VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Nástroje pro správu OS Windows Server

Tools for Managing OS Windows Server

2014 Petr Šupčík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Petr Šupčík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Nástroje pro správu OS Windows Server
Tools for Managing OS Windows Server

Zásady pro vypracování:

Cílem bakalářské práce je získat přehled v oblasti administrace Windows Server, implementovat webové nástroje umožňující správu uživatelských účtů.

1. Nastudovat problematiku Active Directory (AD), se zaměřením na vzdálenou správu přes zabezpečené připojení.
2. Pro vytvoření webové aplikace použít ASP.NET.
3. Umožňovat uživatelům aktualizaci údajů jejich účtů v AD, resetovat si vlastní heslo.
4. Administrátoři budou moci aktualizovat údaje uživatelům, zobrazit statistiky používání aplikace.
5. Otestovat funkčnosti aplikace.

Seznam doporučené odborné literatury:

[1] Brian Desmond, Joe Richards, Robbie Allen and Alistair G. Lowe-Norris. Active Directory: Designing, Deploying, and Running Active Directory, ISBN-10: 1449320023
[2] Kenneth Schaefer, Jeff Cochran, Scott Forsyth and Dennis Glendenning. Professional Microsoft IIS 8, ISBN-10: 1118388046
[3] Zambon, Giulio. Beginning JSP, JSF and Tomcat: Java Web Development, ASIN: B009SRRCLQ

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Lukáš Vojáček**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 7. května 2014 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2014 .....

Děkuji svému vedouciímu bakalářské práce, panu Ing. Lukáši Vojáčkovi, za odborné rady, konzultační hodiny a cenné připomínky při zpracování práce.

Abstrakt

Cílem práce je poskytnout uživatelům Active Directory domény možnost vyresetovat a změnit heslo v Active Directory. Administrátorům Active directory je umožněno základní nastavení domény a samotné aplikace. Resetování hesla je v aplikaci vyřešeno pomocí unikátních požadavků a ověřováním přes e-mailovou zprávu. Veškeré dotazy a změny nastavení Active directory jsou řešeny pomocí Powershell příkazů spouštěných z webové stránky. Byl vytvořen systém, který dovoluje uživatelům resetovat a měnit heslo, aniž by museli žádat pověřenou osobu. Přínosem této práce je zefektivnění a zautomatizování častých problému v Active Directory doméně.

Klíčová slova: Powershell, Active Directory, Windows Server, Self reset heslo

Abstract

The aim of the thesis is providing the possibility to reset and change password for users of Active Directory domain. There is the posibility to realize basic setting of domain and application for administrator of Active directory. The password reset is solved in the application by unique requirement and authorization with e-mail. The questions and changes of settings in the Active Directory are solved by the Powershell which can be runned from the website. There was created the system that allows to reset and change password for users without verification by authorized person. The benefit of this thesis is the streamlining and the automatization of common problems in the Active Directory domain.

Keywords: Powershell, Active Directory, Windows Server, Self reset password

Seznam použitých zkratek a symbolů

PS	–	PowerShell
AD	–	Active Directory
IIS	–	Internet Information Services
API	–	Application Programming Interface
OS	–	Operační systém

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Implementace Active Directory domény(dále jen AD) do firemního prostředí vyžaduje odborníka, který této problematice rozumí. Kromě nasazení AD je tedy potřeba zaměstnat dalšího člověka, který bude AD spravovat a řešit nastálé problémy a uživatelské požadavky. A právě uživatelské požadavky mohou správce AD brzdit v práci nebo naopak brzdit uživatele domény, protože je správce odkládá jako nedůležité.

Nejčastějšími požadavky ze strany uživatelů jsou ty na vyresetování hesla. Cílem bakalářské práce je automatizovat tento proces. Tím by počet uživatelských požadavků na správce AD výrazně klesl a reset hesla by tedy byl zcela v kompetenci uživatelů.

Automatizování takového procesu je hlavním tématem bakalářské práce. Běžní uživatelé domény nemají potřebná oprávnění k takovému úkonu, a proto bylo zapotřebí vymyslet postup, který by tuto funkcionalitu uživatelům zpřístupňoval a zároveň nebyl bezpečnostním rizikem pro doménu.

Textová část bakalářská práce je rozdělena do několika kapitol. V následující kapitole jsou pro čtenáře stručně popsány technologie použité při vývoji aplikace. Další část, rozdělena do několika kapitol, obsahuje popis jednotlivých problémů, s kterými se bylo nutné při vývoji aplikace vypořádat, popis jejich řešení a odůvodnění použitého řešení.

Poslední kapitola je zaměřena na budoucí rozšíření funkcionality a úpravy stávajicích funkcí aplikace.

V závěru pak bude zhodnoceno zda byly naplněny cíle bakalářské práce.

2 Použité technologie

Tato kapitola slouží k seznámení s technologiemi použitých v aplikaci.

2.1 Windows Server

Windows Server je označení pro operační systémy od firmy Microsoft orientované pro použití na serverech. Takový operační systém poskytuje různé služby, v terminologii zmíněných operačních systému též role a funkce.

2.2 ASP.Net

ASP.Net je webový aplikační framework na serverové straně, který programátorovi poskytuje možnost vytvářet dynamické webové stránky, aplikace nebo služby.

Pro vývoj aplikace byl tento programovací jazyk vybrán především proto, že každý ASPX soubor, v kterém programátor píše ASP kód, disponuje také souborem ASPX.CS, tzv. codebehind, který programátorovi umožňuje psát plnohodnotný C# kód. Rozvržení aplikace na jednotlivé logické vrstvy je díky tomu velmi snadné. Při správném rozvržení použitch vrstev je tedy možné docílit velmi lehké ASP.Net stránky, která je čitelná a do jejích kompetencí spadá pouze volání nižších vrstev, ošetřování vstupu a zobrazování výsledků.

2.3 C#

Objektově orientovaný jazyk C# je založen na .Net Frameworku. Tento jazyk je pro svou syntaxi, podobnou C++ nebo jazyku JAVA, velmi rozšířeným ve všech zařízeních podporující .Net Framework.

V aplikaci je C# využit ve všech vrstvách.

