Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Principy přenosu souborů a jejich automatizace pro zabezpečení důvěrnosti a integrity přenášených dat

Principles of file transfer and their automation to ensure confidentiality and integrity of transferred data

Bakalářská práce

Autor: Kryštof Matuška
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Tomáš Svoboda, Ph.D.

Hradec Králové duben 2024

Prohlášení:

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

V Hradci Králové dne 26.4.2024 Kryštof Matuška

Poděkování:

Tímto bych rád poděkoval vedoucímu bakalářské práce Ing. Tomáši Svobodovi, Ph.D. za metodické vedení práce, konzultace a veškerou další pomoc, kterou mi při vypracovávání práce poskytl. Dále bych chtěl poděkovat rodině a manželce za podporu při dokončování této práce.

Anotace

Tato práce se zabývá problematikou přenosu souborů a jejich automatizace s ohledem na zabezpečení důvěrnosti a integrity dat. Práce systematicky analyzuje historii internetu a koncept CIA triad, který zdůrazňuje důležitost důvěrnosti, integrity a dostupnosti dat. Dále zkoumá různé protokoly pro přenos souborů a emailů, spolu s opatřeními pro zajištění jejich bezpečnosti. Součástí práce je také přehled programovacích jazyků, emailových klientů a nástrojů vhodných pro automatizaci procesů. Cílem této práce je poskytnout komplexní pohled na možnosti automatizace a zabezpečení přenosu dat v prostředí IT.

Annotation

Title: Principles of file transfer and their automation to ensure confidentiality and integrity of transferred data

This work deals with the issue of file transfer and their automation with regard to ensuring the confidentiality and integrity of data. The thesis systematically analyzes the history of the Internet and the concept of CIA triads, which emphasizes the importance of confidentiality, integrity and availability of data. It also examines various file and email transfer protocols, along with measures to ensure their security. The work also includes an overview of programming languages, email clients and tools suitable for process automation. The aim of this work is to provide a comprehensive view of the possibilities of automation and data transfer security in the IT environment.

Obsah

1 Úvod

V dnešní době neustále rostoucí digitální společnosti je zabezpečení přenosu dat zásadním faktorem pro organizace i jednotlivce. S narůstajícím objemem a důležitostí přenášených informací se zvyšuje i riziko jejich zneužití či úniku. Proto je nezbytné nejen efektivně přenášet data, ale také zajistit jejich důvěrnost a integritu.

Tato práce se zaměřuje na problematiku přenosu souborů a jejich automatizace s důrazem na zabezpečení důvěrnosti a integrity přenášených dat. Cílem je analyzovat existující postupy, protokoly a nástroje a navrhnout efektivní řešení pro praktické využití.

Teoretická část práce bude představovat základní koncepty, které jsou nezbytné pro pochopení problematiky. To zahrnuje historii internetu a jeho vývoj, principy zabezpečení dle CIA triad (Confidentiality, Integrity, Availability) a přehled existujících protokolů pro přenos souborů a emailů a jejich zabezpečení. Dále se práce zabývá vhodnými programovacími jazyky, emailovými klienty a nástroji pro automatizaci, které jsou klíčové pro praktickou část.

Praktická část práce bude demonstrovat aplikaci teoretických poznatků v konkrétních scénářích. V práci budou scénaře zaměřeny zejména na využití Powershell skriptů a automatizačních nástrojů jako je Task Scheduler a Outlook. Tyto nástroje budou aplikovány v reálných situacích, aby bylo možné demonstrovat jejich efektivitu a praktickou hodnotu pro zabezpečení a automatizaci přenosu dat.

2 Cíl práce

Cílem této práce je analyzovat a zhodnotit současné postupy a nástroje pro přenos souborů s důrazem na zabezpečení důvěrnosti a integrity dat. Na základě této analýzy bude navrženo a implementováno efektivní řešení v podobě Powershell skriptů a automatizačních nástrojů, které budou demonstrují praktické využití pro zabezpečený a automatizovaný přenos dat. Cílem je poskytnout ucelený přehled o možnostech zabezpečení a automatizace přenosu dat a doporučit konkrétní postupy pro zlepšení současných procesů v organizacích i jednotlivcích.

3 Metodika zpracování

Metodika zpracování této práce je založena na podrobné analýze současných postupů a nástrojů pro přenos souborů s důrazem na zabezpečení důvěrnosti a integrity dat. Za tímto účelem bude provedena rozsáhlá literární rešerše, která poskytne ucelený přehled o existujících metodách, technologiích a nástrojích v oblasti zabezpečení a automatizace přenosu dat.

Pro dosažení stanovených cílů bude navrženo a implementováno řešení pro zabezpečený a automatizovaný přenos dat pomocí Powershell skriptů a automatizačních nástrojů. V této fázi bude vytvořeno několik konkrétních scénářů využití, na nichž budou provedeny experimenty a vyhodnocení výsledků.

4 Teoretická část

4.1 Historie internetu a přenosu souborů

Začátek bakalářské práce, věnovaný historii internetu a přenosu souborů, je klíčový, neboť poskytuje pevný základ pro pochopení současného stavu této oblasti a jejího vývoje. Přístup k informacím a sdílení dat prostřednictvím internetu se stalo zásadním prvkem moderní společnosti, přičemž znalost historického kontextu umožňuje pochopit proměny, kterými tato globální síť prošla a jakým způsobem ovlivňovala každodenní život lidí. Studium historie internetového přenosu dat a zabezpečení souborů nám umožňuje nejen reflektovat minulé úspěchy a výzvy, ale i identifikovat trendy a perspektivy pro budoucí vývoj. Tímto způsobem je pochopení historického kontextu nezbytné pro efektivní navrhování a implementaci moderních řešení v oblasti automatizace přenosu souborů a jejich zabezpečení, což je klíčovým cílem této práce.

Internet a přenos souborů představují základní pilíře moderního digitálního světa, který umožňuje rychlou výměnu informací a dat mezi uživateli po celém světě. Definovat internet lze jako globální síť propojující miliardy zařízení, která používají standardizované komunikační protokoly, umožňující přenos dat napříč různými geografickými oblastmi a různými platformami. Jak uvádí tato definice, internet je "*Termín Intersíť(Internet) se používá pro označení několika sítí propojených prostřednictvím některého mechanismu pro propojování síťových segmentů, nejčastěji směrovači*."[1] Tato definice zdůrazňuje univerzální povahu internetu a jeho základní funkci jako prostředku komunikace a výměny dat mezi různými zařízeními.

4.1.1 Vývoj přenosových technologií od počátků až po současnost

Ministerstvo obrany Spojených států amerických sehrálo klíčovou roli ve vzniku internetu prostřednictvím svého projektu nazvaného ARPANET. ARPANET, který byl první decentralizovanou síťí propojující počítače na vědeckých institucích a výzkumných centrech financovaných ministerstvem obrany USA. Cílem

ARPANETu bylo vytvořit robustní komunikační síť. ARPANET tedy položil základy pro moderní internet, ačkoli jeho původním účelem byla armádní i vědecké spolupráce.[2]

Během vývoje internetu a přenosu souborů se objevilo několik významných událostí, které formovaly současnou podobu těchto technologií. Po ARPANETu v roce 1972 následovalo spuštění elektronické pošty(email), která je považována za nejvýznamější internetovou aplikaci. V letech 1973-1979 byly položeny základy pro otevřenou architekturu propojování sítí TCP/IP. Pravidla TCP/IP však v průběhu času dostála pár změn. Do experimentálního provozu v síti ARPANETu se architektura TCP/IP dostala až v roce 1980 a 1.1. v roce 1983 následovalo oficiální uvedení architektury TCP/IP do provozu a tím tak logický počátek internetu. V roce 1994 se internet komercializuje a dochází tak k nárustu objemu digitálních dat a rostoucí nároky na rychlost, spolehlivost a bezpečnost dat zapříčinily vznik Internetu 2. Internet 2 je vysokorychlostní síťová infrastruktura a výzkumná síť, navržená pro podporu pokročilých výzkumných projektů a akademické spolupráce s důrazem na inovace.[1]

Historie internetu poskytuje kontext pro vývoj protokolu FTP (File Transfer Protocol). Vznik internetu a jeho rané fáze, včetně ARPANETu v 60. a 70. letech, představovaly základní infrastrukturu pro vývoj nových technologií pro sdílení a přenos dat. V této době byly definovány základní principy přenosu souborů, které vedly k vytvoření FTP.

FTP, zkratka pro File Transfer Protocol, je protokol navržený pro přenos souborů mezi počítačovými systémy připojenými k síti. Jeho hlavními cíli jsou:

podpora sdílení souborů (programů a/nebo dat),

podpora nepřímého nebo implicitního (prostřednictvím programů) využívání vzdálených počítačů,

odstínění uživatele od rozdílů v systémech ukládání souborů mezi hosty a

spolehlivý a efektivní přenos dat.

FTP je navržen především pro použití programy, ale může být použit i přímo uživatelem na terminálu. Jeho specifikace se snaží splnit různorodé potřeby uživatelů různých typů počítačů, včetně maxi-hostů, mini-hostů, osobních

pracovních stanic a terminálů, pomocí jednoduchého a snadno implementovatelného návrhu protokolu.

FTP prošel významným vývojem, jak dokumentuje Příloha III s chronologickou kompilací RFC dokumentů. Začíná to s RFC 114 v roce 1971, které navrhlo původní mechanismy přenosu souborů. Další revize, jako RFC 172, 265 a 281, přinesly další vylepšení. V roce 1982 byla navržena transakce "Nastavit typ dat" v RFC 294. RFC 354 poté obsoletní RFCs 264 a 265. Významné změny přineslo RFC 542 v červenci 1973. RFC 765 pak specifikoval FTP pro použití na TCP, přinášející nové příkazy a vylepšení.[3]

4.2 Zabezpečení přenosu souborů

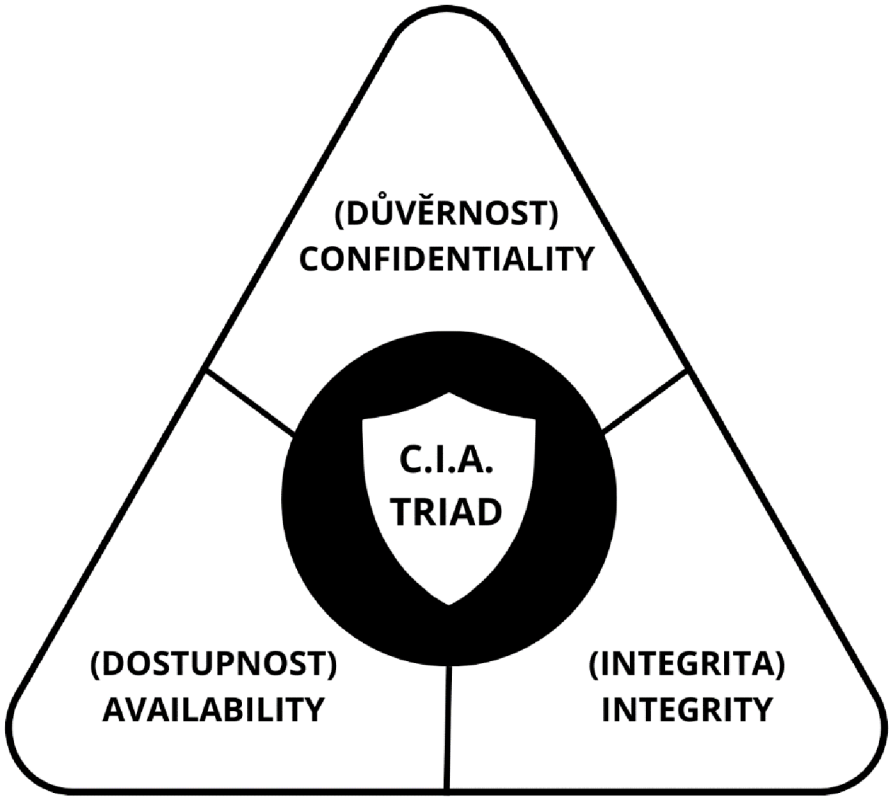
Druhá část této práce, která se zaměřuje na zabezpečení přenosu souborů, je neodmyslitelně spojena s historií internetu a vývojem protokolů aplikační vrstvy TCP/IP. Vzhledem k historickému vývoji internetu a jeho protokolů, jako je FTP, který byl považován za základní prostředek pro přenos souborů, je zabezpečení klíčové pro ochranu citlivých dat přenášených přes síť. S ohledem na moderní bezpečnostní hrozby, které se vyskytují v internetovém prostředí, je nezbytné zkoumat metody zabezpečení na úrovni aplikační vrstvy TCP/IP, jako je šifrování dat, ověřování identity a kontrola přístupu, aby byla zachována důvěrnost, integrita a dostupnost dat během jejich přenosu. Proto bude tato část práce analyzovat nejen historii internetu a přenosu souborů, ale také současné trendy a technologie v oblasti zabezpečení souborového přenosu v rámci aplikační vrstvy TCP/IP.

4.2.1 CIA Triad

Důvěrnost, integrita a dostupnost, z anglického překladu Confidentality, Integrity, Availability, jsou tři složky CIA triády, modelu informační bezpečnosti navrženého k ochraně citlivých informací před únikem dat.
CIA triáda je důležitým konceptem v oblasti informační bezpečnosti a používá se v normě ISO 27001, celosvětovém standardu pro řízení informační bezpečnosti.
GDPR také zmiňuje CIA triádu v článku 32, který vyžaduje, aby organizace používaly vhodná opatření k ochraně důvěrnosti, integrity, dostupnosti a odolnosti svých informačních systémů a služeb.[4]

Jednotlivé principy této triády existovaly již před vznikem počítačových dat v polovině dvacátého století a byly nezávislé využívány v oblasti bezpečnosti dat. Přesné datum vzniku této koncepce jako triády není známo.
Termín "CIA Triáda" se poprvé objevil v knize z roku 1998 nazvané "Boj proti počítačové kriminalitě" a zdá se, že se stal standardem mezi bezpečnostními praktikami té doby. Bez ohledu na to, kdy byla myšlenka Triády poprvé formulována, principy jsou již dlouho využívány bezpečnostními profesionály, kteří chápou potřebu zvýšit bezpečnost informací. Efektivní ochrana digitálních aktiv začíná právě těmito principy. Všechny tři zásady jsou nezbytné pro ochranu dat, a

bezpečnostní incident ohrožující jednu z těchto složek může vést k problémům i u ostatních. Přestože důvěrnost a integrita jsou často vnímány jako protichůdné v kybernetické bezpečnosti (například šifrování může ohrozit integritu), je třeba je vyvážit proti rizikům při návrhu bezpečnostního plánu.[5]



4-1 CIA TRIAD
Zdroj: Vlastní

4.2.1.1 Důvěrnost

Důvěrnost dat jako první komponent CIA triády je klíčová pro každou organizaci, která se snaží udržet citlivé informace v tajnosti a chránit je před neoprávněným přístupem. Aby organizace zajistily důvěrnost, musí pečlivě kontrolovat přístup k datům a zavést účinná opatření k prevenci neoprávněného sdílení dat, ať už záměrného nebo náhodného. Útoky na důvěrnost dat mohou být různého charakteru - mohou se jednat o přímé útoky, jako je proniknutí do systémů, nebo o důsledek lidské chyby nebo nedostatečných bezpečnostních opatření. K ochraně důvěrnosti dat organizace mohou využít různé nástroje a postupy, včetně

klasifikace dat, implementace striktního řízení přístupu, šifrování dat a implementace systémů vícefaktorové autentizace (MFA). Kromě technologických opatření je důležité také poskytnout zaměstnancům odpovídající školení a povědomí o bezpečnostních rizicích, aby byli schopni rozpoznat a minimalizovat možné hrozby pro důvěrnost dat.

4.2.1.2 Integrita

Integrita dat jako druhý komponent CIA triády je zásadní pro zajištění důvěryhodnosti a neporušenosti informací. Bez integrity může docházet k různým útokům a manipulacím, které mohou poškodit důvěryhodnost a spolehlivost dat. Útoky na integritu mohou být prováděny jak úmyslně, tak neúmyslně. K úmyslnému narušení integrity může docházet prostřednictvím pokusů o změnu dat nebo konfigurací s cílem zpochybnit jejich autentičnost nebo spolehlivost. To může mít zásadní dopad na důvěru uživatelů a pověst organizace. Nesprávné konfigurace, nedostatečná ochrana nebo lidské chyby mohou také vést k náhodnému narušení integrity dat.

K zajištění integrity dat mohou organizace využít různé bezpečnostní mechanismy a technologie. Mezi tyto patří hašování, šifrování, digitální podpisy a ověřování pravosti pomocí důvěryhodných certifikačních autorit. Důležitou roli hraje také ne-repudiační princip, který zabraňuje zpochybnění autenticity nebo původu dat. Zavedení těchto opatření a bezpečnostních postupů pomáhá organizacím chránit integritu svých dat a minimalizovat rizika manipulace či narušení datových záznamů.

4.2.1.3 Dostupnost

Dostupnost, jako poslední ze tří klíčových složek CIA triády, zajišťuje, že data a systémy jsou přístupné pro uživatele a zákazníky organizace, kdykoliv je potřeba. To vyžaduje, aby systémy, sítě a aplikace fungovaly správně a bezproblémově, a aby uživatelé měli snadný přístup k informacím bez zbytečného zpoždění. Dostupnost může být ohrožena různými faktory, včetně přírodních katastrof, technických poruch a úmyslných útoků, jako jsou DoS útoky nebo ransomware. Pro zajištění dostupnosti mohou organizace využívat redundantní sítě, servery a aplikace, a

mohou také provádět pravidelné aktualizace softwaru a bezpečnostních systémů. Dále mohou vytvářet zálohy a plány pro obnovu po havárii, aby mohly rychle obnovit dostupnost po negativní události.[6]

4.2.2 Způsoby zabezpečení souborů

Tato kapitola navazuje na předchozí kapitolu o CIA Triádě, která představuje základní koncepční rámec pro ochranu dat a informací v digitálním prostředí. Po pochopení principů důvěrnosti, integrity a dostupnosti je nyní klíčové prozkoumat konkrétní metody a strategie, které organizace mohou použít k zabezpečení svých systémů a dat. Tato kapitola je důležitá pro celkový kontext bakalářské práce, protože poskytne ucelený pohled na různé přístupy k zabezpečení a umožní porovnání jejich účinnosti a vhodnosti v konkrétních scénářích. Zároveň bude důležité zdůraznit, že zabezpečení je dynamický proces, který vyžaduje neustálou aktualizaci a adaptaci na nové hrozby a technologické vývoje. Porozumění různým způsobům zabezpečení a jejich aplikace ve spolupráci s principy CIA Triády je klíčové pro efektivní ochranu informací a dat v organizaci.

4.3 Protokoly

V digitální éře, kde se informace stávají nepostradatelným zdrojem pro jednotlivce i organizace, je zajištění bezpečnosti přenosu dat klíčovým prvkem. Bezpečný a spolehlivý přenos souborů a komunikace prostřednictvím různých protokolů je základem moderního informačního toku. V této kapitole se zaměříme na analýzu a popis dvou hlavních kategorií protokolů, které se významně podílejí na zajištění bezpečnosti a efektivity přenosu dat.

4.3.1 Protokoly pro přenos souborů

První podkapitola bude věnována protokolům určeným pro přenos souborů. Mezi tyto protokoly patří FTP (File Transfer Protocol), SFTP (SSH File Transfer Protocol) a FTPS (FTP Secure) Tato část se zabývá mechanismy, které tyto protokoly využívají k efektivnímu a bezpečnému přenosu dat mezi různými uzly v síti, a porovná jejich vlastnosti a vhodnost pro různé použití.

