Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Bezpečnostný konvertor e-mailov

Bakalárska práca

2017
Peter Vašut

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Bezpečnostný konvertor e-mailov

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	RNDr. Jaroslav Janáček, PhD.

Bratislava, 2017
Peter Vašut

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Peter Vašut
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Bezpečnostný konvertor e-mailov
E-mail Security Converter

Cieľ: E-mail je v súčasnosti jedným z často využívaných kanálov na zanesenie škodlivého softvéru do vnútorných počítačových sietí. Cieľom bakalárskej práce je navrhnúť a implementovať riešenie, ktoré umožní konvertovať prílohy e-mailov vybraných typov do iného formátu s cieľom zamedziť využitie bezpečnostných zraniteľností útočníkom pôvodne predpokladaného softvéru na spracovanie pôvodnej prílohy. Riešenie musí podporovať minimálne konverziu dokumentov bežného kancelárskeho softvéru (doc, docx, xls, xlsx, ppt, pptx, odt, ods, odp, rtf, ...) do PDF, konverziu PDF do PDF, konverziu bežných formátov obrázkov (JPEG, GIF, PNG) do zvoleného formátu pre obrázky. Riešenie by malo pozostávať z SMTP proxy a virtuálneho počítača zabezpečujúceho konverziu, pričom musí byť kladený dôraz na bezpečnosť, keďže komponenty riešenia budú prichádzať do styku s dokumentami obsahujúcimi škodlivý softvér.

Vedúci: RNDr. Jaroslav Janáček, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Dátum zadania: 31.10.2016

Dátum schválenia: 31.10.2016 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Abstrakt

Navrhli a implementovali sme riešenie umožňujúce konverziu vybraných typov príloh, s cieľom zamedziť využitie bezpečnostných zraniteľností. Navrhnuté riešenie pozostáva z virtuálneho počítača, ktorý obsahuje mailový server, nástroje na konverziu dokumentov a originálny program napísaný v jazyku *Python* prepájajúci jednotlivé komponenty.

Kľúčové slová: e-mail, proxy, bezpečnosť

Abstract

We have proposed and implemented technological solution, enabling conversion of selected attachment types to prevent exploitation of security problems. Proposed solution is composed of virtual machine containing mail-server, tools for attachment conversion and original *Python* script for connecting all components.

Keywords: e-mail, proxy, security

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

E-mail je často používaná forma elektronickej komunikácie. Používa sa na súkromné účely, ale aj v školách, firmách alebo štátnych inštitúciách. Keďže mail vie poslať ktokoľvek s prístupom na internet bez súhlasu prijímateľa, počet potenciálnych útočníkov je obrovský. Možnosť posielať rôznorodý obsah zas prináša príležitosť zneužiť rôzne bezpečnostné zraniteľnosti. Škodlivé prílohy môžu pomôcť útočníkovi získať prístup k vnútornej infraštruktúre firmy, odcudziť tajné informácie alebo osobné údaje, prípadne zablokovať činnosť organizácie. Preto je cieľom práce implementovať riešenie, ktoré takéto útoky značne sťaží, respektíve za bežných okolností úplne znemožní.

Kapitola 1

Úvod k e-mailu

E-mail je v súčastnosti často využívaná forma elektronickej komunikácie. Dá sa povedať, že z veľkej časti nahrádza správy posielané poštou (napríklad pohľadnice). V tejto kapitole sa oboznámime s históriou a základmi fungovania e-mailu. Po prečítaní tejto kapitoly bude čitateľovi zrejmé, že prirovnanie k pohľadnici je pomerne vhodné, aspoň čo sa týka bezpečnosti a dôvernosti podanej informácie.

1.1 História a štandardy

Prvé náznaky (moderného) mailu prišli už v sedemdesiatych rokoch dvadsiateho storočia. Napríklad štandard RFC 561 [2] z roku 1973 obsahuje definované niektoré údaje, ktoré sa v hlavičkách e-mailov vyskytujú dodnes. Ako príklady môžeme uviesť presné zadefinovanie polí `FROM`, `DATE`, `SUBJECT`. Tento štandard bol aktualizovaný v roku 1975 štandardom RFC 680 [9].

Štandard *SMTP* (z anglického `Simple Mail Transfer Protocol`) používaný na prenos mailov bol definovaný v RFC 821 [11] v roku 1982. K tomuto štandardu pribudlo mnoho ďalších aktualizácií (napríklad *DRAFT STANDARD RFC 5321* [6]). Najnovšia navrhovaná aktualizácia (so statusom *PROPOSED STANDARD*) je RFC 7504 [7] z roku 2015, čo možno považovať za aktuálny vývoj.

Základy pre samotný formát správy položil štandard RFC 822 [3] z roku 1982. Definuje napríklad známe polia `TO`, `CC` a `BCC`, s ktorými sa stretáva používateľ pri tvorbe mailových správ. Aj tento štandard bol ďalej aktualizovaný. Spomeňme RFC 2822 (*PROPOSED STANDARD*) [12] z roku 2001, RFC 5322 (*DRAFT STANDARD*) [10] z roku 2008 a najnovšiu aktualizáciu z roku 2013 - RFC 6854 (*PROPOSED STANDARD*) [8].

1.2 Základný popis fungovania e-mailu

Na prenos mailov sa používa protokol *SMTP* (z anglického `Simple Mail Transfer` Protocol). Tento protokol je zodpovedný za prenos takzvaných `mailových objektov`.

Mailový objekt (*mail object*) sa skladá z *obálky* a *obsahu*. Samotný štandard umožňuje prenášať iba znaky kódované pomocou *US ASCII*. Avšak používatelia e-mailu sú pravdepodobne zvyknutí na to, že správy, ktoré odosielajú a prijímajú, sú často formátované pomocou *html*, obsahujú znaky z *UTF-8* sady, prípadne obsahujú prílohy v rôznych formátoch (napríklad *PDF*, rôzne typy obrázkov, textové dokumenty, ...). Toto je docielené pomocou rozšírení *MIME* [4].

Systém zodpovedný za prebratie správy od používateľa sa označuje skratkou *MUA* (z anglického ***M**ail **U**ser **A**gent*). Servery a klienti poskytujúci prenos mailových správ sa označujú skratkou *MTA* (z anglického ***M**ail **T**ransfer **A**gent*). Niekedy sa označenie *MUA* nepoužíva, namiesto neho sa použije *MTA*. V praxi sa zvyčajne používajú len dva *MTA* (jeden na strane odosielateľa a druhý u príjemcu), avšak po ceste ich môže byť viac, niektoré dokonca v navzájom neprístupných sieťach (napríklad oddelených *firewallom*). V štandarde a anglickej terminológií sa toto *predávanie správ medzi sieťami* označuje ako *SMTP relaying*.[1] Jednotlivé systémy si postupne predávajú zodpovednosť za úspešné doručenie správy do cieľa (alebo informovanie odosielateľa o chybe).

