VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SCP MODUL PRO ALTAP SALAMANDER 3

SCP PLUGIN FOR ALTAP SALAMANDER 3

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAKUB PAGÁČ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. ZBYNĚK KŘIVKA, Ph.D.**
SUPERVISOR

BRNO 2018

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Pagáč Jakub**
Program: Informační technologie
Název: **SCP modul pro Altap Salamander 3**
SCP Plugin for Altap Salamander 3
Kategorie: Počítačové sítě

Zadání:

1. Seznamte se s bezpečnými variantami přenosu souborů na server se zaměřením na SCP (Secure Copy Protocol).
2. Navrhněte modul umožňující bezpečnou manipulaci se soubory na vzdáleném počítači v souborovém správci Altap Salamander verze 3 (se zaměřením na 64-bitovou variantu), kde tento modul zatím není k dispozici.
3. Zásuvný modul implementujte a řádně otestujte jeho funkčnost.
4. Zhodnoťte kladné vlastnosti a nedostatky vašeho řešení a doplňte možnosti budoucího rozvoje projektu.

Literatura:

- Developing for Altap Salamander File Manager. *Altap Salamander* [online]. ALTAP, 2018. Dostupné z: https://www.altap.cz/developers/ [cit. 2018-09-25]
- libssh - The SSH Library! Dostupné z: https://www.libssh.org/ [cit. 2018-09-25]
- Michael W. Lucas: SSH Mastery: OpenSSH, PuTTY, Tunnels and Keys. Tilted Windmill Press, 2nd Edition, 2018.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a dílčí prototypy k bodu 3.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Křivka Zbyněk, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 28. října 2018

Abstrakt

Tato práce popisuje tvorbu modulu pro program Altap Salamander, který umožňuje bezpečný přenos souborů po šifrovaném kanále vytvořeném za pomoci protokolu Secure Shell. Modul je postavený na knihovně Libssh. Ta implementuje jednak protokol Secure Shell, tak i přenosové protokoly SCP a SFTP, které daný protokol pro přenos využívají. Vytvořený modul dovoluje soubory nejen přenášet, ale také je na vzdáleném serveru spravovat, jelikož pracuje v režimu vzdáleného souborového systému.

Abstract

This thesis describes creation of a module for Altap Salamanader, which allows secure file transfer over encrypted channel created using Secure Shell protocol. Module is based on Libssh library. It implements both the Secure Shell protocol and SCP and SFTP transfer protocols, which use it for transmission. The created module allows not only file transfer, but also remote file management.

Klíčová slova

přenos souborů, správce souborů, Altap Salamander, Secure Shell, SCP, SFTP, libssh

Keywords

file transfer, file manager, Altap Salamander, Secure Shell, SCP, SFTP, libssh

Citace

PAGÁČ, Jakub. *SCP modul pro Altap Salamander 3*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Zbyněk Křivka, Ph.D.

SCP modul pro Altap Salamander 3

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doktora Zbyňka Křivky. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Pagáč
16. května 2019

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Zbyňkovi Křivkovi, Ph.D. za jeho vstřícnost a odezvu při řešení práce.

Obsah

Kapitola 1

Úvod

Správa souborů je pro uživatele důležité téma. Pokud nemají soubory dobře pojmenované či zorganizované, vznikají při práci s nimi zbytečné problémy. Aby se těmto problémům vyhnuli, tak mnozí uživatelé používají programy pro správu souborů, které jim tuto práci výrazně ulehčují. Jedním z těchto programů je i Altap Salamander, umožňující rozšířit svou funkčnost za pomoci modulů. Jeden z těchto modulů dovoluje přenášet soubory mezi počítači pomocí zabezpečeného spojení vytvořeného protokolem Secure Shell. Problém tohoto modulu spočívá v tom, že je dostupný pouze ve 32bitové verzi programu a tudíž je nutné pro majitele 64bitového operačního systému používat tuto verzi programu, pokud chtějí daný modul využívat.

Cílem práce je vytvořit modul pro 64bitovou verzi programu Altap Salamander, který umožní přenášet soubory za pomoci protokolu Secure Shell. Modul bude využívat přenosových protokolů SCP a SFTP, které právě Secure Shell využívají. Pro tvorbu modulu bude použita sada nástrojů poskytovaná firmou Altap, v kombinaci s knihovnou Libssh, implementující protokol Secure Shell v jazyce C.

V kapitole 2 je popsán princip fungování protokolu Secure Shell a jeho součásti. Dále tato kapitola obsahuje seznámení s přenosovými protokoly SCP a SFTP. Kapitola 3 pojednává o programu Altap Salamander, pro který je vytvářený modul určen. Je v ní také uvedeno, jak systém modulů funguje a jaké druhy modulů existují. Kapitola 4 ukazuje vybraná existující řešení pro zabezpečený přenos souborů, přičemž klade důraz na jejich vlastnosti a funkce. Tyto poznatky jsou poté použity v kapitole 5 rozebírající návrh modulu. Implementace je poté popsaná v kapitole 6. Jsou v ní také popsány použité nástroje a knihovna Libssh. Kapitola 7 uvádí, jak byl modul testován. Závěrečná kapitola 8 hodnotí výsledný modul a navrhuje jeho možná rozšíření.

Kapitola 2

Secure Shell

Tato kapitola pojednává o protokolu Secure Shell (dále jen SSH), popisuje jeho stavbu a fungování jednotlivých částí. Dále uvádí dva protokoly, které používají SSH při přenosu souborů, jejich rozdíly a použití.

SSH je protokol, pracující na aplikační vrstvě, jehož cílem je vytvořit zabezpečené spojení mezi dvěma počítači na nezabezpečené síti. Skládá se ze tří částí [21]:

- Transportní vrstvy — zajišťuje šifrování a integritu komunikace, důvěryhodnost serveru
- Vrstvy ověření uživatele — ověřuje identitu klienta
- Spojovací vrstvy — rozděluje šifrované spojení na kanály, přes ně běží požadované služby

Využívá architekturu klient-server a pro komunikaci využívá TCP s číslem portu 22. Umožňuje vzdálený přihlášení na počítač, přenos souborů, vykonávání příkazů či směrování portů přes zabezpečenou linku. První verze protokolu nyní nazývaná SSH-1 vznikla v roce 1995. Jejím tvůrcem byl Tatu Ylonen [6]. Tato verze nebyla standardizovaná a byla volně dostupná. Vlivem velkého zájmu o SSH se Ylonen rozhodl založit firmu SSH Communications Security, které stará o vývoj a podporu SSH. Ta v roce 1996 přišla se současnou verzí protokolu SSH nazývanou SSH-2. Ta není zpětně kompatiblní s původní verzí SSH, ale přináší mnohá vylepšení oproti původní verzi. Rozšíření SSH-2 mezi uživatele přinesla implementace s názvem OpenSSH. Je to zdarma dostupná implementace, dostupná pro mnoho operačních systémů. Protokol SSH-2 byl standardizován roku 2006.

2.1 Transportní vrstva

Transportní vrstva SSH je popsána v RFC 4253 [22]. Je to protokol pracující na aplikační vrstvě, poskytující šifrování komunikace, ověření identity serveru a zabezpečení integrity spojení. Při používání přes TCP/IP server standardně naslouchá na portu 22. Na začátku komunikace, poté co je ustaveno spojení mezi klientem a serverem, si tyto objekty zašlou pakety s identifikačními řetězci. Tyto řetězce mají tvar:

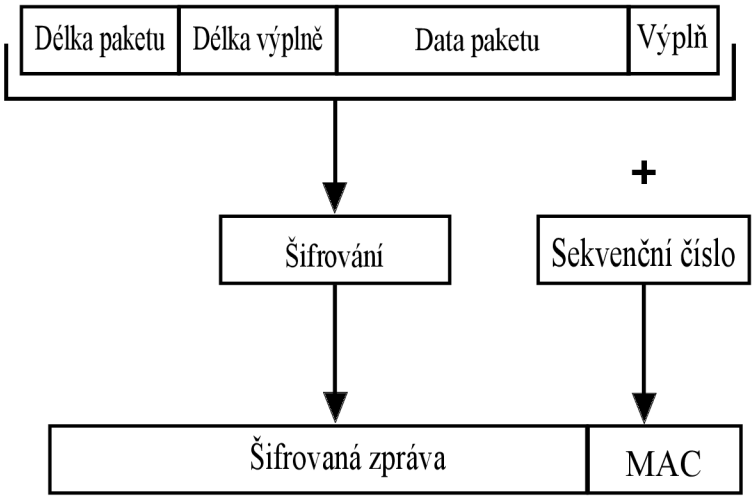
```
SSH-protoversion-softwareversion SP comments CR LF
```

Protoversion značí verzi protokolu SSH, softwareversion je verze programu implementující SSH protokol a comments je volitelný doplňující text. Pokud je v řetězci položka comments

musí být od verze programu oddělena mezerou neboli znakem ASCII 32. Na konci řetězce musí být znaky ASCII 13 a 10. Maximální délka řetězce je 255 znaků. Používá se SSH protokol verze 2.0. Pokud server podporuje starší verzi protokolu SSH, bude uvádět číslo verze 1.99.

Binary Packet Protocol

Tento protokol definuje formát SSH paketů a jaké zabezpečovací mechanismy jsou na ně používány. Struktura paketu je ukázána na obrázku 2.1.



Obrázek 2.1: Struktura paketu v transportní vrstvě SSH

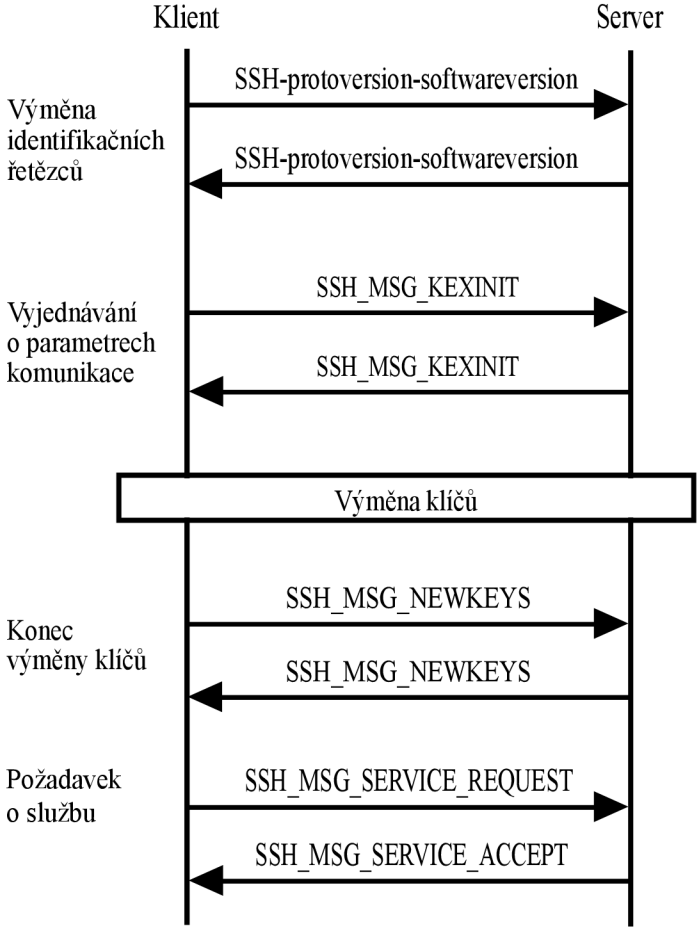
Paket se skládá z délky paketu, délky výplně, dat paketu, výplně a autentizačním kódem zprávy neboli MAC. Délka paketu se uvádí v bytech, nezahrnuje toto pole a pole MAC. Délka výplň určuje velikost pole výplň. Data paketu jsou užitečné informace, které paket nese. Tyto data lze komprimovat, na začátku se však nekomprimují. SSH podporuje volitelný formát komprese dat jménem ZLIB, popsaný v RFC 1950 a RFC 1951. Výplň zaručuje, že celková délka polí bez MAC je násobkem velikosti šifrovacího bloku nebo čísla 8. Minimum jsou 4 byty, maximum 255 bytů. MAC se používá pro ověření integrity dat. Tento kód se vypočítá po výměně klíčů z nezašifrovaných hodnot paketu a sekvenčního čísla. Sekvenční číslo se inicializuje na 0 pro první paket komunikace, pro každý další paket se inkrementuje. Na začátku je tedy toto pole prázdné. Standard říká, že implementace protokolu SSH musí dokázat zpracovat pakety velikost až 35000 bytů, protože pakety můžou přenášet například velké množství certifikátů.

Šifrování a zajištění integrity dat, používané algoritmy

O tom jaké algoritmy budou použity pro šifrování a kontrolu integrity dat se rozhoduje na začátku fáze výměny klíčů. Šifrují se všechny pole paketu kromě pole MAC. Standard definuje povinně podporovat šifru "3des-cbc", zbytek jich je volitelných. Nyní je doporučeno používat šifry uvedené v RFC 4344 označované jako "aes128-ctr", "aes196-ctr"a "aes256-ctr"[15]. U integrity dat je požadován algoritmus "hmac-sha1", doporučuje se používat algoritmy "hmac-sha2-256"a "hmac-sha2-512"z RFC 6668. U výměny klíčů jsou požadovány metody "diffie-hellman-group1-sha1"a "diffie-hellman-group14-sha1". Metodu "diffie-hellman-

group1-sha1"se však nedoporučuje používat, kvůli jednoduchosti prolomení. Místo ní je doporučeno používat algoritmy využívající eliptické křivky "ecdh-sha2-nistp256", "ecdh-sha2-nistp384"a "ecdh-sha2-nistp521"dle RFC 5656. Tyto křivky se také využívají pro doporučené metody ověření totožnosti serveru "ecdsa-sha2-nistp256", "ecdsa-sha2-nistp384"a "ecdsa-sha2-nistp521", přičemž požadovaná je metoda "ssh-dss".

Průběh komunikace



Obrázek 2.2: Diagram komunikace transportní vrstvy

Ukázka komunikace transportní vrstvy je ukázána na obrázku 2.2. Začátek komunikace tvoří ustanovení spojení a výměna identifikačních řetězců. Poté začne fáze výměny klíčů, kdy se nejdříve klient se serverem dohodnou na tom, jaké budou používat algoritmy. To udělají zasláním zprávy `SSH_MSG_KEXINIT`, která nese informace o podporovaných algoritmech dané strany. V paketu jsou reprezentovány jako seznamy jmen rozdělené dle funkce. První prvek seznamu je vždy preferovaný algoritmus dané strany. Při výběru se požaduje, aby vybraný algoritmus byl podporován jak klientem, tak serverem. Pokud se nenalezne, bude spojení ukončeno. Po výměně zpráv `SSH_MSG_KEXINIT` započne zvolený algoritmus výměny klíčů. Výstupem z výměny klíčů jsou hodnoty sdíleného tajemství a hash. Z těchto hodnot se generují klíče pro ověření integrity dat a pro šifrování. Hash hodnota se také používá jako identifikátor relace. Konec výměny klíčů je signalizován zasláním zpráv `SSH_MSG_NEWKEYS`.

Poté se komunikace stává zabezpečenou. Nakonec je zaslán požadavek na službu zprávou `SSH_MSG_SERVICE_REQUEST`. Server odpovídá zprávou `SSH_MSG_SERVICE_ACCEPT`. Během komunikace je třeba obměňovat klíče používané pro zabezpečení komunikace, aby nedošlo k jejich zneužití, a tak k narušení bezpečnosti komunikace. Doporučuje se měnit klíče po každém gigabytu zaslaných dat, po každé hodině trvání relace nebo po zaslání 2^{31} paketů od poslední výměny. Při obměně se nemění relační identifikátor. Obměna probíhá jako standardní výměna klíčů. Nové klíče začnou platit po výměně `SSH_MSG_NEWKEYS` zpráv.

2.2 Vrstva ověření uživatele

Tato podkapitola vychází z RFC 4252 [19]. Tato vrstva zajišťuje ověření uživatelů, kteří se připojují na server. Pracuje nad transportní vrstvou, přičemž na začátku od ní musí obdržet identifikátor relace. Ten se využívá při některých autentizačních metodách. Ověřovací proces řídí server, jelikož říká, které metody je možné pro autentizaci použít. Klient má při tom možnost použít tu, která mu vyhovuje a neomezí tím nijak fungování serveru. Existuje několik různých metod pro autentizaci, přičemž jedna z nich (publickey) je povinná, jedna je speciální (none) a zbytek jich je volitelných. Zahájení autentizace se provede zasláním zprávy typu `SSH_MSG_USERAUTH_REQUEST`, jenž obsahuje položky pro:

- uživatelské jméno
- službu, která se má spustit po přihlášení
- jméno autentizační metody
- položky podle zvolené metody

Server na tuto žádost může odpovědět kladně zprávou `SSH_MSG_USERAUTH_SUCCESS`, anebo záporně zprávou `SSH_MSG_USERAUTH_FAILURE`. Záporná zpráva navíc seznam metod, které mohou pokračovat a příznak značící částečný úspěch. Tento příznak je nastaven, pokud byla žádost úspěšná, ale server požaduje doplňující informace nebo další ověřování. Na metodu none musí server vždy odpovědět záporně, pokud server vyžaduje autentizaci, jinak je odpověď kladná. Tato metoda slouží k zjištění, které autentizační metody jsou dostupné na serveru. Server také může na začátku nebo během autentizace zaslat zprávu `SSH_MSG_USERAUTH_BANNER`. Ta nese textovou zprávu v kódování UTF-8 například s informacemi pro klienta.

