Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Návrh a implementace programu v jazyce Powershell pro automatizovaný deployment síťových tiskáren

Ilavský Lukáš

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Lukáš Ilavský

Informatika

Název práce

Návrh a implementace programu v jazyce Powershell pro automatizovaný deployment síťových tiskáren.

Název anglicky

Design and implementation of a Powershell program for automated deployment of network printers.

Cíle práce

Cílem této diplomové práce je návrh a implementace programu v jazyce PowerShell, poskytující alternativní způsob pro automatizaci procesu nasazování síťových tiskáren do pracovních stanic. Součástí práce je analyzování aktuální metody nasazení (deployment) síťových tiskáren do pracovních stanic v prostředí Active Directory u vybrané organizace z Integrovaného záchranného systému (IZS).

Metodika

Metodika řešení diplomové práce bude vycházet ze studia odborné literatury, odborných internetových zdrojů potažmo z osobních zkušeností. V rámci práce proběhne analýza aktuálního stavu domény Active Directory a stávajícího procesu implementace síťových tiskáren u vybrané organizace. Na základě zjištěných poznatků z analýzy a požadavků tiskového administrátora bude navržen program, který bude poskytovat automatizovanou instalaci, aktualizaci a odinstalaci síťových tiskáren a jejich komponent na různých pracovních stanicích. Navržený program bude implementován a testován v reálném prostředí organizace. Výsledkem bude uživatelsky přívětivý program, který bude poskytovat dostatečné informace o stavu zavedených síťových tiskáren na klientské stanici.

Doporučený rozsah práce

50-60 stran

Klíčová slova

Active Directory, Powershell, program, nasazení síťových tiskárnen, Windows Server

---

Doporučené zdroje informací

MALINA, Patrik. Powershell: podrobný průvodce skriptováním. Brno: Computer Press, 2007. ISBN 9788025118160.

Mastering Active Directory: Deploy and Secure Infrastructures with Active Directory, Windows Server 2016, and PowerShell. 2nd edition. Birmingham: Packt Publishing, 2019. ISBN 9781789800203

PATEL, Harshul. Instant Windows PowerShell Functions. India: Packt Publishing Limited, 2013. ISBN 9781849686785

STANEK, William R. Active Directory: kapesní rádce administrátora. Brno: Computer Press, 2009. Microsoft (Computer Press). ISBN 9788025125557

STANEK, William R. Microsoft Windows Server 2012: kapesní rádce administrátora. Přeložil Jiří HUF. Brno: Computer Press, 2015. ISBN 9788025138175.

Předběžný termín obhajoby

2023/24 LS – PEF

Vedoucí práce

Ing. Marek Pícka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 4. 9. 2023

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 3. 11. 2023

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 04. 11. 2023

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Návrh a implementace programu v jazyce Powershell pro automatizovaný deployment síťových tiskáren" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 28. 3. 2024 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Marku Píckovi, Ph.D., za rady a odborné vedení mé diplomové práce. Dále bych chtěl poděkovat mé rodině za psychickou podporu při studiu, především mé ženě za obětavou péči, kterou věnovala našim dětem v průběhu celého studia a psaní této diplomové práce.

Návrh a implementace programu v jazyce Powershell pro automatizovaný deployment síťových tiskáren.

Abstrakt

Diplomová práce se zabývá návrhem a implementací programu v jazyce Powershell, pro alternativní možnosti zavádění síťových tiskáren v reálném prostředí Active Directory. A to při minimalizaci použití doménových zásad s rozšířenou správou AGPM u vybrané organizace z Integrovaného záchranného systému (IZS). V teoretické části práce jsou představeny důležité charakteristiky Powershellu, Active Directory, doménových zásad GPO a principy regulárních výrazů. Praktická část je rozdělená na několik částí, zahrnujících analýzu prostředí, návrh a implementaci programu, testování a nasazení programu do reálného prostředí organizace a vyhodnocení tohoto nasazení. Výsledkem práce je vytvořený program, který instaluje, aktualizuje, odebírá síťové tiskárny a příslušné komponenty na klientských stanicích dle požadované konfigurace tiskového serveru, jehož průběh je generován do html souboru. Generovaný html soubor je možné zobrazit manuálním spuštěním programu uživatelem. Ten na základě strukturovaného a barevného rozlišení jednotlivých stavů může rozlišit případné problémy s instalovanými tiskárnami. Pro administrátory se tak jedná o nástroj, který přináší zjednodušený způsob zavádění tiskáren a rychlou detekci při řešení problému s instalací či konfigurací tiskáren.

Klíčová slova: Active Directory, Powershell, program, nasazení síťových tiskárnen, Windows Server

Design and implementation of a Powershell program for automated deployment of network printers.

Abstract

The thesis deals with the design and implementation of a Powershell program for alternative ways of deploying network printers in a real Active Directory environment. This is while minimizing the use of domain policies with extended AGPM management in a selected organization from the Integrated Rescue System (IRS). The theoretical part of the thesis presents important characteristics of Powershell, Active Directory, GPO domain policies and regular expression principles. The practical part is divided into several parts, including environment analysis, program design and implementation, testing and deployment of the program in a real organization environment and evaluation of this deployment. The result of the work is a program that installs, updates, removes network printers and relevant components on client stations according to the required configuration of the print server, the flow of which is generated into an html file. The generated html file can be viewed by manually running the program by the user. Based on the structured and colourful differentiation of the individual states, the user can distinguish possible problems with the installed printers. For administrators, this is a tool that provides a simplified way of booting printers and quick detection when solving a problem with printer installation or configuration.

Keywords: Active Directory, Powershell, program, network printer deployment, Windows Server

Obsah

1 Úvod

Ve světě informačních technologií je jedním z neustálých úkolů administrátorů v informačních technologií efektivně a spolehlivě nasazovat síťové tiskárny na pracovní stanice. Toto nasazování se obvykle provádí prostřednictvím GPO (Group Policy Objects) v síti využívající AD (Active Directory). GPO umožňují správcům IT centrálně spravovat a konfigurovat operační systémy, aplikace a nastavení uživatelů v AD prostřednictvím politik implementovaných na počítači.

Autor této diplomové práce, který pracuje jako systémový administrátor v jedné ze složek působící v Integrovaném záchranném systému (IZS), byl požádán o pomoc s řešením problému instalací síťových tiskáren na pracovní stanice. Tiskový administrátor organizace dlouhodobě bojuje s problémy při nasazování síťových tiskáren do pracovních stanic. Problém spočívá v důvěryhodnosti mezi tiskovým serverem s operačním systémem Windows 2019 Server a některými klientskými počítači s Windows 8 a Windows 10, které odmítají nainstalovat některé ovladače tiskáren. I když se tiskový administrátor snažil tento problém řešit s administrátory domény dle doporučených nastaveních uvedených na stránkách Microsoft, i tak se problém opakoval. Tento problém neustále zatěžuje oblastní administrátory tím, že musejí prohledávat systémové logy a hledat tak příčinu chybějící tiskárny. Jelikož chyby, které jsou zaznamenány v systémových lozích, jsou obvykle obecné a odkazují na číselné kódy chyb, musejí administrátoři složitě vyhledávat informace na internetu. To obvykle zabere nějaký čas, který by jinak mohli věnovat jiné činnosti.

Jelikož má autor značné zkušenosti se správou systému Windows Server 2012 a 2019 pomocí nástroje PowerShell, navrhl tiskovému administrátorovi zkusit nasazování tiskáren pomocí tohoto nástroje, s čímž ten souhlasil. Vzhledem k tomu, že autor tento problém také párkrát řešil, tak ví, jaký je problém identifikovat příčinu chybějící tiskárny. Někdy je to problém s ovladači, jindy zas lidská chyba při zavedení tiskárny do GPO či chyba systému.

Celý problém s procesem nasazení tiskáren byl také z komplikovaný tím, když organizace zavedla pokročilou správu zásad skupiny (AGPM). Tímto zavedení AGPM nemohl tiskový administrátor nasazovat tiskárny konvenčním způsobem.

Práce se tedy zabývá návrhem a implementací programu pro alternativní způsob zavádění síťových tiskáren za pomocí programu PowerShell. V první části se zabývá analýzou domény AD a GPO, která má za účel seznámení se s prostředím domény a identifikovat jmennou konvenci organizačních jednotek a klientských stanic, které jsou při

návrhu programu využity. Dále je nutné identifikovat skupinové politiky, které jednak zavádějí tiskárny do klientských stanic a politiky, které by mohly omezit či zcela zamezit použití programu PowerShell. Identifikované politiky budou navrženy k odstranění, aby nebyly v konfliktu s programem. Při deskripci programu je využito grafické notace UML, která usnadňuje rychlou orientaci pracovního postupu programu. Dále za pomocí notace BPMN je nastíněn aktuální proces nasazování síťových tiskáren tiskovým administrátorem a jeho následná optimalizace za pomocí využití navrženého programu. Cílem tedy je odstranit neustále vracející se problém s instalací tiskáren a přinést tak tiskovému administrátorovi stabilní nástroj pro zavádění tiskáren. Zároveň oblastním administrátorům či běžným uživatelům umožnit rychlou detekci chybějících tiskáren formou souhrnného výpisu o aktuálním stavu zavedených tiskáren.

2 Cíl práce a metodika

2.1 Cíl práce

Cíl této diplomové práce je navrhnout a implementovat program v jazyce PowerShell, který poskytne alternativní metodu pro automatizaci procesu nasazování síťových tiskáren do klientských stanic. Tento program bude navržen s cílem minimalizovat závislost na GPO (Group Policy Object), což je aktuálně používaná metoda nasazování tiskáren. Ale může způsobit problémy s důvěryhodností ovladačů tiskáren při jejich instalaci ze serveru na klientské počítače. V rámci práce bude rovněž provedena analýza aktuální metody nasazení tiskáren v dané organizaci. Tato analýza poskytne důležitý kontext pro návrh a implementaci alternativního řešení a pomůže identifikovat klíčové výzvy a omezení současného procesu. V kontextu prostředí Active Directory vybrané organizace působící v rámci IZS (Integrovaný záchranný systém) bude program identifikovat, v jaké organizační jednotce se daný počítač nachází. Na základě toho určí, které tiskárny se mají na daný počítač nainstalovat. Bude také schopen spravovat různé typy tiskáren od různých výrobců a generovat HTML sestavy o stavu nasazení tiskáren, které budou snadno přístupné a srozumitelné pro správce IT i běžného uživatele. Výsledný program by měl nejen nabídnout alternativní metodu nasazování tiskáren, ale také zlepšit přehlednost a efektivitu procesu nasazení. A snížit potenciální problémy spojené s důvěryhodností ovladačů tiskáren.

2.2 Metodika

Metodika práce bude zahrnovat kombinaci teoretických studií a praktického vývoje programu. Teoretická část bude zahrnovat studium odborné literatury a internetových zdrojů týkajících se Active Directory, PowerShellu a Group Policy Objects. Praktická část práce bude zahrnovat popis řešeného problému, analýzu aktuálního procesu nasazování tiskáren v cílové organizaci, analýzu aktuálního stavu domény Active Directory. A zároveň budou prozkoumané charakteristiky doménových zásad z důvodu zjištění případného možného konfliktu s navrhovaným programem. Ten by mohl být vlivem nevhodného nastavení doménových zásad zcela blokován ke spuštění. Formou rozhovoru budou získány potřebné informace, jakožto důležité vlastnosti pro klíčové součásti vývoje programu, kterými jsou identifikace funkčních a nefunkčních požadavků na program. Dále na základě analýzy požadavků bude proveden návrh, implementace a testování programu v testovací doméně

postavené na reálném prostředí organizace s pozměněnými daty z důvodu zachování anonymity a bezpečnosti organizace. Výsledný program bude validován na základě jeho schopnosti efektivně a spolehlivě nasazovat tiskárny s minimálním využitím GPO. Dále výstup programu bude logicky uspořádaný a zároveň pro lepší čitelnost bude schopen z výstupu rozlišit, zdali se vyskytla při procesu zavádění síťových tiskáren chyba či nikoliv. V poslední částech práce po nasazení programu do prostředí bude provedeno vyhodnocení nasazení a případné vyskytnuté problémy vysvětleny a opraveny. Dále bude analyzován vlastní organizační systém HelpDesk ke zjištění, zdali byl problém s instalacemi tiskáren zcela eliminován či nikoliv. Poslední řadě budou navrhnuté možnosti na zlepšení či rozšíření funkcionality programu.

3 Teoretická východiska

3.1 Powershell

Jedná se o nástroj vytvořený firmou Microsoft, který lze najít v aktuálních klientských systémech od verze Windows 8 a v serverových systémech od verze Windows Server 2012. Toto prostředí je rozšířením prostředí příkazového řádku, které lze samostatně spustit. Není tedy možné zapnout příkazový řádek a používat v tomto řádku PowerShellové příkazy. Pokud ovšem spustíme konzolové okno Windows PowerShell, lze tak využívat i některé programy, které obvykle používáme v příkazovém řádku a tyto dál můžeme zpracovávat. Význam slova PowerShell vychází ze dvou spojení slov, a to „POWERful“ jako výkonný a „Shell“ známý z UNIXového uživatelského prostředí „BASH“. PowerShell není pouze skriptovacím nástrojem pro spouštění pár příkazů, je to nástroj, který toho umí zdaleka víc v oblasti správy IT technologií. [7]

3.1.1 Příkazy Cmdlets

Cmdlety jsou v PowerShellu vnitřními příkazy prostředí, které po zavolání nespouští novou instanci programu, ale jsou spouštěny v daném prostředí. Pokud bychom to mohli přirovnat v prostředí příkazové řádky (CMD.EXE), tak například příkaz „DIR“ je možné spustit pouze z tohoto prostředí, jinými slovy není možné ho spustit samostatně, protože se jedná o vnitřní příkaz. Pokud bychom zadali například „IPCONFIG“, dostali bychom rovnou textový výstup do konzole. Příkaz „IPCONFIG“ není totiž vnitřním příkazem, ale samostatným konzolovým programem, který nemá grafické rozhraní a je volán externě – spustí se tak nový proces, který se po vygenerování textového výstupu ihned ukončí. [13]

Cmdlet, jiným označením „rutiny“, jsou složeny ze dvou slov oddělených od sebe pomlčkou. Na prvním místě se uvádí sloveso např. „Get, Set, Add, Remove a Format“, který naznačuje způsob operace s příkazem. Na druhém místě za pomlčkou se uvádí podstatné jméno, který ukazuje na objekt, na který se bude sloveso aplikovat. Takové podstatné jméno může být třeba „Process“, „Disk“, „NetAdapter“. Celý název rutiny pak může být například „Get-Process“ nebo „Get-Disk“. [7]

- **Význam sloves**
  a) **GET** – Provádí načtení, zobrazení informací k objektu zadané podstatným jménem za pomlčkou. Například již zmíněný „Get-Process“ zobrazí aktuálně běžící procesy v systému. Výsledek rutiny je zobrazen na obrázku č. 1. [7]
  b) **SET** – Slouží pro nastavení objektu. Například „Set-PsDebug“ aktivuje nebo deaktivuje možnosti PowerShell ladění skriptu nebo příklad „Set-PsBreakpoint“ nastavuje možnost zastavení běhu skriptu v určitém bodě. [1]
  c) **REMOVE** – Odebírá zvolený objekt nebo při použití v souvislosti s předchozím nastavením bodu přerušení, je možné nastavený bod odebrat příkazem „Remove-PsBreakpoint“. [1]
  d) **FORMAT** – Obvykle se používá při formátování výstupu předchozího příkazu do formátu tabulky pomocí příkazu „Format-Table“ nebo seznamu pomocí „Format-List“. [1]
  e) **ADD** – Lze použít pro přidání objektu definovaný podstatným jménem. Například příkazem „Add-WindowsFeature“ lze přidat různé funkce do systému Windows. [1]

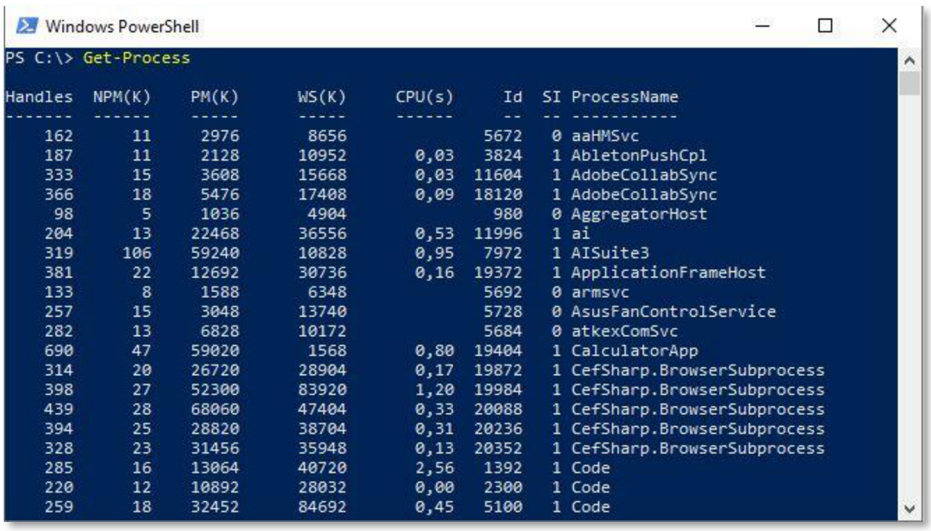
3.1.2 Prostředí PowerShellu

Prostředím, ve kterém lze používat PowerShell, je samotná konzole PowerShell. Tu lze vyvolat pomocí nabídky „start“ zadáním příkazu „powershell“, někdy lze zadat pouze „Pow“ a položka by se měla zobrazit k možnosti ke spuštění. V nabídce se obvykle zobrazí dvě varianty programu ke spuštění, a to buď interaktivní konzole Windows PowerShell, nebo „Windows PowerShell ISE“. [7]

1) **Windows PowerShell**

   V případě využití první varianty se spustí uživatelská konzole podobná příkazovému řádku, avšak mimo černé barvy pozadí se zobrazí pozadí s tmavě modrou barvou a s titulkem okna „Windows PowerShell“. Pomocí této konzole je možné používat jakékoliv příkazy shodné s příkazovou řádkou včetně zkrácených PowerShellových příkazů. Výstup těchto příkazu je obvykle formátovaný do seznamu, tabulek a je možné je i uživatelsky filtrovat. Tyto výstupy lze rovněž nasměrovat do souboru typu CSV, XML. Obrázek č. 1 zobrazuje spuštěnou konzoli Windows PowerShell s příkazem „Get-Process“. [7]

Obrázek 1 - Konzole Windows PowerShell

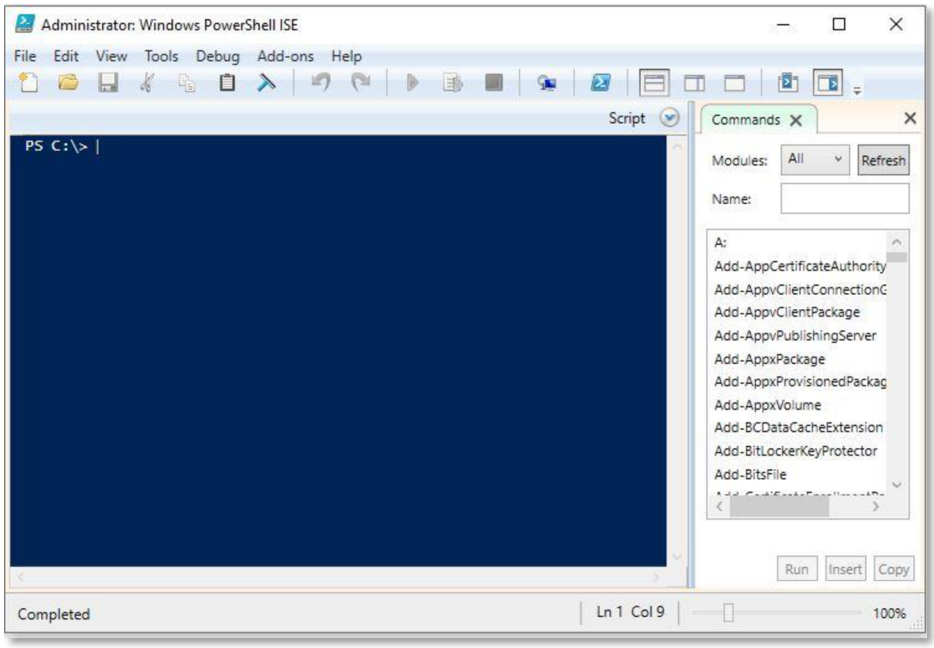


Autor: [7]

2) Windows PowerShell ISE

I když se lze domnívat, že dle názvu „Windows PowerShell Intergované Skriptovací Prostředí“ slouží pouze pro skriptování, tak tomu tak nemusí vždy být. I experti v oboru IT tento nástroj používají pro zadávání konzolových příkazů, protože lze využít vylepšenou kompletaci příkazů pomocí klávesnice TAB. Je lehčí na úpravu vložených příkazů a disponuje nabídkou s možností výběru příkazů „. V systému Windows 8 je nutné vložit pro spuštění „PowerShell_ISE“ anebo lze spustit vložení „Pow“ a kliknutím na „Windows PowerShell“ pravým tlačítkem pro otevření kontextové nabídky, kde se zvolí „Windows PowerShell ISE“ [7]

Obrázek 2 - Integrované skriptovací prostředí (ISE)



Autor: [7]

3.1.3 Bezpečnostní politiky

1) Práva běžného uživatele

Výchozí oprávnění při spuštěném prostředí Windows PowerShellu je bez zvýšených práv, tedy jako běžný uživatel. To však neplatí u serverového systému Windows Server 2012. Zde platí, že při spuštění prostředí se spouští s právy uživatele. Jakmile je tedy prostředí spuštěno, je respektována restrikce systému Windows nad uživatelskými právy, a tedy není možné z prostředí spustit nic, na co nemá přihlášený uživatel právo. [7] [8]

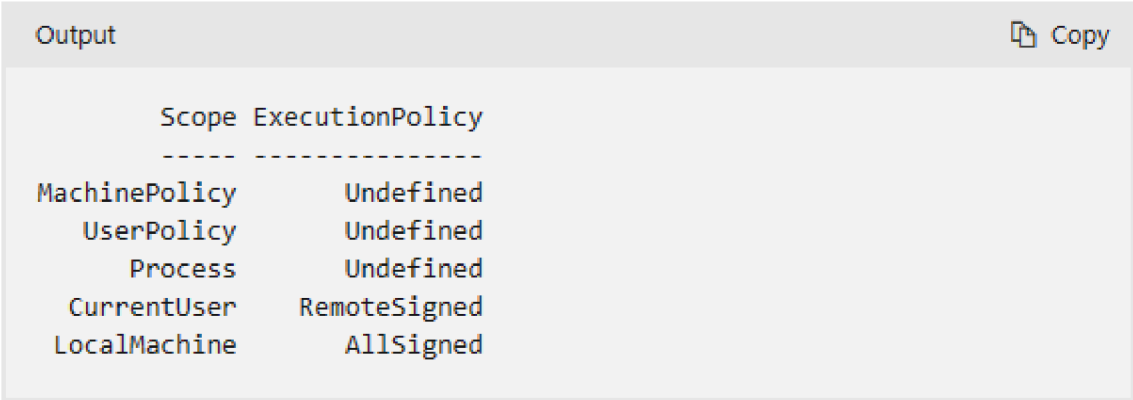
2) Práva správce

Pro případ, že je zapotřebí spustit prostředí s vyššími právy pro určité operace, je nutné spustit prostředí jako administrátor. Lze využít postup dle kapitoly 3.1.1 s tím, že kliknutím pravým tlačítkem na požadovaný program vyvoláme kontextovou nabídku, ze které vybereme „Spustit jako správce“. Pokud je kontrola uživatelských účtů UAC (User Account Control) defaultně povolená, pak se zobrazí dialogové okno, ve kterém povolíme aplikaci změnu v systému. Přičemž samotné spuštění aplikace neprovede žádné změny, ale je již možné ovlivňovat systém. [7] [8]

3) **Zásady spouštění**

Nastavuje pravidla, která umožní nebo zakáže spouštění PowerShellových skriptů v různých scénářích. Pro načtení aktuálních pravidel lze použít příkaz „Get-ExecutionPolicy -list“ jehož výsledkek může být stejný jako je na obrázku č. 3.

Obrázek 3 - Výstup příkazu "Get-ExecutionPolicy -list"



Autor: [14]

Pro nastavení konkrétního pravidla lze použít například „Set-ExecutionPolicy -ExecutionPolicy ByPass“ [14]

a) **AllSigned** – Lze spouštět pouze digitálně podepsané skripty důvěryhodným uživatelem.
b) **Bypass** – Lze spouštět bez varování.
c) **RemoteSigned** – Lze spustit skripty vytvořené na daným počítači a v případě, že je soubor skriptu pochází z internetu, je vyžadován digitální podpis důvěryhodného uživatele. Toto je výchozí nastavení pro serverové systémy Windows.
d) **Restricted** – Nelze spouštět skripty, ale lze spustit konkrétní příkazy.
e) **Undefined** – Pokud je ve všech oborech hodnota „undefined“, pak platí pravidlo „Restricted“ na klientských systémech a „RemoteSigned“ na serverových systémech Windows
f) **Unrestricted** – Před spuštěním skriptu bude uživatel nejdříve varován. [14]

4) **Pravidla AllSigned a RemoteSigned**

Jak již bylo zmíněno v bodě 3) písmena a), c), lze spouštět pouze podepsané soubory skriptů. Aby bylo možné skript podepsat, je potřeba disponovat certifikátem, a to buď vlastním anebo vydanou důvěryhodnou certifikační autoritou. V případě vlastního certifikátu bude soubor skriptu spustitelný pouze na svém počítači. V druhém případě bude možné spouštět skript i na jiném počítači, který disponuje certifikátem kořenové autority. Pro zobrazení certifikátu vhodného k podepsání skriptu lze zjistit pomocí příkazu:

```
Get-ChildItem cert:\CurrentUser\my -codesigning
```

Pokud existuje vhodný certifikát, zobrazí se výstup podobný obrázku č. 4

Obrázek 4 - Výstup příkazu pro zobrazení certifikátu

Výstup Kopírovat

```
Directory: Microsoft.PowerShell.Security\Certificate::CurrentUser\My

Thumbprint                                Subject
----------                                -------
4D4917CB140714BA5B81B96E0B18AAF2C4564FDF  CN=PowerShell User ]
```

Autor: [15]

Pokud existuje certifikát pro podepisování kódů, pak ho lze použít a soubor skriptu s ním podepsat. Pro ukázku je zvolen certifikát dle obrázku č. 4 a soubor skriptu s názvem „add-signature.ps1“ za pomocí příkazu:

```
„$cert = Get-ChildItem Cert:\CurrentUser\My -CodeSigningCert|
      Select-Object -First 1
Set-AuthenticodeSignature add-signature.ps1 $cert“
```

Jakmile dojde k podepsání souboru skriptu digitálním podpisem, lze ho na počítači spustit. [15]

3.1.4 Filtrování, seskupování a třídění

- **Roura (Pipeline)**

  Pro psaní skriptů je důležité znát možnosti používání různých rutin a jejich výstupy, potažmo jejich zřetězování s dalšími rutinami. Je to standard při používání spojování dvou příkazů, například je možné provést restart více počítačů. Jedním příkazem načteme seznam počítačů a odesláním do druhého příkazu provedeme restart. [7]

  Pro používání roury mezi příkazy se používá svislítko **„|“**, jehož princip je velmi snadný. Příkaz před svislítkem provede nějakou operaci a jeho výsledek může druhý příkaz za svislítkem zpracovat. A to bez toho, aniž by se musel předchozí výstup uložit do proměnné. Kromě předávání textových výstupů dalšímu příkazu lze předávat i celé objekty, které lze pak dál zpracovávat. Například příkaz „Get-Service | Format-List“ provede formátovaný výpis s větší podrobností výstupu, než kdybychom použili samotný příkaz „Get-Service“ [13]

- **Třídění výstupu**

  Ve výchozím nastavení se výstup příkazu „Get-Process“ seřadí automaticky dle abecedy. To je proto, že obvykle hledáme konkrétní proces. Abychom však mohli najít proces, který zabírá v paměti největší kapacitu virtuální paměti nebo opačně nejmenší kapacitu paměti, lze zřetězit výstup rutiny „Get-Process“ s rutinou „Sort-Object“ a parametrem „-Property VM“. Ve výchozím nastavení bude výstup s procesy seřazen dle nejmenší zabrané kapacity ve virtuální paměti. Pro tyto případy lze použít parametr „-Descendig“, který provede otočení seřazení zabrané kapacity od největšího k nejmenšímu. [7] [10]

- **Seskupování výstupu**

  Po předchozím třídění je možné provést další zřetězení pomocí rutiny „Group-Object“, která nám seskupí výsledek do skupin dle zadaného parametru „-Property“. Pokud bylo použito předchozí třídění dle konkrétní vlastnosti, pak je důležité použít tuto vlastnost i pro seskupení. V opačném případě nebude výsledek dle očekávání. [7] [9]

- **Filtrování výstupu**

  Dalším možnou zřetězenou rutinou je „Where-Object“, která nám zajistí zkrácení výpisu. Pokud si vezmeme například setříděnou rutinu „Get-Process“ z předchozího bodu, dostaneme výpis všech procesů. Trvalo by poměrně delší dobu, kdybychom měli hledat konkrétní hodnoty. Abychom nemuseli procesy dlouho hledat ve velkém výstupu, můžeme výpis zkrátit. K tomu nám slouží snadná rutina „Where-Object“. Pokud tedy vezmeme z příkladu rutinu „Get-Process“ a zřetězíme ji s rutinou „Wehere-Object“ s parametrem VM a operátorem -gt 1000MB, získáme tím všechny běžící procesy využívající více než 1000MB virtuální paměti. Celý příkaz bude vypadat takto: [7] [11]

```
„Get-Process | Where-Object vm -gt 1000MB“
```

  Filtrovat lze i podle data. To se obzvlášť hodí, pokud řešíme nějaký problém a potřebujeme zjistit, zdali se problém vyskytl i po určitém datu. Některé rutiny mají přímo implementovaný parametr s filtrováním a některé nikoliv. Z tohoto důvodu je právě možnost využít rutinu „Where-Object“, kdy rutinu s nepodporovaným přímým filtrováním přesměrujeme pomocí roury do rutiny „Where-Object“ a upřesníme si tak potřebný výstup. Primárně je vhodné využívat přímé filtrace v rutinách vlevo od svislítka, tím je zajištěno výkonnější a na paměť méně náročnější zpracování. Je to vhodnější právě pro to, že první rutina může vrátit mnoho dat, které se musejí za svislítkem dále zpracovat. Tím vzniká větší výpočetní náročnost na procesor, na paměť a mnohdy i na vstupně-výstupní operace disku. [7] [11]

3.1.5 Zajištění vzdáleného přístupu s WinRM

Jednou z mnoha výhod PowerShellu je, že lze spravovat nejen lokální počítač, ale také vzdálený počítač poměrně jednoduše. Stačí spustit PowerShell a použít příkazy, které disponují parametrem „ComputerName“. Pro výpis těchto parametrů lze použít rutinu „Get-Help“ + zástupný symbol „*“ a parametr „-Parameter“ s hodnotou computername. Výstup můžeme předat do roury pro setřídění podle jména. Kompletní příkaz by mohl vypadat takto: [7]

```
Get-Help * -Parameter computername | sort name | ft name,
synopsis -auto -wrap
```

Pro vzdálené připojení k počítačům se využívají dva protokoly typu DCOM (Distributed Component Object Model) a RPC (Remote Procedure Call). Je však nutné na vzdáleném počítači otevřít ve firewallu mnoho portů a spustit služby, se kterými rutiny chtějí na vzdáleném počítači komunikovat. Powershell využívá pověření účtu implicitně zosobněného uživatele k přístupu ke vzdálenému počítači, pokud ho explicitně nepřidáme do rutiny pomocí parametru -Credential. Ovšem ne každá rutina může disponovat takovým parametrem, to však nebrání využít PowerShell pro vzdálenou správu. Stačí, když se prostředí spustí pomocí přidržením klávesnice „SHIFT“ a pravým tlačítkem vybere spustit jako jiný uživatel. [7] [18]

- **Služba WinRM**

  Rutiny prostředí PowerShell využívající příkazy pro vzdálenou komunikaci, používají na vzdáleném počítači službu WinRM (Windows Remote Management). Služba WinRM na klientských stanicích je ve výchozím nastavení vypnutá, kdežto od serverové verze Windows Server 2012 je ve výchozím stavu nastavená na příjem vzdálených příkazů. [7]

- **Povolení služby WinRM na klientských počítačích**

  Pro povolení příjmu vzdálených příkazu postačí spustit v prostředí PowerShell příkaz „Enable-PsRemoting -Force“, tedy za předpokladu, že prostředí běží s povýšenými právy. Parametr -Force vynechá potvrzovací dotazy pro nastavení těchto kroků: [7]

  1) Provede se spuštění nebo restartování služby WinRM.
  2) Nastaví se automatické spuštění služby WinRM.
  3) Nastaví příjímací proces, aby akceptoval požadavky z jakékoliv adresy IP
  4) Na firewallu nastaví pravidla pro příjem dat protokolu WS_Management.
  5) Proces s názvem Microsoft.powershell nastaví pro naslouchání požadavkům.
  6) Proces s názvem Microsoft.powershell.workflow nastaví pro naslouchání požadavkům.
  7) Proces s názvem Microsoft.powershell32 nastaví pro naslouchání požadavkům.

Pokud vše proběhlo v pořádku, je možné tuto konfiguraci ověřit pomocí příkazu „Test-WSMan“ s parametrem „-Computername“ a patřičný název vzdáleného počítače. [7] [18]

3.1.6 Proměnné a jejich charakteristiky

Proměnné v prostředí Windows PowerShell mají své výhody a nevýhody. Při používání proměnných je totiž nemusíme před použitím definovat, protože vložením hodnoty do proměnné se rovnou definují. Proměnné využíváme k ukládání určitých hodnot k dalšímu pozdějšímu použití. Proměnné mohou být různých typu jako je text, číslo a může být i celý objekt. Každá proměnná musí být označená symbolem „$“ na začátku názvu proměnné, aby byla odlišná od ostatních prvků v prostředí. Například zavoláním rutiny „Get-Process“ a uložením jejího výstupu do proměnné „$process“, lze vytvořit tímto příkazem: [7] [12]

```
$process = Get-Process
```

Tím jsme vytvořili a naplnili proměnnou $process s objekty běžících procesů. Prostředí PowerShellu si vytváří své vlastní systémové proměnné, jejíž názvy by měly být při vytváření vlastních proměnných vynechány. V následující zkrácené tabulce č. 1 je popsán význam speciálních proměnných a jejich názvy. [7] [12]

Tabulka 1 - Speciální proměnné prostředí Windows PowerShell

$	Obsahuje první token posledního řádku zadaného v prostředí.
$$	Obsahuje poslední token posledního řádku zadaného v prostředí
$_	Aktuální objekt zřetězení
$?	Obsahuje úspěšný či neúspěšný stav posledního příkazu.
$Error	Pokud dojde k chybě, je objekt chyby uložen do proměnné $error
$Match	Zatřiďovací tabulka, která obsahuje položky nalezené operátorem -match.
$MyInvocation	Informace o aktuálně spouštěném skriptu nebo příkazovém řádku.
$LastExitCode	Úkončovací kód posledně spuštěné nativní aplikace.
$true	Logická hodnota true.
$False	Logická hodnota false.
$null	Objekt typu null.