Samotný protokol potrebuje pre svoje fungovanie službu spoľahlivého usporiadaného prúdu bajtov. Využíva sa najmä protokol *TCP*. Komunikácia prebieha formou výmeny správ. Klient a server sa pri posielaní *požiadaviek* a *odpovedí* striedajú (pokiaľ nie je nejakým rozšírením dohodnuté inak). Požiadavky zväčša pozostávajú zo slova tvoreného písmenami anglickej abecedy, pričom za medzerou môžu nasledovať parametre. Odpovede sú tvorené trojciferným kódom zapísaným ako *ASCII* znaky, za ktorým môže nasledovať slovný popis alebo ďalšie informácie. Tvorcovia protokolu dbali na jednoduchosť, takže jednotlivé správy sa dajú jednoducho čítať a ručne písať, čo môže pomôcť napríklad pri diagnostike chýb.

Pre určenie cieľa mailu sa používajú *mailové adresy*. Skladajú sa z *lokálnej časti* (*local part*) a *domény* (*domain*). Pre doménovú časť platia podobné pravidlá ako pre záznamy v systéme *DNS*, napríklad sa nerozlišujú veľké a malé písmená. Lokálna časť by nemala byť interpretovaná zo sémantického hľadiska žiadnym *MTA* okrem cieľového.

[1] Možnosť umiestniť medzi odosielateľa a príjemcu *MTA* navyše je dôležitá pre fungovanie systému, ktorý je navrhnutý v tejto práci.

Kapitola 2

Problém a porovnanie riešení

V tejto kapitole bližšie rozoberieme problém, ktorý budeme riešiť. Ďalej uvedieme výhody a nevýhody rôznych možných riešení, čím zároveň ukážeme relevantnosť ponúknutého riešenia.

2.1 Bezpečnosť príloh a motivácia

Ako už bolo uvedené v sekcii 1.2, samotný protokol *SMTP* je pomerne jednoduchý. Okrem toho, postupný vývoj e-mailu trvá už niekoľko desaťročí, takže jeho súčasti sú dobre formalizované. Avšak *dáta* obsiahnuté v maili môžu byť mimoriadne rôznorodého charakteru. Rôzne druhy príloh sú spracovávané rôznymi aplikačnými softvérmi, pričom každý z nich môže obsahovať vlastné bezpečnostné zraniteľnosti (pretečenie v pamäti, nežiadúce získanie privilégií - napríklad cez makrá v dokumentoch, nedostatočná kontrola vstupu, ...). Útočník môže posielať e-mailom ľubovoľný obsah. V lepšom prípade môže pokus o zneužitie chyby spôsobiť neštandardné správanie aplikácie, napríklad náhle ukončenie činnosti, alebo úpravu dokumentu podľa údajov z počítača obete. V horšom prípade môže byť útočník schopný spustiť ľubovoľný kód na počítači obete.

Rôzne zraniteľnosti predstavujú komplikácie najmä pre inštitúcie a firmy, u ktorých je bezpečnosť nadštandardne dôležitá, napríklad rôzne štátne inštitúcie, prípadne firmy narábajúce s osobnými údajmi. Nie je úplne neočakávané, že práve pracovníci v takýchto firmách často potrebujú prijímať maily z externého prostredia (napríklad od klientov), čo situáciu značne komplikuje. Ešte väčším problémom je, že e-mail je jedna z mála technológií v takýchto spoločnostiach, s ktorými prichádzajú do styku pracovníci, ktorí nemajú vzdelanie v oblasti informačnej bezpečnosti alebo informačných technológií všeobecne, napríklad úradníci, asistenti, referenti a podobne. Problém môže spôsobiť, keď takýto pracovník dôveruje zobrazeným informáciám ako obsah správy alebo odosielateľ. Preto je vývoj a štúdium prostriedkov týkajúcich sa bezpečnosti

e-mailovej komunikácie mimoriadne dôležitý.

2.2 Možnosti riešenia

2.2.1 Riešenia na strane klienta

Bezpečnosť príloh sa dá riešiť rôznymi riešeniami na strane klienta. Jedným z nich je nasadenie antivírusového riešenia. Antivírusový softvér môže pomôcť pri niektorých druhoch hrozieb. Napríklad ak by dokument obsahoval nejaký známy škodlivý kód, antivírusový program by ho mohol odhaliť. Alternatívnym riešením by bolo sledovať rôzne nečakané operácie v systéme. Existujú napríklad riešenia upozorňujúce užívateľa v prípade, že chce nedôveryhodná aplikácia pristupovať na sieť, zapisovať do systémových priečinkov alebo používať systémové volania považované za nebezpečné. Prípadne je možné rôzne kontroly podľa potreby implementovať. Avšak takéto systémy sa zväčša spoliehajú na istú dávku „šťastia", a už z princípu nemôžu fungovať stopercentne. Napríklad útočník s dostatkom zdrojov a informácií si vie vopred otestovať, či jeho unikátne navrhnutý útok bude na cieľovú infraštruktúru účinný.[1] Preto riešenia na strane klienta môžu byť jednou z posledných vrstiev obrany, avšak samé o sebe nie sú dostatočné.

2.2.2 Riešenia formou proxy serveru

Pre nedostatky riešení v časti 2.2.1 navrhujeme principiálne iný prístup (ktorý samozrejme môže byť použitý súčasne s riešeniami v časti 2.2.1).

Prvou zmenou je umiestnenie bezpečnostného riešenia na samostatný server. Stačí ho teda spravovať iba na jednom mieste, čo môže byť prínosné najmä pre väčšie organizácie. Bežný používateľ nemôže zasahovať do konfigurácie - dokonca nemá ani fyzický prístup k serveru zabezpečujúcemu bezpečnosť mailov.

Ďalšia, o dosť zásadnejšia zmena, je, že návrhy uvedené nižšie sa nebudú pokúšať detegovať hrozbu, ale rovno odstránia pôvodnú správu a vytvoria úplne novú, pričom nová správa bude obsahovať skonvertované náhľady pôvodných príloh bez akéhokoľvek aktívneho obsahu (makrá . . .), bez ohľadu na to, či ide o obsah považovaný za bezpečný alebo nie. Problém informačných zraniteľností je, že sa často na ne nepríde, kým ich niekto dostatočne očividne nezneužije. Všeobecne môžeme predpokladať, že keď vývojári aplikačného softvéru dokážu spraviť chyby, rovnako sa môžu vyskytnúť chyby

[1] Napríklad pomocou webu `http://virustotal.com`.

aj pri odhaľovaní nebezpečného obsahu. Preto je na mieste považovať všetok obsah za nebezpečný a náležite tomu s ním manipulovať.