2.4 PowerShell

Powershell, dále jen PS, je skriptovací prostředí založené na .Net Framework. Na rozdíl od ostatních skriptovacích prostředí, jakým je například Bash v GNU/Linuxu nebo základní prostředí cmd.exe ve Windows, Powershell pracuje s objekty. Veškerou funkcionalitu, kterou obsahuje .Net Framework, Powershell dokáže poskytnout.

V aplikaci je Powershell využíván především pro existenci ActiveDirectory modulu, který poskytuje sadu příkazů pro kompletní správu Active Directory.

2.5 Active Directory

Active Directory, dále jen AD, je specializovaná databáze pro bezpečné, strukturované, hierarchické uložení dat o uživatelích, počítačích, tiskárnách a službách v doméně.

Aplikace s AD přímo pracuje. Konkrétně se dotazuje na uživatelské údaje, nastavuje a resetuje hesla, přiřazuje uživatele do skupin.

2.6 SQL Server

Pro ukládání dat, s kterými pracuje aplikace byl zvolen SQL Server 2014 v edici Express. Tato edice SQL serveru obsahuje veškeré potřebné nástroje, které při práci s daty v této aplikaci můžeme požadovat:

- Uložené procedury a funkce
- Triggery
- Transakce a zotavení

Licence SQL Server Express 2014 dovoluje také použití ve firemním prostředí, což je pro aplikaci zásadní.

2.7 CAPTCHA test

CAPTCHA test je forma testu vyskytujícího se na internetu, který rozlišuje reálné uživatele od robotů. Po uživateli je požadováno vyplnění textového pole na základě obrázku obsahujícího zdeformovanou, otočenou nebo jinak modifikovanou posloupnost písmen a čísel.

Využití CAPTCHA testu na internetových stránkách a v síti internet obecně je dvojího charakteru:

- Je uživatel skutečný člověk či robot? - Rozhodnutí záleží na správném či špatném vyplnení CAPTCHA testu. Obtížnost, neboli čitelnost, může být dle implementace deformací a zbarvení textu různá, proto i skutečný uživatel, jakožto člověk, může být vyhodnocen jako robot. Využití na registračních formulářích (databáze takové stránky nebude zaplněna neaktivními a umělými uživateli, robot nemůže zahlcovat diskuzní fóra, atd. . .) nebo veřejných webových anketách (výsledky anket nebudou zkresleny autorem vlastnícího robota hlasujicího z více míst internetu).
- Uživatel si opět zkontroluje zadané údaje - Vzhledem k použitým deformacím, obarvením, otočením a překryvům textu není občas jednoduché test CAPTCHA správně vyplnit. Uživatele to navádí ke zpětné kontrole zadaných dat.

Použití CAPTCHA testu v aplikaci je důležitým prvkem v modulu samoresetovacího hesla (viz kapitola 8.1.2 na straně 19)

V aplikaci je použita reCAPTCHA od firmy Google. V dalších verzích aplikace se počítá s vlastní implementací (viz kapitola 9.5 na straně 24)

3 Konfigurace serverů domény

Tato kapitola popíše potřebné hardwareové vybavení a nastavení software pro korektní funkčnost aplikace.

Podrobnější informace o potřebné konfiguraci lze nalézt v příloze A na straně 29

3.1 HW konfigurace

V rámci testovacího prostředí bylo použité jednotné prostředí. Na doménový řadič a IIS server byly nainstalovány stejné verze systému Windows Server. První testovací prostředí používá Windows Server 2008 R2, druhé Windows Server 2012 R2. Popsané požadavky na hardware jsou tedy vztaženy vůči použitým operačním systémům.

Tabulka 1: HW požadavky operačních systémů[2]

Komponenta	Požadavky	
	Minimální	Doporučené
Procesor	1.0 GHz (x86), 1.4 GHz (x64)*	2 GHz a více
Operační paměť	512 MB	2 GB a více
Volné místo na disku	10 GB** pro 2008 R2	40 GB a více
	32 GB** pro 2012 R2	
Optická mechanika	DVD-ROM mechanika	
Periférie	Monitor s rozlišením 800x600*** nebo vyšší Klávesnice Myš	
Síťová karta	Ano	

* Od verze Windows Server 2008 R2 jsou tyto operační systémy poskytovány pouze v 64 bitové verzi.

** Tento údaj je absolutní minimum, které by mělo být použito pro systémový oddíl.

*** Minimální požadavek na rozlišení od verze Windows Server 2012 je 1024x768. Pokud toto nebo vyšší rozlišení nebude dostupné, pak nebude aktivní prostředí Modern UI.

[2]Uvedené údaje jsou čerpány ze stránek Microsoftu - `http://technet.microsoft.com/en-us/library/dn303418.aspx` a `http://technet.microsoft.com/en-us/windowsserver/bb414778.aspx`

3.2 Podporované verze použitých technologií

Tabulka 2: Podporované verze použitých technologií

Verze Windows Server	.Net Framework[1][2]	RSAT(PS AD modul)[3]	ADWS[4]
Windows Server 2003 (R2)	3.5	ne	ne
Windows Server 2008	4.5*	ano	ano
Windows Server 2008 R2	4.5**		
Windows Server 2012	dodávano s verzí 4.5		
Windows Server 2012 R2	dodávano s verzí 4.5.1		

* V minimalistické instalaci Server Core není instalace .Net Frameworku možná.
** Od SP1 podporován v minimalistické instalaci Server Core kromě edice "for Itanium-Based Systems"

Z tabulky 3.2 je patrné, že lze aplikaci používat pouze na operačních systémech Windows Server 2008 a vyšší. Pokud by měla být zajištěna funkcionalita pro Windows Server 2003 (R2), bylo by potřeba vytvořit speciální verzi této aplikace, která používá odlišné technologie pro vzdálenou správu AD. Vzhledem ke stáří této verze Windows Server a nefinální verzi aplikace pro novější verze Windows Server, se neoplatí vytvářet speciální verzi pro Windows Server 2003 (R2).

3.3 Nastavení doménového řadiče

Jediný požadavek na nastavení doménového řadiče, který klade aplikace, je nainstalovaná a běžící služba Active Directory Web Services. Služba je dostupná od verze Windows Server 2008 R2 a instalována je automaticky při povýšení serveru na doménový řadič.