Zdroj: [7]

3.1.7 Charakteristika speciálních funkcí

- **Podmínka IF**

  Příkaz „IF“ se používá pro vyhodnocení jedné či více podmínek zadané do kulatých závorek. Vyhodnocení je provedeno za pomocí operátoru uvedených v tabulce č. 2, kde můžou nastat dva výsledky typu boolean tzn. pravda nebo nepravda. Syntaxe příkladu je pro ukázku následující: Za příkaz „IF“ je uvedena podmínka, která je ohraničena kulatými závorkami např. pro vyhodnocení proměnné $a s hodnotou 4, která je položena otázkou rovnosti k číslu 5. Za tímto vyhodnocením je očekáván blok skriptu ohraničenými složenými závorkami, který se provede v případě podmínky vyhodnocené jako pravda. Pro případ, že bychom chtěli zachytit a provést operaci i v případě vyhodnocení nepravdy, je možné za složené závorky uvést slovo „ELSE“ a opět vložit blok skriptu do složených závorek. Je také možné napojovat více podmínek tak, že ihned za „ELSE“ napojíme znovu „IF“ a vložíme opět podmínku do kulaté závorky. Celý příkaz podmínky „IF“ a „ELSEIF“ by mohl vypadat takto: [7] [16]

```
$a = 4
 IF ( $a -eq 5 )
{
‘$a equals 5‘
}
 ElseIf ($a -eq 3)
{
‘$a is not equal 5‘
}
Else
{
 ‘$a does not equal 3 or 5‘
}
```

Spojování příkazu „IF“ a „ELSEIF“ lze používat do nekonečna, avšak tím začne být zdrojový kód značně nepřehledný a složitý pro čtení. Je tedy pravidlem se takovým napojováním vyvarovat, protože existuje snadnější příkaz pro takové účely. Tím příkazem je „SWITCH“, který bude popsán v dalším bodě. [7]

Tabulka 2 - Operátory porovnání

-eq	rovná se
-ne	nerovná se
-gt	vetší než
-ge	větší než nebo rovno
-lt	menší než
-le	menší než nebo rovno
-like	operátor zástupného znaku
-notlike	operátor zástupného znaku
-match	porovnání regulárního výrazu
-notmatch	porovnání regulárního výrazu

Zdroj: [7]

- **Příkaz Switch**

  Pomocí příkazu „SWITCH“ lze provádět další větvení skriptu obdobné příkazu „IF“ s tím rozdílem, že hodnota proměnné nemusí být porovnávaná podmínkou. Konstrukce příkazu „Switch“ je jednoduchá a je lépe čitelná. Vytvoří se příkazem „Switch“ a do kulatých závorek se vloží proměnná, kterou chceme vyhodnotit. Následuje blok skriptu ve složených závorkách. Do tohoto bloku vkládáme očekávanou hodnotu, za kterou opět vložím blok skriptu, který se má vykonat. Tyto bloky se mohou opakovat pro novou nebo stejnou hodnotu. Na pořadí zde nezáleží, protože při vyhodnocení nedojde k přerušení v případě výskytu očekáváné hodnoty. Při zpracování daného blok skriptu se pokračuje dalším vyhodnocováním. Toto vyhodnocení lze přerušit při zpracovávání bloku následujícím příkazem „BREAK“, který ukončí příkaz „Switch“. Pokud bychom zapomněli definovat všechny možné výsledky, je zde možné použít obdobný alternativní blok jako je tomu při vyhodnocení „ELSE“. Zde se používá klíčové slovo „DEFAULT“, za kterým následuje poslední blok skriptu, který se má provést v případě nenalezení žádné předchozí shody. Celá syntaxe příkazu by mohla vypadat následovně: [7] [17]

```
$a = 2
Switch ($a)
{
    1 { '$a = 1' ; break }
    2 { '$a = 2' ; break }
    3 { '$a = 3' ; break }
    Default { 'unable to determine value of
$a' }
}
"Statement after switch"
```

- **Iterace pomocí ForEach**

  Pomocí příkazu „ForEach“ lze procházet jednotlivé položky v poli nebo kolekci a jednotlivé položky jsou při iteraci vraceny do proměnné, se kterou lze dále pracovat v bloku skriptu. K opakované iteraci dochází do té doby, dokud se neprojde celé pole či kolekce. Pro názornou ukázku syntaxe příkazu si nadefinujeme proměnou $ary s hodnotami 1 až 5 a nad tímto polem provedeme příkaz ForEach, který postupně vloží každou hodnotu do proměnné $i a vypíše ji: [7]

```
$ary = 1..5
foreach ($i in $ary)
{
      $i
 }
```

  Lze také použít zřetězení a předat tak hodnoty pomocí roury a využít příkaz „ForEach-Object“, který dokáže zpracovat zřetězený vstup. Pro práci s výstupem lze využít systémovou proměnou „$_“. Ta byla představena v kapitole 3.1.6. Odpadá tak vytváření pomocné proměnné např. $i, která se používá pro iteraci nad polem. Uveďme si příklad syntaxe příkazu „ForEach-Object“, který do roury odešle hodnoty 1 až 5 a každou hodnotu vypíše: [7]

```
    „1..5 | ForEach-Object { $_ }“
```

  V případě, že bychom potřebovali přerušit aktuálně probíhající iteraci, lze použít přerušovací příkaz „BREAK“. Například dejme tomu, že potřebujeme zastavit probíhající iteraci na čísle 3. Toto se dá aplikovat pomocí podmínky „IF“. Příkaz pro přerušení by vypadal následovně:

```
„$ary = 1..5
foreach ($i in $ary)
{
     If ($i -eq 3) { Break }
     $i
}„
```

Jelikož se proměnná $i nachází za podmínkou IF, tak při splnění dané podmínky nebude již číslo 3 vypsáno. [7]

- **Iterace pomocí For**

  Příkaz pro iteraci pomocí příkazu „FOR“ využívá tradiční způsob opakování na základě vyhodnocení podmínky přímo zadané v příkazu a pevným začátkem. Při zadávání příkazu stanovíme do kulatých závorek proměnou, která nám stanoví počáteční hodnotu, následuje podmínka oddělená z obou stran středníkem „ ; “ a ukončením inkrementačním operátorem proměnné. Proměnná nám automaticky zvyšuje hodnotu o jednotku. Následující příkaz provede výpis písmen od A do F, kde proměnnou $i nastavíme na hodnotu 65 a iterace bude pokračovat, dokud proměnná bude menší nebo rovna číslu 70: [13]

```
For ($i = 65; $i -le 70; $i++)
{
    ([char]$i)
}
```

3.2 Active Directory

Active Directory (AD) je v oblasti IT označení pro adresářové služby založené na protokolu LDAP (Lightweight Directory Access Protocol), který vyvinula společnost Microsoft pro systémy z rodiny Windows NT. Tyto služby byly poprvé zavedeny ve Windows 2000 jako vylepšení původních Windows Domén a nabízely možnost centralizovaného ukládání informací v hierarchické databázi ve tvaru stromu. Databáze AD je uložena na tzv. řadiči domény (Domain Controller), jehož úkolem v síti je mimo jiné ověřování (autentizace), udělování oprávnění (autorizace) uživatelům, počítačům a různým dalším službám. Počínaje verzí Windows Server 2008 byly do AD začleněny další funkce a služby. [6]

3.2.1 Adresářová služba

Adresářová služba centralizuje informace pro správu a využití síťových prostředků. Zajišťuje ověření přístupu, spravuje identity a kontroluje vztahy mezi zdroji. Je propojena s bezpečnostními funkcemi operačního systému. Umožňuje definici a údržbu sítě a podporuje uživatele i správce při výběru a vyhledávání zdrojů, např. serverů s Windows Server 2008. S narůstající sítí se její význam zvyšuje. [6]

3.2.2 Protokoly komunikace

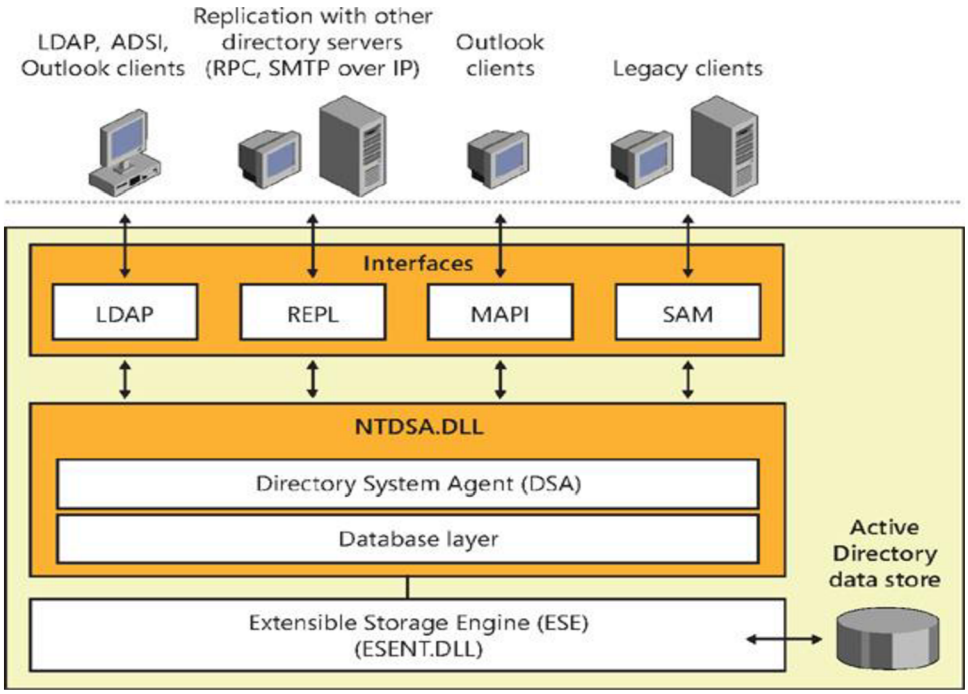
Protokol **LDAP** využívá AD pro vzájemnou kompatibilitu komunikace s jinými adresářovými službami a umožňuje příjem klientů komunikujících na různých rozhraních. AD používá jako výchozí protokol LDAP, což je standardní protokol pro adresářové služby. Veškerý přenos pomocí tohoto protokolu je standardně podepisován a šifrován. Podepisování a šifrování komunikace zajišťuje důvěru přenášených dat. Takto obdržená data zaručují, že jsou od důvěrného odesílatele a po dobu přenosu nebyla změněná.

Protokol **REPL** je využíván službou AD pro vzájemnou výměnu datových informací mezi servery z rodiny Microsoft Windows takzvanou „vzájemnou replikaci dat“.

Pro starší aplikace klientů např. aplikace Outlook používá služba AD pro vzájemnou výměnu zpráv protokol **MAPI.** Architekturu adresářové služby lze vidět na obrázku 5.

K autentizaci a autorizaci používají služby AD nativní protokol **Kerberos.** Kerberos zajišťuje ochranu výměny dat tím, že zabraňuje odposlouchávání nebo opakování stejné komunikace, čímž zaručuje integritu dat. [6] [19]

Obrázek 5 - Architektura adresářové služby



Autor: [19]

3.2.3 Zabezpečení v AD

Stejně jako známe zabezpečení oprávnění např. u NTFS (New Techology File System) na systémech Microsoft Windows, je služba AD součástí zabezpečení Microsoft Windows serveru a klientských stanic. Je tedy možné nastavovat různé doménové skupiny k přístupům jednotlivým síťovým zařízením a doménovým prostředkům. A to na jakékoliv úrovni v celé doménové síti. Pomocí zásad zabezpečení lze nastavovat pro uživatele a počítače různá nastavení povolených akcí a omezení k různým vlastnostem distribuovaných prostředků. [6]

3.2.4 Domény DNS

Pro využívání AD v síti je nezbytné mít nainstalovaný server se službou DNS (Domain Name System). AD je zcela závislá na její službách. Jedná se o hierarchickou strukturu doménových názvu, která je totožná se strukturou služby AD. Samozřejmě veřejné sítě jako je internet nebo soukromé sítě – privátní, mají odlišnou hierarchii. Služba DNS se

využívá k nalezení síťových prostředků pomocí názvu zařízení. Zajistí převod snadno zapamatovatelného názvu zařízení na síťovou adresu IP (Internet Protocol), která je důležitou části síťového prostoru. Každé zařízení v doménové sítí AD má za svým názvem připojenou příslušnou doménu, takovému názvu říkáme **FQDN** (Fully Qualified Domain Name). Například serverové zařízení s názvem „server34" v doméně „microsoft.com" má FQDN „server34.microsoft.com". Při dotazu s tímto FQDN na server DNS by mohl server vrátit IP adresu daného zařízení, například 192.168.100.35. Důležitým prvkem serveru DNS je jeho aktualizace záznamů, ty totiž mohou zastarat. Proto je nutné se o aktualizaci záznamů starat. Pokud k tomu nebude docházet, může nastat situace, že bude více záznamů názvu zařízení v databázi serveru DNS a při dotazování na název vrátí server zastaralou adresu, která nemusí být již platná pro dané zařízení. Význam uvedeného příkladu FQDN vyjadřuje úrovně domén, kdy název „Microsoft" je subdoménou a „com" je doménou nejvyšší úrovně. Úrovně domén se dělí na [6] [20]

- **Domény nejvyšší úrovně**

  Ty mohou být rozděleny dle:

  a) **geografické polohy**

  mají dvoupísmenné označení kódů, např. pro Českou republiku mají servery koncovku „cz"

  b) **typu organizace**

  mají třípísmenné označení, např. komerční organizace mají servery koncovku „com"

  c) **funkce použití**

  mají též třípísmenné označení, např. pro americkou armádu koncovku „mil"

3.2.5 Rozdělení struktury

AD a její vrstvy lze použít k popisu struktury adresáře. Tyto vrstvy jsou rozděleny na fyzické a logické vrstvy. A ty jsou od sebe izolované. Fyzická vrstva má na starosti ověřování a vzájemné propojování lokalit a jejich podsítě. Logická vrstva zajišťuje a ověřuje, jak jsou objekty v adresáři zobrazovány. [6]

- **Fyzická vrstva**

  Mezi fyzické komponenty lze řadit lokality (**sites**) a jejich podsítě. Lokalita může mít jednu i více podsítí. Podsítě jsou mezi sebou fyzicky propojeny spojením LAN nebo WAN o minimální rychlosti 512 kb/s. Podsíť se skládá z určitého

rozsahu IP adres např. podsíť s označením 10.1.11.0/24 je vytvořen z rozsahu IP adres 10.1.11.0 až 10.1.11.255, kde první a poslední IP adresa se nepřiděluje žádnému síťovému zařízení.

- **Logická vrstva**

  Mezi logické vrstvy patří organizační jednotky, domény, stromy domén a lesy domén.

  a) **Organizační jednotka (OU)**

  Je nejmenší možným rozsahem v doméně, na kterou lze přenést různé pravomoci. Případně lze OU přidat k jiné OU a vytvořit tak další hierarchii v doméně.

  b) **Domény**

  Doména představuje logický soubor síťových prvků, např. počítačů, uživatelů a zařízení, kteří využívají společnou AD databázi.

  c) **Stromy domén a lesy**

  Všechny domény v AD mají svůj DNS název. Pokud jedna nebo více těchto domén sdružuje stejná adresářová data, jsou označeny jako LES.

3.2.6 Řadiče domény

Řadič domény je nejdůležitější součásti domény AD, protože skrze něj probíhají veškeré interakce uživatele, jako např. žádosti o ověření, kdy se například uživatel snaží přihlásit na nějaký server ve stejné doméně. Server po úspěšné autentizaci uživatele buď přidělí, nebo zamítne přístup, a to na základě odpovědi od doménového řadiče. Role řadiče není však jen v ověřování, ale také ukládá veškeré doménové informace. Informace replikuje mezi dalšími řadiči v doméně. Z tohoto důvodu je důležité pro zachování kontinuity funkčnosti celé domény mít alespoň dva řadiče domény. [6] [21]

3.3 Zásady skupiny

Zásady skupiny jsou sady konfiguračních nastavení aplikovaných na profily uživatelů a počítačů. Díky nim lze jednoduše spravovat a automatizovat opakované či složité úkony, jako je autoinstalace softwaru či omezení povolených programů k instalaci. Zásady skupiny umožňují seskupování konfiguračních možností do konkrétních kategorií, což umožnuje přizpůsobení prostředí každého počítače v souladu s požadavky firmy. V zásadě poskytují rámec pro vytvoření univerzálních firemních směrnic. Tyto zásady mohou ovlivňovat různé oblasti operačního systému, například bezpečnost, přesměrování složek nebo spuštění uživatelských skriptů. [4] [22]

3.3.1 Nastavení zásad skupiny

Zásady skupiny umožňují centrální řízení a upravení prostředí na klientských stanicích. Jednou stanovená pravidla obvykle nelze uživateli měnit. Tyto zásady kontrolují aspekty jako bezpečnost, instalaci programů, konfiguraci sítě a další charakteristiky operačního systému. Při manipulaci s těmito zásadami v systémech Microsoft Windows je důležité zvážit několik zásadních aspektů:

- **Umístění v registru:** Většina nastavení zásad se ukládá do specifické sekce registru systému, která je určena pro zásady.
- **Vynucení nastavení:** Tato nastavení jsou vynucena administrátorem a nemohou být jednoduše změněna uživateli.
- **Omezení uživatelského rozhraní:** Možnost měnit tyto volby přímo z uživatelského rozhraní může být deaktivována, což znemožní uživatelům je upravit.
- **Automatické aktualizace:** Nastavení zásad se pravidelně aktualizují, často z centrálního místa jako je AD.
- **Kompatibilita s aplikacemi:** Pro efektivní fungování některých nastavení zásad je nutné, aby používané aplikace byly kompatibilní s tímto typem administrativního řízení.
- **Neměnnost původního nastavení:** Aplikace zásad na počítač nebo uživatele nemění původní konfiguraci systému; místo toho se přidá nová vrstva nastavení.
- **Obnovení původního nastavení:** V případě odstranění nastavení zásad je původní konfigurace obvykle obnovena, což umožňuje návrat k výchozím parametrům.

Tyto aspekty zdůrazňují komplexnost a význam správného používání nastavení zásad skupiny. [4] [22]

3.3.2 Předvolby zásad skupiny

Předvolby zásad skupiny jsou jakýmsi doplňkem ke standardním zásadám skupiny. Zatímco klasické zásady skupiny jsou striktní a neměnné po jejich nastavení administrátorem, předvolby zásad skupiny umožňují uživatelům provádět úpravy v předdefinovaných nastaveních. Tento přístup poskytuje větší flexibilitu jak pro správce, tak pro koncové uživatele.

Předvolby zásad skupiny v systémech Microsoft Windows mají několik základních charakteristik, které je odlišují od tradičních zásad skupiny. Zde jsou hlavní rysy:

- **Lokalizace v registru:** Předvolby zásad skupiny jsou uloženy v těch částech registru, kde ukládají svá nastavení také aplikace a operační systém.
- **Nepovinná aplikace:** Na rozdíl od tradičních zásad skupiny předvolby nejsou vynuceny, což znamená, že uživatelé mohou tato nastavení změnit.
- **Upravitelnost uživatelem:** Uživatelé mohou měnit nastavené předvolby přímo z uživatelského rozhraní.
- **Flexibilita aktualizací:** Předvolby mohou být buď pravidelně aktualizovány, nebo nastaveny tak, aby byly aplikovány pouze jednou.
- **Široká kompatibilita:** Fungují i s aplikacemi, které nemají speciální podporu pro zásady skupin.
- **Přepsání existujících nastavení:** Pokud jsou předvolby aplikovány, přepíší původní, již existující nastavení.
- **Návrat k původnímu stavu:** V případě odstranění předvoleb se původní nastavení automaticky neobnoví.

Tyto rysy činí předvolby zásad skupiny flexibilním a užitečným nástrojem pro administrátory, kteří chtějí kombinovat centrální řízení s určitou mírou uživatelské autonomie. [4]

3.3.3 Objekty zásad skupiny

Centralizovanou manipulaci s počítači a uživatelskými účty v prostředí AD usnadňují objekty zásad skupiny (GPO). Pomocí GPO lze přizpůsobit uživatelské nastavení na základě konkrétních potřeb organizace a kategoricky oddělit různé skupiny nastavení.

V praxi GPO slouží jako pokyny, které lze vynutit v celé síti. K ovládání operačního systému lze použít sadu konfiguračních parametrů a volitelných nastavení známých jako zásady skupiny. To zahrnuje nastavení jako je zabezpečení, skripty, instalace programů, struktura adresářů a další. [4] [5]

3.3.4 Typy objektů zásad skupiny

Objekty zásad skupiny mohou být v prostředí Active Directory rozděleny podle různých kritérií, jako jsou rozsah a účel. Několik běžných kategorií zahrnuje:

- **Místní objekty zásad skupiny (LGPO):** Tyto objekty jsou uloženy přímo na počítači a platí pouze pro něj. Jsou základní úrovní GPO a aplikují se i když počítač není členem žádné domény.
- **Objekty GPO pro Lokalitu (Site):** Jsou aplikovány na všechny počítače a uživatele, kteří jsou členy určitého síťového místa v Active Directory.
- **Objekty GPO pro Doménu:** Jsou aplikovány na všechny počítače a uživatele v rámci celé Active Directory domény.
- **Objekty GPO pro Organizační Jednotku (OU):** Jsou aplikovány na počítače a uživatele v rámci specifické organizační jednotky v Active Directory.
- **Objekty GPO pro Podřízenou Organizační Jednotku:** V Active Directory může být hierarchická struktura organizačních jednotek a GPO mohou být aplikovány na podřízené OU. Nastavení z těchto GPOs jsou děděna od nadřazených OU, pokud není nastavena možnost "Block Inheritance" (blokování dědění).

Různé typy GPO umožňují granulární kontrolu nad nastaveními a pravidly v rámci organizace, což administrátorům usnadňuje správu a zabezpečení sítě. [4] [22]

3.3.5 Propojení GPO

Propojování GPO je proces, ve kterém se konkrétní vytvořený objekt zásad skupiny přiřadí k určité organizační jednotce, doméně nebo síťovému místu v Active Directory. Tímto způsobem se nastavení v GPO aplikují na počítače a uživatele, kteří spadají do dané organizační jednotky.

Veškeré propojování a vytváření GPO se provádí prostřednictvím nástroje Group Policy Management Console (GPMC), ve kterém administrátor vybere GPO a propojí jej k vybrané organizační jednotce. Propojení může být také nastaveno tak, že jedno GPO je propojeno s více OU, doménami nebo místy podle potřeb organizace.

Propojování GPO je flexibilní a umožňuje administrátorům vytvářet hierarchické a přehledné struktury nastavení, které se dají snadno spravovat a aktualizovat. Je také možné nastavit filtry na základě vlastností uživatelů nebo počítačů (například členství ve skupině), což umožňuje ještě preciznější aplikaci zásad. [4] [5]

3.3.6 Pokročilá správa GPO

Pro pokročilou správu zásad skupiny je k dispozici nástroj AGPM (Advanced Group Policy Management). AGPM je rozšíření standardního nástroje pro správu zásad skupiny, známého jako GPMC (Group Policy Management Console). Toto rozšíření přináší větší zabezpečení a kontrolu v zacházení s doménovými objekty zásad skupiny a má dvě hlavní komponenty, jednu na straně serveru a jednu na straně klienta.

V AGPM se objekty zásad skupiny (GPO) klasifikují do dvou hlavních typů: řízené a neřízené. Zatímco neřízené GPOs jsou uloženy pouze produkčním prostředí, řízené GPOs se udržují jak v produkčním prostředí, tak ve zvláštním archivu mimo síť. Před provedením jakýchkoliv změn na řízeném GPO je potřeba jej "vyjmout" z tohoto archivu (proces zvaný Check Out). Jakmile jsou změny dokončeny, upravený GPO se vrátí do archivu a jeho nasazení (Deploy) do produkčního prostředí může provést pouze autorizovaný administrátor.

AGPM také nabízí možnost vrátit GPO do předchozího stavu nebo znovu aplikovat dřívější nastavení. Dokonce umožňuje obnovit smazané GPO i jejich spojení z „odpadkového koše". Tento systém pro kontrolu a řízení změn je klíčovou částí funkcionalit AGPM. [4]

3.4 Regulární výrazy

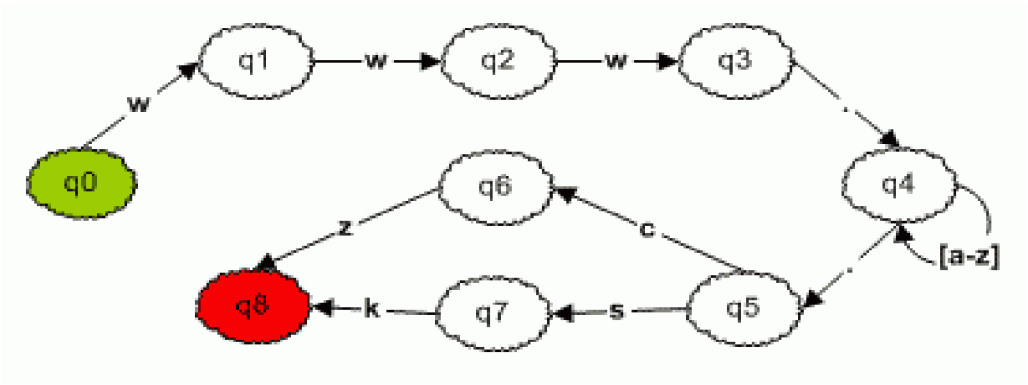
Regulární výrazy jsou výkonným nástrojem pro práci s textem, které lze najít v různých aplikacích a textových editorech. Například v prostředí Linuxu snad neexistuje editor, který by neumožňoval regulární výrazy. Jejich princip fungování je podobný jako "všeobecné zástupné znaky", které lze použít při hledání souborů. [2]

3.4.1 Princip fungování

Regulární výrazy fungují tak, že vyhodnocují textový vstup s využitím konečného automatu. Ten dokáže zjistit, zda vstupní text odpovídá kritériím stanoveným v regulárním výrazu. Automat funguje na principu orientovaných grafů, kde uzly představují stavy automatu, a orientované hrany symbolizují přechody mezi nimi. Některé uzly mají označení koncového bodu, kterým se říká, že pokud po přečtení posledního znaku vstupu automat skončí v tomto bodě, pak automat vstup přijal.

Jako příklad si vezměme regulární výraz: www.[a-z]*.(cz|sk). Tento výraz vyhodnocuje, zda vstupní text splňuje formát webové adresy s koncovkou ".cz" nebo ".sk". Proces vyhodnocování tohoto výrazu by začal v počátečním stavu, odkud by se postupně přesouval mezi stavy v závislosti na znacích vstupu. Pokud vstup splňuje kritéria definovaná výrazem, automat končí v koncovém stavu a vstup je přijat. [2]

Obrázek 6 - Deterministický konečný automat



Autor: [2]

Regulární výrazy jsou velmi užitečné pro zpracování textu, protože usnadňují definici složitých vzorů a gramatiky a přitom jsou relativně rychlé. Existují však gramatická pravidla, které regulární výrazy nezvládnou, jako je například bezkontextová gramatika.

Mnoho implementací regulárních výrazů je populárních, včetně implementací Perl5+, AWK, Vim a GNU grep. Tyto implementace poskytují více přístupů pro práci s regulárními výrazy, včetně vyhledávání, odkazování a úprav textu. Jazyky jako jsou Perl, Python, Ruby, Java a C# mají implementaci regulárních výrazů nativně podporovanou. Jiné jazyky, jako je C nebo C++, nemají nativní podporu regulárních výrazů, lze ji však získat z internetu jako doplňkovou knihovnu. [2]

3.4.2 Metaznaky

Metaznaky v regulárních výrazech jsou speciální znaky, které mají v rámci těchto výrazů zvláštní funkce. Také nejsou interpretovány jako běžné znaky, ale slouží k řízení vzorců a jejich chování. Zde jsou některé základní metaznaky v regulárních výrazech:

Tabulka 3 - Metaznaky v regulárních výrazech

Metaznak	Popis
.	Zastupuje jakýkoliv jediný znak s výjimkou nového řádku.
*****	Označuje, že předchozí znak nebo skupina znaků může být opakována 0 nebo vícekrát.
+	Označuje, že předchozí znak nebo skupina znaků musí být opakována alespoň jednou.
?	Označuje, že předchozí znak nebo skupina znaků může být opakována 0 nebo 1 krát.
{n}	Označuje, že předchozí znak nebo skupina znaků se musí opakovat přesně n-krát.
{n,}	Označuje, že předchozí znak nebo skupina znaků se musí opakovat alespoň n-krát.
{n,m}	Označuje, že předchozí znak nebo skupina znaků se musí opakovat alespoň n-krát, ale ne více než m-krát.
^	Určuje začátek řádku.
$	Určuje konec řádku.
[]	Definuje třídu znaků. Například, [abc] odpovídá jakémukoliv jednomu znaku, kterým je a, b, nebo c.
[^]	Definuje třídu znaků, které nejsou zahrnuty. Například, [^abc] odpovídá jakémukoliv jednomu znaku, který není a, b, nebo c.
()	Skupina znaků nebo výrazů.
****	Escapovací znak, používá se k ošetření speciálních znaků, nebo k označení speciálních sekvencí.
\d	Označuje jakékoliv číslo od 0 do 9.
\D	Označuje jakýkoliv znak, který není číslo.
\w	Označuje jakýkoliv alfanumerický znak včetně podtržítka.
\W	Označuje jakýkoliv znak, který není alfanumerický ani podtržítko.
\s	Označuje jakýkoliv whitespace (mezery, tabulátory, nové řádky).
\S	Označuje jakýkoliv znak, který není whitespace.

Autor: [2]

Toto jsou pouze základní metaznaky v regulárních výrazech. Existuje mnoho dalších metaznaků a technik, které umožňují vytvářet sofistikovanější vzorce. [3]

3.4.3 Kvantifikátory

Kvantifikátory v regulárních výrazech jsou speciální znaky nebo sekvence znaků, které specifikují, jak často se může daný vzorec vyskytovat. Následuje přehled některých základních kvantifikátorů: [3]

Tabulka 4 - Kvantifikátory v regulárních výrazech

Kvantifikátor	Popis	Příklad	Příklad význam
{n}	Označuje, že předchozí znak nebo skupina znaků se musí opakovat přesně n-krát.	**a{3}**	Odpovídá "aaa".
{n,}	Označuje, že předchozí znak nebo skupina znaků se musí opakovat alespoň n-krát.	**a{2,}**	Odpovídá "aa","aaa","aaaa", atd.
{n,m}	Označuje, že předchozí znak nebo skupina znaků se musí opakovat alespoň n-krát, ale ne více než m-krát.	**a{2,4}**	Odpovídá "aa","aaa","aaaa", ale ne "a" nebo "aaaaa".
*	Označuje, že předchozí znak nebo skupina znaků může být opakována 0 nebo vícekrát.	**a***	Odpovídá "","a","aa","aaa", atd.
+	Označuje, že předchozí znak nebo skupina znaků musí být opakována alespoň jednou.	**a+**	Odpovídá "a","aa","aaa", atd., ale ne "".
?	Označuje, že předchozí znak nebo skupina znaků může být opakována 0 nebo 1 krát.	**a?**	Odpovídá "a" nebo "".

Autor: [3]

3.4.4 Příklady regulárních výrazů

V následující tabulce č. *5* jsou uvedeny všechny metaznaky používané v regulárních výrazech, jejich popis a jak je lze použít v praxi. Každý metaznak má svou specifickou funkci při vytváření vyhledávacích vzorů. Význam metaznaků se může měnit v závislosti na jejich pozici a kontextu. Uvedené příklady ukazují, jak jednotlivé metaznaky mohou být použity pro zpracování textu. Celkově tato tabulka slouží jako užitečný průvodce pro pochopení a aplikaci metaznaků v regulárních výrazech. [3]

Tabulka 5 - Příklady regulárních výrazů

Regulární výraz	Odpovídá...
`a+`	sekvence písmen `a` (1 a více znaků)
`a*`	sekvence písmen `a` (0 a více znaků)
`o?kov`	`okov` či `kov`
`tel(efon)?`	`tel` či `telefon`
`telef(on|ax)`	`telefon` či `telefax`
`[0-9]|[1-9][0-9]`	čísla 0 až 99
`\d{2}`	sekvence dvou číslic desítkové soustavy (`00`, `01`, ...,`98`, `99`)
`[0-9a-fA-F]|[1-9a-fA-F][0-9a-fA-F]+`	hexadecimální čísla
`(19|20)\d{2}`	letopočty 1900-2099
`\d{2,6}`	sekvence dvou až šesti číslic
`[^ ,.]+`	neprázdná sekvence znaků mezi nimiž nesmí být mezera ( ), čárka (`,`) či tečka (`.`)
`^P.*`	řetězec, který začíná písmenem `P` za nímž následuje libovolný (i nulový) počet libovolných znaků
`\d+0$`	řetězec, který končí znakem `0` (nula), kterému předchází minimálně jedna číslice
`a+b`	`ab`, `aab`, `aaab` atd.
`a\+b`	`a+b`

Autor: [3]

4 Vlastní práce

V následujících kapitolách jsou využity znalosti získané v průběhu studia, zpracováním literární rešerše a v neposlední řadě praxí ze zaměstnání v oborou IT jako systémový administrátor. Zároveň jsou využity znalosti pokročilého skriptování v PowerShell získané v průběhu intenzivního týdenního kurzu „Pokročilá správa systému Windows Server 2019 za pomocí PowerShell“. Podkapitoly jsou seřazeny tak, aby korespondovaly s postupy pro návrh a implementaci programů.

Důležité: Následující analýza Active Directory a návrh programu v PowerShell je prováděná na testovací doméně s názvem „IZS.CZ“, dále také označována jako testovací prostředí. Tato doména byla zkopírována a modifikována pro testovací účely. Veškeré názvy počítačů, serverů, organizačních jednotek a dalších síťových zařízení jsou záměrně přejmenovány, stejně tak je pozměněná doménová struktura. Tyto změny byly provedeny s cílem zachovat anonymitu a zajištění celkové bezpečnosti citlivých informací organizace. Avšak testovací doména zachovává všechny aspekty hierarchie běžné domény, které jsou následně využity pro analýzu AD s následným návrhem programu pro automatizované alternativní nasazení síťových tiskáren pomocí PowerShellu. Reálné prostředí domény bude dále v této práci uváděno jako produkční prostředí.

4.1 Popis řešeného problému

V organizaci IZS se v posledních letech setkává tiskový administrátor s problémy při konvenčním zavádění síťových tiskáren v rámci domény Active Directory. Na různých konfiguracích klientských stanic se systémem Windows 8 až po Windows 10 dochází u některých již nainstalovaných či nově instalovaných klientských operačních systémů k selhání instalace ovladačů síťových tiskáren. Obvykle – dle sdělení jednotlivých oblastních administrátorů a tiskového administrátora – z důvodu odmítnutí důvěry tiskového serveru. Pracovní stanice na základě konfigurace GPO obdrží informaci o požadované instalaci tiskáren a cestu, odkud má stáhnout a nainstalovat ovladače, avšak tato instalace obvykle selže. Vzhledem k tomu, že se tiskový administrátor spolu s administrátory domény snažili celou situaci řešit doporučenou konfigurací dle postupů uvedených na stránkách firmy Microsoft, nepodařilo se jim problém zcela odstranit. Vždy se objeví problém na některých stanicích, kdy tiskový administrátor například přidá do seznamu novou tiskárnu.

Dále po zavedení rozšířené správy GPO pomocí AGPM, který zajišťuje zabezpečení v podobě kontroly změn politik, verzování nastavených politik, zálohování či obnovu politik a další nastavení, došlo k narušení konvenčního zavádění tiskáren. AGPM bylo zavedeno z důvodu zamezení neodborných a neznalých nastaveních doménových politik, které by negativně ovlivňovali chod všech klientských stanic. Zejména by docházelo ke konfliktům mezi různými nastaveními, které by byly jinak funkční. Tímto zavedením došlo k omezení práv pro práci s doménovými politikami u některých administrátorů. Z tohoto důvodu nemůže tiskový administrátor využívat dosavadní zavádění tiskáren, neboť pro editaci GPO musí použít rozšíření AGPM, který sám může pouze politiku editovat a dále pak musí vyčkat na odsouhlasení pověřeného správce domény. Správce tuto politiku zkontroluje a nasadí do produkčního prostředí. Na základě výše uvedených problémů byl autor diplomové práce požádán o pomoc tiskovým administrátorem a vedením oddělení informačních technologií o nalezení vhodného způsobu alternativního zavádění síťových tiskáren. Autor tuto žádost přijal jako výzvu, jak využít jazyk PowerShell pro toto automatizované nasazování tiskáren.

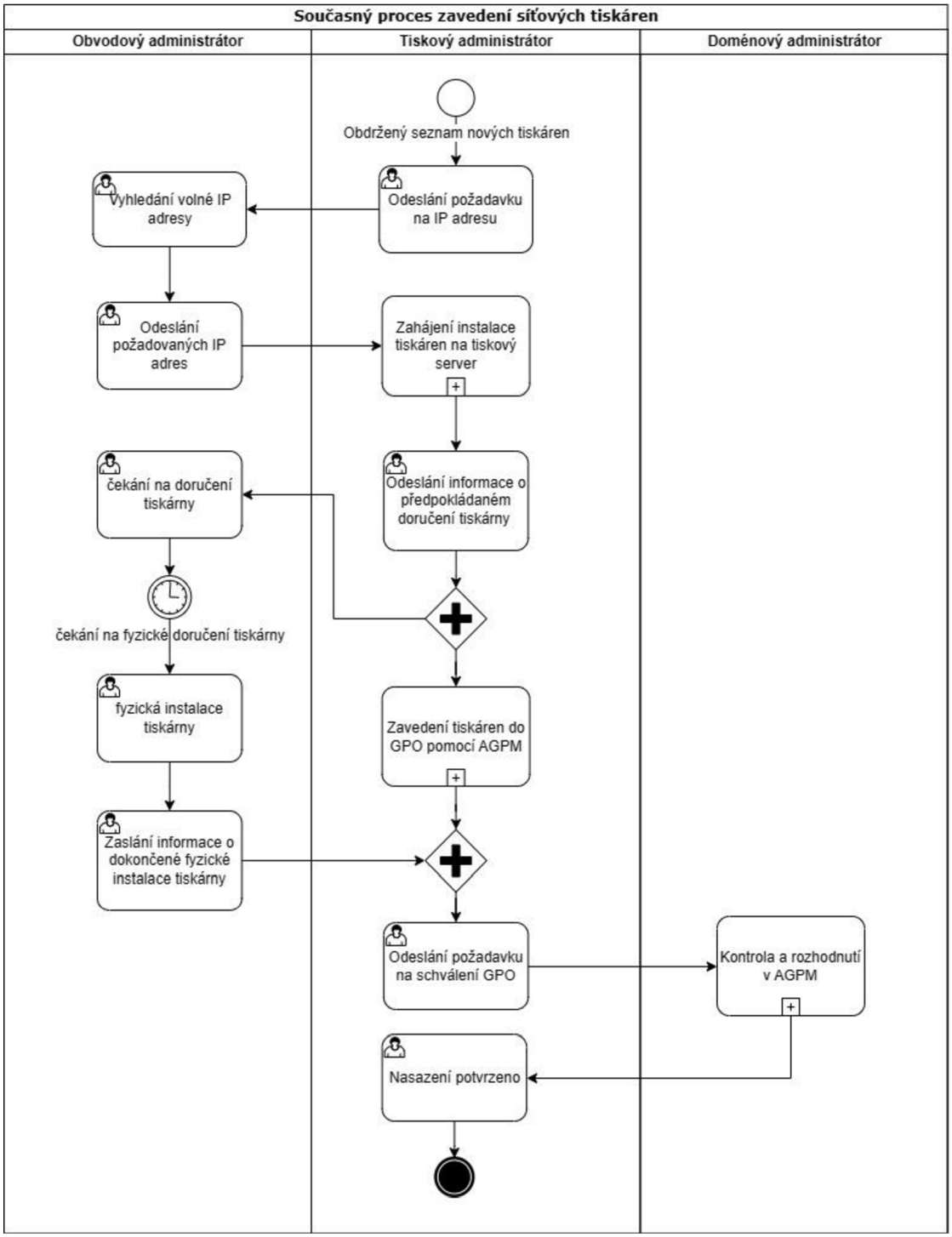
4.2 Analýza současného procesu nasazování síťových tiskáren

Byla provedena analýza současného nasazování síťových tiskáren formou rozhovoru s tiskovým administrátorem. Z rozhovoru vyplynulo, že na tiskovém serveru s operačním systémem Windows 2019 Server jsou nainstalované tiskárny, které mají názvy stejné jako organizační jednotky jednotlivých oddělení a jejich pořadí je rozlišován znakem z abecedy. Jako příklad uvedl tiskárnu pro organizační jednotku O00-000005, která má označení „O00-000005-B“. Další tiskárna by měla pořadí C, D, atd. Celková velikost znaků za názvem tiskárny není nějak stanovena, avšak dodržuje max. 3 znaky. Dále sdělil zavedený postup, jakým způsobem provádí zavádění síťových tiskáren do klientských stanic.

V případě obdržení seznamu obměňovaných či nových tiskáren, provede emailovou komunikaci s oblastními administrátory se žádostí o předběžné přidělení síťových IP adres pro jejich budoucí síťové tiskárny. Po obdržení IP adres od oblastních administrátoru provede instalaci tiskáren na tiskový server (pozn. bez IP adresy není možné tiskárnu na server nainstalovat). Po nainstalování tiskáren vyčká na fyzické dodání tiskáren na oddělení. Tuto skutečnost (datum doručení tiskáren na oddělení) oznámí oblastním administrátorům v emailové zprávě. Po fyzickém doručení a zapojení tiskárny do sítě obdrží tiskový administrátor informaci o dokončené fyzické instalace, přičemž zahájí dokončení zavádění

tiskáren pomocí nástroje AGPM. Po editaci příslušné skupinové politiky musí kontaktovat emailovou zprávou pověřené správce domény se žádostí o zavedení skupinové politiky do produkčního prostředí. Potvrzení aplikování změn na příslušné politice je zasláno zpět tiskovému administrátorovi emailovou zprávou. Tímto proces nasazení síťových tiskáren pro tiskového administrátora končí. Celý tento proces je znázorněn na obrázku č. 7 za pomocí diagramu v BPMN (Business Process Model and Notation). Jedná se o grafickou notaci pro modelování podnikových procesů, která usnadňuje čitelnost a srozumitelnost pro každého, kdo se podílí na životním cyklu procesu.