Všetky riešenia uvedené nižšie používajú virtuálne počítače, vnútri ktorých sa bude vykonávať konverzia. Stav virtuálnych počítačov bude pravidelne obnovovaný do predefinovaného bezpečného stavu, takže aj v prípade, že by bol systém na virtuálnom stroji kompromitovaný, hrozba bude jednoducho automaticky odstránená.

Virtuálny počítač s bitmapou ako výstupom Mailový proxy server pošle prílohy e-mailu na virtuálny počítač. Ten ich skonvertuje na bitmapu, ktorú čo najjednoduchším kanálom prenesie naspäť proxy serveru. Keďže z virtuálneho počítača sú povolené spojenia len na určený port proxy servera, a ten všetky dáta považuje za pixely obrazu, toto riešenie je možné považovať za relatívne bezpečné. Avšak prináša aj isté nevýhody. Často sa totiž v mailoch ako prílohy vyskytujú rôzne formy textových dokumentov. Keďže jedinou formou výstupu je obraz, výsledné maily by boli oproti originálom zbytočne veľké.[2] Veľkosť prílohy by sa dala znížiť menším rozlíšením alebo použitím kompresie. Avšak ideálne by bolo použiť kompresiu až na strane proxy servera, aby sme mohli zachovať jednoduchosť komunikačného kanálu. To ale znamená, že spracovanie mailov by nebolo dostatočne efektívne.

Viacero zreťazených virtuálnych počítačov Toto riešenie sa od predošlého líši tým, že medzi odovzdaním príloh mailovým proxy a prijatím skonvertovaných dát sa nachádza viacero virtuálnych počítačov, z ktorých každý vykonáva iný druh konverzie (napríklad pôvodný dokument $\rightarrow$ *PS* $\rightarrow$ *PDF*). Virtuálny počítač môže komunikovať len s príslušným nasledujúcim virtuálnym počítačom (alebo v prípade posledného s mailovým proxy serverom). Proxy server by v tomto prípade dostal prílohu skonvertovanú priamo do formátu *PDF*. Útočník by musel vytvoriť dokument formátu 1, pomocou zraniteľnosti konvertoru v prvom virtuálnom počítači tento počítač ovládnuť, následne prinútiť kompromitovaný systém odoslať do druhého virtuálneho počítača súbor formátu 2, ktorý by dokázal využiť zraniteľnosť v tomto počítači, a následne odoslať infikovaný súbor proxy serveru. Takýto útok je očividne v praxi pomerne ťažko realizovateľný. Okrem toho je možné takúto reťaz virtuálnych počítačov ľubovoľne predlžovať a na každom používať rôzny softvér. Tým vieme znížiť pravdepodobnosť útoku takmer na nulu. Problémom je akurát zbytočne veľká zložitosť riešenia a veľké požiadavky na výkon pre beh viacerých virtuálnych počítačov.

Jeden virtuálny počítač Predošlé riešenie sa dá zjednodušiť použitím len jedného virtuálneho počítača, ktorý by priamo posielal proxy serveru prílohu vo formáte *PDF*.

[2]Napríklad 1 A4 pri rozlíšení 300 dpi a farebnej hĺbke 24 bitov by zaberala viac ako 24 MB.

Toto riešenie síce neposkytuje takú mieru bezpečnosti ako predošlé dve, ale je rozumným kompromisom medzi bezpečnosťou a jednoduchosťou riešenia. V praxi by útočník aj tak musel vyvinúť netriviálne úsilie a musel by sa zameriavať špecificky na tento systém. Bežný útočník útočiaci hromadne na veľké množstvo cieľov prakticky nemá šancu náhodne vykonať úspešný útok na takýto systém.[3]

[3]Samozrejme riešime len technickú stránku e-mailov. Existujú iné spôsoby ako útočiť na cieľ.

Kapitola 3

Implementačné detaily

V tejto kapitole sa budeme podrobnejšie venovať implementačným detailom riešenia a postupne rozoberieme dôvody použitia príslušných častí.

3.1 Časti riešenia

Výsledné riešenie sa skladá z viacerých komponentov. Pri väčšine častí bolo možné nájsť softvér dostatočne spĺňajúci požiadavky. Samozrejme, špecializované časti bolo nutné implementovať. Základné úlohy systému sú prijať správu, spracovať všetky prílohy príslušnými nástrojmi a odoslať filtrovanú správu ďalej. Tieto úlohy sa samostatne vyskytujú aj v bežnej „používateľskej" praxi, čo značne zjednodušuje ich vykonávanie, napríklad vďaka rozšírenosti implementácií. Na druhú stranu, zorientovanie sa medzi ponúkanými možnosťami a výber správnej podmnožiny možností (vzhľadom na jednoduchosť a funkčnosť výsledného riešenia) je, ako sa ukázalo, už menej triviálny problém.

3.2 Prijímanie mailov

Podľa špecifikácie protokolu *SMTP* ([11] prípadne [6]) má mailový server viacero úloh, ktorých vykonávanie musia tvorcovia implementácií zabezpečiť. Samotné prijatie mailu (v prípade, že by neboli použité žiadne rozšírenia alebo autentifikácia) nie je extrémne zložité. Komunikácia prebieha pomocou jednoduchých textových správ. Napriek tomu, server musí zabezpečiť mnoho ďalších úloh, napríklad opätovné preposielanie správy v prípade, že je ďalší server na ceste nedostupný, prípadne informovať odosielateľa o probléme s doručením správy. S pridávaním úloh rastie aj komplexnosť mailového serveru, a teda aj možnosť na nejakú úlohu zabudnúť, prípadne spraviť chybu pri implementácií. Navyše, v praxi nemôžeme vylúčiť použitie rozšírení a autentifikácie. Keďže sa zodpovednosť za správu prenáša z jedného serveru na ďalší, chyba v našej implementácií, ktorá by zabránila preposielaniu správ, by bola fatálna.

Pre prijímanie správ sme sa preto rozhodli použiť mailový server *Postfix*.[1] Prináša to mnoho výhod.

Vyhneme sa množstvu bezpečnostných a iných chýb, ktoré sú súčasťou prakticky každého väčšieho projektu. Samozrejme, žiaden softvér nie je bez chyby, ale dá sa predpokladať, že desiatky rokov používania majú istý pozitívny dopad na bezpečnosť a stabilitu softvéru.

Ďalšou výhodou je, že *Postfix* dokáže fungovať samostatne.[2] Je to výhodné najmä v prípade nasadenia výsledného riešenia v praxi, keďže stačí, aby správca systému vedel obsluhovať *Postfix*, a potom vie nakonfigurovať zároveň interné mailové servery aj všetky filtračné uzly podľa potreby. Toto je dôležité, keďže filtračných serverov môže byť viac, a môžu byť rôzne nakonfigurované a prepojené podľa špecifických potrieb a prostriedkov danej organizácie. Keďže ide o samostatný program, je možné ho v budúcnosti bez väčších problémov nahradiť iným.