4.3.1.1 FTP

File Transfer Protocol (FTP) je standardní síťový protokol používaný pro přenos souborů mezi počítači připojenými k síti podle architektury TCP/IP. Tento protokol aplikační vrstvy TCP/IP umožňuje uživatelům jednoduchý a spolehlivý způsob sdílení a přenosu souborů mezi různými zařízeními bez ohledu na jejich operační systém, avšak uživatelské rozhraní bývá charakteristické podle zvyklostí operačního systému. FTP je široce používaným nástrojem v oblasti informačních technologií a je klíčový pro různé účely, včetně správy serverů nebo možností uživatelům přistupovat k souborům uloženým na vzdálených serverech a provádět různé operace s těmito soubory, jako je jejich stahování, nahrávání, přejmenování nebo mazání.

Princip fungování FTP spočívá v navázání spojení mezi klientem a serverem, kde klient pošle příkazy (například stahování, nahrávání nebo odstranění souborů) a server na ně reaguje odpověďmi. Pro přenos dat FTP používá dva kanály: příkazový pro zasílání požadavků a datový pro samotná data. To umožňuje efektivní a spolehlivý přenos souborů bez ztráty integrity dat.

Příkazový kanál funguje na principu, ve kterém klient odesílá řídící příkazy serveru přes tento kanál. Tyto příkazy řídí průběh FTP relace, jako je například autentizace uživatele, změna pracovního adresáře, zobrazení obsahu adresáře nebo požadavek na přenos souboru. Pro spojení očekává server příkazy na portu 21/tcp.

Princip druhého z nich, datového kanálu, spočívá v tom, že server po obdržení příkazu pro přenos souboru otevírá datový kanál, přes který jsou samotná data přenášena mezi klientem a serverem. Datový kanál může být otevřený buď pro čtení (pro stahování souborů) nebo pro zápis (pro nahrávání souborů).

Klient a server komunikují pomocí sady standardních FTP příkazů a odpovědí. Například pro stahování souboru klient odešle příkaz get následovaný názvem požadovaného souboru, a server odpoví odpovědí obsahující data souboru nebo informací o chybě. Datový kanál je specifický proto, že má možnost obrátit roli klient server. U datového kanálu proto rozlišujeme dva režimy, ve kterých protokol FTP komunikuje. Aktivní a pasivní. Hlavním rozdílem mezi nimi je směr iniciace datového spojení.

V aktivním módu FTP spojení, klientovi je přidělen jakýkoliv port s větším číslem než 1023 a klient iniciuje datové spojení se serverem.

Klient odesílá příkaz PORT obsahující IP adresu a číslo portu, na kterém bude poslouchat pro příchozí datová spojení.

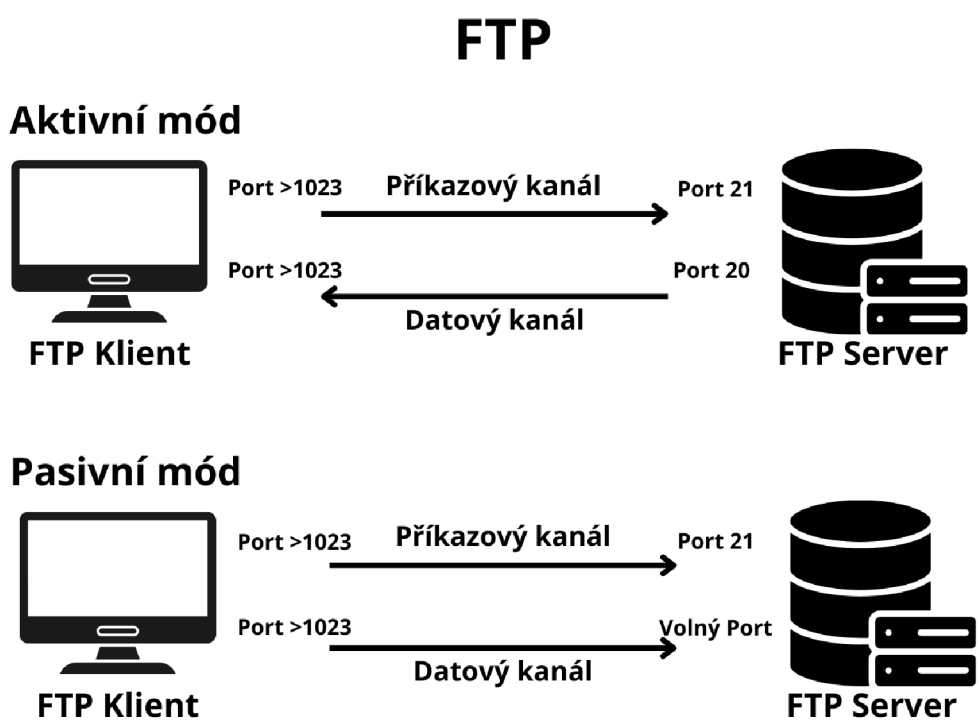
Server se poté pokouší navázat datové spojení s klientem pomocí adresy a portu poskytnutého v příkazu PORT.

V pasivním módu server iniciuje datové spojení s klientem.

Po obdržení příkazu PASV od klienta server otevře nový port, zašle klientovi šestici čísel, přičemž první čtyři čísla reprezentují IP adresu serveru a zbylá dvě čísla port. Server poté naslouchá na portu pro příchozí datová spojení.

Klient se poté pokusí navázat datové spojení se serverem pomocí poskytnuté adresy a portu.

Celkově se aktivní a pasivní mód FTP odlišují v tom, kdo iniciuje datové spojení. V aktivním módu iniciátorem je klient a v pasivním módu je to server. Pasivní mód je často doporučován jako volba pro spolehlivější a kompatibilnější přenos souborů přes internet při ochraně například před filtrací na přístupovém směrovači. Kromě typu kanálu FTP protokol rozeznává i typ přenášeného souboru.



4-2 FTP
Zdroj: Vlastní

FTP protokol rozlišuje až čtyři typy přenášených souborů, ale obvykle jsou implementovány pouze dva z nich, a to ASCII a binární. Tyto typy určují způsob, jakým jsou data v souborech interpretována a přenášena mezi klientem a serverem.

American Standard Code for Information Interchange(dále jen ASCII) mód je vhodný pro přenos textových souborů, které obsahují pouze znaky ASCII. Při přenosu v ASCII módu jsou data interpretována jako text a jsou aplikována různá pravidla pro kódování a přenos koncových znaků. ASCII mód se používá pro přenos souborů, jako jsou textové dokumenty nebo zdrojové kódy programů.

Binární mód je vhodný pro přenos všech ostatních typů souborů, které nejsou čistě textové a/nebo mohou obsahovat binární data, jako jsou obrázky, zvukové soubory, spustitelné soubory atd. Při přenosu v binárním módu jsou data přenášena v jejich původní podobě bez jakýchkoli změn. To znamená, že nejsou aplikována žádná kódování ani transformace dat.

Výběr mezi ASCII a binárním modem je důležitý, protože pokud je použit nesprávný mód, mohou být data poškozena nebo zkreslena. Použití správného módu zajišťuje,

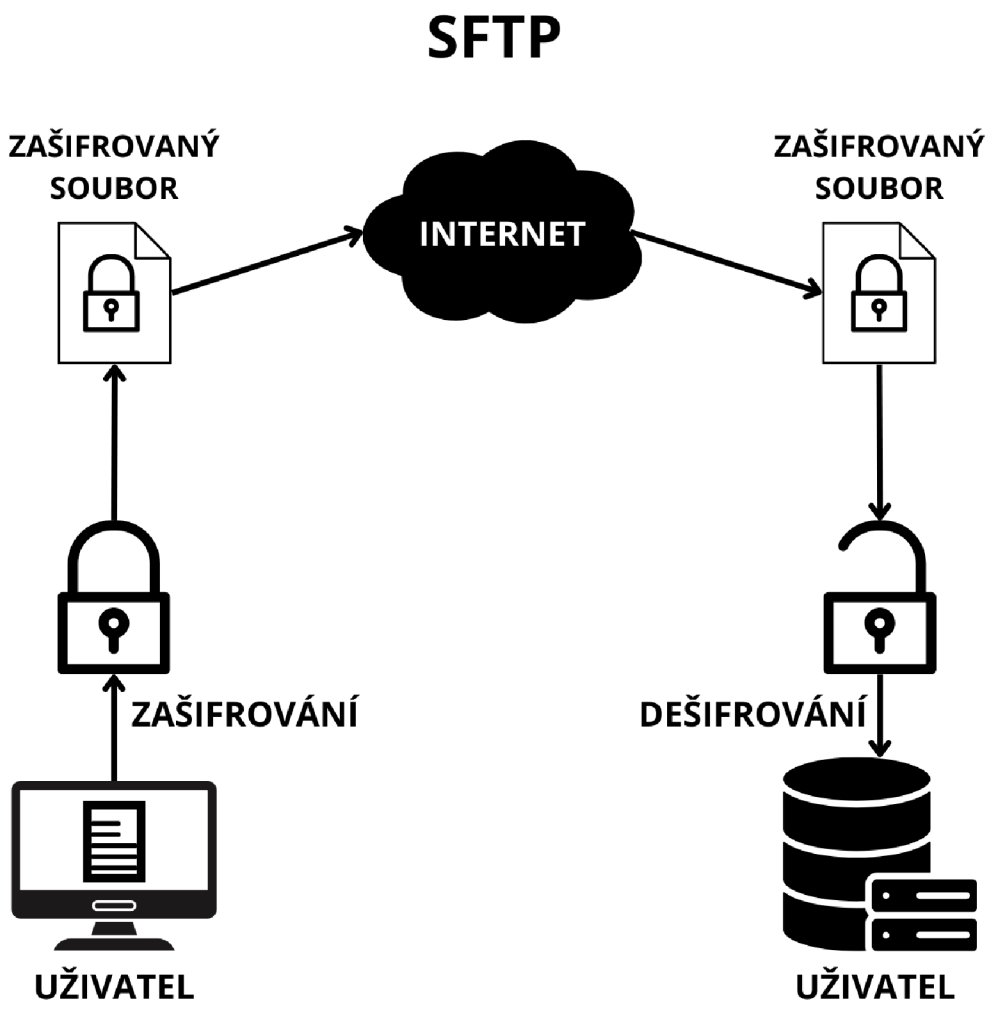
že soubory jsou přeneseny přesně tak, jak byly uloženy na původním serveru, a zachovává jejich integritu a strukturu. [7]

4.3.1.2 SFTP

SFTP (SSH File Transfer Protocol) je bezpečný protokol pro přenos souborů, který využívá šifrovací mechanismy protokolu SSH (Secure Shell) k zajištění zabezpečené komunikace mezi klientem a serverem. Podobně jako FTP, umožňuje SFTP uživatelům přenášet soubory mezi svým zařízením a vzdáleným serverem.
SFTP se připojuje na standardní port 22/tcp, který je definován pro protokol SSH. Tím je zajištěno, že komunikace probíhá přes zabezpečené spojení.

Na rozdíl od FTP nebo FTPS, SFTP využívá jediný kanál pro komunikaci mezi klientem a serverem, který slouží k přenosu příkazů, odpovědí a dat. Tento kanál je šifrovaný a chráněn proti odposlechu a manipulaci, což zajišťuje integritu a bezpečnost dat během celého přenosového procesu.

Dalším významným rozdílem mezi SFTP a FTP je to, že SSH je paketový nebo kryptografický protokol, který šifruje pakety dat před jejich odesláním.
SFTP není jen pro přenos souborů, ale pro přístup ke vzdáleným serverům, což znamená, že je více podobný protokolu souborového systému než například SCP .
Ačkoli SFTP představuje pokročilou alternativu k protokolu FTP, která poskytuje vysokou úroveň zabezpečení a ochrany dat během přenosu, SFTP a FTPS (Secure FTP) jsou dvě odlišné metody pro zabezpečený přenos souborů, které by neměly být zaměňovány. Zatímco SFTP je protokol, který šifruje veškerá data a vychází z protokolu SSH, FTPS se od běžného FTP liší tím, že přidává SSL/TLS kryptografii. SFTP je tedy spíše rozšířením SSH. [8]



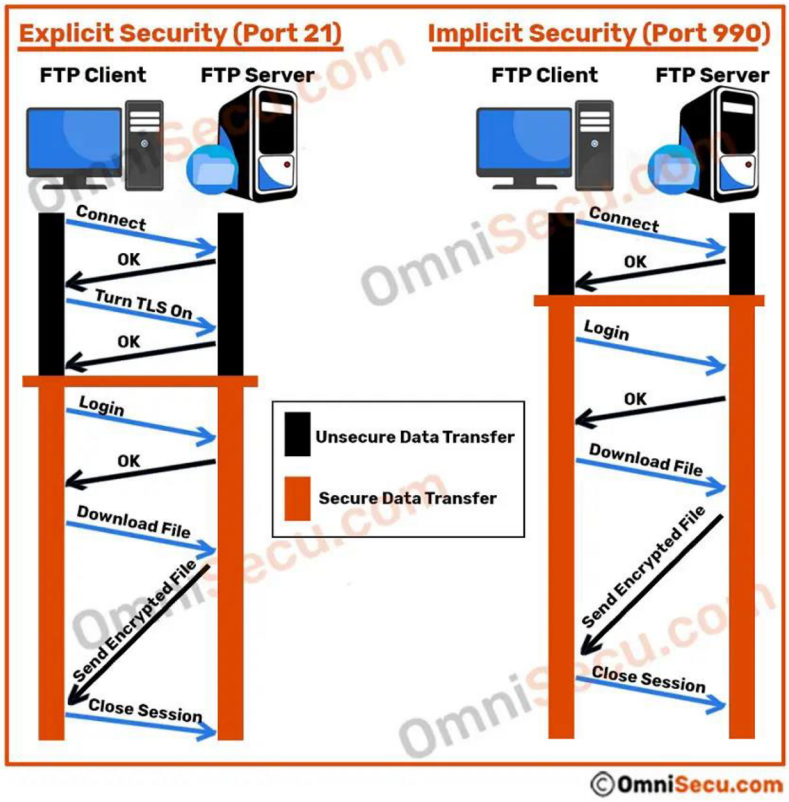
4-3 SFTP
Zdroj: Vlastní

4.3.1.3 FTPS

FTPS (FTP Secure) je zabezpečený protokol pro přenos souborů, který přidává SSL/TLS kryptografii k běžnému protokolu FTP (File Transfer Protocol). Tato šifrování poskytují zabezpečenou komunikaci mezi klientem a serverem, čímž chrání přenášená data před odposlechem a útoky. Podobně jako protokol FTP používá opět dva kanály pro komunikaci: řídící a datový.

FTPS nabízí dva základní režimy provozu: explicitní a implicitní. Explicitní FTPS, častěji používaný režim, zahajuje standardní FTP spojení na výchozím portu 21 a následně na stejném portu vytváří zabezpečené SSL spojení. Tento režim poskytuje uživatelům flexibilitu při stanovení rozsahu šifrování požadovaného pro

přenos dat. Naopak implicitní FTPS vytváří SSL spojení okamžitě na portu 990, šifruje celou FTP relaci od začátku. Zatímco explicitní FTPS je široce podporován a nabízí větší flexibilitu, implicitní FTPS je přísnější v bezpečnostních opatřeních, ale je považován za zastaralý ve prospěch explicitního FTPS.



4-4 FTPS
Zdroj: OmniSecu [13]

FTPS přináší několik výhod, včetně snadné integrace s existující infrastrukturou FTP a robustních bezpečnostních funkcí poskytovaných protokoly SSL/TLS. To umožňuje organizacím chránit svá data před neoprávněným přístupem a úniky informací.[9]

4.3.2 Protokoly elektronické pošty a jejich zabezpečení

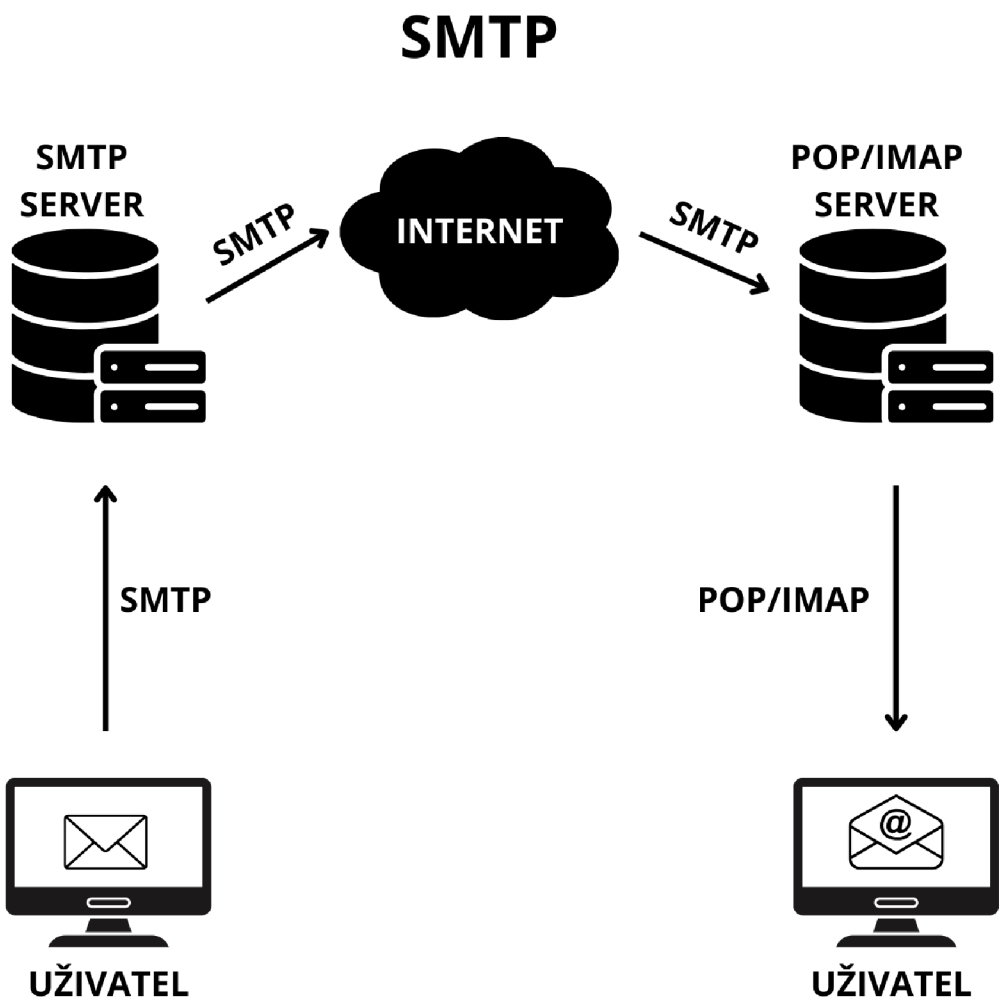
Tato kapitola se zaměří na protokoly elektronické pošty, konkrétně SMTP, POP a IMAP, a jejich zabezpečení. V kontextu této bakalářské práce je důležité porozumět těmto protokolům a jejich bezpečnostním aspektům, protože

elektronická pošta je jedním z klíčových komunikačních kanálů v dnešní digitální době.

SMTP (Simple Mail Transfer Protocol) je protokol pro přenos elektronických zpráv přes síť, zatímco POP3 (Post Office Protocol 3) nebo IMAP4 (Internet Message Access Protocol 4) jsou protokoly pro přístup ke zprávám uloženým na serveru. [1] Detailní analýza těchto protokolů umožní porozumět jejich fungování, zranitelnostem a možnostem zabezpečení. Důležité bude zkoumat, jaký vliv mají různé zabezpečovací mechanismy, jako je šifrování, ověřování identity a ochrana proti spamu, na celkovou bezpečnost elektronické pošty.