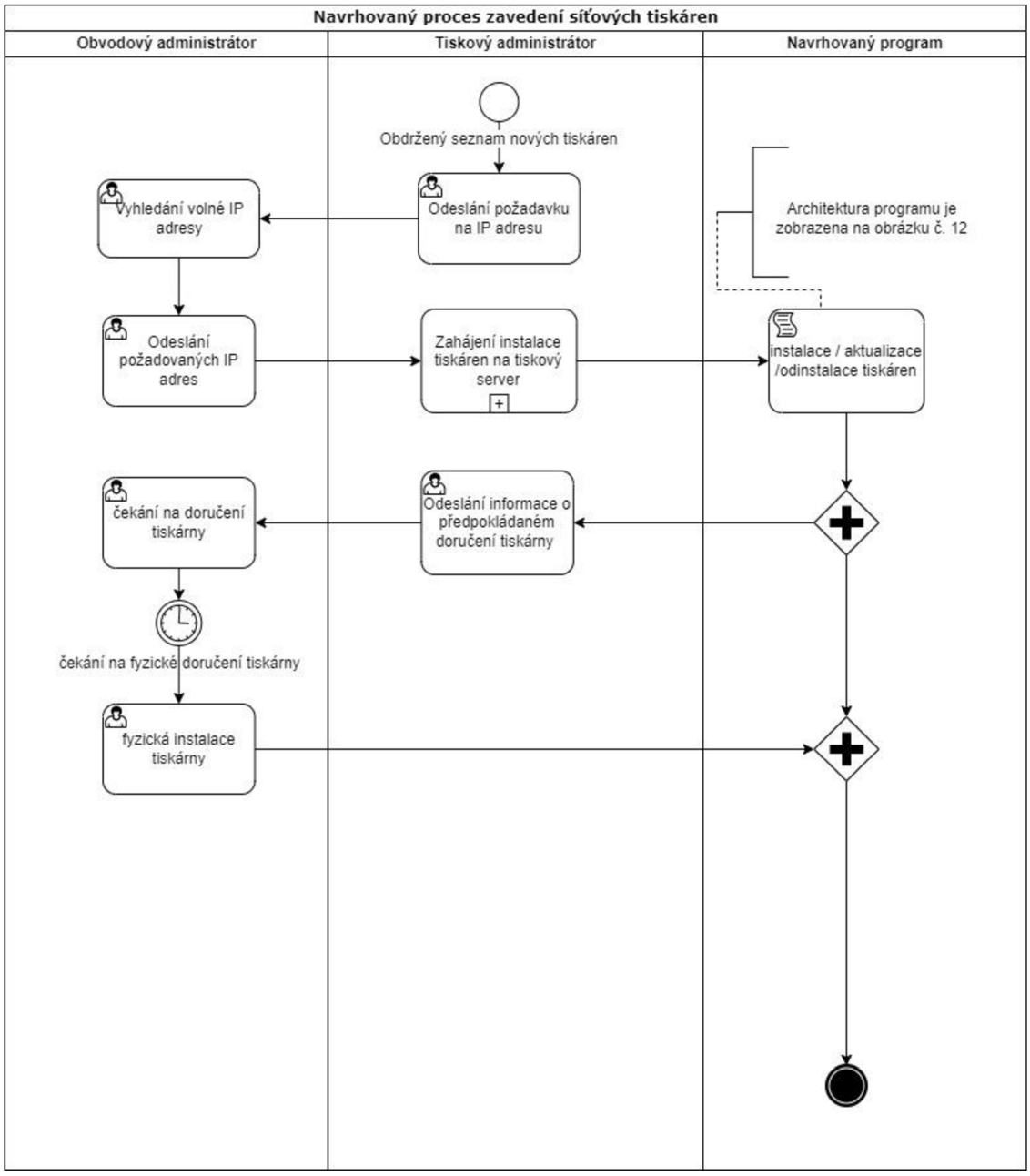
Obrázek 7- Současný proces zavádění síťových tiskáren



Autor: [vlastní zpracování]

V následujícím obrázku č. 8 je znázorněn tentýž proces s návrhem využití programu pro alternativní zavádění síťových tiskáren.

Obrázek 8 - Navrhovaný proces zavádění síťových tiskáren



Autor: [vlastní zpracování]

Z návrhu byly odebrány procesy zavedení tiskáren do GPO pomocí AGPM, odeslání požadavku k nasazení editované skupinové politiky doménovému administrátorovi a zaslání informace o fyzické instalaci síťové tiskárny tiskovému administrátorovi. Díky tomuto

návrhu došlo k zefektivnění zavádění síťových tiskáren – snížení počtu kroků pro jejich zavedení vynecháním nutnosti editování politiky GPO. A dále se předejde zbytečnému čekání na vyřízení schválení editované politiky GPO doménovým administrátorem.

4.3 Požadavky na program

Pro návrh vhodného programu pomocí PowerShellu byla provedena analýza na funkční a nefunkční požadavky formou rozhovoru se zainteresovanými stranami, kterými byli oblastní administrátoři a hlavní tiskový administrátor. Z rozhovoru vyplynuly následující požadavky:

- **Funkční požadavky programu**
  a) Jakýkoliv uživatel si bude moci spustit program dle svého uvážení.
  b) Program musí zaznamenávat průběh stavu instalace ve formátu HTML.
  c) Bude umožněno tiskovému administrátorovi mimo jiné instalovat i tiskárny určené pro jiné oddělení.
  d) Kromě instalací ovladačů bude umět i ovladače aktualizovat a odinstalovat.
  e) Bude přebírat veškeré nastavení předvoleb, vlastností a zabezpečení tiskáren z tiskového serveru.

- **Nefunkční požadavky programu**
  a) Musí být zachovaná aktuální instalace síťových tiskáren na tiskovém serveru.
  b) Musí být spouštěna v pravidelných intervalech. Minimálně při startu operačního systému.
  c) Ve výstupu programu musejí být barevně odlišené úspěšné a neúspěšné operace.
  d) Doba běhu programu při opětovné kontrole by neměla přesáhnout 60 vteřin.
  e) Pro instalaci není nutné restart či odhlášení uživatele ze systému.
  f) Program musí být kompatibilní minimálně od verze Windows 7 a novější.

Tyto funkční a nefunkční požadavky, které byly konzultovány se zainteresovanými stranami, poskytly dostačující základ pro představu návrhu a následného vývoje programu. Dále v rozhovoru uvedl tiskový administrátor, že první verzi programu lze otestovat v malé dodávce na začátku července, kde budou dodány 4 kopírky na 2 oddělení. Větší dodávku

očekává v polovině měsíce srpna. Důležité však je, aby program splnil očekávání, a to bezproblémové instalace síťových tiskáren.

4.3.1 Use Case Diagram a Aktoři

Program bude navržený tak, aby byl jeho výstup čitelný pro každého, kdo s ním přijde do styku. Aktoři, kteří budou s programem interagovat, jsou dva.

- **Uživatel** – Jedná se o běžného uživatele systému, který může a nemusí mít technické znalosti počítače nebo administrátor. Jeho jedinou akcí je spuštění programu pomocí zástupce v nabídce start sytému Windows.
- **Systém** – Spustí program při startu operačního systému.

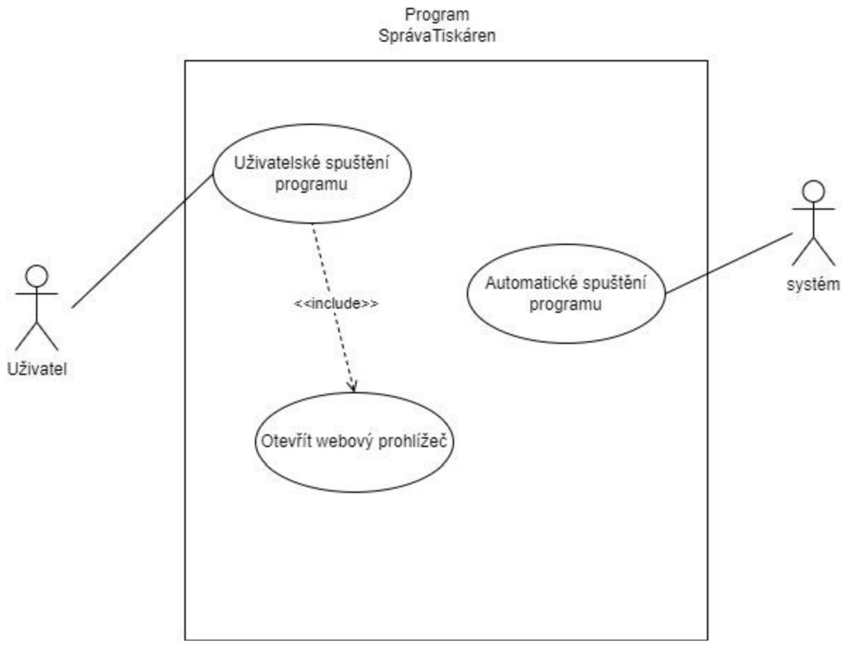
Use Case: Uživatelské spuštění programu
Aktor: Uživatel

- Uživatel otevře nabídku start a vybere zástupce pro aktualizaci tiskáren.
- Systém spustí požadovaný program na pozadí, který provede kontrolu nainstalovaných tiskáren a případně je nainstaluje, aktualizuje či odebere.
- Zároveň po spuštění programu se automaticky spustí webový prohlížeč, který zobrazí html soubor s dynamickým výpisem postupu programu. Jeho obsah se automaticky aktualizuje každých 5 vteřin.

Use Case: Automatické spuštění programu
Aktor: Systém

- Při zapnutí počítače se spustí operační systém.
- Po spuštění operačního systému dojde ke spuštění programu aktualizace tiskáren bez interakce uživatele.
- Program provede na pozadí kontrolu nainstalovaných tiskáren a případně je nainstaluje, aktualizuje či odebere.

Obrázek 9 - Use Case Diagram spuštění programu



Autor: [vlastní zpracování]

4.4 Analýza Active Directory

Pro účely návrhu a implementace programu, který spolupracuje s doménou Active Directory, je nutné provést její analýzu za účelem zjištění její struktury a organizaci sítových zařízení v organizačních jednotkách. Cílem není popsat detailní strukturu celé domény, ale pouze důležité části pro nalezení způsobu identifikace pracovní stanice a její umístění.

Pro tuto analýzu byla využita konzole pro správu uživatelů a počítačů složby Active Directory ve zkratce ADUC (Active Directory Users and Computers), která je dostupná až poté, co je doinstalované RSAT (Remote Server Administrator Tools for Windows). Tento balíček také obsahuje moduly pro PowerShell, se kterými lze spravovat také Active Directory. Po spuštění této konzole je zobrazen kontejner s názvem domény „IZS.CZ“, který po rozkliknutí zobrazí názvy výchozích objektů domény. Mezi výchozí objekty patří například Builtin, v němž se nachází výchozí předdefinované bezpečnostní skupiny domény např. Administrators, Users a dalších 25 skupin. Dále je třeba zmínit výchozí kontejner „Computers“. Do tohoto kontejneru je automaticky vytvořen objekt počítače při jeho prvním připojením do domény, pokud již nebyl vytvořen v jiné organizační jednotce. Pro autora je důležitá nalezená organizační jednotka s názvem „SEKTORY“.

4.4.1 Struktura organizační jednotky „SEKTORY“

V rozbalovacím seznamu organizační jednotky „SEKTORY“ se nachází celkem 5 organizačních jednotek s názvem „OBLAST00, OBLAST11 až OBLAST14“. Tyto jednotky byly identifikovány jako hlavní jednotky oblastí, které spravují jednotlivé týmy administrátorů. Při bližším zkoumání jednotky „OBLAST00“ bylo zjištěno, že obsahuje organizační jednotky s názvy Admins, Computers, Servers, ServiceAccounts, services, Users. V Admins byly nalezeny doménové administrátorské účty, jejichž názvy začínali prefixem „sad_“. Běžné uživatelské účty se nacházejí v organizační jednotce Users a nemají žádný prefix. Při průzkumu organizační jednotky Computers byly nalezeny další organizační jednotky identifikovány jako jednotlivé oddělení organizace. Tato oddělení mají prefix „O00-“, což je zkratka názvu OBLAST00. Následuje šestimístné číselné označení. Výjimku tvoří poslední dvoučíslí, které obsahuje mimo čísla i písmena. Celý název takového oddělení v organizační jednotce je například „O00-000005“ nebo „O00-0000IT“. Některé oddělení mají ještě na konci přidané dvou až čtyřmístné rozlišení v podobě znaků písmen či čísel. Obsahem organizační jednotky oddělení jsou již objekty počítačů, které mají obdobné pojmenování. Také začínají stejným prefixem, jako organizační jednotky oddělení a se čtyřmístnou číselnou řadou. Příklad takového názvu počítače je „O00-1234“. V organizační jednotce „Servers“ jsou také objekty počítačů, tedy serverů, které mají pouze shodný prefix jako počítače na odděleních. Začínají tedy „O00-“ a dále již pokračují různými názvy, ze kterých by se dal odhadnout jejich účel. Například server s názvem O00-AXIS by mohl být server obsluhující software na správu bezpečnostních kamer.

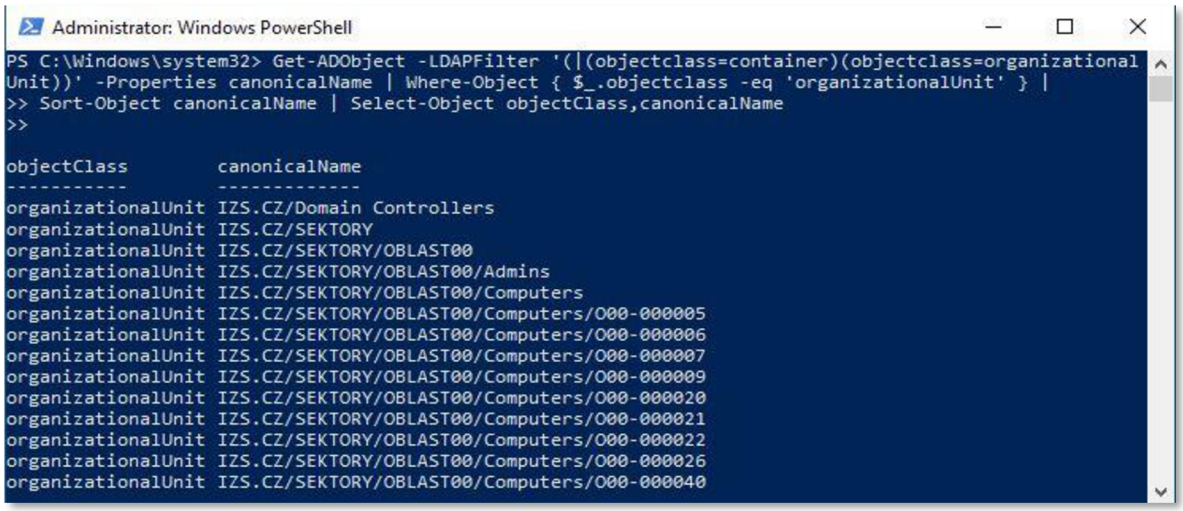
Při dalším zkoumání ostatních organizačních jednotek OBLAST11 až OBLAST14 bylo zjištěno, že jejich struktura odpovídá stejné struktuře jako ve výše zmíněné organizační jednotce OBLAST00. Oddělení a počítačové objekty taktéž začínají prefixem například „O11-“. Ten je odvozen z OBLAST11 a dále má stejné pokračování.

Celou strukturu organizačních jednotek by bylo možné vygenerovat do textového formátu, například jako tabulku, a to následujícím příkazem za pomocí PowerShellu.

```
Get-ADObject -LDAPFilter
'(|(objectclass=container)(objectclass=organizationalUnit))' -Properti
es canonicalName | Where-Object { $_.objectclass -eq
'organizationalUnit' } |
Sort-Object canonicalName | Select-Object objectClass,canonicalName
```

Zkrácený výsledek příkazu lze vidět na obrázku č. 10.

Obrázek 10 - Zkrácený stromový výpis organizačních jednotek domény IZS.CZ



Autor: [vlastní zpracování]

4.4.2 Analýza aktivních počítačových stanic v doméně Active Directory

Cílem této podkapitoly je poskytnout detailní přehled o používaných počítačových stanicích v doméně „IZS.CZ“, které mohou být případnými cíli pro instalaci síťových tiskáren. Aby byla zajištěna maximální kompatibilita navrženého programu, jež byla specifikována v kapitole 4.3, je nezbytné identifikovat všechny stávající aktivní počítačové stanice a zhodnotit jejich specifikace.

1. Postup při výběru počítačových stanic

Pro účely této analýzy byl proveden dotaz v Powershellu, jehož cílem je získání počtu všech aktivních počítačových stanic v doméně s jejich operačním systémem. Servery byly z tohoto dotazu vyřazeny, neboť navržený program nemá za cíl instalaci tiskáren na serverech. Příkaz pro výpis počítačů seskupený podle operačního systému:

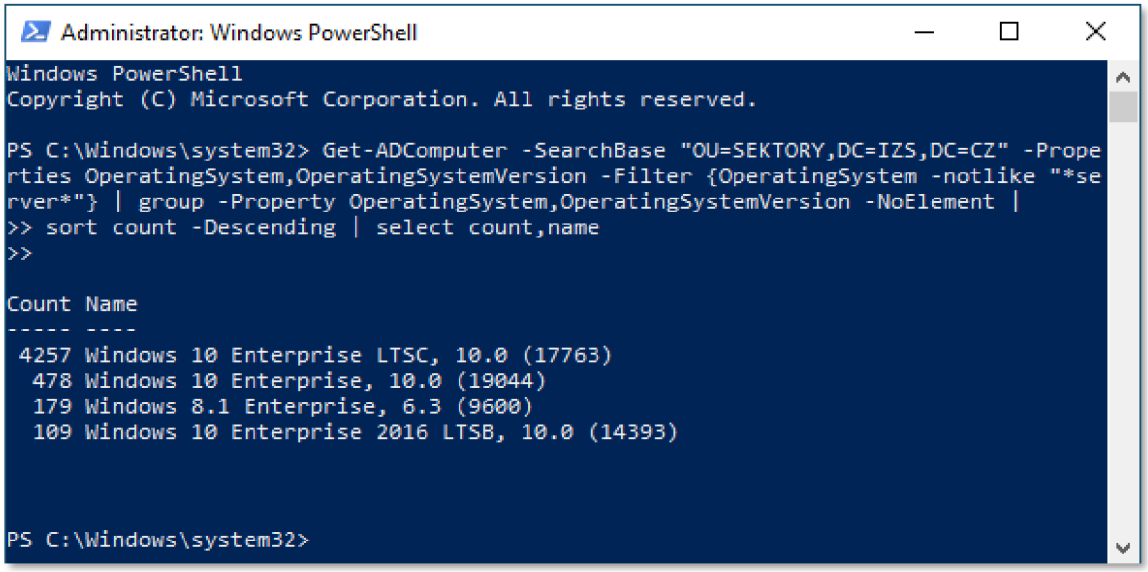
```
Get-ADComputer -SearchBase "OU=SEKTORY,DC=IZS,DC=CZ" -Properties
OperatingSystem,OperatingSystemVersion -Filter
{OperatingSystem -notlike "*server*"} | group -Property
OperatingSystem,OperatingSystemVersion -NoElement |
sort count -Descending | select count,name
```

2. Výsledky dotazu

Výsledek je zobrazen na obrázku č. 11, ze kterého je zjištěno celkem 5023 aktivních počítačových stanic. Z nich:

- 4257 stanic používá operační systém Windows 10 Enterprise verze LTSC sestavení 17763.
- 478 stanic funguje na operačním systému Windows 10 Enterprise verze 21H2 sestavení 19044.
- 179 stanic je vybaveno operačním systémem Windows 8.1 Enterprise sestavení 9600.
- 109 stanic využívá operační systém Windows 10 Enterprise 2016 LTSB sestavení 14393.

Obrázek 11 - Výpis aktivních pracovních stanic



Autor: [vlastní zpracování]

Z výsledku je tedy patrné, že se v doméně nenachází žádný operační systém, který by nepodporoval jazyk PowerShell. A proto se může předpokládat, že navrhovaný program bude aplikovatelný na všechny počítačové stanice.

4.5 Analýza doménových zásad skupin GPO

V této kapitole je popisována hierarchie struktury GPO za účelem zjištění dosavadního nasazování síťových tiskáren pomocí doménových politik. Hlavním účelem je zjistit možné komplikace omezující spuštění programu PowerShell, případně jaké úpravy budou zapotřebí provést pro nasazení navrhovaného programu.

K průzkumu doménové struktury správy zásad skupin GPO byla využita konzole GPMC (Group Policy Management Console). Po spuštění této konzole se zobrazí podobná stromová struktura domény, jako bylo popsáno v kapitole 4.4. Po rozkliknutí prvního objektu s názvem „Doménová struktura: IZS.CZ“ následuje objekt „Domény“ a pod tímto objektem je již samotná doména „IZS.CZ“. První spatřená doménová zásada, je výchozí doménová politika (Default Domain Policy). Při analyzování jejího obsahu bylo zjištěno, že tato politika nastavuje výchozí chování systému, například nastavení zabezpečení počítače v oblasti zabezpečení účtu např. zásady hesel či počet neplatných pokusů do uzamčení účtu atd. Jak již bylo zmíněno v kapitole 4.1, v doméně je využívána rozšířená správa zásad skupin pomocí AGPM. V této testovací doméně nebylo rozšíření použito. V případě aktivního rozšíření se ve stromové struktuře zobrazí na posledním místě nový objekt rozšíření pod názvem „Change Control“, kde se nacházejí všechny přidané spravované doménové zásady. Tyto zásady se nacházejí v záložce „Controlled“. Z důvodu zjištění možného ovlivnění chování nastavení politik v organizační jednotce „SEKTORY“ a její podřízených organizačních jednotkách, bylo přistoupeno rovnou do těchto jednotek.

4.5.1 Struktura a konvence pojmenování politik

Po kompletním rozbalení organizačních jednotek od OBLAST00 až OBLAST14 bylo zjištěno, že každá organizační jednotka s názvem „Computers“ má přiřazeno celkem 10 doménových politik. Při tomto je hned patrná jmenná konvence politik, která ihned naznačuje, jak a k čemu se přiřazená politika používá. Všechny politiky mají jmennou konvenci „O00-C-CENTRAL“, která za posledním oddělovačem „-“ má název jejího zaměření. Písmeno „C“ značí, že se jedná o politiku, nastavující chování systému na úrovni celého počítače. Slovo „CENTRAL“ je pak informace, že se jedná o centrální politiku, která je používaná napříč organizačními jednotkami jednotlivých oblastí. Posledním názvem je upřesnění konkrétního nastavení např. Certificates, ConfigAccountLocal, Firewall atd. Celý název politiky je tedy např. „O00-C-CENTRAL-Certificates“.

V organizační jednotce „Users“, která je součástí každé nadřazené jednotky oblasti, se nachází 6 centrálních politik a 3 politiky určené pouze pro dané lokality. Tyto politiky – na rozdíl od předchozích centrálních politik – mají písmeno „U“, které značí nastavení určená pro uživatele. Zaměření politik mají názvy např. ControlPanel-Admin, ControlPanel-User, DesktopIcons či MSEdge. Celý název politiky je např. „O00-U-DesktopIcons“.

Organizační jednotky oddělení, které jsou umístěné v podřízené jednotce „Computers“ v každé oblasti, mají přiřazenou vždy jednu politiku. Ta má název např. „O00-C-000006“, což je označení prefixu „O00“ až „O14“ dle oblasti. Písmenko „C“ je již známé pro použití na úrovni celého počítače, číselný šestimístný řetězec je kód oddělení. Z politiky je tak patrné, k jaké organizační jednotce patří. Také dle pojmenování je ihned jasné, na které politiky se lze zaměřit v případě vyskytnutého problému.

4.5.2 Analýza konkrétních doménových politik

Pro prověření, zdali politiky budou či nebudou ovlivňovat funkčnost programu, je zapotřebí provést jejich kontrolu a případně provést nutnou úpravu. Pro kontrolu byly vybrány politiky s názvem „O00-C-CENTRAL-PowerShell“ a „O00-C-CENTRAL-Printer“ a politika oddělení „O00-C-00005“. Při kontrole v produkčním prostředí domény je politika „O00-C-CENTRAL-PowerShell“ nastavená na „Execution Policy – Allow only signed scripts“. Z důvodu chybějící certifikační autority v testovací doméně byla tato politika změněná na „Execution Policy – Allow all scripts“. Jinak by při každé editaci zdrojového kódu a testování muselo být pokaždé provedeno podepsání programu, jak bylo zmíněno v kapitole 3.1.3 v bodě 4.

V následující politice „O00-C-CENTRAL-Printer“ bylo zjištěno nastavení, které by mělo umožňovat instalaci tiskáren z tiskového serveru „O00-PRINT01“. Dle výpisu nastavení se jedná o hodnoty, které jsou vkládány přímo do registru. Jedná se o tato konkrétní nastavení klíče registru

„*HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\Printers\PointAndPrint*“

První vložený registr je s názvem hodnoty „*NoWarningNoElevationOnInstall*“, druhý „*RestrictDriverInstallationToAdministrators*“, třetí je „*UpdatePromtSettings*“. Všechny registry mají nastavenou datovou hodnotu 0.

Dalším vloženým klíčem registru je

„*HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\Printers\PackagePointAndPrint*“

Do klíče je vložena hodnota s názvem „PackagePointAndPrintServerList“ s datovou hodnotou 1. Posledním vloženým registrem je podklíč „ListOfServers, zde je vložena hodnota s názvem serveru „O00-PRINT01.izs.cz“ a s datovou hodnotou se shodně vloženým názvem.

Poslední prověřovanou politikou je politika oddělení „O00-C-000006“, která má pouze nastavené hodnoty pro připojení 3 tiskáren. Hodnoty jsou uvedené jako síťové cesty

\\o00-print01\O00-000005-MyQ
\\o00-print01\O00-000005-F
\\o00-print01\O00-000005-B

Na základě těchto informací bylo konstatováno, že spouštění programu v PowerShell, po podepsání podpisem vygenerovaným certifikační autoritou produkční domény, nebude problém. Dále je navrhováno – z důvodu nasazení programu pro instalaci síťových tiskáren do klientských stanic – zcela odebrat politiky „O00-C-CENTRAL-Printer“. A také všechny politiky na jednotlivých odděleních, které definují připojení tiskáren. Tyto doménové politiky již nebudou potřeba, byly by v konfliktu s programem.

4.6 Návrh a implementace programu

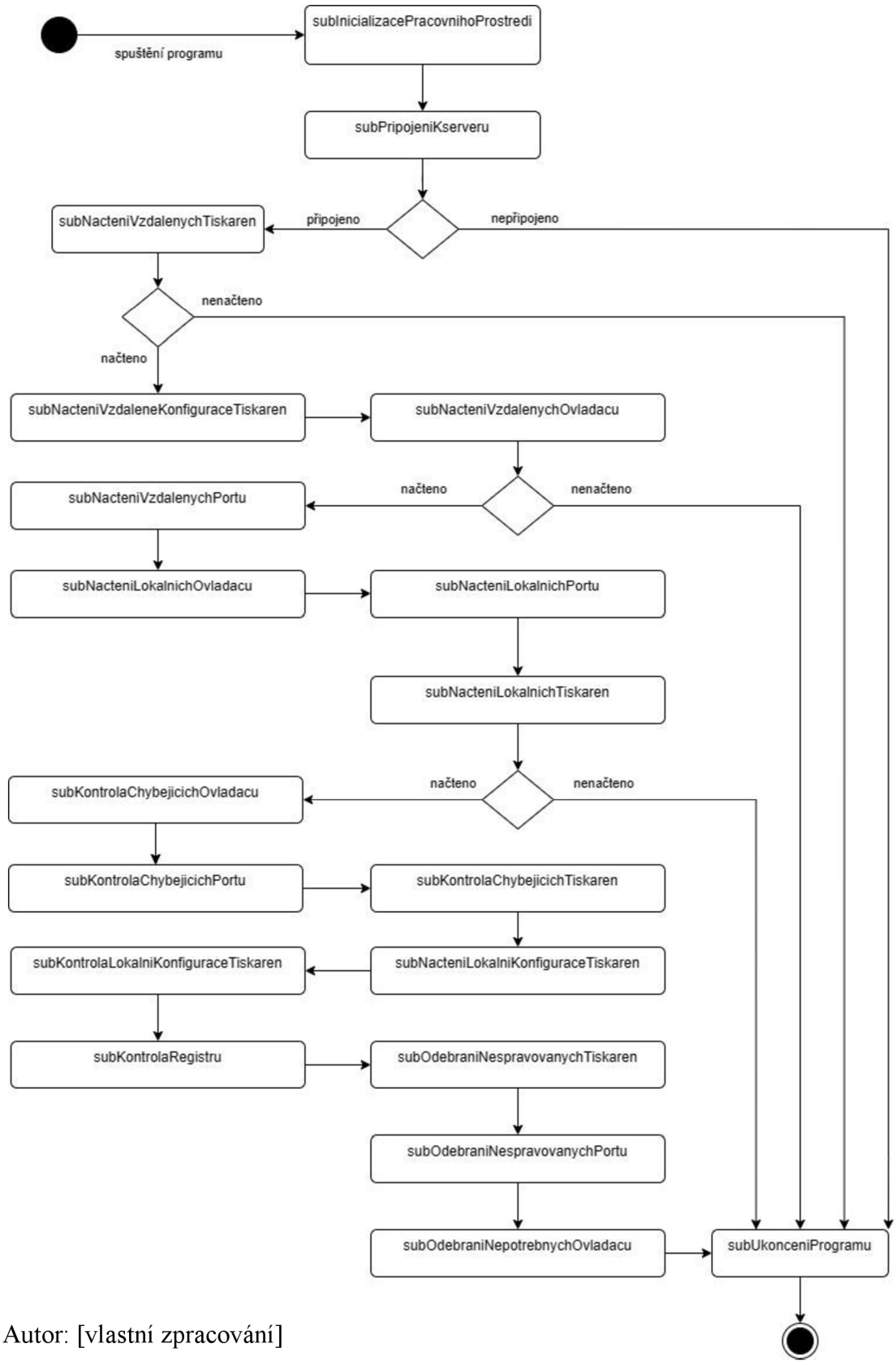
Na základě provedených analýz a provedeného sběru funkčních a nefunkčních požadavků je využit PowerShell, který umožňuje mimo jiné automatizovat zdlouhavé či náročné opakující se administrátorské procesy. Při navrhování programu je také kladen důraz na skutečnost, aby při spuštění programu co nejméně vyrušoval uživatele pracující na počítačích. Například aby nebylo nutné v případě problému restartovat počítač.

Následující popis programu a jeho procedur bude z důvodu obsáhlosti zdrojového kódu stručně popisován, detailněji budou popisovány pouze důležité funkce či procedury. **Celý zdrojový kód je součásti diplomové práce jako Příloha 5 – Zdrojový kód: Správa tiskáren**. Při psaní zdrojového kódu byly stanoveny názvy proměnných a funkcí ve stylu "camelCase – malá a velká počáteční písmena spojených slov". Pro rozlišení mezi bloky kódu, které nevracejí žádné hodnoty, ale pouze vykonávají nějakou sekvenci rutin nebo jiné operace, byl pro lepší orientaci a efektivnější využití našeptávače "IntelliSense" zaveden prefix "sub", například "subPripojeniKserveru". Pro funkce vracející hodnoty či objekty byl stanoven prefix "fnc", například "fncInstalaceOvladace". Tato metoda pojmenování umožňuje při psaní kódu v této práci rychlejší vyhledávání požadovaných funkcí či procedur.

4.6.1 Struktura navrhovaného programu

Navrhovaný program je členěn do několika fází, ve kterých se volají procedury. Ty vykonávají operace s rutinami pro správu tiskáren a ovladačů. Jedná se o základní procedury, které jsou programem volány. Uvnitř těchto procedur dochází k volání jednotlivých funkcí, nad kterými jsou prováděny podmíněné operace. Celkem se jedná o 19 základních procedur. Vzhledem k tomu, že program vzdáleně komunikuje s tiskovým serverem a jednotlivé pracovní stanice mají různorodé konfigurace a datové připojení, které může být v některých lokalitách nestabilní, bylo rozhodnuto pro hromadném načtení vzdálených tiskáren do skriptových proměnných. Z těchto proměnných pak bude program čerpat potřebná data. Na obrázku č. 12 je pro přehlednost zobrazen aktivity diagram nejvyšší úrovně programu.

Obrázek 12 - Aktivity diagram nejvyšší úrovně programu



Autor: [vlastní zpracování]

4.6.2 Popis procedur programu a jeho průběh

I. **Inicializace** – v této fázi jsou volány 2 procedury.

a) **subInicializacePracovnihoProstredi**

Procedura inicializuje skriptové proměnné, které se používají v rámci celé instanci programu. V rámci této procedury je také volaná funkce „fncVratitNazevOU“, do které je předán parametr nazevPC s názvem počítače. Pod ním aktuálně program běží. Volaná funkce vytvoří konstruktor třídy DirectorySearcher, který je součástí platformy .NET Framework. Tato třída umí vyhledat objekty v Active Directory. V tomto případě je hledán objekt počítače a jeho atribut „DistinguishedName“. Z něho je vrácen název organizační jednotky. Tato třída byla použitá, neboť výchozí instalace klientských stanic nemají přístup k rutinám PowerShellu, který umí komunikovat s Active Directory.

b) **subPripojeniKserveru**

Je procedura, která vytváří jak vzdálenou relaci (angl. „Remote Session“) s tiskovým serverem, tak i místní relaci s tiskovou službou. Tyto relace jsou uloženy do pracovních proměnných $script:CimSessionVzdaleny a $script:CimSessionLokalni. Ty jsou využívány při každém volání rutin pro práci s tiskárnami. Jelikož je procedura pro chod programu důležitá, tak v případě neúspěchu při spojení dojde k zaznamenání do výpisu a k ukončení programu viz zdrojový kód 1:

Zdrojový kód 1 - Procedura subPripojeniKserveru

```
function subPripojeniKserveru {
    Param (
        $nazevServeru = $script:tiskovyServer,
        $protokol = "Dcom"
    )
    subZapsatLogDoTxt "start subPripojeniKserveru"
    $CimOption = New-CimSessionOption -Protocol $protokol
    try {
        subNapsatZpravuDoKonzole -sloupec1 'Připojování na tiskový server:' -
sloupec2 $nazevServeru -zarovnatNaPredchoziSloupce:$true
        if ($script:logujOperace) { subObsahKonzoleDoHtml }
        $script:CimSessionVzdaleny = New-CimSession -ComputerName
$nazevServeru -SessionOption $CimOption -ErrorAction Stop  #
        $script:CimSessionLokalni = New-CimSession -ComputerName
$env:computername -SessionOption $CimOption -ErrorAction Stop
        subNapsatZpravuDoKonzole -sloupec3 "připojeno" -nahradSloupec sloupec3
-barva Green
    }
    catch {
        subNapsatZpravuDoKonzole -stavZpravy chyba -nahradSloupec sloupec3
        subZapsatLogDoTxt "      PripojeniKserveru - chyba:
$(fncVratitDuvodChyby -psitem $psitem)"
        subUkonceniProgramu
    }
    if ($script:logujOperace) { subObsahKonzoleDoHtml }
    subZapsatLogDoTxt " exit subPripojeniKserveru"
}
```

II. Vzdálené načítání

Objekty tiskáren z pohledu systému jsou složeny z několika komponent. Tyto komponenty jsou ovladače, porty, vlastnosti a předvolby. Proto jsou uvedené části zvlášť načítány pomocí následujících 4 procedur.

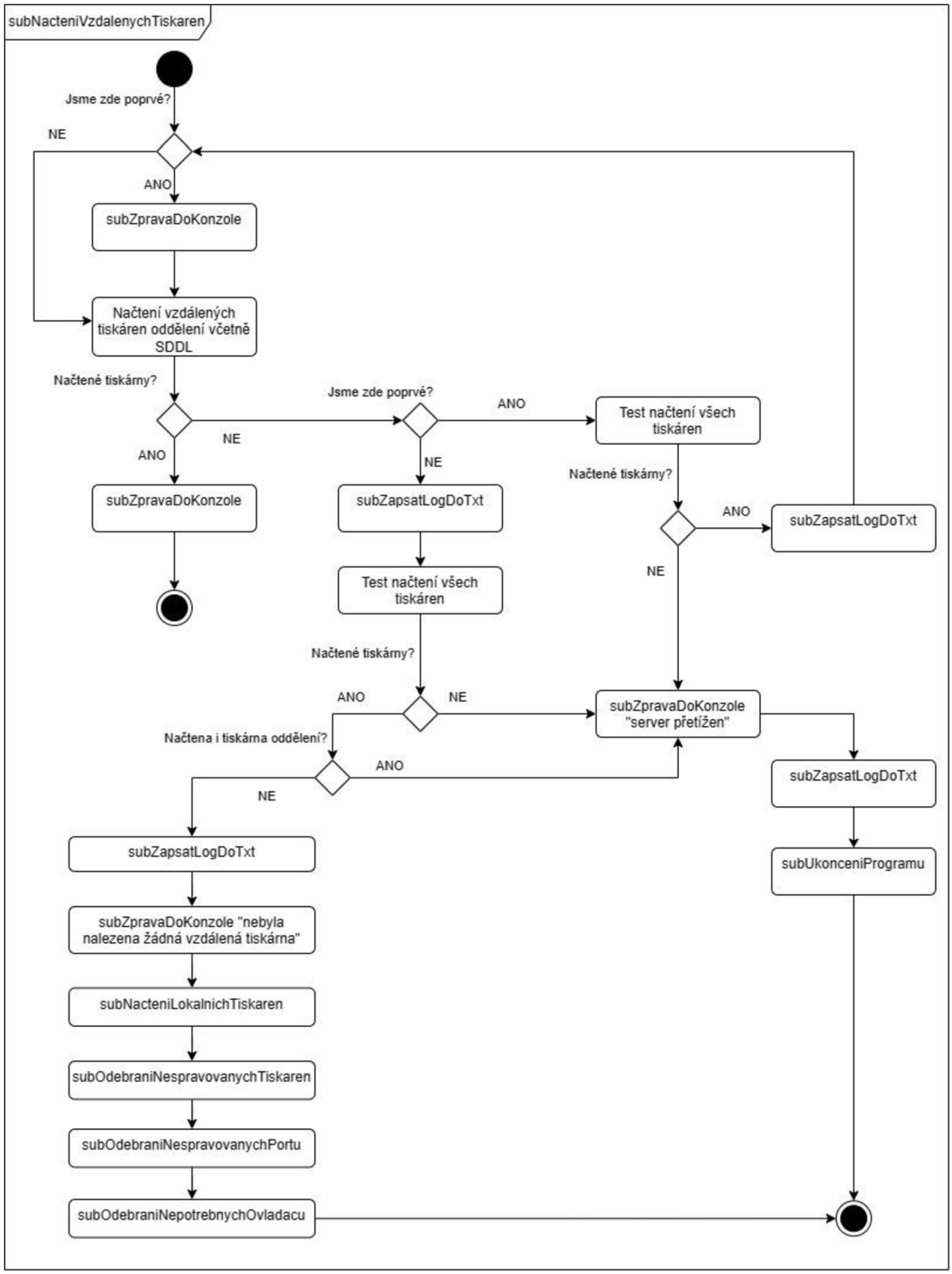
a) subNacteniVzdalenychTiskaren

Jedná se o komponentu vlastností tiskáren. Procedura na základě nalezeného názvu organizační jednotky z funkce fncVratitNazevOU načte všechny potřebné tiskárny, které jsou uloženy do proměnné. Ta má platnost v celé instanci. Pokud by nedošlo k načtení tiskáren, dojde k ukončení programu. Na obrázku č. 13 je detailní náhled procedury pomocí aktivity diagramu a zkrácená ukázka zdrojového kód viz zdrojový kód 2.

Zdrojový kód 2 - Zkrácená ukázka procedury subNacteniVzdalenychTiskaren

```
function subNacteniVzdalenychTiskaren {
    Param(
        $opakovani = $false, #slouží pro opakovaný dotaz, v případě, že
server vrátil prázdný seznam - např. server zaneprázdněn,
        #tiskárny nenalezeny pro oddělení
        $kodOddeleni = $script:oddeleni
    )
    try {
        $vzorRegEx = fncPrevodNaRegEx -hodnota  $kodOddeleni -poziceVzoru
'na začátku'
        subZapsatLogDoTxt "start subNacteniVzdalenychTiskaren"
        if ($opakovani -eq $false) {
            subNapsatZpravuDoKonzole -sloupec1 'Načítám vzdálené tiskárny:'
-sloupec2 $kodOddeleni -zarovnatNaPredchoziSloupce:$true
            subNapsatZpravuDoKonzole -stavZpravy čekejte -nahradSloupec
sloupec3
            if ($logujOperace) { subObsahKonzoleDoHtml }
        }
        if ($nastavovatSecurity) {
            $script:tiskarnyVzdalene = { Get-Printer -CimSession
$script:CimSessionVzdaleny -Full -ErrorAction Stop | ? { $_.name -match
$vzorRegEx } |
                select name, DriverName, PortName, PermissionSDDL,
Location, Comment, PrintProcessor, Datatype, SeparatorPageFile }.Invoke()
        }
        else {
            $script:tiskarnyVzdalene = { Get-Printer -CimSession
$script:CimSessionVzdaleny -ErrorAction Stop | ? { $_.name -match
$vzorRegEx } |
                select name, DriverName, PortName, Location, Comment,
PrintProcessor, Datatype, SeparatorPageFile }.Invoke()
        }
        if ($script:tiskarnyVzdalene.count -eq 0) {
            if ($opakovani -eq $false) {
                $testNacteni = (Get-Printer -CimSession
$script:CimSessionVzdaleny -ErrorAction Stop | measure).count # pokud je 0
je server přetížen,
                # pokud více než 1, pak opakujeme načtení
                #pokud $opakování je $false jedná se o první průchod a
testujeme, jestli server vrací nějaké tiskárny,
                                . . .
```

Obrázek 13 - Dekompozice procedury subNacteniVzdalenychTiskaren



Autor: [vlastní zpracování]

b) subNacteniVzdaleneKonfiguraceTiskaren

Tato procedura provede načtení konfigurace požadovaných tiskáren z tiskového serveru do skriptové proměnné $script:tiskarnyVzdalenePredvolby. V kontextu konfigurace se jedná o komponentu předvolby tisku, ve které se nastavuje například oboustranný, barevný nebo černobílý tisk a další možnosti. Pro první načtení konfigurace není pro správný chod programu až tak důležitá, protože není v případě výskytu chyby program ukončen. Je to z důvodu, že každá instalace tiskárny má výchozí konfiguraci převzatou z ovladače. Tedy v případě nespecifikované chyby v aktuální běhu programu bude konfigurace načtena v dalším spuštěním programu viz zdrojový kód 3.