Postfix je sám o sebe vyspelý nástroj, ktorý sa dá dobre nastaviť a prispôsobiť podľa špecifických potrieb. Môže to napomôcť nasadeniu riešenia aj mimo akademickej sféry.[3]

3.3 Konverzia dokumentov

Na konverziu dokumentov používame štandardne dostupné nástroje. Dôvody použitia existujúcich nástrojov sú podobné ako v časti 3.2. Rozdiel je najmä v dôležitosti jednotlivých dôvodov.

V prípade, že by bol komponent prijímajúci maily nasadený mimo virtuálneho počítača, sú bezpečnostné chyby pri konverzii dokumentu menej fatálne v porovnaní s chybami tohto komponentu. Avšak v prípade, že je konverzia dokumentov vykonávaná v tom istom virtuálnom počítači ako prijímanie mailov, riziko potenciálnej zraniteľnosti je takmer ekvivalentné. Výnimkou môžu byť napríklad špecifické chyby mailového serveru, ktoré by znefunkčnili prijímanie správ. Samotná neschopnosť skonvertovať prílohu nie je kritický problém. Obmedzí to iba dostupnosť danej prílohy pre príjemcu. V prípade útoku príjemcovi príloha pravdepodobne nebude chýbať. Pri legitímnej správe prípadná náhodná chyba spôsobí prinajhoršom drobnú nepríjemnosť pre

[1] Viac informácií o *Postfix*-e sa dá dozvedieť na web-stránke projektu [1].

[2] V zmysle *na rozdiel od knižníc alebo rôznych zásuvných modulov.*

[3] Predsa len, bezpečnostné riešenie, ktoré nebude nikde nasadené, nezabráni žiadnemu útoku.

príjemcu. Keďže hlavným cieľom tejto práce je zabraňovať útokom, drobná komplikácia pre príjemcu v hypotetickej a nepravdepodobnej situácii je ospravedlniteľná.[4]

Ďalej v krátkosti rozoberme podporované typy dokumentov a konkrétne vybraté nástroje.

3.3.1 Dokumenty kancelárskych balíkov

Ako bolo spomenuté v časti 2.1, očakávame nasadenie v prostredí pripomínajúcom kancelárske prostredie. Zjavne teda bude funkcionalita prijímania kancelárskych dokumentov vyžadovaná. To prináša ale viacero komplikácií, keďže dokumenty sú rôznych typov a formátov. Minimálne zo začiatku sa to javilo ako pracne riešiteľný problém.

Zjavným kandidátom bolo použitie kancelárskeho balíka *LibreOffice*, najmä kvôli jeho dostupnosti vďaka otvorenému zdrojovému kódu a licencii umožňujúcej bezplatné použitie. Podporuje viacero formátov dokumentov, napríklad *odt* pre obohatené textové dokumenty, *ods* pre tabuľky a *odp* pre prezentácie. Veľkou výhodou je aj podpora bežne používaných formátov dokumentov vytváraných pomocou *Microsoft Office* - *doc*, *docx*, *xls*, *xlsx*, *ppt*, *pptx*.

Prirodzenou obavou je, či budú skonvertované dokumenty dostatočne verne zodpovedať originálom. Po krátkom teste na rôznych dokumentoch sa ukázalo, že konverzia funguje prekvapivo dobre. Problém môže nastať s tabuľkami. Pri konverzií tabuľky do *PDF* sa príliš široký hárok rozdelí na viacero strán. Väčší problém je, že s tabuľkami je nutné často ďalej pracovať. Preto je nutné pri nasadení riešenia tento problém brať do úvahy, a zvážiť, či nie je výhodnejšie konvertovať tabuľkové dokumenty radšej do formátu *csv*. V takomto prípade sa nezachovajú vzorce v bunkách. Taktiež je možné zvážiť konverziu do iného formátu dokumentov.

LibreOffice sa štandardne používa s grafickým rozhraním. Problém by bol, ak by sa *LibreOffice* nedal zakomponovať do skriptu. Ako alternatíva sa ponúkal napríklad program *unoconv*, ktorý je dostupný v *Debian* repozitári. Z manuálovej stránky sa však dozvieme, že tento program vlastne používa *LibreOffice*. Používať práve *LibreOffice* dáva zmysel, keďže ide o pomerne známy balík s aktívnymi vývojármi.

[4]Predsa len, naše riešenie je určené pre špecifické prípady nasadenia. Správcovi infraštruktúry nič nebráni použiť nejaký proxy server, ktorý bude len pasívne kontrolovať komunikáciu, v prípade, že je to pre danú organizáciu dostatočné.

Ukázalo sa, že *LibreOffice* ponúka možnosť konverzie aj pomocou príkazového riadku. Preto je rozumné preskočiť ďalšie medzičlánky a použiť priamo *LibreOffice*. Skonvertovať dokument do formátu *PDF* je možné nasledovným príkazom:

```
libreoffice --headless --convert-to pdf subor.ods
```

Obdobne je možné konvertovať dokument napríklad do formátu *csv*.

3.3.2 Obrázky

Obrázkov je viacero druhov (*jpg*, *png*, *bmp*, *tiff*, a mnoho ďalších). Je vhodné použiť jednotné riešenie pre všetky bežne používané druhy obrázkov. Ako vhodný sa ukazuje softvérový balík *imagemagic* dodávaný v repozitároch *Linux Mint*u. Podporuje všetky z vyššie uvedených formátov súborov a mnohé ďalšie. Výhodou je, že nie je nutné prispôsobovať príkaz typu vstupného obrázku, stačí zadať názov súboru. Keďže obrázky majú spoločný hlavný typ *MIME* (`image`), písanie samotného pravidla je jednoduché.

V priloženom programe, pre ďalšiu inšpiráciu, okrem konvertovania obrázku meníme jeho veľkosť (ak je príliš veľký). Výsledný príkaz na konverziu do formátu *png* môže vyzerať napríklad nasledovne:

```
convert -resize 1024x1024 vstup.JPG vstup.JPG.png
```

3.3.3 PDF

Príloha formátu *PDF* je najskôr skonvertovaná do formátu *postscript* (*ps*), a následne je tento súbor skonvertovaný naspäť do formátu *PDF*. Tu sa ukazuje možnosť meniť zdrojový kód ako mimoriadne výhodná. Nič totiž nebráni použiť pri jednom pravidle dve konverzie za sebou.

Pre praktickosť používame programy *pdftops* a *ps2pdf* ktoré sú distribuované spolu s operačným systémom *Linux Mint*. Uveďme ešte možný príklad použitia týchto príkazov:

```
pdftops subor.pdf subor.pdf.ps
ps2pdf subor.pdf.ps subor.pdf.ps.pdf
```

3.4 Obslužná logika

Pri vyššie uvedených komponentoch stačilo nájsť najvhodnejšiu existujúcu implementáciu. Avšak podstatnou časťou je aj skombinovať všetky časti dohromady. Cieľom

je mať účinné, ale jednoducho modifikovateľné riešenie. V prípade pridania alebo výmeny komponentov musí byť jednoduché upraviť aj program, ktorý tieto komponenty obsluhuje.

