VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

MODUL ANTIVIROVÉ KONTROLY PRO THUNDERBIRD

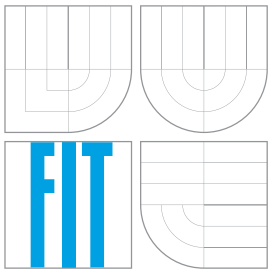
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LUKÁŠ ĎURFINA

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

MODUL ANTIVIROVÉ KONTROLY PRO THUNDERBIRD

VIRUS CHECKING MODULE FOR THUNDERBIRD

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LUKÁŠ ĎURFINA
AUTHOR

VEDOUCÍ PRÁCE — Dr. Ing. PETR PERINGER
SUPERVISOR

BRNO 2008

Abstrakt

Táto bakalárska práca sa zaoberá vývojom modulu antivírovej ochrany pre e-mailový klient Thunderbird. Modul kontroluje obsah e-mailov pomocou nástroja AVG. Kontrola prebieha testovaním príloh v doručených a odoslaných správach. Modul je implementovaný použitím technológií XUL, XPCOM, JavaScript a DOM. Funguje s protokolmi POP3 a IMAP a umožňuje vykonanie rôznych akcií s infikovanou správou.

Klíčová slova

Thunderbird, antivirová ochrana, e-mail, XUL, XPCOM, JavaScript, DOM, AVG

Abstract

This bachelor thesis deals with development of virus checking module for e-mail client Thunderbird. The module checks e-mail content using AVG tools. It is accomplished by scanning attachments in incoming and outgoing e-mails. The module is implemented by using technologies XUL, XPCOM, JavaScript and DOM. It works successfully with protocols POP3 and IMAP, and provides execution of several actions with infected message.

Keywords

Thunderbird, antivirus security, e-mail, XUL, XPCOM, JavaScript, DOM, AVG

Citace

Lukáš Ďurfina: Modul antivirové kontroly pro Thunderbird, bakalářská práce, Brno, FIT VUT v Brně, 2008

Modul antivirové kontroly pro Thunderbird

Prohlášení

Vyhlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Dr. Ing. Petra Peringera. Ďalšie informácie mi poskytol Ing. Karel Obluk, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Lukáš Ďurfina
28. dubna 2008

Poděkování

Ďakujem spoločnosti AVG Technologies CZ za možnosť pracovať na tejto bakalárskej práci, pánovi Ing. Karelovi Oblukovi, Ph.D. za pripomienky a usmernenia, a takisto Dr. Ing. Petrovi Peringerovi za vedenie na univerzitnej pôde.

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Ďurfina Lukáš**
Obor: Informační technologie
Téma: **Modul antivirové kontroly pro Thunderbird**
Kategorie: Bezpečnost

Pokyny:

1. Seznamte se s metodami pro testování přítomnosti škodlivého kódu v souborech a s implementací modulů pro e-mailový klient Thunderbird.
2. Navrhněte modul antivirové kontroly pro Thunderbird s využitím programového rozhraní AVG API. Uživatelské rozhraní modulu musí umožňovat nastavení parametrů kontroly, její aktivaci/deaktivaci a zobrazení základních informací o stavu AVG.
3. Implementujte navržený modul pro Thunderbird v prostředí operačního systému Microsoft Windows.
4. Výsledné řešení řádně otestujte a zhodnoťte jeho přínos.

Literatura:

- Dle zadání vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění prvních dvou bodů zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Peringer Petr, Dr. Ing.**, UITS FIT VUT
Konzultant: Obluk Karel, Ing., Ph.D., Grisoft
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. **Pan**

Jméno a příjmení: **Lukáš Ďurfina**
Id studenta: 79044
Bytem: Záhumenice 631/83, 951 48 Jarok
Narozen: 19. 11. 1986, Nitra
(dále jen "autor")

a

2. **Vysoké učení technické v Brně**

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Modul antivirové kontroly pro Thunderbird
Vedoucí/školitel VŠKP: Peringer Petr, Dr. Ing.
Ústav: Ústav inteligentních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Obsah

Kapitola 1

Úvod

Základom ľudského spolužitia je komunikácia. V modernej dobe informačných technológií je často využívanou formou komunikácie elektronická pošta. Pre elektronickú poštu sa bežne používa označenie e-mail. Na prácu s e-mailami slúži e-mailový klient, napr. voľne dostupný produkt organizácie Mozilla, *Thunderbird* [11]. Dôležitým aspektom pokojného a bezproblémového posielania a prijímania e-mailov je bezpečnosť, tá zohľadňuje viacero hľadísk, napr. totožnosť odosielateľa, nepozmenenie obsahu a samotný obsah e-mailu, ktorým sa zaoberá táto práca. Je dôležité, aby sa k užívateľovi nedostal prostredníctvom e-mailu nebezpečný obsah, ako počítačové vírusy, trójske kone, spyware apod. Na takúto bezpečnosť sa zameriava napr. antivírový program *AVG* spoločnosti AVG Technologies [7].

Avšak problém nastáva, keď e-mailový klient komunikuje s poštovým serverom pomocou šifrovaného protokolu, pretože tým sa stráca možnosť kontroly obsahu elektronickej pošty. Cieľom mojej práce bolo vytvoriť zásuvný modul pre poštový klient Thunderbird, ktorý bude kontrolovať prichádzajúcu a odchádzajúcu poštu pomocou AVG API. Tým, že modul je integrovaný priamo v klientskej aplikácií bude možné kontrolovať e-maily, ktorých kontrola by inak bola znemožnená šifrovaním komunikačného protokolu, a takisto modul umožní užívateľovi lepšiu kontrolu nad vykonávanými akciami s infikovanou poštou.

Táto práca je rozdelená do kapitol podľa časového postupu jej vývoja. Kapitola 2 obsahuje základné informácie o danej problematike, je tu načrtnutá problematika bezpečnosti, hlavne elektronickej pošty. Nasleduje popis aplikácie Thunderbird a použitých technológií pri vytváraní modulu. V kapitole 3 je rozobratá analýza požiadavkov. Najskôr sú zhrnuté požiadavky, ktoré by bolo vhodné implementovať. Následne je nahliadnuté na prostriedky, ktoré ponúka Thunderbird programátorom modulov a mohli by byť užitočné pre návrh a následne implementáciu. Po analýze je v kapitole 4 predstavený návrh aplikácie, kompletnej štruktúry a vnutorných väzieb medzi jednotlivými časťami. Na záver je ukazaný objektový návrh XPCOM komponentu poskytujúceho AVG rozhranie. V predposlednej kapitole 5 je opísaný proces implementácie. Sú tu predstavené problémy, ktoré ho sprevádzali, ale taktiež sú spomenuté veci, ktoré ho spríjemnili. Nakoniec je opísaný proces testovania, nasledovaný záverom so zhodnotením a víziami do budúcnosti.

Kapitola 2

Prehľad aktuálneho stavu

Práca sa zaoberá bezpečnosťou e-mailovej komunikácii a bude založená na platforme Mozilla. V nasledujúcich odstavcoch bude uvedený prehľad, ktorý priblíži momentálnu situáciu počítačovej bezpečnosti a technológie, na ktorých bude postavená realizácia práce.

2.1 Bezpečnosť

Počítačová bezpečnosť je čoraz častejšie skloňovaný termín [2]. V dnešnej dobe, keď väčšina procesov je riadených alebo spracovávaných počítačom, je to úplne prirodzené. Zvyšuje sa povedomie užívateľov o dôležitosti si chrániť svoje osobné údaje voči odcudzeniu a zneužitiu. Tlak na zvyšovanie bezpečnosti spôsobuje činnosť rôznych skupín ľudí, ktorí sa snažia systematicky narúšať integritu systémov a obohacovať sa s ilegálne získanými údajmi. Preto sa bezpečnosť dostáva na popredné priečky pri hodnotení systémov, programov či zariadení a bol vytvorený štandard ISO pre vytváranie aplikácií so zvýšenou bezpečnosťou.

Možností ako prekonať zabezpečenie systému a narušiť jeho integritu je viac, najčastejšie používané sú:

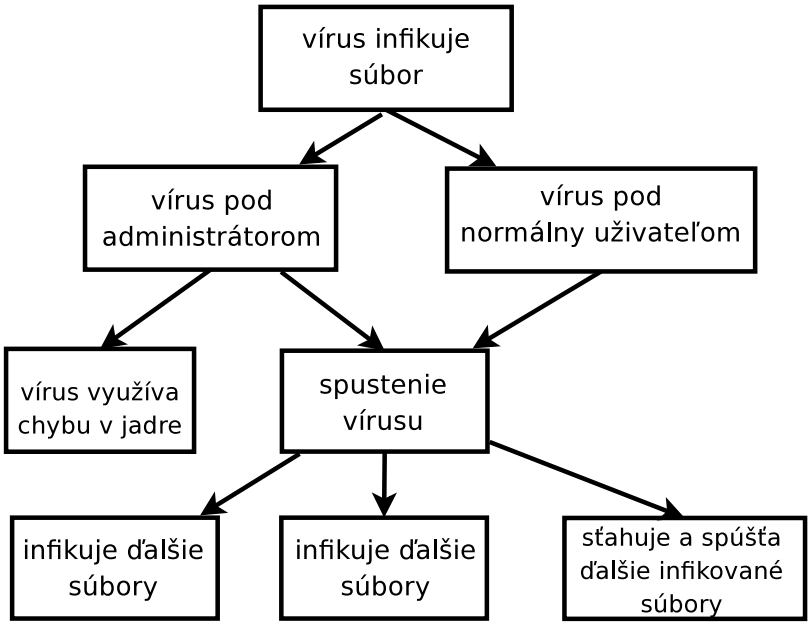
- počítačový vírus – je to program, ktorý zneužívajú chybu systému alebo dôveru užívateľa k svojmu aktivovaniu
- sociálne inžinierstvo – označuje priame získanie prístupu do systému vďaka neopatrnosti užívateľa, ktorý môže mať heslo napísané na monitore alebo ho prezradí neznámej osobe
- rootkit – je program umožňujúci získať práva administrátora systému bez jeho autorizácie
- backdoor – popisuje možnosť programu poskytnúť vzdialený prístup k systému bez toho, aby bol tento prístup detekovaný
- trójsky kôň – program, ktorý sa vydáva za neškodný a na pozadí vykonáva nebezpečnú činnosť
- červ – vyznačuje sa rýchlym šírením po sieti, čím ju zahlcuje a vytvára DoS útok
- spyware – špionážny program, ktorý sa nemusí nejako špeciálne prejaviť, ale kradne osobné údaje

- exploit – môže byť kúsok programu, zhluk dát alebo postupnosť príkazov, ktoré zneužívajú chybu v programe, najčastejšie pretečenie zásobníku k spusteniu nebezpečného kódu

2.1.1 Počítačové vírusy

Táto práca sa zaoberá ochranou užívateľa práve pred počítačovými vírusmi, preto budú bližšie popísané a rozdelené. Pri trochu väčšej abstrakcii by sme sem mohli zaradiť aj trójske kone, červy a spyware, pre ktoré platí, že každý z nich má svoje špecifické správanie a zameriava sa na niečo iné, ale spája ich spoločný cieľ poškodiť užívateľa.

Medzi ich hlavné vlastnosti patrí schopnosť sebakopírovania a napadnutia počítača bez povolenia a znalosti vlastníka PC. Na svoje šírenie používa všetky možné cesty ako prenosné média (CD, DVD, USB flash disky), alebo sieťové spojenia v rámci lokálnej či globálnej siete, sieťový súborový systém atď, pričom zneužíva najviac používané protokoly.