Metoda publickey

Metoda publickey je jediná metoda, která musí být povinně podporována na každém serveru. Tato metoda pracuje s veřejným a soukromým klíčem uživatele. K veřejnému klíči má přístup server a soukromý klíč vlastní uživatel žádající o přístup. Jako součást této metody se zasílá příznak, který určuje, co s žádostí server bude dělat. Pokud je nastaven na FALSE, tak server pouze zkontroluje, zda je možné provést autentizaci na základě zaslaného veřejného klíče. Server odpoví zprávou `SSH_MSG_USERAUTH_FAILURE` nebo `SSH_MSG_USERAUTH_PK_OK`, podle toho jestli klíč příjme. Nastavením příznaku na TRUE bude server zprávu brát jako žádost o autentizaci. V samotné autentizační zprávě se nachází jméno algoritmu klíčů, veřejný klíč uživatele a podpis vytvořený soukromým klíčem uživatele. Server při přijetí této zprávy musí ověřit, že lze veřejný klíč použít k autentizaci a že podpis soukromého klíče odpovídá veřejnému klíči. Pokud vše proběhne v pořádku server zašle zprávu `SUCCESS`, jinak `FAILURE`.

Metoda password

Tato metoda provádí autentizaci na základě hesla, které uživatel zašle ve zprávě. Server toto přijaté heslo porovná vůči jeho databázi hesel. Heslo je ve zprávě v otevřené formě, jeho šifrování probíhá v transportní vrstvě. Zasílá se v kódování UTF-8. Pokud není heslo aktuální, vypršela mu platnost, tak tato metoda navíc nabízí možnost změny hesla. Při zasílání zprávy tu máme stejně jako u metody publickey příznak, který když má hodnotu FALSE, tak se jedná o autentizaci a pokud má příznak hodnotu TRUE, tak se jedná o změnu hesla. Pokud heslu vypršela platnost a server požaduje jeho změnu zašle zprávu `SSH_MSG_USERAUTH_PASSWD_CHANGEREQ`. Heslo je možné změnit i bez výzvy serveru. Server odpoví zprávou `SUCCESS` nebo `FAILURE` s částečným úspěchem, pokud bylo heslo úspěšně změněno, anebo zprávou `FAILURE` bez částečného úspěchu či `PASSWD_CHANGEREQ`, pokud heslo změněno nebylo.

Metoda hostbased

Poslední metoda, která je definována v základu SSH je metoda hostbased. Tato metoda provádí autentizaci mezi serverem a počítačem, na kterém je uživatel přihlášen. Používají se zde také veřejný a soukromý klíč, jenže v tomto případě klíč identifikuje počítač nikoli pouze uživatele. Při zasílání zprávy klient přikládá síťové jméno stanice, uživatelské jméno na klientském počítači, veřejný klíč a případné certifikáty k němu patřící, a podpis soukromým klíčem. Server při přijetí zprávy musí potvrdit, že klíč patří ke jménu stanice, podpis odpovídá veřejnému klíči a že uživatel má povolen přístup z dané stanice. Je také doporučeno, aby byla ověřena síťová adresa připojovaného počítače, kvůli možnosti zneužití soukromého klíče stanice.

Metoda keyboard-interactive

Tato metoda byla definována v RFC 4256 [7]. Je označována jako univerzální, protože server může měnit požadavky na autentizaci a přitom neovlivní implementaci klienta. To umožňuje vytvářet nové metody autentizace s použitím mechanismů jako heslo na jedno použití či výzva-odpověď. Klient na začátku zašle zprávu `SSH_MSG_USERAUTH_REQUEST`, načež server odpoví zprávou `SSH_MSG_USERAUTH_INFO_REQUEST`. Tato zpráva obsahuje instrukce k autentizaci, počet výzev, na které chce server znát odpověď, jednotlivé výzvy a příznaky k nim určující, zda se má odpověď uživatelovi vypsat na terminál. Poté, co klient získá od uživatele požadovaná data, odešle je zprávou `SSH_MSG_USERAUTH_INFO_RESPONSE` a server buď autentizaci potvrdí, zamítne nebo požaduje další informace od klienta. Pokud chce server další informace, musí počkat na odezvu od klienta.

Metody GSS-API

Generic Security Service Application Program Interface neboli GSS-API je standardizované rozhraní, umožňující programům používat různě bezpečnostní mechanismy, aniž by se musela měnit implementace, kvůli jejich nekompatibilitě. GSS-API je definováno v RFC 2743, metody pro SSH obsahuje RFC 4462 [8]. Uvádí se zde dvě metody, první z nich se označuje jako "gssapi-with-mic". Tato metoda autentizuje na základě výměny tokenů mezi klientem a serverem. Na začátku klient zašle zprávu `SSH_MSG_USERAUTH_REQUEST`, která obsahuje identifikátory podporovaných metod seřazených podle priority. Server odpovídá zprávou `SSH_MSG_USERAUTH_GSSAPI_RESPONSE` obsahující identifikátor vybraného mecha-

nismu. Poté si server a klient vyměňují zprávy SSH_MSG_USERAUTH_GSSAPI_TOKEN, dokud mechanismus neskončí. Pro úspěch autentizace musí být po skončení mechanismu ustaven kontext mezi klientem a serverem.
Druhá metoda se nazývá "gssapi-keyex". Funguje na tom, že se využívá již ustanoveného kontextu, který vznikne, pokud byli při výměně klíčů v transportní vrstvě použity metody založené na GSS-API. Poté stačí klientovi zaslat zprávu SSH_MSG_USERAUTH_REQUEST obsahující integritní kód na základě již vzniklého kontextu. Server tento integritní kód ověří a rozhodne o autentizaci uživatele.

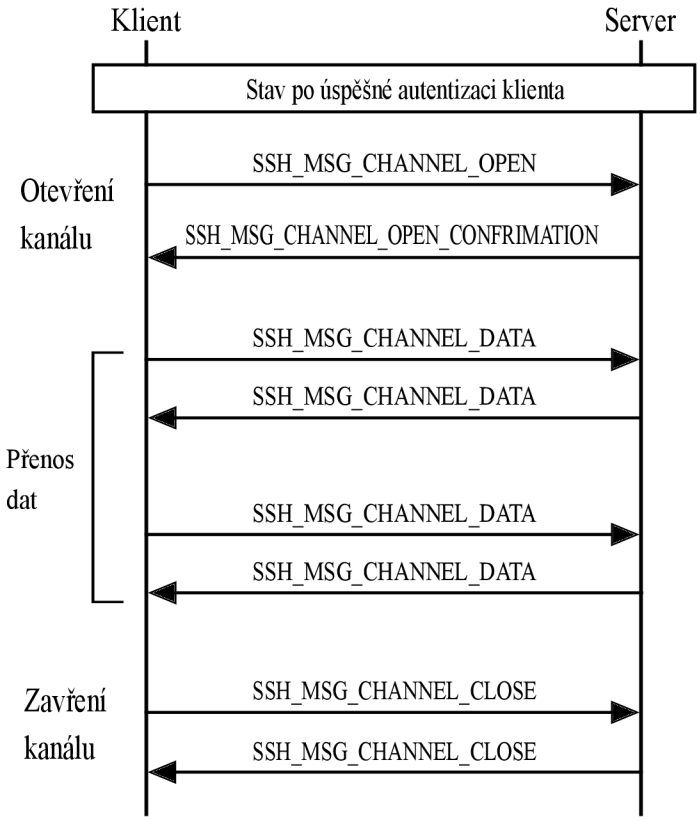
2.3 Spojovací vrstva

Spojovací vrstvu definuje RFC 4254 [20]. Tato vrstva pracuje nad transportní a autentizační vrstvou. Umožňuje provádět například vzdálené přihlašování, vzdálené provádění příkazů nebo směrování TCP/IP spojení a protokolu X11. Většina komunikace je vedena přes takzvané kanály, což jsou logicky oddělené komunikační roury, které vedou po jedné fyzické lince.

Systém kanálů

Kanál je identifikován svým číslem, o jeho vytvoření může požádat klient i server. Číslo kanálu nemusí být na obou koncích stejné. Kromě čísla má kanál svůj typ a velikost okna. Typ kanálu určuje jeho funkci. Velikost okna specifikuje množství dat, které je možné kanálu zaslat. Pokud je okno moc malé, musí druhá strana počkat na upravení jeho velikosti. Maximální velikost okna je $2^{32} - 1$ bytů. Při vytvoření kanálu si nejdříve daná strana alokuje lokální číslo kanálu, a poté zašle zprávu typu SSH_MSG_CHANNEL_OPEN. Tato zpráva obsahuje typ kanálu, číslo kanálu odesílatele, počáteční velikost okna, maximální velikost paketů a data dle požadovaného typu kanálu. Pokud žádost uspěje, tak je zaslána zpráva SSH_MSG_CHANNEL_OPEN_CONFRIMATION. Tato zpráva nese stejné informace jako zpráva žádosti, ale navíc nese i číslo kanálu druhé strany, které se používá při zasílání zpráv přes kanál. Při neúspěchu se odpovídá zprávou SSH_MSG_CHANNEL_OPEN_FAILURE, nesoucí kód důvodu neustavení kanálu s jeho popisem.

Po úspěšném vytvoření kanálu lze přenášet data zprávou SSH_MSG_CHANNEL_DATA. Nelze zaslat více dat než je aktuální velikost okna nebo maximální velikost paketu, podle toho, která hodnota je menší. Velikost paketu také nesmí být větší, než dovoluje transportní vrstva. Velikost okna se snižuje množstvím odeslaných dat. Jakmile se okno kanálu zmenší natolik, že již nelze posílat žádná další data, tak je potřeba upravit jeho velikost. K tomu je určena zpráva SSH_MSG_CHANNEL_WINDOW_ADJUST. Ta nese počet bytů, o které se okno zvětší. Některé kanály podporují několik typů dat, například data z chybového výstupu. Pro ně se používá zpráva SSH_MSG_CHANNEL_EXTENDED_DATA, nesoucí informaci o typu dat, jenž přenáší. Pokud již strana nebude zasílat žádna data přes kanál, měla by zaslat zprávu SSH_MSG_CHANNEL_EOF. Tato zpráva pouze upozorňuje na konec komunikace, ale přitom neuzavírá kanál. Pro zavření kanálu slouží zpráva SSH_MSG_CHANNEL_CLOSE, kterou musí zaslat obě strany. Až poté, co obě strany danou zprávu obdrží, je kanál považován za zavřený. Tyto dvě ukončovací zprávy neovlivňují velikost okna kanálu, tudíž mohou být odeslány, i když není okno dostatečně velké. Příklad komunikace spojovací vrstvy ukazuje obrázek 2.3. Nejdříve se po úspěšné autentizaci uživatele mezi klientem a serverem vytvoří kanál, poté proběhne několik datových přenosů a nakonec je kanál uzavřen.



Obrázek 2.3: Diagram komunikace spojovací vrstvy

Žádosti a funkce spojovací vrstvy

Pro využití některých funkcí je potřeba zaslat žádost o povolení této služby. Spojovací vrstva obsahuje dva typy žádostí: globální žádost a žádost určenou pro kanál. Globální požadavky ovlivňují celé spojení, používají se například pro směrování TCP/IP portů. Globální požadavek je zaslán zprávou `SSH_MSG_GLOBAL_REQUEST`, která nese název požadavku, zda je očekávaná odpověď na tuto zprávu a data potřebná pro požadavek. Pokud je požadována odpověď, tak druhá strana pošle buď zprávu `SSH_MSG_REQUEST_SUCCESS` nebo zprávu `SSH_MSG_REQUEST_FAILURE`, podle toho, zda byla žádost rozpoznána a zda je tato služba podporována. Žádost pro kanál slouží k rozšíření funkčnosti určitého typu kanálu nebo získání informací o stavu nějaké operace. Lze je identifikovat podle zprávy `SSH_MSG_CHANNEL_REQUEST`. Ta je podobná zprávě pro globální požadavek, navíc však nese číslo kanálu, pro který je požadavek určen. Jako odpověď na žádost pro kanál se zasílají zprávy `SSH_MSG_CHANNEL_SUCCESS` a `SSH_MSG_CHANNEL_FAILURE`. Ty v sobě taktéž nesou číslo kanálu.

Vzdálený přístup

Pro vzdálené spouštění programů se používá kanál typu session. Vytvoří se zasláním zprávy pro vytvoření kanálu, kde se uvede jako typ kanálu právě řetězec "session". Tento typ kanálu podporuje několik druhů žádostí. První je žádost o alokaci pseudo-terminálu, přes který lze se zařízením pracovat. K tomu slouží požadavek s názvem "pty-req", se kterým se navíc zasílají parametry jako výška a šířka terminálu či režimy, které budou upravovat některé

funkce terminálu. Pokud se rozměry okna terminálu změní, může o tom klient informovat server požadavkem s názvem "window-change". Ten nese nové parametry terminálu. Dále lze terminálu povolit řízení toku programu zasláním požadavku typu "xon-xoff", což umožní používat zkratky Ctrl-S a Ctrl-Q. Po ustavení kanálu se na serveru spustí určitý program. Tím může být shell, nějaká aplikace nebo subsystém. V jednom kanálu může běžet pouze jeden program. Pro spuštění aplikace se odešle požadavek s názvem "exec", pro shell s názvem "shell"a pro subsystém s názvem "subsystem". U aplikace a subsystému se také zasílá jméno programu, který chceme spustit. Pro tyto požadavky je doporučeno požadovat odpověď. Lze také nastavit proměnnou prostředí, se kterou mohou programy později pracovat. K tomu slouží požadavek typu "env". Ten nese název dané proměnné a její hodnotu. Programu je možné také zaslat signál přes žádost "signal", kde se uvede jméno signálu bez předpony SIG. Pokud byl spuštěn příkaz a je třeba znát jeho návratovou hodnotu, lze o ni požádat žádostí s názvem "exit-status". Po jejím získání je třeba kanál zavřít. Násilné skončení programu se signalizuje žádostí typu "exit-signal".

Směrování TCP/IP portů a směrování protokolu X11

U směrování portů existují dvě možnosti, jak směrovat. Lze směrovat místní port nebo port vzdálený. Místní port se nachází u klienta, vzdálený na serveru. Pro přeposílání dat ze vzdáleného portu je nutné zaslat globální požadavek typu "tcpip-forward". Tento požadavek nese IP adresu nebo doménové jméno a port, který chceme směrovat. Jakmile na danou adresu a port přijde spojení, tak se otevře kanál typu "forwarded-tcpip". Přes něj je dané spojení vedeno ke klientovi. Při otevření se přitom specifikuje z jaké IP adresy a portu spojení přišlo, a také se pro identifikaci uvádí směrovaná adresa a port. Pro zrušení vzdáleného směrování slouží globální žádost typu "cancel-tcpip-forward". U místního směrování se žádná žádost neodesílá. Pokud přijde spojení na směrovaný místní port, tak se odešle zpráva pro otevření kanálu typu "direct-tcpip". Ta nese původní zdrojovou IP adresu a port spojení, a navíc cílovou IP adresu a port, se kterým se má server spojit. Pro směrování protokolu X11 se zašle požadavek typu "x11-req"na kanál typu session. Tento požadavek nese informace o autentizačním protokolu, autentizační cookie a čísle obrazovky. Bez této žádosti by nebylo možné otevřít samostatný kanál pro data směrování. Tento kanál poté používá typ "x11".

2.4 SSH File Transport Protocol

Cílem protokolu SSH bylo vytvořit zabezpečené spojení po internetu. Součástí měl být také systém pro zabezpečený přenos souborů. Protokoly pro tento přenos byli navrženy a implementovány, ovšem již nebyli zahrnuty do standardu protokolu SSH. I přesto jsou tyto protokoly v současné době jedny z nejpoužívanějších v oblasti bezpečného přenosu dat a jsou součástí implementací protokolu SSH. Jde o protokoly SFTP a SCP, které jsou dále popsány.

Protokol SFTP (SSH File Transfer Protocol) umožňuje bezpečně přenášet soubory a vzdáleně s nimi manipulovat přes spolehlivé zabezpečené síťové spojení vytvořené za pomoci SSH protokolu [18]. Funguje na principu klient-server s komunikačním modelem žádost-odpověď. Každá zpráva má své identifikační číslo, které je stejné pro dotaz i odpověď. Protokol funguje na spojovací vrstvě SSH, kde spadá do kategorie subsystémů.

Formát paketů a inicializace protokolu

Paket je tvořen třemi částmi: délkou, typem a daty. Délkou paketu se myslí množství dat v paketu. Formát datové části paketu se liší podle toho, o jaký typ paketu se jedná. Maximální velikost paketu je omezena na maximální velikost paketu možnou přenést přes SSH spojení. Při zahájení komunikace klient zašle paket typu `SSH_FXP_INIT`. Tento paket nese číslo verze protokolu SFTP a případně data o používaných rozšířeních. Server na něj odpovídá paketem `SSH_FXP_VERSION`. Tento paket má stejný obsah jako paket od klienta. Server zasílá nejmenší společné číslo verze protokolu. Jednotlivé verze protokolu mezi sebou nejsou kompatibilní. Nejpoužívanější verzí je protokol číslo 3. Pokud klient a server podporují stejnou verzi protokolu, je spojení inicializováno a klient může zasílat požadavky.