3.4.1 Použitý programovací jazyk a knižnice

Hlavný program je napísaný v jazyku *Python 3*. Použitie tohto interpretovaného jazyka prináša viacero výhod.

Používanie dátových štruktúr s vyššou úrovňou abstrakcie zabráni množstvu potenciálnych zraniteľností. Ako príklad môžeme uviesť rozdiel medzi použitím poľa v *C++* a zoznamu v *Python*e. Za istých okolností program pri zápise mimo poľa fixnej dĺžky môže pokračovať ďalej bez akéhokoľvek varovania. Podobné kontroly sú v *Python*e považované za samozrejmosť. Vyhodnocujú sa za behu programu. Samozrejme, vždy je potrebné čítať dokumentáciu k funkciám a štruktúram, avšak ak sú tieto definované rozumne, je to ďalšia vrstva ochrany pred chybami.

K ďalším výhodám patrí jednoduchá syntax a možnosť upravovať kód. Vďaka prehľadnej syntaxi dokáže aj človek neznalý programu pochopiť význam jednotlivých častí programu. Keďže program netreba kompilovať, úpravy a prispôsobenia sú relatívne jednoduché (minimálne na úrovni očakávaných úprav pre prispôsobenie programu špecifickým potrebám).

Nakoniec nesmieme zabudnúť aj na dostupnosť knižníc. Všetky použité knižnice sú štandardne dodávané spolu s interpreterom. To opäť zjednodušuje nasadenie na rôznorodých systémoch.

Uveďme niekoľko príkladov použitých knižníc, ktoré značne zjednodušujú implementáciu. Pre prijímanie a odosielanie mailov sú použité moduly *smtpd* a *smtplib*. Pre prácu s mailovými objektami na vyššej úrovni abstrakcie je použitý modul *email*. Umožňuje objektovo pristupovať k správam a ich častiam. Hrozí tak menšie riziko chyby pri extrahovaní textu a príloh. Na vytváranie a správu dočasných súborov sa ukázal ako vhodný modul *tempfile*. Samozrejme, je možné implementovať vytváranie dočasných súborov aj samostatne, ale vďaka použitiu knižnice netreba zvlášť riešiť výber názvu súboru a jeho vytváranie. Keďže na konverziu sú použité externé programy, použijeme na ich volanie modul *subprocess*. Pre asynchrónne spracovanie mailov modul *asyncore*. Ďalšie, známejšie moduly, netreba ďalej popisovať. Zo spomenutých príkladov je dostatočne zjavné, ako obšírna je štandardná zostava modulov.

Ďalším kritériom na zváženie je rýchlosť. Interpretované jazyky sú často v porovnaní s kompilovanými o poznanie pomalšie. Avšak počítače sú dostatočne rýchle. Okrem toho, väčšinu času zaberie samotná konverzia. Napríklad, bežnému notebooku trvalo skonvertovať dvojstranový dokument zaplnený iba textom 347 milisekúnd. Rovnakému notebooku trvalo spracovať správu bez príloh pomocou *Python*u menej ako milisekundu.

Uveďme ešte ďalší príklad. Majme dokument typu *odt* s textom v rozsahu pol strany, tentokrát aj s nadpismi a rôznym formátovaním. Vyhodnoťme teraz celý priebeh činnosti mailového filtra v jednom behu.

prijatie mailu - vytvorenie dočasného súboru	2 ms
konverzia prílohy	603 ms
zvyšok behu programu	6 ms

Tabuľka 3.1: Časy ukážkového behu programu

Ako je zjavné z tabuľky 3.1, čas konverzie prílohy je o niekoľko rádov väčší ako zvyšný čas behu programu.

3.4.2 Detaily implementácie

Program je podľa obsahu rozdelený do dvoch súborov - `SMTPfilter.py` a `content_types.py`. Hlavný program je v súbore `SMTPfilter.py`. Súbor `content_types.py` je len pomocný súbor, kde sú zadefinované rôzne kategórie príloh. Toto rozdelenie má praktickú funkciu. Keď chce správca systému definovať pravidlá pre prílohy podľa špecifických potrieb, upravuje prvý zo spomenutých súborov. Do druhého stačí pridávať popisy chýbajúcich kategórií príloh, avšak nie všetky musia byť použité. V prípade, že je nutné urýchlene získať prehľad o aktuálne povolených typoch príloh a spôsoboch ich konverzie, štandardným postupom je skontrolovať súbor `SMTPfilter.py`, pričom je možné predpokladať, že v ňom spomenuté pravidlá sú aktuálne aktívne. Samozrejme, správca by mal byť s takýmto postupom oboznámený.

SMTPfilter.py

Samotné spracovanie správ je vykonávané triedou `SMTPfilter`, ktorá dedí od triedy `smtpd.SMTPServer`.

Konštruktor vykonáva funkcionalitu konštruktora triedy `smtpd.SMTPServer`. Okrem toho príjme adresu a port - parametre potrebné pre odoslanie filtrovaného mailu.

Keďže sa konštruktor vykonáva prakticky po spustení serveru, je to vhodné miesto pre zapísanie informácie o začiatku behu filtra do logu. Zároveň v prípade, že je serverov viac, bude v logu zapísaný štart každého filtra. Log obsahuje aj adresu serveru pre zjednodušenie riešenia problémov alebo reakcie na bezpečnostný incident.

Metóda `process_message` zabezpečuje spracovanie po prijatí správy. Informácie o význame parametrov je možné nájsť v dokumentácií k modulu *smtpd*. Každá správa dostane náhodný desaťmiestny alfanumerický identifikátor. Ako vhodný by sa mohol zdať aj skrátený *hash* informácií o správe, avšak viacero rovnakých správ prijatých v rovnakom čase by mohlo spôsobiť problém. Riešenie s náhodným reťazcom je pomerne jednoduché. Navyše, pravdepodobnosť kolízie je pre účely záznamov do logu dostatočne malá (aspoň v realisticky krátkom časovom úseku). Okrem toho kolízie môžu vzniknúť aj pri skrátení *hash*u.

Následne je prijatá správa dekódovaná pomocou funkcie `message_from_string` z modulu `email`, spracovaná filtrom (popísané nižšie) a odoslaná za pomoci modulu `smtplib`. Základné informácie o správe a typoch príloh sú zapísané do logu spolu s identifikátorom správy.