Službu je dále potřeba zapnout a nastavit automatické spuštění. Toho lze dosáhnout například pomocí PS příkazu:
`Set-Service -Name ADWS -Status Running -StartupType Automatic.`

3.4 Nastavení IIS serveru

Aby bylo možné využívat možností ActiveDirectory modulu pro PS, je nutné nainstalovat funkci Remote Server Administration Tools. Instalaci funkce lze provést pomocí PS příkazu:
`Add-WindowsFeature RSAT-AD-PowerShell,RSAT-AD-AdminCenter.`

Poznámka 3.1 V případě jednoserverové instalace - doménový řadič a IIS na jednom stroji - není potřeba instalovat Remote Server Administration Tools

[1] `http://msdn.microsoft.com/en-us/library/8z6watww(v=vs.110).aspx`
[2] `http://msdn.microsoft.com/en-us/library/bb882520(v=vs.90).aspx`
[3] `http://technet.microsoft.com/en-us/library/cc731209.aspx`
[4] `http://technet.microsoft.com/en-us/library/dd391908(v=ws.10).aspx`

3.5 Nastavení IIS poolu

Použité šifrování hesel v aplikaci vyžaduje povolení načítání uživatelského profilu pro použitý IIS pool. Nastavení je možné provést pomocí příkazu:
appcmd.exe set config -section:applicationPools
“[name=’NÁZEV_POUŽITÉHO_IIS_POOLU’].processModel.loadUserProfile:true”, který lze nalézt v C:\Windows\System32\inetsrv.

3.6 Příprava SQL serveru

Před prvním spuštěním aplikace je potřeba nakonfigurovat databázi, tabulky a přihlašovací údaje. Pro jednoduchou konfiguraci je dostupný PS skript Create_db_and_userlogin.ps1 ve složce Install\DatabasePreparation. Ten na základě uživatelských vstupů vytvoří na SQL serveru databázi, tabulky, přihlašovací údaje do databáze a uložené procedury a triggery.

3.7 Oprávnění souboru web.config

Při prvním spuštění aplikace je uživatel automaticky přesměrován na instalační stránku. Aby bylo možné pomocí instalační stránky korektně nastavit aplikaci, je nutné nastavit použitému IIS poolu oprávnění pro zápis do souboru web.config. To lze provést například pomocí příkazu
icacls web.config /grant “IIS AppPool\DefaultAppPool”:(M)

4 Persistentní uložení konfiguračních dat

Aplikace potřebuje v době běhu pracovat s daty jako jsou konfigurační údaje a uživatelské požadavky na reset hesla. Samotné konfigurační údaje by bylo možné mít napevno ve zdrojových kódech, ale tím by se omezila funkčnost aplikace jen na konkrétní doménu, případně by editace pro každou instalaci této aplikace byla velmi složitá. Proto je vhodné tato data ukládat do úložiště, kde je s nimi možno později libovolně pracovat. Úložištěm pro tento účel byl zvolen SQL Server 2014 v edici Express. Důvody pro zvolení SQL serveru namísto ukládání dat do souboru byly následující:

1. Přesunutí datové režie z aplikace na databázi
2. Větší možnosti zpracování dat mimo aplikaci
3. Jednodušší možnosti rozšíření stávající databáze

Aplikační vrstva je díky tomuto odlehčena od režie správy dat a jejich zpracování. Jediný prvek, který spravuje aplikační vrstva napůl s databázovým serverem, je životnost požadavku na reset hesla. Použitý SQL Server bohužel neobsahuje funkci SQL Server Agent umožňující spouštět požadované úlohy po určitém čase nebo na základě konkrétní události. Tuto funkci je proto potřeba nasimulovat vhodným voláním uložených procedur, což bohužel nelze zcela zajistit na straně SQL serveru vzhledem k návrhu a provedení modulu pro self reset password.

5 Vykonání PS skriptu v kontextu jiného uživatele

Aplikace je založena na poskytování funkčnosti, které mohou ke svému korektnímu průběhu vyžadovat konkrétní uživatelská oprávnění. Vzhledem k charakteru aplikace jako webové stránky je vykonávajícím uživatelem IIS pool, který nemá potřebná oprávnění pro vykonání požadovaných PS příkazů. Potřebujeme tedy docílit toho, aby byla aplikace spouštěna v kontextu uživatele s odpovídajícími oprávněními.

Při řešení tohoto problému je možné se vydat třemi směry.

5.1 Zosobnění IIS poolu

V této kapitole popíšu možnosti zosobnění kontextu použitého IIS poolu.

5.1.1 Zosobnění autorizovaného uživatele

Jednoduchým zápisem do souboru web.config povolíme zosobnění autorizovaných uživatelů.

```
<identity impersonate="true" />
```

Výpis 1: Nastavení zosobnění pro autorizovaného uživatele

Pokud je uživatel prohlížející si stránku zároveň platným uživatelem v cílové doméně, pak se stránky spustí v kontextu tohoto uživatele. V opačném případě je použit IIS pool pro anonymní uživatele IUSR_název_počítače.

Toto řešení je ovšem pro účely aplikace nevhodné, protože běžný uživatel, který aplikaci bude používat, typicky nemá potřebná oprávnění pro PS skripty, které mění vlastnosti a nastavení.

5.1.2 Zosobnění konkrétního uživatele

Pouhým přidáním uživatelského jména a hesla k předešlému záznamu v souboru web.config docílíme požadované funkcionality. PS skripty budou spuštěny a provedeny v kontextu zadaného uživatele, i když s aplikací bude pracovat nepřihlášený uživatel (self reset hesla) nebo uživatel přihlášený (změna hesla).

```
<identity impersonate="true"
          userName="domain\user"
          password="password" />
```

Výpis 2: Nastavení zosobnění pro konkrétního uživatele

Nicméně i toto řešení je nevhodné pro použití:

Možnost zadat pouze jednoho uživatele - pokud bychom uživatelům chtěli poskytnout možnost resetovat heslo a zálohovat složku s dokumenty, pak je třeba vybrat účet v jehož kompetenci je provedení těchto věcí. Byli bychom tedy omezení pouze na jeden účet a nemohli bychom si za běhu programu vybírat vhodný účet.