Zprávy od klienta

Zpráva `SSH_FXP_OPEN` slouží k vytváření a otevírání souborů. Kromě identifikačního čísla nese navíc jméno souboru, položku pflags a část pro atributy. Pflags určuje, jak se bude se souborem manipulovat, zda bude jen pro čtení či i pro zápis, popřípadě zda se má vytvořit, pokud neexistuje. Názvy souboru jsou reprezentovány jako řetězce, jednotlivé složky ve jméně jsou oddělované znakem "/". Lze používat absolutní i relativní cestu k souboru. Dále jméno souboru může obsahovat ".", což reprezentuje aktuální adresář a ".."označující rodičovský adresář. Atributy jsou zasílány tak, že na začátku je pole příznaků flags, podle kterého lze poznat, jaké za ním následují hodnoty. Přenášené atributy mohou být velikost, číselné identifikátory, oprávnění k souboru, časy přístupu a modifikace a počet dalších rozšiřujících atributů. Pokud nějaký atribut není uveden, nastaví mu server danou výchozí hodnotu. Při otevření souboru či otevření adresáře server vrací deskriptor na daný cíl. Pomocí něj lze se souborem dále manipulovat. Pro zavření souboru slouží paket `SSH_FXP_CLOSE`. Ten nese právě deskriptor souboru ve formátu řetězce. Při úspěchu server zasílá deskriptor zprávou `SSH_FXP_HANDLE`, při neúspěchu zasílá zprávu `SSH_FXP_STATUS`.

Poté, co je soubor otevřen, ho lze přečíst pomocí zprávy `SSH_FXP_READ`. Zasílá se v něm identifikátor, deskriptor souboru, offset a maximální délka v bytech. Zápis do souboru se provádí zprávou `SSH_FXP_WRITE`. Má podobný formát jako zpráva pro čtení, ale namísto délky nese data pro zápis. Na obě tyto zprávy odpovídá server zprávou `SSH_FXP_STATUS`. Dále lze soubory mazat či přejmenovávat. Maže se za pomoci zprávy `SSH_FXP_REMOVE`, která nese jméno mazaného souboru. Zpráva pro přejmenování `SSH_FXP_RENAME` umožňuje pracovat se soubory i adresáři. Přenáší staré a nové jméno objektu. Vytvoření nového adresáře je přiřazeno zprávě `SSH_FXP_MKDIR`. V ní je potřeba uvést jméno adresáře a jeho atributy. K mazání adresáře slouží zpráva `SSH_FXP_RMDIR` se jménem mazaného adresáře. K získání obsahu adresáře slouží zprávy `SSH_FXP_OPENDIR` a `SSH_FXP_READDIR`. Nejdříve se daná složka otevře pomocí první zprávy, potom se jednotlivé soubory prochází pomocí druhé zprávy. Zpráva `SSH_FXP_OPENDIR` obsahuje jméno složky, server jako odpověď vrací deskriptor dané složky, který je použit u zprávy `SSH_FXP_READDIR`. Server vrací buď zprávu `SSH_FXP_NAME`, anebo zprávu `SSH_FXP_STATUS`. Pro zjišťování atributů souboru jsou definovány 3 zprávy: `SSH_FXP_STAT`, `SSH_FXP_LSTAT` a `SSH_FXP_FSTAT`. První dvě pracují s neotevřenými soubory, přičemž první navíc pracuje se symbolickými odkazy. Obě tyto zprávy nesou jméno souboru, zatímco zpráva `SSH_FXP_FSTAT` vrací atributy otevřeného souboru na základě jeho deskriptoru. Server na tyto zprávy odpovídá zprávami `SSH_FXP_ATTRS` nebo `SSH_FXP_STATUS`. Nastavení atributů souboru provádí zprávy `SSH_FXP_SETSTAT` a `SSH_FXP_FSETSTAT`. Opět se liší v tom, zda je soubor otevřený nebo

ne. Je možné také zasílat zprávy, které jsou specifické pro určitou implementaci tohoto protokolu. K tomu je určená zpráva `SSH_FXP_EXTENDED`, jejíž obsah lze upravovat dle vlastních potřeb. Odpovědí na tuto zprávu může být libovolná zpráva podporovaná serverem nebo vlastní definovaná zpráva `SSH_FXP_EXTENDED_REPLY`.

Zprávy od serveru

Zpráva `SSH_FXP_STATUS` oznamuje stav požadavku. Nese stavovou hodnotu a chybovou zprávu. Stavová hodnota může nabývat hodnot od 0 do 8, přičemž 0 značí úspěšné dokončení operace, 1 konec souboru nebo adresáře a zbytek značí, že při vykonávání požadavku nastala chyba. Další zprávou je `SSH_FXP_HANDLE`, která nese deskriptor souboru nebo adresáře. Deskriptor má formát řetězce. Zpráva `SSH_FXP_DATA` nese řetězec bytů z požadovaného souboru. V odpovědi `SSH_FXP_NAME` server vrací jméno nebo několik jmen souborů s jejich atributy. To lze poznat pomocí položky count, která označuje počet vrácených jmen. Poslední zprávou je `SSH_FXP_ATTRS` nesoucí atributy souboru.

2.5 Secure Copy Protocol

SCP neboli Secure Copy Protocol je druhý přenosový protokol, který využívá SSH. Svojí funkčností se dá přirovnat k Remote Copy protokolu s tím, že soubory jsou zde přenášeny přes zabezpečený kanál SSH [9]. Oproti SFTP neumožňuje tolik funkcí, soubory a adresáře lze pouze přenášet. Protokol SCP implementuje stejnojmenný program scp. Ten má dva módy komunikace. Jeden se nazývá "source"mód a druhý "sink"mód. Source mód čte a odesílá soubory, sink mód slouží k jejich přijímání. Pro přechod do těchto módů se používají přepínače "-f"a "-t". Tyto přepínače jsou skryté a využívá je SSH. Je definován ještě třetí skrytý přepínač "-d", který se používá při přenášení adresářů. Když pak probíhá komunikace, tak na vzdáleném zařízení je spuštěn program scp v jednom z těchto módů, přičemž na lokálním zařízení se druhý z módů jen simuluju. Jedna strana tedy zasílá informace a druhá potvrzuje jejich správné přijetí.

Source mód a sink mód

Při zasílání dat se nejdříve zašlou informace o souboru a za nimi jdou data. Informace jsou přenášeny v textové podobě, data souborů v binární podobě. Zpráva nesoucí informace o souboru má formát: typ zprávy, oprávnění souboru, jeho velikost, jméno a znak konce řádku. Typ zprávy je zde označen písmenem "C". Informace o adresáři nese zpráva stejného formátu, typ zprávy však označuje písmeno "D". Zpráva pouze s písmenem "E"označuje konec adresáře. Toho je využito při přenášení adresářů, které je realizováno rekurzivně. Poslední zprávou source módu je zpráva nesoucí informace o časových značkách souboru. Ta není standardně odesílána a musí se povolit přepínačem "-p". Lze ji identifikovat písmenem "T" na začátku zprávy, dále nese čas modifikace, znak 0, čas přístupu a další znak 0. Při odesílání zpráv se musí počkat na odpověď druhé strany, až poté může být zaslána další zpráva.
Sink mode musí každou zprávu a dokončený přenos souboru potvrdit. K tomu používá tři různé zprávy, každá končí znakem nového řádku. Buď zašle binární 0, značící, že vše proběhlo v pořádku, nebo zašle binární 1, značící varování, anebo binární 2, značící závažnou chybu, přičemž spojení bude ukončeno. Zprávy 1 a 2 mohou být následovány řetězcem popisující nastalý problém.

Kapitola 3

Altap Salamander

Tato kapitola se věnuje programu Altap Salamander. Je v ní uveden základní výčet informací o funkcích a vlastnostech programu, popis uživatelského rozhraní, a také rozbor funkci pluginů. Pro více informací lze navštívit stránky produktu [1].

Altap Salamander je program typu správce souborů, který slouží k zobrazení a manipulaci souborů a adresářů, dostupných v daném operačním systému. Pracuje na platformě Microsoft Windows, přičemž je kompatibilní s Windows 10, 8.1, 7, Vista, XP a 2000 a podporuje jak 32 bitové, tak 64 bitové verze operačního systému. Dostupný je v 11 jazykových verzích. První verzi programu vydal Petr Šolín v roce 1997 [4]. Ta byla volně dostupná a nesla označení Servant Salamander.Po vytvoření firmy Altap v roce 1999 se Salamander změnil na komerční produkt. K přejmenování programu došlo u verze 2.5 z roku 2007. Momentálně poslední vydanou verzí je verze 3.08 z roku 2016 [5].

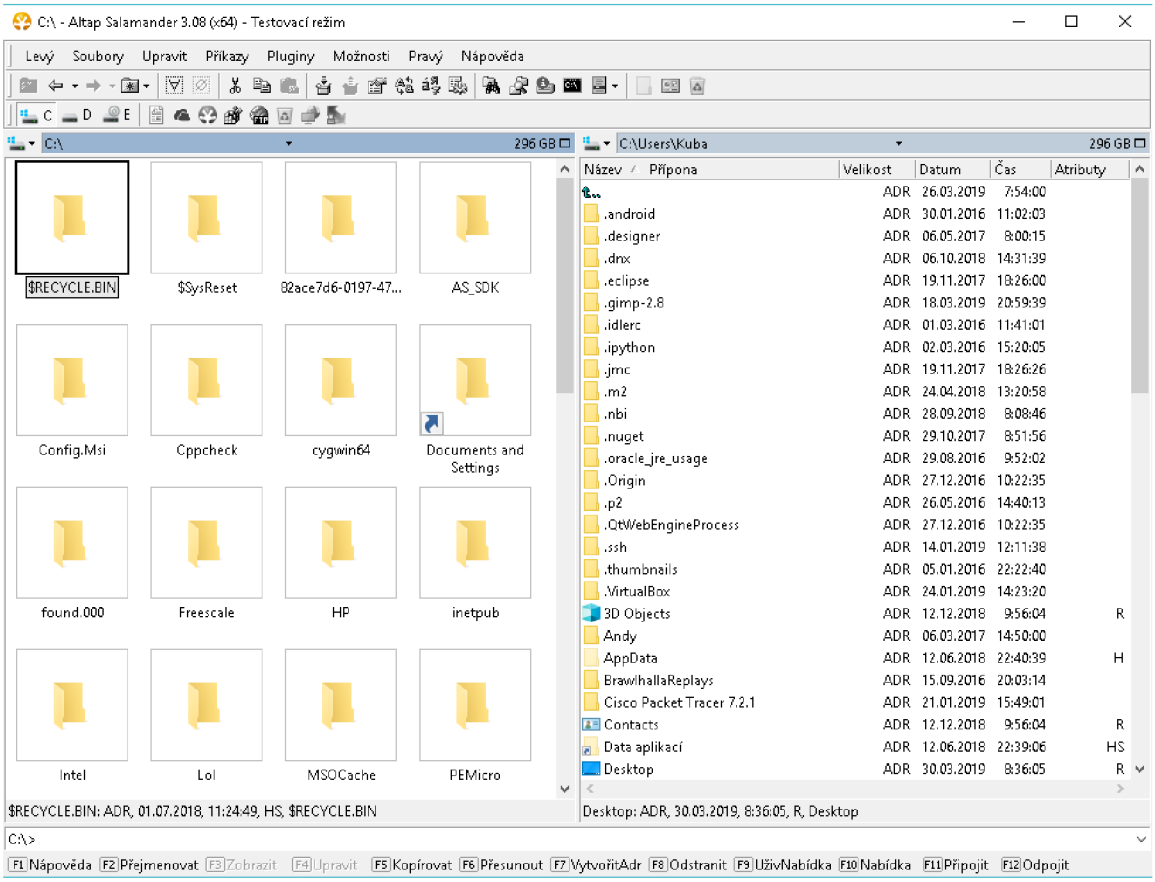
3.1 Funkce a vlastnosti

Salamander podporuje základní operace pro správu souborů jako kopírování, odstraňování, přejmenování, přesun, prohlížení a úprava souborů a adresářů. Dovoluje také vytvářet nové soubory a adresáře, pracovat s archivy nebo soubory zasílat emailem. Navíc však obsahuje příkazy pro konverzi kódování a konců řádků souboru, porovnání adresářů, změnu velikosti písmen v názvu souboru nebo jeho atributů. Lze zobrazit kolik místa soubory zabírají, jaká je velikost právě zobrazeného adresáře nebo získat podrobné informace o disku. Funkce nazvaná Uživatelská nabídka umožňuje rychlé spuštění externího programu nebo otevření dokumentu. Pro spouštění příkazů je dostupná příkazová řádka. Zajištěna je i podpora pro Windows Encrypting File System, díky níž je možné šifrovat a dešifrovat soubory v rámci souborového systému Windows. To také umožňuje zachovávat vlastníky a práva během kopírování adresářů a souborů.
Pro vzdálenou správu souborů lze používat protokoly FTP, FTPS, SCP, SFTP a SMB. Jelikož některé tyto protokoly pracují s hesly, obsahuje Salamander funkci Správce hesel, umožňující uložené hesla zabezpečit, aby k nim neměl žádný jiný software přístup, pokud není zadáno hlavní heslo. Interní prohlížeč dovoluje zobrazit soubory v textové či hexadecimální podobě. Také navíc dovede prohledávat soubory za pomoci regulárních výrazů. Regulární výrazy je možné použít i při vyhledávání souborů v systému. Hledat lze také na základě data, času, atributů či obsahu souboru.
Salamander je přizpůsoben tak, aby jej bylo možné ovládat jak myší, tak klávesnicí za po-

moci klávesových zkratek. Dostupné jsou příkazy Zpět a Vpřed pro přecházení mezi nedávno procházenými adresáři, popřípadě pokud je nějaký adresář často využíván, tak pro něj můžeme vytvořit adresářovou zkratku. Program poskytuje možnost měnit zobrazení nabídek na základě úrovně dovednosti uživatele. Dostupné jsou tři úrovně, jak pro zkušené uživatele, tak pro začátečníky. Pro začátečníky jsou zobrazeny jen nejdůležitější funkce, zatímco pro zkušené uživatele nejsou skryté žádné funkce. Nastavení programu se ukládá do registru, je možné ho importovat a exportovat, a tak přenášet mezi různými počítači. Všechny prvky programu jsou popsány v nápovědě, což umožňuje uživatelovi dohledat informace o příkazu, kterému například nerozumí citesalafeat.

3.2 Vzhled

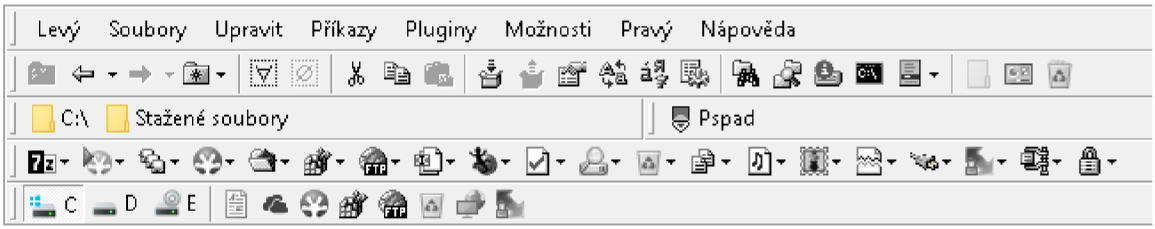
Salamander se označuje jako dvoupanelový správce souborů [1]. To znamená, že většinu velikosti hlavního okna zabírají dvě podokna, ve kterých jsou zobrazovány informace o adresářích a souborech. Základní rozhraní je ukázáno na obrázku 3.1. Skládá se ze dvou hlavních



Obrázek 3.1: Hlavní okno programu Altap Salamander

panelů, které jsou shora a zespodu obklopené nástrojovými lištami. Jeden z panelů je vždy aktivní. Aktivní panel je zvýrazněn a jedna z jeho položek je zaměřená. Zaměřené položky jsou v panelu taktéž zvýrazněny. Panely dovolují zobrazovat soubory v několika režimech, ať už řádkově s informacemi o souboru, jak je vidět v pravém panelu nebo v režimu miniatur, jak je ukázané v levém panelu. Režim miniatur navíc zobrazuje náhledy obrázků. Hlavní

panel má v sobě dva menší panely. V horní části je to panel adresáře. Ten obsahuje jméno právě zobrazovaného adresáře. Je možné ho ještě rozšířit přidáním přepínačů pro změnu řazení položek v panelu, změnu režimu zobrazení panelu či změnu diskové jednotky. V dolní části se nachází panel informací, který zobrazuje informace o právě zaměřených souborech. Vzhled hlavních panelů lze dále upravovat změnou používaného fontu, změnou barvy písma či pozadí panelu. Uživatel si také může vytvořit pravidla, která upravují vzhled pouze pro danou skupinu souborů.
Obrázek 3.2 blíže ukazuje horní nástrojovou lištu. Ta se skládá z několika panelů. První panel je označován jako panel nabídek. Obsahuje všechny dostupné nabídky programu. Je jediný povinný, nelze ho, jako ostatní panely v této liště, skrýt. Za panelem nabídek následuje panel nástrojů. Ten obsahuje zvolené nástroje programu a jeho obsah jde dále upravovat. Další jsou panely adresářových zkratek a uživatelské nabídky. Jejich obsah musí nastavit uživatel a ve výchozím stavu jsou skryté. Za nimi následuje panel pluginů. Ten nese nabídky pro dané pluginy. Poslední je panel jednotek, kterým lze přepínat zobrazované jednotky v aktivním panelu. Salamander dovoluje tyto panely libovolně řadit, jak pod sebe, tak za sebou. Při řazení za sebou se jednotlivé panely mohou překrývat. Dolní nástrojovou lištu na obrázku 3.1 tvoří příkazový řádek a dolní panel nástrojů. Příkazový řádek spouští příkazy v rámci adresáře z aktivního panelu. Dolní panel nástrojů nese položky, pro které jsou přidruženy klávesy F1-F12 na klávesnici. Položky se mění při stisku kláves Alt, Ctrl a Shift.