Obrázek 2.1: Znázornenie možného vývoja infekcie vírusom

Vírusy by sa historicky mohli rozdeliť do nasledovných skupín:

- polymorfné – menia svoju štruktúru
- makrovírusy – vírusy v makrách textových dokumentov
- stealth – používajú maskovacie techniky na ukrytie v systéme, napr. kryptovanie, polymorfizmus alebo metamorfózu
- boot – zapisujú sa do zavádzacieho sektora
- rezidentné – po spustení zostávajú v operačnej pamäti

Dnes už tieto kategórie splývajú a nebezpečný kód sa delí do kategórií podľa činnosti akú vykonáva a nie podľa spôsobu ako ju vykonáva, tak ako je to uvedené v predchádzajúcom odseku.

2.1.2 Antivírová ochrana

Hlavným cieľom antivírovej ochrany je detekovať infiltrácie, pokúšať sa ich liečiť, v prípade neúspechu zmazať súbor s infekciou alebo ho presunúť do antivírovej truhly. Dôležitým aspektom je včasná reakcia antivírového systému, aby bol užívateľ informovaný o možnom nebezpečenstve skôr, ako môže byť ohrozený.

V dnešných dňoch sa presúva ťažisko z klasických vírov, ktoré sa pokúšali užívateľovi poškodiť údaje na disku alebo znepríjemniť jeho činnosť na počítači, na infiltrácie, ktoré sa snažia maximálne maskovať, tak aby si ich infikovaný užívateľ ani nevšimol. Pričom často využívajú pokročilé metódy ako šifrovanie. Toto kladie väčšie nároky na antivírovú kontrolu, keďže užívateľ si vôbec nemusí všimnúť, že jeho PC je infikované a využívané na nelegálnu činnosť ako napr. účasť na DDoS útokoch, rozosielanie spamu alebo sledovanie činnosti užívateľa s cieľom získať citlivé informácie.

2.2 Platforma Mozilla

Základnými časťami platformy sú XUL, XPCOM a JavaScript. Jej sila spočíva v budovaní vizuálnych a interaktívnych aplikácií, tým je predurčená na vývoj front-end aplikácií a je nevhodná na tvorbu ovládačov zariadení, serverových aplikácií a systémov spracovávajúcich data v dávkach.

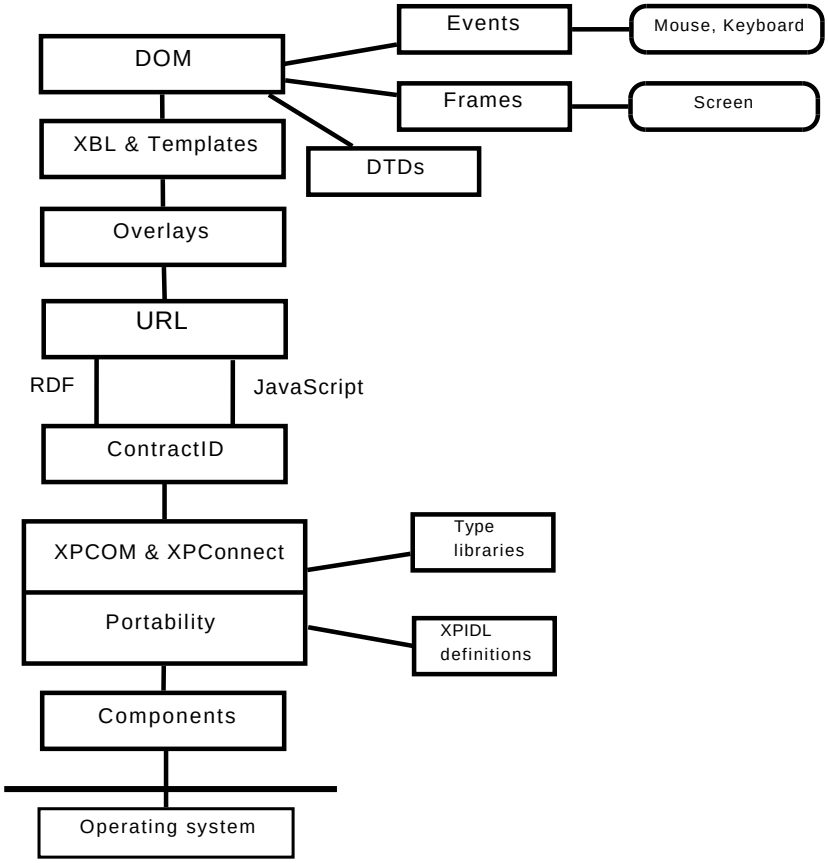
Schéma na obrázku 2.2 predstavuje zjednodušený pohľad na architektúru platformy Mozilla, kde boli vypustené niektoré menej známe časti a ponechané hlavné piliere platformy a prvky, ktoré budú v tejto práci spomenuté.

Medzi jej výhody patrí multiplatformnosť, ktorá je zabezpečená na najnižšej vrstve, vďaka ktorej je možné jej použitie na všetkých rozšírených operačných systémoch. Funkcionalita je rozdelená do komponentov vytvorených v C/C++, čo sa pozitívne odráža na konečnom výkone. Tvorba grafických rozhraní je riešená deklaratívnym zápisom v štýle XML a bude opísaný v kapitole o XUL.

Na zobrazovanie slúži renderovacie jadro *Gecko*. Podporuje najvýznamnejšie priemyselné štandardy. Keďže zobrazovanie webového obsahu je nevyhnutnosťou, tak obsahuje systaktické analyzátory (parsery) HTML, XML, CSS a implementáciu DOMu. Gecko je modulárne a ľahko zabudovateľné do aplikácií tretích strán, čo je odzkradlené flexibilným programovacím rozhraním ako Web Shell API, Java wrapper API a Gecko ActiveX Control.

Každá platforma má zväčša svoju špecifickú sieťovú vrstvu, tento problém odstraňuje Mozilla's Network library tzv. *Necko*. Zabezpečuje niekoľko vrstiev pre prístup transportnej a aplikačnej vrstvy, generický a rozšíriteľný systém na spracovanie URL bežných protokolov ako http, ftp, file a môže byť obohatený o ovládače ďalších protokolov. Okrem toho poskytuje abstraktné rozhranie pre nízko úrovňový prístup k súborom a ďalšie sieťové služby ako asynchrónne kešovacie DNS rozlíšenie, web proxy a HTTPS.

Vývoj na tejto platforme je pomerne rýchly, ale jeho hlavnou nevýhodou je to, že nie je medzi vývojármi známa, čo môže byť zapríčinené aj nedostatkom informácií, kvalitnej dokumentácie a masívnej reklamy, tak ako to je pri platformách od veľkých medzinárodných spoločností. Väčšiemu rozšíreniu by určite pomohlo partnersvo s veľkou firmou, ktorá by zabezpečila financie a podporu vývoja.



Obrázek 2.2: Zjednodušený pohľad na architektúru platformy Mozilla

2.3 Thunderbird

Emailový klient Thunderbird je multiplatformová aplikácia založená na platforme Mozilla. Zdrojový kód je otvorený distribuovaný pod licenciami MPL, GPL, LGPL [9], pričom užívateľ si môže vybrať licenciu, ktorá mu najviac vyhovuje. Momentálne je dostupná rada 2, konkrétne verzia 2.0.0.12.

Thunderbird je moderná aplikácia na organizáciu, zabezpečenie a sprístupnenie elektronickej pošty, služby RSS a elektronických konferencí netnews. E-maily je možné prehľadne rozdeľovať do užívateľom vytvorených položiek alebo ich označiť tzv. tagom, napr. „To Do“. K podobnému rozhraniu, ako má internetový prehliadač prispievajú tlačítka spať a dopredu na navigáciu v historií správ. Uživateľ ocení vylepšené vyhľadávanie s možnosťou ukladania výsledku vyhľadávania. Aj samotný Thunderbird dbá na bezpečnosť jeho užívateľov a integruje antiphishingovú ochranu a podporuje automatické aktualizácie.

Najväčší konkurenti Thunderbirdu sú Microsoft Outlook, The Bat! od firmy RITLabs, toto sú však komerčné produkty s uzavretým zdrojovým kódom, takže užívateľ za ne musí zaplatiť, a ešte je aj ochudobnený o možnosť vlastnej úpravy modifikovaním zdrojového kódu. Ďalším kvalitným konkuretnom by mohla byť Eudora, ktorej vývoj však bol pozastavený a firma, ktorá ju vlastnila teraz podporuje vývoj Penelope, čo je rozšírenie pre Thunderbird, ponúkajúce funkcie Eudory, aj toto dokazuje výnimočné postavenie Thunderbirdu.

Rozšírenie aplikácie je možné vo forme pluginov. Pre plugin sa používa viac pojmov, najčastejšie rozšírenie, doplnok a modul. Medzinárodným označením je anglický termín *extension*. K ľahkému úvodu do tvorby rozšírení a získaniu základnému prehľadu o platforme Mozilla je vhodná publikácia [4].

2.4 Použité technológie

V dôsledku toho, že Thunderbird je založený na pomerne veľkom množstve technológií, tak môj modul ich využíva tiež. Toto množstvo plynie zo snahy dosiahnutia multiplatformnosti, kvôli čomu bolo potrebné vytvoriť abstraktné vrstvy. Väčšina technológií pochádza od Mozilly a poskytujú veľmi dobrý prostriedok na vývoj mohutných a prenositeľných aplikácií, veľmi dobre to je popísané vo voľne dostupnej knihe *Rapid Application Development with Mozilla* od *Nigela McFarlane* [5].

2.4.1 XPInstall

XPInstall [16] je skratka z *Cross-Platform Install*. Táto technológia slúži na inštalovanie prídavných modulov do aplikácií od Mozilly [10]. Jej základom je inštalačný súbor s koncovkou XPI, vyslovuje sa "zippy". Tento súbor je v skutočnosti archív vo formáte ZIP, ktorý obsahuje hierarchiu adresárov a súborov potrebných na zavedenie modulu do aplikácie a jeho následného používania.

V adresári *components/* bývajú uložené XPCOM komponenty, ktoré rozšírenie používa a nie sú súčasťou rozširovanej aplikácie. V systéme Windows to je súbor dll s implementáciou funkcí a binárny súbor xpt s rozhraním pre tieto funkcie a typovými informáciami.

Ďalším adresárom je *defaults/*. Tu sú zhrnuté východzie nastavenia aplikácie. Je zvykom vytvoriť ďalší podadresár preferences/, kde sa v súbore s koncovkou .js uložia nastavenia rozšírenia, ktoré budú použité pri prvom použití. Väčšinou sú to stavy zaškrtávacích políčok, textových polí apod.

Hlavným úložiskom býva adresár *chrome/*. Je to z toho dôvodu, že tu sú uložené všetky súbory, ktoré popisujú grafické rozhranie, skripty v jazyku JavaScript, kaskádové štýly CSS a ďalšie pomocné súbory ako obrázky. Súbory s popisom GUI majú príponu .xul a spolu so skriptami sa ukladajú do podadresára content/. Súbory ovplyvňujúce konečný vzhľad aplikácie sa ukladajú do podadresára skin/.

Adresár *locale/* môže byť aj súčasťou adresára chrome/ ako podadresár, záleží na voľbe vývojára, ktorú potom musí zohľadniť v manifest súbore. Adresár locale/ obsahuje lokalizačné súbory, ktoré sú ukladané do podadresárov nazvaných podľa jazykovej mutácie, napr.: cs-CZ/, sk-SK/, en-US/ atď. Použije sa jazyková mutácia zhodná s mutáciou rozširovanej aplikácie, ak rozšírenie takúto mutáciu neposkytuje, tak je použitá mutácia, ktorá je ako prvá deklarovaná v manifest súbore.

