VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

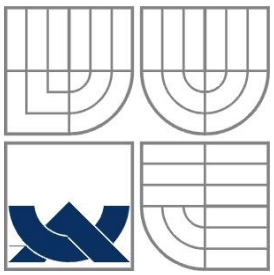
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

SYSTÉM PRO SLEDOVÁNÍ ZMĚN V OPERAČNÍM SYSTÉMU

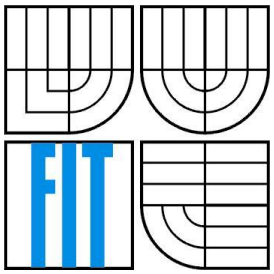
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAN PEČEŇA
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

SYSTÉM PRO SLEDOVÁNÍ ZMĚN V OPERAČNÍM SYSTÉMU

SYSTEM FOR LOGING CHANGES IN OPERATING SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAN PEČEŇA
AUTHOR

VEDOUCÍ PRÁCE — Dr. Ing. PETR PERINGER
SUPERVISOR

BRNO 2009

Abstrakt

Bakalářská práce pojednává o návrhu a implementaci aplikace pro zjišťování změn, způsobených instalací softwaru, v operačním systému. V úvodní části je rozebrána struktura registrů Windows, uložení hodnot v nich a informace o hashovacích funkcích, potřebných k rozpoznání změněných souborů. Vytvořená aplikace je plně funkční a pracuje s 32 i 64 bitovými verzemi systému Windows. Při prvním spuštění provede zálohu aktuálního stavu operačního systému a při druhém odhalí úpravy v operačním systému, instalovaném ve virtuálním počítači VMWare.

Abstract

Bachelor's thesis describes the design and implementation of applications for the detection of changes in the operating system. Window´s registry, registry key, subkey and value are explained in introductory part. Hash function needed to identify the changed files is explained too. The created application is fully functional and works with 32 and 64 bit versions of Windows. Actual state is saved in the first run. Changes in operating system are detected in the second run. Operating system is installed in a VMWare virtual machine.

Klíčová slova

Registr Windows, Hashovací funkce, MD5, Virtuální počítač, VMWare

Keywords

Windows registry, Hash function, MD5, Virtual machine, VMWare

Citace

Pečeňa Jan: Systém pro sledování změn v operačním systému, bakalářská práce, Brno, FIT VUT v Brně, 2009

Systém pro sledování změn v operačním systému

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením:
Dr. Ing. Petra Peringera, UITS FIT VUT.
Další informace mi poskytli:
Pavel Krčma, AVG,
Ing. Karel Obluk, Ph.D., AVG
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Pečeňa
10.5.2009

Poděkování

Děkuji vedoucímu bakalářské práce Dr. Ing. Petru Peringerovi za metodické vedení, pedagogickou a odbornou pomoc při zpracování mé bakalářské práce. Dále bych chtěl poděkovat Pavlu Krčmovi za poskytnutí odborných rad při tvorbě programu.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Cílem práce je navrhnout a vytvořit aplikaci pro firmu Grisoft. Firma rozšiřuje programy o části, které se vyvíjí odděleně. Aby se mohly do výsledného produktu jednotlivé části přidat, je potřeba vědět, jak ovlivňují operační systém. V současnosti se tyto změny musí odhalovat ručně, tj. procházením souborů a vyhledáváním změn, což je časově velmi náročné a neefektivní. Aplikace, kterou popisuje tato práce, má za úkol změny v operačním systému vyhledat a podat o nich zprávu.

Tento dokument je rozdělen do několika částí, které popisují řešení problému a následnou tvorbu celého systému. Úvod se zabývá použitou technikou při sledování změn v systému. Dále získáváme základní informace o virtuálním počítači VMWare a možnosti jeho ovládání z hostitelského počítače. Následně je rozebrán popis struktury registru Windows, způsob uložení dat v registru a možnosti čtení hodnot. Pro pochopení techniky odhalování změn u souborů je vysvětleno použití funkcí pro výpočet kontrolního součtu souboru a podrobnější informace o algoritmech MD5, SHA-1 a SHA-2.

Druhá část dokumentu popisuje návrh vytvořeného systému, implementaci jednotlivých součástí a obsahy souborů se zdrojovými kódy. Během vysvětlení obsahu zdrojových kódů jsou uvedeny možnosti úprav některých funkcí, např. pro použití jiné hashovací funkce nebo změnění formy výstupního tisku. Po části zabývající se implementací je vysvětlen způsob ovládání celého systému a práce s konfiguračním souborem. Na závěr jsou sděleny poznatky získané při testování systému a vysvětlena důležitá chybová hlášení.

Na médiu, které je přiložené k práci, jsou uložené krom souborů potřebných pro samotný běh aplikace i ukázky několika výstupů, získaných během testování. Zhlédnutím těchto příloh je možné vytvořit si představu o závislosti výstupu na nastavení konfiguračního souboru.

2 Přehled současného stavu

Při běhu operačního systému Windows dochází ke změnám, které jsou způsobené jednak instalací nových aplikací, ale i úpravou uživatelských profilů. Hlavní obměny jsou k vidění v registrech Windows, kdy se pro začlenění aplikace do systému vytvoří nebo upraví záznamy o nastavení, přístupových právech a o asociaci souborů. Další úpravy nastávají na pevném disku, kdy se během instalačního procesu vytvoří, upraví nebo vymažou soubory a adresáře.

V dnešní době na tuto vlastnost v systému Windows dohlíží aplikace "Obnovení systému," kdy si před každým upravením systému vytvoří tzv. bod obnovení, tj. uložení zálohy registrů a nastavení do složky začínající _restore v adresáři System Volume Information. Při pádu nebo vyskytujících se problémech s funkcí systému, je možné vrátit nastavení do tohoto bodu obnovení.

Na disku jsou data uložena v hierarchické stromové struktuře. Pro odhalení změn stačí určit, kde se adresářový strom změnil. Určit přidané a odstraněné soubory je snadné. Ovšem u upravených vzniká problém. Porovnávat soubor podle velikosti nebo času poslední změny nemusí být vždy optimálním řešením. Pokud se při ze souboru něco smaže a následně se data o stejné délce zase přidají, dojde sice ke změně, ale ve velikosti souboru se to neprojeví. Taktéž u porovnávání času poslední úpravy se může stát, že instalační proces pozmění čas směrem do minulosti. Optimální metodou se jeví porovnávat kontrolní součty souborů. Jaký vhodný algoritmus pro vytvoření součtu však použít. U MD5 algoritmu jsou sice známé kolize, ale ve srovnání s algoritmy SHA-1, resp. SHA-2 je rychlejší, což při porovnávání souborů v systému, kterých jsou řádově desetitisíce, hraje velkou roli. Více o funkcích sloužících k vytváření kontrolních součtů souborů je v kapitole 2.3.

Funkce pro odhalení změn v registrech může pracovat na podobném principu jako u souborů, registry jsou taktéž uspořádány do stromové struktury. Hodnoty je však výhodnější ukládat přímo jako konkrétní čísla nebo řetězce, ne jen jejich otisky, protože při následném porovnávání a vyhodnocování změn je vidět nejen co se změnilo, ale i jak se to změnilo. Struktura registrů a jednotlivé druhy hodnot v nich uložené jsou popsány v části 2.2.

2.1 VMWare

Společnost VMWare, Inc. je vývojářem virtualizačního softwaru pod názvem VMWare. Programy VMWare slouží pro obsluhu více virtuálních počítačů na jednom hostitelském stroji a lze je používat na platformách Microsoft Windows, Linux a Mac OS X.

Pro desktopová řešení je vhodná aplikace VMWare Workstation, umožňující vytváření a spouštění virtuálních počítačů. Pro serverové řešení je zdarma dostupná aplikace VMWare Server. Společnost vyvinula i další programy vhodné k virtualizaci počítačů, ty však nepodporují knihovnu

VIX API, pomocí které se dají virtuální počítače a systémy v nich nainstalované ovládat z hostitelského systému.

Knihovna VIX API obsahuje desítky funkcí pro ovládání VMWaru. Některé funkce pro svoji činnost vyžadují přihlášení do systému. Na začátku práce s virtuálním počítačem je potřeba si vytvořit dva handly, jeden pro spojení s aplikací VMWare a druhý pro spojení se samotným virtuálním počítačem. Na výsledek funkcí této knihovny se nemusí aktivně čekat, protože jsou připravené pro použití zpětného volání, kdy se při ukončení úkolu vyvolá obslužná funkce.

2.2 Registr Windows

Registr je definován jako centrální hierarchická databáze požívaná v operačních systémech Microsoft Windows 98, Windows CE, Windows NT a Windows 2000 a slouží k ukládání informací potřebných ke konfiguraci systému pro jednotlivé uživatele, aplikace a hardwarová zařízení.