Obrázek 3.2: Detail horní nástrojové lišty hlavního okna programu

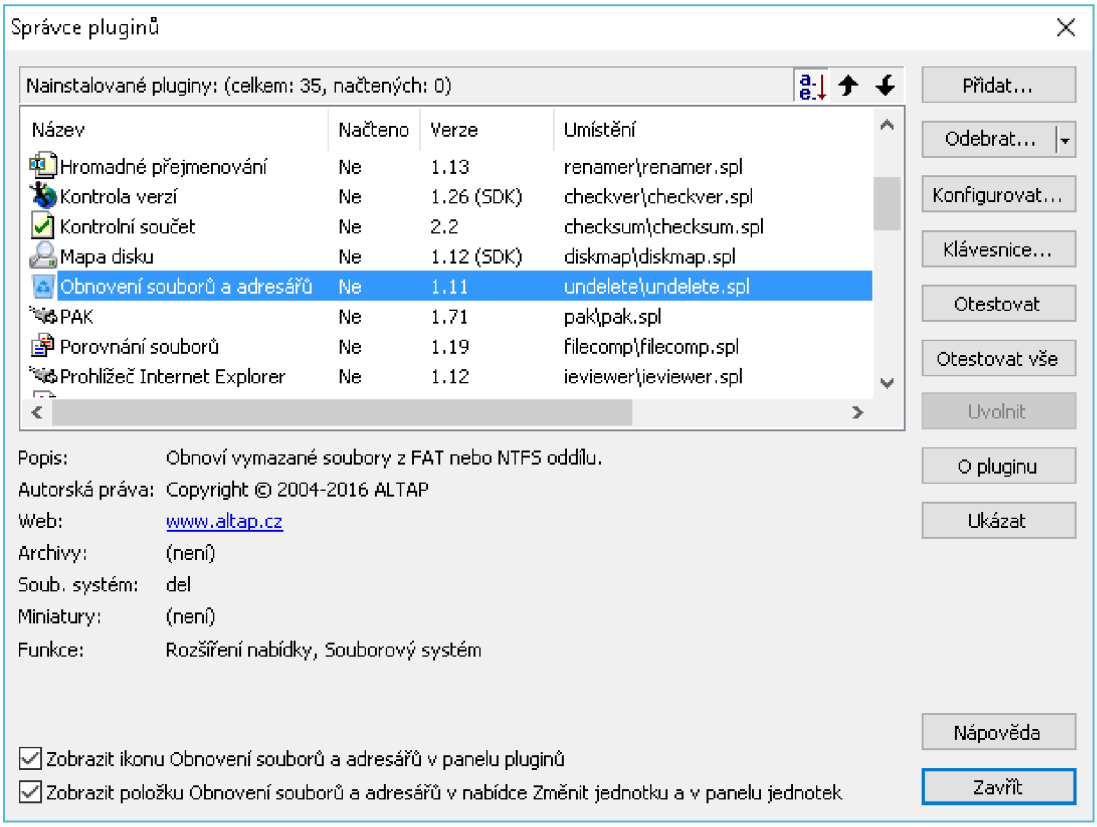
3.3 Pluginy

Salamander umožňuje rozšířit své funkce pomocí pluginů. Plugin je přídavný soubor, ve kterém jsou dané funkce popsány. Aby šlo plugin používat, musí se nejdříve do programu přidat. V Salamanderu k tomu slouží příkaz Správce pluginů ukázaný na obrázku 3.3. Ten dovoluje přidávat a odebírat pluginy, měnit nastavení pluginů, testovat jejich funkčnost nebo je odebrat z paměti programu. Pluginy jsou totiž nahrány do paměti, až když je hlavní program potřebuje, a poté co vykonají svoji funkci, lze tuto paměť uvolnit a znovu využít. To také zaručuje rychlé načítání programu při spuštění. Pluginy dělíme podle jejich funkcí, přičemž některé spadají do více kategorií [3].

Práce s archivy

Základními funkcemi těchto pluginů je zobrazit soubory v archivu a následně je extrahovat. Některé pluginy navíc dovolují provádět úpravy v archivu, případně soubory komprimovat. S archivy lze poté pracovat jako s normálními adresáři. Pokud některý plugin nepodporuje extrakci souborů nebo komprimaci do daného formátu, tak Salamander umožňuje přidat

externí program, který danou funkci zajistí. Zajištěná je podpora pro mnoho formátů například ZIP, RAR, TAR, aj.



Obrázek 3.3: Okno příkazu Správce pluginů

Souborový systém

Tyto pluginy umožňují přístup k souborům mimo základní souborový systém Windows. Dovolují komunikovat se vzdálenými servery, službami nebo zařízeními a nakládat s jejich soubory jako by byli dostupné lokálně. Salamander díky tomu podporuje protokoly FTP, FTPS, SFTP, SCP, komunikaci s počítači na lokální síti nebo úpravu lokálního registru Windows.

Rozšíření nabídky

Plugin, který náleží do této skupiny, je přidán jako položka do nabídky Pluginy a do panelu pluginů, odkud lze spouštět jeho funkce. Patří sem plugin Automatizace, umožňující běh skriptů napsaných v jazycích podporovaných ve Windows Script Host[1]. Dále sem také náleží pluginy Porovnávání souborů, Hromadné přejmenování nebo Rozdělení a spojení souborů.

Prohlížeč souborů

Jelikož Interní prohlížeč podporuje pouze textový a hexadecimální režim, byli pro Salamander vyvinuty pluginy pro zobrazení obsahu různých typů souborů. Přibyla tak podpora

[1] Automatizační technologie operačních systémů Microsoft Windows.

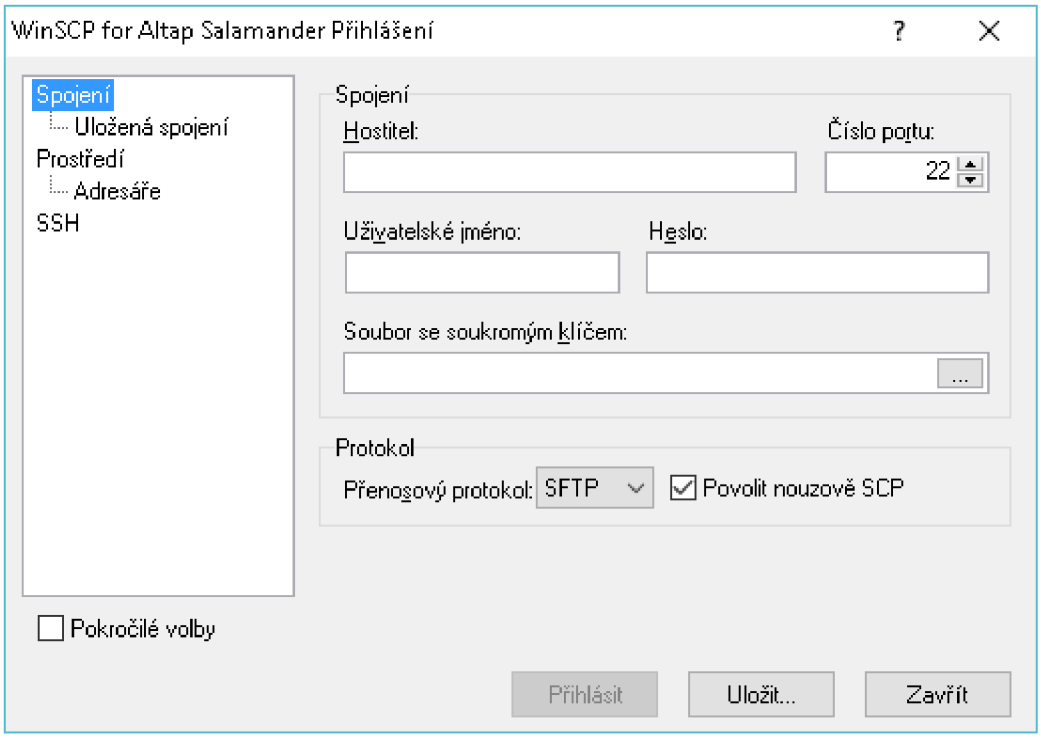
bitmapových obrázků, databázových nebo spustitelných souborů. Pro nepodporované formáty lze nastavit externí prohlížeč, ve kterém se otevřou.

Zavaděč miniatur

Salamander podporuje zobrazování miniatur souborů. Díky tomu lze vidět jejich náhled ještě před otevřením. Tuto funkci poskytuje plugin Prohlížeč PictView, který podporuje mnoho bitmapových grafických formátů. Pro další formáty je potřeba přidat další pluginy.

WinSCP plugin

Tento plugin vychází z programu WinSCP. Do Altapu přenáší většinu svých funkcí, přičemž zachovává i vzhled svých nabídek. Podporuje obě verze protokolu SSH, a také jak protokol SCP, tak protokol SFTP. Nejsou zde dostupné nejnovější zabezpečovací algoritmy, ale i tak je spojení zabezpečeno dostatečně bezpečnými algoritmy. Dostupné jsou autentizační metody public key, keyboard-interactive, password a GSS-API. Umožněno je vykonávat základní operace se soubory a měnit jejich atributy nebo spouštět vzdálené příkazy [2]. Plugin také povoluje upravovat parametry přenosu, nastavit synchronizaci adresářů nebo provádět přenosy v pozadí. Obrázek 3.4 ukazuje přihlašovací okno pro vytvoření spojení. Spojení je možné ukládat a mít je tak lehce dostupné. Plugin přebírá také uložená spojení z původního programu. Nabídka nalevo také umožňuje měnit nastavení spojení. V tomto stavu je většina nabídky skryta, rozšíří ji zatrhnutí pole Pokročilé volby. Po připojení je možné se vzdálenými položkami manipulovat jako s lokálními. Omezením je pouze to, že soubory nelze vzdáleně upravovat. Pro přenosu souborů lze aktivovat režim textového přenosu, který zaručí dle potřeby změnu kódování souboru.



Obrázek 3.4: Přihlašovací okno WinSCP pluginu pro připojení na vzdálený server

Kapitola 4

Existující řešení

Tato kapitola má za cíl přiblížit vybrané implementace protokolů SCP a SFTP dostupné na platformě Windows. Hlavní důraz bude kladen na podporované vlastnosti SSH vrstev a na funkce klientů daných protokolů. Z těchto informací poté určíme požadavky na vytvářený plugin.

4.1 OpenSSH

Tento projekt vzniknul s cílem vytvořit implementaci protokolu SSH, která by byla volně licencovaná a jejíž kód by byl jednoduchý a udržovatelný. Vývojáři projektu OpenBSD, kteří s touto myšlenkou přišli, vytvořili první verzi OpenSSH v roce 1999. Ta vyšla právě na operační systém OpenBSD, kde se stala jeho součástí. Šlo tehdy o implementaci protokolu SSH verze 1. Podpora pro SSH verze 2 přišla následující rok, stejně jako podpora protokolu SFTP. Ihned po vydání první verze OpenSSH také započali práce na vytvoření verzí pro další operační systémy [11]. První přišla verze pro Linux, následovali verze pro další Unixové systémy, nyní je dostupná i verze pro systém Windows.
Sada programů OpenSSH je nejpoužívanější implementací protokolu SSH. Proto je důležitá jejich vnitřní stavba, jaké části SSH podporují a jaké nikoliv. Příkladem může být například zrušení podpory SSH verze 1 a potenciálně nebezpečných algoritmů. Nyní je podporován pouze protokol verze 2. U transportní vrstvy SSH jsou dostupné nejnovější doporučené algoritmy, přičemž některé mají dokonce dvě verze. Podporovány jsou všechny metody autentizace a funkce spojovací vrstvy SSH [10].
Balík OpenSSH tvoří klienti protokolů, programy pro správu autentizačních klíčů a programy zajišťující služby serveru. Práce s nimi je uzpůsobena pro příkazovou řádku. Program sftp je klient protokolu SFTP verze 3. Na základě argumentů při spuštění lze měnit parametry SSH spojení ale i mód práce programu. Pokud uvedeme cestu ke vzdáleným souborům, tak klient dané soubory po autentizaci stáhne a skončí, jinak se klient připojí k serveru a začne pracovat v interaktivním módu, kde naslouchá příkazům uvedeným na obrázku 4.1a. Díky nim lze procházet vzdálené i lokální adresáře, měnit jména a atributy souborů nebo soubory přenášet. Pokud by přenos selhal, je možné jej obnovit za pomoci příkazů reget nebo reput [12]. Klient protokolu SCP se nazývá scp. Ten umožňuje kopírování souborů mezi ním a serverem. Jeho rozhraní je ukázáno na obrázku 4.1b. I zde je umožněno měnit šifrování, metody autentizace či jiné parametry spojení. Při spuštění se první uvádí zdrojová cesta a za ní cílová cesta. Je možné přenášet i adresáře, je však potřeba definovat argument r. Pokud již soubor existuje, bude po přenesení přepsán.

```
sftp> help
Available commands:
bye                                Quit sftp
cd path                            Change remote directory to 'path'
chgrp grp path                     Change group of file 'path' to 'grp'
chmod mode path                    Change permissions of file 'path' to 'mode'
chown own path                     Change owner of file 'path' to 'own'
df [-hi] [path]                    Display statistics for current directory or
                                   filesystem containing 'path'
exit                               Quit sftp
get [-afPpRr] remote [local]       Download file
reget [-fPpRr] remote [local]      Resume download file
reput [-fPpRr] [local] remote      Resume upload file
help                               Display this help text
lcd path                           Change local directory to 'path'
lls [ls-options [path]]            Display local directory listing
lmkdir path                        Create local directory
ln [-s] oldpath newpath            Link remote file (-s for symlink)
lpwd                               Print local working directory
ls [-1afhlnrSt] [path]             Display remote directory listing
lumask umask                       Set local umask to 'umask'
mkdir path                         Create remote directory
progress                           Toggle display of progress meter
put [-afPpRr] local [remote]       Upload file
pwd                                Display remote working directory
quit                               Quit sftp
rename oldpath newpath             Rename remote file
rm path                            Delete remote file
rmdir path                         Remove remote directory
symlink oldpath newpath            Symlink remote file
version                            Show SFTP version
!command                           Execute 'command' in local shell
!                                  Escape to local shell
?                                  Synonym for help
```

(a) Nápověda aplikace sftp

```
usage: scp [-346BCpqrv] [-c cipher] [-F ssh_config] [-i identity_file]
           [-l limit] [-o ssh_option] [-P port] [-S program] source ... target
```

(b) Rozhraní programu scp s jeho přepínači

Obrázek 4.1: Klienti protokolů ze sady OpenSSH

4.2 PuTTY

PuTTY je SSH a Telnet klient vyvinutý pro systém Windows a unixové platformy. Autorem je Simon Tatham. První verze vyšla v roce 1999, podpora pro SSH verze 2 byla přidána v roce 2000 a pro protokol SFTP v roce 2002 [16]. Kromě SSH klienta jsou také dostupní i klienti pro protokoly SCP a SFTP nebo aplikace pro práci s autentizačními klíči. PuTTY používá svůj vlastní formát pro ukládání privátních klíčů. Je však možné převést privátní klíče vygenerované v OpenSSH do tohoto formátu.
Podporovány jsou oba protokoly SSH, přičemž využívání SSH verze 1 se musí povolit v nastavení programu. Dostupné jsou všechny doporučené algoritmy pro zabezpečení komunikace a kvůli podpoře starších serverů i některé již nepodporované algoritmy. Autentizovat lze uživatele pomocí metod public key, keyboard-interactive nebo GSS-API. PuTTY umožňuje jak vzdálenou správu, tak směrování portů a protokolu X11. Klient protokolu SFTP zvaný psftp je interaktivní aplikace pro příkazovou řádku. Při spuštění není potřeba specifikovat server, na který se má připojit. Nastavení spojení lze upravovat uvedením dalších parametrů. To umožňuje měnit autentizační metody nebo číslo cílového portu. Po spuštění lze zadávat příkazy zobrazené na obrázku 4.2a. Ty jsou téměř totožné jako v implementaci od OpenSSH, součástí jsou však i jinak pojmenované příkazy jako pwd, lpwd, ren a mv

nebo vlastní příkazy jako open, close a mget. Výhodou je také možnost změny serveru bez nutnosti vypínání programu. Pro zobrazení parametrů příkazů lze použít příkaz help následovaný názvem příkazu [17]. Aplikaci pscp lze považovat za klienta protokolu SCP, i když

```
psftp> help
!       run a local command
bye     finish your SFTP session
cd      change your remote working directory
chmod   change file permissions and modes
close   finish your SFTP session but do not quit PSFTP
del     delete files on the remote server
dir     list remote files
exit    finish your SFTP session
get     download a file from the server to your local machine
help    give help
lcd     change local working directory
lpwd    print local working directory
ls      list remote files
mget    download multiple files at once
mkdir   create directories on the remote server
mput    upload multiple files at once
mv      move or rename file(s) on the remote server
open    connect to a host
put     upload a file from your local machine to the server
pwd     print your remote working directory
quit    finish your SFTP session
reget   continue downloading files
ren     move or rename file(s) on the remote server
reput   continue uploading files
rm      delete files on the remote server
rmdir   remove directories on the remote server
```

(a) Nápověda aplikace psftp ukazující dostupné příkazy

```
Usage: pscp [options] [user@]host:source target
       pscp [options] source [source...] [user@]host:target
       pscp [options] -ls [user@]host:filespec
Options:
  -V        print version information and exit
  -pgpfp    print PGP key fingerprints and exit
  -p        preserve file attributes
  -q        quiet, don't show statistics
  -r        copy directories recursively
  -v        show verbose messages
  -load sessname  Load settings from saved session
  -P port   connect to specified port
  -l user   connect with specified username
  -pw passw login with specified password
  -1 -2     force use of particular SSH protocol version
  -4 -6     force use of IPv4 or IPv6
  -C        enable compression
  -i key    private key file for user authentication
  -noagent  disable use of Pageant
  -agent    enable use of Pageant
  -hostkey aa:bb:cc:...
            manually specify a host key (may be repeated)
  -batch    disable all interactive prompts
  -no-sanitise-stderr  don't strip control chars from standard error
  -proxycmd command
            use 'command' as local proxy
  -unsafe   allow server-side wildcards (DANGEROUS)
  -sftp     force use of SFTP protocol
  -scp      force use of SCP protocol
  -sshlog file
  -sshrawlog file
            log protocol details to a file
```

(b) Formát spuštění programu pscp s podporovanými argumenty a jejich významem

Obrázek 4.2: Rozhraní klientů ze sady Putty

ve výchozím nastavení pracuje s protokolem SFTP. Nejde o interaktivní aplikaci, takže po přenosu aplikace skončí. Při spuštění bez parametrů se zobrazí nápověda, jak zobrazuje obrázek 4.2b. Parametry jsou opět shodné s programem z balíku OpenSSH. Je tedy možné přenášet soubory oběma směry, navíc lze vypsat obsah zvoleného vzdáleného adresáře. Pro používání protokolu SCP je potřeba použít parametr scp.