Okrem adresárov sa v inštalačnom balíku nachádzajú ešte ďalšie dva súbory. Prvým je *chrome.manifest*, ktorý popisuje cesty do adresárov pre obsah, lokalizácie a vzhľad. Ďalej deklaruje tzv. *overlays*, čím určuje cieľovej aplikácii, ktoré jej súbory budú doplnené o obsah daných súborov. Týmto sa do aplikácie pridávajú ďalšie grafické prvky, ale aj aplikačná logika.

Posledným súborom je *install.rdf*, ktorý je vo formáte RFD, čo znamená resource description framework a je založený na XML. Jeho náplňou je popis daného rozšírenia a programu, pre ktoré je určené. Základnými údajmi uvedenými o rozšírení je jeho ID, verzia, typ (2 - označuje rozšírenie), názov, popis, tvorca, url s domácou www stránkou, url

s novými verziami, cieľovú platformu a ďalšie. Cieľová aplikácia je určená pomocou jej UUID, napr. pre Thunderbird je to 3550f703-e582-4d05-9a08-453d09bdfdc6 a najstaršou a najnovšou verziu aplikácie, pre ktorú môže byť rozšírenie nainštalované.

URL s novými verziami smeruje na súbor typu RDF. Tento súbor obsahuje výpis všetkých doteraz dostupných verzií. Pri zázname každej verzie je uvedené, pre ktoré verzie cieľovej aplikácie je určená a URL s umiestením rozšírenia. Špeciálne je označená posledná verzia, kde je okrem spomínaných vlastností uvedený sha256 hash na kontrolu korektného stiahnutia.

2.4.2 XUL – XML User Interface Language

XUL je značkovací jazyk od Mozilly založený na XML [17], ktorý slúži na popis užívateľského grafického rozhrania. Na jeho renderovanie sa používa renderovacie jadro Gecko. Týmto jazykom sú popísané všetky užívateľské rozhrania aplikácií od Mozilly a je ďalším pilierom prenositeľnosti. XUL je veľmi podobný XHTML, keďže takisto používa známe webové technológie CSS, JavaScript a DOM.

Prvky UI sú definované XML tagmi [18]. Základom býva prvok *window*, *dialog*, *prefwindow* alebo *wizard*, ktoré určujú zameranie konkrétneho okna a obsahujú ďalšie prvky, ktoré tvoria obsah okna.
Mohli by sme ich rozdeliť do niekoľkých kategórií:

- informačné – label, caption, description, tooltip
- nastaviteľné – checkbox, radiogroup, textbox
- definujúce vzhľad – groupbox, hbox, vbox, separator
- definujúce menu – menu, menupopup, menuitem, menubar
- akčné(vyvolávajúce akciu) – button, key, observes

Všetky prvky sú dedičnosťu odvodené od základných prvkov XUL(XUL Element) a DOM(Element), od ktorých získavajú vlastnosti a metódy. Podstatné dedené vlastnosti od XUL elementu sú align, height, hidden, id, style, width a od DOM elementu attributes, firstChild, lastChild, parentNode, tagName. Jednotlivé prvky pridávajú svoje osobitné vlastnosti ako disabled, checked, selected apod. XUL element poskytuje len pár metód ako blur, focus, click, dôležité metódy sú dedené z DOM elementu: addEventListener, dispatchEvent, appendChild, removeChild, replaceChild, hasAttribute, setAttribute, removeAttribute atď.

Interakcia s užívateľom je naprogramovaná pomocou javascriptu a vyvoláva sa na základe udalostí. Používané udalosti sú bežne známe, takže s ich porozumením nie sú ťažkosti, pre istotu si ich môžeme zopár pripomenúť.

Príklady udalostí:

- onload – po načítaní okna, pred jeho zobrazením
- onunload – po uzatvorení okna
- onclick – po kliknutí na prvok
- onmousemove – pri pohybe myšou nad prvkom
- ondblclick – pri dvojkliku myšou

- onkeypress – pri stlačení klávesy
- oninput – pri zadaní vstupu napr. do textboxu
- ondragover – pri presune elementu nad prvkom, kurzor môže reagovať, ak je možné prvok uvoľniť

Vzhľad prvkov môže byť upravený pomocou CSS, do ktorého sú pridané špecialne vlastnosti špecifické pre Mozillu. Tieto vlastnosti majú predponu `-moz`. Umožňujú veci ako zaoblenie okrajov, úpravu pozadia, priehľadnosť apod. K jednotlivým prvkom XUL dokumentu je možné zo skriptov pristúpiť pomocou DOMu. Aby to bolo uskutočniteľné, je potrebné každému prvku, ku ktorému sa chceme takto dostať, definovať jeho id. Získanie prvku je už potom triviálne pomocou funkcie `getElementById()`. Následne môžeme meniť jeho vlastnosti a volať jeho metódy.

V tomto jazyku je veľmi dobre podporovaná lokalizácia. Je štandardom všetky textové reťazce ukladať do externých súborov. Na lokalizáciu slúžia 2 typy súborov. Sú to DTD a properties. DTD obsahujú tzv. dtd entity, ktoré sa skladajú z identifikátora a textu, napr. `<!ENTITY tab.general "Všeobecné">`, kde `tab.general` je identifikátor. Pri použití v XUL súbore bývajú uvedené v doctype hlavičke. Druhý typ slúži na použitie v skripte. Pomocou tagu stringbundleset sa vkladá do súboru XUL. Tento prvok musí mať určené id, pomocou ktorého ho získame v skripte, a následne získame žiadaný reťazec funkciou `getString(id_reťazca)`. Príkladom záznamu je napr. `ok=je v poriadku!`, pričom `ok` je id_reťazca. Ak `id_reťazca` nie je nájdené, tak je vyvolaná výnimka.

2.4.3 XPCOM

XPCOM – Cross Platform Component Object Model [13] je multiplatformový komponentový objektový model od Mozilly. Je podobný modelu CORBA a tiež čiastočne COMu od firmy Microsoft. Pomocou tejto technológie je možné funkcionalitu rozdeliť do menších častí, tzv. *komponentov*. Výborným sprievodcom pri vytváraní komponentov je [1].

Podstatnou vecou je rozdelenie implementácie a rozhrania komponentov. Implementácia býva obsiahnutá v dynamickej knižnici, v systéme Windows je to súbor dll, a rozhranie je popísané pomocou XPIDL – interface description language pre XPCOM [15], súbor má príponu idl a je z neho generovaný binárny súbor xpt s popisom rozhrania a typovými informáciami na pripojenie komponentu pomocou XPConnect. XPConnect zabezpečuje most na prepojenie komponentov a programom v podporovanom programovacom jazyku.

Komponent XPCOM môže byť vytvorený v rôznych jazykoch, zvyčajne sú to C, C++, JavaScript, Java alebo Python. Pomocou XPConnectu je však možné sa k nej pripojiť z väčšieho množstva jazykov. Okrem už spomínaných je to Perl a ďalej sa pracuje na podpore ďalších jazykov ako napr. Ruby. Samotné XPCOM poskytuje množinu jadrových komponentov a tried na prácu so súbormi, pamäťou, vláknami, základnými dátovými štruktúrami apod.

Na pripojenie sa ku konkrétnemu komponentu je potrebné uviesť jej ContractID, čo je jej jednoznačný reťazcový identifikátor. Trieda sa jednoznačne identifikuje tvarom *Components.classes["ContractID"]*, ktorý je nasledovaný jednou z metód `getService()` alebo `getInstance()`. Tieto metódy majú jeden parameter a to je názov požadovaného rozhrania. Metóda getService() sa používa na triedy typu singleton, tj. také, ktoré majú len jednu instanciu v pamäti. Použitie getInstance() na takýto typ triedy spôsobí nepredvídateľné správanie. Metóda getInstance() sa používa na všetky ostatné triedy.

Špecifikácia tried je popísaná pomocou jazyka XPIDL v súbore s koncovkou idl. Na základe tohoto súboru sa pomocou nástroja xpidl vytvorí C++ hlavičkový súbor alebo súbor s java rozhraním a súbor xpt. XPIDL popisuje rozhranie, ktoré je jazykovo a platformovo nezávislé. K rozhraniu sa uvádza UUID, aby malo jedinečné označenie. Deklaruje sa jeho názov, napr. `interface IAvg7Component : nsISupports`, čím sa údava, že rozhranie dedí nsISupports, čo je základné rozhranie pre všetkých komponentov. V deklarácii rozhrania sa uvedú všetky atribúty a metódy, ktoré budú verejne dostupné.

Ak chceme aby komponent bol prístupný aj zo skriptového prostredia, musí sa celé rozhranie deklarovať ako `scriptable` pre uvedením UUID. Týmto sa zabezpečí možný prístup z JavaScriptu. K atribútom sa môže uviesť modifikátor `readonly`, čím sa zamedzí ich zmene z vonkajšieho prostredia. Niektoré metódy tried môžu byť označené modifikátorom `[noscript]`, to značí, že ich nie je možné volať z javascriptového kódu, ale len z C++, resp. C kódu. Pri parametroch metód sa musia uvádzať modifikátory `in`, `out` alebo `inout` na určenie a sprehľadnenie ich použitia. Modifikátorom `[optional]` je možné deklarovať voliteľné parametre metódy. Tieto musia byť následne použité v C++ kóde ako implicitné.

2.4.4 AVG API

AVG API je založené na technológií COM – Component Object Model. Obsahuje niekoľko abstraktných tried, ktoré sú zdokumentované v [20] a zameriavajú sa na určitú oblasť, sú to informácie o licencii, o aktuálne nainštalovanej verzii AVG, ovládanie nastavení AVG, možnosť testovať súbory, vyhodnocovať výsledky z týchto testov, presúvať súbory do antivírovej truhly atď.

Základom technológie COM je poskytnutie rozhrania ku komponentu, v prípade jazyka C++ sú na tento účel použité abstraktné triedy, ktoré sa skladajú z čisto virtuálnych metód, týmto sa obchádza absencie rozhraní implementovaných napr. v Jave alebo C#.

Kapitola 3

Analýza požiadavkov

Hlavnou úlohou rozšírenia je vykonať kontrolu elektronickej pošty tam, kde to pre antivírový program nie je možné. Na pripojenie sa k AVG bude potrebné vytvoriť XPCOM komponent, ktorá bude vytvárať komunikačnú vrstvu medzi rozšírením a AVG pomocou AVG API. Na vývoj rozšírenia bude použitý jazyk JavaScript, ktorý spojí obsluhu užívateľského rozhrania, AVG XPCOM komponenty a ostatných XPCOM komponentov obsiahnutých v Thunderbirde.

3.1 Požiadavky na funkcie

Základnou požiadavkou je dať užívateľovi možnosť zapnúť, resp. vypnúť ochranu, ktorú bude rozšírenie poskytovať. To môže byť niekedy vhodné, keďže kontrola hlavne väčších správ zaberá čas, ktorého nie je nikdy dostatok. Pre lepší komfort je však potrebné umožniť vypnúť len kontrolu odchádzajúcej pošty, pričom prichádzajúca bude kontrolovaná, alebo naopak, takže možnosť vypnúť len konkrétnu kontrolu bude tiež do rozšírenia zahrnutá.