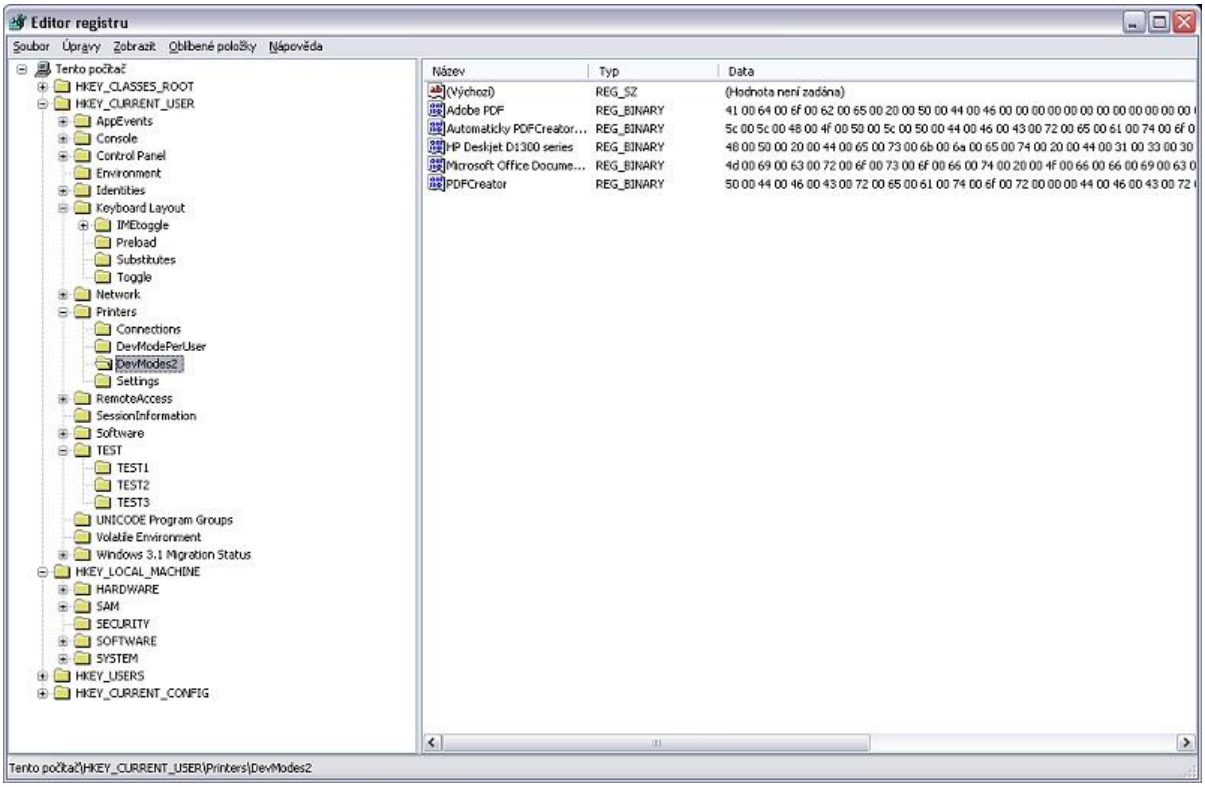
Data v registru obsahují informace o hardwaru existujícím v systému a používaných portech, dále registr obsahuje nastavení profilů jednotlivých uživatelů, aplikací nainstalovaných v systému a typech dokumentů, které mohou programy vytvářet. Při provedení jakýchkoliv změn v systému se změny projeví i v registru. Registr nahrazuje konfigurační soubory MSDOS, jako jsou Autoexec.bat a Config.sys.

Registry jsou jako databáze fyzicky uloženy na pevném disku v několika souborech, odkud se informace při startu systému natahují do paměti. Umístění souborů se liší použitým operačním systémem. Windows NT ukládá registry do šesti souborů, DEFAULT.DAT, SAM.DAT, SECURITY.DAT, SOFTWARE.DAT, SYSTEM.DAT a NTUSER.DAT, nacházejících se v adresáři %SystemRoot%\System32\Config.

Celý hierarchicky uspořádaný strom je možné prohlížet a editovat pomocí nejrůznějších aplikací. Mezi základní, vyskytující se v každém operačním systému Windows, patří aplikace "editor registru." Tato aplikace zobrazuje hodnoty, uložené v registrech, ve formátu vhodném pro čtení, umožňuje jejich editaci, vyhledávání a vkládání. Pro zálohování registrů nebo pro čtení klíčů a hodnot pomocí vlastního programu je vhodnější použít funkce implementované společností Microsoft.

Struktura registrů

Informace v registrech jsou hierarchicky uspořádány do stromové struktury, viz například obrázek 2.1. Hlavní stromovou strukturu představují větve (hives). Každá větev je tvořena skupinou klíčů (keys), podklíčů (subkeys) a hodnot (value), které obsahují data různých typů. Data v registrech jsou uložena v několika záložních kopiích.



Obrázek 2.1 - Výpis z editoru registru Windows

Větve registru (hives)

Větev registru je speciální klíč, jehož název předdefinuje operační systém. V každé větvi jsou uložena specifické data.

HKEY_CURRENT_USER obsahuje informace o aktuálně přihlášeném uživateli, jeho nastavení plochy, monitoru, ovládacích panelů a složek. Informace o aktuálním uživateli jsou získávány z HKEY_USERS.

HKEY_USERS obsahuje všechny aktivně načtené uživatelské profily v počítači.

HKEY_LOCAL_MACHINE obsahuje informace o hardwarových a softwarových konfiguracích daného počítače a informace týkající se bezpečnosti, uživatelských práv a sdílení.

HKEY_CLASSES_ROOT obsahuje informace potřebné pro běh aplikací, asociaci souborů, o názvech ovladačů, zkratkách Windows, atd.

HKEY_CURRENT_CONFIG obsahuje informace o aktuálním hardwarovém profilu používaném místním počítačem při spuštění.

Klíče a podklíče registrů (keys, subkeys)

Klíče a podklíče tvoří v registrech celou stromovou strukturu. Klíč, resp. podklíč, obsahuje další podklíče nebo hodnoty, neobsahuje však žádná konkrétní data. Maximální délka názvu klíče je 255 znaků. Registr v 64bitových verzích systému Windows je rozdělen do 32bitových a 64bitových klíčů.

32bitové klíče jsou v těchto 64 bitových verzích systému vyobrazeny pod uzlem HKEY_LOCAL_MACHINE\Software\WOW6432Node.

Hodnoty registrů (values)

Hodnoty klíče registru ukládají konkrétní data různého typu. Délka názvu hodnoty je u Windows NT 16383 znaků. Délka samotné hodnoty je u Windows NT omezena dostupnou pamětí, ovšem hodnoty větší než 2048 bajtů jsou pro zajištění lepší efektivity uloženy do souboru a v registru se vyskytuje pouze název souboru.

REG_BINARY	Binární hodnota. Nezpracovaná binární data. Použití pro uložení informací o hardwarových součástech.
REG_DWORD	Hodnota DWORD. 32bitové číslo uložené jako LITTLE_ENDIAN (nejnižší bajt je na nejnižší adrese) nebo BIG_ENDIAN.
REG_EXPAND_SZ	Rozšiřitelná řetězcová hodnota. Datový řetězec s proměnou délkou.
REG_MULTI_SZ	Víceřetězcová hodnota. Vícenásobný řetězec. Použití obvykle pro seznamy.
REG_SZ	Řetězcová hodnota. Textový řetězec s pevnou délkou.
REG_QWORD	Hodnota QWORD. 64bitové číslo.
REG_NONE	Není k dispozici. Data bez určitého typu.
REG_LINK	Odkaz. Řetězec udávající název symbolického odkazu.
REG_RESOURCE_LIST	Binární hodnota. Řada vnořených polí, do kterých se ukládá seznam prostředků používaný ovladačem hardwarových zařízení.

Tabulka 2.1 - Hodnoty registrů

2.3 Hashovací funkce

Hashovací funkce je metoda pro převod vstupních dat na výstupní řetězec pevné délky (hash, otisk). Délka výstupního řetězce závisí na zvoleném algoritmu. Mezi další vlastnosti funkce patří vysoká pravděpodobnost, že dvě zprávy se stejným otiskem jsou stejné. Algoritmus by měl být jednosměrný, kdy nelze jednoznačně určit původní zprávu z otisku.

Nevýhodou hashovací funkce je kolize. Jelikož výstupní řetězec je pevné délky a možnosti vstupních dat jsou neomezené, může vzniknout stejný hash pro dvě různé zprávy. Dobrá funkce by se měla snažit těchto kolizí vyvarovat.

Hashovací funkce se používají pro kontrolu integrity dat (testování shodnosti soborů bez nutnosti porovnávat celé soubory, stačí porovnat jen jejich hashe). Užití se nalezne také v kryptografii, pro předávání a ukládání otisků hesel. Mezi nejznámější algoritmy patří MD5, SHA-1 a SHA-2.

