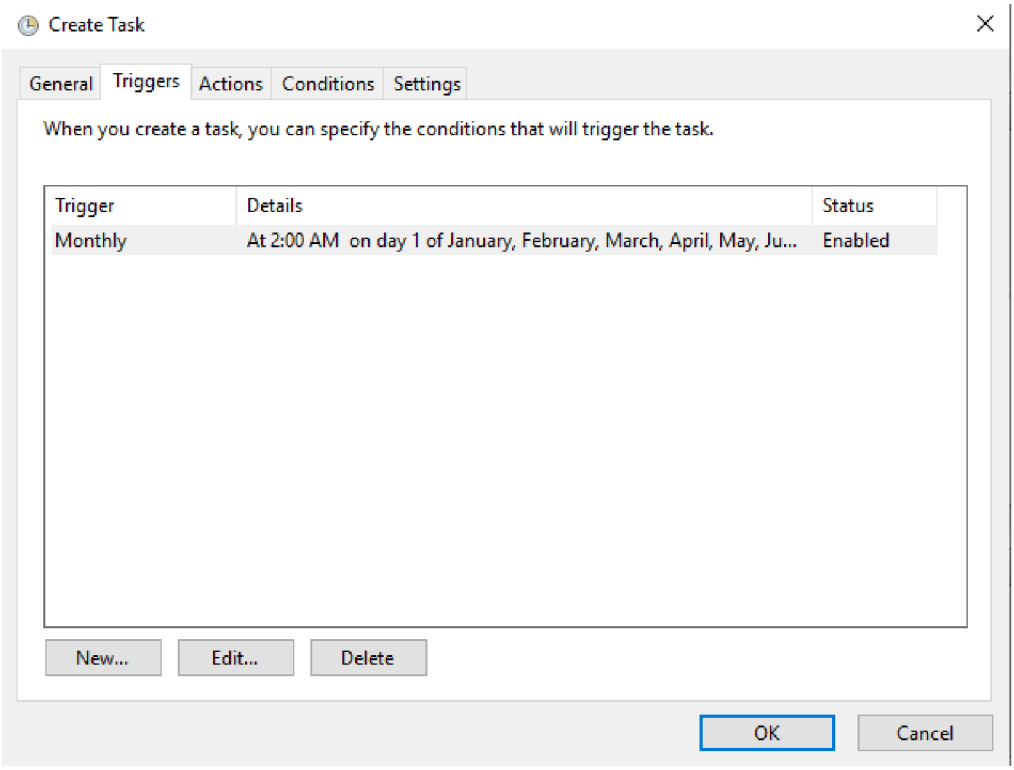
Obrázek 22: LAB 4 - Task - general



Zdroj: (Autor)

V sekci Triggers na obrázku 23 se nastaví spouštěč, v této úloze je to každou neděli ve 02:00 ráno.