Okrem automatickej kontroly je potrebné užívateľovi umožniť i manuálnu kontrolu príloh, aby si v prípade pochybností mohol opäť overiť, aký je problém s danou prílohou. V prípade, ak mu prišla podozrivá príloha, ktorú sa rozhodol uchovať a neskôr skontrolovať s novšou vírusovou databázou, aby sa presvedčil, že nejde o nový typ nebezpečného kódu apod.

Pri každom type kontroly bude možné nastaviť parametre kontroly, ide o možnosť testovať súbory vo vnútri archívu, oznamovať zamknuté a poškodené archívy, takže užívateľ si bude môcť každý typ kontroly „ušiť na mieru“. Toto bude vhodné, lebo väčšinou nie je potrebné, aby bola vytvorená varovná správa o odosielaní zamknutého archívu, avšak pri doručenej správe to môže byť užitočné.

3.1.1 Kontrola odchádzajúcej pošty

Tak, ako je dôležité, aby sa nebezpečný obsah nedostal k užívateľovi, tak je dôležité, aby sa nešíril aj ďalej od užívateľa k ďalším, vačšinou nič netušiacim užívateľom. Ak je táto činnosť vytváraná úmyselne napr. organizovaným zločinom, je problém jej takýmto štýlom zabrániť. Našou úlohou bude zamedziť neúmyselnému rozosielaniu infikovaných súborov, aby sa nestalo, že užívateľ v dobrej viere pošle kamarátovi zábavnú aplikáciu s trójskym koňom.

Jedinou požiadavkou na túto kontrolu bude oznamovať odosielateľovi správy, že jeho prílohy nie sú v poriadku, a poskytnúť aj krátky popis identifikovaných problémov. V každom

prípade však umožní aj odoslanie takéhoto obsahu, lebo niekedy je to potrebné, napr. v prípade, že užívateľ chce poslať infikovaný súbor antivírovej spoločnosti na podrobnejšie skúmanie. Implicitne bude posielať správy bez týchto príloh.

3.1.2 Kontrola prichádzajúcej pošty

Na kontrolu prichádzajúcej pošty sa kladie najväčší dôraz. Vyplýva to z dôvodu, že prostredníctvom e-mailu sa k nám môže dostať rôzny obsah od neznámych odosielateľov, ktorí sa väčšinou vydávajú za niekoho iného a ďalšie možnosti, ktoré nám môžu ľahko znepríjemniť život.

S počtom užívateľov rastie aj počet predstáv o najvhodnejšom spôsobe informovania o nájdenej infikácií. Pre snahu priblížiť rozšírenie predstavám mnohých užívateľov, by malo poskytovať rôzne formy upozornení o nebezpečenstve a poskytovať dostatočnú paletu vykonávateľných reakcií na tieto podnety. Poďme si predstaviť tie najviac zaužívané.

Základnou možnosťou musí byť informovanie príjemcu pošty v prípade odhaleného nebezpečenstva pomocou varovného dialógu. Varovný dialóg by mal obsahovať základné informácie o správe ako: adresa odosielateľa, predmet správy a výpis nebezpečných príloh s popisom nájdenej infekcie. Vytvorený dialóg musí byť modálneho typu, aby sa nemohlo stať, že bude prehliadnutý, čím by stratil svoju informačnú hodnotu.

Neodmysliteľnou možnosťou je označenie infikovanej správy v predmete správy. Bolo by vhodné, aby si užívateľ mohol daný výstražný text zvoliť sám, napr. pre lepšiu orientáciu, filtrovanie apod. Implicitne by mal byť poskytnutý vystižný reťazec napr. „**INFECTED**“. Reťazec sa bude pripájať na začiatok textu predmetu a od predmetu bude oddelený pomlčkou.

Po upozornení užívateľa je tiež potrebné vykonať s danými správami špecifikované úlohy. Vhodným riešením je presunúť celú správu do vyhradenej položky, určenej len pre daný typ správ. Táto možnosť je vhodná aj pre jednoznačné oddelenie nebezpečných správ od ostatných, ale taktiež na ďalšie prípadné skúmanie správy alebo infikovaných príloh. Výhodou je neporušenie správy, ktoré bude ocenené hlavne pri chybnej detekcii antivírusom, ktorá nastáva ojedinele, ale nikdy sa nedá úplne vylúčiť. Toto správanie bude nastavené implicitne, keďže je bežne používané rôznymi spamovými filtrami.

Ďalšou vhodnou voľbou by mala byť možnosť odstrániť infikované prílohy zo správy. Týmto sa zabráni napr. nechcenému otvoreniu prílohy užívateľom. Ako doplnok takejto možnosti by malo byť umožnené presúvať takto odstránené prílohy do antivírovej truhly, kde by sa k nim mohol užívateľ v prípade potreby vrátiť.

Do rozšírenia môže byť zahrnutá aj možnosť zmazania celej správy. Kvôli vyššie zmieneným faktom môže byť toto dvojsečná zbraň, ale pre niekoho to môže byť vhodná voľba. V každom prípade je potrebné môcť zároveň umožniť zobrazovanie upozornení, aby sa nestalo, že správa bude zmazaná bez užívateľovho vedomia.

Pri prechode správy rôznymi systémami a pri spracovaní rozličnými programami zvyknú tieto aplikácie zapísať do hlavičky správy tzv. X-headers tag. Naše rozšírenie by sa tiež malo do hlavičky prijatej správy podpísať svojím tagom, napr. X-Antivirus: AVG Thunderbird plugin. Toto opatrenie prináša viac informácií o spracovaní správy a môže byť použité k filtrovaniu správ apod.

3.2 Možnosti API Thunderbirdu

Thunderbird obsahuje veľké množstvo XPCOM rozhraní, javascriptových tried a funkcií, ktoré sú prístupné aj programátorovi rozšírenia. Základom vývoja je využívanie XPCOM komponentov. Tieto sa delia na *frozen* (zmrazené) a *unfrozen* (nezmrazené). Výhodnejšie je používať frozen, keďže tieto nie je v pláne meniť, zatiaľ čo unforzen sú neustále vo vývoji a môžu mať aj problémy so stabilitou. Komponenty by sa mohli rozdeliť do niekoľkých kategorií: jadrové, sieťové, týkajúce sa užívateľského rozhrania, vývoja aplikácií, na prácu s DOM a poštou a ostatné.

Niekedy sa dá výhodne použiť aj niektorá globálna premenná z dostupného prostredia, ale je v dobrej vôli vývojára obmedziť svoj prístup k nej len na čítanie. Používanie dostupných funkcií je výhodné, lebo zväčša implementujú bežné často opakované činnosti.

Medzi najdôležitejšie rozhrania k XPCOM komponentom [14] patria

nsISupports – je to základné rozhranie. Zabezpečuje vytváranie, uvoľňovanie a počítanie instancí. Je obvyklým rodičom nových rozhraní.

nsIFile, nsILocalFile – slúžia na prístup k súborom a adresárom, ich vytváranie, mazanie, premenovanie, zisťovanie posledného prístupu, attribútov apod.

nsIPrefBranch – poskytuje prístup k užívateľským nastaveniam, tzv. preferences, dovoľuje ich čítanie aj zapisovanie. Bežne sa používa na prácu s užívateľskými nastaveniami.

nsIMessenger – obsahuje metódy na prácu so správami, prílohami a zložkami, napr. ich mazať, presúvať a ukladať.

nsIMsgFolder – zabezpečuje prístup a obsluhu zložiek. Medzi prístupné parametre priradenej zložky patrí názov, URI zložky, veľkosť na disku, rodičovská zložka a príznaky ako možnosť mazať správy, vytvárať podzložky apod. Obsahuje metódy na prácu so správami, ktoré sú v nej uložené.

nsIFolderListener – sleduje zmeny, ktoré sa udejú v priradenej zložke. Konkrétne ide o pridanie, odstránenie prvku, zmenu vlastností alebo nastanie udalosti.

nsIMsgDatabase – interná databáza správ. Umožňuje zistiť rôzne informácie o správach, napr. či obsahujú prílohy, boli prečítané apod.

nsIMsgDBHdr – poskytuje rozhranie ku konrétnej správe. Obsahuje informácie ako autor, dátum prijatia, kľúč správy, predmet a metódy na ich úpravu. Thunderbird každej správe pomocou tohto rozhrania nastavuje príznaky podľa jej aktuálneho stavu. Na ich základe je možné zistiť to, či je správa nová v adresári od jeho posledného otvorenia, či užívateľ na ňu odoslal odpoveď, či ju preposlal ďalej apod.

nsISimpleEnumerator – poskytuje rozhranie enumerátora, ktorý sa podobá doprednému iterátoru. Má len dve metódy a to na zistenie, či je ešte prístupný ďalší prvok a na prístup k tomuto prvku.

nsIMsgHeaderSink – rozhranie na priebežné kontrolovanie stavu sťahovania správy.

Kapitola 4

Návrh modulu

Návrh modulu sa skladá z niekoľkých častí. Najprv sú popísané jednotlivé možnosti kontroly, následne riešenie okna s nastaveniami a nakoniec AVG XPCOM komponent. Komponent je založený na objektovom návrhu s využitím dedičnosti a programovacieho jazyka C++ a skriptová časť má skôr podobu modulárneho návrhu, avšak s určitými prvkami objektovosti, čo je dosiahnuteľné vlastnosťami JavaScriptu [3].

4.1 Manuálna kontrola

Manuálna kontrola je vyvolávaná užívateľom, preto mu musí byť ľahko a intuitívne dostupná. Voľba padla na umiestnenie tejto možnosti do kontextového menu, ktoré sa vyvoláva kliknutím pravým tlačítkom myši do oblasti okna s prílohami. Táto oblasť sa vytvára v dolnej časti okna za predpokladu, že správa obsahuje aspoň jednu prílohu.

Toto umiestnenie by malo byť logické aj z pohľadu užívateľa, kedže v tomto menu sa nachádzajú aj všetky bežne vykonateľné operácie, ktoré ponúka e-mailový klient. To znamená, že táto možnosť bude umiestnená pri voľbách ako uloženie, odpojenie a odstránenie príloh, to môže prispieť bezpečnosti z hľadiska toho, že užívateľ bude mať možnosť kontroly na očiach a môže si zvyknúť ju preventívne používať.

Veľký počet správ obsahuje viac ako jednu prílohu, preto by bolo maximálne neefektívne môcť kontrolovať prílohy len po jednej. Na uľahčenie kontroly viacerých príloh existujú dve možnosti. Prvá možnosť je označiť viac príloh pomocou držania klávesy CRTL a kliknutím na konkrétne prílohy a následne zvolenie kontroly správ. Nájdu sa prípady, kedy by ani toto riešenie nebolo ideálne, preto je zaradená ešte možnosť skontrolovať všetky prílohy. Táto sa vyvolá jednoduchým zvolením možnosti *skontroluj všetky s AVG*.

Po vykonaní kontroly bude užívateľovi ukázané okno s výsledkami kontroly. Bude v ňom uvedený názov konta, pod ktorým je v danej chvíli e-mail uložený, meno odosielateľa, predmet správy a výpis kontrolovaných príloh s výsledkom kontroly. V prípade nájdenia infekcie bude užívateľovi ponúknuté automatické odstránenie príloh pri kliknutí na tlačítko OK.

4.2 Kontrola doručenej pošty

Podstatným aspektom pri kontrole doručenej pošty je jej včasnosť. Z toho vyplýva, že musí byť zahájená čo najskôr, aby odhalila infekciu skôr, ako by ju mohol užívateľ aktivovať.