4.3.2.1SMTP

SMTP neboli Simple Mail Transfer Protocol je aplikační protokol, který umožňuje přenos e-mailů mezi různými servery a počítačovými sítěmi tím, že definuje pravidla komunikace. Původní model byl představen v roce 1982. Podle RFC 821 uživatel vytvoří žádost o spojení. Na to reaguje odesílací SMTP inicializací obousměrného spojení s přijímacím SMTP. V moderním kontextu jsou tyto dva entity označovány jako SMTP klient a SMTP server. SMTP klient a SMTP server komunikují pomocí příkazů a odpovědí, podobně jako v reálném životě. RFC 821 také definoval model pro použití SMTP. Jakmile je spojení navázáno, SMTP klient postupně přenáší hlavičky, příjemce, těla zpráv (včetně příloh) a všechna data na SMTP server. Když je přenos dokončen, spojení se uzavře. Podle definice plné formy SMTP klienti podporují přeposílání, fronty e-mailů a alternativní adresní funkce. To se nazývá plně schopný SMTP. Pokud tyto funkce nejsou podporovány, SMTP není plně schopný. V tomto případě příslušná RFC doporučuje použití protokolu pro odesílání zpráv. Mějte na paměti, že SMTP může posílat pouze jednoduché zprávy, tj. prostý text bez příloh. K odesílání příloh, zpráv s těly přesahujícími maximální počet znaků stanovený SMTP, zpráv v jiných jazycích než angličtině a formátů HTML/CSS používáme samostatný protokol, Multipurpose Internet Mail Extensions, MIME (RFC 2045). MIME je rozšiřující protokol, který rozšiřuje možnosti SMTP, ale nespolupůsobí samostatně. Většina moderních e-mailových služeb podporuje MIME.[10]



4-5 SMTP
Zdroj: Vlastní

4.3.2.2 POP a IMAP

Protokoly POP (Post Office Protocol) a IMAP (Internet Message Access Protocol) jsou oba navrženy pro přístup ke zprávám uloženým na e-mailovém serveru, avšak se liší v jejich přístupových metodách a funkcionalitě. V této části si blíže popíšeme oba protokoly a hlavní rozdíly mezi nimi.

IMAP (Internet Message Access Protocol) je preferován uživateli, kteří potřebují přístup ke svým e-mailům z více zařízení. IMAP nevyžaduje stahování ani ukládání e-mailů na uživatelovo zařízení. Místo toho uživatelé čtou své e-maily přímo ze služby e-mailu. To umožňuje uživatelům kontrolovat své e-maily z různých

zařízení kdekoliv ve světě, včetně chytrých telefonů, počítačů nebo dokonce dočasných přístupových zařízení (jako je například počítač kolegy). V IMAP jsou e-maily obvykle stahovány až poté, co je uživatel „otevře“. Přílohy se také automaticky nestahují, což šetří internetové šířky pásma a výpočetní výkon na uživatelově zařízení. To uživatelům poskytuje rychlý a plynulý zážitek při prohlížení jejich zpráv.

POP3 (Post Office Protocol version 3) funguje tím, že naváže spojení se službou e-mailu a stáhne všechny nové e-maily na koncové zařízení. Jakmile je stahování dokončeno, jsou e-maily obvykle smazány ze služby, pokud administrátor neupraví konfiguraci tak, aby tomu zabránil. V podstatě, jakmile jsou e-maily staženy, lze k nim přistupovat pouze na stejném koncovém zařízení. Pokud by se uživatel pokusil připojit k e-mailové službě z jiného zařízení, nebylo by možné přistupovat k dříve staženým zprávám na novém zařízení, pokud server není specificky nakonfigurován tak, aby si zprávy uchoval. Když uživatel odešle e-mail přes POP3, je také uložen lokálně na jeho koncovém zařízení a e-mailová služba si kopii neuchovává (pokud není specificky nakonfigurována tak, aby to udělala).

Z porovnání těchto dvou protokolů bylo zjištěno, že IMAP umožňuje pokročilejší správu e-mailů a synchronizaci napříč mnoha zařízeními, zatímco POP3 je vhodnější pro konfigurace, kde je třeba přistupovat k e-mailům pouze z jednoho zařízení. (Zatímco IMAP ukládá e-maily na serveru a umožňuje jejich správu přímo na serveru, POP3 stahuje e-maily na lokální zařízení a po stažení je obvykle vymaže ze serveru. IMAP také umožňuje přístup ke všem e-mailům kdykoliv a odkudkoliv, zatímco POP3 umožňuje přístup pouze k e-mailům, které byly staženy na lokální zařízení.)

Co se týká funkcionalit, IMAP funguje synchronizací všech připojených zařízení s jedním centrálním serverem, což umožňuje uživatelům přistupovat ke stejným e-mailům a složkám z různých zařízení s kontinuitou stavu. Na rozdíl od toho POP3 umožňuje uživatelům stahovat e-maily na koncové zařízení, kde jsou poté uloženy pouze lokálně, a po stažení jsou odstraněny ze serveru, což omezuje jejich dostupnost na pouze jednom zařízení. Zatímco IMAP poskytuje uživatelům flexibilitu v přístupu ke svým e-mailům online, offline i v režimu odpojení, POP3 je ideální pro čtení e-mailů offline a nevyžaduje připojení k internetu k jejich

zobrazení. IMAP umožňuje uživatelům organizovat a spravovat e-maily přímo na serveru, včetně jejich třídění do různých složek a provádění vyhledávání v jejich obsahu, což je funkce, která chybí u POP3. Další rozdíly zahrnují možnost nastavení zprávových vlajek, flexibilitu v konfiguraci stahování e-mailů a možnost organizace složek na straně serveru. Nakonec, zatímco POP3 je jednodušší a populárnější pro určité účely díky své jednoduchosti, IMAP je častěji preferován uživateli, kteří potřebují přístup ke svým e-mailům z více zařízení a vyžadují pokročilejší správu a synchronizaci.

V porovnání zabezpečení mezi IMAP a POP3 je zřejmé, že IMAP poskytuje vyšší úroveň bezpečnosti než POP3 v mnoha ohledech. IMAP je založen na synchronizaci všech zařízení s jedním centrálním serverem, což znamená, že všechny změny jsou zpracovány na straně serveru, což umožňuje centralizované zabezpečení dat a snižuje rizika spojená s lokálním zpracováním na koncových zařízeních, jak je tomu u POP3. IMAP také nabízí pokročilé funkce, jako je podpora TLS pro šifrování komunikace a možnost volby mezi implicitním TLS nebo STARTTLS pro zabezpečení komunikace. Oproti tomu POP3 je považováno za méně bezpečné, zejména kvůli zranitelnostem spojeným s lokálním zpracováním e-mailů na koncových zařízeních a nedostatku podpory pro moderní bezpečnostní funkce, jako je synchronizace a ochrana před škodlivými přílohami. I když lze POP3 zabezpečit pomocí TLS nebo SSL, nedostatek podpory pro synchronizaci dat mezi zařízeními a další moderní funkce z něj činí méně bezpečnou volbu ve srovnání s IMAP. Vzhledem k těmto rozdílům v zabezpečení je IMAP často preferováno před POP3 ve více citlivých prostředích a pro uživatele, kteří vyžadují vyšší úroveň bezpečnosti a funkcí.[11]

4.3.2.3 Bezpečnostní technologie elektronické pošty

Bezpečnost elektronické pošty spočívá v zajištění čtyř základních složek: utajení poštovní zprávy, integrity poštovní zprávy, autenticity odesílatele a nezpochybnitelnosti odesílatele. Tyto aspekty jsou klíčové pro ochranu důvěrnosti, integritu a autenticitu elektronických zpráv. K dosažení těchto cílů jsou využívány různé bezpečnostní technologie, jako je Privacy Enhanced Mail (PEM), Pretty Good Privacy (PGP) a Secure/Multipurpose Internet Mail Extensions (S/MIME). Každá z

těchto technologií poskytuje šifrování zpráv, digitální podpisy a další bezpečnostní mechanismy, které umožňují uživatelům komunikovat prostřednictvím elektronické pošty s vysokou úrovní ochrany a důvěryhodnosti.

Mechanismus PEM byl nejznámějším bezpečnostním mechanismem, nicméně dnes je na něj nahlíženo spíše jako na řešení s historickým významem. Umožňuje šifrování a digitální podepisování e-mailových zpráv. Šifrování zajišťuje, že obsah e-mailů je chráněn a není čitelný pro neoprávněné osoby, zatímco digitální podpisy umožňují ověřit autenticitu, integritu i nezpochybnitelnost autora zpráv, čímž se zabrání falšování nebo modifikaci obsahu zpráv.

PGP je šifrovací program, implementace PEM, který využívá asymetrickou kryptografii, kde každý uživatel má svůj pár veřejného a privátního klíče. Uživatelé mohou šifrovat své zprávy pomocí veřejného klíče adresáta, který pak použije svůj privátní klíč k dešifrování zprávy. Veřejné klíče si program PGP spravuje sám. Vygenerovaný veřejný klíč se posílá spolu se zprávou, která je tímto tajným klíčem zašifrována. Veřejný klíč je distribuován a je určen k šifrování zpráv, zatímco soukromý klíč zůstává v držení uživatele a slouží k dešifrování zpráv. Rizikem tohoto programu může být správa klíčů, které bývají uloženy v souborech v adresářích uživatelů a mohou být tím pádem vyzrazeny, pokud bude veden útok na souborový systém.

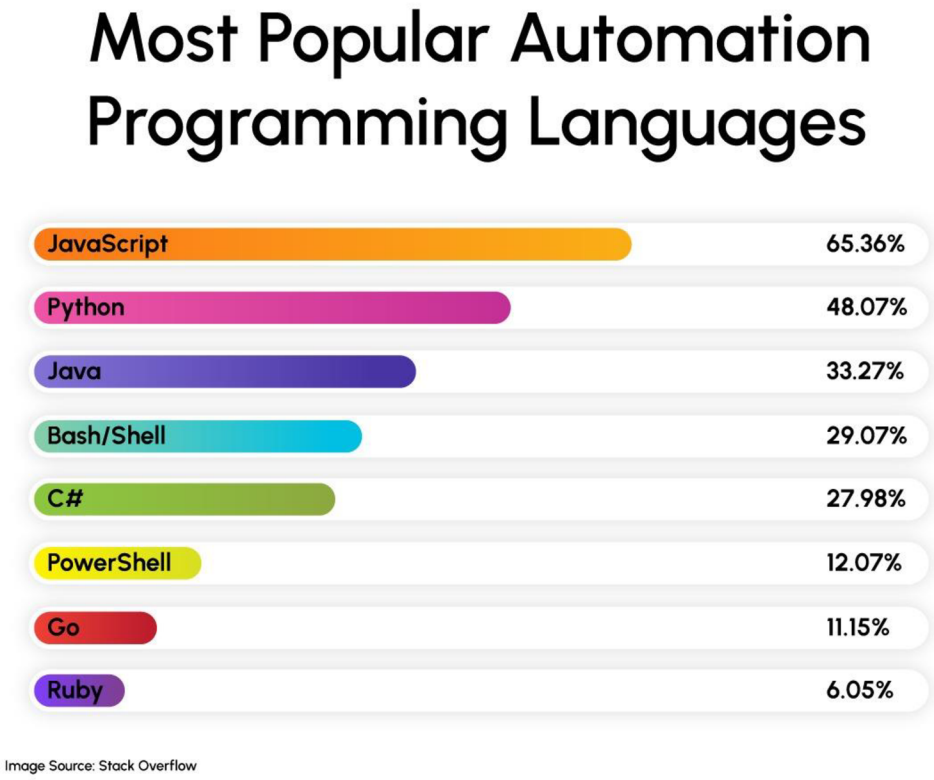
Bezpečnostní rozšíření elektronické pošty, známé jako S/MIME, specifikuje technologii šifrování použitou pro vytvoření šifry integrované do zprávy. Organizace Internet Society projevuje značný zájem o vývoj S/MIME a situace napovídá tomu, že S/MIME se stane průmyslovým standardem pro komerční využití, zatímco PGP zůstane aplikací pro individuální potřeby uživatelů internetu.[1]

4.4 Automatizace přenosu souborů

Automatizace přenosu souborů hraje klíčovou roli v moderním IT prostředí, umožňující efektivní správu a bezpečný přenos dat mezi různými systémy a uživateli. Tato kapitola se zaměřuje na metody a nástroje, které umožňují automatizovat tento proces s důrazem na zabezpečení a spolehlivost. V dnešní době existuje široká škála programovacích jazyků, nástrojů a softwaru, které lze využít k automatizaci přenosu souborů.

Populární programovací jazyky používané pro automatizaci business procesů (BPA) zahrnují Python, JavaScript, Bash/Shell scripting, PowerShell, Java, Ruby, Go (Golang) a C#. Tyto jazyky se liší v použití a výhodách, které nabízejí. Python se vyznačuje jednoduchou syntaxí a je vhodný pro širokou škálu úloh včetně webového scrapování, testování a správy systému. JavaScript je preferovaný pro automatizaci webových a prohlížečových interakcí, zatímco Bash/Shell scripting je vhodný pro automatizaci správy systému. PowerShell je určen pro automatizaci úloh v prostředí Windows, zatímco Java je často používána pro podnikovou automatizaci. Ruby se vyznačuje elegantní syntaxí a je vhodný pro webovou automatizaci a testování. Go (Golang) je preferovaný pro automatizaci v cloudu a mikroslužeb díky svým schopnostem současného zpracování. C# je často využíván v ekosystému Microsoftu pro automatizaci úloh týkajících se aplikací a služeb na platformě Windows.

Organizace často kombinují různé jazyky a nástroje pro efektivní automatizaci různých aspektů svých procesů. Důležité je vybrat jazyk, který nejlépe odpovídá potřebám organizace a umožňuje efektivní automatizaci jejích obchodních procesů.[12]



4-6 Populární jazyky pro automatizaci
Zdroj: [12]

Vzhledem ke zkušenostem autora je pro praktickou část zvolen Windows Powershell.

4.4.1 Powershell

PowerShell, vyvinutý společností Microsoft, představuje objektově orientovaný automatizační engine a skriptovací jazyk s interaktivním příkazovým řádkem. Jeho účelem je usnadnit konfiguraci systémů a automatizaci administrativních úkolů pro IT profesionály. Tento nástroj, založený na .NET frameworku, pracuje s objekty, což ho odlišuje od většiny příkazových řádek, jež jsou založeny na textu. Díky svým skriptovacím schopnostem se PowerShell stal významným nástrojem pro systémové administrátory, nejen v odděleních IT, ale i v externích subjektech, jako jsou poskytovatelé spravovaných služeb.

Původně dostupný pouze na platformě Windows, PowerShell se postupně rozšířil a je nyní k dispozici i pro Linux a macOS. Microsoft otevřel zdrojový kód PowerShellu v roce 2016, čímž umožnil jeho širší použití a integraci do různých operačních systémů. PowerShell nabízí uživatelům několik možností automatizace úkolů, včetně práce s cmdlety, skripty, vykonatelnými soubory a standardními .NET třídami. Díky těmto možnostem mohou administrátoři snadno extrahovat informace z operačních systémů, manipulovat se soubory a registry nebo provádět jiné úkoly.

Jedním z hlavních důvodů pro používání PowerShellu je jeho schopnost poskytnout uživatelům přesnou a opakovatelnou kontrolu nad úkoly. Na rozdíl od grafického uživatelského rozhraní (GUI), které může být časově náročné a náchylné k chybám, PowerShell umožňuje vytvářet detailní skripty pro opakované úkoly. Díky pipeliningu a dalším funkcím mohou uživatelé provádět složité operace s vysokou mírou automatizace a přesnosti.

PowerShell je také vybaven funkcemi pro objevování, poskytování nápovědy a provádění vzdálených příkazů, což zvyšuje jeho flexibilitu a použitelnost v různých prostředích. Tento nástroj se stal nedílnou součástí systémové správy a automatizace, a jeho význam dále roste s každou novou verzí a aktualizací. [14]

4.4.1.1 Powershell skripty ve Windows

V PowerShellu jsou rutiny klíčovými stavebními bloky pro vytváření skriptů. Tyto rutiny jsou jednofunkčními příkazy, které lze použít samostatně nebo v kombinaci k provádění složitějších úloh. Každá rutina funguje jako samostatná funkce, která přispívá k výsledku skriptu, a je základním prvkem v PowerShellu, umožňujícím programátorům a správcům IT vytvářet efektivní a modulární skripty.

Skripty PowerShellu jsou vytvářeny pomocí rutin a slouží k automatizaci různých úkolů. Rutiny jsou tvořeny dvěma částmi. Slovesem a podstatným jménem. Jako příklad lze uvést Add, Format nebo nejpoužívanější tři typy příkazů v skriptech PowerShellu: příkaz „get“, který tvoří více než dvacet pět procent všech rutin, je využíván pro načítání dat ze systému souborů, příkaz „set“ pro úpravu informací o komponentě systému Windows a příkaz „remove“ pro úplné odstranění operací. Tyto skripty snižují složitost kódu při psaní a umožňují efektivní automatizaci

různých procesů a úloh. S použitím rutin, funkcí a skriptů v PowerShellu je možné efektivně spravovat a konfigurovat systém Windows a automatizovat opakující se úkoly v IT prostředí.

Typickou příponou pro Powershell skripty je „.ps1“. Pro spouštění skriptů Windows Powershell je nutné nastavení jejich podpory, protože výchozí nastavení spouštění skriptů neumožňuje. Povolení pro spouštění skriptů lze konfigurovat i pomocí již zmíněných rutin, tedy konkrétně rutinou Set-ExecutionPolicy, která nabízí šest úrovní pro podporu spouštění skriptů. První z nich a zároveň výchozí úrovní je úroveň „Restricted“, která nepovoluje načítání jak konfiguračních souborů a taktéž skriptů. Úroveň „AllSigned“ nařizuje podepisování veškerých skriptů, včetně lokálních, důvěryhodným vydavatelem. V úrovni „RemoteSigned“ je nutné podepsání konfiguračních souborů i skriptů stažených z internetu důvěryhodným vydavatelem. Úroveň „Unrestricted“ načítá a spouští veškeré konfigurační soubory i skripty, pouze zobrazuje žádost o potvrzení u nepodepsaných souborů stažených z internetu. Nejvolnější úroveň „Bypass“ neblokuje jediný soubor a zároveň nezobrazuje jediné upozornění. Poslední úrovní „Undefined“ je spouštění odebráno z aktuálně používaného oboru.

Pro spouštění jsou definovány tři obory pravidel. Pravidlo „Process“, které reguluje spouštění daného procesu prostředí Powershell. Pravidlem „CurrentUser“ je regulováno spouštění pouze pro současného uživatele. Posledním pravidlem „LocalMachine“, pro které je nutné nabývat práv správce počítače a vztahuje se tedy na všechny uživatele daného počítače.

Po zřízení pravidla pro spouštění je vhodné využít rutiny Get-ExecutionPolicy k získání informací o aktuálně platném pravidle spouštění skriptů.[15]

4.4.2 Nástroje pro automatizaci skriptů

Automatizace skriptů je v současné době klíčovým aspektem v oblasti správy IT infrastruktury a procesů. Tato kapitola si klade za cíl zkoumat nástroje a metody, které umožňují automatizaci skriptů, s důrazem na využití PowerShellu.

Pro plánování a spouštění automatizovaných úloh je nezbytné používat vhodné nástroje pro plánování úloh. Mezi výchozí pro základní operační systémy patří Task Scheduler pro Windows, KRON pro Linux a Automator pro MacOS.

Mailoví klienti hrají klíčovou roli v automatizaci procesů spojených s elektronickou poštou. Pomocí těchto klientů lze nastavit automatická pravidla pro odesílání a přijímání zpráv, což může být využito pro automatizaci různých procesů. Příklady zahrnují nastavení automatických pravidel pro zpracování příchozí pošty nebo automatizaci odesílání zpráv a příloh.