4.3 WinSCP

Tuto aplikaci lze považovat za nejpopulárnějšího SCP a SFTP klienta pro Windows. Autorem je Martin Přikryl. První verze vyšla již v roce 2000. Ta implementovala protokol SCP běžící přes protokol SSH verze 1. Jednalo se pravděpodobně o prvního klienta pro přenos souborů přes SSH s grafickým uživatelským rozhraním [13]. V dalších verzích přibyla například podpora SSH verze 2, podpora protokolů SFTP a FTP[1] nebo lokalizace do mnoha dalších jazyků. WinSCP vychází z programu PuTTY, přičemž lze přenést uložená spojení z PuTTY nebo spustit PuTTY klienta pro aktuální spojení ve WinSCP. WinSCP je také možné spustit v příkazové řádce.
Výchozí používaný protokol je SSH verze 2. Protokol verze 1 je také podporován, ale jeho používání se nedoporučuje. Zabezpečení transportní vrstvy zajišťují nejnovější doporučené algoritmy, dostupné jsou i některé algoritmy ve verzích od OpenSSH. Aby bylo možné komunikovat se staršími servery, jsou součástí i algoritmy, které by už neměli být podporovány, avšak při jejich použití je uživatel informován varováním a spojení může zrušit. Autentizaci uživatele lze provést metodami GSSAPI, public key, keyboard-interactive nebo password.

[1] File Transfer Protocol – Protokol pro přenos souborů po sítí, kde přenášená data nejsou šifrována

WinSCP není pouze SFTP a SCP klient, a tak dovoluje využívat funkcí spojovací vrstvy SSH vzdálené spouštění příkazů a směrování spojení. Při vytváření spojení je důležité, zda bude používán protokol SFTP nebo protokol SCP, protože oba pracují ve WinSCP jinak. Problém není v přenosu souborů, ale v jejich manipulaci na serveru, jelikož Winscp je interaktivní program a SCP tyto funkce nepodporuje. Výchozí protokol je tedy nastaven na SFTP, které tyto operace podporuje. Při používání protokolu SCP je potřeba, aby byl na serveru dostupný shell. Poté se využije spojovací vrstvy SSH a za pomoci vzdálených příkazů lze soubory na serveru procházet. Přenos pak provede protokol SCP. U SFTP je podporováno více verzí protokolu. Další výhodou oproti SCP je možnost obnovení přenosu souboru při jeho selhání. Pro oba protokoly lze využít textový mód přenosu. Ten se liší od binárního tím, že při přenášení se podle potřeby zajistí změna kódování souboru. Tento mód se používá pouze pro některé soubory na základě jejich přípony, ostatní jsou přenášeny standardně [14].

4.4 Požadavky na plugin

Z existujících implementací lze definovat funkční požadavky na vytvářený plugin. Při zaměření na protokol SSH, jsou nejdůležitějšími prvky:

- Použité algoritmy transportní vrstvy — používat prioritně nejnovější doporučené algoritmy, omezit používání zastaralých algoritmů, které už nemají být podporované, možnost komprese dat
- Podporované metody autentizační vrstvy — podpora nejpoužívanějších metod public key, keyboard-interactive, GSS-API a password
- Možnost využívat funkcí spojovací vrstvy — spouštění vzdálených příkazů a směrování spojení přes další server

U přenosových protokolů se projevuje rozdíl v jejich zaměření. Zatímco SCP umožňuje pouze přenos souborů, SFTP podporuje mnohem více funkcí, které by výsledný plugin měl umět. I tak lze využít SCP díky jeho efektivnosti. Hlavním protokolem by měl být SFTP, který poskytne:

- Čtení a zápis do souborů
- Procházení adresářů a zobrazení jejich obsahu
- Přesun a mazání souborů a adresářů
- Tvorba souborů a adresářů na serveru
- Změna atributů souborů

Tyto funkce pomůžou v prostředí Salamandera vytvořit klienta pro bezpečný přenos souborů. Pro větší usnadnění práce lze přidat funkce jako obnovení zrušeného stahování nebo ukládání informací o spojeních. Ty lze implementovat v případné rozšiřování funkcí pluginu a pomůžou tak usnadnit práci uživatele. Pokud uživatel nemá dostatečně výkonné spojení, určitě by také uvítal možnost zmenšit přenášené množství dat uchováváním obsahu již zobrazených adresářů a souborů. Toto ukládání do mezipaměti také zrychlí načítání adresářů, protože zruší povinnost čekání na síťovou komunikaci.

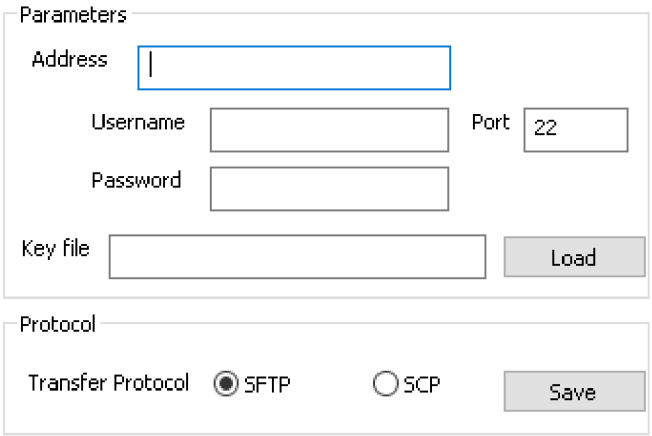
Kapitola 5

Návrh pluginu

Obsah této kapitoly je objasnění postupu řešení při tvorbě pluginu. Popsány jsou zde vlastnosti a funkce, které je třeba v implementaci řešit. Součástí je také návrh některých grafických prvků pluginu a části modelu dat. Výsledek je použit k implementaci pluginu typu souborový systém ze sekce 3.3.

5.1 Komunikace se serverem

Jedním z požadavků je, aby plugin při komunikaci využíval protokol SSH. V předchozí kapitole jsou již uvedeny jaké vlastnosti a funkce tohoto protokolu by měl plugin využívat. K nim také patří to, že plugin musí mít možnost připojit se a odpojit se od uživatelem určeného serveru. K připojení bude sloužit dialogové okno ukázané na obrázku 5.1. Do něj uživatel vyplní potřebné údaje, přičemž heslo nemusí vyplňovat ihned, ale až ho bude server požadovat. Navíc lze tyto informace uložit a při dalším připojování je načíst a vynechat potřebu je znova zadávat. Ještě před vytvořením spojení by také uživateli mělo být umožněno změnit jeho parametry. Provedené změny je potřeba zaznamenávat a ukládat, aby je uživatel nemusel opakovaně nastavovat. Pokud se připojení nezdařilo nebo bylo přerušeno, tak je třeba uživatele o této události informovat. Při přerušení by se také plugin měl pokusit o obnovení spojení.

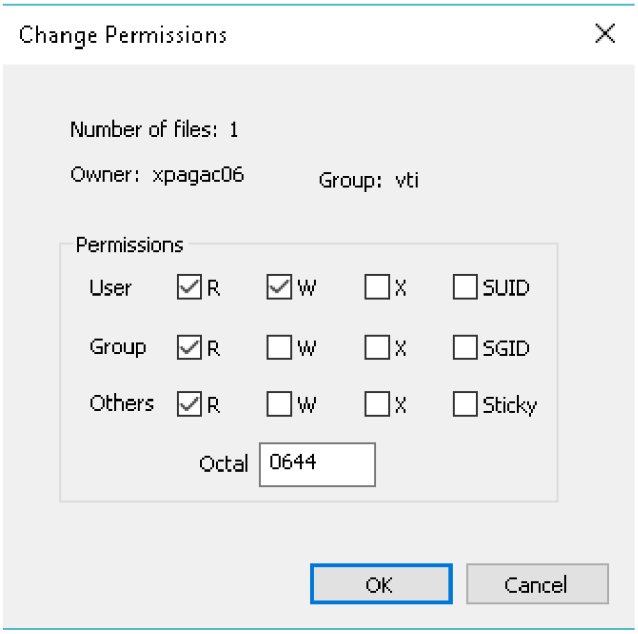


Obrázek 5.1: Podokno pro připojení ke vzdálenému serveru

5.2 Souborový systém

Jakmile dojde k připojení ke vzdálenému serveru, je potřeba získat informace o tamních souborech a adresářích, které se promítnou do zvoleného panelu Salamandera. K tomu bude použit protokol SFTP, který byl k právě tomuto účelu vytvořen a na většině serverů by měl být dostupný. Pokud by dostupný nebyl, tak operace skončí neúspěchem a spojení se ukončí. Jelikož se jedná o plugin typu souborový systém, měl by plugin podporovat množinu funkcí, kterou po něm Salamander požaduje. Jedná se hlavně o získání obsahu vzdáleného adresáře, identifikace cest na serveru, přenášení souborů mezi Salamanderem a serverem, úpravy atributů vzdálených souborů, zobrazení obsahu souboru anebo jeho mazání a přesun v rámci systému. Většina těchto funkce je také podporována v protokolu SFTP, tím pádem by neměl být problém zajistit jejich podporu.

Například změna přístupových práv souboru bude prováděna dialogem na obrázku 5.2. Ten také zobrazuje vlastníka souboru a skupinu, do které patří. Lze také měnit práva více souborům najednou. Navíc se však plugin souborového systému také stará o zavádění systému do panelu a zobrazování jeho informací. Je tedy důležité stanovit, jaké informace se budou zobrazovat. Jde hlavně o režim podrobného zobrazování, který byl ukázaný v sekci 3.2, kde plugin může definovat vlastní sloupce, které se budou zobrazovat. Vytvářený plugin by měl zobrazovat například velikost souborů nebo jejich přístupová práva.



Obrázek 5.2: Okno pro úpravu přístupových práv vybraných souborů a adresářů v panelu se souborovým systémem pluginu

5.3 Konfigurace pluginu

Jelikož plugin potřebuje mít některá nastavení od uživatele vždy k dispozici, je potřeba tyto informace někde udržovat a ukládat. Udržování konfiguračních vlastností je potřeba uzpůsobit tomu, aby jednotlivé vlastnosti byli dostupné z různých částí pluginu. Ty navíc budou reprezentovány různými datovými typy, takže je potřeba vyřešit jejich získání

a změny. Konfiguraci je také nutné ukládat, aby byla dostupná v takovém stavu, v jakém byla při ukončení programu. Je tedy nutné vyřešit jejich formát ukládání a způsob načítání při zavádění pluginu. Konfigurace by také měla mít svou implicitní podobu, která se použije v případě, že není dostupná její uložená podoba. V pluginu se k udržování konfigurace používá třída `PConfiguration`, jejíž atributy slouží k uchovávání hodnot nastavení. Tyto hodnoty se poté ukládají do Registru Windows, namísto toho, aby byli uloženy například v souboru. Tento způsob byl zvolen, protože samotný Salamander takto ukládá své nastavení a také kvůli tomu, že k tomuto přístupu definuje rozhraní pro práci s registrem. Navíc je ukázáno, jak s tímto rozhraním pracovat, takže je jednoduché ho použít.

5.4 Prvky uživatelského rozhraní

Souborovým systém pluginu používá pro svoje příkazy standardně přiřazené klávesové zkratky Salamandera. Jelikož je plugin reprezentován v liště Změnit jednotku tlačítkem s ikonou, bylo potřeba vytvořit jednoduchou ikonu pro plugin v rozměrech 16x16 pixelů. Plugin nepřidává svoje vlastní ikony pro reprezentaci souborů a adresářů, ale využívá funkce Salamandera, který načítá ikony z registru podle přípon souborů. Jak již bylo zmíněno výše, plugin obsahuje několik dialogových oken například pro vytváření spojení a správu jeho nastavení, přenos souborů nebo změnu atributů souboru. V nich je nutné zajistit načítání hodnot do prvků dialogu a zpracování hodnot získaných od uživatele. Většina dialogů má toto chování definováno ve vlastní třídě. Některé dialogy mají pouze přiřazenou funkci, která zpracovává příchozí zprávy pro daný dialog. Při vytváření jednoduchých oznamovacích dialogových oken je využito rozhraní Salamandera.

Kapitola 6

Implementace

Tato kapitola rozebírá jednotlivé prvky tvořící plugin. Na začátku jsou popsány použité nástroje a knihovna. Dále jsou zde uvedeny jednotlivé funkce pluginu a popis jejich implementace. U nic jsou zmiňované definované třídy, metody a další prvky tvořící plugin. Jednotlivé části jsou řazeny podle očekávaného pořadí využívání Salamanderem.

6.1 Použitý software

Tato část rozebírá použité nástroje a knihovnu Libssh. Popisuje jejich základní vlastnosti, funkce a stavbu. Některé části zachází více do detailu, pro lepší pochopení jejich následného využití v implementaci. Veškerý software zde uveden je volně dostupný, liší se však použitými licencemi.

Altap Salamander SDK

Tato sada nástrojů slouží ke tvorbě pluginů pro Salamander. Je poskytována zdarma na stránkách programu. Součástí jsou ukázky zdrojových kódů a soubory nutné pro jejich kompilaci, dokumentace částí programu a pluginů, a speciální verze programu, která je dostupná ve 32 a 64 bitové verzi. Od normální verze se odlišuje tím, že obsahuje ladící funkce a navíc možnost sledovat tok programu pomocí zpráv, které shromažďuje aplikace Altap Trace Server. Ta se zapne automaticky při spuštění programu. Dostupné jsou tři verze této sady, přičemž nejnovější z nich není kompatibilní se staršími verzemi. Požadavky pro fungování této sady jsou: OS MS Windows 2000 a novější, překladač Microsoft Visual C++ 2008 nebo Microsoft Visual Studio 2015.

Ukázkové pluginy

Celkem jsou přiloženy 4 ukázkové pluginy různé složitosti, ze kterých lze při budování vlastního pluginu vycházet. V dokumentaci k nim je uvedeno jak je sestavit, a také jak je poté v Salamanderu otestovat. Při sestavení je potřeba vybrat cílovou platformu, tj. zda jde o plugin pro 32 nebo 64 bitovou architekturu. Pro testování je důležité daný plugin přidat přes Manažer pluginů.

Demoplug

Tento plugin je největším z ukázkových pluginů, jelikož slouží k demonstraci implementace všech typů pluginů popsaných v sekci 3.3. Navíc také obsahuje příklady použití mnoha

metod pluginového rozhraní. Funkce archivátoru lze vyzkoušet na fiktivním formátu DOP, který se využívá i pro předvedení prohlížeče souborů. U rozšíření nabídky jsou důležité položky Controls provided by Altap Salamander a File on Disk. První ukazuje ovládací prvky dialogu, které jsou implementovány v Salamanderu. Druhá funguje pouze v případě, že je v okně pluginu označen soubor, jde tedy o příklad jak vytvořit položku s podmínkou. Souborový systém nese jméno DFS a pracuje s lokálním souborovým systémem. Slouží k představení rozhraní pluginů souborového systému a ukázání jejich funkcí. Poslední součástí je zavaděč miniatur pro soubory BMP. Při jeho testování je doporučeno odebrat plugin PictView, protože ten podporuje stejný formát, a tak může docházet k tomu, že miniatury jsou zobrazovány z něho namísto pluginu Demoplug.