Zdrojový kód 3 - Procedura subNacteniVzdaleneKonfiguraceTiskaren

```
function subNacteniVzdaleneKonfiguraceTiskaren {
    try {
        subZapsatLogDoTxt "start subNacteniVzdaleneKonfiguraceTiskaren"
        subNapsatZpravuDoKonzole -sloupec1 'Načítám konfiguraci
vzdálených tiskáren'
        subNapsatZpravuDoKonzole -stavZpravy čekejte -nahradSloupec
sloupec2
        if ($logujOperace) { subObsahKonzoleDoHtml }
        [boolean]$script:chybaNacteniVzdaleneKonfigurace = $false
        subZapsatLogDoTxt "      subNacteniVzdaleneKonfiguraceTiskaren -
start get-printer"
        $tiskarnyVzdalenePredvolbyTMP = $script:tiskarnyVzdalene | select
name
        $script:tiskarnyVzdalenePredvolby = @()
        $tiskarnyVzdalenePredvolbyTMP | % {
            subZapsatLogDoTxt "
subNacteniVzdaleneKonfiguraceTiskaren -       $($_.name)"
            $script:tiskarnyVzdalenePredvolby += Get-PrintConfiguration
$_.name -CimSession $script:CimSessionVzdaleny
        }
        subZapsatLogDoTxt "      subNacteniVzdaleneKonfiguraceTiskaren -
exit  get-printer"
        if ($tiskarnyVzdalenePredvolby.count -eq 0) {
            subNapsatZpravuDoKonzole -stavZpravy nenalezena -
nahradSloupec sloupec2
            $script:chybaNacteniVzdaleneKonfigurace = $true
        }
        else {
            subNapsatZpravuDoKonzole -stavZpravy ok -nahradSloupec
sloupec2
        }
    }
    catch {
        $script:chybaNacteniVzdaleneKonfigurace = $true
        subNapsatZpravuDoKonzole -stavZpravy chyba -nahradSloupec
sloupec2
        subZapsatLogDoTxt " exit subNacteniVzdaleneKonfiguraceTiskaren -
chyba: $(fncVratitDuvodChyby -psitem $psitem)"
    }
    if ($logujOperace) { subObsahKonzoleDoHtml }
    subZapsatLogDoTxt " exit subNacteniVzdaleneKonfiguraceTiskaren"
}
```

c) subNacteniVzdalenychOvladacu

Další důležitá procedura, jak již napovídá název, zajišťuje načtení komponenty ovladačů vzdálených tiskáren. V případě neúspěchu načtení nemůže dále program pokračovat, a proto bude o tomto neúspěchu informace zapsána a následně program ukončen. Ukázka kódu viz zdrojový kód 4

Zdrojový kód 4 - Procedura subNacteniVzdalenychOvladacu

```
function subNacteniVzdalenychOvladacu {
    try {
        subZapsatLogDoTxt "start subNacteniVzdalenychOvladacu"
        subNapsatZpravuDoKonzole -sloupec1 'Načítám vzdálené ovladače'
        subNapsatZpravuDoKonzole -stavZpravy čekejte -nahradSloupec
sloupec2
        if ($logujOperace) { subObsahKonzoleDoHtml }
        $osArchitektura = fncVratitBitovouVerziOS
$script:ovladaceVzdalene = { Get-PrinterDriver -CimSession
$script:CimSessionVzdaleny -ErrorAction Stop |
            ? { $_.name -notmatch $ignorovatTiskoveOvladace -and
$_.PrinterEnvironment -match $osArchitektura } |
            select Name, InfPath, PrinterEnvironment, DriverVersion
}.Invoke()
        if ($script:ovladaceVzdalene.count -eq 0) {
            $script:ovladaceVzdalene = @()
            throw "nebyly nalezeny žádné ovladače"
        }
        else {
            subNapsatZpravuDoKonzole -stavZpravy ok -nahradSloupec
sloupec2
        }
    }
    catch {
        subNapsatZpravuDoKonzole -stavZpravy chyba -nahradSloupec sloupec2
        subZapsatLogDoTxt " exit subNacteniVzdalenychOvladacu - chyba:
$(fncVratitDuvodChyby -psitem $psitem)"
        subUkonceniProgramu
    }
    subZapsatLogDoTxt " exit subNacteniVzdalenychOvladacu"
    if ($logujOperace) { subObsahKonzoleDoHtml }
}
```

d) subNacteniVzdalenychPortu

Poslední komponentou je načítání vzdálených portů. Tato komponenta nese informaci o síťové adrese IP tiskárny viz zdrojový kód 5.

Zdrojový kód 5 - Procedura subNacteniVzdalenychPortu

```
function subNacteniVzdalenychPortu {
    try {
        subZapsatLogDoTxt "start subNacteniVzdalenychPortu"
        subNapsatZpravuDoKonzole -sloupec1 "Načítám vzdálené porty"
        subNapsatZpravuDoKonzole -stavZpravy čekejte -nahradSloupec
sloupec2
        if ($logujOperace) { subObsahKonzoleDoHtml }
        $porty = $script:tiskarnyVzdalene | % { $_.portname }
        $script:portyVzdalene = { Get-PrinterPort -CimSession
$script:CimSessionVzdaleny -ErrorAction Stop | ? { $porty -match $_.name }
|
            select Name, PrinterHostAddress, PortNumber, Protocol,
SNMPCommunity, SNMPEnabled, SNMPIndex, LprByteCounting, LprQueueName
}.Invoke()
        if ($portyVzdalene.count -eq 0) {
            $script:portyVzdalene = @()
            throw "nebyly nalezeny žádné porty"
        }
        else {
            subNapsatZpravuDoKonzole -stavZpravy ok -nahradSloupec
sloupec2
        }
    }
    catch {
        subNapsatZpravuDoKonzole -stavZpravy chyba -nahradSloupec sloupec2
        subZapsatLogDoTxt "start subNacteniVzdalenychPortu - chyba:
$(fncVratitDuvodChyby -psitem $psitem)"
    }
    subZapsatLogDoTxt " exit subNacteniVzdalenychPortu"
    if ($logujOperace) { subObsahKonzoleDoHtml }
}
```

III. Lokální načítání

Tato fáze je shodná s předchozí fází vzdáleného načítání. S tím rozdílem, že jsou volány pouze 3 procedury pro načtení lokálních komponent tiskáren. Z tohoto důvodu nebudou procedury rozepisovány.

a) **subNacteniLokalnichOvladacu**

b) **subNacteniLokalnichPortu**

c) **subNacteniLokalnichTiskaren**

IV. Kontrola a synchronizace

V této fázi jsou porovnávány vzdálené a lokální komponenty mezi sebou, přičemž lokální části jsou aktualizovány na základě vzdálených komponent.

a) **subKontrolaChybejicichOvladacu**

Provede kontrolu chybějících nebo rozdílných ovladačů a v případě potřeby je doinstaluje nebo aktualizuje. Procedura iteruje nad nalezenými vzdálenými tiskárnami a porovnává jejich verze ovladačů s verzemi ovladačů lokální tiskové

službě – viz ukázka zdrojového kódu 7. Zde můžou nastat 3 stavy, na základě kterých se provedou různé operace:

- **Neexistuje** – Provede volání funkce „fncInstalaceOvladace“ – viz zdrojový kód 6. Té se předá parametr „-nazevSouboruCAB“ s hodnotou systémového identifikátoru ovladače. Následně provede jeho stažení do pracovní stanice a nainstalování do repositáře operačního systému. Pozn.: Pro instalaci ovladače do operačního systému není k dispozici rutina PowerShell. Je však použit alternativní způsob instalace pomocí externí aplikace „PNPUTIL.EXE“, který pouze přidá ovladač do systémového repositáře ovladačů. Následně, aby mohl být ovladač použit pro instalaci tiskárny, je potřeba ho přidat pomocí rutiny „Add-PrinterDriver“ do tiskové služby.
- **Rozdílná verze** – Protože byla nalezena rozdílná verze ovladače v tiskové službě, provede se volání funkce „fncInstalaceOvladace“ se stejným parametrem jako ve stavu „neexistuje“. Tím se ovladač aktualizuje v systémovém repositáři. Následně dojde k přidání ovladače do tiskové služby.
- **Shodná verze** – Zapíše informaci o shodě a pokračuje na další ovladač.

Zdrojový kód 6 - Funkce fncInstalaceOvladace

```
function fncInstalaceOvladace {
    Param(
        [Parameter(Mandatory = $true)]
        [string]$nazevSouboruCAB
    )
    subZapsatLogDoTxt "start fncInstalaceOvladace"
    #instalace ovladače do repozitáře
    $architekturaOvladace = (fncVratitBitovouVerziOS).replace("x86", "W32X86")
    $sitovaCestaSouborCab = "\\$tiskovyServer\print$\" + $architekturaOvladace
+ "\PCC\$nazevSouboruCAB"
    $lokalniCestaAdresarCab = ($cestaPracovnihoAdresareprogramu +
"\_tiskove_ovladace\" + $nazevSouboruCAB).Replace(".cab", "")
    $lokalniCestaSouborCab = $lokalniCestaAdresarCab + "\" + $nazevSouboruCAB
    $infNazev = ($nazevSouboruCAB -split "_")[0]
    $infpath = $lokalniCestaAdresarCab + "\" + $infNazev
    if (!(test-path -Path $lokalniCestaAdresarCab)) {
        $null = new-item -ItemType Directory -Path $lokalniCestaAdresarCab -
ErrorAction SilentlyContinue
    }
    if (!(test-path $lokalniCestaSouborCab)) {
        Copy-Item -Path $sitovaCestaSouborCab -Destination
$lokalniCestaAdresarCab
    }
    expand -F:* $lokalniCestaSouborCab $lokalniCestaAdresarCab | out-null
    $vysledek = pnputil -a $infpath
    [int]$vysledekKod = (($vysledek | select-string -Pattern "Number
successfully imported:") -split ":")[1]
    subZapsatLogDoTxt " exit fncInstalaceOvladace"
    return $vysledekKod
}
```

Zdrojový kód 7 - Zkrácená ukázka procedury subKontrolaChybejicichOvladacu

```
function subKontrolaChybejicichOvladacu {
    subZapsatLogDoTxt "start subKontrolaChybejicichOvladacu"
    foreach ($tiskarnaVzdalena in ($script:tiskarnyVzdalene)) {
        subZapsatLogDoTxt "      subKontrolaChybejicichOvladacu - ověřuji
ovladač pro: $($tiskarnaVzdalena.name)"
        subNapsatZpravuDoKonzole -sloupec1 'Ověřuji ovladač pro tiskárnu:' -
sloupec2 $tiskarnaVzdalena.name
        if ($logujOperace) { subObsahKonzoleDoHtml }
        if (($script:ovladaceLokalni | measure).count -gt 0) {
            $indexLokalni =
$script:ovladaceLokalni.name.IndexOf($tiskarnaVzdalena.DriverName)
#nalezneme pozadovany ovladač -1 ovladač nenalezen 0<= ovladač nalezen
        }
        else {
            $indexLokalni = -1
        }
        $indexVzdalene =
$script:ovladaceVzdalene.name.IndexOf($tiskarnaVzdalena.DriverName) #nalezneme
pozadovany ovladač -1 ovladač nenalezen 0<= ovladač nalezen
        $verzeVzdalena =
$script:ovladaceVzdalene[$indexVzdalene].DriverVersion
        $infPath = $script:ovladaceVzdalene[$indexVzdalene].infPath
        [array]$splitCestaOvladace = $infPath -split '\\' # rozdělíme cestu
oddělovačem zpětné lomítko s použitím escape \
        #$ovladacID = $nazevSouboruCAB =
$splitCestaOvladace[($splitCestaOvladace.count - 2)]
        $ovladacID = $splitCestaOvladace[($splitCestaOvladace.count - 2)] # z
cesty vybereme název, pod kterým tiskový server balí ovladače
        $nazevSouboruCAB = $ovladacID + ".cab" # přidáme příponu
        if ($indexLokalni -ge 0) {
            $verzeLokalni =
$script:ovladaceLokalni[$indexLokalni].DriverVersion
            if ($verzeLokalni -eq $verzeVzdalena) {
                subNapsatZpravuDoKonzole -sloupec3 'shodná verze' -
nahradSloupec sloupec3 -barva Green
            }
            else {
                subNapsatZpravuDoKonzole -sloupec3 'rozdílná verze' -
nahradSloupec sloupec3 -barva Yellow
                subNapsatZpravuDoKonzole -stavZpravy opravuji -nahradSloupec
sloupec4
                $vysledek = fncInstalaceOvladace -nazevSouboruCAB
$nazevSouboruCAB # nainstalujeme ovladač do windows repozitáře
                try {
                    if ($vysledek -ge 1) {
                        # byl nainstalován ovladač do windows repozitáře
                        Add-PrinterDriver -Name $tiskarnaVzdalena.DriverName #
přidá ovladač do tiskové služby
                    }
                    else {
                        throw "Nepovedl se nainstalovat ovladač pomocí
pnputil!"
                    }
                    subNapsatZpravuDoKonzole -stavZpravy opraveno -
nahradSloupec sloupec4
                }
                catch {
                    subNapsatZpravuDoKonzole -stavZpravy neopraveno -
nahradSloupec sloupec4
                    subZapsatLogDoTxt "start subKontrolaChybejicichOvladacu -
ověřuji ovladač pro: $($tiskarnaVzdalena.name) - neopraveno:
$(fncVratitDuvodChyby -psitem $psitem)"
                }
            }
        }
        else {
            subNapsatZpravuDoKonzole -stavZpravy neexistuje -nahradSloupec
sloupec3
            subNapsatZpravuDoKonzole -stavZpravy opravuji -nahradSloupec
sloupec4
                                    . . .
```

b) subKontrolaChybejicichPortu

Tato procedura má za úkol zkontrolovat všechny nalezené spravované tiskárny a porovnat správnost nastavení portů každé tiskárny vůči nastavení vzdálených tiskáren. Jedná se o nejsložitější proceduru, co se týče logiky zpracování a porovnání vlastností portů. Existují 2 nutné kontroly, a to zdali má port tiskárny nastavený protokol RAW nebo LPR. Na základě zjištěného protokolu jsou stanoveny vlastnosti k porovnání. Pro porovnání vlastností je použita rutina „Compare-Object", přičemž jakýkoliv rozdíl v hodnotě jedné položky postačuje pro neshodu v konfiguraci. Neshoda vyvolá aktualizaci konfigurace nainstalované tiskárny. Nativní rutiny nemají možnost změnit vlastností portu přímo, tedy neexistuje rutina Modify-PrinterPort. Pro jeho aktualizaci se musí port odebrat a znovu přidat. Toto však nelze provést, pokud je již nastaven a používán některou z tiskáren. Je tedy nutné na tiskárně (u které je potřeba nastavit správné hodnoty vlastností) nastavit jiný port. Jelikož má každý operační systém Windows nainstalované výchozí porty LPT1 až LPT3, je navrhováno nastavit tiskárnu dočasně na port LPT3. Tento port nebude z největší pravděpodobnosti obsazen jinou tiskárnou a nedojde tak ke konfliktu přidělení portu. Po uvolnění portu je následně port odebrán a přidán nový port se správně nastavenými vlastnostmi. Ten je následně nastaven na požadovanou tiskárnu. Tento proces se iterativně aplikuje na všechny nalezené tiskárny. V případě, že nebyl nalezen potřebný port na klientské stanici dle vzdálené tiskárny, provede se jeho nainstalování a zároveň se přidělí k dané lokální tiskárně. Ukázka kódu není z důvodu většího rozsahu zobrazena.

c) subKontrolaChybejicichTiskaren

Procedura pro detekci chybějících tiskáren. V případě, že existuje nainstalovaná tiskárna, je uživatel informován ve výstupu informací, že byla tiskárna nalezena. V opačném případě je tiskárna doinstalována – viz ukázka zdrojového kódu 8.

Zdrojový kód 8 - Zkrácená ukázka procedury subKontrolaChybejicichTiskaren

```
function subKontrolaChybejicichTiskaren {
    foreach ($tiskarnaVzdalena in $script:tiskarnyVzdalene) {
        subZapsatLogDoTxt "      subKontrolaChybejicichTiskaren - tiskárna:
$($tiskarnaVzdalena.Name)"
        subNapsatZpravuDoKonzole -sloupec1 'Ověřuji tiskárnu:' -sloupec2
$tiskarnaVzdalena.Name
        if ($logujOperace) { subObsahKonzoleDoHtml }
        if (($script:tiskarnyLokalni | measure).count -gt 0) {
            $indexLokalni =
$script:tiskarnyLokalni.name.IndexOf($tiskarnaVzdalena.Name)    #nalezneme
pozadovany název tiskárny index -1 = nenalezen, index větší nebo rovno 0 =
nalezen
        }
        else {
            $indexLokalni = -1
        }
        $indexVzdalene =
$script:tiskarnyVzdalene.name.IndexOf($tiskarnaVzdalena.Name) #nalezneme
pozadovany název tiskárny index -1 = nenalezen, index větší nebo rovno 0 =
nalezen
        $tiskarnaVzdalenaConfig = $script:tiskarnyVzdalene[$indexVzdalene]
        $tiskarnaVzdalenaName = $tiskarnaVzdalenaConfig.name
        $tiskarnaVzdalenaDriverName = $tiskarnaVzdalenaConfig.DriverName
        $tiskarnaVzdalenaPortName = $tiskarnaVzdalenaConfig.PortName
        $tiskarnaVzdalenaComment = $tiskarnaVzdalenaConfig.Comment
        $tiskarnaVzdalenaLocation = $tiskarnaVzdalenaConfig.Location
        if ($indexLokalni -ge 0) {
            subNapsatZpravuDoKonzole -sloupec3 'nalezena' -nahradSloupec
sloupec3 -barva Green
        }
        else {
            subNapsatZpravuDoKonzole -stavZpravy neexistuje -nahradSloupec
sloupec3
            subNapsatZpravuDoKonzole -stavZpravy opravuji -nahradSloupec
sloupec4
            if ($logujOperace) { subObsahKonzoleDoHtml }
            try {
                if ($nastavovatSecurity) {
                    $tiskarnaVzdalenaPermissionSDDL =
$tiskarnaVzdalenaConfig.PermissionSDDL
                Add-Printer -Name $tiskarnaVzdalenaName -DriverName
$tiskarnaVzdalenaDriverName -PortName $tiskarnaVzdalenaPortName -
PermissionSDDL $tiskarnaVzdalenaPermissionSDDL -Comment
$tiskarnaVzdalenaComment -Location $tiskarnaVzdalenaLocation
                }
                                    . . .
```

d) subNacteniLokalniKonfiguraceTiskaren

Procedura pro naplnění globální proměnné $tiskarnyLokalniPredvolby s nastavenými předvolbami lokálně nainstalovaných tiskáren, která je použita pro následnou kontrolu konfigurace.

e) subKontrolaLokalniKonfiguraceTiskaren

Pro každou nainstalovanou lokální tiskárnu je provedena kontrola její aktuální konfigurace (nově nainstalovaná tiskárna má vždy výchozí konfiguraci) s konfigurací vzdálené tiskárny. Provádí se porovnání 2 typů. Porovnání vlastností

tiskárny, tzn. zda odpovídá název ovladače, název portu, nastavená zabezpečení tiskárny, tiskový procesor, protokol (RAW/LPR), oddělovací stránka. V případě nesrovnalostí dojde k nápravě.

A porovnání nastavených předvoleb tiskárny, tzn. řazení tisku (collate), barevnost tisku, oboustrannost tisku a další rozšířených předvoleb v datovém formátu XML viz ukázka zkráceného zdrojového kódu 9.

Zdrojový kód 9 - Zkrácená ukázka procedury subKontrolaLokalniKonfiguraceTiskaren

```
function subKontrolaLokalniKonfiguraceTiskaren {
    subZapsatLogDoTxt "start subKontrolaLokalniKonfiguraceTiskaren - call
subprocess subNacteniLokalnichTiskaren"
    subNacteniLokalnichTiskaren
    foreach ($tiskarnaVzdalena in $script:tiskarnyVzdalene) {
        $nazevKontrolovaneTiskarny = $tiskarnaVzdalena.Name
        subZapsatLogDoTxt "      subKontrolaLokalniKonfiguraceTiskaren -
ověřuji konfiguraci -  Předvolby: $nazevKontrolovaneTiskarny"
        #subNapsatZpravuDoKonzole -sloupec1 ' ' -
zarovnatNaPredchoziSloupce:$true #slouží k zarování následujícího sloupce
        subNapsatZpravuDoKonzole -sloupec1 'Ověřuji konfiguraci tiskárny:' -
sloupec2 $nazevKontrolovaneTiskarny
        if ($logujOperace) { subObsahKonzoleDoHtml }
        ### začátek vlastnosti
        subNapsatZpravuDoKonzole -sloupec2 'Vlastnosti'
        subZapsatLogDoTxt "      subKontrolaLokalniKonfiguraceTiskaren -
ověřuji konfiguraci - Vlastnosti: $nazevKontrolovaneTiskarny"
        $indexLokalniVlastnosti =
$script:tiskarnyLokalni.Name.IndexOf($nazevKontrolovaneTiskarny)    #nalezneme
pozadovany port -1 port nenalezen 0<= port nalezen
        $indexVzdaleneVlastnosti =
$script:tiskarnyVzdalene.Name.IndexOf($nazevKontrolovaneTiskarny) #nalezneme
pozadovany port -1 port nenalezen 0<= port nalezen
        $tiskarnaLokalniVlastnosti =
$script:tiskarnyLokalni[$indexLokalniVlastnosti]
        $tiskarnaVzdalenaVlastnosti =
$script:tiskarnyVzdalene[$indexVzdaleneVlastnosti]
        [string]$tiskarnaVzdalenaVlastnostiName =
$tiskarnaVzdalenaVlastnosti.name
        [string]$tiskarnaVzdalenaVlastnostiDriverName =
$tiskarnaVzdalenaVlastnosti.DriverName
                                    . . .
        [string]$tiskarnaVzdalenaVlastnostiLocation =
$tiskarnaVzdalenaVlastnosti.Datatype
        [string]$tiskarnaVzdalenaVlastnostiSeparatorPageFile =
$tiskarnaVzdalenaVlastnosti.SeparatorPageFile

        if ($tiskarnaVzdalenaVlastnosti.SeparatorPageFile.length -eq 0) {
$tiskarnaVzdalenaVlastnosti.SeparatorPageFile = "" } # v případě nevyplnění na
serveru vrací $null, ale pracovní stanice vrací prázdný string, tak $null
změníme na prázdný string pro porovnání
        if ($tiskarnaVzdalenaVlastnosti.Location.length -eq 0) {
$tiskarnaVzdalenaVlastnosti.Location = "" }
                                    . . .
$tiskarnaLokalniVlastnosti.Location = "" }
        if ($tiskarnaLokalniVlastnosti.Comment.length -eq 0) {
$tiskarnaLokalniVlastnosti.Comment = "" }

        if ($nastavovatSecurity) {
            [string]$tiskarnaVzdalenaVlastnostiPermissionSDDL =
$tiskarnaVzdalenaVlastnosti.PermissionSDDL
           $vysledek = Compare-Object -ReferenceObject
$tiskarnaVzdalenaVlastnosti -DifferenceObject $tiskarnaLokalniVlastnosti  -
Property name, DriverName, PortName, PermissionSDDL, Location, Comment,
PrintProcessor, Datatype, SeparatorPageFile | % { $_.SideIndicator -eq "=>" }
        }
                                    . . .
```

V. Čištění

Následující procedury nejsou podrobně popisované, neboť je z jejich názvu patrný jejich účel.

a) **subKontrolaRegistru**
b) **subOdebraniNespravovanychTiskaren**
c) **subOdebraniNespravovanychPortu**
d) **subOdebraniNepotrebnychOvladacu**

Je třeba ještě zmínit proceduru „**subObsahKonzoleDoHtml**“, která není vytvořena autorem diplomové práce, ale je využita na základě licence MIT (Massachusetts Institute of Technology) z volně dostupného uložiště GitHub Gist. Zde je vedena pod originálním názvem „ConsoleBufferToHtml.ps1“ od autora Mattiase Karlssona.

Jelikož licence MIT umožňuje svobodně nakládat se zdrojovým kódem, byl tento kód využit v této práci. A zároveň byla do kódu implementována meta informace pro automatickou obnovu stránky každých 5 s.

Princip procedury je, že při každém její zavolání dojde k převodu obsahu konzolového bufferu do HTML formátu a k jeho následnému uložení do souboru s příponou html.

4.7 Testování programu

Cílem testování je zjistit, zdali navrhovaný program pracuje dle očekávání, plní všechny stanovené požadavky a v případě zjištěných chyb tyto chyby opravit. Pro názornou ukázku navrhovaného programu byly stanoveny 3 testovací scénáře.

4.7.1 Testovací scénář – Instalace chybějících tiskáren

Předpoklady

- Tiskový server je dostupný a obsahuje konfiguraci pro několik tiskáren.
- Naplánovaná úloha je správně nastavená pro spuštění na vyžádání uživatele či spuštění systémem po startu počítače.
- Vytvořený souborový zástupce „Správa tiskáren – aktualizace“ v nabídce start pro spuštění naplánované úlohy uživatelem.
- Na testovací stanici nejsou nainstalovány žádné tiskárny.

Testovací kroky

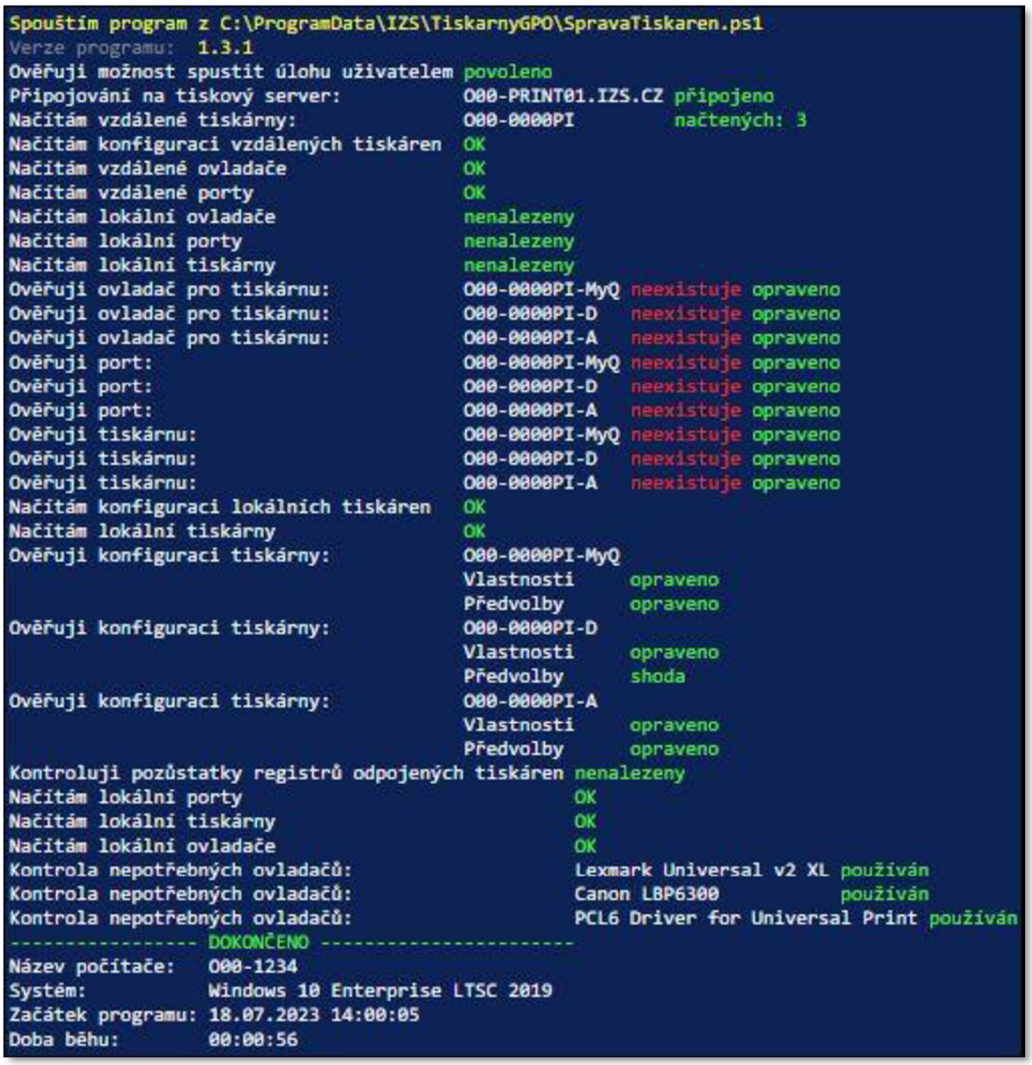
1. Spuštění programu pomocí souborového zástupce „Správa tiskáren – aktualizace“ v nabídce start. Očekává se spuštění programu a otevření webového prohlížeče se stavovým výstupem instalace.
2. Program by měl detekovat úspěšné připojení na tiskový server, provést detekci umístění stanice v OU a načtení potřebných tiskáren pro instalaci.
3. Program by měl postupně nainstalovat a nakonfigurovat všechny chybějící tiskárny dle zjištěných informací z tiskového serveru.
4. Po dokončení instalace by měl program ověřit nepotřebné ovladače a případně je odebrat.

Očekávané výsledky

- Všechny potřebné tiskárny zjištěné z tiskového serveru, které nebyly před spuštěním programu nainstalovány, by měly být úspěšně nainstalovány a nakonfigurovány.

- Ve stavovém výstupu jsou vypsány všechny provedené kroky o instalaci a konfiguraci potřebných tiskáren s úspěšným výsledkem – podbarvené zelenou barvou dle obrázku č. 14.

Obrázek 14 - Výsledek testovacího scénáře – Instalace chybějících tiskáren



Autor: vlastní zpracování

4.7.2 Testovací scénář – Přesun stanice do jiné OU

Předpoklady

- Tiskový server je dostupný a obsahuje konfiguraci pro několik tiskáren.
- Naplánovaná úloha je správně nastavená pro spuštění na vyžádání uživatele či spuštění systémem po startu počítače.

- Vytvořený souborový zástupce „Správa tiskáren – aktualizace“ v nabídce start pro spuštění naplánované úlohy uživatelem.
- Na testovací stanici jsou nainstalovány tiskárny cizí OU.

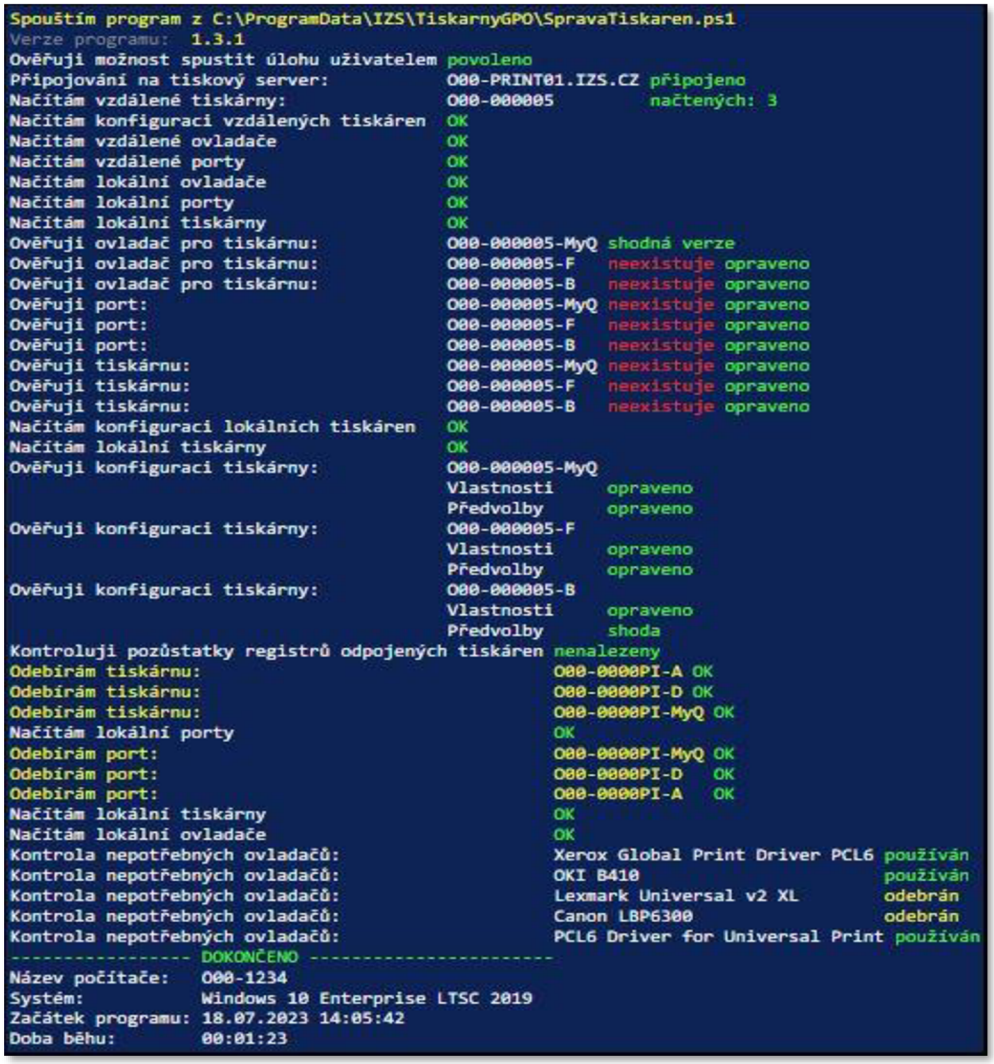
Testovací kroky

1. Spuštění programu pomocí souborového zástupce „Správa tiskáren – aktualizace“ v nabídce start. Očekává se spuštění programu a otevření webového prohlížeče se stavovým výstupem instalace.
2. Program by měl detekovat úspěšné připojení na tiskový server, provést detekci umístění stanice v OU a načtení potřebných tiskáren pro instalaci.
3. Program by měl postupně nainstalovat a nakonfigurovat všechny chybějící tiskárny dle zjištěných informací z tiskového serveru.
4. Po dokončení instalace by měl program odstranit cizí tiskárny nepatřící k dané OU.
5. Následně by měl odebrat nepoužívané tiskové porty a ovladače.

Očekávané výsledky

- Všechny potřebné tiskárny, zjištěné z tiskového serveru, které nebyly před spuštěním programu nainstalovány, by měly být úspěšně nainstalovány a nakonfigurovány.
- Všechny tiskárny nepatřící pro OU, ve které je umístěna testovací stanice by měly být odebrány, stejně tak pozůstatky tiskárny jako jsou tiskové porty a ovladače.
- Ve stavovém výstupu jsou vypsány všechny provedené kroky o instalaci a konfiguraci potřebných tiskáren s úspěšným výsledkem – podbarvené zelenou barvou dle obrázku č. 15.

Obrázek 15 - Výsledek testovacího scénáře – Přesun stanice do jiné OU



Autor: vlastní zpracování

4.7.3 Testovací scénář – Ověření shodné konfigurace tiskáren ve stejné OU

Předpoklady

- Tiskový server je dostupný a obsahuje konfiguraci pro několik tiskáren.
- Naplánovaná úloha je správně nastavená pro spuštění na vyžádání uživatele či spuštění systémem po startu počítače.
- Vytvořený souborový zástupce „Správa tiskáren – aktualizace“ v nabídce start pro spuštění naplánované úlohy uživatelem.
- Na testovací stanici jsou nainstalovány tiskárny pro danou OU.

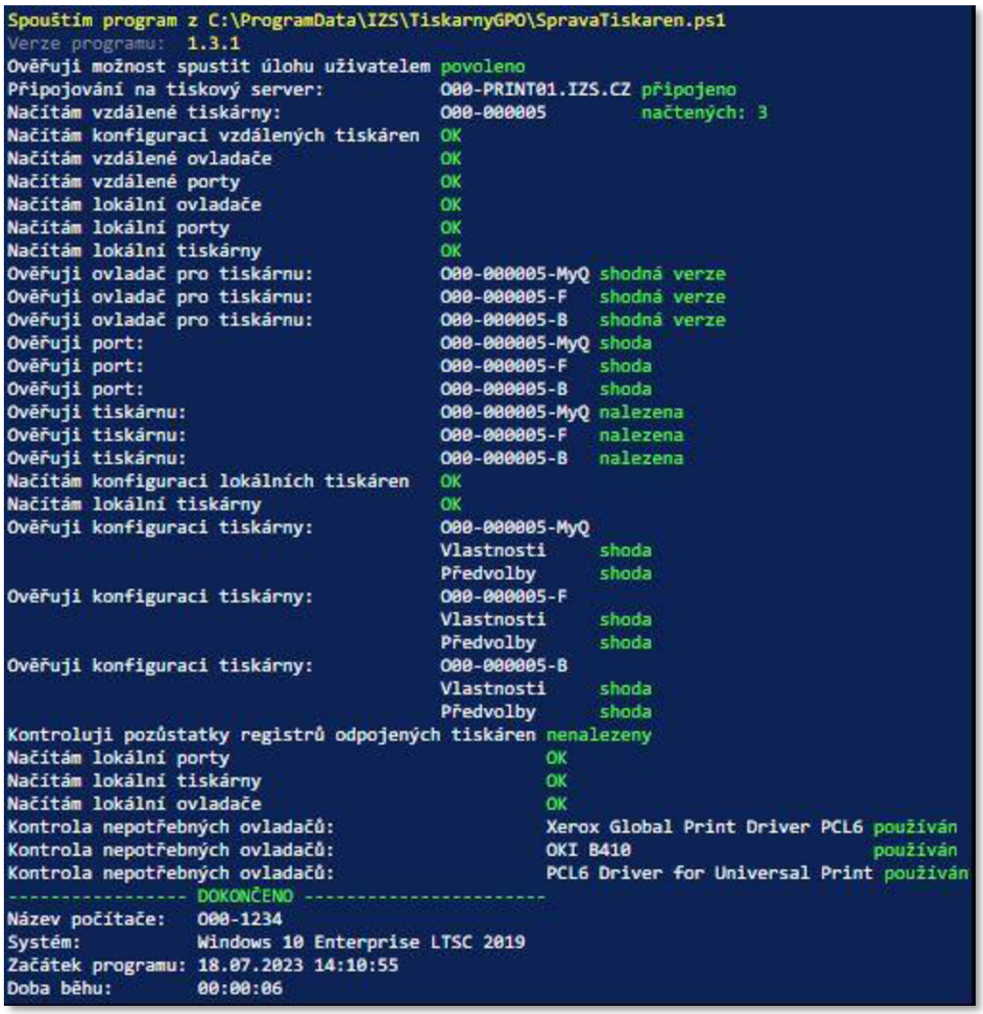
Testovací kroky

1. Spuštění programu pomocí souborového zástupce „Správa tiskáren – aktualizace“ v nabídce start. Očekává se spuštění programu a otevření webového prohlížeče se stavovým výstupem instalace.
2. Program by měl detekovat úspěšné připojení na tiskový server, provést detekci umístění stanice v OU a načtení potřebných tiskáren pro instalaci.
3. Program by měl detekovat aktuálně nainstalované tiskárny a porovnat je dle zjištěných informací z tiskového serveru.
4. Nemělo by dojít k žádné instalaci, odinstalaci ani aktualizaci tiskáren, tiskových portů a ovladačů.