Jediný použitelný účet pro účely aplikace náleží skupině Administrátors - problém vycházející z předešlého. Čím větší množina funkcí bude skrz aplikaci poskytována, tím budou rozsáhlejší požadavky na oprávnění použitého účtu. Brzo tedy zjistíme, že účet skupiny Administrators je jediný, který lze použít. To by ale mohlo vést k bezpečnostnímu riziku. Heslo je uložené v souboru web.config v lidsky čitelné podobě a aplikace by v kontextu uživatele s administrátorskými běžela neustále.

5.2 Zosobnění voláním funkcí jádra Windows

Při použití zdrojového kódu, který je poskytován Microsoftem[3], jsme schopni proces aplikace za běhu interaktivně přepínat ze základního IIS poolu na různé uživatelské účty.

Tento typ zosobnění bohužel trpí přetrvávající chybou[4], kdy se za neurčitých okolností při použití dvojice ASP.Net a PS nepředá informace o zosobnění ze zdrojového vlákna do cílového.Jelikož tento problém nelze nijak predikovat ani programátorsky ošetřit, je nevhodný pro použití v aplikaci.

Poznámka 5.1 Přestože na uvedenou chybu dle informací existuje řešení, na testovací konfiguraci zůstávalo toto řešení nadále nefunkční.

5.3 PowerShell Credentials

Poslední a v aplikaci použité řešení. Příkazu, který je potřeba spustit v kontextu jiného uživatele, přidáme parametr -Credential a jako hodnotu předáme instanci třídy System.Management.Automation.PSCredential.

U příkazů, kterým chybí v základu možnost spuštění pomocí parametru Credential, lze tohoto chování docílit pomocí příkazu Start-Process nebo Invoke-Command, které parametr Credential mají a dovolují spouštět procesy nebo vykonávat PS příkazy.

[3] http://msdn.microsoft.com/en-us/library/w070t6ka(v=vs.110).aspx

[4] http://blogs.msdn.com/b/powershell/archive/2007/09/10/impersonation-and-hosting-powershell.aspx

6 Spouštění PS skriptu v C#

Pro vykonání požadovaných PS skriptů je v aplikaci dostupná třída PSHandler založená na jmenných prostorech .Net frameworku System.Management.Automation a System.Management.Automation.Runspaces. Tato třída poskytuje programátorovi jednoduché rozhraní pro vytvoření, nastavení a vykonání PS příkazu. Není tedy nutné, aby programátor znal proces vytváření C# prostředí pro PS. Jednoduchou ukázku použití třídy PSHandler je možné vidět níže.

```
using (PSHandler ps = new PSHandler(PSCredentials.AccountOperator))
        {
            ps.CreateCommand("Set-ADAccountPassword");
            ps.Command.Parameters.Add("Identity", userName);
            ps.Command.Parameters.Add("Reset", null);
            ps.Command.Parameters.Add("NewPassword", SecureStringConverter.
                ToSecureString(password));

            ps.PrepareCommand();
            ps.RunScript();
        }
```

Výpis 3: Ukázkové použití třídy PSHandler

Třída implementuje rozhraní IDisposable, což po skončení bloku using zajistí automatické uvolnění zdrojů. Implementace tohoto rozhraní a použití bloku using má ovšem výhodu v čitelnosti kódu, protože je hned zjevné, kde vytváření PS příkazu začíná a kde končí jeho vykonávání.

Při vytváření instance třídy PSHandler se konstruktoru předává konkrétní hodnota výčtového typu PSCredentials, kterým programátor definuje požadovaný uživatelský kontext, v kterém se PS příkaz bude spouštět.

Dalším krokem je vytvoření PS příkazu a přirazení požadovaných parametrů. Parametry, kterým se předává hodnota null, jsou přepínače - jejich explicitní použití je už tedy samo o sobě vyjádřením požadované hodnoty.

Metoda PrepareCommand() připraví PS příkaz pro spuštění - interně přidá požadovaná PS oprávnění a přidá jej do předpřipraveného PS prostředí. Vykonání příkazu je zajištěno voláním metody RunScript().

Jak lze vidět na výše uvedeném případě, je spouštění PS příkazů podstatně složitější než v samotném Powershellu. Na druhou stranu je tento zápis intuitivnější a čitelnější ve srovnání s řešením napsaným pomocí C#[5]

[5] http://stackoverflow.com/a/21252845

```
public void ChangeMyPassword(string domainName, string userName, string currentPassword,
     string newPassword)
{
    string ldapPath = "LDAP://192.168.1.xx";
    DirectoryEntry directionEntry = new DirectoryEntry(ldapPath, domainName + "\\" + userName,
        currentPassword);
    if (directionEntry != null)

    {
        DirectorySearcher search = new DirectorySearcher(directionEntry);
        search.Filter = "(SAMAccountName=" + userName + ")";
        SearchResult result = search.FindOne();
        if (result != null)
        {
            DirectoryEntry userEntry = result.GetDirectoryEntry();
            if (userEntry != null)
            {
                userEntry.Invoke("ChangePassword", new object[] { currentPassword,
                    newPassword });
                userEntry.CommitChanges();
            }
        }
    }
}
```

Výpis 4: Reset hesla pomocí C#

7 Bezpečnost aplikace

Tato kapitola pojednává o prvcích napomáhající většímu zabezpečení aplikace.

7.1 HTTPS

HTTPS je šifrované spojení mezi serverem a klientem. Potřeba takového spojení je v aplikaci prozatím malá, protože aplikace běží v intranetu domény, nevyužívá uživatelských cookies a do serverové session se ukládá pouze šifrované uživatelské jméno přihlášeného uživatele. Možnosti, co odposlechnout, jsou tedy poměrně malé.

Do budoucna bude ovšem aktivní běh HTTPS vyžadován, naopak nebude poskytováno HTTP spojení. Důvodem je plán poskytnout uživatelům více funkcionality(kapitola 9.3) a vestavěné PS prostředí (kapitola 9.1) a zpřístupnit aplikaci i ze sítě internet.