MD5 je hashovací funkce s otiskem o velikosti 128 bitů, je popsána ve standardu RFC 1321. MD5 hash je typicky reprezentován jako 32 ciferné hexadecimální číslo. Výpočet otisku řetězce nebo souboru je tvořen jedním průchodem. Bylo dokázáno, že není odolný kolizi, proto se nehodí pro použití v aplikacích, spoléhajících na tuto vlastnost.

SHA-1 vytváří otisk o velikosti 160 bitů. Algoritmus rozseká původní zprávu do bloků o velikosti 512 bitů a každý blok pak zpracovává samostatně pomocí vnitřní kompresní funkce, která má dva vstupy, 160 bitový výstup předchozího kroku (v prvním průchodu je požit předepsaný inicializační vektor) a 512 bitový blok nových dat. Tímto rozdělením se snižuje riziko vzniku kolizí mezi zprávami různých délek. Maximální velikost vstupní zprávy je 2^{64}-1 bitů, což je přes 2,3 TB. V roce 2005 byl objeven algoritmus schopný nalézt kolizi rychleji než hrubou silou, výpočetní a časová náročnost je ovšem stále vysoká.

SHA-2 definovaná standardem FIPS 180-2 je rodina hashovacích funkcí SHA-256, SHA-384, SHA-512 a SHA-224. Označení za SHA určuje délku výstupního hashe. Velikost bloků, na které funkce rozdělí původní zprávu, je u SHA-224/256 512 bitů, u SHA-384/512 1024 bitů. Délka bloku 1024 bitů dovoluje načítat vstupní zprávy o velikosti až 2^{128}-1 bitů. U těchto algoritmů zatím nebyly nalezeny bezpečnostní slabiny.

3 Návrh systému

Systém pro sledování změn v operačním systému bude mít za úkol vytvořit zálohu adresářového stromu a struktury registrů Windows. Jelikož bude aplikace používána při testování softwarových produktů, což znamená neustálé instalování neprověřených programů do systému, je vhodné toto testování provádět ve virtuálním počítači, dle požadavků zadávající firmy, virtuálním počítači VMWare Workstation 6.0 a vyšší nebo VMWare server 1.0 a vyšší. U těchto verzí je možné použít knihovnu VIX API. Jedna z hlavních výhod této knihovny je, že celý systém ve virtuálním počítači je možné ovládat z hostitelského systému, takže nebudou při vytváření zálohy adresářového stromu do zálohy zahrnuty žádné soubory navrhované aplikace.

Změny v operačním systému budou zjištěny ve dvou hlavních krocích. V prvním kroku aplikace archivuje adresářový strom a hodnoty v registrech Windows. V kroku druhém se tento archiv porovná s aktuálním stavem systému. Archivace bude spočívat v načtení všech adresářů a souborů v systému, dopočítáním kontrolního součtu k nim a uložením získaných informací do souboru. V případě registrů bude postup obdobný, jen místo adresářového stromu se načtou hodnoty klíčů registrů.

Po provedení úprav v operačním systému se opětovně spustí navrhovaná aplikace, vytvoří dočasnou zálohu a porovná ji s dříve vytvořenou. Změny vypíše do souboru nebo na standardní výstup. Nové a změněné soubory aplikace kopíruje z virtuálního počítače na určené místo v hostitelském systému.

V knihovně VIX se ovšem nenacházejí funkce pro výpočet kontrolních součtů souborů ani pro prácí s registry, je proto potřebné vytvořit pomocné aplikace, které budou spuštěny ve virtuálním systému a zajistí potřebné úkony, které nelze získat z hostitelského počítače.

Systém tedy bude tvořen třemi aplikacemi. První, s názvem scan.exe, bude řídit celý systém, ovládat virtuální počítač, vytvářet zálohu adresářového stromu, porovnávat a vyhodnocovat změny v zálohách a spouštět další dvě aplikace, get_md5.exe a registry.exe. Ty budou mít za úkol dopočítat kontrolní součty k souborům a vytvářet zálohu registrů do souboru.

Jelikož bude řada vstupů v navrhovaném systému měnitelná, je vhodné nastavení neprovádět pomocí příkazové řádky, ale použít konfigurační soubor.

Záloha adresářového stromu bude obsahovat informace:

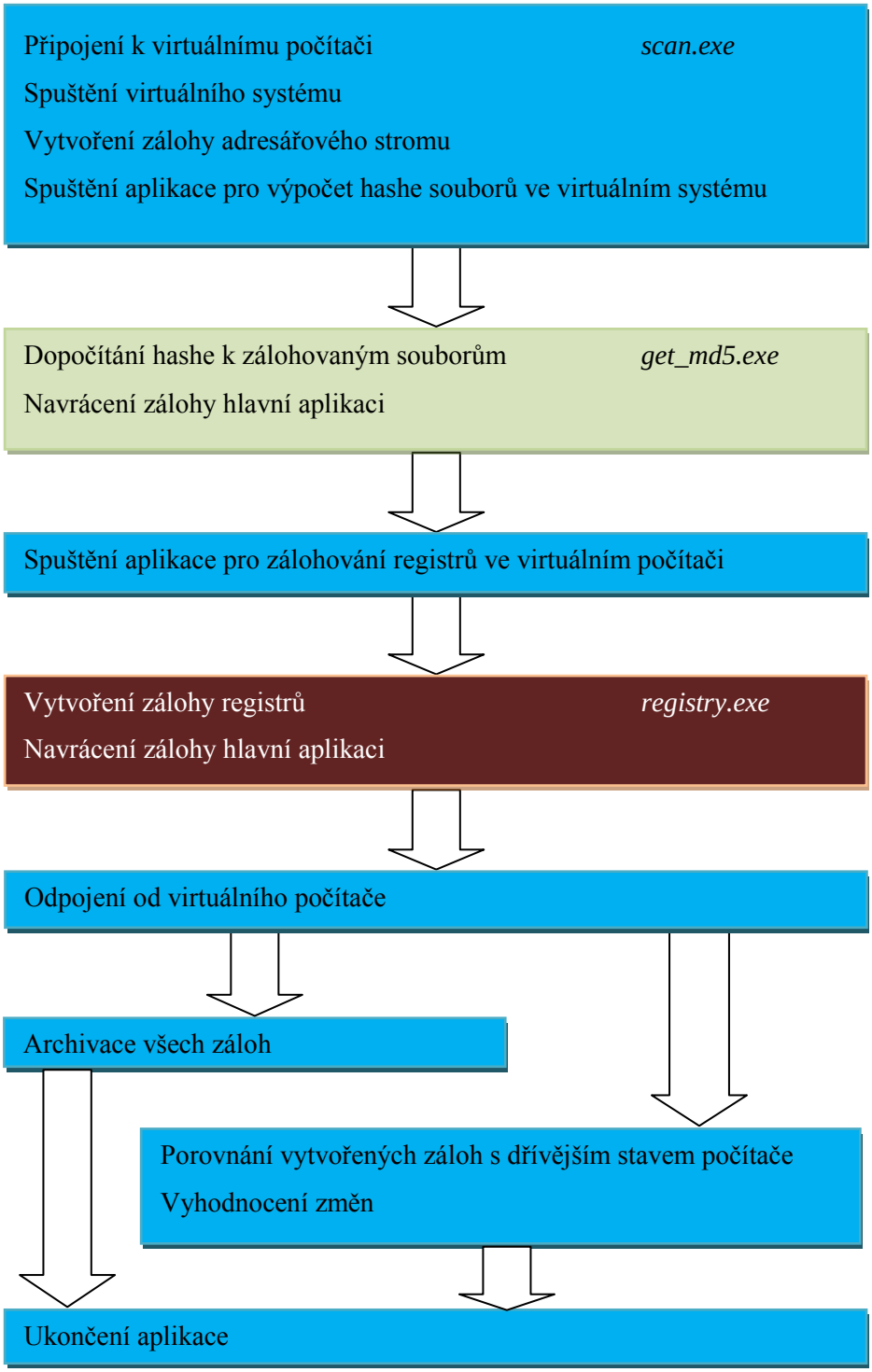
- jméno souboru
- cesta k souboru
- informace o souboru (soubor, adresář nebo symlink)
- kontrolní součet souboru

Záloha registrů bude obsahovat informace:

- cesta ke klíči i s názvem klíče

- jméno hodnoty
- typ hodnoty
- hodnota převedena na řetězec

Na obrázku 3.1 je znázorněn chod celého systému, kdy scan.exe dvakrát předává řízení aplikacím get_md5.exe a registry.exe a čeká na jejich dokončení, než pokračuje ve svém běhu. Jednotlivé barvy určují, o jakou aplikaci se jedná.