Demoview a Demomenu

Tyto dva pluginy jsou odvozené od pluginu Demoplug. Demoview je implementací prohlížeče obrázků formátu DMV a zavaděče miniatur souborů BMV. DMV je fiktivní formát a BMV je přejmenovaný formát BMP. Plugin také obsahuje jednoduchou ukázku rozšíření nabídky. Demomenu je nejjednodušším ukázkovým pluginem, jelikož implementuje pouze základní funkce rozhraní pluginu a rozšíření nabídky. To lze využít při vytváření pluginu pro rozšíření nabídky.

UnFAT

Tento plugin je příkladem reálně používaného pluginu typu archivátor. Je tedy dostupný při stažení standardní verze programu. Používá se k otevírání archivů s koncovkou IMA, což jsou obrazy disket, pevných nebo USB disků se souborovým systémem FAT. V jeho zdrojovém kódu je kromě metod rozhraní archivu také ukázané, jak využívat rozhraní pro bezpečnou práci se soubory.

Sdílené soubory a knihovny

Přestože jsou ukázky pluginů dobře konstruovány a popsány, nejsou v nich ukázány všechny prvky, které lze při tvorbě pluginu využít. Proto v této části uvádím výčet souborů dostupných v SDK ve složce shared, obsahujících metody a třídy používané Salamanderem pro práci s pluginem nebo rozhraní pro využívání funkcí dostupných v Salamanderu. Díky znalosti obsahu těchto souborů lze při vývoji pluginu použít dodaných tříd a vyhnout se tak například používání externích knihoven.

Soubory svázané se Salamanderem

Tato skupina souborů obsahuje deklarace tříd a metod potřebných pro tvorbu pluginu, a také další volitelná rozhraní a funkce, které jsou pro plugin dostupné ze Salamandera. U většiny metod se nachází popis jejich funkce a parametrů, a případná doporučení a požadavky na ně.

- spl_arc.h — sada metod pluginu, ktere potřebuje Salamander pro práci s archivy
- spl_base.h — metody pro hledání chyb, pro praci se systemovymi registry, pro navázání pluginu do Salamandera a pro práci s pluginem, funkce pro získaní verze Salamandera

- spl_bzip2.h,spl_zlib.h — rozhraní pro knihovny BZIP2 a ZLIB poskytnutých ze Salamandera
- spl_com.h — reprezentace vnitřní adresářové struktury, struktura záznamu souboru a adresáře, metody upravující vzhled zobrazování souborů a sloupců v panelu
- spl_crypt.h — rozhraní pro šifrovací knihovny dostupné v Salamanderovi
- spl_file.h — skupina metod sloužících pro ošetřenou práci se soubory, při chybe zobrazují chybové hlášky
- spl_fs.h — metody Salamandera manipulující se souborovým systémem pluginu a rozharní pro jeho připojení do Salamanaderu
- spl_gen.h — obecné použitelné metody a rozhraní Salamandera pro všechny typy pluginů, například rozhraní pro vyhledávání v textu, výpočet MD5 hashe nebo Správce hesel
- spl_gui.h — metody upravující grafické rozhraní dialogových oken, umožňují přidávat a upravovat ovládací prvky jako tlačítka, text nebo ukazatele průběhu
- spl_menu.h — sada metod Salamandera pro stavbu nabídky pluginu
- spl_thum.h — metody vytvářející miniatury
- spl_vers.h — informace o verzování Salamandera
- spl_view.h — metody představující rozhraní pro prohlížeč souborů pluginu
- spl_xpth.h — metody volající dynamicky načtenou knihovnu UXTHEME.DLL stylů Windows XP, tato knihovna dokáže upravit vzhled dialogových oken a textu

Dodatečné implementační soubory

Tato skupina souborů se od předchozí odlišuje tím, že jejich implementace není skryta. Díky tomu však neobsahují tolik komentářů. Používají se k sjednocení implementace částí pluginu, aby byla zaručena kompatibilita se staršími verzemi systému Windows. Jejich použití není povinné, avšak protože Salamander používá pouze standartních knihoven Windows, je nutné, aby si programátor, při použití alternativního řešení, ohlídal problémy, které mohou vzniknout.

- winliblt — implementace oken uživatelského rozhraní, dialogů nebo fronty oken
- dbg, mhandles — funkce pro komunikaci s programem Trace Server
- auxtools — definuje třídy pro vlákna a frontu vláken
- arraylt — odlehčená verze šablony Array ze standradní knihovny C++

Práce Salamandera s pluginem

Salamander začíná využívat plugin, jakmile je přidán ve Správci pluginů. První se volají funkce `SalamanderPluginGetReqVer()` a `SalamanderPluginGetSDKVer()`, které vracejí verzi Salamandera, pro kterou lze plugin používat a verzi SDK použitého pro stavbu pluginu. Pro připojení pluginu se volá funkce `SalamanderPluginEntry()`. Ta při úspěchu vrací referenci na instanci třídy dědící z obecného rozhraní pluginu. Instanci se poté zašle zpráva `LoadConfiguration()` pro načtení základní konfigurace pluginu. Další zpráva žáda o vykonání metody `Connect()`, která naváže plugin do Salamandera. Poté je plugin připraven k práci.

Pro využívání určité funkce pluginu je potřeba nastavit její podporu ve funkci `SalamanderPluginEntry()`, a poté pro ni vytvořit instanci třídy, která dědí rozhraní definované Salamanderem pro daný modul. Tyto instance jsou předávané referencí rozhraním pluginu. Každý modul má v rozhraní přidělenou svou vlastní metodu. Salamander poté na základě práce uživatele zasílá instancím zprávy a ty pomocí třídních metod vykonávají dané funkce.

Při ukončení programu nebo při manuálním uvolnění pluginu z paměti přes Správce pluginů se instanci pluginu, pokud je plugin načtený v paměti, před uvolnění zašlou dvě zprávy. První je `SaveConfiguration()`, ta vyvolá uložení nastavení pluginu. Po ní následuje zpráva `Release()`, pro případné ukončení běžících vláken a uvolnění zdrojů pluginu, načež bude plugin uvolněn z paměti. Pokud však bude plugin ve Správci pluginů namísto uvolňování odebírán, tak se jeho nastavení nebude ukládat. Při dalším spuštění se plugin při startu programu načte a hned poté uvolní, protože jak je zmíněno v sekci 3.3, pluginy jsou načítané až dle potřeby. Salamander tedy otestuje funkčnost pluginu a potom jej může kdykoliv požadovat.

Microsoft Visual Studio

Jedná se o vývojové prostředí od firmy Microsoft podporující tvorbu nejen desktopových, ale také mobilních či webových aplikací a dalších projektů. Prostředí je dostupné pro operační systémy Windows a MacOS, přičemž verze pro MacOS neposkytuje všechny funkce dostupné ve verzi pro Windows. K tomu se váže i to, že verze pro Windows vychází ve třech edicích: Community, Professional a Enterprise. Zdarma dostupná edice Community je cílená na jednotlivce a malé firmy, kdežto placené edice Professional a Enterprise se zaměřují na podporu vývoje ve firmách a přidávají například rozsáhlejší možnosti ladění a testování projektů. Podporovány jsou například jazyky C/C++, C#, Javascript nebo Python.

Pro práci jsem si vybral verzi Visual Studio 2015, protože SDK Salamanderu obsahuje již vytvořená řešení projektů a také jelikož jsem již několik projektů v tomto programu dělal a jsem seznámen s jeho funkcemi a rozhraním. Při vytváření projektu je projekt součástí takzvaného řešení. Řešení může tvořit několik projektů. Ty potom sdílí informace pro sestavení řešení. Projekt je poté tvořen svými zdrojovými a konfiguračními soubory. Díky tomu lze každý projekt sestavovat s různými parametry na základě cílové platformy.

U program bych chtěl vyzdvihnout funkce pro podporu psaní kódu. Zejména jde o orientaci a vyhledávání v kódu a jeho rychlé úpravy. Klíčová slova, makra nebo názvy tříd jsou barevně odlišeny od zbytku kódu, což přidává na čitelnosti. Prohledávat lze buď právě prohlížený soubor, nebo celý projekt. Po otevřeném souboru se dá pohybovat na základě definovaných konstrukcí, například metod třídy nebo struktur. Při označení proměnné nebo funkce je umožněno nahlédnout na její definici, což způsobí otevření podokna s požadovanými informacemi, aniž by se měnil zobrazovaný soubor. Jednoduše se dá také přecházet mezi deklaracemi proměnných uvnitř jednoho i mezi více soubory. Při psaní kódu také po-

máhá našeptávač, který napovídá na základě již definovaných symbolů a dovoluje je automaticky dokončit. Funguje i v kontextu tříd a viditelnosti jejich atributů. Jelikož v projektu bylo potřeba přijímat vstup od uživatele, využil jsem i funkcí Editoru dialogů. Ten umožňuje vytvářet vlastní okna s běžně používanými ovládacími prvky. Ty lze dále upravovat, měnit jejich velikost, rozložení či vlastnosti. Po dokončení návrhu dialogu se výsledek uloží do souboru se zdroji, kde jsou dialog a jeho prvky identifikovaný svým ID řetězcem. Ten se poté používá v kódu při tvorbě a manipulaci s dialogem.

Libssh

Knihovna libssh implementuje protokol SSH verze 2 v jazyce C. Umožnuje tvorbu jak klienta, tak serveru. Součástí implementace jsou nejpoužívanějších algoritmy transportní vrstvy, včetně nejnovějších doporučených. Podporovány jsou autentizační metody publickey, password, keyboard-interactive a gssapi-with-mic. Na spojovací vrstvě SSH lze spouštět vzdálené příkazy, směrovat spojení TCP i protokol X11. Implementovány jsou také protokoly SFTP a SCP. Knihovna funguje na unixových operačních systémech i na platformě Windows. Tuto knihovnu jsem si vybral, protože je udržovaná a podporuje jak SFTP, tak SCP. Její rozhraní je pečlivě navržené a i díky přehledné dokumentaci se s ní dobře pracuje.

Struktura knihovny

Tato část popisuje přibližný postup použití knihovny pro připojení na vzdálený server a jak se dále serverem komunikovat. Uvádí se zde popis struktur a funkcí použitých v kódu pluginu. Nejedná se o návod k použití, ale jde spíše o seznámení s jednotlivými prvky knihovny.

Vytvoření spojení

Základním prvkem knihovny je struktura `ssh_session_struct`. Ta uchovává informace o daném SSH spojení včetně socketu. Následující funkce slouží k vytvoření spojení se serverem a jeho ukončení:

- `ssh_new()`, `ssh_free()` — první provádí alokaci struktury spojení a druhá její uvolnění z paměti
- `ssh_options_set()` — nastavuje parametry spojení jako je přihlašovací jméno uživatele, doménové jméno serveru nebo cílový port
- `ssh_connect()`, `ssh_disconnect()` — funkce pro připojení a odpojení od serveru, připojení probíhá v rámci transportní vrstvy

Zde bych chtěl zmínit odlišnost, na kterou jsem během tvorby pluginu narazil. U implementací zmíněných v kapitole 4 se ověření identity serveru provádí během výměny klíčů. Až poté, co je ověření dokončeno, se zašle zpráva `SSH_NEW_KEYS`, čímž je výměna dokončena. U této knihovny je však během vykonávání funkce `ssh_connect()` již tato zpráva zaslána ještě než došlo k ověření identity. Je to z důvodu podpory pro více druhů ověření. Toto ověřování je totiž výhradně v režii klienta, takže se může u různých implementací lišit.

Ověření identity serveru lze provádět buď automaticky, nebo ručně. U automatické metody se používá porovnávací funkce `ssh_session_is_known_server()`, která využívá stejný princip jako SSH klient ze sady OpenSSH. Jde o prověření souboru known_hosts, do kterého se ukládají klíče ověřených serverů. Při prvním připojení na server klíč nebude

známí, proto by ho měl uživatel ověřit. Po schválení lze klíč do daného souboru přidat funkcí `ssh_session_update_known_hosts()`. Pokud soubor neexistuje, bude vytvořen. Při následujícím připojení na server se uložený klíč porovnává s nově obdrženým. Pokud jsou klíče od serverů skladovány jinak, musí se porovnání provést manuálně. To spočívá v získání hodnot klíče funkcemi `ssh_get_server_publickey()` a `ssh_get_publickey_hash()`. Řešení porovnání klíčů je již na programátorovi. Po těchto funkcích je potřeba uvolnit paměť, k čemuž slouží funkce `ssh_key_free()` a `ssh_clean_pubkey_hash()`.

Autentizace uživatele

Po ověření serveru je nutné ověřit totožnost uživatele. Funkce týkající se autentizace lze identifikovat tím, že mají ve jméně řetězec userauth. Pokud server nevyžaduje provedení autentizace, tedy je povolena metoda autentizace none, tak k jejímu použití slouží funkce `ssh_userauth_none()`. Ověření metodou password umožňuje funkce `ssh_userauth_password()`, která jako jeden z parametrů přijímá heslo uživatele. U metody publickey se pro identifikaci využívá klíčů ve formátu sady OpenSSH. I zde je možné použít funkci, která prohledá adresář implicitního umístění klíčů a pokusí se s nimi o autentizaci. Jedná se o funkci `ssh_userauth_publickey_auto()`. Klíč se však může nacházet i v jiném adresáři nebo může mít jiné jméno, než je očekáváno. V tom případě ho lze načíst pomocí funkce `ssh_pki_import_privkey_file()` do struktury pro klíč `ssh_key_struct`. Pro autentizaci tímto klíčem slouží funkce `ssh_userauth_publickey()`. Po načteném klíči je nutné uvolnit paměť funkcí `ssh_key_free()`.

Autentizace metodou keyboard-interactive je rozdělena na několik kroků, přičemž hlavní roli hraje funkce `ssh_userauth_kbdint()`, která ověření zahajuje i ukončuje. Jako svou návratovou hodnotu tato funkce totiž vrací stav autentizace. Při zahájení klient obdrží od serveru informace, jak je popsáno v podsekci 2.2. Ty se dají získat funkcemi, které lze identifikovat tím, že mají v názvu řetězec kbdint_get. Pro předání odpovědi od uživatele se používá funkce `ssh_userauth_kbdint_setanswer()`.

Práce na vzdáleném serveru

Základní strukturou spojovací vrstvy je `ssh_channel_struct` sdružující informace o kanále. Funkce pracující s touto strukturou obsahují v názvu řetězec channel. Stejně jako u jiných struktur je potřeba nejdříve zavolat funkci pro alokaci. Až poté lze kanál otevřít pro komunikaci. To provádí funkce `ssh_channel_session_open()`. Jakmile je kanál otevřený, lze přes něj zasílat požadavky. Ty zasílají funkce identifikovatelné tím, že v názvu obsahují část channel_request. Jakmile je požadavek odeslán, lze využít například funkcí `ssh_channel_read()` a `ssh_channel_write()` pro příjem a zasílání dat. Dostupné jsou i další alternativní funkce, které zajišťují komunikaci přes kanál.

Pro protokoly SFTP a SCP jsou vytvořeny vlastní struktury `sftp_session_struct` a `ssh_scp_struct`, které jsou nadstavbou nad strukturou kanálu. Každý tento protokol má poté definovanou vlastní množinu funkcí. U protokolu SCP začínají názvy funkcí řetězcem ssh_scp. Realizovat lze přenos v obou módech uvedených v podsekci 2.5. Jelikož je protokol SFTP obsáhlejší a složitější než SCP, jsou k němu definovány další struktury například pro vzdálené soubory či adresáře. Knihovna implementuje protokol verze 3, stejně jako program sftp ze sady OpenSSH. Většina prvků knihovny spojených s tímto protokolem má na začátku jména slovo sftp. Dále také funkce nepracují pouze se základní strukturou SFTP spojení, takže je důležité s těmito zdroji správně zacházet. Funkce však většinou sdílejí označení se zprávami protokolu, a díky tomu je jasné, co daná funkce zajišťuje.