Aby toho bylo možné dosáhnout, bude nutné heslo přihlášeného uživatele v nějaké velmi bezpečné formě ukládat. Například heslo uložené v session, které se šifruje náhodně vygenerovaným klíčem uloženým v cookies, jež je unikátní pro každé přihlášení. Odposlechnout obě tyto věci se šifrovaným HTTPS spojením bude pro případného útočníka velmi obtižné.

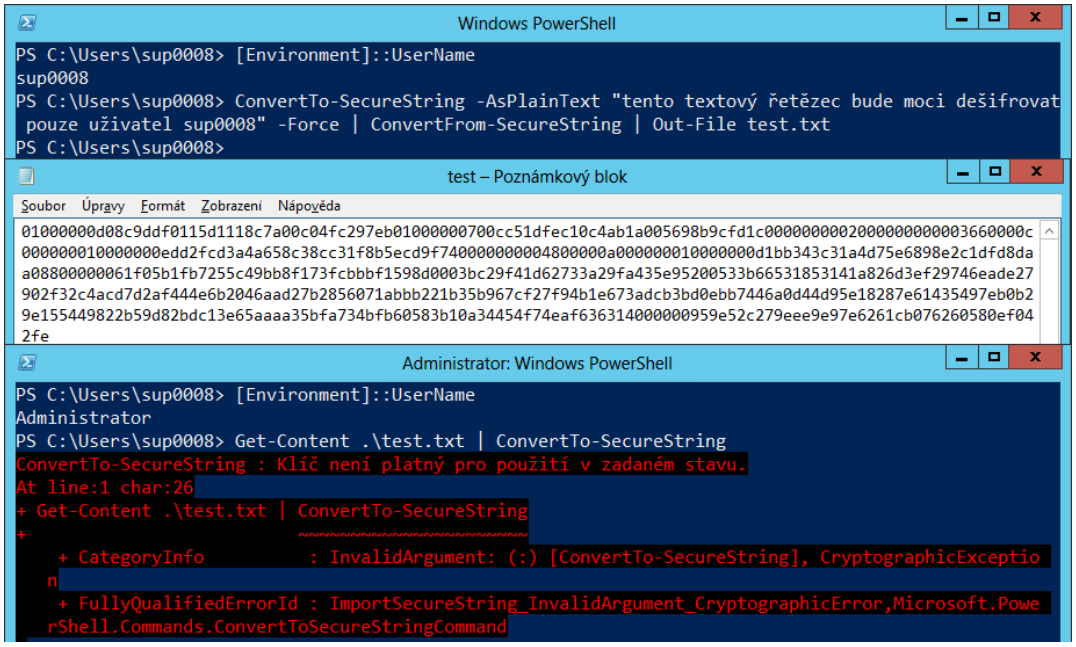
7.2 Bezpečnost hesla

Aplikace se v době běhu potřebuje přihlašovat na SMTP server a spouštět PS skripty v kontextu jiného uživatele. Jelikož je to právě aplikace, která se někam přihlašuje, je potřeba hesla ukládat a číst. Vzhledem k použitému účtu s opravněním skupiny Account Operator, který má v AD doméně poměrně vysoká oprávnění, je potřeba zajistit, aby hesla byla uložena na zabezpečeném úložišti a aby byla zašifrována.

První bod splňuje uložení v databázovém SQL serveru, kde je, v případě potřeby, možné dále nastavovat restrikce přístupu a oprávnění pro práci s daty jednotlivým uživatelům.

Druhý bod zabezpečení hesla, tedy šifrování, je v aplikaci vyřešen pomocí třídy SecureString. Veškerý text, který je do instance takové třídy vložen, je automaticky zašifrován. Dalším, a pro účely aplikace nejvýznamějším, bezpečnostním prvkem třídy SecureString je způsob jakým se vstupní textový řetězec šifruje. Při šifrování se využívá bezpečnostního klíče, který je unikátní pro každého uživatele operačních systému Windows a Windows Server. Takovou šifru je potom možné odšifrovat jen na uživatelské účtu, kde bylo provedeno prvotní šifrování. To v praxi znamená jediné - šifrá je nepřenositelná a její znalost je nepovolané osobě bezcenná, jak lze vidět na obrázku 1.

Jediná možnost získání hesla v čitelné podobě, kterou není možné ovlivnit na úrovni aplikace, je přístupem ke zdrojovým kódům běžící aplikace. Restriktivním nastavením přístupových práv do kořenové složky aplikace lze jednoduše eliminovat i tuto možnou bezpečnostní díru.



Obrázek 1: Demonstrace nepřenositelnosti šifry

8 Rozhodnutí při vývoji

V této kapitole budou popsány rozhodnutí učiněná při vývoji aplikace, které měly, případně budou do budoucna mít, největší dopad na funkčnost a použitelnost aplikace jak z pohledu uživatele pracujícího s aplikací, tak programátora, který aplikaci bude chtít rozšířit, změnit.

8.1 Self Reset Password

Vyresetování hesla je nejčastějším uživatelským požadavkem na správce AD nebo zaměstnance podpory. Proto je vhodné tento úkon zautomatizovat, aby takovými žádostmi nebyli zaměstnanci podpory zahlceni. Výhody jsou zřejmé - podpora nemusí řešit tyto triviální záležitosti a uživatel nemusí čekat na vyřízení požadavku.

Problém nastává již ve chvíli, kdy zamýšlíme poskytnout tuto funkcionalitu uživatelům. Jak ověřit, že je požadavek opravdu od uživatele, pro kterého je žádán reset hesla? Nejčastěji používané způsoby verifikace autentičnosti požadavku u těchto jsou popsány v následujících podkapitolách.