Samotná konverzia a filtrovanie príloh prebieha v metóde `convert_message`. To, či sa správa skladá z viacerých častí, zisťuje funkcia `message.is_multipart`. Ovplyvňuje to návratový typ funkcie `message.get_payload` (konkrétne *string* alebo zoznam správ). V prípade, že je správa zložená z viacerých častí, sa konverzia zavolá rekurzívne na všetky časti, ktoré sú následne spojené do jednej správy (`email.mime.multipart.MIMEMultipart`). V prípade, že sa spracovávaná správa skladá len z jednej časti, prebehne jej konverzia. Samotné dáta je už v tomto prípade možné získať pomocou funkcie `message.get_payload`, ktorú interne volá funkcia `run_external_filter` (popísaná nižšie) vykonávajúca volanie externých nástrojov.

Cieľom funkcie `run_external_filter` je, ako názov napovedá, spúšťať externé programy na konverziu dokumentov. Cieľom je zabezpečiť, aby boli externé programy volané štandardným spôsobom. Umožňuje to jednoduchšie pridanie podpory ďalších typov príloh. Pozrime sa na fungovanie tejto funkcie podrobnejšie.

Argument `command`: Obsahuje zoznam častí príkazu, ktorý spúšťa externý filter. V prípade, že je nastavený argument `shell`, jednotlivé časti budú pospájané medzerami a odovzdané systémovému interpreteru príkazov. V opačnom prípade zoznam popisuje spustený program a jeho argumenty. Použitie zoznamu prináša viacero výhod. Nielenže funkcia *subprocess.run* pracuje so zoznamom ako argumentom, ale do

zoznamu vieme pridať aj iné informácie ako text. Zoznam môže obsahovať aj prvky zadefinovaného typu `Value(Enum)`, ktoré sa vnútri funkcií interne nahradia za *názov originálneho dočasného súboru* (`ORIGINAL_FILE_NAME`), *názov súboru po konverzii* (*CONVERTED_FILE_NAME*) alebo *cestu k dočasným súborom* (`TMP_PATH`).

Argument `f_original_to_new_filename`: Ide o ďalší zaujímavý argument. Očakáva funkciu, ktorá prevedie *názov originálneho dočasného súboru* na *názov súboru po konverzii*. Niektoré nástroje totiž neprepisujú originálny súbor. Keďže názov po konverzii môže závisieť aj od názvu originálneho súboru, aj od použitého príkazu, ukázalo sa použitie funkcie ako vhodný spôsob. Uveďme typický príklad lambda-funkcie, ktorá môže byť použitá:

```
lambda x: x + ".pdf"
```

Popis činnosti: Príloha je uložená do dočasného súboru v určenom priečinku. Ďalej je skonvertovaná príslušným externým programom. Samozrejmosťou je zápis chýb do logu.

Kapitola 4

Popis nasadenia

V tejto kapitole sa bližšie pozrieme na najjednoduchší príklad nasadenia implementovaného riešenia. Cieľom tejto kapitoly nie je podať čitateľovi komplexný návod na konfiguráciu mailového serveru. **Konfigurácia podľa tejto kapitoly nie je určená na reálnu prevádzku.** V praxi bude konfigurácia pravdepodobne prispôsobená požiadavkám danej inštancie. Niektoré možné prispôsobenia budú podľa potreby okrajovo spomenuté. Táto kapitola pomôže pri zostavení minimálnej funkčnej verzie pre demonštráciu výslednej implementácie. Niektoré časti je možné podľa potreby preskočiť a nahradiť vlastnou konfiguráciou.

Poznámka: V tejto kapitole pre jednoduchosť predpokladáme použitie operačného systému *Linux Mint 18 64-bit* nainštalovaný na virtuálnom počítači simulovanom pomocou programu *Virtualbox* vo verzií `5.0.40_Ubuntu r115130`. Samozrejme, nemal by byť problém (s prípadnými drobnými zmenami) použiť aj iný systém. V prípade, že sa podarí útočníkovi zneužiť nejakú zraniteľnosť, napadne virtuálny počítač ktorý je možné pravidelne obnovovať do čistého stavu. Pri praktickom nasadení je vhodné obmedziť sieťovú komunikáciu virtuálneho počítača na minimum tak, aby toto obmedzenie nebolo možné zvnútra obísť.

4.1 Konfigurácia *Postfixu*

Inštalácia *Postfix*u z repozitárov je pomerne priamočiara:

```
sudo apt install postfix
```

Následne stačí zodpovedať na niekoľko dotazov počas inštalácie. Konkrétne nastaviť typ konfigurácie „`Internet site`" a názov mailového serveru. V našom prípade použijeme „`VirtualBox`" ako názov mailového serveru, keďže neplánujeme používať reálnu doménu. Takýto postup vytvorí základnú konfiguráciu.

Pre nastavenie komunikácie stačí pridať nasledujúce riadky do konfiguračných súborov:
/etc/postfix/master.cf

```
scan unix - - n - 10 smtp -o smtp_send_xforward_command=yes -o ···
  disable_mime_output_conversion=yes
localhost:10026 inet n - n - 10 smtpd -o content_filter= -o ···
  receive_override_options=no_unknown_recipient_checks,···
  no_header_body_checks,no_milters -o smtpd_authorized_xforward_hosts···
  =127.0.0.0/8
```

/etc/postfix/main.cf

```
content_filter = scan:127.0.0.1:10025
receive_override_options = no_address_mappings
```

Postfix teda pošle mail filtru na porte 10025. Filter vráti mail na port 10026.

Samotná konfigurácia súborov `/etc/postfix/master.cf` a `/etc/postfix/main.cf` je inšpirovaná projektom, ktorý má za cieľ umožniť filtrovanie mailov pomocou *Python*u a poskytuje práve základnú kostru demonštrujúcu výmenu správ [5].

4.2 Spustenie programu

V prípade, že je *Postfix* nakonfigurovaný podľa časti 4.1, stačí program spustiť v stave, v akom je:

```
python3 SMTPfilter.py
```

Po úspešnom spustení by sa malo objaviť iba hlásenie o štarte.

Pre praktické použitie odporúčame zmeniť niektoré nakonfigurované vlastnosti, najmä cestu, kam sa majú ukladať dočasné súbory, adresu pre odchádzajúce správy, porty pre prichádzajúce a odchádzajúce správy. Nastavením portov a adries sa dajú docieliť rôzne spôsoby zreťazenia virtuálnych počítačov alebo je možné použiť len jeden virtuálny počítač (viď. časť 2.2.2). Taktiež je možné pridať alebo odstrániť pravidlá pre konverziu. (Konkrétne príklady pre prispôsobenie sa dajú nájsť v časti 4.3.)

Je pravdepodobne očakávané, že skript bude spustený po štarte, a v prípade, že nastane neočakávaná výnimka, mal by byť znovu spustený. To sa dá docieliť rôznymi spô-

sobmi. Automatické spúšťanie po chybe sa dá zabezpečiť vytvorením krátkeho skriptu, ktorý bude v cykle spúšťať program. Tento skript je možné následne automaticky volať pomocou systému *cron* tak, aby sa spúšťal vždy po reštarte. Elegantnejšie je použiť inicializačný program a spustiť server v nejakom štádiu štartu systému. Tým sa vyrieši aj prípadný neočakávaný reštart.