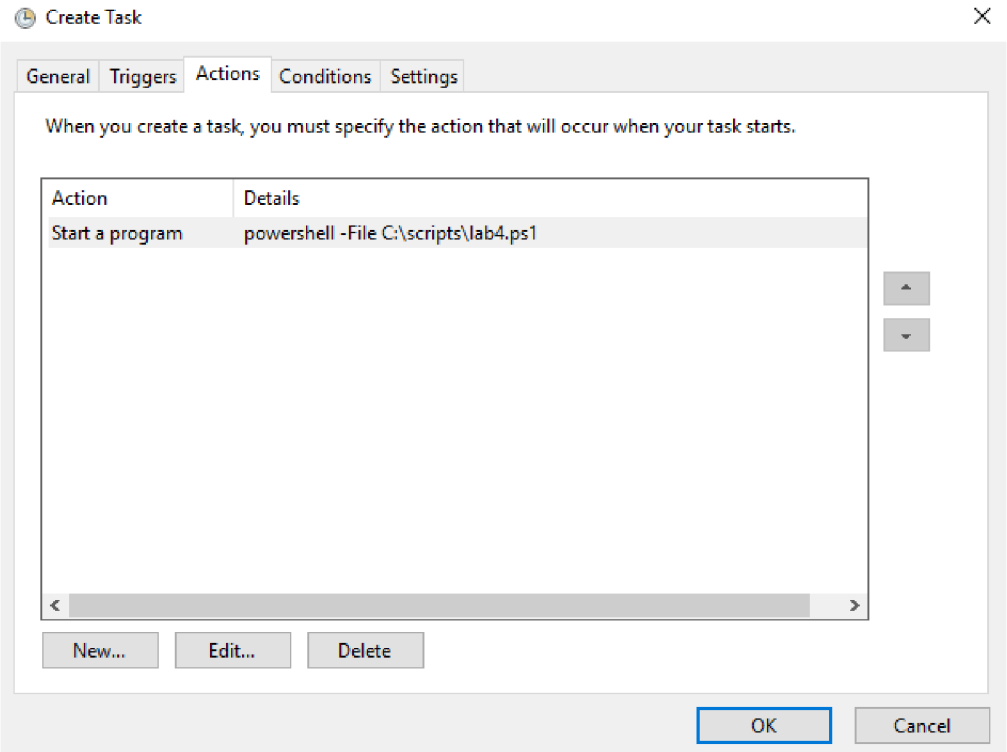
Obrázek 23: LAB 4 - Task - triggers



Zdroj: (Autor)

Nakonec se v sekci Action na obrázku 12 nastaví, jaký script se bude spouštět. Po kliknutí na tlačítko OK bude ještě zapotřebí zadat administrátorské heslo a plánovač úloh bude nastaven.

Obrázek 24: LAB 4 - Task - actions



Zdroj: (Autor)

6 Závěr

Cílem bakalářské práce bylo navrhnout, implementovat a otestovat řadu skriptů pro optimalizaci a automatizaci životního cyklu uživatelských identit.

V teoretické části autor popsal všechny procesy ITIL 2011, službu Active Directory a scriptovací jazyk Windows PowerShell.

V prvním laboratorním cvičení 5.1 bylo řešeno hromadné přidávání uživatelů do Active Directory. Nejdříve se načetly informace o uživatelích ze souboru, poté proběhlo roztřídění uživatelů podle oddělení a tvorba loginu. Na závěr byl uživatel vytvořen a zařazen do organizační jednotky a skupiny podle oddělení. Následně byl vytvořen plánovač úloh, který bude script spuštět každý měsíc.

Ve druhém cvičení 5.2 byl řešen problém, kdy se uživateli opakovaně zablokuje účet v Active Directory. Pro Administrátora je zapotřebí, aby znal přesný zdroj zamčení. Proto byla vytvořena GUI aplikace, do které stačí napsat login zamčeného uživatele. Aplikace poté vypíše zdroj a čas zamčení. Administrátor má poté možnost uživatele i odblokovat.

Třetí cvičení 5.3 kladlo důraz na bezpečnost, kdy bylo zapotřebí neaktivní účty zablokovat a přesunout do organizační jednotky `Inactive`. Script prohledal všechny uživatele i počítače, kteří se do domény nepřihlásili více, jak týden. Jejich účty následně zablokoval a přesunul do patřičné organizační jednotky. Byl vytvořen i plánovač úloh, který script budu spouštět na týdenií bázi.

V posledním laboratorním cvičení 5.4 byla řešena reakce na změny ve firmě. Změny mohou nastat z různých důvodů, například: změna pozice, změna příjemní, ukončení pracovního poměru. To vše je vhodné řešit automaticky, proto autor napsal script, který tyto situace řeší, následně vytvořil plánovač úloh, který bude tento script spuštět každý měsíc.

Bakalářská práce bude využita jako doplňující podklad pro výuku předmětu Operační systémy. Na práci je možné dále navázat například optimalizací a automatizací rutinních operací v Microsoft Exchange Serveru.