Obrázek 3.1 - Chod systému

3.1 Diagram tříd

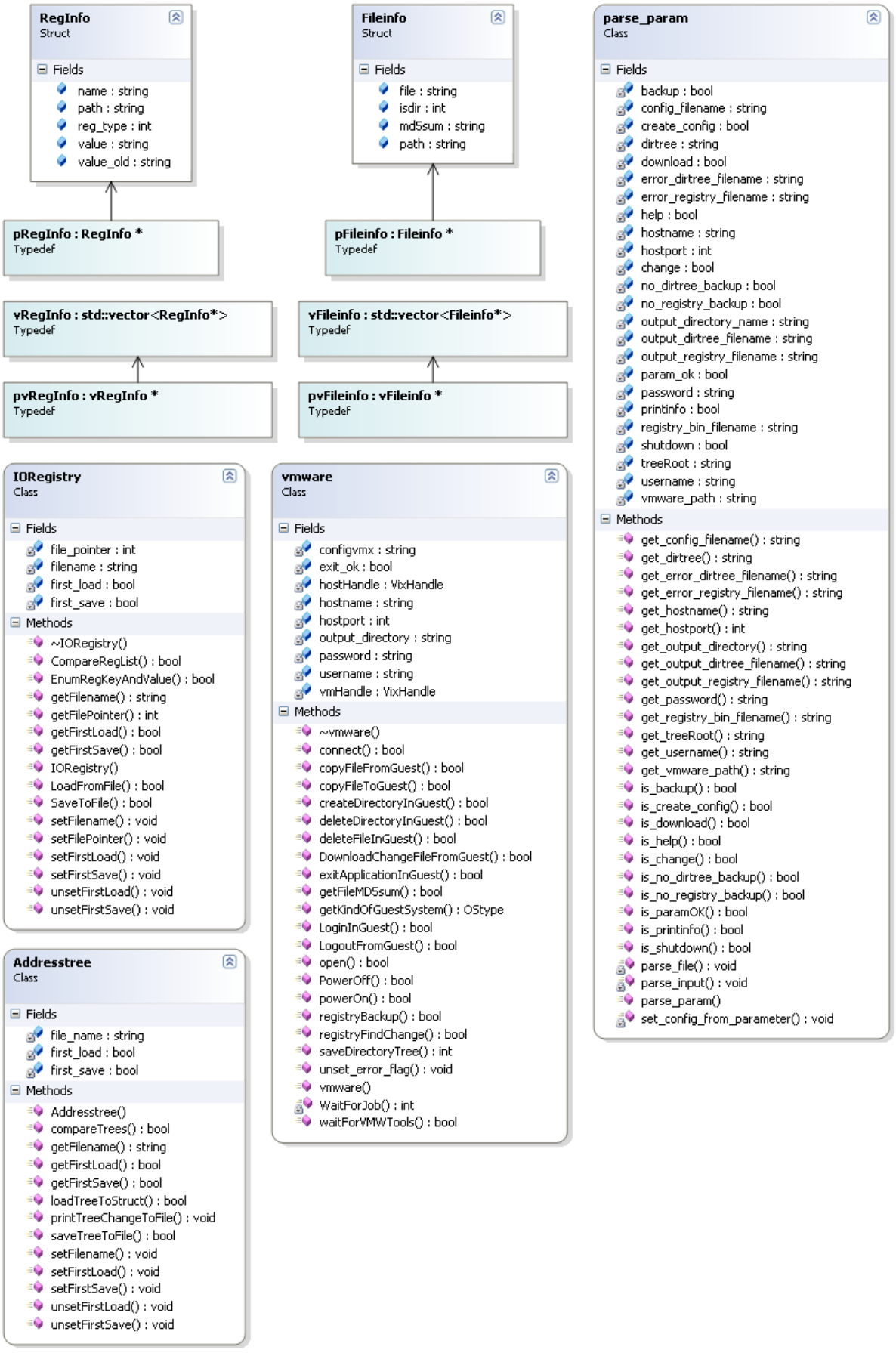
Celý navrhovaný systém obsahuje několik tříd, které spojuje instance třídy vmware. Ta používá ostatní metody k práci s vektory struktur a vstup-výstupním operacím s binárními soubory. Třída vmware zajišťuje spojení s virtuálním počítačem. Důležitými metodami jsou saveDirectoryTree, getFileMD5sum a registryBackup, vytvářející zálohu systému.

Třída parse_param ve svém konstruktoru načte hodnoty z konfiguračního souboru a parametry zadané při spuštění aplikace scan.exe. Všechny hodnoty, které může uživatel nějak ovlivnit, jsou uložené v této třídě a při běhu programu se pomocí jejich metod předávají ostatním funkcím.

Třída Addrestree slouží pro načítání a ukládání vektoru s informacemi o adresářovém stromu do souboru a k porovnání dvou takových vektorů. Do souboru je možné zapisovat vícekrát, kdy se při prvním zápisu soubor vytvoří a při dalších už se jen doplňují data na konec.

Metody třídy HashSum obsahují algoritmus pro výpočet kontrolního součtu souboru nebo řetězce. Pokud by bylo zapotřebí použít jiný algoritmus pro získání otisku, stačí obměnit metody této třídy.

Třída IORegistry načítá informace z registrů Windows do vektoru, porovnává dva vektory, načítá a ukládá jejich obsah z a do souboru. Třída dokáže obsluhovat jen jeden soubor se zálohou registrů. Vzhledem k použití proměnné file_pointer, která zaznamenává aktuální polohu v souboru, je možné vstupní a výstupní operace provádět v několika krocích.



Obrázek 3.2 - Diagram tříd

4 Implementace systému

Aplikace je implementovaná pomocí Microsoft Visual Studia 2005. Pro přístup k VMWare je využito VIX API. Tato knihovna vyžaduje pro svůj běh několik dll soborů a upřesňující nastavení Visual Studia. Celý systém je tvořen třemi aplikacemi, scan.exe, registry.exe a get_md5.exe. Z pohledu uživatele je důležitá aplikace scan.exe, která řídí celý systém pro vyhledávání změn v operačním systému, ostatní dvě se spouští jen pomocí hlavní aplikace. Pro kompilaci a správný chod aplikací je potřeba nastavit ve Visual Studiu kódování projektu na "Multi-Byte Character Set."

Na začátek exportovaných souborů je vložena hlavička, u zálohy adresářového stromu HEAD_OF_BINFILE nacházející se v addresstree.h a u zálohy registrů HEAD_OF_FILE v registry.h. Hlavička slouží pro rozpoznání typu binárního souboru

Během načítání parametrů nebo samotného běhu systému, může dojít k nepředvídatelným chybám, o kterých je uživatel informován na standardním chybovém výstupu. Příklady důležitých chybových hlášení je uveden v příloze A. Zdrojové kódy a ukázkový konfigurační soubor je uložen na přiloženém médiu.

Problém při implementaci vznikal při práci s řetězci. Knihovna VIX API pracuje s kódováním UTF-8, příkazový řádek systému Windows však s kódováním ANSI. Při převodu řetězců mezi jednotlivými funkcemi by mohlo docházet ke ztrátě nebo znehodnocení znaků. Proto celý systém ukládá řetězce do binárního souboru ve tvaru a s kódováním, v jakém ho získal od funkcí knihovny VIX API. Až na závěr při výpisu názvů na standardní výstup převede řetězce z UTF-8 na ANSI.

4.1 Aplikace registry.exe

Aplikace načítá klíče a hodnoty klíčů z registrů po jednotlivých větvích a ukládá je do binárního souboru. Název souboru je defaultní nebo určen parametrem při spuštění. Aplikaci je možné použít i bez hlavního programu, kdy se vytvoří záloha hodnot registrů do souboru. Pokud je k nějakému klíči odepřen přístup, vypíše jeho jméno na chybový výstup. Aby nebylo zatížení operační paměti aplikací příliš vysoké a zároveň se nemuselo stále zapisovat do souboru, ukládají se informace o registrech v dávkách určených proměnou MAX_REGISTRY_VECTOR_SIZE nacházející se v registry.h.

Popis chování programu:

1. Spuštění aplikace
2. Načtení klíčů a hodnot registrů do struktur
3. Export struktur do souboru
4. Ukončení aplikace

4.2 Aplikace get_md5.exe

Aplikace počítává k souborům kontrolní součty, založené na algoritmu MD5. Seznam souborů a adresářů musí být uložen v binárním souboru se zálohou adresářového stromu exportovaného metodou Addresstree::loadTreeToStruct(…). U adresářů se kontrolní součet nepočítá. Pokud by bylo zapotřebí používat jiný než MD5 hash, stačí upravit definici metody HashSum::getFileHash(…) uložené v HashSum.cpp

Popis chování programu:

1. Spuštění aplikace
2. Načtení struktur ze souboru
3. Výpočet kontrolního součtu k souborům
4. Export struktur do souboru
5. Ukončení aplikace

4.3 Aplikace scan.exe

Celý systém řídí aplikace scan.exe, která se připojí k VMWaru, spustí operační systém virtuálního počítače, přihlásí se do systému a načte adresářový strom do vektoru ukazatelů na strukturu Fileinfo. Po celkovém načtení vše exportuje do binárního souboru, zkopíruje program get_md5.exe a binární soubor se zálohou do virtuálního počítače, kde ho spustí. Výsledný binární soubor s dopočítanými otisky souborů si stáhne zpět a importuje do řídící aplikace. Pokud se má vytvářet záloha, uloží vše potřebné do určených souborů. Pokud se mají zjišťovat změny, importované struktury s informacemi o souborech se porovnají s binárním souborem, obsahujícím zálohu z minulosti. Nově přidané a změněné soubory se stáhnou z virtuálního počítače do hostitelského. Aplikace se odhlásí, odpojí od virtuálního počítače a ukončí se.