6.2 Zavedení pluginu do Salamandera

Při tvorbě jsem vycházel z ukázkového pluginu Demoplug. Vstupním bodem pluginu je funkce `DllMain()`. Jedná se o volitelnou funkci dynamických knihoven, která se volá při jejich zavádění a uvolňování. Zde slouží pro získání jazykově nezávislých zdrojů pluginu, jako je například ikona pluginu. Další postup již odpovídá podkapitole 6.1. Ve funkci `SalamanderPluginEntry()` se nastavují základní informace o pluginu, načítají se jazykově závislé zdroje nebo získávají ukazatelé na globální rozhraní Salamandera. Obecné rozhraní pluginu implementuje třída `CPluginInterface`. Metoda `Connect()` této třídy načte ze zdrojů ikonu pluginu a přidá ji do nabídek programu. Další metody této třídy jsou:

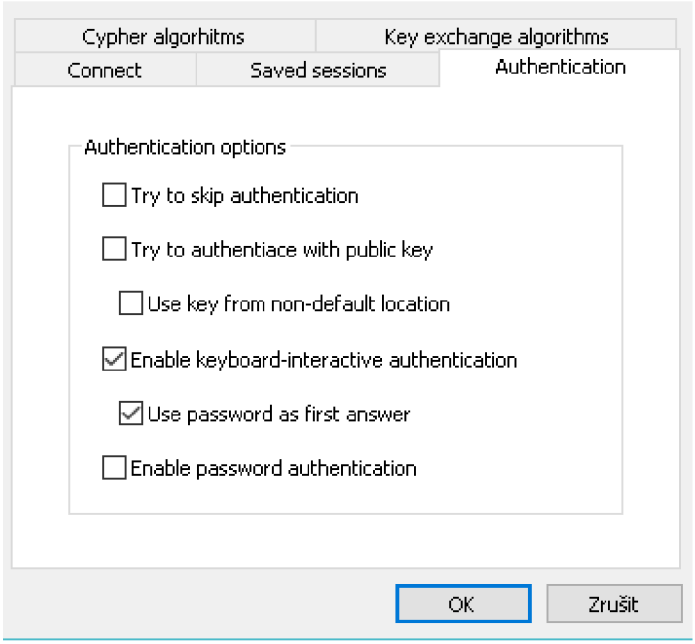
- `About()` — metoda volaná z nabídky Salamandera Nápověda, zobrazuje okno s informacemi o pluginu
- `Configuration()` — spustí dialogové okno, ve kterém lze měnit nastavení pluginu
- `GetIntrefceForForFS()` — metoda vrací referenci na instanci třídy `CPluginInterfaceForFS`, tato třída implementuje rozhraní pro správu jednotlivých souborových systémů vytvořených pluginem
- `LoadConfiguration()` — načítá konfiguraci pluginu
- `Release()` — volá se před uvolněním pluginu z paměti pro uvolnění jeho prostředků
- `SaveConfiguration()` — ukládá konfiguraci pluginu

6.3 Připojení k serveru

Výše zmíněná třída `CPluginInterfaceForFS` se stará o zavádění a odebírání instancí souborového systému do panelu. Mezi její další funkce patří tvorba podnabídek a provádění příkazů v rámci nabídky Změnit disk a spouštění položek v panelu otevřeného systému. Základní způsob vytvoření spojení je spuštěn kliknutím na položku pluginu v panelu jednotek nebo v nabídce Změnit jednotku. Tím se spustí metoda `ExecuteChangeDriveMenuItem()`. Tato metoda nejprve otevře okno pro vyplnění informací pro připojení. Toto okno se skládá z několika podoken, ve kterých lze upravovat parametry spojení anebo spravovat uložená spojení.

Obrázek 6.1 ukazuje podokno, ve kterém lze upravovat používané autentizační metody. Dané metody se používají v takovém pořadí, v jakém jsou zde uvedeny. Při povolení všech se tedy začne metodou none a poslední se vyzkouší metoda password. Další podokno dialogu pro připojení, zobrazené na obrázku 6.2, slouží k úpravě preferovaných šifrovacích algoritmů. Implicitně jsou povoleny všechny algoritmy podporované knihovnou Libssh. Uživatel zde může změnit jejich pořadí a případě některé úplně zakázat, jestliže algoritmus posune za položku zarážky. Pokud jsou za zarážkou nastaveny všechny algoritmy, tak se namísto zamítnutí připojení použije implicitní nastavení.

Při připojování je také důležité, zda byla zvolena některá položka již uloženého spojení. Ta se totiž použije namísto informací zadaných uživatelem. Obrázek 5.1 z kapitoly o návrhu ukazuje podokno s potřebnými informacemi. Z tohoto podokna se také uložená spojení vytváří a upravují. Informace o spojení udržuje třída `Session`. Její instance `CurrentSession` uchovává informace o posledním vytvořeném spojení. Uložená spojení se shromažďují v kontejneru vector `LoadedSessions`, který ukládá objekty třídy `Session`. Obsah tohoto kontej-



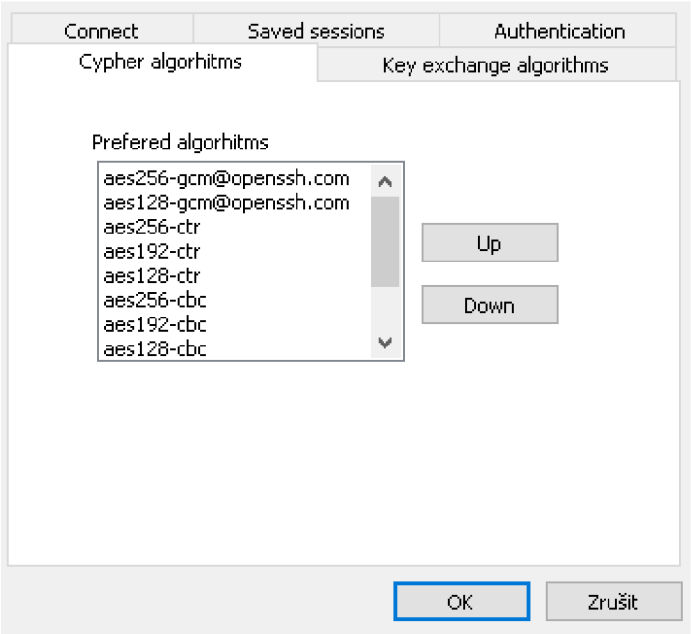
Obrázek 6.1: Podokno pro výběr metod autentizace, které se budou zkoušet během připojování na server

neru se naplňuje během načítání konfigurace, jelikož uložená spojení jsou udržována v registru. Pokud se uchovává i heslo, je v registru uloženo v zašifrovaném tvaru. Pro šifrování se používá rozhraní Salamanadera `CSalamanderCryptAbstract`, které využívá algoritmu AES. Při uložení je heslo zašifrováno, přičemž jeho dešifrování se provádí, až když je vyžádáno během autentizace.

Po potvrzení dialogu se volá metoda obecného rozhraní Salamandera `ChangePanelPathToPluginFS()`. Ta slouží nejen k nastavení požadované cesty panelu na souborový systém pluginu, ale navíc zasílá požadavek na vytvoření instance systému, pokud neexistuje. Tato funkce tedy provede, za případu že v panelu není systém pluginu, volání metody `OpenFS()` třídy `CPluginInterfaceForFS`, která vrací novou instanci třídy `CPluginFSInterface`. Tato instanci je poté nastavena do příslušného panelu a je jí zaslána zpráva `ChangePath()`, která mění aktuální cestu systému. Během provádění metody `OpenFS()` probíhá samotné připojování k serveru pomocí funkcí `setupSSH()` a `setupSFTP()`. První z nich zajišťuje připojení k serveru a veškerou autentizaci, zatímco druhá nastaví kanál pro komunikaci protokolem SFTP. Obě také pracují s funkcemi knihovny Libssh, přičemž funkce připojení, autentizace i vytvoření kanálu byli popsány v podkapitole 6.1. Poté je připojení k serveru dokončeno.

6.4 Operace nad souborovým systémem

Vzdálený souborový systém reprezentuje třída `CPluginFSInterface`. Jedná se o podtřídu abstraktní třídy `CPluginFSInterfaceAbstract` definující metody, které musí třída souborového sytému implementovat. Ty jsou většinou volány přímo ze Salamandera nebo ze třídy `CPluginInterfaceForFS()`.



Obrázek 6.2: Podokno pro úpravu podporovaných šifrovacích algoritmů, které se budou používat pro zabezpečení komunikace

Procházení a zobrazení obsahu adresářů

Jakmile je vytvořena instance souborové systému, tak se jako první požaduje provedení metody `GetSupportedServices()`. Ta vrací výčet funkcí, které jsou systémem podporovány. Jako další následuje požadavek na změnu cesty v pluginu. Tuto práci má v režii funkce `ChangePath()`. Ta zkontroluje, zda přijatá cesta patří danému systému a zda se jedná o cestu do adresáře. Pokud skončí úspěchem, tak se pro danou cestu volá metoda `ListDirectory()`. Ta slouží k získání dat o souborech a adresářích na dané cestě. Načtené informace se v ní předávají Salamanderu, který při úspěchu metody dané položky zobrazí. Také se v ní nastavuje způsob získávání ikon pro plugin a vytváří se časovač obnovující obsah adresáře. Když poté časovač vyprší, tak jeho obnovení provede metoda `Event()`.

Pokud v panelu přecházíme mezi adresáři, tak se nejprve volá metoda `CPluginInterfaceForFS::ExecuteOnFS()`. Ta provede změnu cesty vybraného systému a následně mu zašle upravenou novou cestu pro zobrazení souborů. Před změnou se v daném systému také upraví instance třídy `CTopIndexMem`, která udržuje cestu pro první index panelu aktuálního adresáře. Ten se používá jako položka pro přechod do nadřazeného adresáře. Systém novou cestu příjme a opět volá metody `ChangePath()`, `ListDirectory()` a `GetSupportedServices()`. Přecházet lze i příkazem Změnit adresář, přičemž tento příkaz může požádat o vytvoření nového připojení, pokud se v daném panelu nenachází systém patřící k tomuto pluginu. Součástí třídy systému jsou také metody pro porovnání cest `IsCurrentPath()` a `IsOurPath()`, které kontrolují předávané cesty. Pokud během práce proběhla změna adresáře a je požadováno jeho obnovení, tak se to systém dozví v metodě `AcceptChangeOnPathNotification()`.

Výpis obsahu adresáře také využívá třídy `CPluginFSDataInterface`, která slouží k vytváření vlastních sloupců pro zobrazení v panelu. Tyto sloupce se zobrazují pouze v podrobném režimu. Dále se také tato třída stará o výpis na spodní informační lištu panelu. Sloupce nastavuje metoda `SetupView()`. Ta pro načítání dat do sloupců využívá struktury `CFSData`. Instance této struktury se vytváří pro každou položku během zjišťování obsahu

adresáře. Salamander si na tyto instance ukládá odkazy a při tvorbě sloupců přes ně k datům přistupuje. Tato struktura se využívá i v metodě `GetInfoLineContent()` pro výpis na informační lištu. Obsah lišty je určen na základě právě označených položek. Pokud je označena pouze jedna, tak se vypíše její jméno, velikost či datum poslední úpravy. U více položek se ukazuje počet vybraných souborů a adresářů.

Přenos souborů

Metody rozhraní pro přenášení souborů se odlišují hlavně tím, jakým směrem se bude soubor přenášet a jak jsou volané. Důležitá je zde také metoda `GetDropEffect()`, která určuje, zda operace drag&drop mezi souborovými systémy představuje kopírování nebo přesouvání souborů. Rozpoznávání se provádí na základě porovnání zdrojové a cílové cesty. U metody `CopyOrMoveFromDiskToFS()` se přenáší soubory z lokálního disku na server. Salamander ji může volat i v případech, kdy souborový systém není aktivní v panelu nebo dokonce není ani vytvořený. V tomto pluginu je implementováno pouze kopírování souborů z disku, aby při neočekávané chybě nedošlo ke ztrátě daných dat. Při přenosu souborů z disku se tedy soubory budou pouze kopírovat.

Pokud není metoda vyvolána operací drag&drop, tak je otevřen dialog, ve kterém uživatel specifikuje cílovou cestu. Cesta musí začínat řetězcem, který systému přidělí Salamanader, aby šlo poznat, že patří k tomuto pluginu. Tento řetězec je vidět buď v panelu na začátku cesty souborového systému, nebo v Manažerovi pluginů u položky pluginu. Po potvrzení dialogu je nejdříve cesta upravena do potřebného tvaru, a poté je otevřeno okno průběhu kopírování. To zobrazuje právě přenášený soubor nebo složku a v procentech, kolik zvolených položek z panelu již bylo přeneseno. Adresář má zde stejnou procentuální váhu jako soubor. Funkce, která bude volána pro přenos položky, se určuje podle zvoleného protokolu a podle toho, zda jde o adresář nebo soubor.

Pokud byl zvolen protokol SCP, tak je pro přenos adresáře i složky použita stejná funkce `scp_write()`. Ta rozlišuje mezi souborem a adresářem pomocí parametru isDir. Uvnitř této funkce se poté nastaví kanál pro SCP přenos knihovními funkcemi `ssh_scp_new()` a `ssh_scp_init()`. Tato implementace protokolu SCP pracuje v rámci složky uvedené při vytvoření kanálu. Dále se na základě parametru isDir volá buď funkce `scp_write_file()` nebo `scp_write_directory()`. U první z nich se nejprve zkusí otevřít zvolený soubor pro čtení, následně se zjistí jeho velikost, kterou musíme předat při jeho vytváření na serveru pomocí funkce `ssh_scp_push_file()`. Jestliže byl vzdálený soubor úspěšně vytvořen, tak je následně lokální soubor čten do bufferu a poté přeposílán funkcí `ssh_scp_write()`. U druhé zmíněné funkce, která slouží pro přenášení adresářů, je zvolený místní adresář, pokud neexistuje, nejprve vytvořen na vzdáleném serveru funkcí `ssh_scp_push_directory()`. Pokud se na serveru již nacházel, tak do něj protokol SCP pouze vstoupí. Tím se změní kontext kanálu a nyní jsou soubory vytvářeny v rámci tohoto adresáře. Následně se v cyklu prochází místní adresář a jeho obsah je přenášen na server. Soubory jsou opět přenášeny funkcí `scp_write_file()` a pro adresáře je rekurzivně volaná aktuální funkce `scp_write_directory()`. Jakmile je celý adresář přenesen, tak se z něj v rámci kontextu SCP vyjde funkcí `ssh_scp_leave_directory()` a tím se pracovní adresář nastaví na jeho rodičovský adresář.

Pokud byl zvolen protokol SFTP, tak se koncepce moc neliší. Největší změna je v tom, že přenášený soubor je nejprve přenesen do dočasného souboru. Jakmile je přenos dokončen, tak je původní soubor na serveru, pokud existuje, vymazán a následně je dočasný soubor přejmenován na jeho jméno. Stejným způsobem jsou přenášeny i adresáře. Přenos souborů obstarává funkce `copyFileToServer()` a adresářů funkce `copyDirectoryToServer()`. Po-

stup přenosu je stejný jako u protokolu SCP, akorát se nepracuje v rámci kontextu, ale na základě cest serveru a používají se funkce příslušné k tomuto protokolu.

Při přenášení souborů ze serveru se volá metoda `CopyOrMoveFromFS()`. Největší rozdíl mezi touto a předchozí metodou je ten, že je nutné řešit přenášení mezi různými souborovými systémy pluginu, a také přesun v rámci právě aktivního systému. Stejně jako u předchozí funkce stojí na začátku dialogové okno nebo operace drag&drop. Následně jsou zpracovány přijaté cesty, zobrazen dialog průběhu a je zjištěno, jestli se bude přenášet soubor nebo adresář. Následně se zkontroluje sousední panel, zdali se v něm nenachází souborový systém pluginu. Pokud ano, musí se upravit cílová cesta, protože bude mít jiný tvar, než kdyby vedla na disk. Následně se podle parametru metody `copy` určí, zda se bude kopírovat nebo přesouvat, a dále se prověří, zda se nejedná o totožné souborové systémy.

Pokud jsou v panelech zobrazeny systémy různých serverů, lze mezi nimi pouze kopírovat, zatímco u stejných systémů lze provést jak kopírování, tak přesun. Při kopírování mezi systémy se položka nejdříve stáhne do lokální dočasného adresáře a následně se nahraje na druhý server. To stejné se děje při kopírování do stejného systému, jelikož protokoly SFTP a SCP nepodporují místní kopírování. Použití operace drag&drop mezi stejnými systémy vyvolá přesun zvolených položek. Jakmile bude v sousedním panelu místní disk, tak se bude rozhodovat mezi kopírováním na něj, nebo kopírováním a přesunem uvnitř systému serveru. Operaci přesunu zajišťuje funkce `moveFileOrDirectory()`. Tato funkce volá knihovní funkci protokolu SFTP `sftp_rename()`. Funkce se používá i při volbě protokolu SCP, jelikož ten nepodporuje přesouvání souborů nebo adresářů.

Při kopírování již záleží na používaném protokolu. U SCP se používá funkce `scp_read()`. Ta nejdříve vytvoří spojení kanálem SCP a nastaví kontext protokolu na daný soubor nebo adresář. Následně se opakovaně zasílají požadavky na stažení položky funkcí `ssh_scp_pull_request()`, dokud server neodpoví chybou nebo zprávou označující, že již nemá co zaslat. Pokud je zadán požadavek na stažení adresáře, tak server klientovi oznamuje jeho jednotlivé položky. Když narazí na soubor nebo další adresář, je o tom klient informován, přičemž může získat jméno daného objektu funkcí `ssh_scp_get_filename()`. Následně je potřeba přijetí potvrdit funkcí `ssh_scp_accept_request()`, což u souboru způsobí jeho zaslání a u adresáře změnu kontextu protokolu na nalezený adresář. U souboru je také potřeba získat jeho velikost pro ověření množství již zaslaných dat. Jakmile již v adresáři není žádná položka, vrací se server zpět do předchozího adresáře.

Kopírování za pomoci protokolu SFTP využívá funkcí `copyFileFromServer()` a `copyDirectoryFromServer()`. Soubory jsou zde přenášeny přímo, což znamená, že pokud soubor na serveru již existuje, bude jeho obsah přepsán. Jestliže se soubor na serveru nenachází, tak bude nově vytvořen. U druhé zmíněné funkce, je vzdálený adresář nejdříve otevřen funkcí `sftp_opendir()`, a poté je jeho obsah procházen funkcí `sftp_readdir()`. Pokud se narazí na adresář, je pro něj rekurzivně volána stejná funkce a když je nalezen soubor volá se funkce pro přenášení souborů.