8.1.1 Bezpečnostní otázka

Jednoduchý systém, kdy uživateli položíme otázku, kterou si nejčastěji pri registraci vybral a na kterou zná tedy odpověď teoreticky pouze on. Bohužel tento způsob verifikace autenticity požadavku má několik praktických a bezpečnostních chyb:

1. Praktické chyby

   - Typicky se používá u služeb, kde uživatel prochází sám registračním procesem - Ve firemním prostředí je uživateli účet vytvořen. Při nasazení aplikace do fungující AD domény bychom museli získávat neurčitou množinu otázek a odpovědi, což by mohlo velmi znesnadnit proces nasazení.
   - Po delší době může uživatel svou odpověď zapomenout - V takovém případě by aplikační modul pro resetování hesla byl nepoužitelný a v konečném důsledku by uživatel opět musel kontaktovat technickou podporu.
   - Bezpečnostní otázky a odpovědi je potřeba ukládat - Musíme vymyslet kde se budou otázky a odpovědi ukládat (popisek uživatele v AD, soubor, databáze). Pokud budou otázky a odpovědi uložené jinde než v AD, jak bude řešeno provázaní jednotlivých otázek s uživatelem, atd. . .

2. Bezpečnostní

   - Množina otázek je většinou omezená - S narůstajícím počtem uživatelů v doméně narůstá také pravděpodobnost výskytu stejné otázky a odpovědi.

- Uživatel má většinou volbu zapsat si vlastní bezpečnostní otázku a odpověď - Uživatelé si nechtějí pamatovat nic těžkého, a proto by se bylo možné postupem času setkat s otázkou typu „Jméno mého psa?“ a odpovědí „Alík“, což je odhadnutelná odpověď a mohlo by dojít k neoprávněné změně hesla.

8.1.2 E-mailová zpráva

Toto ověření autenticity požadavku je použito v aplikaci, a proto bude podrobněji popsáno, společně s průběhem uživatelské interakce a aplikačními postupy.

Ověřování přes e-mailovou zprávu vytváří na uživatele jediný požadavek - znát přístupové údaje k e-mailové schránce. Vzhledem k faktu, že ve firemním prostředí je e-mailová komunikace prakticky standardem a používá se každý den, je pro splnění tohoto požadavku větší předpoklad než u bezpečnostní otázky.

Na AD doménu je ze strany aplikace kladen jediný požadavek - mít u uživatelů zadané jejich e-mailové adresy.

Průběh tohoto procesu je následující:

1. Uživatel vyplní své heslo a CAPTCHA test
    1.1. Kontrola CAPTCHA testu a existence zadaného uživatelského jména v AD
    1.2. Aplikace vytvoří unikátní sekvenci čísel a písmen, uloží ji společně s datem vytvoření a svázaným uživatelským jménem do databáze
    1.3. Aplikace odešle e-mailovou zprávu s unikátním odkazem pro daný požadavek na e-mailovou adresu uvedenou v AD
2. Uživatel otevře odkaz v e-mailové zprávě
    2.1. Aplikace porovná unikátní odkaz vůči záznamu v databázi a zkontroluje jestli nevypršel jeho časový limit
    2.2. Vygeneruje se náhodné 8 místné heslo skládající se z malých/velkých písmen a číslic
    2.3. Heslo je vyresetováno
    2.4. Požadavek je v databázi označen jako neaktivní
3. Heslo je uživateli zobrazeno na obrazovce
    3.1. Aplikace heslo nikam neukládá

Poznámka 8.1 Hlavní body seznamu popisují uživatelskou interakci, vnořené body naopak akce aplikace probíhající na pozadí

Z předešlého výčtu postupu celým procesem resetování hesla je zřejmé, že verifikace autenticity požadavku probíhá až po zapsání samotného požadavku do databáze, odeslání e-mailové zprávy a otevření unikátního odkazu. Z toho vyplývá, že každý, kdo zná

uživatelské jméno a vyplní správně CAPTCHA test, je schopen vygenerovat požadavek pro reset hesla. To je samozřejmě nežádoucí, jak z pohledu cílového uživatele (byla by mu zahlcena e-mailová schránka), tak z pohledu aplikace (generování a zápis požadavku, mnoho otevřených požadavků v jednu chvíli). Proto je vhodné implementovat různé způsoby, jak tomuto zabránit. V aplikaci je ochrana proti zahlcení databáze řešena jednoduše - dokud je uživatelský požadavek aktivní, nový požadavek není vygenerován. Ochrana proti zahlcení uživatelový e-mailové je programově složitější a fungovala by na bázi dočasného zamezení tvorby žádostí, pokud by počet vygenerovaných požadavků přesáhl určitou jednotku za čas. Tato funkcionalita není v aktuální verzi naimplementována.

8.2 Rozdělení působnosti C# a Powershellu

V raných fázích vývoje nebyly pevně stanovené hranice mezi použitím jazyka C# a PS skriptů. Zpočátku to nečinilo žádné potíže, ale s přibývající funkcionalitou, požadavky na přehlednost a konzistenci bylo třeba učinit rozhodnutí, který jazyk bude mít v kompetenci určitou doménu problému.

Důležitými faktory pro toto rozhodnutí bylo, jakým prostředkem je možné dosáhnout určité funkcionality a jednoduchost nasazení a použití. Aby bylo možné učinit relevantní rozhodnutí, je nejprve nutné zrekapitulovat možnosti C# a Powershellu.

- C#
    1. Nabízí velké množství různorodých tříd
    2. Codebehind ASPX stránek je C#
    3. Dostupné na všech platformách podporující .Net framework
    4. Zosobnění řešeno pomocí volání funkcí jádra Windows nebo zosobnění konkrétního IIS poolu
- Powershell
    1. ActiveDirectory modul
    2. Rozsáhlé možnosti pro vlastní skriptování
    3. Možnost využít .Net tříd
    4. Zosobnění řešeno předáváním Credential objektu do spuštěného skriptu

8.2.1 Powershell

Dostupnost ActiveDirectory modulu a použitý typ zosobnění byl rozhodující pro použití Powershellu jako vykonávající vrstvy pro práci s AD. Svůj význam ovšem hraje také použité názvosloví PS skriptů, kdy je okamžitě jasné, co můžeme očekávat za výsledek. Tímto se velmi ulehčuje proces přidávání další funkcionality bez nároku na hlubší znalost C#. Díky tomu lze docílit podobně jednoduchého procesu - vytvoření nové ASPX stránky, v codebehind zavolání C# třídy zajišťující práci s PS, vložení PS skriptu a zpracování

výsledků. Při použití C#, jako režijní vrstvy nad AD, by tohoto zjednodušeného procesu nebylo jednoduché dosáhnout. Programátor by musel napsat kód, který by nebyl na první pohled všeříkající a především by se musel rozhodnout mezi třídami .Net frameworku poskytující možnosti práce s AD(PrincipalContext, UserPrincipalContext, DirectorySearcher a DirectoryEntry).