Seznam ukázek kódu

Seznam obrázků

Bibliografie

Antoš, Filip (2018 [cit. 2022-04-08]). *Efektivní návrh služby Active Directory pro potřeby středně velké firmy [online]*. Bakalářská práce. SUPERVISOR: Mgr. Josef Horálek, Ph.D. URL: https://theses.cz/id/nkvqm4/.

Bouška, Petr (2010 [cit. 2022-04-08]). *Group Policy - řízení aplikace politik*. Článek. URL: https://www.samuraj-cz.com/clanek/group-policy-rizeni-aplikace-politik/.

Desmond, Brian (květ. 2013). *Active directory*. en. 5. vyd. Sebastopol, CA: O'Reilly Media. ISBN: 978-1-449-32002-7.

Francis, Dishan (srp. 2019). *Mastering Active Directory*. en. 2. vyd. Birmingham, England: Packt Publishing. ISBN: 978-1-78980-020-3.

Holmes, Lee (led. 2013). *Windows PowerShell cookbook*. en. 3. vyd. Sebastopol, CA: O'Reilly Media. ISBN: 978-1-449-32068-3.

Malina, Patrik (2007). *Microsoft Windows PowerShell*. cs. Computer Press. ISBN: 978-80-251-1816-0.

— (2010). *Jak vyzrát na Microsoft Windows PowerShell 2.0*. cs. Computer Press. ISBN: 978-80-251-2732-2.

Payette, Bruce (ún. 2007). *Windows PowerShell in Action*. en. New York, NY: Manning Publications. ISBN: 1932394-90-7.

The Cabinet Office (čvc. 2011a). *ITIL Continual Service Improvement*. en. Ed. The Stationery Office. 2. vyd. Norwich, England: Stationery Office. ISBN: 9780113313082.

— (čvc. 2011b). *ITIL Service Design*. en. Ed. The Stationery Office. 2. vyd. Norwich, England: Stationery Office. ISBN: 9780113313051.

— (čvc. 2011c). *ITIL Service Operation*. en. Ed. The Stationery Office. Norwich, England: Stationery Office. ISBN: 9780113313075.

The Cabinet Office (čvc. 2011d). *ITIL Service Strategy*. en. Ed. The Stationery Office. 2. vyd. Norwich, England: Stationery Office. ISBN: 9780113313044.

— (čvc. 2011e). *ITIL Service Transition*. en. Ed. The Stationery Office. 2. vyd. Norwich, England: Stationery Office. ISBN: 9780113313068.

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor: **Tomáš Valenta**

Studium: I1900664

Studijní program: B1802 Aplikovaná informatika

Studijní obor: Aplikovaná informatika

Název bakalářské práce: **Optimalizace správy Active directory za využití Windows PowerShell**

Název bakalářské práce AJ: Optimization Active Directory via Windows PowerShell

Cíl, metody, literatura, předpoklady:

Cíle práce je navrhnout a implementovat nástroje pro optimalizaci životního cyklu uživatelské identity definované v Active directory za využití Windows PowerShell dle relevantních postupů plynoucích z ITIL. V teoretické části práce autor představí životní cyklus uživatelské identity na principech ITIL (Plan-Do-Check - Act), principy AD pro řízení identit a možnosti Windows PowerShell pro řízení AD. V praktické části autor navrhne, implementuje a otestuje řadu skriptů pro optimalizaci a automatizaci životního cyklu uživatelských identit.

MALINA, Patrik.Jak vyzrát na Windows PowerShell 2.0. Brno: Computer Press, 2010. ISBN 9788025127322.

SEGUIS, Steve.Windows PowerShell 2 For Dummies. Chichester, United Kingdom: John Wiley, 2009. ISBN 9780470371985.

GALLACHER, Liz. ITIL foundation exam study guide. Chichester: Wiley, c2012. ISBN 9781119942757.

Garantující pracoviště: Katedra informačních technologií,
Fakulta informatiky a managementu

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

Datum zadání závěrečné práce: 21.10.2019