Pro vytvoření zálohy registrů aplikace scan.exe zkopíruje do virtuálního systému program registry.exe, spustí jej a výsledný binární soubor stáhne do hostitelského systému. Při zjišťování změn se stáhnutý soubor porovná s jinou zálohou.

Soubory potřebné ke kompilaci scan.exe

- vix.h
- vix.lib
- vm_basic_types.h

Soubory potřebné ke spuštění scan.exe

- iconv.dll
- libxml2.dll
- vix.dll
- vmcryptolib.dll

- zlib1.dll

Popis chování programu:

1. Spuštění aplikace
2. Načtení a vyhodnocení parametrů a konfiguračního souboru

   Pokud je nějaký parametr zadán špatně, aplikace upozorní uživatele hláškou na chybovém výstupu a ukončí se
3. Vytvoření adresáře pro výstupy

   Ověří se, zda adresář určený pro výstupy existuje, pokud ne vytvoří se
4. Připojení k VMWare

   Aplikace vytvoří handle, odkazující na aplikaci VMWare

   Pokud se připojení nezdaří, pokusí se o něj znovu a až po druhém neúspěchu se aplikace ukončí
5. Připojení k image virtuálního počítače

   Aplikace vytvoří handle, odkazující na image s virtuálním počítačem

   Pokud image neexistuje nebo je cesta k němu špatně zadaná, aplikace na to upozorní a ukončí se. Když při připojování vznikne chyba, pokusí se aplikace připojit ještě jednou a po druhém neúspěchu se ukončí.
6. Spuštění operačního systému ve virtuálním počítači
7. Čekání na odezvu VMWare Tools

   Ve chvíli, kdy je operační systém virtuálního počítače načten, dá znamení aplikaci
8. Přihlášení do operačního systému virtuálního počítače

   Úspěšnost se kontroluje funkcí, která nedokáže bez přihlášení pracovat
9. Načtení adresářového stromu do struktur
10. Export struktur do dočasného souboru

    Aplikace uloží informace o souborech a adresářích, informace neobsahují kontrolní součty souborů
11. Zkopírování programu get_md5.exe do VMWare a jeho spuštění
12. Čekání na ukončení programu get_md5.exe ve VMWare

    Pokud čas pro běh aplikace get_md5.exe přesáhne zadanou hranici, aplikace ho sama ukončí
13. Zkopírování souboru se zálohou stromu z VMWare do hostitelského počítače
14. Smazání vytvořených souborů ve virtuálním počítači
15. Načtení struktur z dočasného souboru
16. Další postup je znázorněn na obrázku Obrázek 4.1

    Backup - byla zvolena záloha aktuálního stavu systému

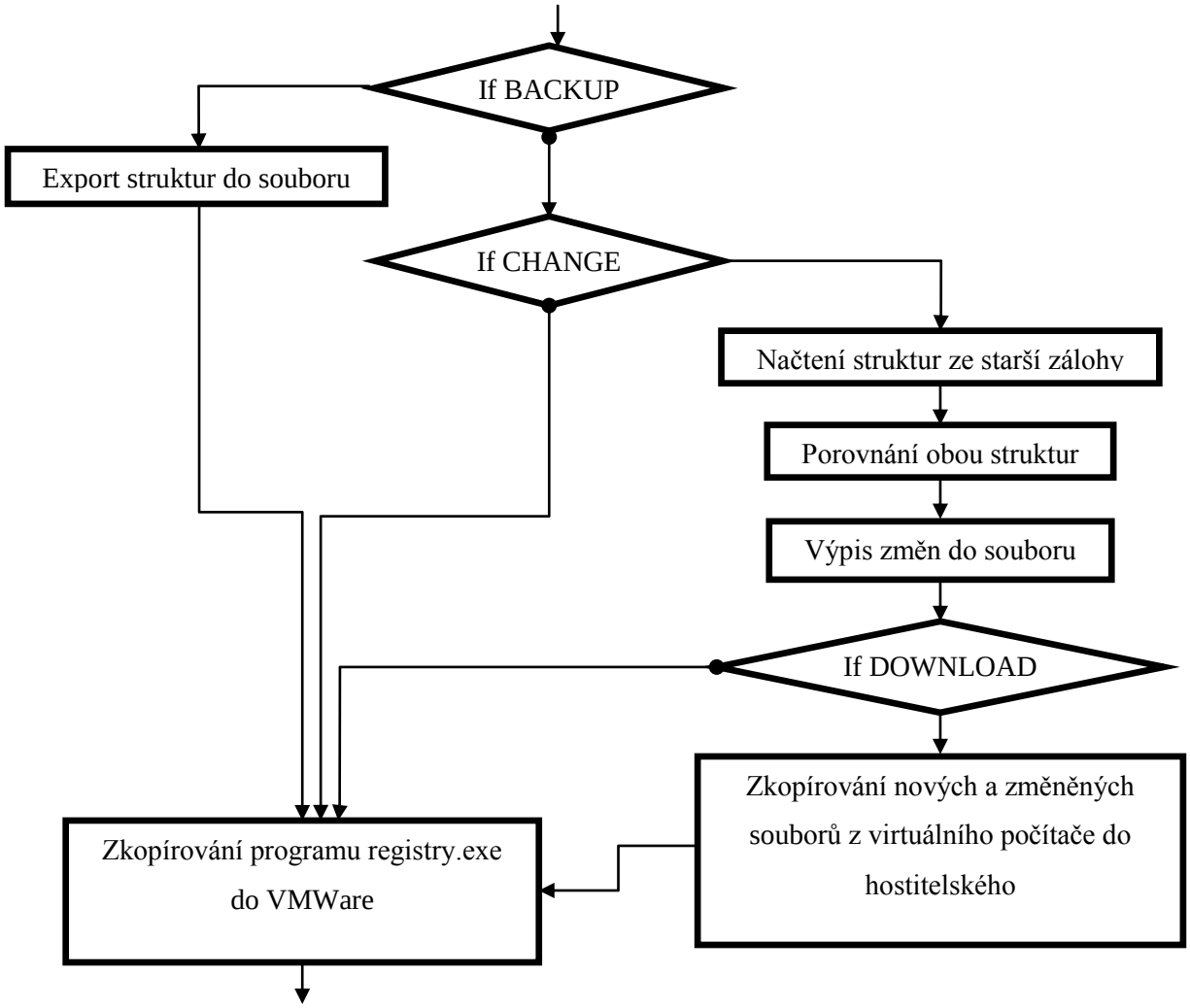
    Change - bylo zvoleno porovnávání změn
17. Čekání na ukončení programu registry.exe ve VMWare

Pokud čas pro běh aplikace registry.exe přesáhne zadanou hranici, aplikace ho sama ukončí

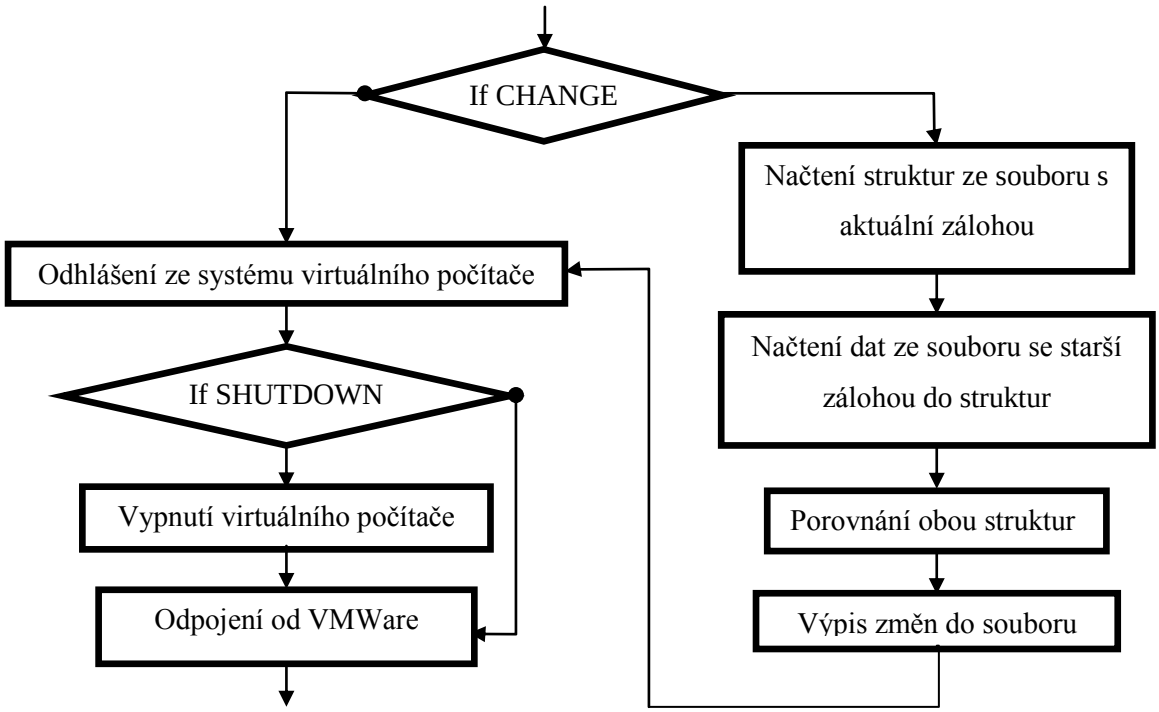
18. Zkopírování souboru se zálohou registrů do hostitelského počítače
19. Smazání vytvořených souborů ve virtuálním počítači
20. Další postup je znázorněn na obrázku Obrázek 4.2