Dalším důležitým aspektem je využití MAPI rozhraní pro integraci skriptů s emailovými klienty, zejména s Microsoft Outlook. Tímto způsobem lze dosáhnout automatizace různých úloh spojených s elektronickou poštou, jako je odesílání a přijímání zpráv, správa kalendáře a správa kontaktů.

Tato kapitola se zaměřuje na zkoumání využití těchto nástrojů a technologií pro automatizaci skriptů v prostředí Windows a dalších operačních systémů, s důrazem na efektivní využití PowerShellu pro řízení a automatizaci různých IT procesů.

4.4.2.1 Task Scheduler

Task Scheduler, jakožto integrovaný nástroj operačního systému Windows, poskytuje uživatelům možnost plánování a automatizace rozmanitých úkolů na jejich počítačích. Bez ohledu na to, zda je cílem provádět konkrétní akce v přesně stanovených časech, či nastavit opakující se úkoly, poskytuje ucelený rámec pro jejich realizaci.

Task scheduler disponuje uživatelsky přívětivým rozhraním, prostřednictvím kterého lze definovat spouštěče, nastavit podmínky a specifikovat akce pro každý jednotlivý úkol.

Princip činnosti Task Scheduler úkolů je založen na hierarchické struktuře, která zahrnuje úkoly, spouštěče a akce. Úkol představuje konkrétní činnost nebo soubor činností, které je možné automatizovat, ať už jde o spuštění programu, provádění údržby systému či generování zpráv.

Task Scheduler určuje čas spuštění úkolu. Spouštění úkolů umožňuje Task Scheduler nastavovat denně, týdně či měsíčně, nebo mohou být vyvolány určitými

akcemi, jako jsou událostmi v systému, spuštění počítače, přihlášení uživatele nebo přítomnost určitého souboru. Je možné konfigurovat akce tak, aby automaticky spouštěly programy, prováděly skripty, odesílaly e-maily, zobrazovaly zprávy nebo prováděly jakoukoli jinou činnost podporovanou systémem Windows 10.

Využitím Task Scheduler nejen lze ušetřit čas, ale také zajistit konzistenci a přesnost prováděných úkolů.[16]

4.4.2.2 Emailový klient

Emailový klient je software, který umožňuje uživatelům odesílat, přijímat a spravovat své elektronické zprávy. Tato aplikace poskytuje uživatelům rozhraní pro práci s jejich emailovými účty a umožňuje jim organizovat svou poštu, vytvářet a odesílat zprávy, spravovat kontakty a kalendáře a provádět další úkoly související s elektronickou poštou.

Existuje mnoho emailových klientů dostupných na trhu, ale několik z nich se těší zvláštní oblibě, které uživatelé instalují na svá zařízení. Mezi populární desktopové klienty patří Microsoft Outlook, Mozilla Thunderbird a Apple Mail, zatímco pro mobilní zařízení jsou dostupné aplikace jako Apple Mail, Gmail, Outlook a Proton Mail, které nabízejí pokročilé funkce pro efektivní správu doručené pošty.

Kromě samostatných aplikací mnoho hlavních poskytovatelů e-mailových služeb také nabízí možnost přístupu ke svým e-mailům prostřednictvím webového rozhraní, čímž se prohlížeč stává jakýmsi klientem e-mailu umožňujícím odesílat, přijímat a organizovat poštu.

Klienti e-mailu využívají různé e-mailové protokoly, jako je SMTP pro odesílání a IMAP/POP3 pro příjem zpráv. Zatímco POP3 je starším protokolem, který zpravidla umožňuje pouze stahování zpráv na jedno zařízení a jejich smazání ze serveru, IMAP umožňuje synchronizaci e-mailů včetně příloh mezi více zařízeními a zpracování dalších funkcí, jako jsou složky a nastavení, na všech připojených zařízeních.

Výhody používání klientů e-mailu zahrnují možnost správy více účtů od různých poskytovatelů e-mailů pomocí jedné aplikace, offline přístup k e-mailům a schopnost zálohování zpráv. To umožňuje uživatelům efektivnější práci s e-maily a zvyšuje jejich produktivitu oproti webovým rozhraním.[17]

4.4.2.3 MAPI

MAPI (Messaging Application Programming Interface) je programovací rozhraní, které umožňuje interakci a správu elektronických zpráv, jako jsou e-maily, kalendářové události a kontakty, v aplikacích a systémech. Je to standardní rozhraní vyvinuté společností Microsoft a používá se zejména v operačních systémech Windows.

MAPI poskytuje soubor funkcí a metod, které umožňují klientům přistupovat k e-mailovým účtům, odesílat a přijímat zprávy, manipulovat s přílohami a spravovat kontakty a kalendáře. Toto rozhraní je široce využíváno v různých typech aplikací, jako jsou e-mailové klienty, kalendářové aplikace, správci úkolů a další.

Jednou z hlavních výhod MAPI je jeho modularita a flexibilita. Díky své architektuře mohou vývojáři vytvářet rozšíření a doplňky pro různé funkce a integrace s dalšími systémy a službami. MAPI také umožňuje centralizovanou správu e-mailových účtů a dat, což zjednodušuje jejich správu a zabezpečení.[18]

Co se týče skriptování v PowerShell, MAPI může být integrováno do skriptů pro automatizaci určitých úloh souvisejících s elektronickými zprávami.

Skriptování v PowerShellu umožňuje efektivní a flexibilní automatizaci procesů souvisejících s MAPI, což může přinést významné úspory času a zdrojů pro organizace a jednotlivé uživatele.

5 Praktická část

V dnešní digitální éře, kde se stále více spoléháme na automatizaci a efektivitu při práci s daty a soubory, je důležité mít k dispozici nástroje, které umožňují bezpečný a spolehlivý přenos dat mezi různými systémy a prostředími. Po důkladné analýze teoretických konceptů a metodiky zabezpečení a automatizace přenosu dat, přichází na řadu praktická aplikace získaných poznatků. Tato část práce se zaměřuje na konkrétní scénáře využití Powershell skriptů pro zabezpečený a efektivní přenos dat.

V této části práce budou popsány step-by-step postupy v rámci konkrétních Use Cases, které demonstrují účinnost a praktickou hodnotu využití Powershell skriptů. Jednotlivé scénáře se zaměří na různé aspekty automatizované archivaci dat, přenosu dat, včetně použití protokolů FTPS a SFTP pro zabezpečený přenos souborů.

První část našeho průzkumu bude věnována detailnímu popisu jednotlivých Use Cases, které pokrývají běžné scénáře v praxi. V tomto kontextu budeme zkoumat, jak efektivně využít PowerShell skripty k automatizaci procesu přenosu souborů pomocí obou zmíněných protokolů.

Plánování úloh pomocí Windows Task Scheduler: Navážeme na předchozí Use Cases a ukážeme, jak lze využít plánovač úloh systému Windows k automatizaci opakovaných úloh spojených s přenosem souborů.

Automatizovaný přenos souborů z emailu: V poslední části se zaměříme na implementaci skriptu, který umožní automatické stahování příloh z příchozích emailů, přejmenování souborů podle určených pravidel a jejich archivaci.

Cílem této práce je poskytnout ucelený průvodce pro praktické využití PowerShellu při práci s přenosem souborů, a to jak v prostředí FTPS a SFTP, tak i při integraci s dalšími nástroji a prostředími, jako je plánovač úloh či emailový klient. Důraz bude kladen na jednoduchost implementace, spolehlivost a zabezpečení přenosu dat, aby čtenáři mohli efektivně využít tyto techniky ve svých vlastních projektech a scénářích.

5.1 Konfigurace uživatele pro spouštění skriptů

Krok 1: Získání aktuálního nastavení spouštění skriptů

Pro získání informací o aktuálním nastavení spouštění skriptů na počítači vykonáme následující příkaz v okně Windows Powershell:

```
Get-ExecutionPolicy -List
```

Tento příkaz poskytne seznam všech dostupných Execution Policy pro všechny uživatele a pro aktuálního uživatele, včetně jejich priorit.

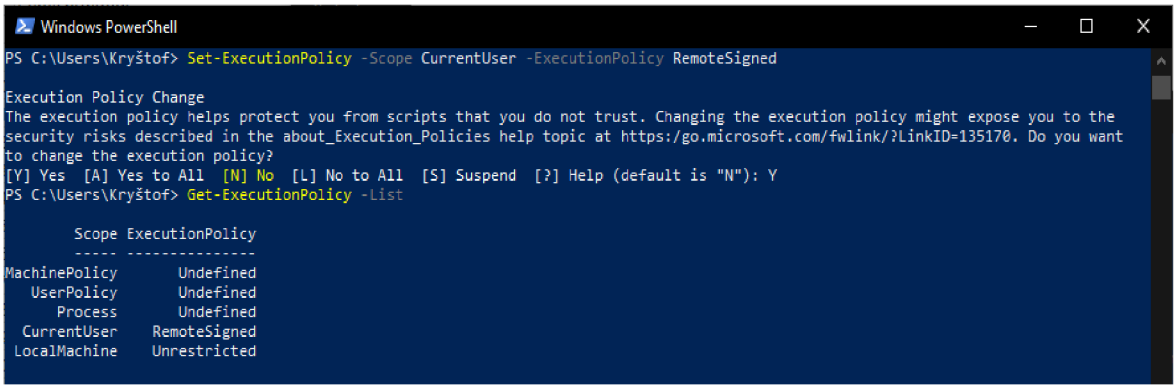
Krok 2: Nastavení Execution Policy a kontrola

Pro změnu nastavení spouštění skriptů použijeme příkaz Set-ExecutionPolicy. Zvolíme jednu z možností politiky spouštění, která nejlépe vyhovuje našim potřebám a bezpečnostním požadavkům.

Například, pro nastavení aktuálního uživatele Execution Policy na RemoteSigned, které umožňuje spouštění podepsaných skriptů stažených z internetu, použijeme následující příkaz:

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

Po provedení tohoto příkazu se zobrazí upozornění o změně nastavení politiky spouštění. Pro potvrzení změny stiskneme klávesu Y a poté stiskneme klávesu Enter.



Pro kontrolu změny Execution Policy je vhodné opět zavolat příkaz Get-ExecutionPolicy jako je tomu v prvním kroku.

Tímto krokem jsme úspěšně nastavili prostředí pro spouštění Windows Powershell skriptů podle vybrané politiky spouštění, a to v souladu s bezpečnostními standardy a požadavky na našem systému.

5.2 Ukázka použití Powershell skriptů a automatizovaného spuštění skriptů pomocí Windows Task Scheduler

5.2.1 UseCase 1 – Základní archivace

Skript, který je v práci uveden v příloze pod názvem „Skript č. 1“, je využitý v různých pracovních prostředích, pro příklad lze uvést:

Správa logů a protokolů:

V IT prostředí je tento skript využitý k archivaci logů a protokolů serverů, aplikací nebo zařízení. Soubory s logy jsou často generovány s časovými razítky, a jsou tak snadno archivovány do strukturovaných adresářů podle data a času, což usnadňuje sledování a analýzu historických dat.

Zálohování důležitých dokumentů:

V kancelářském prostředí je skript využíván k zálohování důležitých dokumentů nebo reportů do archivního úložiště. Soubory jsou automaticky archivovány a přejmenovány podle času jejich poslední úpravy, což usnadňuje jejich správu a obnovení v případě potřeby.

Sledování změn a revizí:

Vývojáři softwaru tento skript využívají k archivaci verzí kódu nebo konfiguračních souborů a sledování jejich změn v průběhu času. Soubory jsou automaticky archivovány do podadresářů podle data a času poslední modifikace, což umožňuje snadné porovnání a obnovení předchozích verzí.

Celkově tento skript přináší efektivní řešení pro archivaci a správu souborů s ohledem na časová razítka jejich poslední modifikace, což je užitečné v různých oblastech, kde je důležité udržovat přehlednost a historii změn dokumentů.

Tento skript provádí automatizovanou archivaci souborů s určitou příponou, v tomto případě "*.pdf", z určeného pracovního adresáře do archivního adresáře. Zde je krok za krokem, co tento skript dělá:

Krok 1 skriptu: Nastavení proměnných

Proměnná $workFolder určuje pracovní adresář, ze kterého budou archivovány soubory.

Proměnná $reportsToArchive určuje filtr souborů, které mají být archivovány. V tomto případě všechny soubory s příponou ".pdf".
Proměnná $archiveFolder určuje adresář, kam budou archivovány soubory.
Proměnná $archiveNamePattern určuje vzor názvu archivního souboru, který obsahuje časové údaje a název původního souboru.

Krok 2 skriptu: Procházení souborů v pracovním adresáři

Pomocí cyklu foreach skript prochází všechny soubory ve specifikovaném pracovním adresáři, které odpovídají filtru $reportsToArchive.

Krok 3 skriptu: Vytvoření cílového názvu a adresáře pro archivaci

Pro každý nalezený soubor je vytvořen název archivního souboru pomocí vzoru $archiveNamePattern, který obsahuje název původního souboru a časové údaje.

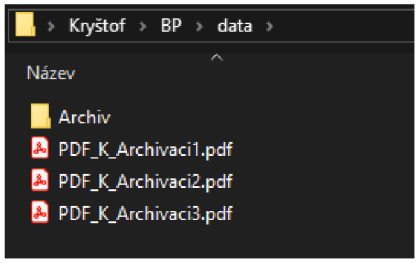
Krok 4 skriptu: Archivace souborů

Nakonec je každý nalezený soubor přesunut do odpovídajícího archivního adresáře pod novým názvem pomocí funkce Move-Item. Současně funkce Write-host vypisuje každý soubor, který byl určen k archivaci, do konzole.

Tento skript tedy zajišťuje automatizovanou archivaci souborů s určitou příponou do specifikovaného adresáře s možností vytvoření hierarchické struktury adresářů podle časových údajů o poslední modifikaci.

Kroky z pohledu uživatele:

V adresáři „data“ jsou nachystané soubory PDF k archivaci do složky Archiv.



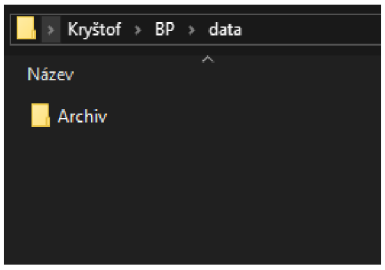
5-1 Skript č. 1 – Adresář „data“ před spuštěním
Zdroj: Vlastní

Spuštěním skriptu přes prostředí Windows Powershell se provedou jednotlivé kroky a do konzole jsou vypsány veškeré soubory, které byly archivovány.

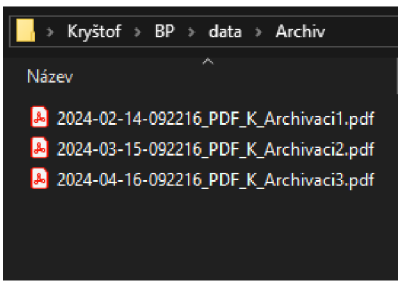
```
PS C:\Users\Kryštof\BP\skripty> .\Skript_c_1.ps1
PDF_K_Archivaci1.pdf archivován.
PDF_K_Archivaci2.pdf archivován.
PDF_K_Archivaci3.pdf archivován.
```

5-2 Skript č. 1 – Spuštění skriptu ve Windows Powershell
Zdroj: Vlastní

Výsledkem tohoto jsou soubory přesunuty ze složky „data“ do podsložky Archiv, přejmenovány podle formátu definovaném ve skriptu se zahrnutím data poslední změny do názvu souboru.



5-3 Skript č. 1 – Adresář „data“ po spuštění
Zdroj: Vlastní



5-4 Skript č. 1 – Archivované soubory v podadresáři „Archiv“
Zdroj: Vlastní

5.2.2 UseCase 2 – Archivace do podadresáře dle roku a měsíce

Tento skript, v příloze v sekci „Skript č. 2“, je využitý pro automatizaci archivace dokumentů nebo reportů v pracovním prostředí, kde je důležité udržovat přehlednost a organizovanost dokumentace. Několik reálných příkladů použití zahrnuje:

Archivace finančních zpráv:

Ve finančním prostředí je tento skript určený k archivaci měsíčních finančních zpráv ve formátu PDF do strukturovaných podadresářů podle roku a měsíce. To umožní snadný přístup k historii finančních dokumentů a usnadní správu účetnictví.

Správa právní dokumentace:

U právnické firmy je tento skript využitý k automatické archivaci právních dokumentů, smluv a dalších právních dokumentů do organizační struktury podle data vytvoření. To pomáhá udržovat přehled o právní dokumentaci a usnadňuje vyhledávání a přístup k dokumentům v případě potřeby.

Archivace obchodních reportů: Firmám zabývajícím se obchodem může tento skript pomoci při archivaci a organizaci obchodních reportů, analýz a prezentací. Soubory by mohly být automaticky archivovány do strukturovaných adresářů podle roku a měsíce, což usnadňuje sledování a vyhledávání historických obchodních dat.

Celkově tento skript přispívá k efektivní správě a organizaci dokumentů v rámci různých profesních prostředí, což zvyšuje produktivitu a umožňuje snadný přístup k důležitým informacím.

Tento skript provádí archivaci souborů s určitou příponou, v tomto případě opět "*.pdf", z pracovního adresáře do archivního adresáře podle data poslední modifikace souborů. Na rozdíl od prvního skriptu, který používá pouze jednoho archivního adresáře, tento skript ukládá soubory do podadresářů podle roku a měsíce, což umožňuje lepší organizaci a snadnější vyhledávání souborů v archivu.

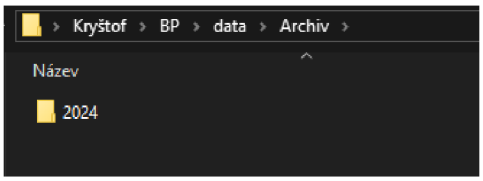
Kroky z pohledu uživatele:

Oproti prvnímu skriptu není v počáteční fázi žádný rozdíl, opět v adresáři „data“ nalezneme soubory PDF připravené k archivaci. Při spuštění skriptu pomocí Windows Powershell nám funkce Write-Host opět vypisuje soubory, se kterými skript pracoval a zarhivoval. V tomto případě je skript komplexnější. Před manipulací se soubory je testováno, jestli cesta do určeného adresáře v proměnných existuje. Pokud ještě adresář „Archiv“ na určené cestě proměnnou $workFolder neexistuje, skript vytvoří adresář pro archivní soubory.

```
PS C:\Users\Kryštof\BP\skripty> .\Skript_c_2.ps1
PDF_K_Archivaci1.pdf archivován.
PDF_K_Archivaci2.pdf archivován.
PDF_K_Archivaci3.pdf archivován.
```

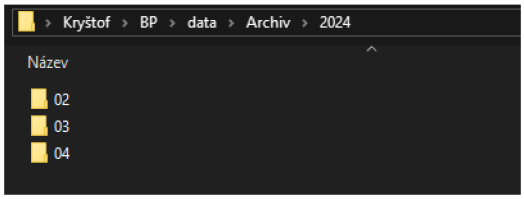
5-5 Skript č. 2 – Spuštění skriptu ve Windows Powershell
Zdroj: Vlastní

Na základě formátování a systematické strukturalizaci je adresář rozdělen podle do roku, ve kterém byly soubory naposledy upraveny. Soubory PDF, určené k archivaci byly naposledy upraveny v roce 2024. Proto vznikl pouze adresář 2024.



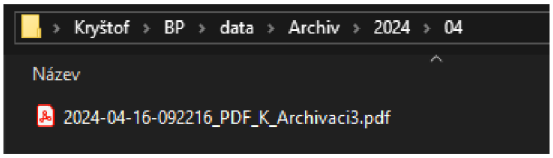
5-6 Skript č. 2 – Adresářová struktura podle roku
Zdroj: Vlastní

V další úrovni adresáře vznikly tři složky na základě měsíce, ve kterém byly PDF Soubory naposledy úpraveny. Každý soubor byl upravený naposledy v jiném měsíci. Konkrétně v únoru(viz. Složka 02), březnu a dubnu.