Očekávané výsledky

- Všechny tiskárny jsou již nainstalovány a mají shodné nastavení s tiskárnami dle konfigurace na tiskovém serveru.
- Ve stavovém výstupu jsou vypsány všechny provedené kroky o instalaci a konfiguraci potřebných tiskáren s úspěšným výsledkem – podbarvené zelenou barvou dle obrázku č. 16.

Obrázek 16 - Výsledek testovacího scénáře – Ověření shodné konfigurace tiskáren ve stejné OU



Autor: vlastní zpracování

4.8 Nasazení programu do klientských stanic

Po otestování programu přichází fáze zavedení programu do firemního prostředí. Důležitým aspektem pro nasazení programu do firemního prostředí je vyhnutí se negativnímu dopadu na práci všech zaměstnanců. V tomto případě je nutné vybrat takovou strategii, aby v případě nečekané chyby nedošlo k zastavení pracovní činnosti zaměstnanců, či se jinak nenarušil průběh jejich práce. Po dohodě s tiskovým administrátorem byla zvolena strategie nasazení po jednotlivých odděleních. A to z důvodu, aby byla zachována okamžitá reakce na opravu pro případ, že by došlo k nečekaným chybám, např. k odinstalování síťových tiskáren nutných pro práci zaměstnanců.

4.8.1 Připravení doménových politik pro nasazení

Pro zrychlení nasazování síťových tiskáren byly v doméně AD nastaveny doménové politiky do každé OU oddělení, kde byly pouze připraveny. Tyto politiky byly dočasně deaktivovány a budou postupně aktivovány při zavádění programu. Taktéž byla připravena globální politika na nejvyšší úrovni OU, která nahradí nastavené dočasné politiky po kompletní instalaci síťových tiskáren.

Definice nastavení globální politiky je uvedena *v **Příloze 1 – Nastavení doménové politiky – globální***. Uvedený postup nastavuje spouštění programu pomocí naplánované úlohy, a to z důvodu pro naplnění funkčního požadavku uvedeného v kapitole 4.3 v bodě a). V případě, že by takový požadavek nebyl, bylo by možné politiku nastavit při nativním podporovaném spuštění po startu nebo vypnutí systému bez nutnosti nastavovat naplánovanou úlohu. To by však přineslo nevýhodu ve smyslu nutného restartu počítače pro aktualizaci tiskáren a zároveň nemožnost spustit program uživatelem.

Jednou z výhod používání AGPM je možnost vytvořit z již vytvořené politiky šablonu, kterou je pak možné opětovně použít při vytváření dalších podobných politik bez nutnosti složitého nastavování.

Další postup nastavení politiky pro rozšíření instalaci tiskáren nad rámec přidělených na oddělení je uveden v příloze s názvem **Příloha 2 – Nastavení doménové politiky – rozšířená**. Tato politika umožňuje spouštění programu s parametrem pro přidání dalších tiskáren, jiných než pro dané oddělení. Toto rozšíření programu je z důvodu, že tiskárny můžou být umístěné na společném patře dvou či více oddělení – viz funkční požadavek v kapitole 4.3 v bodě b).

Poslední přidanou položkou do některých globálních politik je vytvoření zástupce na plochu či do nabídky start. V tomto případě byla použita jedna z centrálních politik, do které byl vytvořen spouštěcí zástupce pro aktualizaci tiskáren s následujícím příkazem pro spuštění:

„C:\Windows\System32\cmd.exe /c schtasks /run /tn "PowerShell-SpravaTiskaren-default" || schtasks /run /tn "PowerShell-SpravaTiskaren" & start C:\ProgramData\IZS\TiskarnyGPO\Tiskarny-GPO-LOG.html“

Tento zástupce umožní uživateli spustit program pro aktualizaci síťových tiskáren a výchozí webový prohlížeč, ve kterém se budou zobrazovat aktuální stavy běhu programu – viz příklady výstupů programu v kapitolách 4.7.1, 4.7.2 a 4.7.3.

4.8.2 Seznámení uživatelů s chystanými kroky

Jedním z důležitých kroků bylo seznámení uživatelů s chystanými kroky, které mohly ovlivnit jejich pracovní den. Bylo tedy nutné na to uživatele předem připravit. Z tohoto důvodu byl stanoven harmonogram plánovaného nasazení pro první týden – viz tabulce č. 6, a to pro vybrané oddělení. Zároveň jim byl vysvětlen způsob instalace a použití vytvořeného zástupce, který v případě různých problémů s tiskárnou musejí spustit a vyčkat na dokončení činnosti programu. V případě, že tímto krokem nebyl problém vyřešen, pak musejí vygenerovaný stav uložit a poslat pomocí emailu tiskovému administrátoru k vyhodnocení. Taktéž nesmíme opomenout informovat oblastní administrátory, kteří jsou v první linii při řešení uživatelských požadavků s chystaným nasazením.

Tabulka 6 - Harmonogram nasazení programu

		Harmonogram nasazení	
Den	**Čas**	**Typ činnosti**	**Typ politiky**
Pondělí	9:00 - 12:00	nasazení na oddělení A, B, C	globalní
	13:00-16:00	kontrola instalovaných tiskáren na oddělení A, B, C	-
Úterý	9:00 - 12:00	nasazení na oddělení C, D, E	rozšířená
	13:00-16:00	kontrola instalovaných tiskáren na oddělení C, D, E	-
Středa	9:00 - 12:00	nasazení na oddělení E, F, G	globalní
	13:00-16:00	kontrola instalovaných tiskáren na oddělení E, F, G	-
Čtvrtek	9:00 - 12:00	nasazení na oddělení H, I, J	rozšířená
	13:00-16:00	kontrola instalovaných tiskáren na oddělení H, I, J	-
Pátek	8:00 - 16:00	vyhodnocení, oprava kódu programu odpojení nepotřebných doménových politik	-
…	…	…	…
…	…	…	…

Autor: vlastní zpracování

4.8.3 Nasazení programu

Po přípravné fázi vytvoření doménových politik a zároveň informování všech zaměstnanců na daných oddělení bylo přistoupeno k aktivování připravených doménových politik. Jelikož aktuální konfigurace síťových tiskáren na klientských stanicích je formou připojení, nikoliv samotnou instalací, může být program spuštěn paralelně s aktuálním nastavením, aniž by došlo k odstranění již zavedených tiskáren. Z tohoto důvodu rozhodl tiskový administrátor, že první týden pro nasazení programu budou ponechány staré konfigurace tiskáren s novými instalacemi. Rozlišení pro uživatele je v názvu tiskáren, kde původní připojené tiskárny mají za názvem tiskárny název tiskového serveru. Příklad názvu dosavadní připojené tiskárny: „O00-00006-F na O00-PRINT01“. Díky tomuto názvu lze tedy rozlišit nově nainstalované tiskárny od původních. Po nainstalování tiskáren si budou muset všichni zaměstnanci přenastavit svou původní výchozí tiskárnu na nově nainstalovanou tiskárnu a zároveň provést zkušební tisk.

5 Vyhodnocení nasazení a možnosti zlepšení

5.1 Vyhodnocení prvního dne nasazení

První den byla rychlost instalace nad očekávání. Instalace všech přiřazených tiskáren na oddělení byly nainstalovány v průměru za 81–98 vteřin při počtu 10 tiskáren na počítač. Následná kontrola nainstalovaných tiskáren a jejich konfigurace při opětovném spuštění programu trvala v průměru očekávaných 19–21 vteřin, což je na prosto v pořádku a v mezích požadavku. Je samozřejmě nutno brát v potaz heterogennost jednotlivých hardwarových konfigurací klientských stanic, jejich aktuální vytížení a rychlost pevného disku. Celková doba instalace také závisí na počtu síťových tiskáren využívajících stejný tiskový ovladač. A zdali jsou tiskárny on-line či off-line. Bylo také zjištěno, že se téměř zdvojnásobuje čas instalace v případě, že jsou tiskárny off-line. Z tohoto důvodu nelze objektivně posoudit výkonnost programu.

Tabulka 7 - Výsledky doby instalace tiskáren

PC	CPU	HDD	Doba instalace (s)	Průměrná doba inst. (s)	Doba kontroly (s)	Průměrná doba kont. (s)
1	AMD A8-7600	SSD	97		21	
2	AMD A8-7600	SSD	99		20	
3	AMD A8-7600	SSD	99	**98**	19	**21**
4	AMD A8-7600	SSD	98		21	
5	AMD A8-7600	SSD	97		22	
6	Intel I5-10500	NVMe	88		19	
7	Intel I5-10500	NVMe	75		18	
8	Intel I5-10500	NVMe	79	**81**	18	**19**
9	Intel I5-10500	NVMe	79		19	
10	Intel I5-10500	NVMe	85		19	

Autor: vlastní zpracování

5.2 Vyhodnocení druhého a dalšího dne nasazení

V druhém dni se nasazení programu pro automatizovanou instalaci síťových tiskáren neobešlo bez obtíží. Při tomto nasazení je použit parametr -tiskarnyNavic, kde jsou přidávány hodnoty jako pole textového řetězce s názvy přidávaných tiskáren. Ty se mají instalovat nad rámec přidělených tiskáren na oddělení. Tyto tiskárny jsou vypisovány ve

stavovém výpisu programu, avšak nepředpokládalo se omezení velikosti konzolového okna v případě spuštění programu na pozadí pod systémovým účtem. Takovou informaci o omezení nebylo možné z dostupných zdrojů zjistit a v provedených testech nebylo použito množství přidaných tiskáren překračující toto omezení. Jelikož je program navržen tak, že nastavuje pozici kurzoru na ose X (pro přepis požadovaného řetězce) na začátek pozice konkrétního stavového řetězce, došlo k předání hodnoty větší než povolené maximální hodnoty pro pozici na ose X. Postupným testováním bylo zjištěno, že pod systémovým účtem nelze nastavit terminálové okno na více než 128 znaků šířky. Pokud dojde k vložení textového řetězce překračující 128 znaků, systém automaticky řetězec zalomí na nový řádek, avšak počet znaků řetězce zůstává stejný. Tento počet znaků byl použit pro nastavení pozice X, který po překročení vyvolal chybu. Z tohoto důvodu byla zapracovaná kontrola, která přepočítá hodnotu na povolený rozsah osy X od 0 do 128.

Po opravě tentýž den došlo v následujících dnech k postupnému nasazení programu na všechna oddělení, u kterých již bylo nasazení bezproblémové. Všechny požadované síťové tiskárny byly dle očekávání správně nainstalovány.

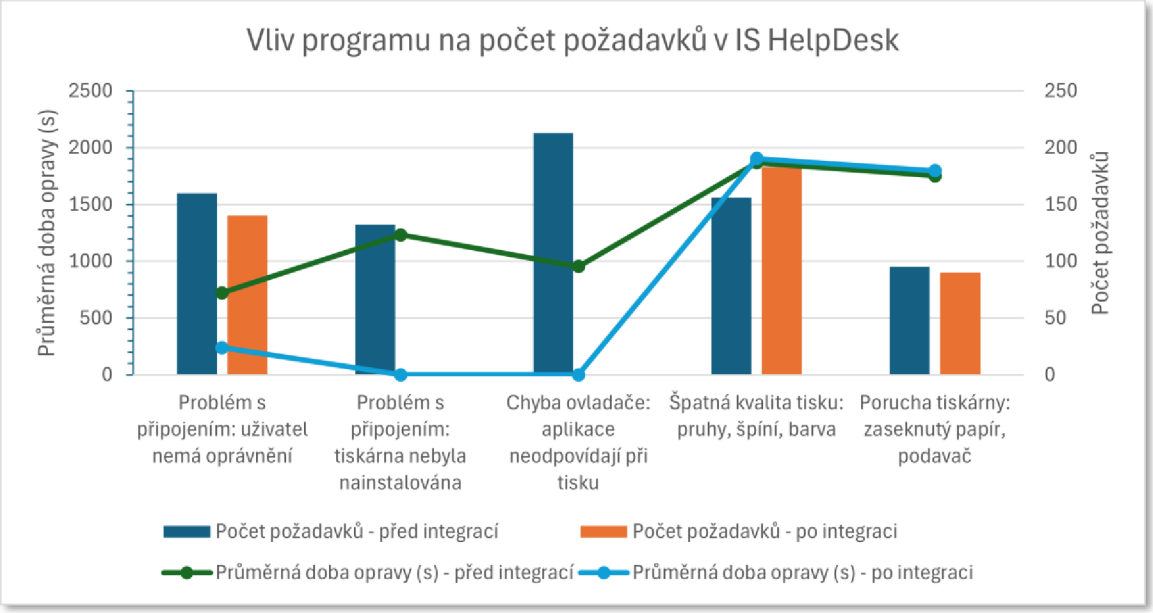
5.3 Vyhodnocení z dlouhodobého hlediska

Pro účely ověření, zda byl odstraněn problém s instalacemi při původním nasazováním síťových tiskáren do klientských stanic, bylo provedeno vyhodnocení vlastního organizačního IS HelpDesk na počet vložených požadavků. Do tohoto systému jsou vkládány všechny požadavky na podporu, dotazů nebo různé technické opravy týkající se informačních a komunikačních technologií.

Ze systému byly vybrány pouze požadavky týkajících se všech tiskáren za půl roku před nasazením a půl roku po nasazení programu. Získané hodnoty byly vloženy do kombinovaného sloupcového grafu – viz graf č. 1.

Z grafu je patrné, že vlivem nasazení programu došlo zcela k odstranění problému s připojením tiskáren, kde z důvodu chybějících ovladačů nebyly tiskárny nainstalovány. Taktéž byly odstraněny pády aplikací z důvodu problému s tiskovými ovladači. Dále je patrný pokles průměrné doby ukončení požadavku při detekci problému s připojením tiskárny, jež je identifikován jako chybějící oprávnění uživatele k dané tiskárně. To může být z důvodu toho, že administrátoři mají přehled o aktuálním stavu nainstalovaných tiskáren prostřednictvím souhrnného výstupu a nemusejí informaci pracně hledat v systémovém logu.

Graf 1 - Vliv programu na počet požadavků v IS HelpDesk



Autor: vlastní zpracování

5.4 Vyhodnocení programu z pohledů administrátorů

Na závěr vyhodnocení byl proveden rozhovor s jednotlivými oblastními administrátory a s tiskovým administrátorem k získání vyjádření přínosu nasazeného programu. Všichni se shodli na tom, že hlavní přínos spatřují v možnosti okamžitě rozpoznat, zda existuje problém s tiskárnou, a to díky barevnému rozlišení jejího aktuálního stavu.

Tiskový administrátor také uvedl, že má konečně více času na jiné činnosti než složitě nastavovat instalaci tiskáren pomocí doménových zásad, které mu díky nasazení programu částečně odpadly. Nyní má konečně jistotu, že dochází k okamžitému a přesnému nastavení nainstalovaných tiskáren na klientských stanicích v případě jeho změny konfigurace na tiskovém serveru.

5.5 Možnosti dalšího zlepšení programu

V průběhu času autora napadlo několik postřehů na rozšíření funkčnosti programu, kterými jsou mimo jiné **přidání možnosti detekce oprávnění tisku** u aktuálně přihlášeného uživatele na dané tiskárně. S odstupem času totiž oblastní administrátoři zapomněli na

nastavená oprávnění tiskáren, která povolují tisk pouze uživatelům daného oddělení. V případě, že přijde na oddělení jiný uživatel např. z jiného oddělení, tak se mu tiskárna nezobrazí v ovládacím panelu a v aplikaci např. Word či Excel vidí tiskárnu jako nepřipojenou. To mylně vyvolává domněnku chyby a vyvolává hledání závady, neboť program ukazuje, že je vše v pořádku. Toto se umocňuje s aktuálním nasazováním aktualizací klientských stanic na operační systém Windows 11, kde se mění grafické prostředí ovládacích panelů tiskáren. S novým grafickým prostředím se seznamují jak samotní uživatelé, tak administrátoři. Ze zkušenosti administrátoři vědí, že s každou novou verzi operačního systému přicházejí i nové problémy.

Dalším možným rozšířením je možnost provedení **kompletní reinstalace všech nainstalovaných síťových tiskáren.** Toto se hodí v případě provedení špatného nastavení předvoleb uživatelem nebo například špatným přenesením nastavení vlivem aktualizace systému na novější verzi.

6 Závěr

Tato diplomová práce si kladla za cíl navrhnout a implementovat program pro alternativní možnosti zavádění síťových tiskáren za pomocí nástroje PowerShell v prostředí Active Directory v organizaci v rámci IZS. Ve zmíněné organizaci docházelo k problémům při nasazování síťových tiskáren konvenčním způsobem.

V teoretické části práce byl představen nástroj Powershell, který je specifický právě pro automatizování různých stereotypních procesů v oblasti informačních technologií jak je například využito v této práci pro instalaci tiskáren bez nutnosti osobního přístupu administrátora na pracovní stanici. Byly představeny nejčastější konstrukce příkazů PowerShellu, možnosti různých operací s nimi, pracovní prostředí a v neposlední řadě vlastnosti zabezpečení spouštění PowerShellových souborů. Dále bylo provedeno seznámení se službou Active Directory a jeho složení, stejně tak představení nedílné součástí doménových zásad. Posledním seznámením v teoretické práci byla prozkoumaná funkce, struktura a aplikace regulárních výrazů, přičemž byl kladen důraz na klíčové aspekty – metaznaky a kvantifikátory.

V praktické části práce byl představen vyskytující se problém při hromadné instalaci síťových tiskáren prostřednictvím doménových zásad v jedné organizaci v rámci IZS. Následovala část analýzy aktuálního procesu nasazování tiskáren v organizaci prostřednictvím rozhovoru s administrátory organizace. Na základě rozhovoru byl proces nasazování tiskáren přenesen do grafické notace pomocí diagramu BPMN, kde byla následně navrhnuta optimalizace procesu za pomocí navrhovaného programu. Nedílnou součástí rozhovoru byl také sběr informací na funkční a nefunkční požadavky programu. Sběr informací je klíčový pro porozumění očekávání a potřeb pro návrh programu.

Byla provedena detailní analýza Active Directory a s tím spojená analýza doménových zásad, která byla důležitým krokem pro zjištění aktuální hierarchie a názvů klientských stanic. A to včetně jejich nastavení, která by mohla být potencionálním problémem při spouštění navrhovaného programu. Taktéž byla analýza důležitá pro zjištění možností identifikace síťových tiskáren prostřednictvím názvu organizační jednotky pro jejich instalaci na dané klientské stanici.

Po klíčových analýzách bylo přistoupeno k návrhu a implementaci programu, při kterém byl program rozdělen na 5 hlavních fází. První fáze programu, tedy fáze inicializace, obstarává zjištění názvu umístění pracovní stanice v organizační jednotce a následné

vytvoření spojení s tiskovým serverem. Druhá fáze je fáze vzdáleného načítání. To obnáší načtení požadovaných síťových tiskáren a jejich důležitých komponent, kterými jsou ovladače, porty, vlastnosti a předvolby nastavení z tiskového serveru. Třetí fáze je fáze lokálního načtení. Ta je obdobná s třetí fází, ale s provedením na klientské stanici. Srdcem všeho je čtvrtá fáze, ve které dochází k porovnávání vlastností komponent mezi vzdálenou a lokální konfigurací. Tato fáze přizpůsobuje komponenty do požadovaného, stavu jako jsou nové instalace tiskáren či aktualizace konfigurací a ovladačů. Poslední fází je fáze čištění, kde dochází k odstranění všech nepotřebných spravovaných tiskáren a jejich komponent.

Po naimplementování programu byl proveden test programu. Vyšel v testovacích scénářích úspěšně dle očekávání, proto bylo přistoupeno k jeho nasazení do klientských stanic. I když se v počátcích vyskytl problém při nasazování tiskáren přidělených nad rámec z jiného oddělení, byl problém nalezen a ihned opraven ve stanoveném čase. A to před nasazením v dalších dnech, kde byly tiskárny bez dalších problému nainstalovány.

Doba běhu instalace závisela na více faktorech – výkon klientských stanic a dostupnost síťových tiskáren. Tedy zdali jsou tiskárny on-online či off-line a jaké měly použité ovladače. Výsledky časů se obecně pohybovaly průměrně mezi 81 až 98 vteřinami při instalaci 10 tiskáren. Následné spouštění programu (kontrola změn) bez provedených rekonfigurací na tiskárnách trvalo v průměru 19 až 21 vteřin (při kontrole 10 tiskáren). To je v toleranci v rámci nefunkčních požadavků.

Při vyhodnocení nasazení programu z dlouhodobého hlediska byl proveden sběr zadaných požadavků v organizačním IS HelpDesk. Z dat bylo zjištěno, že nasazení programu přineslo očekávány výsledek. A to zejména úplné odstranění problému s instalacemi ovladačů síťových tiskáren. Dále došlo ke zrychlení rozlišení nahlášeného požadavku na opravu chybějící tiskárny v seznamu tiskáren uživatele. Tedy rozlišení, jestli se jedná o nedostatečná oprávnění uživatele tisknout na tiskárnu nebo tiskárna nebyla vůbec přiřazená pro instalaci tiskovým administrátorem na dané oddělení. Tiskovému administrátorovi se také snížil počet nutných kroků pro přidání síťových tiskáren na oddělení. Snížení počtu kroků mu přineslo velkou časovou úsporu, kterou může využít na další pracovní činnost.

Závěrem lze říci, že implementovaný program splnil veškeré stanovené cíle. Zároveň se nabízí další potenciál pro využití i v jiných organizacích, které se potýkají s podobnými problémy při nasazování síťových tiskáren.

7 Seznam použitých zdrojů

1. LEE, Thomas. Windows PowerShell 2.0 Bible. USA: Wiley, 2011. ISBN 9781118021989.
2. Regulární výrazy v příkladech. Root.cz [online]. [cit. 2023-08-03]. Dostupné z: https://www.root.cz/clanky/regularni-vyrazy-v-prikladech/
3. Regulární výrazy - Základy, příklady (datum, telefonní číslo). *Regularnivyrazy.info* [online]. 2018 [cit. 2023-08-03]. Dostupné z: https://www.regularnivyrazy.info/regularni-vyrazy-zaklady.html
4. STANEK, William R. *Group Policy: zásady skupiny ve Windows : kapesní rádce administrátora*. Brno: Computer Press, 2010. ISBN 9788025129203
5. Co jsou skupiny zásad (Group Policy). *Ondřej Soukup* [online]. 2016 [cit. 2023-09-22]. Dostupné z: https://www.ondrej-soukup.cz/2009/09/co-jsou-skupiny-zasad-group-policy
6. STANEK, William R. *Active Directory: kapesní rádce administrátora*. Brno: Computer Press, 2009. Microsoft (Computer Press). ISBN 9788025125557.
7. WILSON, Ed. *PowerShell*. 2. vydání. Přeložil Jakub GONER. Brno: Computer Press, 2022. ISBN 9788025150498.
8. How User Account Control works - Windows Security. *Windows Security | Microsoft Learn* [online]. 2023 [cit. 2023-10-26]. Dostupné z: https://learn.microsoft.com/en-us/windows/security/application-security/application-control/user-account-control/how-it-works
9. Group-Object (Microsoft.PowerShell.Utility) - PowerShell. *(Microsoft.PowerShell.Utility) - PowerShell | Microsoft Learn* [online]. 2023 [cit. 2023-10-27]. Dostupné z: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/group-object?view=powershell-5.1
10. Sorting objects - PowerShell. *PowerShell | Microsoft Learn* [online]. 2022 [cit. 2023-10-26]. Dostupné z: https://learn.microsoft.com/en-us/powershell/scripting/samples/sorting-objects?view=powershell-5.1
11. Where-Object (Microsoft.PowerShell.Core) - PowerShell. *(Microsoft.PowerShell.Core) - PowerShell | Microsoft*

Learn [online]. 2023 [cit. 2023-10-27]. Dostupné z: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/where-object?view=powershell-5.1

12. *About Variables - PowerShell | Microsoft Learn* [online]. 2023 [cit. 2023-10-28]. Dostupné z: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_variables?view=powershell-5.1
13. MALINA, Patrik. *Powershell: podrobný průvodce skriptováním*. Brno: Computer Press, 2007. ISBN 9788025118160
14. About Execution Policies - PowerShell. *About Execution Policies - PowerShell | Microsoft Learn* [online]. 2023 [cit. 2023-10-03]. Dostupné z: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_execution_policies?view=powershell-5.1
15. About Signing - PowerShell. *About Signing - PowerShell | Microsoft Learn* [online]. 2023 [cit. 2023-10-03]. Dostupné z: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_signing?view=powershell-5.1
16. *About If - PowerShell | Microsoft Learn* [online]. 2023 [cit. 2023-10-30]. Dostupné z: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_if?view=powershell-5.1
17. *About Switch - PowerShell | Microsoft Learn* [online]. 2023 [cit. 2023-10-26]. Dostupné z: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_switch?view=powershell-5.1
18. WELTNER, Dr. Tobias Weltner. Welcome to powershell.one!. *Powershell.one* [online]. 2020 [cit. 2023-10-29]. Dostupné z: https://powershell.one/wmi/remote-access

19. *Windows Server 2012 R2 Inside Out: Active Directory Architecture | Microsoft Press Store* [online]. 2014 [cit. 2023-11-05]. Dostupné z: https://www.microsoftpressstore.com/articles/article.aspx?p=2217264
20. What Is DNS in Active Directory? *Netwrix* [online]. 2017 [cit. 2023-11-11]. Dostupné z: https://blog.netwrix.com/2017/02/17/dns-in-active-directory
21. What Is a Domain Controller? *Techtarget.com* [online]. 2021 [cit. 2023-11-11]. Dostupné z: https://www.techtarget.com/searchwindowsserver/definition/domain-controller
22. What Is Group Policy in Active Directory. *NinjaOne* [online]. 2023 [cit. 2023-11-26]. Dostupné z: https://www.ninjaone.com/blog/what-is-group-policy-in-active-directory
23. KANISOVÁ, Hana a MÜLLER, Miroslav. *UML srozumitelně*. 2. aktualiz. vyd. Brno: Computer Press, 2006. ISBN 8025110834
24. FOWLER, Martin. *Destilované UML*. Knihovna programátora (Grada). Praha: Grada, 2009. ISBN 9788024720623
25. *About the Business Process Model And Notation Specification Version 2.0* [online]. 2010 [cit. 2023-10-19]. Dostupné z: https://www.omg.org/spec/BPMN/2.0
26. KRAUSE, Jordan. *Mastering Windows Server 2019: The Complete Guide for IT Professionals to Install and Manage Windows Server 2019 and Deploy New Capabilities*. 2. Birmingham: Packt Publishing, Limited, 2019. ISBN 9781789804539
27. O'LEARY, Mike. *Cyber Operations: Building, Defending, a Attacking Modern Computer Networks*. 2nd edition. United States: APress, 2019. ISBN 9781484242933.

8 Seznam obrázků, tabulek, grafů, zdrojových kódů a zkratek

Seznam obrázků

Seznam tabulek

Seznam grafů

Seznam ukázek zdrojových kódů

Seznam použitých zkratek

CmdLet	Nativní příkazy PowerShelllu
IPCONFIG	Příkaz pro zobrazení síťového nastavení systému Windows
UAC	Kontrola uživatelských účtů
DCOM	Distributed Component Object Model
RPC	Remote Procedure Call
WinRM	Windows Remote Management
AD	Adresářová služba Active Directory
LDAP	Protokol pro ukládání a přístup k datům na adresářovém serveru
MAPI	Protokol pro vzájemnou výměnu zpráv obvykle používané ke komunikaci se server Microsoft Exchange
NTFS	Souborový systém pro Windows
DNS	Systém pro překlad názvů domén na IP adresy
FQDN	Plně kvalifikovaný název domény
IP	Internetový protokol
OU	Organizační jednotka ve struktuře Active Directory
GPO	Skupinová politika ve struktuře Active Directory
AGPM	Pokročilá správa skupinových politik GPO
ADUC	Konzole pro správu uživatelů a počítačů ve struktuře Active Directory
RSAT	Nástroje pro vzdálenou správu serveru
RAW	Protokol pro přímý tisk bez zpracování
LPR	Protokol pro řízení tisku v síti
MIT	Svobodná licence vytvořená na Massachusettském technologickém institutu

Přílohy

Příloha 1 – Nastavení doménové politiky – globální

Příloha 2 – Nastavení doménové politiky – rozšířená

Příloha 3 – Úprava nastavení Windows Server 2019 R2

Příloha 4 – Skript pro nastavení oprávnění na tiskárnách tiskového serveru

Příloha 5 – Zdrojový kód: Správa tiskáren

Příloha 1 – Nastavení doménové politiky – globální

1) Vytvořit nebo upravit politiku na nejvyšší úrovni OU, jež má program působit (OU s PC)
2) Ve vytvořené politice nastavit v sekci „Konfigurace počítače“:
   - **Zásady> Nastavení systému Windows> Nastavení zabezpečení> Systém souborů> zabezpečení na soubor**
     „*%AllUsersProfile%\IZS\TiskarnyGPO\SpravaTiskaren.ps1*“
     >odebrat „Users“ a po potvrzení vybrat ***„Nahradit existující oprávnění ke všem souborům a podsložkám dědičnými oprávněními“***
   - v **Předvolby> Nastavení systému Windows> Soubory> vytvořit soubor**. Zvolit akci „Aktualizovat“ a následně nastavit zdrojový soubor „*\\NazevServeru.FQDN\program$\SpravaTiskaren.ps1*“ a cílový soubor „*%ProgramData%\IZS\TiskarnyGPO\SpravaTiskaren.ps1*“
   - v ***Předvolby> Nastavení systému Windows> Složky*** vybrat kontextové nabídce „**Nová položka“> Složka**
   - Zvolit akci „**Vytvořit“** a do pole cesta vložit „%AllUsersProfile%\IZS\TiskarnyGPO_tiskove_ovladace“
3) Vytvořit naplánovanou úlohu pro zavedení tiskáren dle následujících kroků:
   a) ***Předvolby> Nastavení ovládacích panelů> Naplánované úlohy***:
      V prvním pořadí 1)
      - Pravým tlačítkem vybrat „Nová položka> Naplánovaná úloha (alespoň Windows 7)
      - Zvolit akci Odstranit.
      - Vložit název „PowerShell-SpravaTiskaren-Default“
      - V možnosti zabezpečení vložit účet „NT AUTHORITY\System“
      - Do záložky „Akce“ vložíme novou akci. Necháme výchozí „Spustit program“ a do políčka „Program či skript“ vložíme text Powershell. Do políčka „Přidat argumenty (volitelně) vložíme
        `-NoProfile "C:\ProgramData\IZS\TiskarnyGPO\SpravaTiskaren.ps1"`
        a potvrdíme OK.
      - dále v záložce „Společné“ doporučuji vložit popis: *„toto odstranění aktivovat pouze pro potřeby aktualizace úlohy, která se pouze vytváří*
        pozn.: v případě, znovu vynucení aktualizace je nutné tuto úlohu znovu nakopírovat na první pozici a původní odstranění odebrat“

- zaškrtnout „Použít jednou a nepoužívat znovu“ a „Cílení na úrovni položky“
- kliknout na „Cílení“
- v cílení vybereme v záložce „Nová položka“ a v nabídce „Shoda souborů“. V „Typ shody“ vybereme „Soubor existuje“ a do „Cesta“ vložíme „`%windir%\System32\Tasks\PowerShell-SpravaTiskaren-default`“
- potvrdíme OK.
- **Klikneme pravým tlačítkem na vytvořenou úlohu a dáme „Všechny úkoly> zakázat“.**

b) ***Předvolby> Nastavení ovládacích panelů> Naplánované úlohy***:

V druhém pořadí 2)

- Pravým tlačítkem vybrat „Nová položka> Naplánovaná úloha (alespoň Windows 7)
- Zvolit akci „Vytvořit“.
- Vložit název „PowerShell-SpravaTiskaren-Default“
- V možnosti zabezpečení vložit účet „NT AUTHORITY\System“ následně vybrat „Spustit nezávisle na přihlášení uživatelem“ (tato volba se může aktivovat až po znovu editování úlohy) a zaškrtnout „Spustit s nejvyššími oprávněními“.
- záložka „Aktivační událost“ budou vloženy 2 události:

  Při spuštění:

  - vložit novou aktivační událost tlačítkem „Nová“ a vybrat začátek úlohy „Při spuštění“
  - zaškrtnout „Zdržení úlohy“ a vybrat 30 sekund.
  - zaškrtnout „Aktivovat“ a „Povoleno“
  - potvrdit OK.

  Při události:

  - kliknout na tlačítko „Nová“, vybrat začátek úlohy „Při události“ a vybrat „Vlastní“.
  - kliknout na „Nový filtr událostí…“
  - záložce „XML“ zaškrtnout „Upravit dotaz ručně“ a vložit

```
<QueryList> <Query Id="0" Path="Microsoft-Windows-TaskScheduler/Operational"> <Select Path="Microsoft-Windows-TaskScheduler/Operational">*[System[Provider[@Name='Microsoft
```

```
-Windows-TaskScheduler'] and (EventID=106) and
TimeCreated[timediff(@SystemTime) &lt;= 600000]]] and
*[EventData[@Name='TaskRegisteredEvent'][Data[@Name='TaskName
']='\PowerShell-SpravaTiskaren']] </Select><Select
Path="Microsoft-Windows-TaskScheduler/Operational">
*[System[Provider[@Name='Microsoft-Windows-TaskScheduler']
and (EventID=106) and TimeCreated[timediff(@SystemTime) &lt;=
600000]]] and
*[EventData[@Name='TaskRegisteredEvent'][Data[@Name='TaskName
']='\PowerShell-SpravaTiskaren-
default']]</Select></Query></QueryList>
```

- potvrdit OK
- zaškrtnout „Zdržení úlohy“, „Aktivovat“ a „Povoleno“
- potvrdit OK
- vybrat záložku „Akce“
- vložit novou akci tlačítkem „Nová“
- nechat výchozí „Spustit program“ a do políčka „Program či skript“ vložit text Powershell.
- do políčka „Přidat argumenty (volitelně) vložit

```
-NoProfile "C:\ProgramData\IZS\TiskarnyGPO\SpravaTiskaren.ps1″
```

- potvrdit OK
- v záložce „Nastavení“ zaškrtnout:
  - *Povolit spouštění úlohy na požádání
  - *Při vynechání naplánovaného spuštění úlohy spustit co nejdříve
  - *Zastavit úlohu, pokud běží déle než: 1 hodina
- v záložce „Společné“ vybrat „Cílení na úrovni položky“ kde klikneme na „Cílení…“
- v nově otevřeném okně vybrat v nástrojové liště „Nová položka“ a v nabídce „Shoda souborů“.
- ve stejné liště vybrat „Možnosti položky“ a vybrat „Není“.
- v „Typ shody“ vybrat „Soubor existuje“ a do „Cesta“ vložíme *„%windir%\System32\Tasks\PowerShell-SpravaTiskaren“*
- potvrdit 2x OK.

Příloha 2 – Nastavení doménové politiky – rozšířená

1) Vytvořit a upravit politiku přiřazenou na konkrétní OU, jež má program působit (OU s PC)
2) V politice vytvořit naplánovanou úlohu pro zavedení tiskáren dle následujících kroků:
   a) ***Předvolby> Nastavení ovládacích panelů> Naplánované úlohy***:
   V prvním pořadí 1)

- pravým tlačítkem vybrat „Nová položka> Naplánovaná úloha (alespoň Windows 7)
- zvolit akci Odstranit
- vložit název „PowerShell-SpravaTiskaren-Default“
- v možnosti zabezpečení vložit účet „System“
- do záložky „Akce“ vložit novou akci. Nechat výchozí „Spustit program“ a do políčka „Program či skript“ vložit text Powershell.
- překliknout na záložku „Společné“ a zaškrtnout „Cílení na úrovni položky“
- kliknout na aktivované tlačítko „Cílení…“
- v nově otevřeném okně vybrat v nástrojové liště „Nová položka“ a v nabídce „Shoda souborů“
- v nově otevřeném okně na nástrojové liště vybrat „Možnosti položky“ a vybrat „Je“
- z rozevírací nabídky „Typ shody“ vybereme „Soubor existuje“ a do „Cesta“ vložíme „%windir%\System32\Tasks\PowerShell-SpravaTiskaren-Default“. Po nastavení této úlohy potvrdíme 2x OK

c) ***Předvolby> Nastavení ovládacích panelů> Naplánované úlohy***:

V druhém pořadí 2)

Pravým tlačítkem vybrat „Nová položka> Naplánovaná úloha (alespoň Windows 7)

- zvolit akci „Vytvořit“
- vložit název „PowerShell-SpravaTiskaren“
- v možnosti zabezpečení vložit účet „NT AUTHORITY\System“ následně vybrat „Spustit nezávisle na přihlášení uživatelem“ (tato volba se může aktivovat až po

znovu editování úlohy, kde může být potřeba znovu vložit „NT AUTHORITY\System“) a zaškrtnout „Spustit s nejvyššími oprávněními“

- v záložce „Aktivační událost“ vložit novou aktivační událost tlačítkem „Nová“ a vybrat začátek úlohy „Při vytvoření či úpravě úlohy“.
- zaškrtnout „Zdržení úlohy“ a vybrat „1 minuta“
- zaškrtnout „Aktivovat“ a „Povoleno“
- potvrdit OK
- do záložky „Akce“ vložíme novou akci. Necháme výchozí „Spustit program“ a do políčka „Program či skript“ vložíme text Powershell
- do políčka „Přidat argumenty (volitelně) vložíme vše, co je mezi hranatými závorkami. Název O00-000006-J je pouze demonstrativní název, proto je nutné napsat část názvu tiskárny (pro nainstalování se shodným názvem) nebo celý název pro konkrétní tiskárnu

  [-NoProfile "C:\ProgramData\IZS\TiskarnyGPO\SpravaTiskaren.ps1 -tiskarnyNavic 'O00-000006-J'"]

  - potvrdit OK.
  - v záložce „Nastavení“ zaškrtneme:
    *Povolit spouštění úlohy na požádání
    *Při vynechání naplánovaného spuštění úlohy spustit co nejdříve
    *Zastavit úlohu, pokud běží déle než: 1 hodina
  - v záložce „Společné“ zaškrtnout „Odstranit tuto položku, není-li již používána“
  - potvrdit OK.

Příloha 3 – Úprava nastavení Windows Server 2019 R2

1) **Úprava registru pro zabalení ovladačů:** Na tiskovém serveru v konzoli pro správu tisku v sekci "ovladače" je třeba ověřit, zda v sloupci "Zabaleno" je hodnota "nepravda". Pokud takový ovladač existuje, měla by být hodnota v registru v atributu "*PrinterDriverAttributes*" změněna na liché číslo, ideálně zvýšením čísla o 1. Tímto krokem dojde k automatickému zabalení ovladače do archivu s příponou CAB. Cesta k registru je:

   Pro 64bitové ovladače:

   „HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Print\Environments\Windows x64\Drivers\Version-3\název ovladače"

   Pro 32bitové ovladače:

   „HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Print\Environments\Windows NT x86\Drivers\Version-3\název ovladače"

2) **Nastavení lokální skupiny pro vzdálený přístup k tiskovém serveru pomocí Powershell:**
   - Ve „Správa počítače > Místní uživatelé a skupiny > Skupiny", vytvořte novou skupinu (přes pravé kontextové tlačítko) s názvem „SpravaTiskuSkript".
   - Přidejte do skupiny jakoukoliv doménovou skupinu, které má být povolen vzdálený přístup ke čtení tiskáren ze správy tisků. V tomto případě se povoluje skupina „Domain Computers".

3) **Nastavení přístupu pro Řízení služby WMI:**
   - Ve „Správa počítače > Služby a aplikace > Řízení služby WMI" klikněte na „Vlastnosti".
   - V záložce „Zabezpečení", expandujte strom „Root", vyberte „CIMV2", a klikněte na „Zabezpečení".
   - Přidejte skupinu „SpravaTiskuSkript", zaškrtněte v sloupci „Povolit" oprávnění „Povolovat vzdálený přístup" a odškrtněte „Povolit účet".