4.3 Prispôsobenie programu

Ako už bolo spomenuté, program je možné podľa potreby prispôsobovať. Je nutné prispôsobiť priamo samotné súbory. Je na to hneď niekoľko dôvodov. Nie je očakávané, že bude nutné vykonávať prispôsobenie pravidelne. Okrem toho, zmeny môžu byť pomerne veľkého rozsahu - nad rámec drobných konfiguračných zmien. Použitie *Python*u takéto priame zásahy do kódu podstatne uľahčuje.

Základným nastavením sú čísla portov a adresy. Tie sa zadávajú ako parametre pri vytváraní samotného objektu filtra:

```
server = SMTPfilter(('127.0.0.1', 10025), ("localhost", 10026), ···
  tmp_path="/tmp")
```

Prvý parameter je adresa a port, na ktorých má konvertor počúvať. Druhý parameter je adresa a port, kam treba poslať správu po konverzií. Porty 10025 a 10026 sa v tomto príklade zhodujú s portami použitými v ukážke v časti 4.1.

Ďalej je možné pridať podporu ďalších formátov vo funkcii `convert_message`. Nasledovný príklad umožní namiesto akéhokoľvek súboru (ak nebolo aplikované iné pravidlo) pridať textový súbor obsahujúci dĺžku pôvodnej prílohy v znakoch. Okrem toho sú prípadné chyby zaznamenané do súboru a stručne ohlásené príjemcovi.

```
elif content_types.any_type.match(msg_type) is not None:
    cmd = ["wc", "-c", Value.ORIGINAL_FILE_NAME, ">", Value.····
      CONVERTED_FILE_NAME]
    converted_message, msg_err_n = self.run_external_filter(message, ···
      msg_id, cmd, lambda x: x + "_fail.txt", "text", "plain", shell=···
      True)
    msg_err += msg_err_n
```

Za povšimnutie stojí použitie parametra `shell`. Zabezpečí, že sa príkaz vykoná pomocou interpretera príkazov. Preto je možné použiť napríklad presmerovanie do súboru. Z príkazu sa zostrojí textový reťazec. Tento spôsob slúži najmä na demonštračné účely. Pri reálnom nasadení je vhodnejšie nepoužiť parameter `shell` a priamo volať

program s argumentmi. V prípade potreby je možné vytvoriť si jednoduchý skript, ktorý bude robiť prípadné presmerovania do súboru. Reťazce „*text*" a „*plain*" určujú hlavný a vedľajší typ súboru. (Výsledný typ *MIME* bude teda *text/plain*.)

Za bežných okolností program neznáme prílohy zahadzuje. Nie je zložité pridať pravidlo, ktoré by všetky neznáme prílohy posielalo bez zmeny ďalej (napríklad ako *application/octet-stream*). Avšak toto nie je odporúčané, nakoľko sa môžu vyskytnúť neočakávané typy príloh. Okrem toho, útočník môže priradiť súboru, ktorý by bol za bežných okolností skonvertovaný, nejaký neznámy *MIME* typ, čím by sa vyhol konverzii. Pri uložení a následnom otvorení prílohy systém nemá k dispozícii *MIME* typ uložený v správe. Preto sa *Linux* pokúsi rozpoznať *MIME* typ tohto súboru, zatiaľ čo *Windows* sa pokúsi otvoriť súbor podľa prípony.

Z rovnakého dôvodu nie je vhodné nechať prejsť bez zmeny súbory typov, ktoré by sa dali považovať za bezpečné. Opäť cieľový systém môže interpretovať typ súboru rozdielnym spôsobom. Preto je vhodné pri vynechaní konverzie minimálne použiť skript, ktorý skontroluje, či je typ súboru (ako ho rozozná systém) zhodný s udaným typom v správe, napríklad pomocou nástroja *file*. Okrem toho by mal tento skript skontrolovať príponu súboru pre prípad, že na cieľovom počítači beží systém, ktorý berie do úvahy príponu súboru. Nechávať súbory bez konverzie však nie je odporúčané.

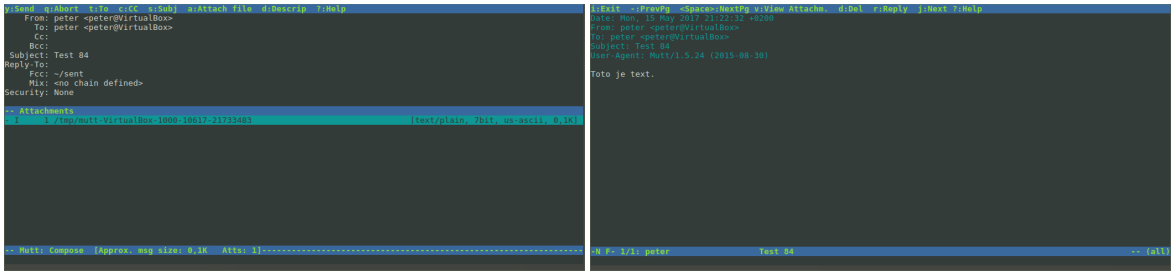
Kapitola 5

Praktická demonštrácia

Táto kapitola obsahuje niekoľko ukážok použitia. Na obrázku 5.1a vidno posielanie obyčajnej textovej správy pomocou programu *Mutt*. Na obrázku 5.2b je vidno zobrazenie prijatej správy. Mailový klient bez problémov rozpoznal že ide o textovú správu, a hneď zobrazil text.

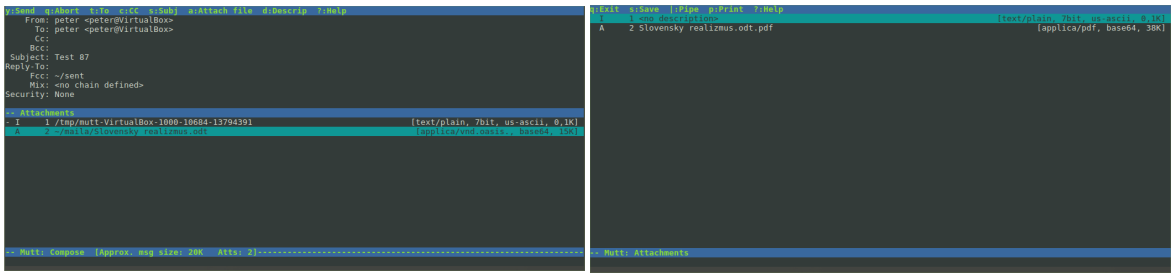
Správa s prílohou (obrázok 5.2) je spracovaná mierne odlišne. Server nemá problém ani so správami pozostávajúcimi z viacerých častí. Súbor formátu *odt* bol skonvertovaný do *PDF* prílohy. V štandardnom zobrazení programu *Mutt* sa samotný text správy zobrazuje priamo v hlavnom náhľade správy. Na obrázku 5.2b je vidno podrobné zobrazenie obsahu mailu. Text správy je označený písmenom `I`, zatiaľ čo `PDF` súbor je rozpoznaný ako príloha označená písmenom `A`.