5-7 Skript č. 2 – Adresářová struktura podle měsíců v daném roce
Zdroj: Vlastní

V jednotlivých složkách, vytvořených na základě měsíce, došlo rozřazení souborů PDF a současném systematickém doplnění názvu souboru o časové údaje.



5-8 Skript č. 2 – Příklad archivovaného souboru
Zdroj: Vlastní

5.2.3 UseCase 3 – Stažení přílohy z Inboxu Outlooku a archivace do podsložek dle roku a měsíce

Use Case 3 pracuje se skriptem pod názvem „Skript č. 3“, který je užíván pro správu dokumentů a reportů v pracovním prostředí, kde je důležité udržovat přehlednost a organizovanost dat z různých zdrojů. Několik příkladů reálného použití zahrnuje:

Správa firemní korespondence:

Skript je využit pro automatické stahování a archivaci důležitých dokumentů a reportů přijímaných emailem. Například archivace faktur, smluv nebo pracovních zpráv do strukturovaných podsložek podle data vytvoření.

Sledování a archivace datových zpráv:

V IT prostředí je tento skript využit pro sledování a archivaci datových zpráv, logů nebo reportů generovaných různými systémy a aplikacemi. Soubory by mohly být automaticky stahovány z emailů a archivovány do podsložek podle data, což usnadňuje správu a analýzu dat.

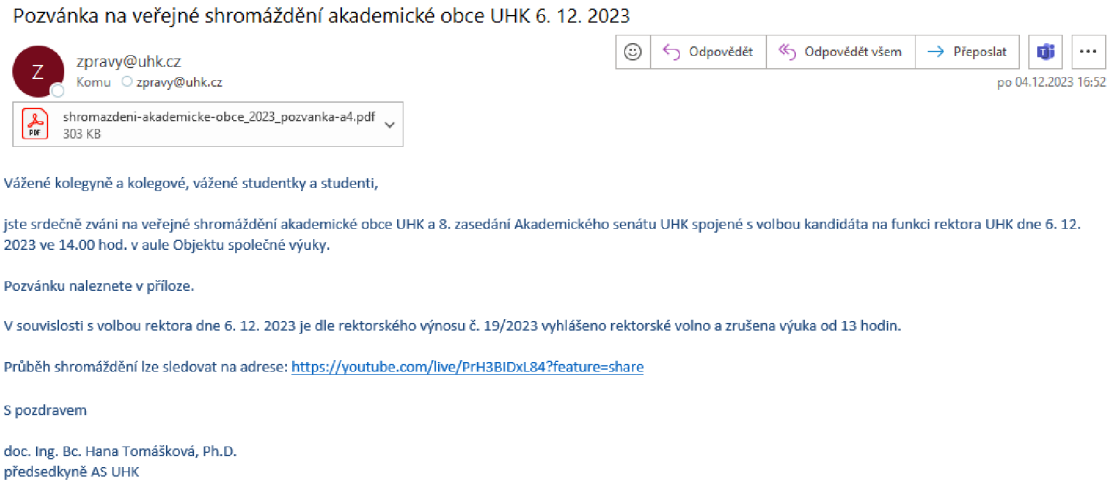
Správa projektové dokumentace:

Firmám provádějícím projektové řízení tento skript pomáhá při správě projektové dokumentace, včetně technických zpráv, plánů projektu nebo výsledků testů. Dokumenty by mohly být automaticky stahovány z emailů týkajících se projektu a archivovány do vhodných podsložek podle data vytvoření.

Celkově tento skript přináší efektivní řešení pro správu a archivaci dokumentů z různých zdrojů, což usnadňuje jejich organizaci, vyhledávání a přístupnost v pracovním prostředí.

V porovnání s předchozími skripty archivace, tento skript umožňuje stahování příloh z emailů, konkrétně využíva MAPI pro možnost vykonávání příkazů nad primární sloužkou Index v aplikaci Microsoft Outlook, což je přidaná funkcionalita oproti prvnímu skriptu, který se zaměřuje pouze na archivaci lokálních souborů. Zároveň archivuje soubory do podsložek podle data poslední modifikace, což umožňuje lepší organizaci a snadnější vyhledávání archivovaných souborů ve srovnání s prvním skriptem, který archivuje soubory do jednoho

adresáře bez hierarchie. V závěru skript kombinuje archivaci lokálních souborů s archivací příloh z emailů, což poskytuje komplexnější řešení pro správu a organizaci dokumentů.



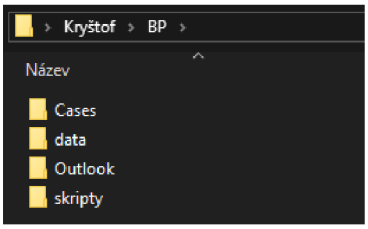
5-9 Skript č. 3 – Příkladný email s přílohou v Outlook
Zdroj: Vlastní

V prvním kroku je nastíněn jeden z mnoha emailů ve složce Index v aplikaci Microsoft Outlook, u kterého je součástí i příloha .pdf.

```
PS C:\Users\Kryštof\BP\skripty> .\Skript_c_3.ps1
```

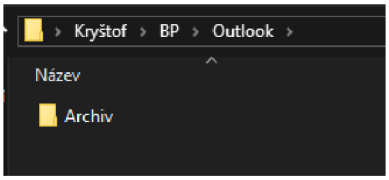
5-10 Skript č. 3 – Spuštění skriptu

V dalším kroku proběhlo spuštění Skriptu č. 3 pomocí Windows Powershell.



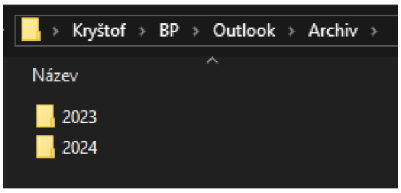
5-11 Skript č. 3 – Adresářová struktura se složkou Outlook
Zdroj: Vlastní

Výstupy, které jsou čerpány z aplikace Microsoft Outlook, jsou odkazovány do adresáře Outlook. Tento krok je opatřen otestováním, jestli cesta do adresáře existuje. V tomto případě cesta neexistovala a skript složku Outlook vytvořil.

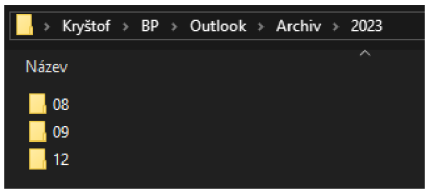


5-12 Skript č. 3 – Adresář „Archiv“ pro uschování PDF příloh
Zdroj: Vlastní

Archivace příloh z emailu je směrována do podadresáře Archiv, kde jsou přílohy systematicky rozřazeny podle data obdržení emailové přílohy. Adresář je hierarchicky rozdělen podle let a následně měsíců.

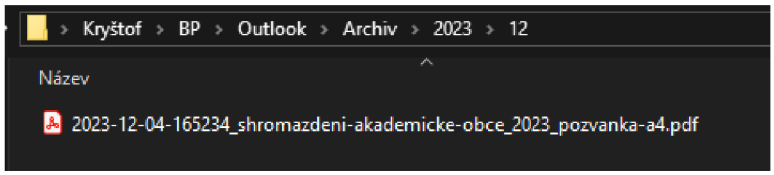


5-13 Skript č. 3 - Adresářová struktura rozdělená podle roku
Zdroj: Vlastní



5-14 Skript č. 3 – Adresářová struktura rozdělená podle měsíce v daném roce
Zdroj: Vlastní

Na posledním obrázku je zobrazen soubor z původního emailu a zároveň formát, ve kterém archivace upravuje soubory pro přejmenovávání souborů



5-15 Skript č. 3 – Archivovaná příloha z původního mailu
Zdroj: Vlastní

5.2.4 UseCase 4 - Stažení přílohy z FTP a archivace do podsložek dle roku a měsíce

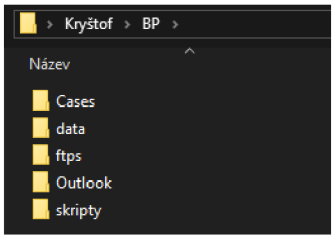
Tento skript umožňuje stahování souborů z FTP serveru a jejich archivaci, zatímco první skript se zaměřuje pouze na archivaci lokálních souborů.

Skript je modulární, s oddělenými funkcemi pro stahování a získávání informací o souborech, což zlepšuje jeho čitelnost a znovu použitelnost.
funkci Get-ReportsListFromFTP může skript pracovat s hierarchií adresářů na FTP serveru a provádět rekurzivní stahování souborů.

```
PS C:\Users\Kryštof\BP\skripty> .\Skript_c_4.ps1
Downloading 'ftp://test.rebex.net/pub/example/imap-console-client.png' as 'C:\Users\Kryštof\BP\ftps\Archiv\../pub\example\2007\02\2007-02-16-180028_imap-console-client.png'...
Downloading 'ftp://test.rebex.net/pub/example/KeyGenerator.png' as 'C:\Users\Kryštof\BP\ftps\Archiv\../pub\example\2007\03\2007-03-19-205258_KeyGenerator.png'...
Downloading 'ftp://test.rebex.net/pub/example/KeyGeneratorSmall.png' as 'C:\Users\Kryštof\BP\ftps\Archiv\../pub\example\2007\03\2007-03-19-205258_KeyGeneratorSmall.png'...
Downloading 'ftp://test.rebex.net/pub/example/mail-editor.png' as 'C:\Users\Kryštof\BP\ftps\Archiv\../pub\example\2007\02\2007-02-16-180027_mail-editor.png'...
Downloading 'ftp://test.rebex.net/pub/example/mail-send-winforms.png' as 'C:\Users\Kryštof\BP\ftps\Archiv\../pub\example\2007\02\2007-02-16-180028_mail-send-winforms.png'...
Downloading 'ftp://test.rebex.net/pub/example/mime-explorer.png' as 'C:\Users\Kryštof\BP\ftps\Archiv\../pub\example\2007\02\2007-02-16-180027_mime-explorer.png'...
Downloading 'ftp://test.rebex.net/pub/example/pocketftp.png' as 'C:\Users\Kryštof\BP\ftps\Archiv\../pub\example\2007\03\2007-03-19-205257_pocketftp.png'...
Downloading 'ftp://test.rebex.net/pub/example/pocketftpSmall.png' as 'C:\Users\Kryštof\BP\ftps\Archiv\../pub\example\2007\03\2007-03-19-205257_pocketftpSmall.png'...
Downloading 'ftp://test.rebex.net/pub/example/pop3-browser.png' as 'C:\Users\Kryštof\BP\ftps\Archiv\../pub\example\2007\02\2007-02-16-180027_pop3-browser.png'...
Downloading 'ftp://test.rebex.net/pub/example/pop3-console-client.png' as 'C:\Users\Kryštof\BP\ftps\Archiv\../pub\example\2007\02\2007-02-16-180027_pop3-console-client.png'...
Downloading 'ftp://test.rebex.net/pub/example/ResumableTransfer.png' as 'C:\Users\Kryštof\BP\ftps\Archiv\../pub\example\2007\03\2007-03-19-205300_ResumableTransfer.png'...
Downloading 'ftp://test.rebex.net/pub/example/winceclient.png' as 'C:\Users\Kryštof\BP\ftps\Archiv\../pub\example\2007\03\2007-03-19-205300_winceclient.png'...
Downloading 'ftp://test.rebex.net/pub/example/winceclientSmall.png' as 'C:\Users\Kryštof\BP\ftps\Archiv\../pub\example\2007\03\2007-03-19-205301_winceclientSmall.png'...
Downloading 'ftp://test.rebex.net/pub/example/WinFormClient.png' as 'C:\Users\Kryštof\BP\ftps\Archiv\../pub\example\2007\03\2007-03-19-205259_WinFormClient.png'...
Downloading 'ftp://test.rebex.net/pub/example/WinFormClientSmall.png' as 'C:\Users\Kryštof\BP\ftps\Archiv\../pub\example\2007\03\2007-03-19-205259_WinFormClientSmall.png'...
```

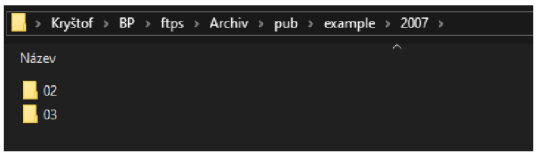
5-16 Skript č. 4 – Spuštění skriptu
Zdroj: Vlastní

V prvním kroku dochází ke spuštění skriptu, který se aktivuje prostřednictvím specifikovaného spouštěcího mechanismu. Jakmile skript zahájí svůj běh, prvním úkolem je navázání spojení s FTP serverem a ověření identity uživatele pomocí poskytnutých přihlašovacích údajů. Tato část skriptu je realizována prostřednictvím funkce Get-ResponseFTP, která umožňuje vytvořit spojení s FTP serverem za použití poskytnutých údajů. Následně se využívají dalších funkcí skriptu, jako je Get-ReportsListFromFTP a Get-FileInfoFTP, jejichž účelem je získání seznamu souborů ze serveru FTP a získání důležitých informací o těchto souborech. Funkce Get-ReportsListFromFTP slouží k získání seznamu souborů z FTP serveru, zatímco funkce Get-FileInfoFTP umožňuje získat informace o každém jednotlivém souboru, jako je například jeho velikost, datum poslední modifikace a další metadata.



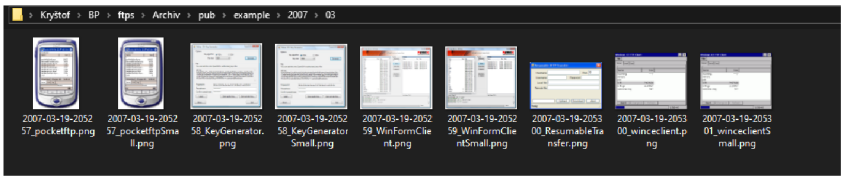
5-17 Skript č. 4 – Automatizovaná deklarace adresáře ftps
Zdroj: Vlastní

Poté, co jsou tyto informace získány, skript pokračuje v procesu vytváření adresářů pro ukládání stažených souborů z FTP serveru do lokálního adresáře ftps. Vytvoření těchto adresářů je realizováno skrze funkci, která vytváří adresáře v pracovním adresáři určeném pro soubory stažené pomocí FTPS.



5-18 Skript č. 4 – Adresář hierarchicky členěný podle roku a následně měsíce
Zdroj: Vlastní

Stažené soubory jsou opět rozřazené podle roku a měsíce, kdy byly naposledy upraveny. Tento krok je zásadní pro organizaci a strukturování dat, která budou následně stažena ze serveru FTP. Díky tomu je zajištěno, že stažené soubory budou správně uloženy do příslušných adresářů, což usnadňuje jejich další zpracování a správu.



5-19 Skript č. 4 – Stažení PNG souborů
Zdroj: Vlastní

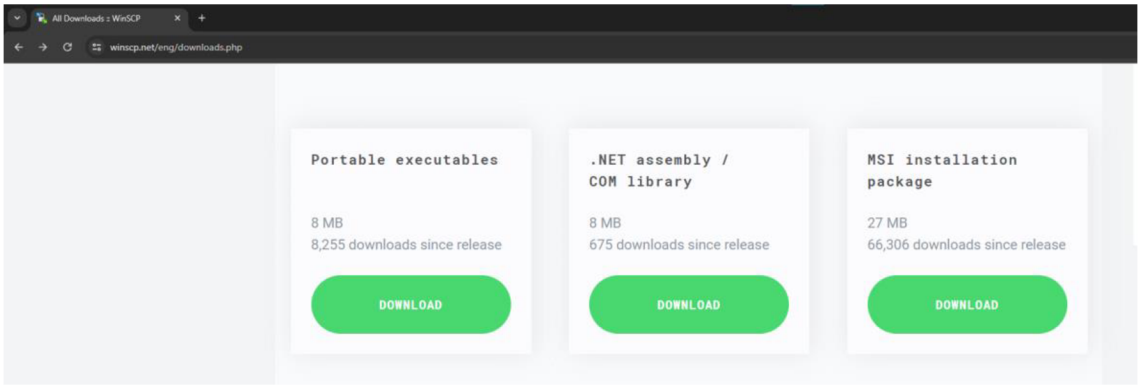
5.2.5 UseCase 6 – Stažení přílohy ze SFTP serveru, archivace do podsložek podle roku a měsíce a zaslání emailu na SMTP server

Tento UseCase představuje využití Powershell skriptů k automatizovanému stažení souborů ze SFTP serveru, jejich archivaci do podsložek strukturovaných hierarchicky podle roku a měsíce a následnému odeslání informačního emailu na SMTP server, který informuje o úspěšném stažení souborů a případně jejich specifikaci.

Možnosti využití takového případu v reálném světě jsou rozmanité. Tento postup může být aplikován v podnicích pro pravidelné zálohování důležitých dat ze vzdálených serverů, v oblasti e-commerce pro automatické stahování objednávek nebo aktualizací produktů a v mnoha dalších oblastech, kde je potřeba pravidelně přenášet data z jednoho místa na druhé.

V tomto konkrétním scénáři jsou využity dva Powershell skripty, z nichž jeden slouží jako knihovna funkcí, zatímco druhý skript tyto funkce volá společně s nezbytnými proměnnými. Tento přístup byl zvolen s ohledem na udržení čistoty a

strukturovanosti samotného skriptu, kdy separace funkcí do knihovny umožňuje snazší správu a údržbu kódu.

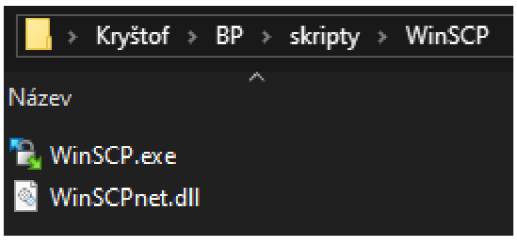


5-20 Skript č. 5 – Stažení .NET assembly WinSCP balíčku z https://winscp.net/eng/downloads.php
Zdroj : Vlastní

Prvním krokem před používáním Powershell skriptu s názvem Skript č. 5. je stažení balíčku WinSCP z oficiálního odkazu. Po stažení je nutné zvolit instalační balíček ".NET assembly / COM library".

Name	Size	Packed	Type	Modified	CRC32
..			Složka souborů		
netstandard2.0			Složka souborů	16.04.2024 15:56	
net40			Složka souborů	16.04.2024 15:56	
WinSCPnet.dll	167 680	66 388	Rozšíření aplikace	16.04.2024 15:52	429E5F5D
WinSCP.exe	22 999 584	8 353 138	Aplikace	16.04.2024 15:51	BF889DF7
readme_automa...	562	321	Textový dokument	16.04.2024 11:09	F7F9BDDB
license-winscp.txt	37 852	12 429	Textový dokument	16.04.2024 11:09	0C86240E
license-dotnet.txt	17 103	5 190	Textový dokument	16.04.2024 11:09	E417C444

5-21 Extrakce WinSCP
Zdroj: Vlastní



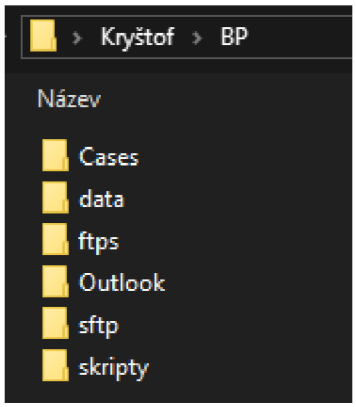
5-22 Adresa WinSCP
Zdroj: Vlastní

Následně je nutné tento balíček extrahovat do adresáře WinSCP, který se nachází se nachází na stejné adresářové úrovni jako skript ke spuštění. Samotné soubory .exe a .dll je nutné uložit do vytvořené WinSCP složky.

```
PS C:\Users\Kryštof\BP\skripty> C:\Users\Kryštof\BP\skripty\Skript_c_5.ps1
Stahuji '/pub/example/readme.txt' jako 'C:\Users\Kryštof\BP\sftp\Archiv\pub\example\2023\09\2023-09-19-131203_readme.txt'...
Stahuji '/readme.txt' jako 'C:\Users\Kryštof\BP\sftp\Archiv\2023\09\2023-09-19-131203_readme.txt'...
```

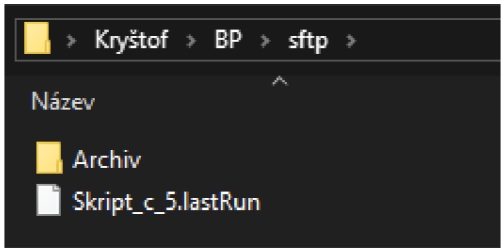
5-23 Skript č. 5 – Spuštění skriptu
Zdroj: Vlastní

Samotný skript je spuštěn pomocí Windows Powershell a jeho hlavním úkolem je stahování souborů ze SFTP serveru a jejich archivace.