4) **Změna DCOM oprávnění:**
   - Spusťte „Služba komponent“ příkazem „dcomcnfg“.
   - V otevřené konzoli expandujte „Služba komponent > Počítače“. Pravým tlačítkem vyvoláte kontextovou nabídku a zvolte „Vlastnosti“.
   - Přejděte na záložku „Zabezpečení COM“, v sekci „Spouštěcí a aktivační oprávnění“ klikněte na „Upravit omezení...“.
   - Přidejte skupinu „SpravaTiskuSkript“, zaškrtněte „Vzdálená aktivace“ a odškrtněte „Místní spuštění“.

5) **Nastavení Správa tisku:**
   - Spusťte konzoli „Správa tisku“, expandujte „Tiskové servery“. Na požadovaném serveru klikněte pravým tlačítkem a zvolte „Vlastnosti“.
   - Klikněte na záložku „Zabezpečení“, poté „Upřesnit“.
     a) ***Pro Správa serveru:***
        - Klikněte na „Přidat“, poté „Vybrat objekt zabezpečení“.
        - Vložte skupinu „SpravaTiskuSkript“, změňte umístění z domény na název serveru a potvrďte.
        - Změňte „Platí pro“ na „Pouze tento server“, odškrtněte „Tisk“, zaškrtněte „Správa serveru“ a potvrďte.
     b) ***Pro Tisk:***
        - Klikněte na „Přidat“, poté „Vybrat objekt zabezpečení“.
        - Vložte skupinu „SpravaTiskuSkript“, změňte umístění z domény na název serveru a potvrďte.
        - Změňte „Platí pro“ na „Pouze tiskárny“, odškrtněte „Zobrazit server“ a potvrďte třikrát OK.

6) **Hromadné přidání oprávnění na tiskárny**

   Pro hromadné přidání nově vytvořené lokální skupiny dle bodu 2) této přílohy můžete použít vytvořený skript v příloze 4 – Skript pro nastavení oprávnění na tiskárnách tiskového serveru, který provede hromadné přidání požadované skupiny.