Aby bolo dosiahnuté dané správanie, je potrebné detekovať doručenú správu v reálnom čase, v momente jej prijatia. Na základe tohto bude v systéme objekt, ktorý bude slúžiť na spracovanie udalosti, ktorá informuje o doručení novej správy. Spracovanie tejto udalosti bude spočívať v overení, či daná správa obsahuje nejaké prílohy, pretože ak neobsahuje prílohy, tak je jej obsah len čisto textový a môžeme ho považovať za bezpečný. Ak máme správu s prílohami, je potrebné overiť, či je to správa nová. Toto overenie sa skladá zo zistenia dvoch faktov. Prvým je to, že správa má nastavený atribút *neprečítaná* a druhým je porovnanie jej príznakov, aby sa zistilo, či je *nová od posledného otvorenia zložky*.

Ak máme novú správu s pripojenými prílohami tak, aby sme ich mohli skontrolovať, tak ich musíme uložiť na disk. Toto by sa dalo vykonať jednou metódou objektu s rozhraním nsIMessenger, avšak k tomu by bolo potrebné mať o prílohách podrobné informácie, ako ich *obsahový typ (content type), URL, zobrazované meno* a *URI správy*. Kedže tieto informácie nie sú dostupné, bude sa musieť uloženie príloh vykonať iným spôsobom. Ako vhodné riešenie sa javí načítanie správy v pozadí, čím sa aktivuje spracovanie príloh, ktoré bude nahradené našou implementáciou, ktorá dané prílohy uloží na disk a v operačnej pamäti si uchová potrebné informácie, ktoré budú potrebné na ďalšie spracovanie.

Po uložení príloh nasleduje ich kontrola pomocou antivírového systému AVG. O každej infikovanej prílohe sa budú ukladať informácie a popis infekcie.

4.3 Operácie s infikovanou správou

Po nájdení infekcie je potreba na ňu reagovať. V požiadavkách boli obsiahnuté možnosti, ktoré by malo rozšírenie implementovať. Teraz prejdeme ku konkrétnemu návrhu každého z nich.

Okno s informáciami o nájdenej infekcii je možné zobraziť vždy, preto bude jeho zapnutie riešené zaškrtávacím políčkom. Keďže jeho funkčnosť je rovnaká ako má okno, ktoré sa zobrazuje pri manuálnej kontrole, budú spolu zdielať implementáciu. Jediný rozdiel bude v tom, že následkom stlačenia tlačidla OK nebude odstránenie príloh, lebo toto sa bude riešiť samostatne v ďalších operáciach a mohlo by to zbytočne miasť užívateľa.

Vytvorenie poznámky v predmete správy nezasahuje výraznejšie do jej štruktúry, preto bude tiež možné to zvoliť zaškrtávacím políčkom v nastaveniach. Na zobrazenie zmeneného predmetu stačí modifikovať správu pomocou rozhrania *nsIMsgDBHdr*, kde sa zmení vlastnosť *subject*. Táto zmena sa vždy udeje len v pamäti a nie je zapísaná do súboru so správou na disku. Na zápis nie je prístupná žiadna metóda, takže je pravdepodobné, že sa zmena stratí pri operáciách, ktoré pozmenia správu a následne ju znovu načítajú z disku. To by sa dalo riešiť spôsobom, ktorý bude predstavený v nasledujúcom odstavci.

Podobnou zmenou ako je zmena predmetu správy, je aj pridanie X-header tagu do hlavičky maily. Avšak žiadne rozhranie neponúka možnosť pridať svoje vlastné informácie do hlavičky a ako bolo spomenuté, nie je dostupná žiadna funkcia, ktorá by dovolila následne tieto zmeny zapísať do správy na disku. Riešenie je možné urobiť implementáciou vlastného prepisu správy. Tento proces bol otestovaný a nepriniesol uspokojivé výsledky, takže táto možnosť nebola do návrhu zahrnutá. Problém vlastnej implementácie spočíva v tom, že v prvom kroku sa musí celá správa uložiť pomocou streameru na disk. Pri procese ukladania sa správa takisto ukladá do operačnej pamäte. Následne sa správa môže v pamäti upraviť a zapísať na disk. Stará správa už týmto stráca zmysel a nie je aktuálna, teda sa vymaže, a vytvorí sa nová správa na zakláde obsahu uloženého na disku. Tento proces je časovo náročný a namiesto úpravy starej správy vytvára novú, ktorá má nový kľúč a ďalšie vlastnosti, a z tohto dôvodu je krajne nevyhovujúci.

Zostáva navrhnúť presúvanie správ do špeciálneho adresára, odstraňovanie príloh a mazanie celých správ. Tieto operácie zasahujú pomerne významne do štruktúry správy, resp. nemá zmysel ich kombinovať. V dôsledku daných faktov bude možné voliť ich pomocou prepínacích tlačítok, takže vždy bude zvolená minimálne jedna možnosť. Implicitne to bude presun do špeciálnej zložky. Je to z principiálnych dovôdov, lebo táto možnosť umožní užívateľovi so správou ďalej manipulovať, avšak s vedomím, že obsahuje nebezpečné prílohy.

Na presunutie správy do špeciálnej zložky je logicky potrebné túto zložku mať. Názov zložky bude „*AVG infected*“. Pri štarte Thunderbirdu sa overí, či už daná zložka existuje, ak nie, tak sa vytvorí. Pri presúvaní správy sa použije instancia rozhrania *nsIMsgCopyService*.

Odstránenie príloh je veľmi jednoduché, pretože vďaka predchádzajúcemu spracovaniu príloh máme o nich uložené všetky potrebné údaje. Stačí použiť globálny object *messenger*, ktorý poskytuje rozhranie *nsIMsgMessenger* a obsahuje metódu *detachAllAttachments()*. Neskôr bude treba zvážiť, či sa táto možnosť neoznačí ako experimentálna, nakoľko v Bugzille pre Thunderbird je ohlásených niekoľko chýb pri odstraňovaní príloh v správoch spravovaných protokolom IMAP. Ďalšou nevýhodou tohto riešenia je, že táto metóda vyvoláva dialóg, kde užívateľ musí odsúhlasiť odstránenie príloh. Toto by určite bolo pre užívateľa nepríjemné a dezorientujúce. Riešením je pridanie parametru metódy, ktorý by umožnil odstránenie príloh na pozadí bez výzvy pre užívateľa. Pri pohľade na zdrojový kód Thunderbirdu [19] bude postačovať ak sa zamedzí volaniu funkcie, ktorá tento dialóg volá. Bude potrebné vytvoriť patch pre Thunderbird a dúfať, že ho vývojári prijmú.

Samotné zmazanie správy bude tvoriť veľmi jednoduchý proces, nakoľko mazanie správy je bežná činnosť. Na vykonanie bude potrebné získať instaciu zložky (nsIMsgFolder), v ktorej je uložená správa. Toto bude možné z uloženej databázovej hlavičky správy (rozhranie nsIMsgDBHdr). Následne sa použije metóda *deleteMessages()*.

4.4 Kontrola odosielanej pošty

Existuje viac možností ako aktivovať odosielanie pošty:

- kliknutie na tlačítko v paneli s nástrojmi
- použitie klávesovej skratky
- voľba v menu

Pri všetkých možných kombináciách je potrebné toto odosielanie prerušiť, vykonať kontrolu a v prípade negatívneho výsledku odoslať správu.

Na začiatku kontroly správy sa musí overiť, či správa obsahuje nejaké prílohy, lebo ak nie, tak nie je potrebné vykonať kontrolu a správa sa môže odoslať. Inak sa získajú objekty popisujúce súbory priložené k správe, ktoré okrem iného obsahujú aj cesty k daným súborom, ktoré sú potrebné pre metódu z AVG API na testovanie.

Ak sa nájde medzi prílohami infekcia, tak sa užívateľovi otvorí okno. Okno bude obsahovať infikované súbory s popisom infekcie. Užívateľ bude mať prístupné tlačítko späť na vrátenie sa k vytváraniu správy a tlačítko odoslať na odoslanie správy. Implicitne sa správa odošle bez infikovaných súborov, lebo tieto budú automaticky vybrané na odstránenie z odosielanej správy pred odoslaním. Toto bude môcť užívateľ zmeniť pomocou odškrtnutia konkrétnych zaškrtávacích políčok.

4.5 Štruktúra okna s nastaveniami

Okno sa rozdelí do záložiek podľa obsahu, ktorý budú jednotlivé záložky obsahovať. V prvej a zároveň aj hlavnej záložke bude možné kompletné zapnutie resp. vypnutie rozšírenia alebo len konkrétnej časti, teda kontroly prichádzajúcej, resp. odchádzajúcej pošty.

V ďalšej záložke bude umiestnené nastavenie kontroly doručenej pošty. Užívateľ si bude môcť nakonfigurovať kontrolu podľa svojej vôle, pritom však budú dodržané obmedzenia spomenuté skôr v tomto návrhu 4.3. Pre užívateľskú prívetivosť sa dialóg bude správať z časti dynamicky. Dynamickosť bude spočívať v znemožnení upravovať poznámku v predmete infikovanej správy, ak nebude povolené túto poznámku robiť, a presúvanie príloh do antivírovej truhly bude umožnené povoliť len ak bude zvolená možnosť odstraňovania infikovaných príloh.

Posledná záložka bude obsahovať výpis informácií o nainštalovanej verzii Thunderbirdu, aktuálne používanej vírusovej databázi a kernelu AVG.

4.6 AVG XPCOM komponent

Z dôvodu novej verzie systému AVG, konktrétne AVG 8, a zároveň snahy podporovať aj staršiu verziu AVG 7, bude potrebné vytvoriť 2 XPCOM komponenty. Kedže jednotlivé AVG API oboch verzií sa líšia veľmi málo, budú implementácie oboch verzií veľmi podobné. Dôležitou zmenou bude hlavne ContractID jednotlivých komponentov, aby boli od seba rozlíšiteľné.

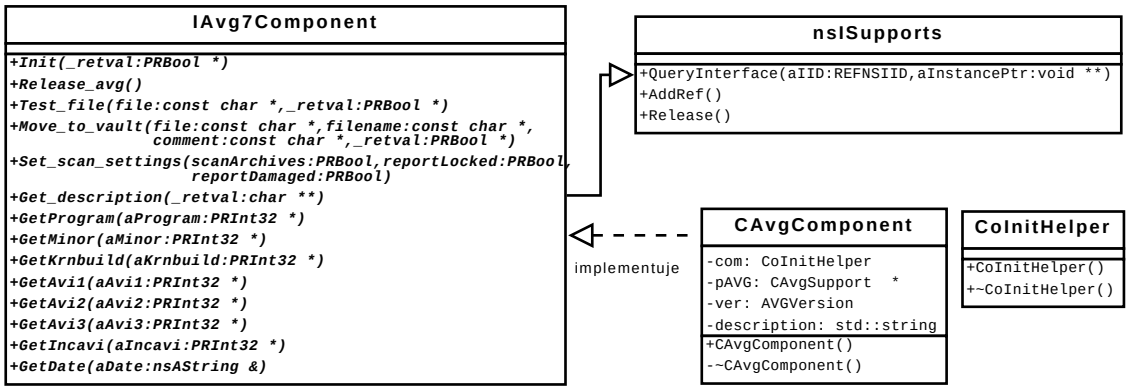
Na vývoj komponentu bude použité grafické vývojové prostredie Microsoft Visual Studio 2005 a jazyk C++. Projekt bude typu windows DLL. Rozhranie pre Mozillu popisuje idl súbor. Oba komponenty budú mať deklarované pomocou jazyka IDL nasledujúce metódy:

- `init()` – inicializuje AVG
- `release_avg()` – uvoľní AVG
- `test_file()` – otestuje súbor
- `move_to_vault()` – presunie súbor do vírovej truhly
- `set_scan_settings()` – nastaví parametre testovania
- `get_description()` – dá popis posledne nájdenej infekcie

Okrem metód komponentu poskytujú údaje o aktuálnej verzií AVG vo forme verejných atribútov, ktoré je možné len čítať. To je dosiahnuté pridaním modifikátoru `readonly` v idl deklarácií. Sú to tieto: `program, minor, krnbuild, avi1, avi2, avi3, incavi` a `date`. Date je dátum poslednej aktualizácie vo forme textového reťazca.