Změna atributů a přejmenování souborů

Plugin dokáže měnit přístupová práva souborů i adresářů. Využívá při tom metody `ChangeAttributes()`. Ta může být volaná pouze v případě, že v panelu se souborovým systémem je označena nebo zvolena některá jeho položka. Nejdříve se ze serveru získají současné hodnoty práv pomocí funkce `sftp_stat()`. Následně se zobrazí okno pro úpravu práv implementované třídou `CChangePermissions`. Vzhled okna byl již ukázán v kapitole Návrh pluginu na obrázku 5.2. Obsah tohoto okna závisí na počtu označených položek. Pokud

je zvolen pouze jeden soubor nebo adresář, tak jsou checkboxy v dialogu dvoustavové, zatímco když je souborů označeno více, tak jsou třístavové. Třetí stav slouží k označení, že daná hodnota se u položek liší a při nastavování práv bude ponechána její původní hodnota. Aktuálně nastavená hodnota se ukazuje ve spodním poli. Pokud bylo označeno více položek a některý checkbox je ve třetím stavu, tak se hodnota nezobrazuje. Spodní pole je navíc upravovatelné, přičemž přijímá čísla od 1 do 7, protože hodnota práv se většinou uvádí oktalově. Jelikož však server zasílá i přijímá hodnotu práv desítkově, tak třída dialogu implementuje metody pro převod mezi těmito soustavami. Pokud byl zvolen alespoň jeden adresář, zobrazí se v levém spodním rohu checkbox, který ovládá rekurzivní změny práv v adresářích. Pokud byl dialog ukončen tlačítkem OK, provede se změna práv pomocí funkcí `sftp_chmod()` a `ChangePermissionsForDirectory()`. Pokud nebyla povolena rekurzivní změna, upraví se práva pouze zvoleného adresáře.

Soubory a adresáře je možné přejmenovat za pomoci metody `QuickRename()`. Při spuštění z kontextového menu nebo nabídky pro soubory, bude pro zvolenou položku otevřeno dialogové okno, do kterého uživatel zadá její nové jméno. Při editaci jména přímo v panelu se tento dialog nezobrazuje. Pokud jméno obsahuje nepovolené znaky, je moc dlouhé nebo soubor již existuje, tak se akce neprovede a dialog je znovu zobrazen. Změna se provádí pouze v rámci právě zobrazovaného adresáře, takže touto metodou nelze soubory přesouvat.

Prohlížení a mazání souborů

O zobrazení obsahu vzdáleného souboru se stará metode `ViewFile()`. Ta nejdříve vytvoří prázdný soubor v lokálním dočasném adresáři uživatel, a poté do něj přenese vybraný soubor ze serveru. Nakonec je soubor zobrazen v interním prohlížeči Salamandera. Tyto dočasné soubory jsou mazány až při rušení daného souborové systému. Důvodem je to, že uživatel může požadovat zobrazení stejného souboru několikrát za sebou. Systém si tak v atributu `TmpFiles` udržuje jména již stažených souborů a pokud na nějaký narazí, tak ho již nemusí stahovat.

Mazaní souborů a adresářů zajišťuje metoda `Delete()`. Při prvním volání může metoda vyvolat dialog pro potvrzení mazání. To se odvíjí od toho, zda je tato volba povolená v nastavení Salamandera. Pokud uživatel povolí mazání, tak se zobrazí dialogové okno s ukazatelem průběhu. Následně jsou zvolené položky mazány za pomoci funkcí `DeleteDirectory()` nebo `sftp_unlink()`. První z nich smaže zvolený adresář, přičemž nejdříve je rekurzivně smazán jeho obsah a poté adresář samotný. `Sftp_unlink()` je funkce knihovny libssh pro smazání zvoleného souboru. Průběh mazání lze také přerušit stisknutím tlačítka Cancel.

Kapitola 7

Testování

Tato kapitola popisuje průběh testování výsledného řešení. Testování se skládalo z testování jednotlivých funkcí pluginu a měření rychlosti přenosu dat. Pro měření byla vytvořena sada testovacích dat. Ta byla přenášena pluginem a existujícími řešeními na vybraný server a zpět. U každého přenosu se zaznamenala jeho doba trvání a výsledky byli porovnány mezi sebou.

Testování funkčnosti pluginu

Ověření funkčnosti pluginu se skládalo z testování jeho jednotlivých funkcí a testování pluginu jako celku. Důraz byl kladen na to, aby nedošlo k nečekanému pádu aplikace a aby případné chyby při provádění operací nezpůsobili ztrátu dat uživatele. U vytváření spojení byli testovány používané algoritmy transportní vrstvy a jednotlivé metody autentizace. Bylo zkoušeno zadávat správné i špatné údaje při autentizaci. Dále se ověřovalo vytváření, úprava a mazání uložených spojení. Kontrolovalo se ukládání položek do registru a jejich formát zápisu. Následně se také zkoušelo, zda se nastavení po spuštění načte a zde lze plugin spustit s implicitním nastavením. Připojovalo se několikrát na stejný server pod stejnými údaji, na několik různých serverů nebo na lokální server.

Po připojení byl zkontrolován výpis vzdálených adresářů a souborů včetně skrytých položek. Procházelo se mezi adresáři v různých částech systému, aby se ověřilo zpracování cest. Vytvářeli se nové adresáře, přesouvali se do nich soubory již uložené na serveru. Měnili se položkám práva a jména, přičemž se pozorovalo, zda se změny do systému promítnou a následně správně zobrazí. Testovalo se mazání souborů a adresářů, zkoušeli se například mazat soubory, které nepatřili uživatelovi nebo několika úrovňové adresáře. U přenosů bylo ověřeno, zda se soubory přenáší celé a bez chyb. Dále se testoval přenos přes SFTP i SCP, jak nad soubory, tak nad adresáři. Kopírovalo se jak na lokální disk, tak v rámci systému a také se ověřovala funkčnost přenášení položek mezi různými servery.

Měření rychlosti přenosu

Testovací data tvoří adresáře a soubory různých velikostí. Z nich bylo vytvořeno 8 testů, přičemž jeden test tvořili 2 přenosy a výsledný čas je průměr obou hodnot. Výchozí srovnávací hodnota byla získaná z programu WinSCP za pomoci logování tohoto programu. Jednotlivé záznamy mají totiž i časovou značku, a tak je možné zjistit, kdy přenos začal a skončil. U pluginu se měření provádělo na základě rozdílu výstupu systémové funkce `GetTickCount()`,

Soubory	Adresáře	Celková velikost	WinSCP	Psftp	Sftp	Plugin
1	0	1MB	8.67s	9.71s	9.85s	9.14s
1	0	10MB	90.17s	91.6s	91.02s	86.92s
10	0	20MB	186.2s	181.54s	182.87s	181.89s
100	0	5MB	46.63s	48.44s	47.35s	45.78s
10	1	5MB	45.95s	48.99s	48.44s	45.83s
15	5	7.5MB	70.44s	71.08s	72.54s	66.172s
100	25	5MB	47.92s	54.88s	54.02s	54.51s
10	20	20MB	181.09s	187.1s	184.62s	177.89s

Tabulka 7.1: Výsledky měření rychlosti přenosu pomocí protokolu SFTP

Soubory	Adresáře	Celková velikost	WinSCP	Pscp	Scp	Plugin
1	0	1MB	9.178s	10.26s	9.96s	9.45s
1	0	10MB	90.12s	92.14s	91.88s	90.16s
10	0	20MB	178.36s	180.92s	181.25s	176.19s
100	0	5MB	44.53s	46.98s	45.72s	43.13s
10	1	5MB	46.06s	46.22s	45.44s	44.31s
15	5	7.5MB	67.03s	69.22s	68.8s	65.38s
100	25	5MB	47.93s	51.41s	50.33s	48.27s
10	20	20MB	179.2s	185.17s	184.45s	180.23s

Tabulka 7.2: Výsledky měření rychlosti přenosu pomocí protokolu SCP

která vrací počet milisekund od zapnutí systému. Přenosy u ostatních implementací byli prováděny v prostředí Windows Powershell a jeho příkazu `Measure-Command`. Ten měří dobu běhu programu nebo zvoleného příkazu. Přestože je do tohoto měření započítáno i vytvoření spojení a autentizace, tak se tyto zjištěné hodnoty dají považovat za validní, jelikož dané operace zabírali pouze minimální část z celkové doby trvání příkazu. Nebylo například potřeba zadávat heslo během provádění autentizace, protože jej aplikace přijali jako argument při spuštění. Pokud byla při měření zjištěna odchylka větší než 15% od srovnávací hodnoty, tak se test opakoval. Jestliže byl výsledek i poté odlišný, bylo provedeno další přeměření a vybrala se nejbližší hodnota.

Tabulky 7.1 a 7.2 ukazují výsledky provedených testů. Testování probíhalo u všech přenosů vůči stejnému serveru, na který se připojovalo přes Internet. To mohlo způsobit lehké kolísání rychlosti přenosu, nicméně rozhodně nedošlo k výpadkům spojení. Testovány byli oba přenosové protokoly, přičemž z výsledků je patrné, že vytvořený plugin se v rychlosti přenosu vyrovná již existujícím implementacím pro bezpečný přenos souborů. Důvodem je jednak zvolená knihovna Libssh, která zajišťuje správu nad spojením a zasílání dat, a také vnitřní stavba a rozhraní Salamandera, umožňující jednoduchou identifikaci položek pro stažení. U přenosů měřených za pomocí příkazu `Measure-Command` je také přibližně vidět, jak dlouho probíhal proces připojení a autentizace.

Kapitola 8

Závěr

Cílem práce bylo vytvořit rozšíření pro program Altap Salamander, které umožní bezpečný přenos souborů po síti za pomoci protokolu SCP. Nejdříve bylo potřeba seznámit se stavbou a specifikacemi protokolu SSH, protože ten zajišťuje vytvoření a správu zabezpečeného spojení a následně také s přenosovými protokoly SFTP a SCP, pro porozumění principů jejich fungování. Dále bylo třeba pochopit, jak funguje program samotný a jak postupovat při vytváření rozšíření. Poté bylo vybráno několik existující řešení pro přenos souborů. U nich se zkoumali vlastnosti a funkce použitých protokolů. Na jejich základě se nakonec vytvořil návrh pluginu s jeho požadovanými vlastnostmi.

Při implementaci se využilo nástrojové sady Altap Salamander SDK, která kromě dokumentace rozhraní programu obsahuje také vytvořené příklady rozšíření, ze kterých lze při tvorbě vycházet. Pro tvorbu zabezpečeného spojení byla vybrána knihovna Libssh. Hlavním důvodem bylo to, že navíc podporuje jak přenosový protokol SCP, tak SFTP. To také umožnilo vytvořit plugin, který funguje jako vzdálený souborový systém a umožňuje nejen přenos souborů, ale také jejich správu. Plugin pro zajištění těchto funkcí používá jako hlavní protokol SFTP, přičemž pro přenos lze využít i SCP. Díky tomu lze však soubory a adresáře na serveru i přesouvat, přejmenovávat nebo mazat. Dále je také možné připojit se na několik různých serverů a přenášet data z jednoho na druhý. Při vytváření spojení lze upravovat jeho parametry a případně si uložit přihlašovací údaje k vybranému serveru pro urychlení tvorby následujících spojení.

Výsledkem je tedy funkční plugin, který lze využít k přenášení souborů a jehož rychlost přenosu nezaostává za jinými řešeními. Pluginu nicméně chybí některé funkce, které právě ostatní implementace podporují. Jedná se hlavně o pokračování ve zrušeném přenosu, spouštění vzdálených příkazů nebo ukládání obsahu již načtených adresářů pro urychlení jejich výpisu. Tyto funkce by se dali přidat během následujících aktualizací pluginu. Bylo by však nutné pro ně navrhnout vhodnou vnitřní reprezentaci a případné ovládací a zobrazovací prvky. Jako další vylepšení by se do pluginu mohla přidat funkce pro úpravu vzdálených souborů. Problém je v tom, že Salamander pro ni zatím neposkytuje rozhraní, a tak by bylo nutné využít některé jiné rozhraní nebo vymyslet jiný způsob, jak by byla funkce prováděna.

Literatura

[1] ALTAP, s. s. r.: *Altap Salamander - Výkonný dvoupanelový správce souborů. [Online; navštíveno 10.04.2019].*
URL https://www.altap.cz/cz/

[2] ALTAP, s. s. r.: Altap Salamander Help: Getting Started with WinSCP Plugin. [Online; navštíveno 10.04.2019].
URL https://www.altap.cz/salamander/help/winscp/introduction_intro/

[3] ALTAP, s. s. r.: Altap Salamander Help: Using Plugins. [Online; navštíveno 10.04.2019].
URL https://www.altap.cz/salamander/help/salamand/plugins_using/

[4] ALTAP, s. s. r.: O společnosti Altap. [Online; navštíveno 10.04.2019].
URL https://www.altap.cz/cz/about/

[5] ALTAP, s. s. r.: Seznam změn pro Altap Salamander. [Online; navštíveno 10.04.2019].
URL https://www.altap.cz/cz/salamander/changelogs/

[6] Barrett, D. J.; Silverman, R. E.; Byrnes, R. G.: SSH, The Secure Shell: The Definitive Guide. Sebastopol, CA: O'Reilly Media, druhé vydání, 2005, ISBN 0596008953.

[7] Cusack, F.; Forssen, M.: Generic Message Exchange Authentication for the Secure Shell Protocol (SSH). RFC 4256, RFC Editor, January 2006, doi:10.17487/RFC4256, [Online; navštíveno 10.01.2019].
URL http://www.rfc-editor.org/rfc/rfc4256.txt

[8] Hutzelman, J.; Salowey, J.; Galbraith, J.; aj.: Generic Security Service Application Program Interface (GSS-API) Authentication and Key Exchange for the Secure Shell (SSH) Protocol. RFC 4462, RFC Editor, May 2006, doi:10.17487/RFC4462, [Online; navštíveno 10.01.2019].
URL http://www.rfc-editor.org/rfc/rfc4462.txt

[9] Pechanec, J.: How the SCP protocol works. [Online; navštíveno 10.01.2019].
URL https://web.archive.org/web/20170215184048/https://blogs.oracle.com/janp/entry/how_the_scp_protocol_works

[10] Project, T. O.: OpenSSH: Features. [Online; navštíveno 10.04.2019].
URL https://www.openssh.com/features.html

[11] *Project, T. O.: OpenSSH: Project History. [Online; navštíveno 10.04.2019].*
URL https://www.openssh.com/history.html

[12] *Project, T. O.: sftp(1) - OpenBSD manual pages. [Online; navštíveno 10.04.2019].*
URL https://man.openbsd.org/sftp.1

[13] *Přikryl, M.: Project History :: WinSCP. [Online; navštíveno 10.04.2019].*
URL https://winscp.net/eng/docs/project_history

[14] *Přikryl, M.: Table of Contents :: WinSCP. [Online; navštíveno 10.04.2019].*
URL https://winscp.net/eng/docs/start

[15] *SSH Communications Security, I.: SSHD_CONFIG - SSH SERVER CONFIGURATION. [Online; navštíveno 10.01.2019].*
URL https://www.ssh.com/ssh/sshd_config/

[16] *Tatham, S.: PuTTY Change Log. [Online; navštíveno 10.04.2019].*
URL https://www.chiark.greenend.org.uk/~sgtatham/putty/changes.html

[17] *Tatham, S.: Using PSFTP to transfer files securely. [Online; navštíveno 10.04.2019].*
URL https://the.earth.li/~sgtatham/putty/0.71/htmldoc/Chapter6.html

[18] *Ylonen, T.; Lehtinen, S.: SSH File Transfer Protocol. Internet-Draft draft-ietf-secsh-filexfer-02.txt, IETF Secretariat, October 2001, [Online; navštíveno 10.01.2019].*
URL https://tools.ietf.org/id/draft-ietf-secsh-filexfer-02.txt

[19] *Ylonen, T.; Lonvick, C.: The Secure Shell (SSH) Authentication Protocol. RFC 4252, RFC Editor, January 2006, doi:10.17487/RFC4252, [Online; navštíveno 10.01.2019].*
URL http://www.rfc-editor.org/rfc/rfc4252.txt

[20] *Ylonen, T.; Lonvick, C.: The Secure Shell (SSH) Connection Protocol. RFC 4254, RFC Editor, January 2006, doi:10.17487/RFC4254, [Online; navštíveno 10.01.2019].*
URL http://www.rfc-editor.org/rfc/rfc4254.txt

[21] *Ylonen, T.; Lonvick, C.: The Secure Shell (SSH) Protocol Architecture. RFC 4251, RFC Editor, January 2006, doi:10.17487/RFC4251, [Online; navštíveno 10.01.2019].*
URL http://www.rfc-editor.org/rfc/rfc4251.txt

[22] *Ylonen, T.; Lonvick, C.: The Secure Shell (SSH) Transport Layer Protocol. RFC 4253, RFC Editor, January 2006, doi:10.17487/RFC4253, [Online; navštíveno 10.01.2019].*
URL http://www.rfc-editor.org/rfc/rfc4253.txt