Příloha 4 – Skript pro nastavení oprávnění na tiskárnách tiskového serveru

```
$lokalniNazevSkupiny = "SpravaTiskuSkript"
$hodnotaSID = $null
$obj = New-Object System.Security.Principal.NTAccount($lokalniNazevSkupiny)
try {
    $SID = $obj.Translate([System.Security.Principal.SecurityIdentifier])
}
catch {
    Write-Warning ('Lokální název skupiny "' + $lokalniNazevSkupiny + '"
nebyl nalezen!')
    exit
}
$hodnotaSID = $SID.Value
write-host "Načítám tiskárny..." -NoNewline
[array]$printers = Get-Printer -Full | select name, PermissionSDDL
$uniqe = (get-date).ToFileTime()
$printers | Export-Csv -Delimiter ";" -Path
"$PSScriptRoot\printers_PermissionSDDL_backup_$uniqe.csv" -NoTypeInformation
$count = $printers.Count
$countdown = $count
write-host $count -ForegroundColor Green
foreach ($printer in $printers) {
    $printerName = $printer.name
    write-host ("{0,3} z {2} Přidávám oprávnění do tiskárny {1}..." -f
$countdown, $printerName, $count) -NoNewline
    $addPerm = "(A;;SWSDRCWDWO;;;$sid)"
    $perm = $printer.PermissionSDDL
    if (!($perm -match $addPerm)) {
        try {
            $perm += $addPerm
            Set-Printer $printerName -PermissionSDDL $perm -ErrorAction Stop
            write-host "OK" -ForegroundColor Green
        }
        catch {
            Write-Host "ERROR" -ForegroundColor Red
        }
    }
    else {
        write-host "oprávnění již existuje" -ForegroundColor Yellow
    }
    $countdown -= 1
}
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
#Requires -Version 5.1
################################################################################
#
# NAME: Správa tiskáren
#
# AUTHOR:  ILAVSKÝ Lukáš
#
# COMMENT: Přidává/Odebírá lokální tiskárny dle konfigurace z tiskového serveru.
################################################################################


Param(
    [string]$tiskarnyNavic,
    [string]$tiskovyServer = "OOO-PRINT01.IZS.CZ",
    [string]$nazevSdileneSlozkySprogramem = "program$", #sdilená složka musí být na
tiskovém serveru
    [string]$ignorovatTiskoveOvladace = "Microsoft,PDF,XPS", #vynecha ovladače obsahující
vzor
    [string]$ignorovatPorty = "USB,LPT,DOT4,\\\\,10\.", # vynechá tiskárny s porty
obsahující vzor
    [string]$ignorovatNazvyTiskaren = "\\\\", # neodstraní tiskárny s názvem obsahující
vzor (\\\\ tzn. první znak \ je escape znakem v reg. jazyce, proto musí být 2x pro
vyhodnocení znaku \, tzn pro vyhodnocení cesty UNC \\ musí být \\\\
    [boolean]$nastavovatSecurity = $true , #$true = bude načítat a nastavovat zabezpečení
tiskárny PermissionSDDL
    [string]$cestaPracovnihoAdresareprogramu = $env:ProgramData + "\IZS\TiskarnyGPO", #
lokální pracovní cesta programu + uložení výstup logu programu a ovladače tiskáren
    [boolean]$vynucovatOdebraniOvladacu = 0, #1 bude se pokoušet odebírat uzamčené ovladače
    [boolean]$ZobrazitSpousteciArgumenty = 0, #1 zobrazí spouštěcí argumenty v konzoli i v
HTML
    [boolean]$logujOperace = $true, #$false = zaloguje pouze výsledek konzole programu v
HTML, $true = loguje každou funkci a zapisuje do html
    [string]$odebratRegistrTiskovychServeru = "OOO-PRINT", #po odebrání připojování
tiskáren z GPO zůstávají v některých případech servery a tiskárny v registrech, které pak
brání odebrání tiskových ovladačů a v některých případech dochází k odesílání tisku na
shodný port
    [boolean]$zapisLog = $true, #$true = loguje průbeh programu do textoveho souboru
    [string]$nazevVystupu = "Tiskarny-GPO-LOG.html",
    [boolean]$spustenJakoNovaInstance = 0 #slouží pouze pro program, kerý si tím nastaví
pouze běh druhé instance programu
)

#------------------------------níže nic neměnit------------------------------
function subInicializacePracovnihoProstredi {
    $script:velikostOknaRozhrani = 128 # to je maximum pro systemovy účet
    $script:verzeProgramu = "1.3.6"
    $script:nazevProgramu = $script:MyInvocation.MyCommand.name
    $script:cestaProgram = $script:MyInvocation.MyCommand.Path
    $script:datumZacatek = Get-Date
    $errorActionPreference = "Stop" # pro zastaveni jakekoliv chyby vložit stop , vychozí
je continue
    [string]$script:nazevPC = $env:computername # aktuální název pc, kde běží program
    $script:posledniDelkaSloupce = @{}
    $script:posledniDelkaSloupce['sloupec1'] = 0
    $script:posledniDelkaSloupce['sloupec2'] = 0
    $script:posledniDelkaSloupce['sloupec3'] = 0
    $script:posledniDelkaSloupce['sloupec4'] = 0
    $script:posledniDelkaSloupce['sloupec5'] = 0
    $script:posledniTextSloupce = @{}
    $script:posledniTextSloupce['sloupec1'] = ""
    $script:posledniTextSloupce['sloupec2'] = ""
    $script:posledniTextSloupce['sloupec3'] = ""
    $script:posledniTextSloupce['sloupec4'] = ""
    $script:posledniTextSloupce['sloupec5'] = ""
    subNastavitPromenneKdeBeziProgram
    $script:cestaLogSouboru = $cestaPracovnihoAdresareprogramu +
"\lastrun_$env:computername.log"
    if (Test-Path $script:cestaLogSouboru) { Remove-Item -Path $script:cestaLogSouboru -
Confirm:$false -Force -ErrorAction SilentlyContinue }
    subZapsatLogDoTxt "----------------- ZAČÁTEK -----------------"
    subZobrazitSpousteciArgumenty
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
    subSpustitProgramJakoSpravce
    subKontrolaVerzeProgramu
    subNastavitVelikostKonzole
    subOpravneniSpusteniUzivatelem -povolit $true
    $script:oddeleni = fncVratitNazevOU -nazevPC $script:nazevPC # slouží k filtrování
tiskáren pro použití dle OU + startovací argument tiskarnyNavic přidá tiskárnu mimo OU
    if ($tiskarnyNavic.Length -gt 0) {
        #  pokud existuje argument tiskarnyNavic, tak je přidáme do proměné
$script:oddeleni jako další výraz pro regex
        $tiskarnyNavic = $tiskarnyNavic -replace ",", "|"
        $script:oddeleni += "|" + $tiskarnyNavic
    }
    if ($script:ignorovatPorty.Length -ge 1) {
        $script:ignorovatPorty = $script:ignorovatPorty -replace ",", "|"
    }

    if ($script:ignorovatTiskoveOvladace.Length -ge 1) {
        $script:ignorovatTiskoveOvladace = $ignorovatTiskoveOvladace -replace ",", "|"
    }

    if ($script:ignorovatNazvyTiskaren.Length -ge 1) {
        $script:ignorovatNazvyTiskaren = $script:ignorovatNazvyTiskaren -replace ",", "|"
    }

}
function subZobrazitSpousteciArgumenty {
    if ($ZobrazitSpousteciArgumenty) {
        write-host "tiskarnyNavic:                    " $tiskarnyNavic
        write-host "tiskovyServer:                    " $tiskovyServer
        write-host "nazevSdileneSlozkySprogramem:     " $nazevSdileneSlozkySprogramem
        write-host "ignorovatTiskoveOvladace:         " $ignorovatTiskoveOvladace
        write-host "ignorovatPorty:                   " $ignorovatPorty
        write-host "ignorovatNazvyTiskaren:           " $ignorovatNazvyTiskaren
        write-host "nastavovatSecurity:               " $nastavovatSecurity
        write-host "cestaPracovnihoAdresareprogramu:  " $cestaPracovnihoAdresareprogramu
        write-host "vynucovatOdebraniOvladacu:        " $vynucovatOdebraniOvladacu
        write-host "ZobrazitSpousteciArgumenty:       " $ZobrazitSpousteciArgumenty
        write-host "logujOperace:                     " $logujOperace
        #write-host "uživatel: " $env:USERNAME
    }
}

function subZapsatLogDoTxt {
    Param (
        [string]$text,
        [string]$plnaCestaKsouboru = $script:cestaLogSouboru
    )
    if ($zapisLog) {
        $cas = (Get-Date).toString("yyyy/MM/dd HH:mm:ss")
        $textProZapis = "$cas $text"
        try {
            Add-content $plnaCestaKsouboru -value $textProZapis -ErrorAction Stop
        }
        catch {
            Write-Warning "Log se nepodařil zapsat do $plnaCestaKsouboru"
        }
    }
}
function subSpustitProgramJakoSpravce {
    Param (
        $uplnaCestaKprogramu = $script:cestaProgram
    )
    $aktualniID = [System.Security.Principal.WindowsIdentity]::GetCurrent()
    $WindowsPrincipal = New-Object System.Security.Principal.WindowsPrincipal($aktualniID)
    $adminRole = [System.Security.Principal.WindowsBuiltInRole]::Administrator
    $argumenty = $script:MyInvocation.BoundParameters # zachytíme parametry zadané při
spuštění programu
    if (!($WindowsPrincipal.IsInRole($adminRole))) {
        # pokud program není spuštěn s právy administrátora, tak si o ně řekneme
        if ($host.name -match "ISE|Studio") { Write-Warning "program není spuštěn jako
správce!!!"; return }
        if ($spustenJakoNovaInstance) {
            # pokud je $true, znamená to, že byl již proces znovu spuštěn, ale opět bez
práv administrátora, nemůžeme pokračovat
            Write-Warning "Program není spuštěn jako správce!!!"
            timeout /10
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
            exit
        }
        subNapsatZpravuDoKonzole -sloupec1 'Spouštím jako administrator' -barva Yellow
        $novyProces = New-Object System.Diagnostics.ProcessStartInfo "PowerShell"
        foreach ($klic in $argumenty.keys) {
            $hodnota = (get-variable $klic).Value -replace " ", "," # v případě více hodnot
parametru BoundParameters rozděluje hodnoty mezerou, tím vrátíme oddělovač ,
            switch ($hodnota) {
                False { $hodnota = 0 }
                True { $hodnota = 1 }
                0 { $hodnota = $false }
                1 { $hodnota = $true }
            }
            $parametry += " -" + $klic + " '" + $hodnota + "'"
        }
        if ($parametry.length -gt 0) {
            $parametry = $parametry.replace("'0'", 0).replace("'1'", 1) #v případě
boleovské hodnoty 0 nebo 1 odeberem apostrofy, jinak se nevyhodnotí argument $true nebo
$false
        }
        $parametry += ' -spustenJakoNovaInstance 1' # přidáváme pojistku proti opakovanému
spuštění, kdyby spuštěná nová instance nebyla správy administratora
        subNapsatZpravuDoKonzole -sloupec1 'parametry:' -sloupec2 $uplnaCestaKprogramu -
sloupec3 $parametry
        $novyProces.Arguments = $uplnaCestaKprogramu + $parametry
        $novyProces.Verb = "runas";
        try {
            [System.Diagnostics.Process]::Start($novyProces) # vyvoláme nový proces, pokud
je UAC aktivní, bude vyvolán přihlašovací dialog
        }
        catch {
            fncVratitDuvodChyby -psitem $PSItem
            timeout /t 10
        }
        exit
    }
}
function fncSpustitProgramVnoveInstanci {
    Param(
        $uplnaCestaKprogramu = $script:cestaProgram,
        $tiskarnyNavicHodnoty = $tiskarnyNavic
    )
    if ($host.name -match "ISE|Studio") { return }
    $novyProces = New-Object System.Diagnostics.ProcessStartInfo "PowerShell"
    $parametry = " -tiskarnyNavic " + $tiskarnyNavicHodnoty # složíme parametry
    if ($tiskarnyNavicHodnoty.length -gt 0) {
        # pokud existuje hodnota, pak spustíme novy proces s parametry či bez
        $novyProces.Arguments = $uplnaCestaKprogramu + $parametry # novy proces s parametry
    }
    else {
        $novyProces.Arguments = "& '" + $uplnaCestaKprogramu + "'" # nový proces bez
parametru
    }
    [System.Diagnostics.Process]::Start($novyProces);
    exit
}
function subKontrolaVerzeProgramu {
    Param(
        $cestaProgramuVzdalenySoubor = '\\' + $tiskovyServer + '\' +
$nazevSdileneSlozkySprogramem + '\' + $script:nazevProgramu, # složíme cestu z proměných
        $cestaProgramuLokalniSoubor = $script:cestaProgram,
        $vzorRegEx = '\s{3}\$script:verzeProgramu\ = "\d{1,2}\.\d{1,2}\.?\d{0,2}"$' #
regulární výraz pro vyhledání přesného řetězce např.
MezeraMezeraMezerea$script:verzeProgramu = "1.2" nebo
MezeraMezeraMezerea$script:verzeProgramu = "10.99.99"
    )
    if (!($script:spustenMimoPracovniAdresar)) {
        subZapsatLogDoTxt "start subKontrolaVerzeProgramu"
        if (test-path $cestaProgramuVzdalenySoubor) {
            # ověříme existenci vzdáleného souboru
            [string]$verzeProgramuVzdaleny = select-string -Pattern $vzorRegEx -Path
$cestaProgramuVzdalenySoubor | % { (($PSitem) -replace '.*=\s+') -replace '"' }  # zjištění
verze programu vzdálený
        }
        if ($verzeProgramuVzdaleny.Length -gt 0) {
            # pokud byla nalezena verze vzdaleneho souboru
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
            if ($script:verzeProgramu -ne $verzeProgramuVzdaleny) {
                # pokud není verze lokalni a vzdalena stejna
                if (fncAktualizujProgram) {
                    fncSpustitProgramVnoveInstanci # vynutíme znovu spuštění nové verze
                }
                else {
                    Write-Warning "Aktualizace se nezdařilo: " $(fncVratitDuvodChyby -
psitem $psitem)
                }
            }
        }
        else {
            Write-Warning 'Vzdálená verze programu nenalezena'
        }
    }
    subNapsatZpravuDoKonzole -sloupec1 "Spouštím program z $script:cestaProgram" -barva
Yellow
    subNapsatZpravuDoKonzole -sloupec1 "Verze programu: " -barva Gray -
zarovnatNaPredchoziSloupce:$false
    subNapsatZpravuDoKonzole -sloupec2 "$script:verzeProgramu" -nahradSloupec sloupec2 -
barva Yellow

}
function fncAktualizujProgram {
    Param(
        $zdroj = '\\' + $tiskovyServer + '\' + $nazevSdileneSlozkySprogramem + '\' +
$script:nazevProgramu, # složíme cestu z proměných
        $cil = $script:cestaProgram
    )
    try {
        $null = Copy-Item -Path $cestaProgramuVzdalenySoubor -Destination
$cestaPracovnihoAdresareprogramu -ErrorAction Stop  #aktualizujeme lokalni verzi a
potlačíme případný výstup
        return $true
    }
    catch {
        return $false
    }
}
function subNastavitVelikostKonzole {
    Param(
        $pozadovanaVyska = 60,
        $pozadovanaSirka = 128
    )
    if ($host.name -match "ISE|Studio") { return }
    subZapsatLogDoTxt "start subNastavitVelikostKonzole"
    $uzivatelskeRozhrani = (Get-Host).UI.RawUI
    # Nastaví barvu pozadí
    $uzivatelskeRozhrani.BackgroundColor = [System.ConsoleColor]::DarkBlue
    # Zjistí maximální rozměry okna
    $maxVyska = $uzivatelskeRozhrani.MaxPhysicalWindowSize.Height
    $maxSirka = $uzivatelskeRozhrani.MaxPhysicalWindowSize.Width
    # Upraví požadované rozměry, pokud přesahují maximální rozměry
    $vyskaOkna = [math]::Min($pozadovanaVyska, $maxVyska)
    $sirkaOkna = [math]::Min($pozadovanaSirka, $maxSirka)
    # Zvětší buffer na maximální hodnoty
    $uzivatelskeRozhrani.BufferSize = New-Object
System.Management.Automation.Host.Size($maxSirka, 32765)
    # Zmenší okno na požadovanou velikost
    $uzivatelskeRozhrani.WindowSize = New-Object
System.Management.Automation.Host.Size($sirkaOkna, $vyskaOkna)
    # Zmenší buffer zpět na požadovanou šířku
    $uzivatelskeRozhrani.BufferSize = New-Object
System.Management.Automation.Host.Size($sirkaOkna, 32765)
    #write-host "DEBUG: Velikost:" $uzivatelskeRozhrani.WindowSize
    $script:velikostOknaRozhrani = $uzivatelskeRozhrani.WindowSize
    subZapsatLogDoTxt " exit subNastavitVelikostKonzole"
}
function subObsahKonzoleDoHtml {
    #MIT License
    #Copyright (c) 2022 Mattias Karlsson
    #https://gist.github.com/devlead/2e006ec4a7e134cf38c0#file-license
    #Add AutoRefresh - Lukas Ilavsky

    Param(
        [Parameter(Mandatory = $false)]
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
        [ValidateScript({ ![String]::IsNullOrWhiteSpace($_) })]
        [string] $FilePath = $cestaPracovnihoAdresareprogramu + "\$nazevVystupu",

        [ValidateSet("Unicode", "UTF7", "UTF8", "UTF32", "ASCII", "BigEndianUnicode")]
        [string] $Encoding = "UTF8",

        [int] $Last = 0,

        [switch]$SkipLast
    )
    if ($host.name -match "ISE|Studio") { Write-Warning "Snímek výsledku se ukládá pouze
při spuštění z konzole PowerShell!!!"; return }
    Function ToHex ([System.ConsoleColor] $color) {
        switch ($color) {
            "Black" { "#000000" }
            "DarkBlue" { "#012456" }
            "DarkGreen" { "#005711" }
            "DarkCyan" { "#007680" }
            "DarkRed" { "#6F0711" }
            "DarkMagenta" { "#6F0780" }
            "DarkYellow" { "#888F11" }
            "Gray" { "#878E98" }
            "DarkGray" { "#666D77" }
            "Blue" { "#0000FF" }
            "Green" { "#00FF00" }
            "Cyan" { "#00FFFF" }
            "Red" { "#FF0000" }
            "Magenta" { "#FF00FF" }
            "Yellow" { "#FFFF00" }
            "White" { "#FFFFFF" }
            default { throw "Unknown color: $color" }
        }
    }

    if ($host.Name -ne "ConsoleHost") {
        throw "Console host $($host.Name) not supported";
    }

    $state = @{
        Width                 = $host.ui.rawui.BufferSize.Width
        Height                = if ($SkipLast.IsPresent) { $host.ui.rawui.CursorPosition.Y
- 1 } else { $host.ui.rawui.CursorPosition.Y }
        Rect                  = (new-object System.Management.Automation.Host.Rectangle 0,
0, ($host.ui.rawui.BufferSize.Width - 1), $host.ui.rawui.CursorPosition.Y)
        DefaultBackgroundColor = [System.ConsoleColor]::DarkMagenta
        DefaultForegroundColor = [System.ConsoleColor]::White
    }

    $buffer = $host.ui.rawui.GetBufferContents($state.Rect)
    $currentForegroundColor = $state.DefaultForegroundColor
    $currentBackgroundColor = $state.DefaultBackgroundColor
    $outputBuilder = new-object System.Text.StringBuilder
    $inSpan = $false;

    [void]$outputBuilder.AppendLine("<html>")
    [void]$outputBuilder.AppendLine("<head>")
    [void]$outputBuilder.AppendLine("<meta http-equiv=`"content-type`" content=`"text/html;
charset=UTF-8`"><meta http-equiv=`"refresh`" content=`"5`">")
    [void]$outputBuilder.AppendLine("</head>")
    [void]$outputBuilder.AppendLine("<body>")
    [void]$outputBuilder.Append("<pre style=`"display: inline-block;padding:2px;border:2px
solid black;font-size:10px;font-family: Consolas, 'Lucida Console',
        Monaco, monospace;color:$(ToHex([System.ConsoleColor]::White));background-
color:$(ToHex([System.ConsoleColor]::DarkBlue))`">"
    )
    $firstRow = 0
    if ($Last -gt 0 -and ($state.Height - $Last) -gt 0) {
        $firstRow = $state.Height - $Last;
    }
    for ($row = $firstRow; $row -lt $state.Height; $row++) {
        for ($col = 0; $col -lt $state.Width; $col++) {
            $cell = $buffer[$row, $col]
            if ($currentForegroundColor -ne $cell.ForegroundColor -or
$currentBackgroundColor -ne $cell.BackgroundColor) {
                $currentForegroundColor = $cell.ForegroundColor
                $currentBackgroundColor = $cell.BackgroundColor
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
                if ($inSpan) {
                    [void]$outputBuilder.Append("</span>")
                    $inSpan = $false
                }
                if (($currentForegroundColor -ne $state.DefaultForegroundColor -and
$currentForegroundColor -ne [System.ConsoleColor]::DarkYellow) -or $currentBackgroundColor
-ne $state.DefaultBackgroundColor) {
                    [void]$outputBuilder.Append("<span style=`"")
                    if ($currentForegroundColor -ne $state.DefaultForegroundColor -and
$currentForegroundColor -ne [System.ConsoleColor]::DarkYellow) {

[void]$outputBuilder.Append("color:$(ToHex($currentForegroundColor))")
                    }
                    if ($currentBackgroundColor -ne $state.DefaultBackgroundColor) {
                        [void]$outputBuilder.Append(";background-
color:$(ToHex($currentBackgroundColor))")
                    }
                    [void]$outputBuilder.Append("`">")
                    $inSpan = $true
                }
            }
            switch ($cell.Character) {
                "<" { [void]$outputBuilder.Append("&lt;") }
                ">" { [void]$outputBuilder.Append("&gt;") }
                default { [void]$outputBuilder.Append($cell.Character) }
            }
        }
        for ($index = $outputBuilder.Length - 1; $outputBuilder.Length -gt 0 -and
[String]::IsNullOrWhiteSpace($outputBuilder[$index]); $index--) {
            [void]$outputBuilder.Remove($index, 1);
        }
        if ($inSpan) {
            [void]$outputBuilder.Append("</span>")
            $inSpan = $false;
        }
        [void]$outputBuilder.AppendLine()
    }
    [void]$outputBuilder.AppendLine("</pre>")
    [void]$outputBuilder.AppendLine("</body>")
    [void]$outputBuilder.AppendLine("</html>")
    $outputBuilder.ToString() | Out-File -FilePath $FilePath -Encoding $Encoding -Force
}
function subPripojeniKserveru {
    Param (
        $nazevServeru = $tiskovyServer,
        $protokol = "Dcom"
    )
    subZapsatLogDoTxt "start subPripojeniKserveru"
    $CimOption = New-CimSessionOption -Protocol $protokol
    try {
        subNapsatZpravuDoKonzole -sloupec1 'Připojování na tiskový server:' -sloupec2
$nazevServeru -zarovnatNaPredchoziSloupce:$true
        if ($logujOperace) { subObsahKonzoleDoHtml }
        $script:CimSessionVzdaleny = New-CimSession -ComputerName $nazevServeru -
SessionOption $CimOption -ErrorAction Stop  #
        $script:CimSessionLokalni = New-CimSession -ComputerName $env:computername -
SessionOption $CimOption -ErrorAction Stop
        subNapsatZpravuDoKonzole -sloupec3 "připojeno" -nahradSloupec sloupec3 -barva Green
    }
    catch {
        subNapsatZpravuDoKonzole -stavZpravy chyba -nahradSloupec sloupec3
        subZapsatLogDoTxt "      PripojeniKserveru - chyba: $(fncVratitDuvodChyby -psitem
$psitem)"
        subUkonceniProgramu
    }
    if ($logujOperace) { subObsahKonzoleDoHtml }
    subZapsatLogDoTxt " exit subPripojeniKserveru"
}
function fncStopky {
    $datumKonec = Get-Date
    $time = New-TimeSpan -Start $datumZacatek -End $datumKonec
    $script:datumZacatek = Get-Date
    return $time.ToString("hh\:mm\:ss")
}
function fncVratitNazevOU {
    Param(
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
        [string]$nazevPC = $env:computername
    )
    subZapsatLogDoTxt "start fncVratitNazevOU"
    $DirectorySearcher = New-Object System.DirectoryServices.DirectorySearcher # vytvoříme
konstruktor DirectorySearcher
    $DirectorySearcher.Filter = '(&(name=' + $nazevPC + ')(objectClass=computer))' #filtr,
co chceme hledat
    $DirectorySearcher.SearchScope = 'Subtree' # nastavíme rozsah hledání na celý podstrom
    $nalezenePC = $DirectorySearcher.FindAll() #spustíme heldání a vrácený výsledek uložíme
do proměné
    if (($nalezenePC | measure).count -gt 0) {
        $distinguishedName = $nalezenePC.Properties.Item('distinguishedName') -split "OU="
# rozdělíme identifikátor dle vzoru OU=
        $distinguishedName = $distinguishedName[1] -replace ",", "" # očistíme výběr od
čárky
        if ($distinguishedName.Length -gt 10) {
            # v případě delšího názvu zkrátíme
            $osetrenyNazevPC = $distinguishedName.Substring(0, 10)
        }
        else {
            $osetrenyNazevPC = $distinguishedName
        }
    }
    else {
        $osetrenyNazevPC = "PC_nenalezen_v_IZS"
    }
    subZapsatLogDoTxt " exit fncVratitNazevOU"
    return $osetrenyNazevPC
}
function fncVratitOvladaceZrepozitare {
    subZapsatLogDoTxt "start fncVratitOvladaceZrepozitare"
    $ovladace = Get-WindowsDriver -online | ? { $_.classname -match "print" -and
$_.providername -notmatch "Microsoft" }
    subZapsatLogDoTxt " exit fncVratitOvladaceZrepozitare"
    $a = $ovladace.GetType()
    return $ovladace
}
function fncVratitBitovouVerziOS {
    if ([System.IntPtr]::Size -eq 4) {
        $osArch = "x86"
    }
    else {
        $osArch = "x64"
    }
    return $osArch
}
function fncInstalaceOvladace {
    Param(
        [Parameter(Mandatory = $true)]
        [string]$nazevSouboruCAB
    )
    subZapsatLogDoTxt "start fncInstalaceOvladace"
    #instalace ovladače do repozitáře
    $architekturaOvladace = (fncVratitBitovouVerziOS).replace("x86", "W32X86")
    $sitovaCestaSouborCab = "\\$tiskovyServer\print$\" + $architekturaOvladace +
"\PCC\$nazevSouboruCAB"
    $lokalniCestaAdresarCab = ($cestaPracovnihoAdresareprogramu + "\_tiskove_ovladace\" +
$nazevSouboruCAB).Replace(".cab", "")
    $lokalniCestaSouborCab = $lokalniCestaAdresarCab + "\" + $nazevSouboruCAB
    $infNazev = ($nazevSouboruCAB -split "_")[0]
    $infpath = $lokalniCestaAdresarCab + "\" + $infNazev
    if (!(test-path -Path $lokalniCestaAdresarCab)) {
        $null = new-item -ItemType Directory -Path $lokalniCestaAdresarCab -ErrorAction
SilentlyContinue
    }
    if (!(test-path $lokalniCestaSouborCab)) {
        Copy-Item -Path $sitovaCestaSouborCab -Destination $lokalniCestaAdresarCab
    }
    expand -F:* $lokalniCestaSouborCab $lokalniCestaAdresarCab | out-null
    $vysledek = pnputil -a $infpath
    [int]$vysledekKod = (($vysledek | select-string -Pattern "Number successfully
imported:") -split ":")[1]
    subZapsatLogDoTxt " exit fncInstalaceOvladace"
    return $vysledekKod
}
function subNacteniVzdalenychTiskaren {
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
    Param(
        $opakovani = $false, #slouží pro opakovaný dotaz, v případě, že server vrátil
prázdný seznam - např. server zaneprázdněn,
        #tiskárny nenalezeny pro oddělení
        $kodOddeleni = $script:oddeleni
    )
    try {
        $vzorRegEx = fncPrevodNaRegEx -hodnota  $kodOddeleni -poziceVzoru 'na začátku'
        subZapsatLogDoTxt "start subNacteniVzdalenychTiskaren"
        if ($opakovani -eq $false) {
            subNapsatZpravuDoKonzole -sloupec1 'Načítám vzdálené tiskárny:' -sloupec2
$kodOddeleni -zarovnatNaPredchoziSloupce:$true
            subNapsatZpravuDoKonzole -stavZpravy čekejte -nahradSloupec sloupec3
            if ($logujOperace) { subObsahKonzoleDoHtml }
        }
        if ($nastavovatSecurity) {
            $script:tiskarnyVzdalene = { Get-Printer -CimSession $script:CimSessionVzdaleny
-Full -ErrorAction Stop | ? { $_.name -match $vzorRegEx } |
                select name, DriverName, PortName, PermissionSDDL, Location, Comment,
PrintProcessor, Datatype, SeparatorPageFile }.Invoke()
        }
        else {
            $script:tiskarnyVzdalene = { Get-Printer -CimSession $script:CimSessionVzdaleny
-ErrorAction Stop | ? { $_.name -match $vzorRegEx } |
                select name, DriverName, PortName, Location, Comment, PrintProcessor,
Datatype, SeparatorPageFile }.Invoke()
        }
        if ($script:tiskarnyVzdalene.count -eq 0) {
            if ($opakovani -eq $false) {
                $testNacteni = (Get-Printer -CimSession $script:CimSessionVzdaleny -
ErrorAction Stop | measure).count # pokud je 0 je server přetížen,
                # pokud více než 1, pako opakujeme načtení
                #pokud $opakování je $false jedná se o první průchod a testujeme, jestli
server vrací nějaké tiskárny,
                #pokud ano, opakujeme načtení vzdálených tiskáren. V opačném případě je
přetížen
                if ( $testNacteni -eq 0) {
                    subNapsatZpravuDoKonzole -stavZpravy 'server přetížen' -nahradSloupec
sloupec3
                    subZapsatLogDoTxt " exit subNacteniVzdalenychTiskaren - server
přetížen"
                    subUkonceniProgramu
                }
                else {
                    subZapsatLogDoTxt "opakovani subNacteniVzdalenychTiskaren"
                    subNacteniVzdalenychTiskaren -opakovani $true -kodOddeleni $kodOddeleni
                }
            }
            else {
                # Jsme tu po druhé, kdy server nevrátil pro dané oddělení tiskárny, proto
ověříme naposledy zda server vrátí jakýkoliv seznam tiskáren
                subZapsatLogDoTxt "druhe volani subNacteniVzdalenychTiskaren"
                $testNacteni = Get-Printer -CimSession $script:CimSessionVzdaleny -
ErrorAction Stop | select Name
                if ($testNacteni.count -gt 0) {
                    # server vrátil tiskárny
                    if ($testNacteni | ? { $_.name -match $vzorRegEx }) {
                        # ověříme jeslti seznam obsahuje tiskárny pro oddělení, pokud ano,
byl server přetížen a program dál neporkačuje,
                        # protože chybí náročné SDDL zabezpečení
                        subNapsatZpravuDoKonzole -stavZpravy 'server přetížen' -
nahradSloupec sloupec3
                        subZapsatLogDoTxt " exit subNacteniVzdalenychTiskaren - server
přetížen"
                        subUkonceniProgramu
                    }
                    else {
                        # finální oveření, seznam neobsahuje tiskárny pro dané oddělení a
tak je odebereme z počítače
                        subZapsatLogDoTxt " exit subNacteniVzdalenychTiskaren - tiskárny
nenalezeny"
                        subNapsatZpravuDoKonzole -sloupec3 "nebyla nalezena žádná vzdálená
tiskárna" -nahradSloupec sloupec3 -barva Yellow
                        subNacteniLokalnichTiskaren
                        subOdebraniNespravovanychTiskaren
                        subOdebraniNespravovanychPortu
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
                    subOdebraniNepotrebnychOvladacu
                    subUkonceniProgramu
                }
            }
            else {
                subZapsatLogDoTxt " exit subNacteniVzdalenychTiskaren - server
přetížen"
                subNapsatZpravuDoKonzole -stavZpravy 'server přetížen' -nahradSloupec
sloupec3
                subUkonceniProgramu
            }
        }
    }
    else {
        subNapsatZpravuDoKonzole -sloupec3 "načtených:
$($script:tiskarnyVzdalene.count)" -nahradSloupec sloupec3 -barva Green
        sleep 5
    }
}
catch {
    subNapsatZpravuDoKonzole -stavZpravy chyba -nahradSloupec sloupec2
    subZapsatLogDoTxt " exit subNacteniVzdalenychTiskaren - chyba:
$(fncVratitDuvodChyby -psitem $psitem)"
    subUkonceniProgramu
}
if ($logujOperace) { subObsahKonzoleDoHtml }
subZapsatLogDoTxt " exit subNacteniVzdalenychTiskaren"
}
function subNacteniVzdalenychPortu {
try {
    subZapsatLogDoTxt "start subNacteniVzdalenychPortu"
    subNapsatZpravuDoKonzole -sloupec1 "Načítám vzdálené porty"
    subNapsatZpravuDoKonzole -stavZpravy čekejte -nahradSloupec sloupec2
    if ($logujOperace) { subObsahKonzoleDoHtml }
    $porty = $script:tiskarnyVzdalene | % { $_.portname }
    $script:portyVzdalene = { Get-PrinterPort -CimSession $script:CimSessionVzdaleny -
ErrorAction Stop | ? { $porty -match $_.name } |
        select Name, PrinterHostAddress, PortNumber, Protocol, SNMPCommunity,
SNMPEnabled, SNMPIndex, LprByteCounting, LprQueueName }.Invoke()
    if ($portyVzdalene.count -eq 0) {
        $script:portyVzdalene = @()
        throw "nebyly nalezeny žádné porty"
    }
    else {
        subNapsatZpravuDoKonzole -stavZpravy ok -nahradSloupec sloupec2
    }
}
catch {
    subNapsatZpravuDoKonzole -stavZpravy chyba -nahradSloupec sloupec2
    subZapsatLogDoTxt "start subNacteniVzdalenychPortu - chyba: $(fncVratitDuvodChyby -
psitem $psitem)"
}
subZapsatLogDoTxt " exit subNacteniVzdalenychPortu"
if ($logujOperace) { subObsahKonzoleDoHtml }
}
function subNacteniVzdalenychOvladacu {
try {
    subZapsatLogDoTxt "start subNacteniVzdalenychOvladacu"
    subNapsatZpravuDoKonzole -sloupec1 'Načítám vzdálené ovladače'
    subNapsatZpravuDoKonzole -stavZpravy čekejte -nahradSloupec sloupec2
    if ($logujOperace) { subObsahKonzoleDoHtml }
    $osArchitektura = fncVratitBitovouVerziOS
    $script:ovladaceVzdalene = { Get-PrinterDriver -CimSession
$script:CimSessionVzdaleny -ErrorAction Stop |
        ? { $_.name -notmatch $ignorovatTiskoveOvladace -and $_.PrinterEnvironment -
match $osArchitektura } |
        select Name, InfPath, PrinterEnvironment, DriverVersion }.Invoke()
    if ($script:ovladaceVzdalene.count -eq 0) {
        $script:ovladaceVzdalene = @()
        throw "nebyly nalezeny žádné ovladače"
    }
    else {
        subNapsatZpravuDoKonzole -stavZpravy ok -nahradSloupec sloupec2
    }
}
catch {
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
        subNapsatZpravuDoKonzole -stavZpravy chyba -nahradSloupec sloupec2
        subZapsatLogDoTxt " exit subNacteniVzdalenychOvladacu - chyba:
$(fncVratitDuvodChyby -psitem $psitem)"
        subUkonceniProgramu
    }
    subZapsatLogDoTxt " exit subNacteniVzdalenychOvladacu"
    if ($logujOperace) { subObsahKonzoleDoHtml }
}
function subNacteniLokalnichTiskaren {
    Param(
        [boolean]$all = $false
    )
    try {
        subZapsatLogDoTxt "start subNacteniLokalnichTiskaren"
        subNapsatZpravuDoKonzole -sloupec1 'Načítám lokální tiskárny'
        subNapsatZpravuDoKonzole -stavZpravy čekejte -nahradSloupec sloupec2
        if ($logujOperace) { subObsahKonzoleDoHtml }
        if ($all) {
            $script:tiskarnyLokalni = { Get-Printer -ErrorAction Stop | sort name |
                select name, DriverName, PortName, Location, Comment, PrintProcessor,
Datatype, SeparatorPageFile }.invoke()
        }
        else {
            if ($nastavovatSecurity) {
                $script:tiskarnyLokalni = { Get-Printer -ComputerName $env:COMPUTERNAME -
CimSession $script:CimSessionLokalni -Full -ErrorAction Stop |
                    sort name | ? { $_.drivername -notmatch $ignorovatTiskoveOvladace } |
                    select name, DriverName, PortName, PermissionSDDL, Location, Comment,
PrintProcessor, Datatype, SeparatorPageFile }.invoke()
            }
            else {
                $script:tiskarnyLokalni = { Get-Printer -ComputerName $env:COMPUTERNAME -
CimSession $script:CimSessionLokalni -ErrorAction Stop |
                    sort name | ? { $_.drivername -notmatch $ignorovatTiskoveOvladace } |
                    select name, DriverName, PortName, Location, Comment, PrintProcessor,
Datatype, SeparatorPageFile }.invoke()
            }
        }
        if ($script:tiskarnyLokalni.count -eq 0) {
            subNapsatZpravuDoKonzole -stavZpravy nenalezeny -nahradSloupec sloupec2
        }
        else {
            subNapsatZpravuDoKonzole -stavZpravy ok -nahradSloupec sloupec2

        }
    }
    catch {
        subNapsatZpravuDoKonzole -stavZpravy chyba -nahradSloupec sloupec2
        subZapsatLogDoTxt " exit subNacteniLokalnichTiskaren - chyba: $(fncVratitDuvodChyby
-psitem $psitem)"
        subUkonceniProgramu
    }
    subZapsatLogDoTxt " exit subNacteniLokalnichTiskaren"
    if ($logujOperace) { subObsahKonzoleDoHtml }
}
function subNacteniLokalnichPortu {
    try {
        subZapsatLogDoTxt "start subNacteniLokalnichPortu"
        subNapsatZpravuDoKonzole -sloupec1 'Načítám lokální porty'
        subNapsatZpravuDoKonzole -stavZpravy čekejte -nahradSloupec sloupec2
        if ($logujOperace) { subObsahKonzoleDoHtml }
        $script:portyLokalni = { Get-PrinterPort -ComputerName $script:nazevPC -ErrorAction
Stop | ? { $_.description -match "TCP/IP" } |
            select Name, PrinterHostAddress, PortNumber, Protocol, SNMPCommunity,
SNMPEnabled, SNMPIndex, LprByteCounting, LprQueueName }.Invoke()
        if ($portyLokalni.count -eq 0) {
            subNapsatZpravuDoKonzole -stavZpravy nenalezeny -nahradSloupec sloupec2
        }
        else {
            subNapsatZpravuDoKonzole -stavZpravy ok -nahradSloupec sloupec2
        }
    }
    catch {
        subNapsatZpravuDoKonzole -stavZpravy chyba -nahradSloupec sloupec2
        subZapsatLogDoTxt " exit subNacteniLokalnichPortu - chyba: $(fncVratitDuvodChyby -
psitem $psitem)"
```

```
    }
    if ($logujOperace) { subObsahKonzoleDoHtml }
    subZapsatLogDoTxt " exit subNacteniLokalnichPortu"
}
function subNacteniLokalnichOvladacu {
    try {
        subZapsatLogDoTxt "start subNacteniLokalnichOvladacu"
        subNapsatZpravuDoKonzole -sloupec1 'Načítám lokální ovladače'
        subNapsatZpravuDoKonzole -stavZpravy čekejte -nahradSloupec sloupec2
        if ($logujOperace) { subObsahKonzoleDoHtml }
        $script:ovladaceLokalni = { Get-PrinterDriver -ComputerName $script:nazevPC -
ErrorAction Stop |
            ? { $_.name -notmatch $ignorovatTiskoveOvladace } | select Name, InfPath,
PrinterEnvironment, DriverVersion }.Invoke()
        if ($script:ovladaceLokalni.count -eq 0) {
            subNapsatZpravuDoKonzole -stavZpravy nenalezeny -nahradSloupec sloupec2
        }
        else {
            subNapsatZpravuDoKonzole -stavZpravy ok -nahradSloupec sloupec2
        }
    }
    catch {
        subNapsatZpravuDoKonzole -stavZpravy chyba -nahradSloupec sloupec2
        subZapsatLogDoTxt " exit subNacteniLokalnichOvladacu - chyba: $(fncVratitDuvodChyby
-psitem $psitem)"
    }
    if ($logujOperace) { subObsahKonzoleDoHtml }
    subZapsatLogDoTxt " exit subNacteniLokalnichOvladacu"
}
function subNacteniLokalniKonfiguraceTiskaren {
    param(
        $kodOddeleni = $script:oddeleni
    )
    try {
        $vzorRegEx = fncPrevodNaRegEx -hodnota  $kodOddeleni -poziceVzoru 'na začátku'
        subZapsatLogDoTxt "start subNacteniLokalniKonfiguraceTiskaren"
        subNapsatZpravuDoKonzole -sloupec1 'Načítám konfiguraci lokálních tiskáren'
        subNapsatZpravuDoKonzole -stavZpravy čekejte -nahradSloupec sloupec2
        if ($logujOperace) { subObsahKonzoleDoHtml }
        [boolean]$script:chybaNacteniLokalniKonfigurace = $false
        $script:chybaNacteniLokalniKonfiguraceNektere = $false
        subZapsatLogDoTxt "      subNacteniLokalniKonfiguraceTiskaren - start get-printer"
        $tiskarnyLokalniPredvolbyTMP = get-printer -ComputerName $env:COMPUTERNAME -
CimSession $script:CimSessionLokalni -ErrorAction Stop | ? { $_.Name -match $vzorRegEx } |
select name
        $script:tiskarnyLokalniPredvolby = @()
        $tiskarnyLokalniPredvolbyTMP | % {
            subZapsatLogDoTxt "      subNacteniLokalniKonfiguraceTiskaren -
$($_.name)"
            $script:tiskarnyLokalniPredvolby += Get-PrintConfiguration $_.name -CimSession
$script:CimSessionLokalni
        }
        subZapsatLogDoTxt "      subNacteniLokalniKonfiguraceTiskaren - exit  get-printer"
        if ($script:tiskarnyLokalniPredvolby.count -eq 0) {
            subNapsatZpravuDoKonzole -stavZpravy nenalezena -nahradSloupec sloupec2
            $script:chybaNacteniLokalniKonfigurace = $true
        }
        else {
            subNapsatZpravuDoKonzole -stavZpravy ok -nahradSloupec sloupec2
        }
    }
    catch {
        $script:chybaNacteniLokalniKonfigurace = $true
        subZapsatLogDoTxt "      subNacteniLokalniKonfiguraceTiskaren - catch get-printer"
        if ($psitem.Exception.message -match "specified printer") {
            # očekává se chyba "Could not access configuration information for the
specified printer."
            $script:tiskarnyLokalniPredvolby = { get-printer -ComputerName
$env:COMPUTERNAME -CimSession $script:CimSessionLokalni -ErrorAction Stop | ? { $_.Name -
match $vzorRegEx } | Get-PrintConfiguration -ErrorAction SilentlyContinue }.invoke()
            subZapsatLogDoTxt "      subNacteniLokalniKonfiguraceTiskaren - exit catch get-
printer"
            if (($script:tiskarnyLokalniPredvolby | measure).Count -gt 0) {
                $script:chybaNacteniLokalniKonfiguraceNektere = $true
            }
        }
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
        subNapsatZpravuDoKonzole -stavZpravy chyba -nahradSloupec sloupec2
        subZapsatLogDoTxt " exit subNacteniLokalniKonfiguraceTiskaren - chyba:
$(fncVratitDuvodChyby -psitem $psitem)"
    }
    if ($logujOperace) { subObsahKonzoleDoHtml }
    subZapsatLogDoTxt " exit subNacteniLokalniKonfiguraceTiskaren"
}
function fncVratitDuvodChyby {
    Param(
        $psitem
    )
    #.Exception.message
    $exception = ($psitem.Exception.message).split(":")
    $pocet = ($exception | measure).count
    switch ($pocet) {
        1 { return ($exception).trim('"') }
        2 { return ($exception[1]).trim('"') }
        3 { return ($exception[2]).trim('"') }
        default { return "neznámá chyba" }
    }
}
function subNacteniVzdaleneKonfiguraceTiskaren {
    try {
        subZapsatLogDoTxt "start subNacteniVzdaleneKonfiguraceTiskaren"
        subNapsatZpravuDoKonzole -sloupec1 'Načítám konfiguraci vzdálených tiskáren'
        subNapsatZpravuDoKonzole -stavZpravy čekejte -nahradSloupec sloupec2
        if ($logujOperace) { subObsahKonzoleDoHtml }
        [boolean]$script:chybaNacteniVzdaleneKonfigurace = $false
        subZapsatLogDoTxt "      subNacteniVzdaleneKonfiguraceTiskaren - start get-printer"
        $tiskarnyVzdalenePredvolbyTMP = $script:tiskarnyVzdalene | select name  #
        $script:tiskarnyVzdalenePredvolby = @()
        $tiskarnyVzdalenePredvolbyTMP | % {
            subZapsatLogDoTxt "      subNacteniVzdaleneKonfiguraceTiskaren -
$($_.name)"
            $script:tiskarnyVzdalenePredvolby += Get-PrintConfiguration $_.name -CimSession
$script:CimSessionVzdaleny
        }
        subZapsatLogDoTxt "      subNacteniVzdaleneKonfiguraceTiskaren - exit  get-printer"
        if ($tiskarnyVzdalenePredvolby.count -eq 0) {
            subNapsatZpravuDoKonzole -stavZpravy nenalezena -nahradSloupec sloupec2
            $script:chybaNacteniVzdaleneKonfigurace = $true
        }
        else {
            subNapsatZpravuDoKonzole -stavZpravy ok -nahradSloupec sloupec2
        }
    }
    catch {
        $script:chybaNacteniVzdaleneKonfigurace = $true
        subNapsatZpravuDoKonzole -stavZpravy chyba -nahradSloupec sloupec2
        subZapsatLogDoTxt " exit subNacteniVzdaleneKonfiguraceTiskaren - chyba:
$(fncVratitDuvodChyby -psitem $psitem)"
    }
    if ($logujOperace) { subObsahKonzoleDoHtml }
    subZapsatLogDoTxt " exit subNacteniVzdaleneKonfiguraceTiskaren"
}
function subKontrolaChybejicichPortu {
    subZapsatLogDoTxt "start subKontrolaChybejicichPortu"
    foreach ($tiskarnaVzdalena in $script:tiskarnyVzdalene) {
        subZapsatLogDoTxt "      subKontrolaChybejicichPortu - port:
$($tiskarnaVzdalena.PortName)"
        subNapsatZpravuDoKonzole -sloupec1 'Ověřuji port: ' -sloupec2
$tiskarnaVzdalena.PortName
        if ($logujOperace) { subObsahKonzoleDoHtml }
        $tiskarnaVzdalenaName = $tiskarnaVzdalena.name
        if (($portyLokalni | measure).count -gt 0) {
            $indexLokalni = $portyLokalni.name.IndexOf($tiskarnaVzdalena.PortName)
#nalezneme pozadovany port -1 port nenalezen 0<= port nalezen
        }
        else {
            $indexLokalni = -1
        }
        $indexVzdalene = $portyVzdalene.name.IndexOf($tiskarnaVzdalena.PortName) #nalezneme
pozadovany port -1 port nenalezen 0<= port nalezen
        $portVzdalenyConfig = $portyVzdalene[$indexVzdalene]
        $portLokalniConfig = $portyLokalni[$indexlokalni]
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
        $portVzdalenyName = $portVzdalenyConfig.name
        $portVzdalenyProtocol = $portVzdalenyConfig.protocol
        $portVzdalenyPrinterHostAddress = $portVzdalenyConfig.PrinterHostAddress
        $portVzdalenyLprByteCounting =
[System.Convert]::ToBoolean($portVzdalenyConfig.LprByteCounting)
        $portVzdalenyLprQueueName = $portVzdalenyConfig.LprQueueName
        $portVzdalenyNumber = $portVzdalenyConfig.PortNumber
        $portVzdalenySNMPCommunity = $portVzdalenyConfig.SNMPCommunity
        $portVzdalenySNMPEnabled =
[System.Convert]::ToBoolean($portVzdalenyConfig.SNMPEnabled)
        $portVzdalenySNMPIndex = $portVzdalenyConfig.SNMPIndex

        $portLokalniName = $portLokalniConfig.name
        $portLokalniProtocol = $portLokalniConfig.protocol
        $portLokalniPrinterHostAddress = $portLokalniConfig.PrinterHostAddress
        $portLokalniLprByteCounting =
[System.Convert]::ToBoolean($portLokalniConfig.LprByteCounting)
        $portLokalniLprQueueName = $portLokalniConfig.LprQueueName
        $portLokalniNumber = $portLokalniConfig.PortNumber
        $portLokalniSNMPCommunity = $portLokalniConfig.SNMPCommunity
        $portLokalniSNMPEnabled =
[System.Convert]::ToBoolean($portLokalniConfig.SNMPEnabled)
        $portLokalniSNMPIndex = $portLokalniConfig.SNMPIndex

        if ($indexLokalni -ge 0) {
            if ($portVzdalenyProtocol -eq "RAW") {
                try {
                    if ($portVzdalenySNMPEnabled) {
                        $vysledek = Compare-Object -ReferenceObject $portVzdalenyConfig -
DifferenceObject $portLokalniConfig -Property PrinterHostAddress, PortNumber,
SNMPCommunity, SNMPEnabled, SNMPIndex  | % { $_.SideIndicator -eq "=>" }
                        if ($vysledek -gt 0) {
                            subNapsatZpravuDoKonzole -stavZpravy rozdíl -nahradSloupec
sloupec3
                            subNapsatZpravuDoKonzole -stavZpravy opravuji -nahradSloupec
sloupec4
                            Set-Printer -Name $tiskarnaVzdalenaName -PortName "LPT3:" -
ErrorAction Stop
                            try {
                                $null = Remove-PrinterPort -Name $portVzdalenyName -
ComputerName $env:computername -ErrorAction stop
                            }
                            catch {
                                subRestartSpooler
                                try {
                                    $null = Remove-PrinterPort -Name $portVzdalenyName -
ComputerName $env:computername -ErrorAction stop
                                }
                                catch {
                                    Set-Printer -Name $tiskarnaVzdalenaName -PortName
$portVzdalenyName -ErrorAction Stop
                                }
                            }
                            Add-PrinterPort -Name $portVzdalenyName -PrinterHostAddress
$portVzdalenyPrinterHostAddress -PortNumber $portVzdalenyNumber -SNMP
$portVzdalenySNMPIndex -SNMPCommunity $portVzdalenySNMPCommunity -ErrorAction
SilentlyContinue
                            Set-Printer -Name $tiskarnaVzdalenaName -PortName
$portVzdalenyName -ErrorAction Stop
                            subNapsatZpravuDoKonzole -stavZpravy opraveno -nahradSloupec
sloupec4
                        }
                        else {
                            subNapsatZpravuDoKonzole -stavZpravy shoda -nahradSloupec
sloupec3
                        }
                    }
                    else {
                        $vysledek = compare-object -ReferenceObject $portVzdalenyConfig -
DifferenceObject $portLokalniConfig -Property PrinterHostAddress, PortNumber, SNMPEnabled
| % { $_.SideIndicator -eq "=>" }
                        if ($vysledek -gt 0) {
                            subNapsatZpravuDoKonzole -stavZpravy rozdíl -nahradSloupec
sloupec3
                            subNapsatZpravuDoKonzole -stavZpravy opravuji -nahradSloupec
sloupec4
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
                            Set-Printer -Name $tiskarnaVzdalenaName -PortName "LPT3:" -
ErrorAction Stop
                            Remove-PrinterPort -Name $portVzdalenyName -ComputerName
$env:computername -ErrorAction SilentlyContinue
                            Add-PrinterPort -Name $portVzdalenyName -PrinterHostAddress
$portVzdalenyPrinterHostAddress -PortNumber $portVzdalenyNumber -ErrorAction Stop
                            Set-Printer -Name $tiskarnaVzdalenaName -PortName
$portVzdalenyName -ErrorAction Stop
                            subNapsatZpravuDoKonzole -stavZpravy opraveno -nahradSloupec
sloupec4
                        }
                        else {
                            subNapsatZpravuDoKonzole -stavZpravy shoda -nahradSloupec
sloupec3
                        }
                    }
                }
                catch {
                    subNapsatZpravuDoKonzole -stavZpravy neopraveno -nahradSloupec sloupec4
                }
            }
            elseif ($portVzdalenyProtocol -eq "LPR") {
                try {
                    if ($portVzdalenySNMPEnabled) {
                        $vysledek = compare-object -ReferenceObject $portVzdalenyConfig -
DifferenceObject $portLokalniConfig -Property PrinterHostAddress, LprByteCounting,
LprQueueName, SNMPCommunity, SNMPEnabled, SNMPIndex  | % { $_.SideIndicator -eq "=>" }
                        if ($vysledek -gt 0) {
                            subNapsatZpravuDoKonzole -stavZpravy rozdíl -nahradSloupec
sloupec3
                            subNapsatZpravuDoKonzole -stavZpravy opravuji -nahradSloupec
sloupec4
                            Set-Printer -Name $tiskarnaVzdalenaName -PortName "LPT3:" -
ErrorAction Stop
                            Remove-PrinterPort -Name $portVzdalenyName -ComputerName
$env:computername -ErrorAction SilentlyContinue
                            Add-PrinterPort -Name $portVzdalenyName -LprHostAddress
$portVzdalenyPrinterHostAddress -LprQueueName $portVzdalenyLprQueueName -
LprByteCounting:$portVzdalenyLprByteCounting  -SNMP $portVzdalenySNMPIndex -SNMPCommunity
$portVzdalenySNMPCommunity -ErrorAction stop
                            Set-Printer -Name $tiskarnaVzdalenaName -PortName
$portVzdalenyName -ErrorAction Stop
                            subNapsatZpravuDoKonzole -stavZpravy opraveno -nahradSloupec
sloupec4
                        }
                        else {
                            subNapsatZpravuDoKonzole -stavZpravy shoda -nahradSloupec
sloupec3
                        }
                    }
                    else {
                        $vysledek = compare-object -ReferenceObject $portVzdalenyConfig -
DifferenceObject $portLokalniConfig -Property PrinterHostAddress, LprQueueName,
LprByteCounting, SNMPEnabled  | % { $_.SideIndicator -eq "=>" }
                        if ($vysledek -gt 0) {
                            subNapsatZpravuDoKonzole -stavZpravy rozdíl -nahradSloupec
sloupec3
                            subNapsatZpravuDoKonzole -stavZpravy opravuji -nahradSloupec
sloupec4
                            Set-Printer -Name $tiskarnaVzdalenaName -PortName "LPT3:" -
ErrorAction Stop
                            Remove-PrinterPort -Name $portVzdalenyName -ComputerName
$env:computername -ErrorAction SilentlyContinue
                            Add-PrinterPort -Name $portVzdalenyName -LprHostAddress
$portVzdalenyPrinterHostAddress -LprQueueName $portVzdalenyLprQueueName -
LprByteCounting:$portVzdalenyLprByteCounting  -ErrorAction stop
                            Set-Printer -Name $tiskarnaVzdalenaName -PortName
$portVzdalenyName -ErrorAction Stop
                            subNapsatZpravuDoKonzole -stavZpravy opraveno -nahradSloupec
sloupec4
                        }
                        else {
                            subNapsatZpravuDoKonzole -stavZpravy shoda -nahradSloupec
sloupec3
                        }
                    }
```

```
                }
                catch {
                    subNapsatZpravuDoKonzole -stavZpravy neopraveno -nahradSloupec sloupec4
                    subZapsatLogDoTxt "start subKontrolaChybejicichPortu - port:
$($tiskarnaVzdalena.PortName) - chyba: $(fncVratitDuvodChyby -psitem $psitem)"
                }
            }
        }
        else {
            subNapsatZpravuDoKonzole -stavZpravy neexistuje -nahradSloupec sloupec3
            if ($portVzdalenyProtocol -eq "RAW") {
                try {
                    subNapsatZpravuDoKonzole -stavZpravy opravuji -nahradSloupec sloupec4
                    if ($portVzdalenySNMPEnabled) {
                        Add-PrinterPort -Name $portVzdalenyName -PrinterHostAddress
$portVzdalenyPrinterHostAddress -PortNumber $portVzdalenyNumber -SNMP
$portVzdalenySNMPIndex -SNMPCommunity $portVzdalenySNMPCommunity -ErrorAction Stop
                    }
                    else {
                        Add-PrinterPort -Name $portVzdalenyName -PrinterHostAddress
$portVzdalenyPrinterHostAddress -PortNumber $portVzdalenyNumber -ErrorAction Stop
                    }
                    subNapsatZpravuDoKonzole -stavZpravy opraveno -nahradSloupec sloupec4
                }
                catch {
                    subNapsatZpravuDoKonzole -stavZpravy neopraveno -nahradSloupec sloupec4
                    subZapsatLogDoTxt "start subKontrolaChybejicichPortu - port:
$($tiskarnaVzdalena.PortName) - chyba: $(fncVratitDuvodChyby -psitem $psitem)"
                }
            }
            elseif ($portVzdalenyProtocol -eq "LPR") {
                try {
                    subNapsatZpravuDoKonzole -stavZpravy opravuji -nahradSloupec sloupec4
                    if ($portVzdalenySNMPEnabled) {
                        Add-PrinterPort -Name $portVzdalenyName -LprHostAddress
$portVzdalenyPrinterHostAddress -LprQueueName $portVzdalenyLprQueueName -
LprByteCounting:$portVzdalenyLprByteCounting  -SNMP $portVzdalenySNMPIndex -SNMPCommunity
$portVzdalenySNMPCommunity -ErrorAction stop
                    }
                    else {
                        Add-PrinterPort -Name $portVzdalenyName -LprHostAddress
$portVzdalenyPrinterHostAddress -LprQueueName $portVzdalenyLprQueueName -
LprByteCounting:$portVzdalenyLprByteCounting  -ErrorAction stop
                    }
                    subNapsatZpravuDoKonzole -stavZpravy opraveno -nahradSloupec sloupec4
                }
                catch {
                    subNapsatZpravuDoKonzole -stavZpravy neopraveno -nahradSloupec sloupec4
                    subZapsatLogDoTxt "start subKontrolaChybejicichPortu - port:
$($tiskarnaVzdalena.PortName) - chyba: $(fncVratitDuvodChyby -psitem $psitem)"
                }
            }
        }
        if ($logujOperace) { subObsahKonzoleDoHtml }
    }
    subZapsatLogDoTxt " exit subKontrolaChybejicichPortu"
}
function subKontrolaChybejicichTiskaren {
    foreach ($tiskarnaVzdalena in $script:tiskarnyVzdalene) {
        subZapsatLogDoTxt "      subKontrolaChybejicichTiskaren - tiskárna:
$($tiskarnaVzdalena.Name)"
        subNapsatZpravuDoKonzole -sloupec1 'Ověřuji tiskárnu:' -sloupec2
$tiskarnaVzdalena.Name
        if ($logujOperace) { subObsahKonzoleDoHtml }
        if (($script:tiskarnyLokalni | measure).count -gt 0) {
            $indexLokalni = $script:tiskarnyLokalni.name.IndexOf($tiskarnaVzdalena.Name)
#nalezneme pozadovany název tiskárny index -1 = nenalezen, index větší nebo rovno 0 =
nalezen
        }
        else {
            $indexLokalni = -1
        }
        $indexVzdalene = $script:tiskarnyVzdalene.name.IndexOf($tiskarnaVzdalena.Name)
#nalezneme pozadovany název tiskárny index -1 = nenalezen, index větší nebo rovno 0 =
nalezen
        $tiskarnaVzdalenaConfig = $script:tiskarnyVzdalene[$indexVzdalene]
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
        $tiskarnaVzdalenaName = $tiskarnaVzdalenaConfig.name
        $tiskarnaVzdalenaDriverName = $tiskarnaVzdalenaConfig.DriverName
        $tiskarnaVzdalenaPortName = $tiskarnaVzdalenaConfig.PortName
        $tiskarnaVzdalenaComment = $tiskarnaVzdalenaConfig.Comment
        $tiskarnaVzdalenaLocation = $tiskarnaVzdalenaConfig.Location
        if ($indexLokalni -ge 0) {
            subNapsatZpravuDoKonzole -sloupec3 'nalezena' -nahradSloupec sloupec3 -barva
Green
        }
        else {
            subNapsatZpravuDoKonzole -stavZpravy neexistuje -nahradSloupec sloupec3
            subNapsatZpravuDoKonzole -stavZpravy opravuji -nahradSloupec sloupec4
            if ($logujOperace) { subObsahKonzoleDoHtml }
            try {
                if ($nastavovatSecurity) {
                    $tiskarnaVzdalenaPermissionSDDL =
$tiskarnaVzdalenaConfig.PermissionSDDL
                    Add-Printer -Name $tiskarnaVzdalenaName -DriverName
$tiskarnaVzdalenaDriverName -PortName $tiskarnaVzdalenaPortName -PermissionSDDL
$tiskarnaVzdalenaPermissionSDDL -Comment $tiskarnaVzdalenaComment -Location
$tiskarnaVzdalenaLocation
                }
                else {
                    Add-Printer -Name $tiskarnaVzdalenaName -DriverName
$tiskarnaVzdalenaDriverName -PortName $tiskarnaVzdalenaPortName -Comment
$tiskarnaVzdalenaComment -Location $tiskarnaVzdalenaLocation
                }
                subNapsatZpravuDoKonzole -stavZpravy opraveno -nahradSloupec sloupec4
            }
            catch {
                subNapsatZpravuDoKonzole -stavZpravy neopraveno -nahradSloupec sloupec4
                subZapsatLogDoTxt "start subKontrolaChybejicichTiskaren - tiskárna:
$($tiskarnaVzdalena.Name) - chyba: $(fncVratitDuvodChyby -psitem $psitem)"
            }
        }
        if ($logujOperace) { subObsahKonzoleDoHtml }
    }
    subZapsatLogDoTxt " exit subKontrolaChybejicichTiskaren"
}
function subOdebraniNespravovanychTiskaren {
    subZapsatLogDoTxt "start subOdebraniNespravovanychTiskaren"
    $nespravovaneTiskarny = $script:tiskarnyLokalni | ? { $_.PortName -notmatch
$ignorovatPorty -and $_.Name -notmatch $ignorovatNazvyTiskaren } # nebudeme odebirat
tiskarnu, ktera používá ignorovaný port
    if (($nespravovaneTiskarny | measure).count -gt 0) {
        foreach ($nespravovanaTiskarna in $nespravovaneTiskarny) {
            if (($script:tiskarnyVzdalene | measure).count -gt 0) {
                $indexLokalni =
$script:tiskarnyVzdalene.name.IndexOf($nespravovanaTiskarna.Name)    #nalezneme pozadovany
port -1 port nenalezen 0<= port nalezen
            }
            else {
                $indexLokalni = -1
            }

            if ($indexLokalni -eq -1) {
                subZapsatLogDoTxt "      subOdebraniNespravovanychTiskaren - odebírám:
$($nespravovanaTiskarna.name)"
                subNapsatZpravuDoKonzole -sloupec1 'Odebírám tiskárnu:' -sloupec2
$nespravovanaTiskarna.name -barva Yellow
                subNapsatZpravuDoKonzole -stavZpravy čekejte -nahradSloupec sloupec3
                if ($logujOperace) { subObsahKonzoleDoHtml }
                try {
                    remove-printer -name $nespravovanaTiskarna.name -ErrorAction stop
                    subNapsatZpravuDoKonzole -stavZpravy ok -nahradSloupec sloupec3
                }
                catch {
                    subNapsatZpravuDoKonzole -stavZpravy chyba -nahradSloupec sloupec3
                    subZapsatLogDoTxt "      subOdebraniNespravovanychTiskaren - odebírám:
$($nespravovanaTiskarna.name) - chyba: $(fncVratitDuvodChyby -psitem $psitem)"
                }
            }
            if ($logujOperace) { subObsahKonzoleDoHtml }
        }
        subZapsatLogDoTxt " exit subOdebraniNespravovanychTiskaren"
    }
```

```
    else {
        subZapsatLogDoTxt " exit subOdebraniNespravovanychTiskaren"
        # není co odebírat, nebyly nalezeny žádné lokální tiskárny
    }
}
function subOdebraniNespravovanychPortu {
    subNacteniLokalnichPortu
    subZapsatLogDoTxt "start subOdebraniNespravovanychPortu"
    $portyLokalni = $portyLokalni | ? { $_.name -notmatch $ignorovatPorty }
    if (($portyLokalni | measure).count -gt 0) {
        foreach ($portLokalni in $portyLokalni) {

            if (($portyVzdalene | measure).Count -gt 0) {
                $indexLokalni = $portyVzdalene.name.IndexOf($portLokalni.Name)
#nalezneme pozadovany port -1 port nenalezen 0<= port nalezen
            }
            else {
                $indexLokalni = -1
            }
            if ($indexLokalni -eq -1) {
                subZapsatLogDoTxt "      subOdebraniNespravovanychPortu - odebírám:
$($portLokalni.name)"
                subNapsatZpravuDoKonzole -sloupec1 'Odebírám port:' -sloupec2
$portLokalni.name -barva Yellow
                subNapsatZpravuDoKonzole -stavZpravy čekejte -nahradSloupec sloupec3
                if ($logujOperace) { subObsahKonzoleDoHtml }
                try {
                    Remove-PrinterPort -name $portLokalni.name -ComputerName
$env:computername -ErrorAction Stop
                    subNapsatZpravuDoKonzole -stavZpravy ok -nahradSloupec sloupec3
                }
                catch {
                    subNapsatZpravuDoKonzole -stavZpravy chyba -nahradSloupec sloupec2
                    subZapsatLogDoTxt "      subOdebraniNespravovanychPortu - odebírám:
$($portLokalni.name) - chyba: $(fncVratitDuvodChyby -psitem $psitem)"
                }
            }
            if ($logujOperace) { subObsahKonzoleDoHtml }
        }
        subZapsatLogDoTxt " exit subOdebraniNespravovanychPortu"
    }
    else {
        subZapsatLogDoTxt " exit subOdebraniNespravovanychPortu"
        return # není co odebírat, nebyly nalezeny žádné lokální tiskárny
    }
}
function subRestartSpooler {
    Stop-Service Spooler -Force -ErrorAction SilentlyContinue -WarningAction
SilentlyContinue
    Start-Service Spooler -ErrorAction SilentlyContinue -WarningAction SilentlyContinue
}
function subOdebraniNepotrebnychOvladacu {
    subZapsatLogDoTxt "start subOdebraniNepotrebnychOvladacu - call subprocess
subNacteniLokalnichTiskaren, subNacteniLokalnichOvladacu"
    subNacteniLokalnichTiskaren -all $true
    subNacteniLokalnichOvladacu
    subZapsatLogDoTxt "start subOdebraniNepotrebnychOvladacu"
    if (($script:ovladaceLokalni | measure).count -gt 0) {
        foreach ($ovladacLokalni in $script:ovladaceLokalni) {
            $ovladacLokalniNazev = $ovladacLokalni.name
            if (($script:tiskarnyLokalni | measure).count -gt 0) {
                $indexLokalni =
$script:tiskarnyLokalni.DriverName.indexof($ovladacLokalniNazev)
            }
            else {
                $indexLokalni = -1
            }
            subNapsatZpravuDoKonzole -sloupec1 'Kontrola nepotřebných ovladačů:' -sloupec2
$ovladacLokalniNazev
            if ($logujOperace) { subObsahKonzoleDoHtml }
            if ($indexLokalni -eq -1) {
                try {
                    subZapsatLogDoTxt "      subOdebraniNepotrebnychOvladacu - odebírám:
$ovladacLokalniNazev"
                    Remove-PrinterDriver -Name $ovladacLokalniNazev -ErrorAction Stop
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
                    subNapsatZpravuDoKonzole -sloupec3 'odebrán' -nahradSloupec sloupec3 -
barva Yellow
                }
                catch {
                    if ($vynucovatOdebraniOvladacu) {
                        subNapsatZpravuDoKonzole -sloupec3 'neodebrán' -nahradSloupec
sloupec3 -barva Red
                        try {
                            subNapsatZpravuDoKonzole -sloupec4 'vynucuji' -nahradSloupec
sloupec4 -barva Yellow
                            subRestartSpooler
                            Start-Process rundll32 -ArgumentList "printui.dll PrintUIEntry
/q /dd /m""$ovladacLokalniNazev""" -Wait
                            $test = ((Get-PrinterDriver | ? { $_.name -match
$ovladacLokalniNazev }) | measure).count
                            if ($test -gt 0) { throw "nepovedlo se" }
                            subNapsatZpravuDoKonzole -sloupec3 'odebráno' -nahradSloupec
sloupec3 -barva Green
                            subNapsatZpravuDoKonzole -sloupec4 '' -nahradSloupec sloupec4
                        }
                        catch {
                            subZapsatLogDoTxt "      subOdebraniNepotrebnychOvladacu -
odebírám: $ovladacLokalniNazev - nepovedlo se"
                            subNapsatZpravuDoKonzole -sloupec4 'vynucení se nepovedlo' -
nahradSloupec sloupec4 -barva Yellow
                        }
                        Start-Service SAPSprint -ErrorAction SilentlyContinue -
WarningAction SilentlyContinue # zapínám tiskovou službu SAPu, pokud existuje
                    }
                    else {
                        subZapsatLogDoTxt "      subOdebraniNepotrebnychOvladacu -
odebírám: $ovladacLokalniNazev - neodebrán"
                        subNapsatZpravuDoKonzole -sloupec3 'neodebrán' -nahradSloupec
sloupec3 -barva Red
                    }
                }
            }
            else {
                subNapsatZpravuDoKonzole -sloupec3 'používán' -nahradSloupec sloupec3 -
barva Green
            }
            if ($logujOperace) { subObsahKonzoleDoHtml }
        }
    }
    subZapsatLogDoTxt " exit subOdebraniNepotrebnychOvladacu"
}
function subKontrolaChybejicichOvladacu {
    subZapsatLogDoTxt "start subKontrolaChybejicichOvladacu"
    foreach ($tiskarnaVzdalena in ($script:tiskarnyVzdalene)) {
        subZapsatLogDoTxt "      subKontrolaChybejicichOvladacu - ověřuji ovladač pro:
$($tiskarnaVzdalena.name)"
        subNapsatZpravuDoKonzole -sloupec1 'Ověřuji ovladač pro tiskárnu:' -sloupec2
$tiskarnaVzdalena.name
        if ($logujOperace) { subObsahKonzoleDoHtml }
        if (($script:ovladaceLokalni | measure).count -gt 0) {
            $indexLokalni =
$script:ovladaceLokalni.name.IndexOf($tiskarnaVzdalena.DriverName)    #nalezneme pozadovany
ovladač -1 ovladač nenalezen 0<= ovladač nalezen
        }
        else {
            $indexLokalni = -1
        }
        $indexVzdalene =
$script:ovladaceVzdalene.name.IndexOf($tiskarnaVzdalena.DriverName) #nalezneme pozadovany
ovladač -1 ovladač nenalezen 0<= ovladač nalezen
        $verzeVzdalena = $script:ovladaceVzdalene[$indexVzdalene].DriverVersion
        $infPath = $script:ovladaceVzdalene[$indexVzdalene].infPath
        [array]$splitCestaOvladace = $infPath -split '\\' # rozdělíme cestu oddělovačem
zpětné lomítko s použitím escape \
        #$ovladacID = $nazevSouboruCAB = $splitCestaOvladace[($splitCestaOvladace.count -
2)]
        $ovladacID = $splitCestaOvladace[($splitCestaOvladace.count - 2)] # z cesty
vybereme název, pod kterým tiskový server balí ovladače
        $nazevSouboruCAB = $ovladacID + ".cab" # přidáme příponu
        if ($indexLokalni -ge 0) {
            $verzeLokalni = $script:ovladaceLokalni[$indexLokalni].DriverVersion
```

```
            if ($verzeLokalni -eq $verzeVzdalena) {
                subNapsatZpravuDoKonzole -sloupec3 'shodná verze' -nahradSloupec sloupec3 -
barva Green
            }
            else {
                subNapsatZpravuDoKonzole -sloupec3 'rozdílná verze' -nahradSloupec sloupec3
-barva Yellow
                subNapsatZpravuDoKonzole -stavZpravy opravuji -nahradSloupec sloupec4
                $vysledek = fncInstalaceOvladace -nazevSouboruCAB $nazevSouboruCAB #
nainstalujeme ovladač do windows repozitáře
                try {
                    if ($vysledek -ge 1) {
                        # byl nainstalován ovladač do windows repozitáře
                        Add-PrinterDriver -Name $tiskarnaVzdalena.DriverName # přidá
ovladač do tiskové služby
                    }
                    else {
                        throw "Nepovedl se nainstalovat ovladač pomocí pnputil!"
                    }
                    subNapsatZpravuDoKonzole -stavZpravy opraveno -nahradSloupec sloupec4
                }
                catch {
                    subNapsatZpravuDoKonzole -stavZpravy neopraveno -nahradSloupec sloupec4
                    subZapsatLogDoTxt "start subKontrolaChybejicichOvladacu - ověřuji
ovladač pro: $($tiskarnaVzdalena.name) - neopraveno: $(fncVratitDuvodChyby -psitem
$psitem)"
                }
            }
        }
        else {
            subNapsatZpravuDoKonzole -stavZpravy neexistuje -nahradSloupec sloupec3
            subNapsatZpravuDoKonzole -stavZpravy opravuji -nahradSloupec sloupec4

            $ovladaceZrepozitareWindows = fncVratitOvladaceZrepozitare
            if ($ovladaceZrepozitareWindows) {
                $indexRepozitar =
$ovladaceZrepozitareWindows.OriginalFileName.indexof($infPath) # zeptam se, jestli
soubor.inf existuje v repozitari -1 neexistuje 0=< existuje
            }
            try {
                if ($indexRepozitar -ge 0) {
                    Add-PrinterDriver -Name $tiskarnaVzdalena.DriverName -ErrorAction
SilentlyContinue #jsou shodné verze, tak ovladač nainstaluji z repozitaře
                }
                else {
                    $vysledek = fncInstalaceOvladace -nazevSouboruCAB $nazevSouboruCAB
                    if ($vysledek -ge 1) {
                        Add-PrinterDriver -Name $tiskarnaVzdalena.DriverName -ErrorAction
Stop
                    }
                    else {
                        subZapsatLogDoTxt "start subKontrolaChybejicichOvladacu - ověřuji
ovladač pro: $($tiskarnaVzdalena.name) - neopraveno: Nepovedl se nainstalovat ovladač
pomocí pnputil!"
                        throw "Nepovedl se nainstalovat ovladač pomocí pnputil!"
                    }
                }
                subNapsatZpravuDoKonzole -stavZpravy opraveno -nahradSloupec sloupec4
            }
            catch {
                subNapsatZpravuDoKonzole -stavZpravy neopraveno -nahradSloupec sloupec4
                subZapsatLogDoTxt "start subKontrolaChybejicichOvladacu - ověřuji ovladač
pro: $($tiskarnaVzdalena.name) - neopraveno: $(fncVratitDuvodChyby -psitem $psitem)"
            }
        }
        if ($logujOperace) { subObsahKonzoleDoHtml }
    }
    subZapsatLogDoTxt " exit subKontrolaChybejicichOvladacu"
}
function subKontrolaLokalniKonfiguraceTiskaren {
    subZapsatLogDoTxt "start subKontrolaLokalniKonfiguraceTiskaren - call subprocess
subNacteniLokalnichTiskaren"
    subNacteniLokalnichTiskaren
    foreach ($tiskarnaVzdalena in $script:tiskarnyVzdalene) {
        $nazevKontrolovaneTiskarny = $tiskarnaVzdalena.Name
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
        subZapsatLogDoTxt "      subKontrolaLokalniKonfiguraceTiskaren - ověřuji
konfiguraci -  Předvolby: $nazevKontrolovaneTiskarny"
        #subNapsatZpravuDoKonzole -sloupec1 ' ' -zarovnatNaPredchoziSloupce:$true #slouží k
zarování následujícího sloupce
        subNapsatZpravuDoKonzole -sloupec1 'Ověřuji konfiguraci tiskárny:' -sloupec2
$nazevKontrolovaneTiskarny
        if ($logujOperace) { subObsahKonzoleDoHtml }
        ### začátek vlastnosti
        subNapsatZpravuDoKonzole -sloupec2 'Vlastnosti'
        subZapsatLogDoTxt "      subKontrolaLokalniKonfiguraceTiskaren - ověřuji
konfiguraci - Vlastnosti: $nazevKontrolovaneTiskarny"
        $indexLokalniVlastnosti =
$script:tiskarnyLokalni.Name.IndexOf($nazevKontrolovaneTiskarny)    #nalezneme pozadovany
port -1 port nenalezen 0<= port nalezen
        $indexVzdaleneVlastnosti =
$script:tiskarnyVzdalene.Name.IndexOf($nazevKontrolovaneTiskarny) #nalezneme pozadovany
port -1 port nenalezen 0<= port nalezen
        $tiskarnaLokalniVlastnosti = $script:tiskarnyLokalni[$indexLokalniVlastnosti]
        $tiskarnaVzdalenaVlastnosti = $script:tiskarnyVzdalene[$indexVzdaleneVlastnosti]
        [string]$tiskarnaVzdalenaVlastnostiName = $tiskarnaVzdalenaVlastnosti.name
        [string]$tiskarnaVzdalenaVlastnostiDriverName =
$tiskarnaVzdalenaVlastnosti.DriverName
        [string]$tiskarnaVzdalenaVlastnostiPortName = $tiskarnaVzdalenaVlastnosti.PortName
        [string]$tiskarnaVzdalenaVlastnostiLocation = $tiskarnaVzdalenaVlastnosti.Location
        [string]$tiskarnaVzdalenaVlastnostiComment = $tiskarnaVzdalenaVlastnosti.Comment
        [string]$tiskarnaVzdalenaVlastnostiPrintProcessor =
$tiskarnaVzdalenaVlastnosti.PrintProcessor
        [string]$tiskarnaVzdalenaVlastnostiPrintProcessorDataType =
$tiskarnaVzdalenaVlastnosti.Datatype
        [string]$tiskarnaVzdalenaVlastnostiSeparatorPageFile =
$tiskarnaVzdalenaVlastnosti.SeparatorPageFile

        if ($tiskarnaVzdalenaVlastnosti.SeparatorPageFile.length -eq 0) {
$tiskarnaVzdalenaVlastnosti.SeparatorPageFile = "" } # v případě nevyplnění na serveru
vrací $null, ale klientské stanice vrací prázdný string, tak $null změníme na prázdný
string pro porovnání
        if ($tiskarnaVzdalenaVlastnosti.Location.length -eq 0) {
$tiskarnaVzdalenaVlastnosti.Location = "" } # v případě nevyplnění na serveru vrací $null,
ale klientské stanice vrací prázdný string, tak $null změníme na prázdný string pro
porovnání
        if ($tiskarnaVzdalenaVlastnosti.Comment.length -eq 0) {
$tiskarnaVzdalenaVlastnosti.Comment = "" } # v případě nevyplnění na serveru vrací $null,
ale klientské stanice vrací prázdný string, tak $null změníme na prázdný string pro
porovnání
        if ($tiskarnaLokalniVlastnosti.SeparatorPageFile.length -eq 0) {
$tiskarnaLokalniVlastnosti.SeparatorPageFile = "" } # v případě nevyplnění na serveru vrací
$null, ale klientské stanice vrací prázdný string, tak $null změníme na prázdný string pro
porovnání
        if ($tiskarnaLokalniVlastnosti.Location.length -eq 0) {
$tiskarnaLokalniVlastnosti.Location = "" } # v případě nevyplnění na serveru vrací $null,
ale klientské stanice vrací prázdný string, tak $null změníme na prázdný string pro
porovnání
        if ($tiskarnaLokalniVlastnosti.Comment.length -eq 0) {
$tiskarnaLokalniVlastnosti.Comment = "" } # v případě nevyplnění na serveru vrací $null,
ale klientské stanice vrací prázdný string, tak $null změníme na prázdný string pro
porovnání

        if ($nastavovatSecurity) {
            [string]$tiskarnaVzdalenaVlastnostiPermissionSDDL =
$tiskarnaVzdalenaVlastnosti.PermissionSDDL
            $vysledek = Compare-Object -ReferenceObject $tiskarnaVzdalenaVlastnosti -
DifferenceObject $tiskarnaLokalniVlastnosti  -Property name, DriverName, PortName,
PermissionSDDL, Location, Comment, PrintProcessor, Datatype, SeparatorPageFile | % {
$_.SideIndicator -eq "=>" }
        }
        else {
            $vysledek = Compare-Object -ReferenceObject $tiskarnaVzdalenaVlastnosti -
DifferenceObject $tiskarnaLokalniVlastnosti  -Property name, DriverName, PortName,
Location, Comment, PrintProcessor, Datatype, SeparatorPageFile | % { $_.SideIndicator -eq
"=>" }
        }
        if ($vysledek -gt 0) {
            try {
                subNapsatZpravuDoKonzole -stavZpravy opravuji -nahradSloupec sloupec3
                if ($nastavovatSecurity) {
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
                    Set-Printer -ComputerName $env:computername -Name
$tiskarnaVzdalenaVlastnostiName -DriverName $tiskarnaVzdalenaVlastnostiDriverName -PortName
$tiskarnaVzdalenaVlastnostiPortName -PermissionSDDL
$tiskarnaVzdalenaVlastnostiPermissionSDDL -Location $tiskarnaVzdalenaVlastnostiLocation -
Comment $tiskarnaVzdalenaVlastnostiComment -PrintProcessor
$tiskarnaVzdalenaVlastnostiPrintProcessor -Datatype
$tiskarnaVzdalenaVlastnostiPrintProcessorDataType -SeparatorPageFile
$tiskarnaVzdalenaVlastnostiSeparatorPageFile -ErrorAction Stop
                }
                else {
                    Set-Printer -ComputerName $env:computername -Name
$tiskarnaVzdalenaVlastnostiName -DriverName $tiskarnaVzdalenaVlastnostiDriverName -PortName
$tiskarnaVzdalenaVlastnostiPortName -Location $tiskarnaVzdalenaVlastnostiLocation -Comment
$tiskarnaVzdalenaVlastnostiComment -PrintProcessor
$tiskarnaVzdalenaVlastnostiPrintProcessor -Datatype
$tiskarnaVzdalenaVlastnostiPrintProcessorDataType -SeparatorPageFile
$tiskarnaVzdalenaVlastnostiSeparatorPageFile -ErrorAction Stop
                }
                subNapsatZpravuDoKonzole -stavZpravy opraveno -nahradSloupec sloupec3
            }
            catch {
                subNapsatZpravuDoKonzole -stavZpravy neopraveno -nahradSloupec sloupec3
                subZapsatLogDoTxt "      subKontrolaLokalniKonfiguraceTiskaren - ověřuji
konfiguraci - Vlastnosti: $nazevKontrolovaneTiskarny - chyba: $(fncVratitDuvodChyby -psitem
$psitem)"
            }
        }
        else {
            subNapsatZpravuDoKonzole -stavZpravy shoda -nahradSloupec sloupec3
        }

        ### začátek předvolby
        subNapsatZpravuDoKonzole -sloupec2 'Předvolby'
        if ($chybaNacteniVzdaleneKonfigurace -or ($chybaNacteniLokalniKonfigurace -and -not
$chybaNacteniLokalniKonfiguraceNektere)) {
            subNapsatZpravuDoKonzole -sloupec3 'přeskakuji - chyba načtení předchozí
vzdálené/lokální konfigurace' -nahradSloupec sloupec3 -barva Yellow
            subZapsatLogDoTxt "      subKontrolaLokalniKonfiguraceTiskaren - ověřuji
konfiguraci - Předvolby: $nazevKontrolovaneTiskarny - přeskakuji - chyba načtení předchozí
vzdálené/lokální konfigurace"
        }
        else {
            $indexLokalniPredvolby =
$script:tiskarnyLokalniPredvolby.PrinterName.IndexOf($nazevKontrolovaneTiskarny)
#nalezneme pozadovany port -1 port nenalezen 0<= port nalezen
            if ($indexLokalniPredvolby -eq -1) {
                subNapsatZpravuDoKonzole -sloupec3 'přeskakuji - chyba načtení předchozí
vzdálené/lokální konfigurace' -nahradSloupec sloupec3 -barva Yellow
                subZapsatLogDoTxt "      subKontrolaLokalniKonfiguraceTiskaren - ověřuji
konfiguraci - Předvolby: $nazevKontrolovaneTiskarny - přeskakuji - chyba načtení předchozí
vzdálené/lokální konfigurace"
            }
            else {
                $tiskarnaLokalniPredvolby =
$script:tiskarnyLokalniPredvolby[$indexLokalniPredvolby]
                $indexVzdalenePredvolby =
$tiskarnyVzdalenePredvolby.PrinterName.IndexOf($nazevKontrolovaneTiskarny) #nalezneme
pozadovany port -1 port nenalezen 0<= port nalezen
                $tiskarnaVzdalenaPredvolby =
$tiskarnyVzdalenePredvolby[$indexVzdalenePredvolby]
                [string]$tiskarnaVzdalenaPredvolbyPrinterName =
$tiskarnaVzdalenaPredvolby.PrinterName
                [boolean]$tiskarnaVzdalenaPredvolbyCollate =
$tiskarnaVzdalenaPredvolby.Collate
                [boolean]$tiskarnaVzdalenaPredvolbyColor = $tiskarnaVzdalenaPredvolby.Color
                $tiskarnaVzdalenaPredvolbyDuplexingMode =
$tiskarnaVzdalenaPredvolby.DuplexingMode
                $tiskarnaVzdalenaPredvolbyPaperSize = $tiskarnaVzdalenaPredvolby.PaperSize
                [string]$tiskarnaVzdalenaPredvolbyPrintTicketXml =
$tiskarnaVzdalenaPredvolby.PrintTicketXml
                $vysledek = Compare-Object -ReferenceObject $tiskarnaVzdalenaPredvolby -
DifferenceObject $tiskarnaLokalniPredvolby  -Property PrinterName, Collate, Color,
DuplexingMode, PaperSize | % { $_.SideIndicator -eq "=>" }
                if ($vysledek -gt 0) {
                    try {
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
                        subNapsatZpravuDoKonzole -stavZpravy opravuji -nahradSloupec
sloupec3
                        Set-PrintConfiguration -PrinterName $nazevKontrolovaneTiskarny -
Collate $tiskarnaVzdalenaPredvolby.Collate -Color $tiskarnaVzdalenaPredvolby.Color -
DuplexingMode $tiskarnaVzdalenaPredvolby.DuplexingMode -PaperSize
$tiskarnaVzdalenaPredvolby.PaperSize -PrintTicketXml
$tiskarnaVzdalenaPredvolby.PrintTicketXml -ErrorAction Stop
                        subNapsatZpravuDoKonzole -stavZpravy opraveno -nahradSloupec
sloupec3
                    }
                    catch {
                        try {
                            Set-PrintConfiguration -PrinterName $nazevKontrolovaneTiskarny
-CimSession $env:computername -Collate $tiskarnaVzdalenaPredvolby.Collate -Color
$tiskarnaVzdalenaPredvolby.Color -DuplexingMode $tiskarnaVzdalenaPredvolby.DuplexingMode -
PaperSize $tiskarnaVzdalenaPredvolby.PaperSize -PrintTicketXml
$tiskarnaVzdalenaPredvolby.PrintTicketXml -ErrorAction Stop
                        }
                        catch {
                            subNapsatZpravuDoKonzole -stavZpravy neopraveno -nahradSloupec
sloupec3
                            subZapsatLogDoTxt "      subKontrolaLokalniKonfiguraceTiskaren
- ověřuji konfiguraci - Předvolby: $nazevKontrolovaneTiskarny - chyba:
$(fncVratitDuvodChyby -psitem $psitem)"
                        }
                    }
                }
                else {
                    subNapsatZpravuDoKonzole -stavZpravy shoda -nahradSloupec sloupec3
                }
            }
        }
        if ($logujOperace) { subObsahKonzoleDoHtml }
    }
    subZapsatLogDoTxt " exit subKontrolaLokalniKonfiguraceTiskaren"
}
function subUkonceniProgramu {
    $osName = (Get-ItemProperty -path "HKLM:\Software\Microsoft\Windows
NT\CurrentVersion").ProductName
    $konecStopky = fncStopky
    subNapsatZpravuDoKonzole -sloupec1 '-----------------' -sloupec2  'DOKONČENO ----------
-------------' -barva Green -zarovnatNaPredchoziSloupce:$false
    subNapsatZpravuDoKonzole -sloupec1 'Název počítače:' -sloupec2 $env:computername
    subNapsatZpravuDoKonzole -sloupec1 'Systém:' -sloupec2 $osName
    subNapsatZpravuDoKonzole -sloupec1 'Začátek programu:' -sloupec2
$datumZacatek.tostring("dd.MM.yyyy HH:mm:ss")
    subNapsatZpravuDoKonzole -sloupec1 'Doba běhu:' -sloupec2 $konecStopky
    $Statistika = @"
---------------- DOKONČENO -----------------------
                    Verze programu:   $script:verzeProgramu
                    Název počítače:  $env:computername
                    Systém:          $osName
                    Začátek programu: $($datumZacatek.tostring("dd.MM.yyyy HH:mm:ss"))
                    Doba běhu:       $konecStopky
"@
    subZapsatLogDoTxt $Statistika
    subObsahKonzoleDoHtml
    exit
}
function fncDoplnitMezeryZaText {
    Param(
        [int]$PocetPredchozichZnaku = 0,
        #[boolean]$zustanNaRadku = $false,
        [string]$zprava = "OK"
    )
    if ($PocetPredchozichZnaku -gt 0) {
        $pocetZnakuZaZpravou = $PocetPredchozichZnaku - $zprava.Length
        if ($pocetZnakuZaZpravou -gt 0) {
            $mezery = fncDoplnitMezery -pocetMezer $pocetZnakuZaZpravou
        }
    }
    #return ($kurzorZpet + $zprava + $mezery)
    return ($zprava + $mezery)
}
function fncDoplnitMezery {
    Param(
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
        [int]$pocetMezer = 0
    )
    $mezery = ''
    if ($pocetMezer -gt 0) {
        for ($i = 0; $i -lt $pocetMezer; $i++) {
            $mezery += ' '
            #$mezery += 'X'
        }
    }
    return $mezery
}
function subNastavitKurzorNaSloupec {
    Param(
        [Parameter(Mandatory = $true)]
        [ValidateSet('sloupec1', 'sloupec2', 'sloupec3', 'sloupec4')]
        [System.String]$naSloupec
    )
    $aktualniPozice = $host.UI.RawUI.CursorPosition
    switch ($naSloupec) {
        'sloupec1' {
            $pozadovanaPoziceX = 0
        }
        'sloupec2' {
            $pozadovanaPoziceX = $script:posledniDelkaSloupce['sloupec1'] + 1
        }
        'sloupec3' {
            $pozadovanaPoziceX = $script:posledniDelkaSloupce['sloupec1'] +
$script:posledniDelkaSloupce['sloupec2'] + 2

        }
        'sloupec4' {
            $pozadovanaPoziceX = $script:posledniDelkaSloupce['sloupec1'] +
$script:posledniDelkaSloupce['sloupec2'] + $script:posledniDelkaSloupce['sloupec3'] + 3
        }
        default { $pozadovanaPoziceX = 0 }
    }
    #zjistíme velikost nastaveného okna terminálu a za okrouhlíme jeho hodnotu dolu na celé
číslo
    [int]$maximalniPocetZnaku = $([math]::Floor($($script:velikostOknaRozhrani -replace
",","."")))
    if ($pozadovanaPoziceX -ge $maximalniPocetZnaku) {
        # zjistíme o kolikrát je větši požadovaná pozice než povolených
$maximalniPocetZnaku a nastavíme hodnotu pro odečtení z požadované pozice, abychom dostali
        # vypočtenou novou pozici na ose x
        $vypoctenaMaximalniHodnotaX = $maximalniPocetZnaku *
[math]::Floor($pozadovanaPoziceX / $maximalniPocetZnaku)
        #protože systém automaticky zalomí řetezce nad hodnotou $maximalniPocetZnaku je
nutné nastavit správnou hodnotu pozice x
        #v rozmezí 0 až $maximalniPocetZnaku (systémový účet má povolených pouze 128 znaků
šířky, hodnota pozice x nad 128 vyvolá chybu)
        #odečteme $vypoctenaMaximalniHodnotaX od $pozadovanaPoziceX a dostaneme hodnotu v
povoleném rozsahu.
        $pozadovanaPoziceX = $pozadovanaPoziceX - $vypoctenaMaximalniHodnotaX

        if ($NahrazenyText.length -ge $script:posledniDelkaSloupce[$naSloupec]) {
            $zkrat = $NahrazenyText.length - $script:posledniDelkaSloupce[$naSloupec]
        }
        else {
            $zkrat = 0
        }
        if ($naSloupec -eq 'sloupec3' -and $pozadovanaPoziceX -ge $($maximalniPocetZnaku -
$zkrat)) {
            $pozadovanaPoziceX = $script:posledniDelkaSloupce['sloupec1']
        }

    }


    if ($aktualniPozice.y -eq 0) {
        $Host.UI.RawUI.CursorPosition =
[System.Management.Automation.Host.Coordinates]::new($pozadovanaPoziceX, $aktualniPozice.Y)
    }
    else {
        $Host.UI.RawUI.CursorPosition =
[System.Management.Automation.Host.Coordinates]::new($pozadovanaPoziceX,
$($aktualniPozice.y - 1))
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
    }
}
function subNapsatZpravuDoKonzole {
    Param(
        [int]$pocetSloupcu = 2,
        [Switch]$zarovnatNaPredchoziSloupce = $true,
        [System.ConsoleColor]$barva,
        [string]$sloupec1,
        [string]$sloupec2,
        [string]$sloupec3,
        [string]$sloupec4,
        [string]$sloupec5,
        [ValidateSet('sloupec1', 'sloupec2', 'sloupec3', 'sloupec4', 'sloupec5')]
        [System.String]$nahradSloupec,
        [ValidateSet('chyba', 'ok', 'opraveno', 'neopraveno', 'server přetížen',
'nenalezeny', 'čekejte', 'shoda', 'rozdíl', 'opravuji', 'nenalezena', 'neexistuje')]
        [System.String]$stavZpravy
    )
    $aktualniTextSloupce = @{}
    $aktualniTextSloupce['sloupec1'] = $sloupec1
    $aktualniTextSloupce['sloupec2'] = $sloupec2
    $aktualniTextSloupce['sloupec3'] = $sloupec3
    $aktualniTextSloupce['sloupec4'] = $sloupec4
    $aktualniTextSloupce['sloupec5'] = $sloupec5
    $aktualniDelkaSloupce = @{}
    $aktualniDelkaSloupce['sloupec1'] = $aktualniTextSloupce['sloupec1'].Length
    $aktualniDelkaSloupce['sloupec2'] = $aktualniTextSloupce['sloupec2'].Length
    $aktualniDelkaSloupce['sloupec3'] = $aktualniTextSloupce['sloupec3'].Length
    $aktualniDelkaSloupce['sloupec4'] = $aktualniTextSloupce['sloupec4'].Length
    $aktualniDelkaSloupce['sloupec5'] = $aktualniTextSloupce['sloupec5'].Length
    if ($stavZpravy -and $nahradSloupec) {
        switch ($stavZpravy) {
            # text zprávy za $aktualniTextSloupce[$nahradSloupec] =  lze nahradit
jakýmkoliv textem
            'chyba' {
                $barva = [System.ConsoleColor]::Red
                $aktualniTextSloupce[$nahradSloupec] = 'CHYBA - ' + $(fncVratitDuvodChyby -
psitem $Error[0])
                break
            }
            'opraveno' {
                $barva = [System.ConsoleColor]::Green
                $aktualniTextSloupce[$nahradSloupec] = 'opraveno'
                break
            }
            'neopraveno' {
                $barva = [System.ConsoleColor]::Red
                $aktualniTextSloupce[$nahradSloupec] = 'neopraveno'
                break
            }
            'server přetížen' {
                $barva = [System.ConsoleColor]::Red
                $aktualniTextSloupce[$nahradSloupec] = 'server přetížen'
                break
            }
            'ok' {
                $barva = [System.ConsoleColor]::Green
                $aktualniTextSloupce[$nahradSloupec] = 'OK'
                break
            }
            'nenalezeny' {
                $barva = [System.ConsoleColor]::Green
                $aktualniTextSloupce[$nahradSloupec] = 'nenalezeny'
                break
            }
            'čekejte' {
                $barva = [System.ConsoleColor]::Yellow
                $aktualniTextSloupce[$nahradSloupec] = 'čekejte'
                break
            }
            'shoda' {
                $barva = [System.ConsoleColor]::Green
                $aktualniTextSloupce[$nahradSloupec] = 'shoda'
                break
            }
            'rozdíl' {
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
                $barva = [System.ConsoleColor]::Yellow
                $aktualniTextSloupce[$nahradSloupec] = 'rozdíl'
                break
            }
            'opravuji' {
                $barva = [System.ConsoleColor]::Yellow
                $aktualniTextSloupce[$nahradSloupec] = 'opravuji'
                break
            }
            'nenalezena' {
                $barva = [System.ConsoleColor]::Yellow
                $aktualniTextSloupce[$nahradSloupec] = 'nenalezena'
                break
            }
            'neexistuje' {
                $barva = [System.ConsoleColor]::Red
                $aktualniTextSloupce[$nahradSloupec] = 'neexistuje'
                break
            }
            default { break }
        }
        $aktualniDelkaSloupce[$nahradSloupec] = $aktualniTextSloupce[$nahradSloupec].Length
    }
    if ($barva) { $param += @{ForegroundColor = $barva } } # v případě vyplněné barvy, tak
vložíme do cmldetu jeho parametr
    if ($nahradSloupec) {
        # nahrazuje parametr NoNewLine, tzn. vrátí se na předchozí řádek
        $pocetZnakuNahrazovanehoSloupce = $script:posledniDelkaSloupce[$nahradSloupec]
        $script:NahrazenyText = fncDoplnitMezeryZaText -PocetPredchozichZnaku
$pocetZnakuNahrazovanehoSloupce -zprava $aktualniTextSloupce[$nahradSloupec]
        subNastavitKurzorNaSloupec -naSloupec $nahradSloupec
        write-host "$("{0,-$pocetZnakuNahrazovanehoSloupce}" -f $NahrazenyText)" @param #
vypisujeme do konzole definovanou zprávu
        $aktualniTextSloupce[$nahradSloupec] = $NahrazenyText
        switch ($nahradSloupec) {
            'sloupec1' {
                $script:posledniDelkaSloupce['sloupec1'] =
$aktualniDelkaSloupce['sloupec1']; $script:posledniTextSloupce['sloupec1'] =
$aktualniTextSloupce['sloupec1']
            }
            'sloupec2' {
                $script:posledniDelkaSloupce['sloupec2'] =
$aktualniDelkaSloupce['sloupec2']; $script:posledniTextSloupce['sloupec2'] =
$aktualniTextSloupce['sloupec2']
            }
            'sloupec3' {
                $script:posledniDelkaSloupce['sloupec3'] =
$aktualniDelkaSloupce['sloupec3']; $script:posledniTextSloupce['sloupec3'] =
$aktualniTextSloupce['sloupec3']
            }
            'sloupec4' {
                $script:posledniDelkaSloupce['sloupec4'] =
$aktualniDelkaSloupce['sloupec4']; $script:posledniTextSloupce['sloupec4'] =
$aktualniTextSloupce['sloupec4']
            }
            'sloupec5' {
                $script:posledniDelkaSloupce['sloupec5'] =
$aktualniDelkaSloupce['sloupec5']; $script:posledniTextSloupce['sloupec5'] =
$aktualniTextSloupce['sloupec5']
            }
        }
    }
    else {
        #pokud jsou vyplněné parametry sloupec1 až sloupec5, tak si uložíme jejich
velikosti a hodnoty
        if ($zarovnatNaPredchoziSloupce.IsPresent) {
            if ($aktualniDelkaSloupce['sloupec1'] -gt
$script:posledniDelkaSloupce['sloupec1']) {
                $script:posledniDelkaSloupce['sloupec1'] =
$aktualniDelkaSloupce['sloupec1']; $script:posledniTextSloupce['sloupec1'] =
$aktualniTextSloupce['sloupec1']
            }
            if ($aktualniDelkaSloupce['sloupec2'] -gt
$script:posledniDelkaSloupce['sloupec2']) {
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
                $script:posledniDelkaSloupce['sloupec2'] =
$aktualniDelkaSloupce['sloupec2']; $script:posledniTextSloupce['sloupec2'] =
$aktualniTextSloupce['sloupec2']
            }
            if ($aktualniDelkaSloupce['sloupec3'] -gt
$script:posledniDelkaSloupce['sloupec3']) {
                $script:posledniDelkaSloupce['sloupec3'] =
$aktualniDelkaSloupce['sloupec3']; $script:posledniTextSloupce['sloupec3'] =
$aktualniTextSloupce['sloupec3']
            }
            if ($aktualniDelkaSloupce['sloupec4'] -gt
$script:posledniDelkaSloupce['sloupec4']) {
                $script:posledniDelkaSloupce['sloupec4'] =
$aktualniDelkaSloupce['sloupec4']; $script:posledniTextSloupce['sloupec4'] =
$aktualniTextSloupce['sloupec4']
            }
            if ($aktualniDelkaSloupce['sloupec5'] -gt
$script:posledniDelkaSloupce['sloupec5']) {
                $script:posledniDelkaSloupce['sloupec5'] =
$aktualniDelkaSloupce['sloupec5']; $script:posledniTextSloupce['sloupec5'] =
$aktualniTextSloupce['sloupec5']
            }
            #if ($posledniDelkaSloupce['sloupec2'] -ge 70){write-host
$posledniDelkaSloupce['sloupec2'] -ForegroundColor Yellow}
            if ($script:posledniDelkaSloupce['sloupec2'] -ge 70 -and
$aktualniDelkaSloupce['sloupec2'] -eq 0){$script:posledniDelkaSloupce['sloupec2'] = 0}
        }
        else {
            if ($aktualniDelkaSloupce['sloupec1'] -gt 0) {
                $script:posledniDelkaSloupce['sloupec1'] =
$aktualniDelkaSloupce['sloupec1']; $script:posledniTextSloupce['sloupec1'] =
$aktualniTextSloupce['sloupec1']
            }
            if ($aktualniDelkaSloupce['sloupec2'] -gt 0) {
                $script:posledniDelkaSloupce['sloupec2'] =
$aktualniDelkaSloupce['sloupec2']; $script:posledniTextSloupce['sloupec2'] =
$aktualniTextSloupce['sloupec2']
            }
            if ($aktualniDelkaSloupce['sloupec3'] -gt 0) {
                $script:posledniDelkaSloupce['sloupec3'] =
$aktualniDelkaSloupce['sloupec3']; $script:posledniTextSloupce['sloupec3'] =
$aktualniTextSloupce['sloupec3']
            }
            if ($aktualniDelkaSloupce['sloupec4'] -gt 0) {
                $script:posledniDelkaSloupce['sloupec4'] =
$aktualniDelkaSloupce['sloupec4']; $script:posledniTextSloupce['sloupec4'] =
$aktualniTextSloupce['sloupec4']
            }
            if ($aktualniDelkaSloupce['sloupec5'] -gt 0) {
                $script:posledniDelkaSloupce['sloupec5'] =
$aktualniDelkaSloupce['sloupec5']; $script:posledniTextSloupce['sloupec5'] =
$aktualniTextSloupce['sloupec5']
            }
        }
        write-host "$("{0,-$($script:posledniDelkaSloupce['sloupec1'])} {1,-
$($script:posledniDelkaSloupce['sloupec2'])} {2,-
$($script:posledniDelkaSloupce['sloupec3'])} {3,-
$($script:posledniDelkaSloupce['sloupec4'])} {4,-
$($script:posledniDelkaSloupce['sloupec5'])}" -f
            $aktualniTextSloupce['sloupec1'], $aktualniTextSloupce['sloupec2'],
$aktualniTextSloupce['sloupec3'], $aktualniTextSloupce['sloupec4'],
$aktualniTextSloupce['sloupec5'])" @param # vypisujeme do konzole definovanou zprávu
    }
}
function subKontrolaRegistru {
    subZapsatLogDoTxt "start subKontrolaRegistru"
    [boolean]$readonly = 0
    subNapsatZpravuDoKonzole -sloupec1 'Kontroluji pozůstatky registrů odpojených tiskáren'
    subNapsatZpravuDoKonzole -sloupec2 'čekejte' -nahradSloupec sloupec2 -barva Yellow
    if ($logujOperace) { subObsahKonzoleDoHtml }

    $ignorovatPripojeneTiskarnyAservery = get-childitem
"HKLM:SOFTWARE\Policies\Microsoft\Windows NT\Printers\PushedPrinterConnectionStore" -
Recurse | ? { $_.property } | % { [pscustomobject]@{"server" =
$_.getvalue('serverName').replace("\\", ""); "tiskarna" = $_.GetValue('printerName') } }
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
    [string]$pripojeneTiskarnyRegEx = $ignorovatPripojeneTiskarnyAservery.tiskarna -join
"|" # z nalezených tiskáren vytvoříme regulární výraz s operandem OR
    if ($pripojeneTiskarnyRegEx.Length -eq 0) { $pripojeneTiskarnyRegEx =
"NejsouZadnePripojeneTiskarny" }
    [string]$pripojeneServeryRegEx = ($ignorovatPripojeneTiskarnyAservery | select -
ExpandProperty server -Unique) -join "|" # z nalezených serverů vytvoříme regulární výraz s
operandem OR
    if ($pripojeneServeryRegEx.Length -eq 0) { $pripojeneServeryRegEx =
"NejsouZadnePripojeneServery" }
    $registryTiskarenKodebrani = Get-childItem "HKLM:\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Print\Providers" -Recurse -Depth 4 | ? { $_.PSChildName -match
$odebratRegistrTiskovychServeru -and $_.PSChildName -notmatch $pripojeneTiskarnyRegEx -and
$_.Name -match "\\Connections\\" } | select @{name = "PSChildNameKlic"; e = {
$_.PSChildName.replace(",,", "") } }, @{name = "nameCesta"; e = { $_.name } }
    $registryServeruKodebrani = Get-childItem "HKLM:\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Print\Providers" -Recurse -Depth 2 | ? { $_.PSChildName -match
$odebratRegistrTiskovychServeru -and $_.PSChildName -notmatch $pripojeneServeryRegEx -and
$_.Name -match "\\Servers\\" } | select @{name = "PSChildNameKlic"; e = {
$_.PSChildName.replace(",,", "") } }, @{name = "nameCesta"; e = { $_.name } }
    if (($registryTiskarenKodebrani | measure).count -gt 0 -or (($registryServeruKodebrani
| measure).count -gt 0)) {
        ###subNapsatZpravuDoKonzole -sloupec3 'opravuji' -nahradSloupec sloupec3 -barva
Yellow
        subNapsatZpravuDoKonzole -stavZpravy opravuji -nahradSloupec sloupec2
        if ($logujOperace) { subObsahKonzoleDoHtml }
        try {
            if (($registryTiskarenKodebrani | measure).count -gt 0) {
                $registryTiskarenKodebrani.nameCesta.replace("HKEY_LOCAL_MACHINE\",
"HKLM:\") | % {
                    if ($readonly) {
                        subNapsatZpravuDoKonzole -sloupec3 (registryTiskarenKodebrani $_) -
nahradSloupec sloupec3
                    }
                    else {
                        remove-item $_ -Force -Recurse -ErrorAction Stop -WarningAction
SilentlyContinue -InformationAction SilentlyContinue
                    }
                }
            }
            if (($registryServeruKodebrani | measure).count -gt 0) {
                $registryServeruKodebrani.nameCesta.replace("HKEY_LOCAL_MACHINE\",
"HKLM:\") | % {
                    if ($readonly) {
                        subNapsatZpravuDoKonzole -sloupec4 (registryServeruKodebrani $_) -
nahradSloupec sloupec4
                    }
                    else {
                        remove-item $_ -Force -Recurse -ErrorAction Stop -WarningAction
SilentlyContinue -InformationAction SilentlyContinue
                    }
                }
                if (!$readonly) {
                    Stop-Service Spooler -Force
                    Start-Service Spooler -ErrorAction SilentlyContinue -WarningAction
SilentlyContinue
                    Start-Service SAPSprint -ErrorAction SilentlyContinue -WarningAction
SilentlyContinue # zapínám tiskovou službu SAPu, pokud existuje
                }
            }
            subNapsatZpravuDoKonzole -stavZpravy opraveno -nahradSloupec sloupec2
        }
        catch {
            subNapsatZpravuDoKonzole -stavZpravy neopraveno -nahradSloupec sloupec2
        }
    }
    else {
        ###subNapsatZpravuDoKonzole -sloupec2 'nenalezeny' -nahradSloupec sloupec2 -barva
Green
        subNapsatZpravuDoKonzole -stavZpravy nenalezeny -nahradSloupec sloupec2
    }
    if ($logujOperace) { subObsahKonzoleDoHtml }
    subZapsatLogDoTxt " exit subKontrolaRegistru"
}
function subOpravneniSpusteniUzivatelem {
    # funkce povolí nebo zakáže spouštění úlohy uživatelem
    Param(
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
        $povolit = $false
    )
    subZapsatLogDoTxt "start subOpravneniSpusteniUzivatelem"
    subNapsatZpravuDoKonzole -sloupec1 "Ověřuji možnost spustit úlohu uživatelem"
    $planovac = new-object -ComObject "Schedule.service"
    $planovac.connect()
    try {
        $uloha = $planovac.getfolder("").gettask("PowerShell-SpravaTiskaren-default")
# načítáme výchozí úlohu, pokud neexistuje, tak se pokusíme načíst úlohu s více tiskárnami
    }
    catch {
        try {
            $uloha = $planovac.getfolder("").gettask("PowerShell-SpravaTiskaren")
        }
        catch {
            subNapsatZpravuDoKonzole -sloupec2 'neověřeno - úloha nenalezena' -
nahradSloupec sloupec2 -barva Yellow
            return
        }
    }
    try {
        $sec = $uloha.getsecuritydescriptor(0xF)                    # načteme oprávnění Sddl
        #$nastavenaOpravneni=ConvertFrom-SddlString -Sddl $sec    # konverze do čitelné
podoby
        #$nastavenaOpravneni.DiscretionaryAcl                       # vypíšeme nastavená
oprávnění
        $rightsAU = '(A;;0x1200a9;;;AU)'  # povolíme spouštění pro Authenticated Users
    }
    catch {
        subNapsatZpravuDoKonzole -sloupec2 'ověření odepřeno' -nahradSloupec sloupec2 -
barva Yellow
        return
    }
    if ($povolit) {
        try {
            if (!($sec -match $rightsAU)) {
                # pokud není ještě nastavené oprávnění, tak ho nastavíme
                $sec += $rightsAU
                $uloha.setSecurityDescriptor($sec, 0)
            }
            subNapsatZpravuDoKonzole -sloupec2 'povoleno' -nahradSloupec sloupec2 -barva
Green
        }
        catch {
            subNapsatZpravuDoKonzole -sloupec2 'ověření odepřeno' -nahradSloupec sloupec2 -
barva Yellow
        }
    }
    else {
        try {
            if ($sec -match $rightsAU) {
                # pokud je nastavené oprávnění, tak ho zrušíme
                $sec = $sec.Replace($rightsAU, '')
                $uloha.setSecurityDescriptor($sec, 0)
            }
            subNapsatZpravuDoKonzole -sloupec2 'nepovoleno' -barva Red
        }
        catch {
            $chyba = $psitem
            subNapsatZpravuDoKonzole -sloupec2 'nelze ověřit - přístup odepřen' -barva
Yellow
        }
    }
    subZapsatLogDoTxt " exit subOpravneniSpusteniUzivatelem"
}
function fncUmistitSpecialniZnakRegEx {
    Param(
        [Parameter(Mandatory = $true)]
        [string]$hodnota,
        [Parameter(Mandatory = $true)]
        [ValidateSet('na začátku', 'na konci', 'kdekoliv', 'přesná shoda')]
        $poziceVzoru
    )
    $hodnotaTMP = $hodnota.Split("|")
    [string]$vzorRegEx = ''
    switch ($poziceVzoru) {
```

```
        'na začátku' {
            $hodnotaTMP | % { $vzorRegEx += '^' + $_ + '|' }  #ke kazdé hodnotě přidat na
začátek ^ a na konec |
            $vzorRegEx = $vzorRegEx.Substring(0, $($vzorRegEx.Length - 1)) # odstranit
nadbytečný znak
            break
        }
        'na konci' {
            $hodnotaTMP | % { $vzorRegEx += '|' + $_ + '$' } #ke kazdé hodnotě přidat na
začátek | a na konec $
            $vzorRegEx = $vzorRegEx.Substring(1) # odstranit nadbytečný znak na začátku
            break
        }
        'kdekoliv' { $vzorRegEx = $hodnota } # vrací nezměněnou hodnotu
        'přesná shoda' {
            $hodnotatmp | % { $vzorRegEx += '|^' + $_ + '$' } #ke kazdé hodnotě přidat na
začátek |^ a na konec $
            $vzorRegEx = $vzorRegEx.Substring(1) #odstranit nadbytečný znak na začátku
        }
        default { $vzorRegEx = $hodnota }
    }

    return $vzorRegEx
}
function fncPrevodNaRegEx {
    Param(
        [Parameter(Mandatory = $true)]
        [string]$hodnota,
        [switch]$escape = $false,
        [Parameter(Mandatory = $true)]
        [ValidateSet('na začátku', 'na konci', 'kdekoliv', 'přesná shoda')]
        $poziceVzoru
    )
    #    '^'+$([regex]::Escape($HledajiciADskupina))+'$'
    $hodnota = $hodnota.Replace(",", "|")
    if ($escape.IsPresent) { $hodnota = [regex]::Escape($hodnota) }
    switch ($poziceVzoru) {
        'na začátku' { $vzorRegEx = fncUmistitSpecialniZnakRegEx -hodnota $hodnota -
poziceVzoru $poziceVzoru; break }
        'na konci' { $vzorRegEx = fncUmistitSpecialniZnakRegEx -hodnota $hodnota -
poziceVzoru $poziceVzoru; break }
        'kdekoliv' { $vzorRegEx = $hodnota }
        'přesná shoda' { $vzorRegEx = fncUmistitSpecialniZnakRegEx -hodnota $hodnota -
poziceVzoru $poziceVzoru; break }
        default { $vzorRegEx = $hodnota }
    }
    return $vzorRegEx
}

function subNastavitPromenneKdeBeziProgram {
    if (!($PSScriptRoot -eq $cestaPracovnihoAdresareprogramu)) {
        # ověřujeme, zdali je spuštěný program v pracovním adresáři
        if (!($spustenJakoNovaInstance)) {
            Write-Warning "Program není spuštěn z pracovního adresáře!`nZ tohoto důvodu
není zaručena aktuálnost programu a jeho správná funkčnost."
            if (!($host.name -match "ISE|Studio")) { timeout /t 10 }
            Clear-Host
        }
        $cestaPracovnihoAdresareprogramu = $PSScriptRoot #přepíšeme cestu puvodního
pracovního adresáře na cestu aktuálně spuštěného programu - týká se ukládání logu a
ovladačů
        $script:spustenMimoPracovniAdresar = $true
    }
}


#------------------------------začátek běhu programu------------------------------

subInicializacePracovnihoProstredi

subPripojeniKserveru

subNacteniVzdalenychTiskaren
subNacteniVzdaleneKonfiguraceTiskaren # Provede načtení konfigurace požadovaných tiskáren z
tiskového serveru.
subNacteniVzdalenychOvladacu
```