Na základe idl súboru je vygenerovaný hlavičkový súbor, ktorý obsahuje deklaráciu rozhrania. Vytvorený objektový návrh je znázornený na 4.1. Môžeme si všimnúť zmeny v názve metód, zatiaľ čo v idl boli deklarované s malým písmenom, pri implementácií budú začínať veľkým. Takisto pre readonly parametre sú vytvorené tzv. *get* metódy. Tým, že sme použili modifikátor readonly, nie je možné do parametrov zapisovať a preto nebolo potrebné ani žiadúce vytvoriť *set* metódy. Takéto riešenie pripomína moderný prístup vo forme vlastností v jazyku C#.

Z objektového návrhu je zrejmé, že naše rozhranie `IAvg7Component` zdedí zo základného rozhrania `nsISupports` metódy, ktoré zabezpečujú elementárne operácie, sú virtuálne a



Obrázek 4.1: Objektový návrh XPCOM AVG komponentu

mohli by sa predefinovať, ale to nebude potrebné. Metódy `AddRef()` a `Release()` slúžia na počítanie vytvorených referencií na daný komponent a tým riadia jej životný cyklus. `QueryInterface()` je metóda na zistenie, či daný komponent podporuje žiadané rozhranie. Jej potreba vyplýva z toho, že niektoré komponenty môžu implementovať viac rozhraní. Naše rozhranie je abstraktná trieda, lebo všetky metódy, ktoré obsahuje sú tiež abstraktné, takže ich bude potrebné implementovať. Implementácia rozhrania bude obsiahnutá v triede `CAvgComponent`.

Pre jednoduchý prístup k rozhraniu AVG bude vytvorený javascriptový objekt, ktorý si podľa nainštalovanej verzií AVG inicializuje správnu verziu. Tento objekt zabezpečí automatickú inicializáciu a uvoľňovanie AVG kernelu, takže ostatné časti rozšírenia budú využívať metódy na kontrolu súborov, presun do antivírovej truhly atď. bez toho, aby rozlišovali, aká verzia AVG je nainštalovaná a bez toho, aby sa starali o inicializáciu a uvoľňovanie.

Kapitola 5

Implementácia

Z dôvodu, že som doteraz nemal žiadne skúsenosti s vývojom na platforme Mozilla, nebol vývoj úplne triviálny a prinášal so sebou rôzne problémy. Výhodou je dostatočné množstvo aplikácií a rozšírení, ktoré sú vyvíjané pod rovnakými licenciami ako Mozilla, takže je možné dostať sa k zdrojovým kódom a zistiť, ako by sa dali niektoré konštrukcie naprogramovať. Ja som pri vývoji študoval a nechal sa z časti inšpirovať od [6] a [12].

5.1 Grafické užívateľské rozhranie

Grafické užívateľské rozhranie je rozdelené do niekoľkých súborov. V nasledujúcich odstavcoch budú popísané funkcie hlavných častí. Súbory *about.xul* a *error.xul* implementujú malé dialógy, pričom about.xul zobrazí krátke info o rozšírení a error.xul upozorňuje užívateľa, ak nebolo nájdené nainštalované AVG.

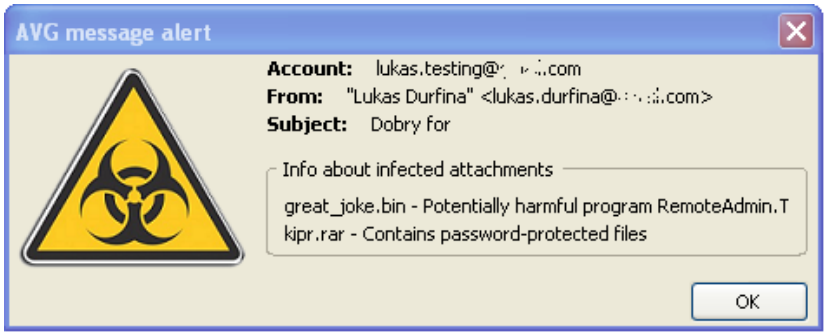
composeOverlay.xul

Vytvára overlay vrstvu nad oknom vytvárania novej správy. Pomocou značiek `command` a `key` prekryje v Thunderbirde implementované príkazy `cmd_sendButton, cmd_sendNow, cmd_sendLater, cmd_sendWithCheck` a dve klávesové skratky `key_send, key_sendLater`. Tieto prepíše vlastnou funkciou `doScanAndCommand(cmd)`, kde `cmd` je pôvodný príkaz. Táto funkcia zahajuje kontrolu príloh a podľa návratovej hodnoty buď zavolá alebo nezavolá príkaz predaný parametrom `cmd`.

infectionDialog.xul a outgoingDialog.xul

Implementácia týchto súborov je veľmi podobná, vyplýva to z ich podobného účelu. Oba implementujú informačný dialóg, ktorý obsahuje informácie o nájdených infekciách v testovaných súboroch. Rozdiel je v tom, že infectionDialog.xul môže byť čiste informačný dialóg (obr. 5.1), ktorý sa využíva na oznamovanie detekovaných infekcií pri kontrole doručenej pošty. Obsahuje len značky `label`, ktoré obsahujú informatívne textové reťazce a tlačítko OK implementované značkou `button`, ktoré uzatvára daný dialóg. Zároveň môže byť aj interaktívny, podobne ako dialóg popísaný v nasledujúcom odstavci, a to v prípade, že je používaný na oznamovanie z manuálneho testovania, s rozdielom, že označené prílohy odstraňuje.

Dialóg implementovaný súborom outgoingDialog.xul je o niečo zaujímavejší, pretože prináša viac možností pre užívateľa. Tento dialóg sa používa pri nájdení infikovaných príloh



Obrázek 5.1: Výstražný dialóg s popisom infekcie v doručenej správe

v správach, ktoré sú odosielané. Na rozdiel od predchádzajúceho dialógu je možné ovplyvniť, ktoré prílohy budú odoslané napriek ich pozitívnemu testu. Pri každej infikovanej prílohe je zaškrtávacie políčko `checkbox`, implicitne sa nastavuje vlastnosť `checked` na true, čím sa vyjadruje, že táto príloha nebude odoslaná. Užívateľ to môže zmeniť. Na dialógu sa nachádzajú 2 tlačítka. Tlačítko *Späť* umožňuje vrátenie sa k editácii správy a tlačítko *Odoslať* odošle danú správu, pričom označené prílohy pred samotným odoslaním odstráni zo zoznamu príloh tzv. `attachmentBucket` a tým pádom sa neodošlú.

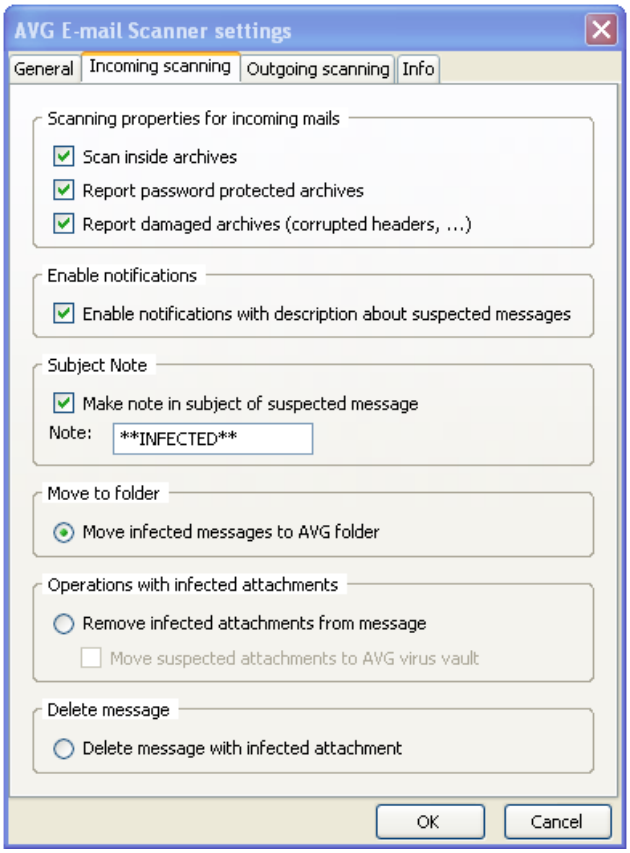
mainOverlay.xul

Obsahuje hlavnú overlay vrstvu rozšírenia. Vkladá všetky podstatné javascriptové súbory, čím spája jednotlivé skripty. V rámci grafického rozhrania pridáva položku do menu *Nástroje*, ktorá otvára dialóg s nastaveniami rozšírenia. Tieto nastavenia je možné vyvolať aj cez ikonu `toolbarbutton` v paneli nástrojov `toolbarpalette`, ktorú takisto poskytuje tento súbor. Okrem toho pridáva dve položky do kontextového menu `attachmentListContext`. Toto menu sa zobrazuje kliknutím pravým tlačítko myši v časti okna so zobrazenými prílohami. Tieto položky volajú kontrolu označených alebo všetkých zobrazených príloh a sú implementované značkou `menuitem`.

settings.xul

V tomto súbore je implementované okno s nastaveniami rozšírenia. Základ okna tvorí `tabbox`, v ktorom sú jednotlivé záložky. Záložky sa najprv deklarujú značkou `tab`, ktoré sú zoskupené medzi ohraničené značkou `tabs`. Následne je uvedená implementácia začínajúca značkou `tabpanel`. Poradie záložiek musí byť zhodné s poradím v deklarácii. Väčšina nastavení je implementovaných ako zaškrtávacie políčko `checkbox`. Výnimkou je iba výber operácie s infikovanou správou. Táto je implementovaná ako výberové tlačítko. Všetky výberové tlačítka musia byť obsiahnuté v rovnakej skupine `radiogroup` a jednotlivé tlačítka sú vyznačené značkou `radio`.

Na obrázku 5.2 je zobrazená podoba najrozsiahlejšej záložky, ktorá obsahuje vyššie popísané prvky.