8.2.2 C# - zprostředkující vrstva

Ač je C# nevhodný pro použití na práci s AD, na propojení vstupně výstupních ASPX stránek s PS skripty spouštěných na pozadí je to logická volba. Pomocí C# je možné kontrolovat vstupy, odchytávat výjimky, generovat potřebná hesla a požadavky, vytvářet logy a připojovat se na použitou databázi. Především ale C# umožňuje na základě uživatelských požadavků pracovat s PS, což dvojice ASP.Net a PS není schopná poskytnout.

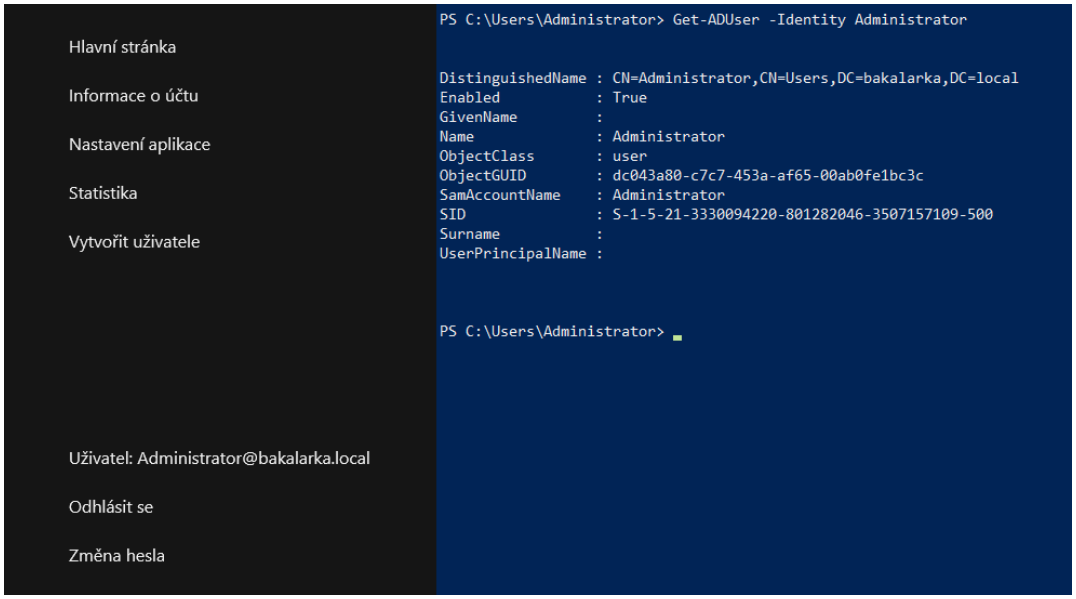
Dále je C# vhodný pro tvorbů PS modulů, čehož bylo využito při vývoji vlastního kompilovaného PS modulu použitého v aplikaci.

9 Výhled do budoucna

9.1 Vestavěné Powershell prostředí

Za určitých okolností by autorizovaný uživatel mohl potřebovat provést PS příkaz, jež není dostupný, a zároveň by mohlo být zdlouhavým procesem připojení přes VPN, případně na vzdálenou plochu.

Pro tento případ by byl do budoucna připraven modul, který by simuloval prostředí Powershell ve Windows. Dovoloval by tedy spouštět všechny dostupné příkazy a vykonával by je dle oprávnění přihlášeného uživatele.



Obrázek 2: Ukázka vestavěného PS prostředí

9.2 Chytřejší třída PSHandler

Momentálně třída PSHandler poskytuje možnost spouštět příkazy, které v době běhu žádným způsobem neiteragují s uživatelem, případně aplikační vrstvou. Použití je tedy omezené pouze na spuštění a zpracování výsledků. To je potřeba do budoucna změnit, zvlášť pokud je zamýšleno dodat funkcionalitu vestavěného PS prostředí.

Další funkcionalita, která je důležitá pro programátora a v důsledku také pro uživatele, je odchytávání výjimek. Programátor by měl možnost výjimky zpracovat a zaznamenat do logu. Uživatel by při vyvolání výjimky a jejím zpracování mohl být obeznámen, že požadovaná akce neproběhla, co může dále dělat, atd. Momentálně tohoto chování třída PSHandler není schopna, což znamená jediné - pokud ve spouštěném PS skriptu nastane

výjimka ani programátor, ani uživatel o tom neví. Zpracování PS výjimek v C# je tedy velmi důležitá funkčnost, kterou bude potřeba implementovat.

9.3 Poskytnout více funkcí všem oprávněným uživatelům

Aktuální množina poskytovaných funkcí je malá a nevyužívá plně ideu aplikace - poskytovat určitý výběr služeb a nastavení mimo rámec jejich oprávnění a zpřístupnit častá nebo těžko dohledatelná nastavení.

Návrh nové funkcionality běžným uživatelům:

- Nahlášení problému přes webový formulář - Uživatel by měl mít možnost informovat IT oddělení o problémech se svým počítačem a účtem v doméně.
- Stahování\nahrávání souborů - Při práci z domova, respektive mimo místo zaměstnání, může být vhodné zpřístupnit možnost práce se soubory z uživatelské a sdílené složky přes webové rozhraní.

Administrátorům a uživatelům s vyšším oprávněním poskytnout:

- Nástroje pro správů různých rolí a funkcí Windows Server - Tato možnost by si nekladla za cíl nahradit grafickou konzoli v systémech Windows Server ani dostupné možnosti správy přes Powershell. Dostupné by byly pouze příkazy rolí a funkcí, jež se často používají.
- Fronta přijatých problémů v doméně od uživatelů - Podle kategorie problému by byly zobrazeny přijaté problémy kompetentním uživatelům.