    Change - bylo zvoleno porovnávání změn

    Shutdown - požadavek na vypnutí virtuálního počítače
21. Ukončení aplikace scan.exe



Obrázek 4.1- Chod aplikace scan.exe část 1



Obrázek 4.2 - Chod aplikace scan.exe část 2

4.4 Popis souborů se zdrojovými kódy

Soubory vmware_control.h, vmware_control.cpp

Obsahují deklaraci a definici třídy vmware. Metody této třídy ovládají a obsluhují virtuální operační systém. Většina funkcí VIX API je zapouzdřena v metodách této třídy. V hlavičkovém souboru jsou definované potřebné konstanty pro připojení k vmware a běh aplikace. Většina nastavení lze změnit pomocí konfiguračního souboru nebo parametrů při spuštění.

Konstanty v vmware_control.h

Konstanty, které lze měnit pomocí konfiguračního souboru:

TREEROOT ... cesta k adresáři, ze kterého se bude vytvářet adresářový strom

VMWARE_PATH ... cesta k image virtuálního systému, k *.wmx souboru

DIRTREE ... jméno binárního souboru se zálohou adresářového stromu

REGISTRY_SAVEFILE ... jméno binárního souboru se zálohou registrů

OUTPUT_DIRTREE_FILENAME ... jméno souboru, kam se uloží výpis změn, při porovnávání záloh adresářového stromu

OUTPUT_REGISTRY_FILENAME ... jméno souboru, kam se uloží výpis změn, při porovnávání záloh registrů

OUTPUT_DIRECTORY ... jméno adresáře, do kterého budou uloženy všechny výstupy

REGISTRY_ERROR FILENAME ... jméno souboru, kam se uloží výpis chyb, vzniklých při vytváření zálohy registrů

FILE_DIRTREE_ERROR FILENAME ... jméno souboru, kam se uloží výpis chyb, vzniklých při vytváření zálohy adresářového stromu

Konstanty neměnné pomocí konfiguračního souboru:

OUTPUT_SUBDIRECTORY_NEW_FILES ... jméno adresáře, kam se po porovnání aktuálního adresářového stromu a zálohy stáhnou nově vytvořené soubory z virtuálního systému

OUTPUT_SUBDIRECTORY_UPDATE_FILES ... jméno adresáře, kam se po porovnání aktuálního adresářového stromu a zálohy stáhnou změněné soubory z virtuálního systému

ERR_TIMEOUT ... (sekundy) defaultní čas, po který čeká aplikace na odpověď vmware funkce

DIR_LIST_TIMEOUT ... (sekundy) čas, po který čeká aplikace na odpověď vmware funkce pro získání obsahu adresáře

GET_HASH_TIMEOUT ... (sekundy) maximální čas, pro běh aplikace, která dopočítává hashe k souborům

REGISTER_TIMEOUT ... (sekundy) maximální čas, pro běh aplikace, která vytváří zálohu registrů

VMWTOOLS_TIMEOUT ... (sekundy) maximální čas, po který aplikace čeká na spuštění virtuálního operačního systému

CONNECT_TIMEOUT ... (sekundy) maximální čas, po který aplikace čeká na připojení se k vmware a k image s virtuálním systémem

DIRTREE_TMP ... jméno dočasného souboru pro zálohu adresářového stromu, po skončení aplikace je soubor smazán

MD5APPLICATION ... jméno aplikace, která dopočítává hash souborů (get_md5.exe)

MD5BAT ... jméno dočasného dávkového souboru, který spustí aplikaci ve virtuálním systému

REGISTRY_TMP ... jméno dočasného souboru pro zálohu registrů, po skončení aplikace je soubor smazán

REGISTRY_APPLICATION ... jméno aplikace, která vytvoří zálohu registrů (registry.exe)

REGISTRY_BAT ... jméno dočasného dávkového souboru, který spustí aplikaci ve virtuálním systému

PATH_IN_GUEST_XP ... cesta ve virtuálním systému Windows XP, kde aplikace vytvoří adresář, do kterého nakopíruje potřebně soubory pro svůj běh, adresář musí mít práva pro řízení pomocí VIX API

PATH_IN_GUEST_VISTA ... cesta ve virtuálním systému Windows Vista, kde aplikace vytvoří adresář, do kterého nakopíruje potřebně soubory pro svůj běh, adresář musí mít práva pro řízení pomocí VIX API

TMP_DIRECTORY_NAME ... jméno dočasného adresáře, který aplikace vytvoří ve virtuálním systému, pokud adresář v systému existuje, aplikace zkouší k názvu přidávat znak "0", dokud nenalezne jedinečný název adresáře

Soubory addresstree.h, addresstree.cpp

Obsahují definici a deklaraci třídy Addresstree. Třída pracuje s vektorem ukazatelů na strukturu Fileinfo, který zaznamenává informace o souborech a adresářích. Metody třídy umožňují import a export záznamů z a do binárního souboru, porovnávání dvou vektorů se záznamy a tisk reportu o změnách, vzniklých při porovnávání. Pokud by bylo zapotřebí zvolit jinou výstupní formu, než jaká je implementovaná, např. do XML, lze toho docílit obměnou metody Addresstree::printTreeChangeToFile(...).

Konstanty v addresstree.h

HEAD_OF_BINFILE ... hlavička, přidávaná na začátek binárního souboru

Soubory parse_param.h, parse_param.cpp

Obsahují třídu parse_param, která v konstruktoru načte a rozpozná nastavení z konfiguračního souboru a parametrů při spuštění. Jednotlivé metody pak předávají získané údaje. Hodnoty načtené z parametrů při spuštění mají vyšší prioritu než hodnoty z konfiguračního souboru.

Soubory registry.h, registry.cpp

Obsahují třídu IORegistry, která pracuje s vektorem ukazatelů na strukturu RegInfo. Do struktury jsou uloženy informace o hodnotách registrů. Metody této třídy importují a exportují záznamy z a do binárního souboru, vytváří a porovnávají zálohy registrů. Tisk reportu o změnách v registrech do souboru nebo na standardní výstup zabezpečuje funkce printRegistryChangeToFile(...). Pokud by bylo potřeba jiného výstupu, např. do XML, stačí přepsat tuto funkci.

Konstanty v registry.h

Program zle parametrizovat při překladu nastavením následujících hodnot. Délka řetězců je nastavená podle informací na stránkách Microsoftu na maximální možnou délku.

MAX_KEY_LENGTH ... maximální délka názvu klíče v registrech

MAX_VALUE_NAME ... maximální délka názvu hodnoty klíče v registrech

FLAG_IS_KEY ... pokud se ukládá klíč, ne hodnota, je použita tato značka místo označení typu hodnoty registru

CYCLE_NUMBER ... maximální počet položek načtených do vektoru ze souboru při porovnávání záloh registrů, pokud je hodnota vysoká, je vyšší zatížení paměti RAM

MAX_REGISTRY_VECTOR_SIZE ... maximální počet položek ve vektoru při vytváření zálohy registrů, pokud je počet položek ve vektoru vyšší, než tato hodnota, uloží se do souboru

HEAD_OF_FILE ... kontrolní řetězec, přidávaný na začátek binárního souboru

Soubor read_reg.cpp

Základní soubor k aplikaci registry.exe. V souboru je definován počet a názvy větví registrů. Pokud by se neměla dělat záloha všech klíčů, je nutné pozměnit konstanty v tomto souboru. Názvy větví registrů jsou uloženy ve statickém poli hkey_array[].

Konstanty v read_reg.cpp

Parametry programu, které lze změnit při překladu. V zásadě slouží pro přidání nebo odebrání větví registru.

HKEY_HIVES_NUMBER ... počet větví registrů

ROOT_SUBKEY ... cesta ke klíči, ze kterého se bude dělat záloha (prázdný řetězec = záloha celé větve)

REGISTRY_SAVE_FILENAME ... defaultní jméno výstupního binárního souboru se zálohou

Soubor get_md5.cpp

Základní soubor k aplikaci get_md5.exe.