Ako posledný príklad uveďme ukážku výsledného *PDF*. Ako je vidno, zachovalo sa formátovanie aj odrážky.



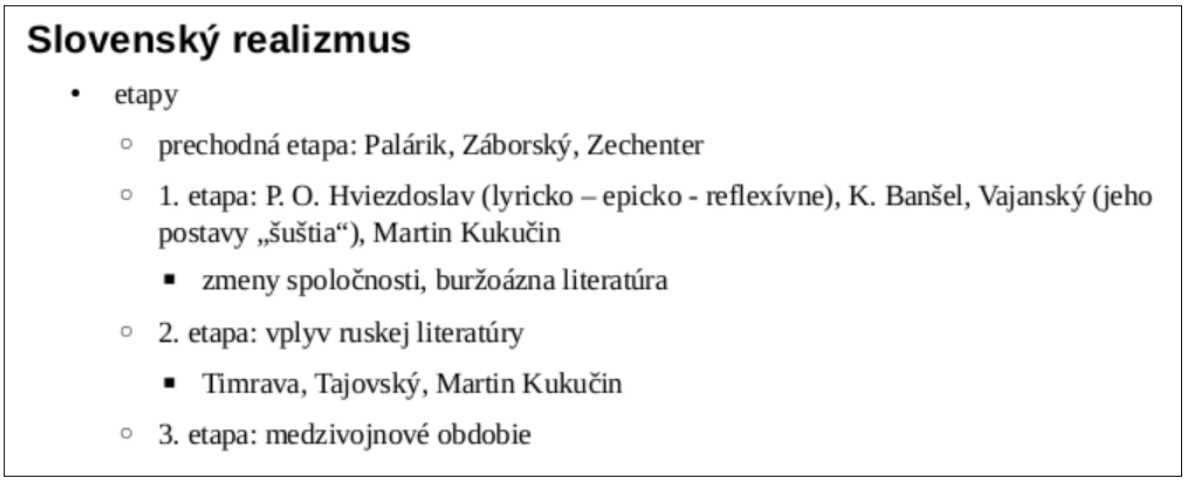
(a) Odoslanie (b) Prijatie

Obr. 5.1: Odosielanie textovej správy



(a) Odoslanie (b) Prijatie

Obr. 5.2: Odosielanie správy s prílohou



Obr. 5.3: Ukážka *PDF* výstupu.

Záver

Cieľom práce bolo vytvoriť riešenie zamerané hlavne na bezpečnosť. V práci sme rozobrali základy problematiky bezpečnosti e-mailu a viacero možných riešení. Poskytli sme aktuálny výber z dostupného softvéru použiteľného v súlade s cieľom tejto práce a teoretický koncept riešenia. Toto riešenie bolo následne implementované v podobe funkčného programu, ktorý prepája jednotlivé použité komponenty.

Aj keď sa pôvodné ciele podarilo naplniť, vždy existuje priestor pre ďalšie vylepšovanie. Budúce práce sa môžu zamerať na zjednodušovanie nasadenia, napríklad vytvorením skriptov pre konfiguráciu v konkrétnych situáciách. Čitateľ by po prečítaní tejto práce mal byť schopný nakonfigurovať server podľa svojich predstáv. Zároveň má široké možnosti prispôsobenia. To však môže brániť širšiemu rozšíreniu nášho spôsobu riešenia medzi menej zdatnými administrátormi systémov.

Ďalšou prakticky neobmedzenou oblasťou pre zlepšenie je podpora viacerých druhov súborov. Kvalitatívne ide o jednoduchú úlohu, avšak kvantitatívne nie, nakoľko rôznych typov súborov je veľké množstvo a kedykoľvek môže pribudnúť ďalší často používaný. V tejto práci pokrývame len nutné minimum pre demonštráciu funkčnosti.

Taktiež je možné zabudovať možnosť nepoužiť konverziu niektorých typov súborov. V časti 4.3 je spomenutý spôsob, ako sa niečo také dá vykonať pri súčasnom riešení. Avšak je možné spraviť praktickejšie a bezpečnejšie riešenie s pridaním rôznych voliteľných kontrol.

Ako poslednú uveďme možnosť navrhnúť odporúčania konkrétnych kombinácií s inými riešeniami, prípadne príklady bezpečnostných politík zahŕňajúcich naše riešenie.

Literatúra

[1] The Postfix Home Page.
URL http://www.postfix.org/

[2] Bhushan, A.; Pogran, K.; Tomlinson, R.; a kol.: Standardizing Network Mail Headers. RFC 561, September 1973.
URL https://tools.ietf.org/html/rfc561

[3] of Electrical Engineering, D.: STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES. RFC 822, University of Delaware, Newark, DE 19711, August 1982.
URL https://tools.ietf.org/html/rfc822

[4] Freed, N.; Borenstein, N.: Multipurpose Internet Mail Extensions... RFC 2045-2049.

[5] Houdek, M.: postfix-filter-loop.
URL https://github.com/MiroslavHoudek/postfix-filter-loop

[6] Klensin, J.: Simple Mail Transfer Protocol. RFC 5321, Október 2008.
URL https://tools.ietf.org/html/rfc5321

[7] Klesin, J.: SMTP 521 and 556 Reply Codes. RFC 7504, IETF, Jún 2015.
URL https://tools.ietf.org/html/rfc7504

[8] Leiba, B.: Update to Internet Message Format to Allow Group Syntax in the "From:änd "Sender:"Header Fields. RFC 6854, Huawei Technologies, Marec 2013.
URL https://tools.ietf.org/html/rfc6854

[9] Myer, T. H.; Henderson, D. A.: Message Transmission Protocol. RFC 680, Network Working Group, Apríl 1975.
URL https://tools.ietf.org/html/rfc680

[10] P. Resnick, E.: Internet Message Format. RFC 5322, Qualcomm Incorporated, Október 2008.
URL https://tools.ietf.org/html/rfc5322

[11] Postel, J. B.: SIMPLE MAIL TRANSFER PROTOCOL. RFC 821, Information Sciences Institute, University of Southern California, 4676 Admiralty Way, Marina del Rey, California 90291, August 1982.
URL `https://tools.ietf.org/html/rfc821`

[12] Resnick, P.: Internet Message Format. RFC 2822, QUALCOMM Incorporated, Apríl 2001.
URL `https://tools.ietf.org/html/rfc2822`