9.4 Poskytnutí API

Aktuální způsob přidávání nové funkcionality má jedno zásadní omezení, které může být na obtíž z pohledu vývojáře základní aplikace a z pohledu vývojáře modulů rozšiřujících funkcionalitu aplikace, a tím je složitost přidávání nové funkionality.

Momentálně je nutné napsat si vlastní ASPX stránku, což zahrnuje nadefinovat použitou master page, znát CSS styl pro nastylování tlačítek, textových polí, atd. . . , v code behind volat třídy a metody aplikačních knihoven.

Při vývoji modulu by neměl být kladen nárok na znalost struktury aplikace.

- Pohled vývojáře základní aplikace
  1. Podrobná znalost aplikace je potřeba pouze při úpravách stávajícího jádra aplikace nebo implementaci nových funkcí, které bude aplikace po instalaci vždy obsahovat.
  2. Při nesprávně vytvořeném modulu se může stát, že nastanou v modulu neočekávané chyby, které vývojář aplikace nemá možnost ovlivnit.
- Pohled vývojáře modulu

1. Vývoj modulu by měl mít jednodušší průběh a neměl by zahrnovat řešení problému, které už jsou v aplikaci vyřešené

Jestliže bychom tedy měli vyhovět požadavkům obou zainteresovaných stran, bude potřeba implementovat určitou formu API.

9.5 Vlastní implementace CAPTCHA testu

Pokud jsme jako hlavní důvod pro poskytnutí Self Reset Password uživatelům uvedli zautomatizování požadavků na reset hesla, pak musíme zajistit funkčnost a dostupnost souvisejících částí. reCAPTCHA od firmy Google je jedinou knihovnou třetí strany, což v případě nedostupnosti této služby způsobí nefunkčnost dialogu pro vytvoření požadavku na reset hesla, díky čemuž bude celý modul Self Reset Password z pohledu uživatele nefunkční a požadavek bude opět směrovat na zaměstnance podpory.

Obrázek 3: Ukázka reCAPTCHA testu[6]

Ač se může zdát, že nedostupnost reCAPTCHA je nepravděpodobná vzhledem k zázemí jejího tvůrce, může i přesto nastat situace, kdy aktuálně použité řešení nebude pracovat korektně:

- reCAPTCHA vyžaduje povolení odchozího spojení na portu 80 - Pro aplikaci s reCAPTCHA běžící na jiném portu je potřeba vytvořit potřebné pravidlo ve firewallu.

Důvod pro nepoužití reCAPTCHA může být i jiného než funkcionálního charakteru. Pro ASP.NET je dostupná pouze kompilovaná knihovna, což znemožňuje úplnou kontrolu nad aplikací. Nedostupnost zdrojových kódu k této knihovně vede ve výsledku k nekonzistenci s rozhodnutím vydat finlní verzi aplikace s otevřenými zdrojovými kódy.

Výše zmíněné je důvodem pro budoucí lokální implementaci vlastní CAPTCHA knihovny, případně použití knihovny třetí strany s otevřenými zdrojovými kódy. Po použití lokálního modulu bychom zcela eliminovali možnou nepoužitelnost Self Reset Password.

9.6 Větší množství prezentačních vrstev

Momentální výstup ve formě internetové stránky je v dnešní době zobrazitelný ve všech typech zařízení obsahující internetový prohlížeč, nicméně ne ve všech zařízeních bude

[6] `https://developers.google.com/recaptcha/images/reCAPTCHA_Sample_Red.png`

používání aplikace příjemným uživatelským zážitkem. Důvodem je především nízké rozlišení (v době psaní této práce je známým problémem také velké rozlišení na malé uhlopříčce displeje) chytrých telefonů a tabletů, pro které aktuální vzhled nebyl navržen a ani optimalizován.

9.6.1 Responzivní design

Východiskem by bylo přestylování aktuálního designu do responzivní formy. Tedy takové, kdy se design přizpůsobuje šířce prohlížené plochy.

9.6.2 Webová služba a nativní aplikace

Druhou a pracnější možností by bylo přepsat aplikaci do podoby webové služby. Webová služba je pouhé rozhraní, které na základě textových požadavků vykoná činnost a případně vrátí výsledek v textové podobě.

V takovém případě by bylo nutné přepsat stávající výstup aplikace, tak aby byl kompatibilní s webovou službou. Nicméně bychom získali možnost vytvořit nativní aplikace s designem respektujícím uživatelské rozhraní mobilních operačních systémů.

10 Závěr

Hlavní cíl bakalářské práce, tedy poskynutní nástroje pro reset hesla uživatelům Active Directory domény, byl úspěšně implementován a otestován na dvou na sobě nezávislých prostředích. U tohoto modulu aplikace bylo potřeba zajistit maximální spolehlivost. Tento nástroj je proto velmi komplexní. Obsáhl všechny představené technologie:

- ASP.Net - prezentační vrstva
- C# - logická vrstva starající se o generování požadavků, hesel, volání databázového serveru a Powershellu v požadovaný okamžik, atd. . .
- Powershell - resetování hesla v Active Directory
- SQL Server 2014 - práce s daty mimo aplikační logiku

Kromě toho byly také vyřešeny přednesené problémy. Aplikace umí bezpečně ukládat kritická data a hesla. Dokáže spouštět Powershell příkazy s oprávněními jiného uživatele.

Rozhodnutí učiněná při vývoji nebylo možné rozhodnout okamžitě. Bylo potřeba zvážit veškerá možná pozitiva a negativa daného rozcestí, ale ve výsledku bylo vždy rozhodnuto dle mého nejlepšího úsudku, s ohledem na čitelnost, relativní jednoduchost a bezpečnost výsledné aplikace (například Powershell pro nastavování a dotazování Active Directory namísto C#, kontrola autenticity požadavku na reset hesla).

Použité technologie a rozhodnutí učiněná při vývoji mi byly cenným zdrojem informací a napomohly většímu pochopení a proniknutí do dané problematiky.

Vývoj aplikace dále vedl k užitečným poznatkům o možném budoucím vývoji a směřování celé aplikace. Závěr bakalářské práce proto nespočívá jen v úspěšné implementaci navrhnutých cílů, ale také v rozhodnutí nadále s vývojem aplikace pokračovat a přednesené možnosti o budoucím vývoji dále implementovat.