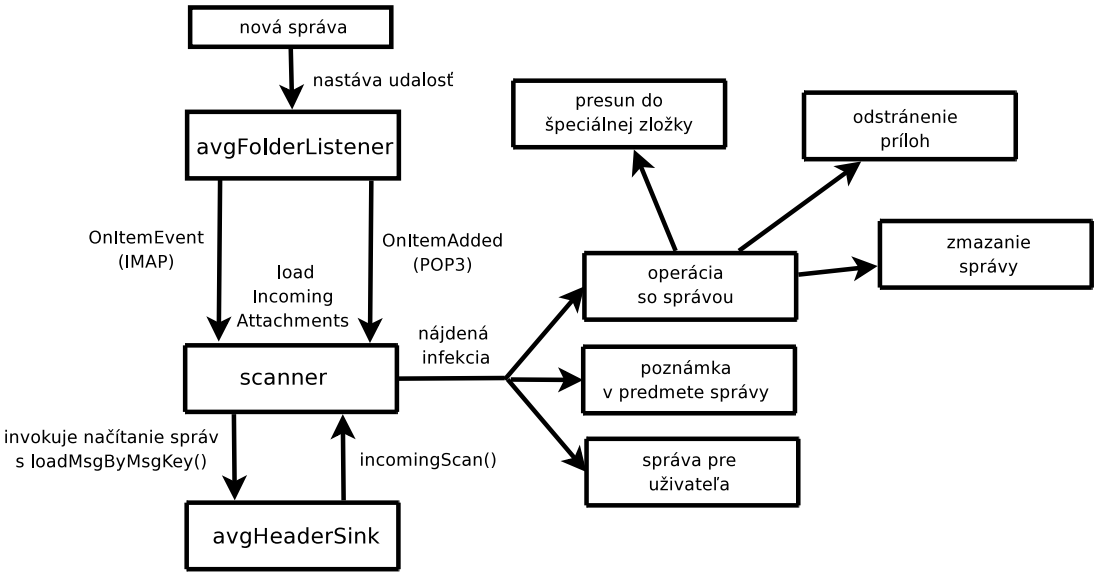
Obrázek 5.2: Záložka s nastaveniami kontroly doručenej pošty

5.2 Aplikačná logika

Aplikačná logika je vytváraná v skriptovacom jazyku JavaScript. Vďaka jeho beztypovosti je uľahčená práca s predávaním parametrov. Dobré prepojenie s jazykom XUL umožňuje reagovať na rôzne udalosti. Väčšina premenných, ktoré sú uzko spojené s obsluhou konkrétneho okna, využíva udalosť `onload` daného okna k vyvolaniu inicializácie danej premennej. Niektoré premenné pripomínajú objekty, lebo obsahujú vlastnosti aj vlastné funkcie. Avšak tým, že sú ako vytvárané pomocou *var* sa stávajú jedinečnými premennými a približujú sa *singletonu*, ale nemajú niektoré vlastnosti objektu ako modifikátory prístupu. Napriek tomu budem tieto premenné pomenovávať ako objektové premenné.

Dobrým pomocníkom pri vývoji skriptov bolo rozšírenie [8]. Vďaka nemu sa dali rýchlo odhaliť syntaktické chyby v kóde a vypisovať kontrolné výpisy, čo bolo určite pohodlnejšie ako používanie príkazu `alert()`. Vo Firefoxe je podobný nástroj nazvaný Error Console. V Thunderbirde je možné takéto výpisy presmerovať len do príkazového riadku, z ktorého sa Thunderbird spustí. Toto je veľmi nepohodlné a neposkytuje rozširujúce funkcie ako vyhľadávanie alebo filtrovanie výpisov.

V nasledujúcich odstavcoch budú popísané súbory so skriptami, ktoré riadia rozšírenie. Najzložitejší proces je kontrola doručenej pošty. Tento proces je zobrazený na diagrame 5.3. Je na ňom zobrazené previazanie medzi jednotlivými časťami rozšírenia a smer ich komunikácie.



Obrázek 5.3: Proces kontroly doručenej pošty

avg.js

Prostredníctvom objektovej premennej `avg` poskytuje prístup k AVG. Zapuzdruje rozlíšenie staršej a novšej verzii AVG. Toto vykonáva tak, že sa pokúsi inicializovať novšie AVG 8, ak sa toto nepodarí prebehne pokus o inicializáciu AVG 7 a ak je aj toto neúspešné, tak sa vypíše chybová správa a poznamená sa to do vlastností `inited` a `isOK`.

Objektová premenná poskytuje niekoľko metód, ktoré v podstate kopírujú možnosti ponúkané AVG XPCOM komponentom. Sú to tieto:

- `avgInit()` – inicializácia AVG kernelu
- `avgRelease()` – uvoľnenie AVG kernelu
- `avgIsOk()` – vracia true, ak je AVG inicializované a v poriadku
- `avgTestFile(path)` – testuje súbor podľa `path`
- `avgSetScanSettings(scanArchives, reportLocked, reportDamaged)` – nadstaví parametre pre kontrolu súborov, kontrétne možnosť testovať súbory vo vnútri archívov, hlásenie zamknutých a poškodených archívov.
- `avgMoveToVault(path, name, comment)` – presunie súbor podľa `path` do antivírovej truhly, môže špecifikovať meno pomocou `name`, ak by napr. išlo o dočasný súbor apod. a nakoniec pridať komentár k súboru cez parameter `comment`
- `avgGetDescription()` – vracia popis posledne nájdenej infekcie
- ďalšie funkcie typu *get*, ktoré poskytujú verejné vlastnosti AVG komponentu

avgFolderListener.js

V tomto súbore je vytvorená vlastná implementácia rozhrania *nsIFolderListener*. Implementuje metódy, ktoré reagujú na udalosti *added, removed, propertyChanged, intPropertyChanged, boolPropertyChanged, propertyFlagChanged* a *event*. Nás však budú zaujímať len udalosti added a event, ktoré budú obslúžené v rutinách:

- `OnItemAdded(parentItem, item, view)` – `parentItem` je instancia `nsIMsgFolder` tj. zložky, kde nastala udalosť, `item` je instancia `nsIMsgDBHdr` a `view` nevyužívam. V tejto udalosti je obslúžené prijatie novej správy do zložky od POP3 serveru.
- `OnItemEvent(folder, event)` – `folder` je instancia `nsIMsgFolder` tj. zložky, kde nastala udalosť a `event` je typ udalosti, ktorý nastal. Nás bude zaujímať iba udalosť *FolderLoaded*. Táto obsluha bude kontrolovať doručenie novej správy do zložky pre IMAP server, kde pri doručení novej správy nie je vyvolávaná udalosť *added*, keďže sa sťahuje len hlavička bez tela, ale sa vyvoláva načítanie zložky, aby sa nová správa objavila.

Implementácie oboch obslúh, ktoré sú si z časti podobné, zistia, či je povolená kontrola, ďalej zistia o správach, či sú nové a obsahujú prílohy. V obsluhe `OnItemEvent` je ešte pridaná kontrola, ktorá kontroluje a nastavuje príznak, či bola daná správa už testovaná. Na toto slúži v globálnom objekte `global` položka `testedMsg`. Význam tejto kontroly spočíva v zamedzení viacnásobných kontrol, pretože pri prijatí viacerých správ je udalosť `FolderLoaded` vyvolaná viackrát.

avgMsgHeaderSink.js

Obsahuje vlastnú implementáciu rozhrania *nsIMsgHeaderSink*. Z tohoto rozhrania je pre nás dôležitá metóda `handleAttachment(contentType, url, displayName, uri, isExtAtt)`, ktorá je vyvolaná pri načítaní správy, aby spracovala všetky jej prílohy. V našej implementácii bude ukladať informácie o prílohách do globálnej premennej `global`, konkrétne zložky `attachmentInfo` a tieto prílohy bude ukladať na disk, aby mohli byť následne skontrolované.

Na ukladanie sa používa globálne prístupný objekt `messenger` implementujúci rozhranie `nsIMessenger` a jeho verejná metóda `saveAttachmentToFolder(contentType, url, displayName, uri, destFolder)`. Táto metóda príjma presne tie informácie o prílohe, ktoré sú aj ukladané. Jediná výminka je `destFolder` alias cieľový adresár sa neukladá, ale získava sa z `global.saveDir`. Táto metóda vracia objekt s rozhraním `nsILocalFile`, ktorý sa ukladá ku spomínaným údajom.

global.js

Implementuje globálny sklad, kde si jednotlivé skripty ukladajú údaje, pomocou ktorých prebieha výmena informácií. Úlohou tejto objektovej premennej je sprehľadniť prístup ku globálnym dátam. Vyhnúť sa globálnemu prístupu, bolo v podstate nemožné, keďže niektoré časti kódu priebehajú súčasne v samostatných vláknach. Obsah tvoria polia:

- `msgHeaders` – hlavičky správ
- `saveDir` – adresár na ukladanie príloh pre danú správu
- `attachmentInfo` – údaje o prílohách

- `attachmentCount` – počet príloh v správe
- `processingMsg` – obsahuje true, ak je daná správa práve spracovávaná
- `testedMsg` – ukladá, ktoré správy už boli testované, aby sa predišlo prípadnému dvojitému testovaniu

Všetky polia majú rovnaký spôsob indexovania pomocou kľúča správy, ktorý je jedinečný v aktuálnej databázi správ. To znamená, že na prístup k údajom týkajúcej sa konkrétnej správy, je potrebné si zistiť kľúč tejto správy a ten následne použiť ako index do požadovaného poľa. Prvkom poľa môže byť ďalšie pole, ako je to použité napríklad v `attachmentInfo`. Na vkladanie do vnútorných polí sa používajú metódy `push()` a `pop()`, čím simulujú zásobník.

scanner.js

Tento súbor obsahuje najväčšiu a najzložitejšiu objektovú premennú *scanner*. Metódou globálneho objektu `window` a jeho metódy `addEventListener` sa zaregistruje reakcia na udalosť `load`, takže pri spustení Thunderbirdu prebehne metóda `onLoad()` nášho scanneru, ktorá pridá náš `avgFolderListener` medzi ďalších príjmateľov udalostí v zložkách a nastaví príjmanie udalostí `added` a `event`. Následne sa vytvorí dočasný adresár v *temp* adresári operačného systému, kde si bude rozšírenie ukladať všetky pomocné súbory. Tento adresár sa aj s celým obsahom vymaže pri ukončení Thunderbirdu. Nakoniec sa naplánuje spustenie metódy `afterLoad()` o 500 ms.

Metóda `afterLoad()` sa spúšťa s oneskorením, aby sa zaistila inicializácia globálnej premennej Thunderbirdu `gDBView`, ak táto premenná nie je pri spustení metódy inicializovaná, naplánuje sa opätovné oneskorené spustenie a metóda sa preruší. V tejto metóde si rozšírenie vytvorí vlastné verzie globálnych objektov Thunderbirdu ako messenger, msgWindow, vytvorí si klon gDBView a nastaví, aby sa prílohy otváraných správ na pozadí spracovávali našim `avgHeaderSink`. Následne sa skontroluje, či existuje špeciálna zložka *AVG infected*, ak nie, tak sa vytvorí.

Metóda `manualScan(scanAll)` je zavolaná pri kontrole vyvolanej užívateľom. Prijíma parameter `scanAll`, ak je true, tak sa kontrolujú všetky aktuálne zobrazené prílohy, inak sa kontrolujú len vybraté. Metóda prílohy uloží na disk ako samostané súbory, lebo Thunderbird všetky prílohy aj text správy uchováva ako 1 súbor, a uloží si objekty popisujúce súbory s prílohami. Nakoniec naplánuje spustenie kontroly týchto správ metódou `scanFile()` s oneskorením 1000 ms, preto aby sa test nezačal skôr ako sa prílohy uložia. V metóde `scanFile()` sa otestujú všetky prílohy a ak sa nájde infekcia, tak je otvorený dialóg s popisom a možnosť tieto prílohy hneď odstrániť.

Úlohou metódy `loadIncomingAttachments(msgHdr)` je vytvoriť podadresár v dočasnom adresári rozšírenia, kde sa budú ukladať prílohy zo správy odpovedajúcej `msgHdr` a následne otvoriť správu na pozadí, čím sa zahajuje ukladanie príloh v našom `avgHeaderSink`.