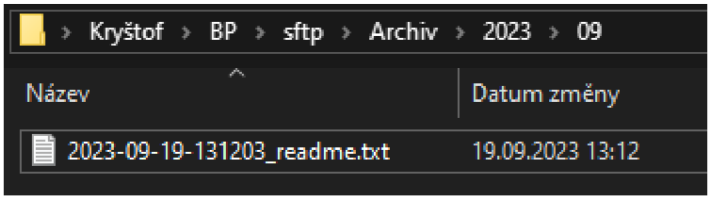


5-24 Skript č. 5 – Adresář s nově vzniklou složkou sftp

V rámci připojení a stahování souborů ze serveru SFTP je skriptem ověřená existence adresáře sftp. Pokud adresář neexistuje, skript jej sám vytvoří.

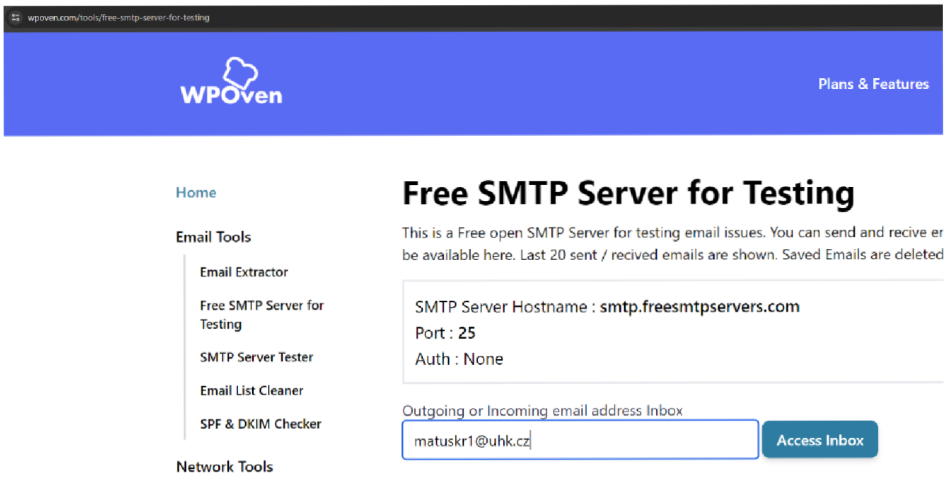


5-25 Skript č. 5 – Adresář sftp
Zdroj: Vlastní



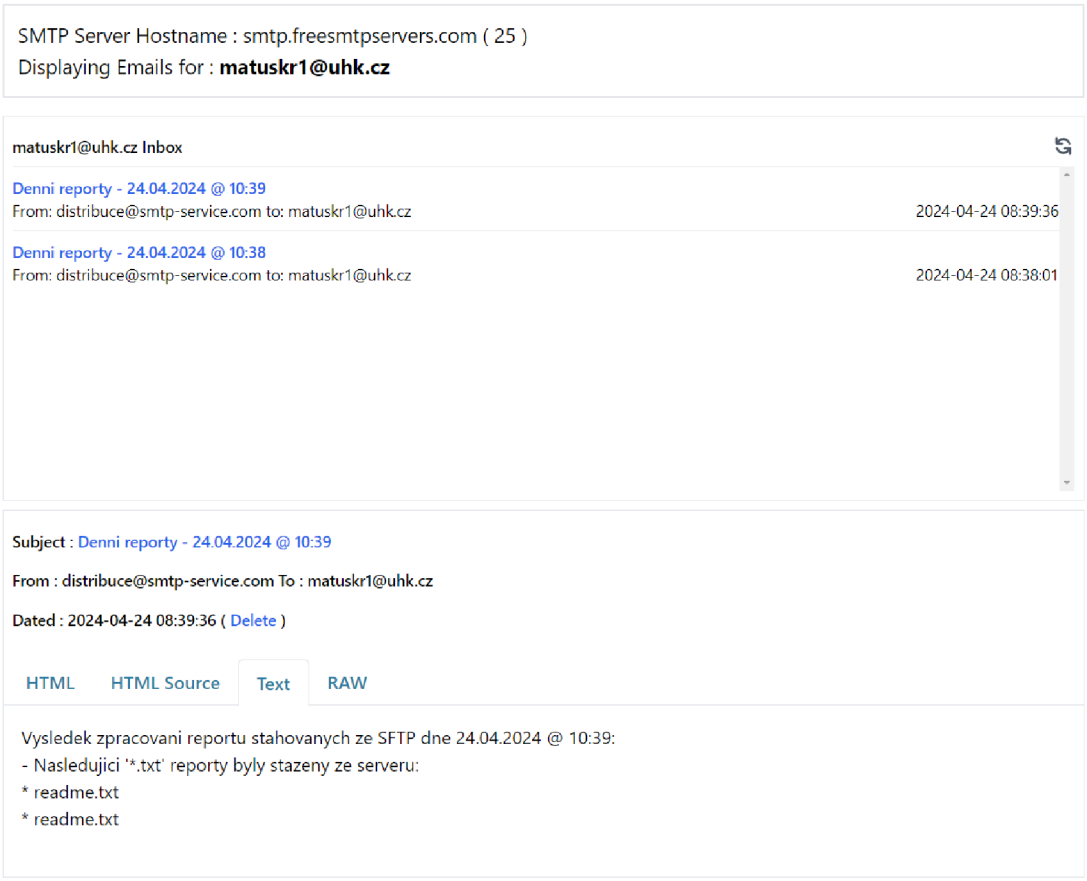
5-26 Skript č. 5 – Adresář sftp
Zdroj: Vlastní

Základ archivace zůstává principiálně stejná a na základě souborů je vytvořený systematický strom adresářů podle data změny.



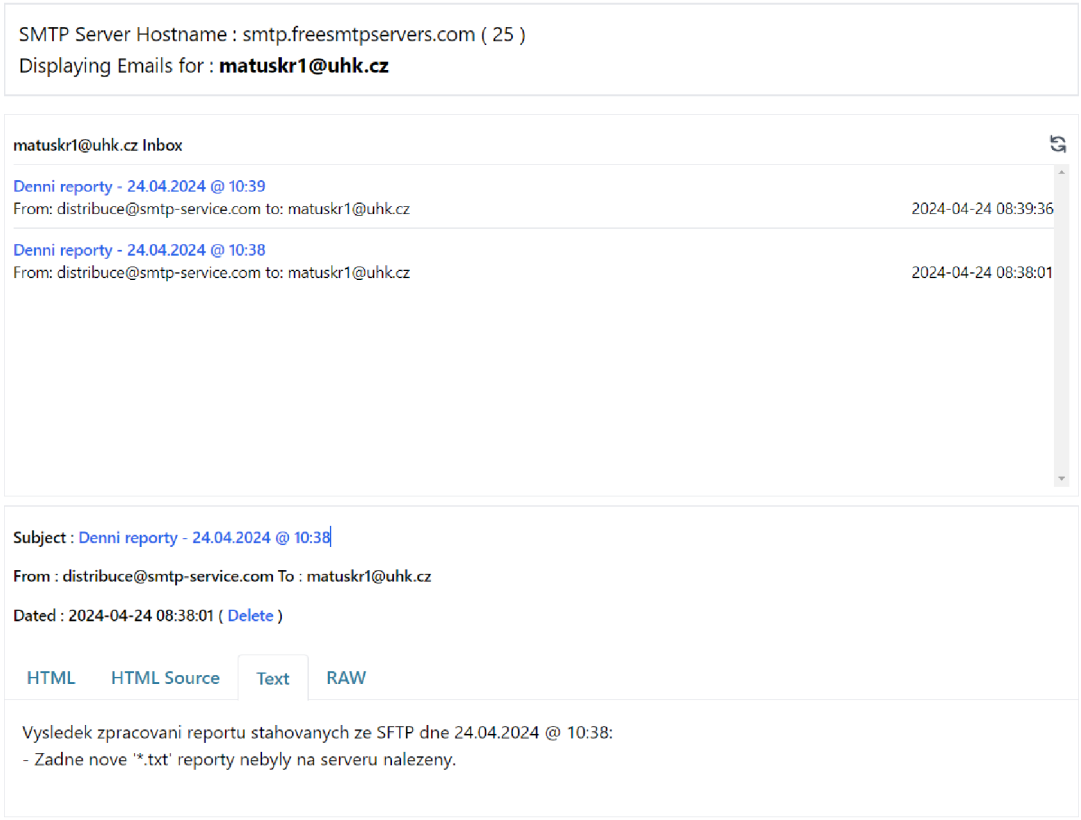
5-27 – SMTP server
Zdroj: Vlastní

Další funkcionalitou skriptu je odeslání informačního emailu obsahujícího seznam stažených souborů na SMTP server. Po zadání adresy, bez ohledu na to, jestli se jedná o adresu odesílatele nebo příjemce, se zobrazí fronta emailových souborů čekající na stažení ze SMTP serveru.



5-28 SMTP server – Formát emailu při nalezení nových souborů
Zdroj: Vlastní

Další funkcionalitou skriptu je odeslání informačního emailu obsahujícího seznam stažených souborů. Pro porovnání je ukázáno, jak email vypadá na serveru SMTP v případě úspěšného stažení souborů a jaké informace obsahuje. V obsahu emailu je informace o výsledku zpracování reportu stahovaných ze SFTP, v jakém čase a jaké soubory byly nebo nebyly na serveru SFTP nalezeny.



5-29 SMTP server – Formát emailu při nenalezení nových souborů
Zdroj: Vlastní

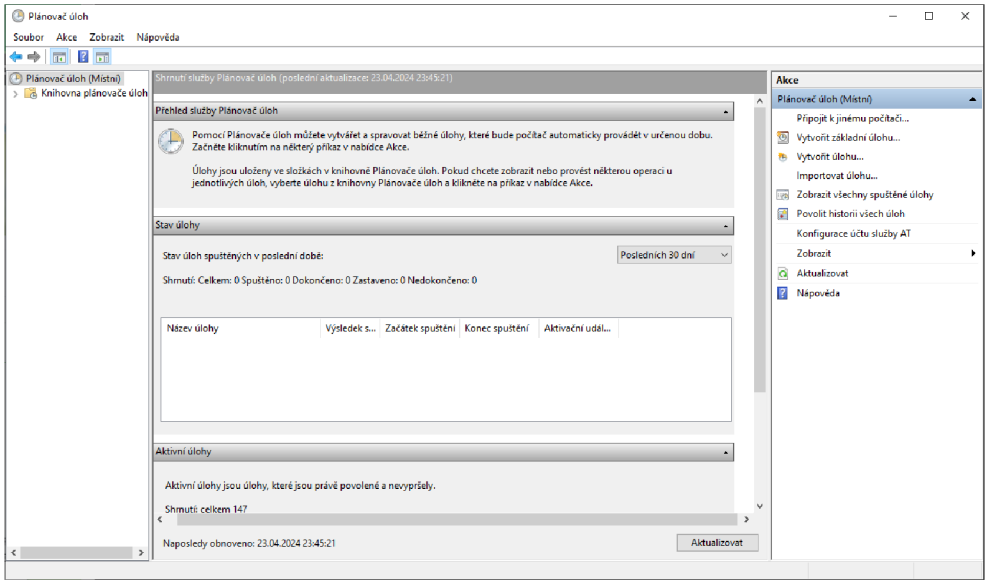
V případě, že na serveru nebyly nalezeny žádné nové soubory ke stažení, je také ukázáno, jakým způsobem je zaslán email s touto informací.

Tímto způsobem je zajištěno pravidelné a automatické stahování a archivace souborů ze SFTP serveru s následným informováním uživatele o průběhu tohoto procesu.

5.2.6 UseCase 7 – Vužití Windows Task Scheduler pro spouštění Powershell skriptů v naplánovaném čase

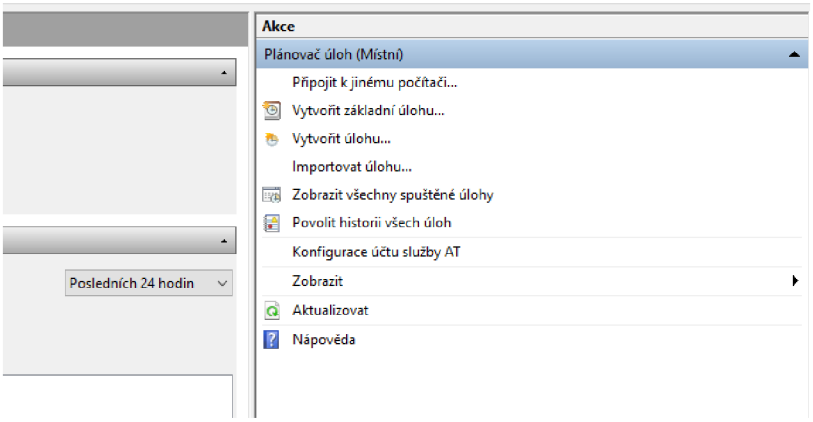
V tomto příkladu je návod zaměřen na podrobný postup nastavení automatizovaného spouštění Powershell skriptů pomocí Windows Task Scheduleru. Tento návod poskytuje uživatelům systematický přehled procesu nastavení spouštění skriptů v souladu s přesně stanoveným časem a četností opakování. Zahrnuje také detailní instrukce k individuálním konfiguracím, které umožňují uživatelům přizpůsobit spouštění skriptů jejich specifickým potřebám a

požadavkům. Tímto se poskytuje komplexní a ucelený návod, který umožňuje efektivní využití možností, které Windows Task Scheduler nabízí pro automatizaci úloh pomocí Powershell skriptů.



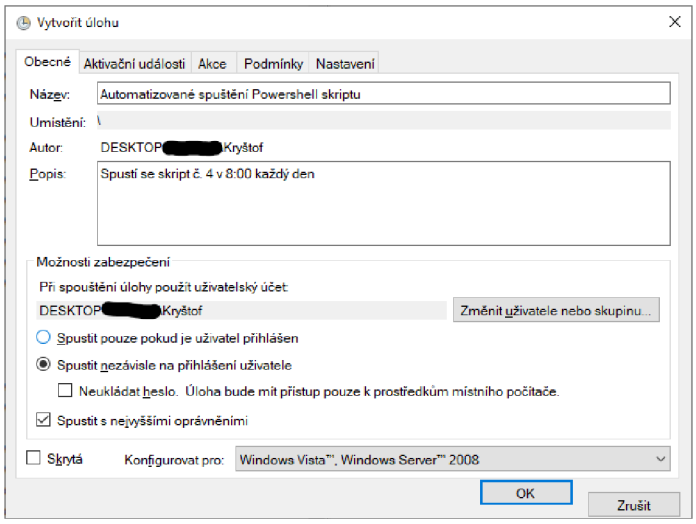
5-30 TaskScheduler – Spuštění aplikace
Zdroj: Vlastní

Nejprve je nutné otevřít aplikaci Task Scheduler, v překladu Plánovač Úloh, což je nativní aplikace součástí operačního systému Windows, umožňující plánování a automatizaci úkolů.



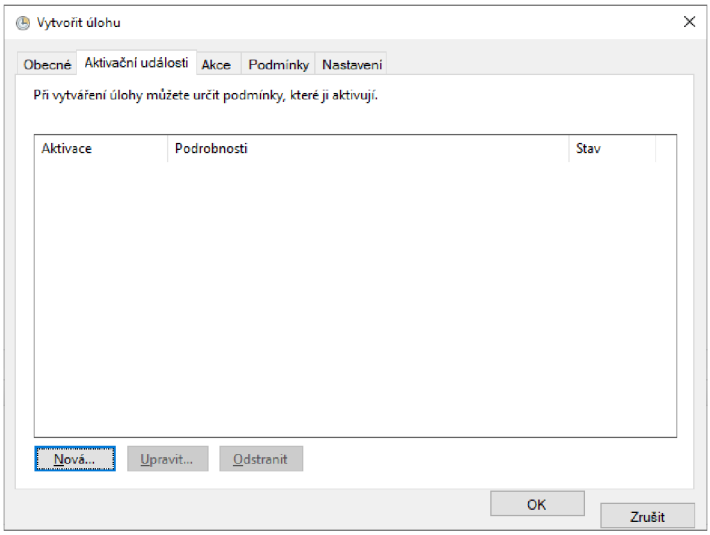
5-31 TaskScheduler – Vytvořit úlohu
Zdroj: Vlastní

Po otevření aplikace je nutné vytvořit novou úlohu, což je základní krok pro začátek procesu nastavení automatického spouštění a z těchto možností je nutné vybrat „Vytvořit úlohu…“.



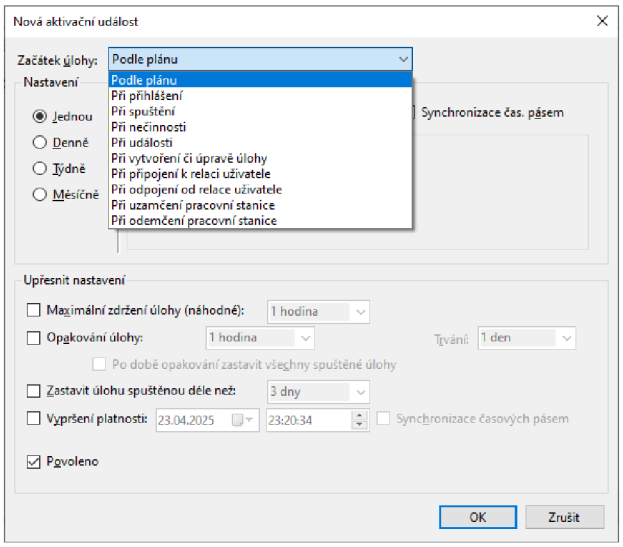
5-32 TaskScheduler – Název a popis úlohy
Zdroj: Vlastní

Při vytváření úlohy je důležité zadat název úlohy a popis, který poskytuje konkrétní informace o účelu úlohy. Dále je možné specifikovat, zda má úloha běžet pouze při přihlášení uživatele, či i nezávisle na přihlášení. Pro efektivní automatizaci je vhodné zaškrtnout volbu "Spustit s nejvyššími oprávněními".