Soubor main.cpp

Základní soubor k aplikaci scan.exe. Soubor obsahuje i funkce pro výpis nápovědy a pro vytvoření defaultního konfiguračního souboru.

4.5 Ovládání aplikace

Celý systém se spouští příkazem scan.exe a odpovídajícími parametry. Při použití více parametrů se nesmí vyskytovat možnosti "-h -backup -change -create_config_file" zároveň. Spuštění s parametry -backup a -change zároveň není možné. Pokud je použit konfigurační soubor a zároveň jsou některá nastavení provedena pomocí parametrů, tak hodnoty parametrů mají vyšší prioritu než hodnoty v konfiguračním souboru. Použití parametrů -no_registry nebo -no_directory zamezí vytváření dané zálohy. Uživatel spouští a nastavuje pouze aplikaci scan.exe, ostatní dvě si řídí systém sám. V tabulce 4.1 je zobrazeno několik ukázek spuštění systému.

Příkaz	Činnost
scan.exe -h	výpis nápovědy
scan.exe --help	
scan.exe -create_config_file	vytvoření konfiguračního souboru
scan.exe -config [FILE]	spuštění s konfiguračním souborem
scan.exe -config [FILE] [OPTION]...	spuštění s konfiguračním souborem a dalšími parametry
scan.exe -backup [OPTION]...	vytvoření zálohy
scan.exe -change [OPTION]...	zjištění změn
scan.exe -no_directory -no_registry -shutdown -config [FILE]	vypnutí virtuálního operačního systému

Tabulka 4.1- Příkazy ke spuštění programu

4.5.1 Parametry příkazové řádky

Systém je navrhnut tak, že se bude většinou spouštět za použití konfiguračního souboru, pokud by však bylo potřeba nějaké nastavení jednorázově upřesnit, stačí při spuštění použít parametr, který má vyšší prioritu než nastavení v konfiguračním souboru.

Parametr	Funkce
-backup	vytvoření zálohy adresářového stromu a registrů
-change	nalezení změn mezi aktuálním stavem virtuálního systému a zálohou
-h	vypíše nápovědu a ukončí aplikaci
--help	
-create_config_file	vytvoří ukázkový konfigurační soubor
-no_registry	nebude vytvářet zálohu registrů
-no_directory	nebude vytvářet zálohu adresářového stromu
-config [FILE]	načte hodnoty z konfiguračního souboru
-dirtree [FILE]	cesta k binárnímu souboru se zálohou adresářového stromu při backup, záloha je uložena do tohoto souboru při change, aktuální stav systému je porovnáván s touto zálohou
-registry_bin_filename [FILE]	cesta k binárnímu souboru se zálohou registrů při backup, záloha je uložena do tohoto souboru při change, aktuální stav systému je porovnáván s touto zálohou

Parametr	Funkce
-treeroot [DIRECTORY]	kořen adresářového stromu ve virtuálním systému, ze kterého se vytvoří záloha
-vmx_file [FILE]	absolutní cesta k *.wmx souboru
-hostname [HOSTNAME]	hostname pro připojení k vmware server
-hostport [PORT]	port pro připojení k vmware server
-username [USERNAME]	uživatelské jméno pro přihlášení do virtuálního operačního systému
-password [PASSWORD]	heslo pro přihlášení do virtuálního operačního systému, je uloženo v nešifrované podobě bez hesla není možné se do systému přihlásit a vytvořit zálohu
-output_directory [DIRECTORY]	cesta k adresáři, kam se uloží veškeré výstupy
-output_dirtree_file [FILE]	název souboru pro uložení výpisu změn, zjištěných při porovnávání adresářových stromů
-output_registry_file [FILE]	název souboru pro uložení výpisu změn, zjištěných při porovnávání registrů
-error_dirtree_file [FILE]	název souboru pro uložení výpisu chyb, vzniklých při výpočtu hashe souborů
-error_registry_file [FILE]	název souboru pro uložení výpisu chyb, vzniklých při vytváření zálohy registrů
-printinfo	tisk aktuálně prováděné práce na standardní výstup
-download	stažení nových a upravených souborů z virtuálního systému do adresářů v " output_directory "
-shutdown	vypne virtuální operační systém po provedení všech operací

Tabulka 4.2 - Parametry při spuštění aplikace scan.exe

4.6 Konfigurační soubor

Konfigurační soubor tvoří textový soubor. Platná hodnota, tedy hodnota, která bude načtena do aplikace, je vždy jen ta vyskytující se na řádku pod danou direktivou. Platná direktiva musí být na začátku řádku a psána velkým písmem. Pokud je řádek pod direktivou prázdný, je načtená hodnota rovna prázdnému řetězci. Všechny ostatní řádky jsou brány jako poznámky.

Ukázka z konfiguračního souboru:

```
VMX_FILE
D:\VMWare\Win XP.vmx
D:\VMWare\Windows Vista x64.vmx
```

Poznámky k ukázce:

VMX_FILE	direktiva značící, že na dalším řádku se bude vyskytovat absolutní cesta k *.wmx souboru
D:\VMWare\Win XP.vmx	absolutní cesta k *.wmx souboru, která bude aplikací načtena
D:\VMWare\Windows Vista x64.vmx	absolutní cesta k *.wmx souboru, která bude při načítání přeskočena

4.7 Výsledky testování

Testování implementované aplikace probíhalo na hostitelském systému Microsoft Windows XP SP3 32bit. K vytvoření a užívání virtuálního stroje byla použita aplikace VMWare Workstation 6.0.0. Ve virtuálním stroji byly nainstalovány operační systémy Microsoft Windows XP SP2 32bit a Microsoft Windows Vista SP1 64bit.

Během testování byly zjištěny možnosti vhodného rozšíření a ty, které by neohrozily již funkční aplikaci, implementovány. Pro běh aplikace není nutné, aby měl účet, pod kterým se přihlašuje do operačního systému virtuálního počítače, administrátorská práva.

Rychlost vytvoření zálohy je úměrná počtu adresářů a souborů. Pokud je v systému větší počet adresářů, čas potřebný k vytvoření zálohy je vyšší než při zálohování pouhých souborů. U Windows XP s cca 12000 soubory a složkami se čas pro vytvoření zálohy adresářového stromu a registrů pohybuje kolem 19 minut. U Windows Vista s cca 55000 soubory a složkami je potřebný čas 4,5 hodiny. Při zjišťování změn by měl být čas o pár minut vyšší, avšak dle provedených testů doba potřebná pro samotné porovnání je vzhledem k době zálohování zanedbatelná.

V tabulce 4.3 jsou k vidění doby běhu aplikace při různě obsáhlém adresářovém stromu. Jelikož je čas měřen v jednotkách minut, je přesnost jen orientační. To je například vidět při práci

s registry, kdy je k vytvoření zálohy registrů a porovnání aktuálního stavu se zálohou potřeba vyššího času, než pro samotné vytvoření zálohy.

Adres. strom	Registry	Záloha / Porovnání	Poznámky	Počet souborů	Čas (min)
C:\Program Files	Ne	Záloha		562	2
C:\Windows	Ne	Záloha		9451	7
C:\Windows	Ano	Záloha		9451	9
	Ano	Záloha	Bez zálohy adresářového stromu		3
C:\	Ano	Záloha		11172	19
C:\	Ano	Porovnání	Bez změny po vytvoření zálohy	11172	19
C:\	Ano	Porovnání	Nainstalován WINRAR	11228	20

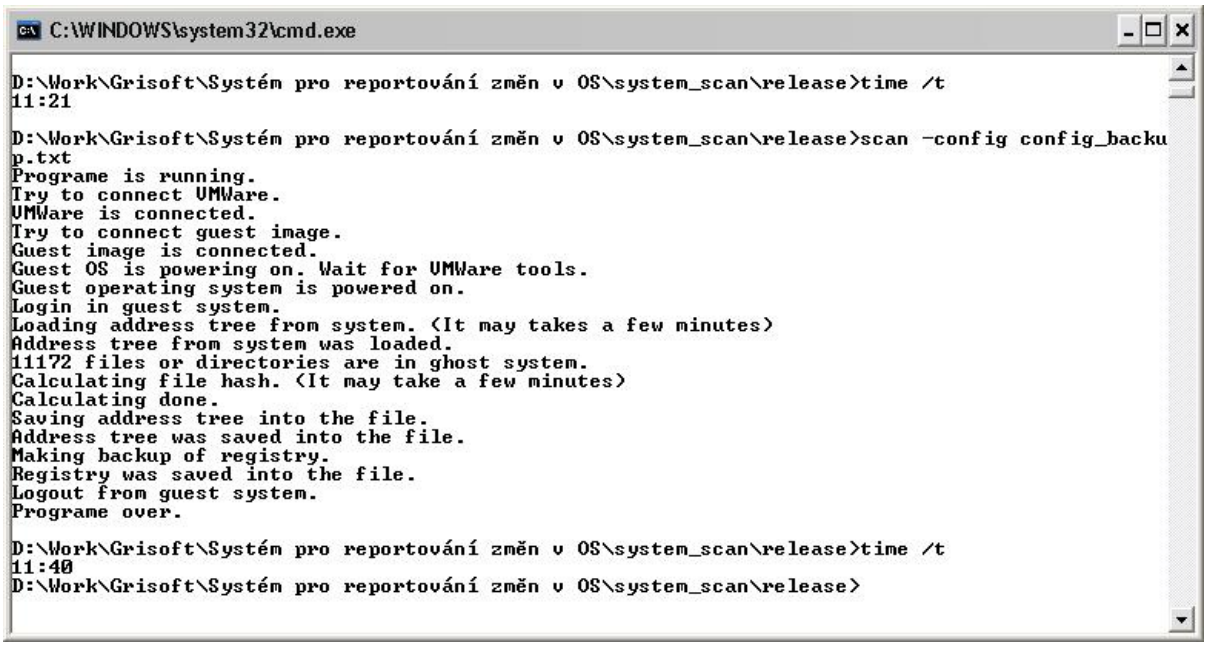
Tabulka 4.3 - Výsledky testů v systému Windows XP