Podstatná metóda je `incomingScan()`. Táto metóda je volaná po uložení všetkých príloh v `avgHeaderSink`. Z dôvodu, že nevieme koľko príloh je v správe, tak je volaná po uložení každej, preto na začiatku metódy sa kontroluje koľká príloha v poradí ju volala a porovnáva sa to s globálne počítaným počtom príloh, čím je možné určiť, kedy bola volaná od posledne uloženej prílohy. Následne prebehne kontrola príloh metódou `scanFileQueitly(path)`, ktorá vracia popis infekcie alebo prázdny reťazec. V prípade nájdenia infekcie je vyhodnotené, ktoré operácie si užívateľ praje previesť a tieto sú postupne prevedené.

outgoingScanner.js

Zabezpečuje kontrolu odosielanej pošty. Rozdiel je hlavne to, že prílohy sú súbory uložené na disku, takže ich nie je potrebné ukladať. Stačí len získať ich handler, na čo slúži rozhranie `nsIFileProtocolHandler`. A ďalší postup procesu je identický s ostatnými druhmi kontroly.

infectionDialog.js

Hlavnou úlohou tohto skriptu je odstrániť problém, ktorý vznikol pri doručení viacerých správ s infikovanými prílohami. V prípade ak bol jeden výstražný dialóg zobrazený a pred jeho zavretím sa otvoril ďalší, tak prvý bol prepísaný a stratil svoju informačnú hodnotu. Súbor obsahuje objektovú premennú `infectionDialog`, ktorá má jedinú metódu `open()`, ktorá prijíma parametre s informáciami o správe, prílohách a možnosti interakcie s užívateľom. Metóda pred otvorením nového dialógu skontroluje, či už nejaký nie je otvorený. Ak existuje otvorený dialóg, tak si vstupné parametre uloží do polí. Zobrazenie týchto informácií prebehne po zatvorení pôvodného okna, lebo metóda skontroluje, či sa v poliach nachádzajú nezobrazené požiadavky a ak áno, tak zavolá sama seba s parametrami získanými z týchto polí.

settings.js

Tento súbor obsahuje objektovú premennú `settings`, ktorá obsluhuje nastavenie rozšírenia. Načítanie nastavení pri otvorení dialógu, nastavenie zaškrtávacích políčok apod. prebieha v metóde `onLoad()`. O uloženie nastavení po kliknutí na tlačítko OK sa stará metóda `saveConfig()`. Táto vo veľkej miere používa instanciu rozhrania `nsIPrefBranch`, ktoré slúži na ukladanie údajov rôznych typov. Pomocou tohto rozhrania je možné zapisovať a načítavať hodnoty popísané textovými reťazcami s hodnotovými typmi bool, int a char (reťazec). Východzie nastavenia sa ukladajú do súboru v adresári defaults/, nasledujúcim spôsobom:

```
pref("extensions.avg_email_scanner.moveToVault", false);
```

5.3 AVG XPCOM komponent

Rozhranie komponentu je popísané v súboroch avg7.idl a avg8.idl, pomocou nástroja *xpidl* sa vygeneruje pre každý hlavičkový súbor s deklaráciou novej virtuálnej triedy s popísaným rozhraním, ktoré dedí od `nsISupports`. Nakoniec obsahuje ukážku ako implementovať vlastnú triedu, ktorá bude implementovať vygenerované rozhranie. Táto ukážka je základom pre súbory avg{7,8}–impl.{cpp,h}, ktoré obsahujú deklaráciu a implementáciu jednotlivých tried. Na prepojenie s AVG slúži objekt COM, ktorý poskytuje AVG API.

Komponent používa niekoľko znakov pre jeho identifikáciu v systéme komponentov. Zvolil som reťazec, ktorý by mal na prvý pohľad dávať dostatok informácií o jeho poslaní. Zahrnul som do neho jeho pôvod a verziu aplikácie AVG, s ktorou dokáže spolupracovať, konečný výsledok je **„@avg.com/avg7component;1“**. Okrem toho identifikátora je ku komponentu priradené ClassName a IID, čo je celosvetove unikátne číslo pozostávajúce zo 128 bitov.

5.4 Testovanie

Testovanie priebehalo manuálne, keďže jeho súčasťou bolo odosielanie nebezpečného kódu, tak nebolo jednoduché nájsť spôsob, ako správu odoslať bez toho, aby bol detekovaný problém a následne odoslanie správy zablokované. Na začiatku som si pripravil jednoduchý program, ktorý som otestoval systémom AVG, aby som si bol istý, že ho korektne detekuje. Následne som zmenil príponu súboru z .exe na .jpg, čo malo zvýšiť krytie, našťastie AVG ho stále detekovalo správne. Na odosielanie slúžila webová aplikácia Gmailu, správy som posielal takisto na Gmail účty, pričom prístup k prvému bol pomocou POP3 a k druhému cez IMAP. Okrem tohoto súboru boli uskutočnené aj testy s archívmi, aby sa overila správnosť detekovania nebezpečneho kódu vo vnútri archívov alebo zamknutých archívov.

Neskôr som k testovaniu pripojil aj školský mailový server Eva, aby som si overil teóriu, že proces spracovania správ protokolom IMAP sa líši od použitého servera a teda ovplyvní aj úspešnosť testovania príloh modulom. Našťastie proces správneho stiahnutia a otestovania prílohy zlyhal veľmi zriedkavo. Problém spočíval v odstraňovaní prílohy zo správy, kde som pozoroval rozdielne správanie na oboch serveroch. Server Gmailu po pokuse odstrániť prílohu celú správu odstránil z inboxu, ale ponechal ďalej nepozmenenú na serveri a v položke „All mails“. Školský server bol v tomto smere uspešnejší, ale to len v prípade, že som mal v momente odstraňovania prílohy zobrazený obsah inboxu, kde sa táto správa nachádzala. Inak server nechal správu nepozmenenú a okrem nej vygeneroval jej kópiu s odstránenou prílohou.

Pozitívnym zistením je, že pri bežných malých prílohách je časový interval potrebný na kontrolu na priemernom súčasnom počítači dostatočne malý na to, aby si užívateľ žiadne oneskorenie ani nevšimol. Pri väčších prílohách, rádovo nad 500 kB je už oneskorenie poznateľné a lineárne sa zväčšuje s veľkosťou prílohy, avšak výkonové problémy so spracovaním väčších príloh má aj samotný Thunderbird.

Kapitola 6

Záver

Modul e-mailového klienta Thunderbird poskytujúci antivírovú kontrolu bol úspešne implementovaný a je pripravený na verejné používanie. Po ďalších uprávach pre lepšiu kompatibilitu s AVG8, pridaní podpory Antispamu a lokalizácií do hlavných svetových jazykov bude modul uvoľnený pre verejnosť. Rozšírenie prispieva k bezpečnejšej e-mailovej komunikácii. Hlavnou výhodou je kontrola správ cez zašifrované protokoly, ktoré bežné súčasti antivírových programov nie sú schopné kontrolovať, toto si však často užívateľ neuvedomuje.

Pozitívnym výsledkom je pomerne úspešné kontrolovanie správ prijatých cez IMAP protokol, lebo s implementáciou tohto zložitého protokolu bývajú problémy. Implementácia bola zameraná tak, aby bolo zlyhanie pri tomto protokole minimalizované, avšak hrozí v naozaj špecifických prípadoch, ktoré sa často ani nepodarí opäť reprodukovať. Protokol POP3 je o poznanie jednoduchší, preto kontrola je v jeho prípade bezproblémová.

Pre zlepšenie bezpečnosti a jeho rozšírenie bude potrebné sa ďalej starať o vývoj a opravu chýb. Najbližším míľnikom bude vydanie Thunderbirdu 3, plánované na 4. kvartál tohoto roku, kde nastanú zmeny a bude potrebné rozšírenie týmto zmenám prispôsobiť. Ďalším cieľom by mohlo byť vylepšenie odstraňovania príloh v IMAP zložkách, lebo toto pre niektoré IMAP servery zlyháva, ale to bude skôr otázka na vývojárov Thunderbirdu a vývojárov konkrétnych IMAP serverov. Pri IMAPe by sa mohlo rozlíšenie vylepšiť v tom smere, aby automaticky nesťahovalo správu s prílohou, ak nie je nastavená možnosť práce so zložkou v offline stave, ale aby kontrola prebehla vtedy, keď bude správa stiahnutá po požiadavke užívateľa.

Otvoreným zostáva pridávanie X-tagov do hlavičky správy, ktoré je momentálne na toľko problematické a neprináša očakávané prínosy, že nebolo implementované, ale možné riešenie snáď príde s novšími verziami Thunderbirdu, ktoré by mohli priniesť jednoduchšiu editáciu hlavičiek správy, hlavne ich zápis do súboru so správou na disku. Zaujímavou funkciou by mohlo byť pridanie možnosti liečenia infikovaného súboru. Aj keď toto liečenie je často neúspešné a užívateľ má možnosť sa pokúsiť dôležitý súbor vyliečiť tým, že ho uloží na disk a použije priamo systém AVG, mohla by byť táto funkcia prijatá kladne. Prínosom by mohlo byť aj odosielanie správ o doteraz neznámych infikáciách, ktoré boli detekované na základe heuristickej analýzy. Týmto by sa mohlo urýchliť ich spoznanie a zaradenie do vírusovej databázi.

Literatura

[1] Doug Turner a Ian Oeschger. *Creating XPCOM Components*. Brownhen Publishing, 2003. http://www.mozilla.org/projects/xpcom/book/cxc/ [cit. 2008-02-21].

[2] Tomáš Doseděl. *Počítačová bezpečnosť a ochrana dat*. Computer Press, 2004. ISBN 80-251-0106-1.

[3] Steven Holzner. *JavaScript profesionálně*. Mobil Media, 2003. ISBN 80-86593-40-1.

[4] Jan Kupčík. Aplikace na platformě mozilla. Master's thesis, FIT VUT v Brně, Brno, 2007.

[5] Nigel McFarlane. *Rapid Application Development with Mozilla*. Prentice Hall, 2003. ISBN 0-13-142343-6.

[6] WWW stránky. Attachment extractor. https://addons.mozilla.org/sk/thunderbird/addon/556. [cit. 2008-02-20].

[7] WWW stránky. AVG Technologies. http://www.avg.com/. [cit. 2008-02-18].

[8] WWW stránky. Console2. https://addons.mozilla.org/sk/firefox/addon/1815. [cit. 2008-02-21].

[9] WWW stránky. Mozilla code licensing. http://www.mozilla.org/MPL/. [cit. 2008-04-06].

[10] WWW stránky. Mozilla developer center. http://developer.mozilla.org/. [cit. 2008-02-21].

[11] WWW stránky. Mozilla Thunderbird. www.mozilla.com/thunderbird/. [cit. 2008-02-18].

[12] WWW stránky. Spamato. http://sourceforge.net/projects/spamato/. [cit. 2008-02-20].

[13] WWW stránky. XPCOM. http://www.mozilla.org/projects/xpcom/. [cit. 2008-02-21].

[14] WWW stránky. XPCOM referencia. http://www.xulplanet.com/references/xpcomref/. [cit. 2008-02-20].

[15] WWW stránky. XPIDL. http://developer.mozilla.org/en/docs/XPIDL. [cit. 2008-02-21].

[16] WWW stránky. XPInstall. http://www.mozilla.org/projects/xpinstall/. [cit. 2008-02-21].

[17] WWW stránky. XUL. http://www.mozilla.org/projects/xul/. [cit. 2008-02-21].

[18] WWW stránky. XUL referencia. http://www.xulplanet.com/references/elemref/. [cit. 2008-02-20].

[19] WWW stránky. Zdrojové súbory thunderbirdu. http://lxr.mozilla.org/mailnews/. [cit. 2008-02-20].

[20] AVG Technologies. *AVG API dokumentácia*, 2008.