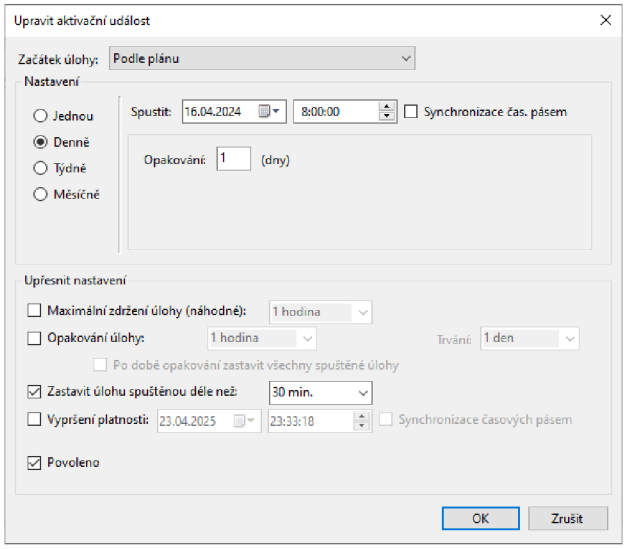
5-33 TaskScheduler – Aktivační událost(Trigger)
Zdroj: Vlastní

Následně je nutné nastavit aktivační událost, známou jako Trigger. Aktivační události určují, kdy má být úloha spuštěna. Je možné definovat více aktivací, a proto je třeba kliknout na tlačítko "Nová".



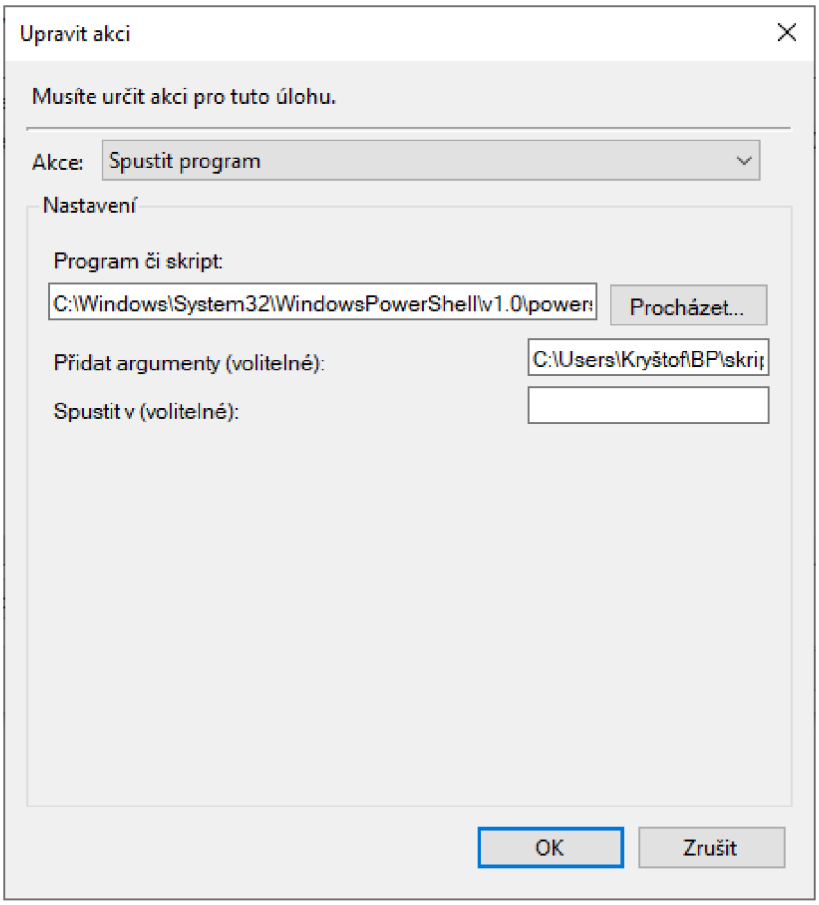
5-34 TaskScheduler – Možnosti aktivační události
Zdroj: Vlastní

Existuje několik způsobů, jak může být aktivační událost spuštěna. Pro případ využití Powershell skriptů je vhodné zvolit spuštění "Podle plánu", což umožňuje pravidelné spouštění úlohy v určeném čase.



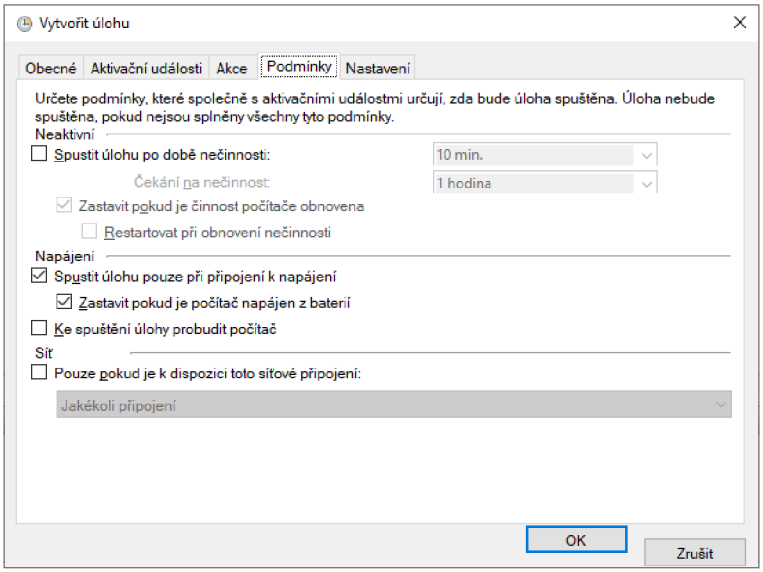
5-35 TaskScheduler – Konfigurace frekvence spouštění
Zdroj: Vlastní

Následujícím krokem je nastavení frekvence spouštění a opatření pro případ, že úloha neskončí v přiměřeném čase. V tomto případě je spouštění nastaveno na každý den od 8:00 a pokud bude úloha spuštěná úloha běžet déle než 30 minut, bude vynuceně zastavena.



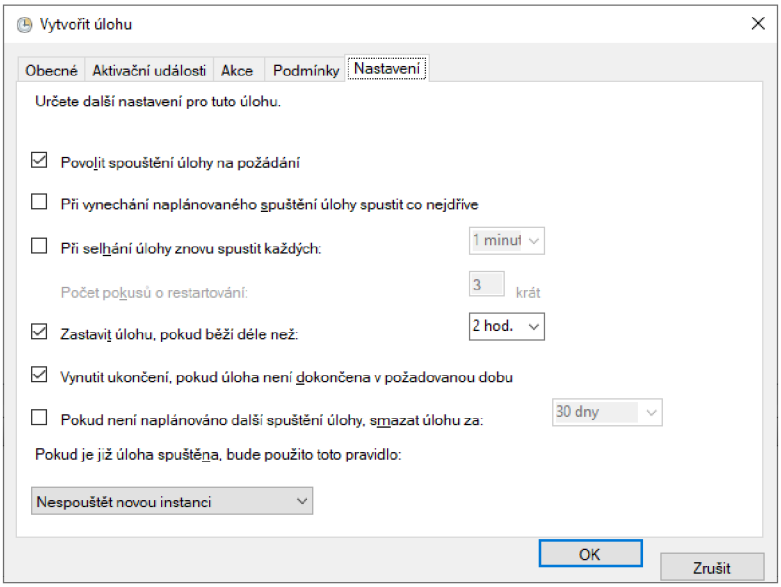
5-36 TaskScheduler – Zahrnutí Powershell.exe a skriptu do úlohy
Zdroj: Vlastní

Důležitým krokem je konfigurace spolupráce s Powershell skripty. Je nutné zadat cestu k Powershell.exe do pole "Program či skript" a cestu k používanému skriptu do pole „Přidat argumenty“. Obvyklou cestou k programu je cesta "C:\Windows\System32\WindowsPowerShell\v1.0". Pokud tomu tak není, je vhodné na tlačítko "Procházet" a najít umístění Powershell.exe.



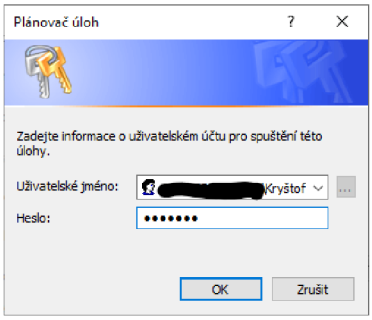
5-37 TaskScheduler – Podmínky pro spuštění
Zdroj: Vlastní

Předposledním krokem v nastavení je konfigurace podmínek, které se týkají neaktivity, napájení a sítě, což je užitečné zejména pro přenosná zařízení.



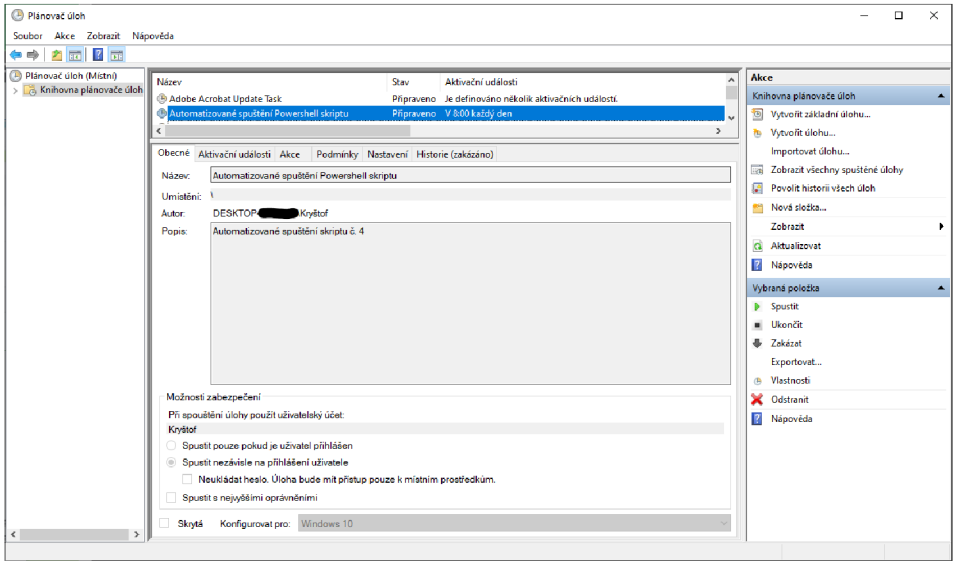
5-38 TaskScheduler – Finální nastavení
Zdroj: Vlastní

Posledním krokem před potvrzením je finální nastavení, kde je třeba zkontrolovat všechna předchozí nastavení.



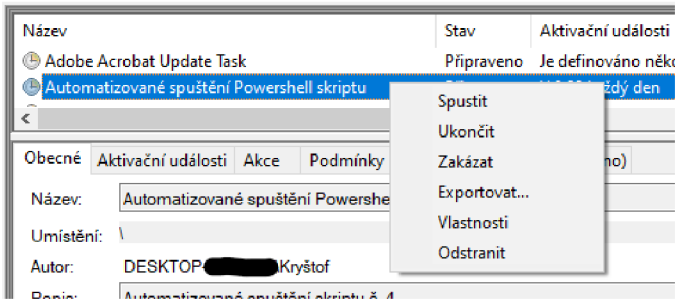
5-39 TaskScheduler – Potvrzení zadáním přihlašovacích údajů
Zdroj: Vlastní

Poté už zbývá jen úlohu uložit potvrzením přihlašovacích údajů do systému.



5-40 TaskScheduler – Zobrazení nově vytvořené úlohy
Zdroj: Vlastní

Vytvořenou úlohu lze dohledat v Knihovně aplikace.



5-41 TaskScheduler -Možnosti úprav vytvořené úlohy
Zdroj: Vlastní

Pro úpravu, spuštění či další události nad úlohou můžeme řídit pravým kliknutím na konkrétní úlohu.

6 Shrnutí výsledků

Při shrnutí této práce je vhodné zrekapitulovat dosažené výsledky, které byly získány prostřednictvím analýzy provedené v teoretické části a ověřeny v praktické aplikaci.

Z poznatků z teoretické sekce lze vyvodit, že intenzivní technologický vývoj v oblasti přenosu souborů a zabezpečení, spolu s rozvojem nástrojů pro automatizaci, výrazně usnadňuje procesy spojené s přenosem dat a manipulací se soubory. Dochází k neustálému posunu vpřed, kdy dynamika přenosu souborů roste a s tím i důraz na bezpečnostní aspekty.

V praktické části byly aplikovány různé typy případů pro efektivní využití Powershell skriptů v kombinaci s protokoly pro přenos souborů a automatizovanou manipulací s nimi. Je však důležité zdůraznit, že ne všechny protokoly uvedené v teoretické části mají v dnešní době praktické využití, zejména protokoly bez zabezpečení. Powershell skripty se vyvíjely, v průběhu praktické části, v souladu s pokročilými programovacími technikami a jejich komplexita rostla v souladu s narůstajícími požadavky.

V institucionálním prostředí, kde je denně spravováno velké množství souborů, se ukazuje využití Powershell skriptů jako zcela opodstatněné. Automatizované spouštění těchto skriptů prostřednictvím nástrojů jako je Windows Task Scheduler může výrazně zefektivnit procesy reportování, minimalizovat rizika lidských chyb při spouštění skriptů a přinést úspory nákladů, zejména v oblasti velkých firem a institucí.

7 Závěry a doporučení

Podle očeávání v úvodu této práce se potvrdilo, že bezpečnost přenosu dat byla již od počátku internetu klíčovým tématem v rámci neustále se rozvíjející digitální společnosti, která čelí narůstajícímu objemu a důležitosti přenášených informací. S tímto nárůstem vzniká také zvýšené riziko jejich zneužití či úniku, což vyžaduje nejen efektivní přenos dat, ale také zajištění jejich důvěrnosti a integrity.

Tato práce se zabývala problematikou přenosu souborů a jejich automatizace s důrazem na zabezpečení důvěrnosti a integrity přenášených dat. Jejím cílem bylo analyzovat existující postupy, protokoly a nástroje a navrhnout efektivní řešení pro praktické využití.

Teoretická část práce představovala základní koncepty nezbytné pro pochopení problematiky. To zahrnovalo historii internetu a jeho vývoj, principy zabezpečení dle CIA triad (Confidentiality, Integrity, Availability) a přehled existujících protokolů pro přenos souborů a emailů a jejich zabezpečení. Dále se práce zabývala vhodnými programovacími jazyky, emailovými klienty a nástroji pro automatizaci, které jsou klíčové pro praktickou část.

Praktická část práce demonstrovala aplikaci teoretických poznatků v konkrétních scénářích, které byly zaměřeny zejména na využití Powershell skriptů a automatizačních nástrojů jako je Task Scheduler a Outlook. Tyto nástroje byly aplikovány v reálných situacích, aby bylo možné demonstrovat jejich efektivitu a praktickou hodnotu pro zabezpečení a automatizaci přenosu dat.

8 Seznam použité literatury

[1] PUŽMANOVÁ, Rita. TCP/IP v kostce. 2. upravené a rozšířené vydání. České Budějovice : Kopp nakladatelství, 2009. ISBN 9978-80-7232-388-3.

[2] PAVLÍČEK, Antonín; GALBA, Alexander a HORA, Michal. Moderní informatika. Druhé, rozšířené vydání. [Praha]: Professional Publishing, 2017. ISBN 978-80-906594-6-9.

[3] POSTEL, Jon; REYNOLDS, J. K. RFC0959: File transfer protocol. 1985. https://datatracker.ietf.org/doc/html/rfc959#autoid-2

[4] IRWIN, Luke. Demystifying the CIA Triad: Why It's Crucial for Cyber Security [online]. 2023 [cit. 2024-04-07]. Dostupné z: https://www.itgovernance.co.uk/blog/what-is-the-cia-triad-and-why-is-it-important

[5] Coretelligent. What is the CIA Triad? Definition & Examples in Cybersecurity [online]. 2022 [cit. 2024-04-07]. Dostupné z: https://coretelligent.com/insights/what-is-the-cia-triad-and-why-does-your-cybersecurity-position-depend-on-it/

[6] Fortinet. CIA Triad [online]. [cit. 2024-04-07]. Dostupné z: https://www.fortinet.com/resources/cyberglossary/cia-triad

[7] KABELOVÁ, Alena a DOSTÁLEK, Libor. Velký průvodce protokoly TCP/IP a systémem DNS. 5., aktualiz. vyd. Brno: Computer Press, 2008. ISBN 978-80-251-2236-5.

[8] VAN GLASS. What Port Does SFTP Use? JSCAPE [online]. 2024 [cit. 2024-04-09]. Dostupné z: https://www.jscape.com/blog/what-port-does-sftp-use

[9] HORAN, Martin. SHARETRU. Explicit FTPS vs. Implicit FTPS: What You Need to Know [online]. 2023, 11.3.2023 [cit. 2024-04-20]. Dostupné z: https://www.sharetru.com/blog/explicit-ftps-vs-implicit-ftps-what-you-need-to-know

[10] LEPILKINA, Diana. Understanding SMTP – The Protocol Behind Email Delivery. Mailtrap [online]. 2024 [cit. 2024-04-09]. Dostupné z: https://mailtrap.io/blog/smtp/

[11] ASHTARI, Hossein. IMAP vs. POP3: 4 Leading Differences You Should Know. SpiceWorks [online]. 2023 [cit. 2024-04-09]. Dostupné z: https://www.spiceworks.com/tech/tech-general/articles/imap-vs-pop3/

[12] ITECH. What are the Most Popular Automation Programming Languages? [online]. 2023, 31.7.2023 [cit. 2024-04-20]. Dostupné z: https://itechindia.co/us/blog/the-most-popular-automation-programming-languages/

[13] Difference between explicit FTP over TLS and implicit FTP over TLS. Online. In: OmniSecu. C2008-2024. Dostupné z: https://www.omnisecu.com/tcpip/difference-between-explicit-ftp-over-tls-and-implicit-ftp-over-tls.php. [cit. 2024-04-20].

[14] BIGELOW, Stephen J., John MOORE a Don JONES. TECHTARGET. What is PowerShell and how to use it: The ultimate tutorial [online]. 2023 [cit. 2024-04-21]. Dostupné z: https://www.techtarget.com/searchwindowsserver/definition/PowerShell

[15] WILSON, Ed. PowerShell. 2. vydání. Přeložil Jakub GONER. Brno: Computer Press, 2022. ISBN 978-80-251-5049-8.