V tabulce 4.4 je na první pohled vidět, že počty souborů ve Windows Vista jsou několikrát vyšší než ve Windows XP, čemuž odpovídá i vyšší čas při zálohování adresářového stromu. K záloze celého disku C: je však potřebný čas neúměrně vyšší, to je způsobené vyšším počtem adresářů, kdy aplikace musí každou složku otevřít a prozkoumat, zda je prázdná.

Adres. strom	Registry	Záloha / Porovnání	Poznámky	Počet souborů	Čas (min)
C:\Program Files	Ne	Záloha		1286	3
C:\Program Files	Ano	Záloha		1286	20
C:\Program Files	Ne	Porovnání	Bez změny po vytvoření zálohy	1286	4
	Ano	Záloha	Bez zálohy adresářového stromu		12
	Ano	Porovnání	Bez změny po vytvoření zálohy		12
C:\	Ne	Záloha		57089	259

Tabulka 4.4 - Výsledky testů v systému Windows Vista

Na obrázku 4.3 je znázorněn výpis informací, které uživatel vidí na standardním výstupu, pokud spustí aplikaci s parametrem -printinfo.



Obrázek 4.3 - Výpis informací při běhu aplikací

5 Závěr

Cílem bakalářské práce bylo nastudovat techniky sledování změn v operačním systému Windows po instalaci software, navrhnout a naprogramovat aplikaci pro odhalení těchto změn. Úvodem práce bylo nastudovat možnosti, jak pracovat s registry Windows a jakým způsobem z nich získávat data. Důležitou částí bylo také odhalení změn u souborů, kdy se pomocí algoritmu pro vytvoření kontrolního součtu porovnával aktuální stav souboru se starším. Další částí je popis a návrh implementace vytvořené aplikace. Na závěr práce jsou vysvětleny způsoby ovládání aplikace a sděleny poznatky získané v průběhu testování, které bylo provedeno jednak mnou samotným a také lidmi z firmy Grisoft, pro kterou byla aplikace navrhována.

Jelikož se některé soubory a hodnoty registrů mění v čase, optimální postup pro zjištění změn v systému je vytvoření zálohy, restartování virtuálního počítače, porovnání zálohy s aktuálním stavem systému a nainstalování software, u kterého chceme zjistit zásahy do operačního systému. Po těchto úvodních krocích provést nové zjištění změn, a to porovnat s předešlým. Změny způsobené samotnou prací systému se mohou vyloučit nebo brát jako změny, které nemusí být způsobené instalací daného softwaru.

Problém při implementaci vznikal při práci s řetězci. Knihovna VIX API pracuje s odlišným kódováním, než příkazový řádek operačního systému Windows. Aplikace proto ukládá vše v kódování knihovny a až při výpisu informací se řetězce převedou na kódování příkazového řádku.

Aplikace je navržená a odzkoušená na Windows XP a Vista, v budoucnosti by však mohl vznikat problém při použití s novějšími verzemi Windows, kdy by program neměl práva zapisovat do určených adresářů. Řešením by mělo být upravení zadaných cest nebo funkce pro rozeznávání operačních systému. Nesmíme opomenout dobu, po kterou aplikace zálohuje systém, která je při vyšším počtu adresářů vysoká. Možnou úsporou by bylo načítání adresářového stromu ve více vláknech, avšak na druhou stranu se zvýší zatížení pevného disku a vzniká problém, jak rozdělit adresářový strom na stejně velké části. V praxi bude aplikace zjišťovat změny u dané instalace jen jednou, proto není časová náročnost tak velký nedostatek.

Literatura

[1] Honeycutt Jerry, Registr Microsoft Windows, Vyd. 1., Brno: Computer Press, 2006. 546 stran. [z anglického originálu přeložila Veronika Matějů]

[2] Informace o registru systému Windows pro pokročilé uživatele, 12.března 2008, Revize: 12.1 URL: http://support.microsoft.com/kb/256986/cs

[3] Windows Registry HOWTO, 19. Březen 2001 URL: http://reboot.cz/howto/windows/windows-registry-howto/articles.html?id=125

[4] VMware, 18. 4. 2009, URL: http://cs.wikipedia.org/wiki/VMware

[5] VIX API Reference, URL: http://www.vmware.com/support/developer/vix-api/vix16_reference/

[6] Hashovací funkce, 10. 4. 2009, URL: http://cs.wikipedia.org/wiki/Hashovací_funkce

[7] Kment Vojtěch, Hašovací funkce: Jak se odolává hackerům, 3. 5. 2005, URL: http://www.lupa.cz/clanky/hasovaci-funkce-jak-se-odolava-hackerum/

[8] Zimola Ondřej, Kryptografické útoky na MD5, Brno: Vysoké učení technické, 2006. 45 stran

Seznam příloh

A. Chybová hlášení
B. CD/DVD obsahující:
    - Projekt aplikace Visual Studio 2005
    - Zdrojové kódy k projektu
    - Ukázkové konfigurační soubory
    - Ukázky výstupních souborů
    - Soubor s uloženou nápovědou
    - Dokumentaci k projektu ve formátu MS Word 2007, MS Word 2003 a pdf

Příloha A: Chybová hlášení

"Error: Some parameter is false."

- Špatně zadané nebo mezi sebou kolidující parametry
- Řešení: Zkontrolovat parametry a konfigurační soubor, jestli se nevyskytují např. možnosti BACKUP a CHANGE zároveň

"Error in memory allocation"

- Chyba při alokaci paměti

"Error in VixHost_Connect (timeout)"

- Připojení k VMWare se nezdařilo v časovém limitu
- Řešení: Spustit opětovně aplikaci. Chyba je způsobena funkcí VIX API VixHost_Connect()

"Error in VixHost_Connect"

- Připojení k VMWare se nezdařilo

"Error in VixVM_Open (timeout)"

- Otevření image virtuálního počítače se nezdařilo v časovém limitu
- Řešení: Spustit opětovně aplikaci. Chyba je způsobena funkcí VIX API VixVM_Open()

"Error in VixVM_Open: "

- Otevření image virtuálního počítače se nezdařilo
- Řešení: Pravděpodobně uvedený soubor s image neexistuje

"Error in VixVM_WaitForToolsInGuest"

- Spuštění operačního systému ve virtuálním počítači se nezdařilo v časovém limitu
- Řešení: Je vhodné operační systém spustit ještě před samotným spuštěním aplikace, nebo pokud se systém nenastartuje během vymezeného časového úseku, spustit aplikaci znova

"Error: Can't login to the guest system."

- Špatně zadané přihlašovací údaje (username a password)

"Error in VM_CopyFileFromHostToGuest (timeout)"

- Kopírování souboru z hostitelského do virtuálního počítače se nezdařilo v časovém limitu
- Řešení: Soubor je pravděpodobně příliš veliký

"Error in VM_CopyFileFromGuestToHost (timeout)"

- Kopírování souboru z virtuálního do hostitelského počítače se nezdařilo v časovém limitu
- Řešení: Soubor je pravděpodobně příliš veliký

"Error in VixVM_ListDirectoryInGuest: "

- Nelze načíst obsah adresáře ve virtuálním počítači

"Error: Can't copy necessary files to the guest system."

- Při kopírování MD5 aplikace a potřebných souborů do VMWare došlo k chybě

"Error in VM_RunProgramInGuest (timeout)"

- Program spuštěny ve virtuálním počítači nedokončil svoji práci v časovém limitu

"Error in program, witch is running in the guest system."

- Program spuštěný ve virtuálním počítači ukončil svoji práci s návratovým kódem jiným než 0

"Error: Can't make any directory."

- Aplikace nemůže vytvořit adresáře v hostitelském počítači

"Error: Can't open file for save registry information."

- Aplikace nemůže otevřít soubor pro výpis nalezených změn
- Změny budou vypsány na standardní výstup

"Error: File [FILENAME] is not valid with application."

- Hlavička binárního souboru se zálohou neodpovídá, soubor není vhodný k načtení aplikací