Příloha 5 – Zdrojový kód: Správa tiskáren

```
subNacteniVzdalenychPortu

subNacteniLokalnichOvladacu
subNacteniLokalnichPortu
subNacteniLokalnichTiskaren

subKontrolaChybejicichOvladacu # Provede kontrolu chybějících nebo rozdílných ovladačů a v
případě potřeby je doinstaluje nebo aktualizuje.
subKontrolaChybejicichPortu # Provede kontrolu chybějících nebo rozdílných tiskových portů
a v případě potřeby je doinstaluje nebo aktualizuje.
subKontrolaChybejicichTiskaren # Provede kontrolu chybějících tiskáren a v případě potřeby
je doinstaluje.
subNacteniLokalniKonfiguraceTiskaren # Provede načtení konfigurace aktuálně nainstalovaných
tiskáren.
subKontrolaLokalniKonfiguraceTiskaren # "Provede porovnání konfigurace lokálních a
vzdálených tiskáren. Pokud najde rozdíly, upraví konfiguraci lokálních tiskáren podle
vzdálených."

subKontrolaRegistru # Zkontroluje v registru pozůstatky informací odebraných tiskáren a v
případě nalezení je odstraní.
subOdebraniNespravovanychTiskaren
subOdebraniNespravovanychPortu
subOdebraniNepotrebnychOvladacu

subUkonceniProgramu
```