[16] SOGBESAN, Tayo. MAKE USE OF. Mastering the Built-in Task Scheduler in Windows 10: A Step-by-Step Guide [online]. 2023 [cit. 2024-04-20]. Dostupné z: https://www.makeuseof.com/task-scheduler-in-windows/

[17] BONE, Harry. PROTON. What is an email client, and should you use one? [online]. 2023, 16.8.2023 [cit. 2024-04-20]. Dostupné z: https://proton.me/blog/what-is-an-email-client

[18] ASHTARI, Hossein. SPICEWORKS. MAPI vs. SMTP: Top 4 Differences You Should Know [online]. 2023, 10.4.2023 [cit. 2024-04-20]. Dostupné z: https://www.spiceworks.com/tech/tech-general/articles/mapi-vs-smtp/

9 Přílohy

Seznam obrázků

10 Přílohy – Powershell skripty

10.1 Skript č.1

```
# # # # # # # # # # # # # # # # # # # # # # #
#                                             #
# Skript c. 1 - Zakladni archivace            #
#                                             #
# # # # # # # # # # # # # # # # # # # # # # #

$workFolder = "C:\Users\Kryštof\BP\data"
$reportsToArchive = "*.pdf"
$archiveFolder = "$workFolder\Archiv"
$archiveNamePattern = "{1:yyyy-MM-dd-hhmmss_}{0}"

foreach ( $_file in (Get-ChildItem "$workFolder\$reportsToArchive") )
{
    Move-Item -Path $_file.FullName -Destination
$("$archiveFolder\$archiveNamePattern" -f ($_file.Name,
$_file.LastWriteTime))
    Write-Host "$($_file.Name) archivován."
}
```

10.2 Skript č.2

```
# # # # # # # # # # # # # # # # # # # # # # #
#                                             #
# Skript c. 2 - Archivace do podadresare      #
#               dle roku a mesice             #
#                                             #
# # # # # # # # # # # # # # # # # # # # # # #

$workFolder = "C:\Users\Kryštof\BP\data"
$reportsToArchive = "*.pdf"
$archiveFolder = "$workFolder\Archiv"
$archiveNamePattern = "{1:yyyy}\{1:MM}\{1:yyyy-MM-dd-hhmmss_}{0}"

foreach ( $_file in (Get-ChildItem "$workFolder\$reportsToArchive") )
{
    $destinationFullName = $("$archiveFolder\$archiveNamePattern" -f
($_file.Name, $_file.LastWriteTime))
    $destinationFolder = $(Split-Path $destinationFullName -Parent)
    if ( !(Test-Path -Path $destinationFolder) ) {
        New-Item -Path $destinationFolder -ItemType Directory | Out-
Null
    }
    Move-Item -Path $_file.FullName -Destination $destinationFullName
    Write-Host "$($_file.Name) archivován."
}
```

10.3 Skript č.3

```
# # # # # # # # # # # # # # # # # # # # # # #
#                                             #
```

```
# Skript c. 3 - Stazeni prilohy z Inboxu    #
#               a archivace do podslozek    #
#               dle roku a mesice           #
#                                           #
# # # # # # # # # # # # # # # # # # # # # # #

$workFolder = "C:\Users\Kryštof\BP\Outlook"
$reportsToArchive = "*.pdf"
$archiveFolder = "$workFolder\Archiv"
$archiveNamePattern = "{1:yyyy}\{1:MM}\{1:yyyy-MM-dd-HHmmss_}{0}"

$outlook = New-Object -ComObject Outlook.Application
$namespace = $outlook.GetNamespace("MAPI")
$inboxFolder =
$namespace.GetDefaultFolder([Microsoft.Office.Interop.Outlook.OlDefaul
tFolders]::olFolderInbox)

foreach ( $_email in $inboxFolder.Items ) {
    foreach ( $_attachment in $_email.Attachments ) {
        if ( $_attachment.FileName -like $reportsToArchive ) {

$_attachment.SaveAsFile("$workFolder\$($_attachment.FileName)")
        }
    }
}

foreach ( $_file in (Get-ChildItem "$workFolder\$reportsToArchive") )
{
    $destinationFullName = $("$archiveFolder\$archiveNamePattern" -f
($_file.Name, $_file.LastWriteTime))
    $destinationFolder = $(Split-Path $destinationFullName -Parent)
    if ( !(Test-Path -Path $destinationFolder -PathType Container) ) {
        New-Item -Path $destinationFolder -ItemType Directory | Out-
Null
    }
    Move-Item -Path $_file.FullName -Destination $destinationFullName
}
```

10.4 Skript č.4

```
# # # # # # # # # # # # # # # # # # # # # # #
#                                           #
#                                           #
# Skript c. 4 - Stazeni prilohy z FTP       #
#               a archivace do podslozek    #
#               dle roku a mesice           #
#                                           #
# # # # # # # # # # # # # # # # # # # # # # #

$workFolder = "C:\Users\Kryštof\BP\ftps"
$reportsToArchive = "*.png"
$archiveFolder = "$workFolder\Archiv"
$archiveNamePattern = "{2}\{1:yyyy}\{1:MM}\{1:yyyy-MM-dd-HHmmss_}{0}"

$UserName = "demo"
$Password = "password"
$RemoteSite = "ftp://test.rebex.net/"
```

```
Function Get-ResponseFTP
{
    [CmdletBinding()]
    Param
    (
        [Parameter(Mandatory=$true)]
        [ValidateNotNullOrEmpty()]
        [string]$Uri,

        [Parameter(Mandatory=$true)]
        [ValidateNotNullOrEmpty()]
        [string]$Method
    )
    Begin
    {
        # Set VerbosePreference to Continue so that verbose messages
are displayed.
        $VerbosePreference = 'Continue'
    }
    Process
    {
        try {
            $_Session = [System.Net.FtpWebRequest]::Create("$Uri")
            $_Session.Credentials = New-Object
System.Net.NetworkCredential($UserName, $Password)
            $_Session.Method = $Method
            $_Session.UsePassive = $true
            $_Session.EnableSSL = $true
            $_Session.UseBinary = $true
            $_Session.KeepAlive = $false
            $_Response = $_Session.GetResponse()
        } catch {
            Write-Host "Parameters:`n`t`$Uri = [$Uri]`n`t`$Method =
[$Method]"
        }
    }
    End
    {
        return $_Response
    }
}

Function Get-ReportsListFromFTP
{
    [CmdletBinding()]
    Param
    (
        [Parameter(Mandatory=$false)]
        [string]$Directory="",

        [Parameter(Mandatory=$false)]
        [switch]$Recursive=$false
    )
    Begin
    {
        # Set VerbosePreference to Continue so that verbose messages
are displayed.
        $VerbosePreference = 'Continue'
        $FileTree = @()
```

```
        if (!"$($RemoteSite)$($Directory)".EndsWith("/")) { $Directory
+= "/" }
    }
    Process
    {
        try {
            $FTPResponse = Get-ResponseFTP -Uri
"$($RemoteSite)$($Directory)" -Method
"$([System.Net.WebRequestMethods+Ftp]::ListDirectoryDetails)"
            $FTPResponseStream = $FTPResponse.GetResponseStream()
            $FTPStreamReader = New-Object
System.IO.StreamReader($FTPResponseStream, "UTF-8")

            $dirListing = $($FTPStreamReader.ReadToEnd()) -split
[System.Environment]::NewLine

            $FTPStreamReader.Close()
            $FTPResponseStream.Close()

            foreach ($_line in $dirListing) {
                if ($_line.Length) {
                    $_lineValues = $($_line -split "\ +")
                    $_fileName = $_lineValues[8..$($_lineValues.Count-
1)]
                    $_isDirectory = $_lineValues[0].StartsWith("d")
                    if ($_isDirectory) {
                        if ($Recursive) {
                            $FileTree += ,(Get-ReportsListFromFTP -
Directory "$($Directory)$($_fileName)/" -Recursive:$Recursive)
                        }
                    } else {
                        if ($_fileName -like $reportsToArchive) {
                            $FileTree += ,(Get-FileInfoFTP
"$($Directory)$($_fileName)")
                        }
                    }
                }
            }
        } catch {
            Write-Host $_.ScriptStackTrace
        }
    }
    End
    {
        return $FileTree
    }
}

Function Get-FileInfoFTP
{
    [CmdletBinding()]
    Param
    (
        [Parameter(Mandatory=$true)]
        [ValidateNotNullOrEmpty()]
        [string]$FileName
    )
    Begin
    {
```

```
        # Set VerbosePreference to Continue so that verbose messages
are displayed.
        $VerbosePreference = 'Continue'
        $FileInfo = @{
            FullName = "$($RemoteSite)$($FileName)"
            Name = Split-Path -Path $FileName -Leaf
            Parent = "./$(Split-Path -Path $FileName -Parent)"
        }
    }
    Process
    {
        $FileInfo.LastWriteTime = (Get-ResponseFTP -Uri
$FileInfo.FullName -Method
"$([System.Net.WebRequestMethods+Ftp]::GetDateTimestamp)").LastModifie
d
        $FileInfo.Size = (Get-ResponseFTP -Uri $FileInfo.FullName -
Method
"$([System.Net.WebRequestMethods+Ftp]::GetFileSize)").ContentLength
    }
    End
    {
        return $FileInfo
    }
}

Function Get-ReportFromFTP
{
    [CmdletBinding()]
    Param
    (
        [Parameter(Mandatory=$true)]
        [ValidateNotNullOrEmpty()]
        [string]$RemoteReport,

        [Parameter(Mandatory=$true)]
        [ValidateNotNullOrEmpty()]
        [string]$LocalReport
    )
    Begin
    {
        # Set VerbosePreference to Continue so that verbose messages
are displayed.
        $VerbosePreference = 'Continue'
        [int]$BufferSize = 1024
    }
    Process
    {
        try {
            $FTPResponse = Get-ResponseFTP -Uri "$($RemoteReport)" -
Method "$([System.Net.WebRequestMethods+Ftp]::DownloadFile)"
            $FTPResponseStream = $FTPResponse.GetResponseStream()
            $LocalReportStream = New-Object System.IO.FileStream
($LocalReport,[System.IO.FileMode]::Create)
            [byte[]] $ReadBuffer = New-Object byte[] $BufferSize
            do {
                $BytesRead = $FTPResponseStream.Read($ReadBuffer, 0,
$BufferSize)
                $LocalReportStream.Write($ReadBuffer, 0, $BytesRead)
            } while ($BytesRead -ne 0)
```

```
        } catch {
            Write-Host $_.ScriptStackTrace
        }
    }
    End
    {
        try {
            $LocalReportStream.Close()
            $FTPResponseStream.Close()
        } catch {
            Write-Host $_.ScriptStackTrace
        }
    }
}

foreach ( $_file in (Get-ReportsListFromFTP -Recursive) ) {
    $destinationFullName = $("$archiveFolder\$archiveNamePattern" -f
($_file.Name, $_file.LastWriteTime, $_file.Parent))
    $destinationFolder = $(Split-Path $destinationFullName -Parent)
    if ( !(Test-Path -Path $destinationFolder -PathType Container) ) {
        New-Item -Path $destinationFolder -ItemType Directory | Out-
Null
    }
    Write-Host "Downloading '$($_file.FullName)' as
'$destinationFullName'..."
    Get-ReportFromFTP -RemoteReport $_file.FullName -LocalReport
$destinationFullName
}
```

10.5 Skript č.5

```
# # # # # # # # # # # # # # # # # # # # # #
#                                           #
# Skript c. 5 - Stazeni prilohy ze SFTP     #
#               a archivace do podslozek    #
#               dle roku a mesice           #
#                                           #
# # # # # # # # # # # # # # # # # # # # # #

. "$PSScriptRoot\SFTP-Functions.ps1"

$workFolder = [System.IO.Path]::GetFullPath("$PSScriptRoot\..\sftp")
$reportsToFind = "*.txt"
$archiveFolder = "$workFolder\Archiv"
$archiveNamePattern = "{2}\{1:yyyy}\{1:MM}\{1:yyyy-MM-dd-HHmmss_}{0}"
$newReportsFound = $false
$flagLastRun = "$workFolder\$((Get-Item -Path
$PSCommandPath).BaseName).lastRun"
$lastRunDateTime = if (Test-Path -Path $flagLastRun) { (Get-Item -Path
$flagLastRun).LastWriteTime } else { (Get-Date).AddDays(-7) }

$sObj = Get-SFTPConnectionObj

foreach ( $_file in (SFTP-GetReportsList @sObj -ReportsToFind
$reportsToFind -Recursive) ) {
    $newReportsFound = $true
```

```
    $destinationFullName = "$archiveFolder\$archiveNamePattern" -f
($_file.Name, $_file.LastWriteTime, $(Split-Path -Path $_file.FullName
-Parent))
    $destinationFolder = $(Split-Path $destinationFullName -Parent)
    if ( !(Test-Path -Path $destinationFolder -PathType Container) ) {
        New-Item -Path $destinationFolder -ItemType Directory | Out-
Null
    }
    Write-Host "Stahuji '$($_file.FullName)' jako
'$([System.IO.Path]::GetFullPath($destinationFullName))'..."
    $downloadedReports += , $(SFTP-GetReport @sObj -RemoteReport
$_file.FullName -LocalReport $destinationFullName)
}

if ( $newReportsFound ) {
    SendMail -Body " - Nasledujici '$reportsToFind' reporty byly
stazeny ze serveru:`n`t* $($downloadedReports -join "`n`t* ")"
} else {
    SendMail -Body " - Zadne nove '$reportsToFind' reporty nebyly na
serveru nalezeny."
}
if ( !(Test-Path -Path (Split-Path -Path $flagLastRun -Parent) -
PathType Container) ) {
    New-Item -Path (Split-Path -Path $flagLastRun -Parent) -ItemType
Directory | Out-Null
}
Set-Content -Path $flagLastRun -Value "If you remove this file, the
next download will include all the relevant reports that are up to 7
days old." -Encoding Default
```

10.5.1 Knihovna ke skriptu č. 5.

```
Add-Type -Path
$([System.IO.Path]::GetFullPath("$PSScriptRoot\WinSCP\WinSCPnet.dll"))

Function SendMail
{
    [CmdletBinding()]
    Param(
        [parameter(Mandatory=$true)]
        [ValidateNotNullOrEmpty()]
        [string]$Body
    )
    Begin
    {
        $today = "{0:dd.MM.yyyy @ HH:mm}" -f (Get-date)
    }
    Process
    {
        $SMTP = @{
            SmtpServer = "smtp.freesmtpservers.com"
            Port = 25
            From = "Distribuce reportu <distribuce@smtp-service.com>"
            To = "matuskr1@uhk.cz"
            Subject = "Denni reporty - $today"
            UseSsl = $false
```

```
            Body = "Vysledek zpracovani reportu stahovanych ze SFTP
dne $($today):`n$($body)"
        }
        try {
            Send-MailMessage @SMTP
        } catch [System.Exception] {
            Write-Host "Error: $($_.Exception.Message)"
        }
    }
}

Function Get-SFTPConnectionObj
{
    [CmdletBinding()]
    Param()
    Begin
    {
        # Set VerbosePreference to Continue so that verbose messages
are displayed.
        $VerbosePreference = 'Continue'
    }
    Process
    {
        $sessionObject = @{
            SessionOptions = New-Object WinSCP.SessionOptions -
Property @{
                Protocol = [WinSCP.Protocol]::Sftp
                HostName = "test.rebex.net"
                PortNumber = 22
                UserName = "demo"
                Password = "password"
                SshHostKeyFingerprint = "ssh-rsa 2048
FFMFsBkaSJbqAeiMb+4c2OJI765czqp6ArYO0GznBJo"
            }
        }
    }
    End
    {
        return $sessionObject
    }
}

Function SFTP-GetReport
{
    [CmdletBinding()]
    Param
    (
        [Parameter(Mandatory=$true)]
        [ValidateNotNullOrEmpty()]
        [WinSCP.SessionOptions]$SessionOptions,

        [Parameter(Mandatory=$true)]
        [ValidateNotNullOrEmpty()]
        [string]$RemoteReport,

        [Parameter(Mandatory=$true)]
        [ValidateNotNullOrEmpty()]
        [string]$LocalReport,
```

```
        [Parameter(Mandatory=$false)]
        [switch]$Remove
    )
    Begin
    {
        # Set VerbosePreference to Continue so that verbose messages
are displayed.
        $VerbosePreference = 'Continue'
        $result = @()
        $session = New-Object WinSCP.Session
        $session.Timeout = New-TimeSpan -Seconds 200
        $TransferOptions = New-Object WinSCP.TransferOptions -Property
@{
            ResumeSupport = New-Object WinSCP.TransferResumeSupport -
Property @{
                State = [WinSCP.TransferResumeSupportState]::Off
            }
            TransferMode = [WinSCP.TransferMode]::Binary
        }

    }
    Process
    {
        try {
            # Connect
            $session.Open($SessionOptions)
            # Download the report
            $transferResult = $session.GetFiles($RemoteReport,
$LocalReport, $Remove, $TransferOptions)
            # Throw on any error
            $transferResult.Check()
            # Fetch results
            foreach ($transfer in $transferResult.Transfers) { $result
+= $(Split-Path -Path $RemoteReport -Leaf) }
        } catch [System.Exception] {
            Write-Host "Error: $($_.Exception.Message)"
            SendMail -body $($_.Exception.Message)
        } finally {
            # Disconnect, clean up
            $session.Dispose()
        }
    }
    End
    {
        return $result
    }
}

Function SFTP-GetReportsList
{
    [CmdletBinding()]
    Param
    (
        [Parameter(Mandatory=$True)]
        [ValidateNotNullOrEmpty()]
        [WinSCP.SessionOptions]$SessionOptions,

        [Parameter(Mandatory=$True)]
        [ValidateNotNullOrEmpty()]
```

```
        [string[]]$ReportsToFind,

        [Parameter(Mandatory=$false)]
        [switch]$Recursive

    )
    Begin
    {
        # Set VerbosePreference to Continue so that verbose messages
are displayed.
        $VerbosePreference = 'Continue'
        $session = New-Object WinSCP.Session
        $fileList = @()
        $_enumOptions = if ( $Recursive ) {
[WinSCP.EnumerationOptions]::AllDirectories } else {
[WinSCP.EnumerationOptions]::None }
        $LastRunTime = if (Test-Path variable:script:lastRunDateTime)
{ $script:lastRunDateTime } else { (Get-Date).AddDays(-7) }
    }
    Process
    {
        try {
            # Connect
            $session.Open($SessionOptions)
            foreach ( $_item in $ReportsToFind.Split(",") ) {
                $_path, $_mask = "/", $null
                if ( $_item -ne "/" ) { $_path, $_mask = ( Split-Path
-Path $_item -Parent
).Replace([System.IO.Path]::DirectorySeparatorChar, '/'), ( Split-Path
-Path $_item -Leaf ) }
                $_path += if ( $_path.EndsWith("/") ) { "" } else {
"/" }
                foreach ($_report in
$session.EnumerateRemoteFiles($_path, $_mask, $_enumOptions)) {
                    if ( $LastRunTime -lt $_report.LastWriteTime ) {
                        $fileList += , $_report
                    }
                }
            }
        } catch [System.Exception] {
            Write-Host "Error: $($_.Exception.Message)"
            SendMail -body $($_.Exception.Message)
        } finally {
            # Disconnect, clean up
            $session.Dispose()
        }
    }
    End
    {
        return $fileList
    }
}
```

Metodologický seminář MES je aktivitou projektu IKLIM spolufinancovaného Evropským sociálním fondem a státním rozpočtem České republiky

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Interdisciplinární, informaticko-kognitivní, lingvistický a modulární rozvoj studia
CZ.1.07/2.2.00/28.0104

Oskenované zadání práce

Zadání bakalářské práce

Autor:	**Kryštof Matuška**
Studium:	I1800673
Studijní program:	B1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název bakalářské práce:	**Principy přenosu souborů a jejich automatizace pro zabezpečení důvěrnosti a integrity přenášených dat**
Název bakalářské práce AJ:	Principles of file transfer and their automation to ensure confidentiality and integrity of transferred data

Cíl, metody, literatura, předpoklady:

Cílem práce je navrnout, zpracovat a otestovat automatizované přenosy souborů s důrazem na zajištění důvěrnosti a integrity přenášených dat. V teoretické části autor představí protokoly pro přenos souborů s důrazem na využitelné parametry a možnosti konfigurace jejich zabezpečení pro ochranu přenášených dat. Autor dále představí možnosti a technologie pro automatizaci přenosu souborů. Na základě analýzy z teoretické části autor v praktické části navrhne, implementuje a ověří automatizaci přenosu souborů. Výstupem praktické části bude sada step-by-step postupů pro využití a implementaci automatického přenosu souborů včetně testovacích scénářů a jejich vyhodnocení.

EBRAHIM, Mokhtar a Andrew MALLETT. *Mastering Linux Shell Scripting - Second Edition: A Practical Guide to Linux Command-line, Bash Scripting, and Shell Programming*. 2. Birmingham: Packt Publishing, 2018. ISBN 978-17-858-8198-5.

WILSON, Ed. *PowerShell: [průvodce skriptováním : pro verzi 3.0. a vyšší]*. Brno: Computer Press, 2015. ISBN 978-80-251-4386-5.

Zadávající pracoviště:	Katedra informačních technologií, Fakulta informatiky a managementu
Vedoucí práce:	Ing. Tomáš Svoboda, Ph.D.
Oponent:	doc. Mgr. Josef Horálek, Ph.D.

Datum zadání závěrečné práce: 21.1.2020