VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

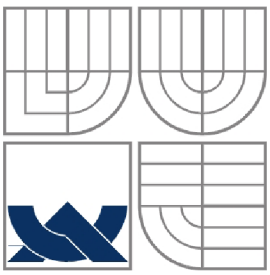
DIAGNOSTIKA A HW/SW AUDIT V POČÍTAČOVÉ SÍTI

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

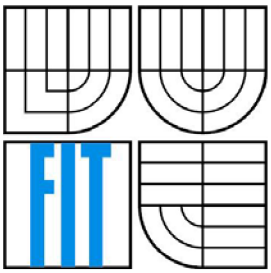
AUTOR PRÁCE
AUTHOR

BC. PETR HANUŠ

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DIAGNOSTIKA SW/HW AUDIT V POČÍTAČOVÉ SÍTI

DIAGNOSTIC AND HW/SW AUDIT IN COMPUTER NETWORK

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. PETR HANUŠ

VEDOUCÍ PRÁCE
SUPERVISOR

ING. PAVEL OČENÁŠEK

BRNO 2007

Zadání diplomové práce/5532/2006/xhanus02

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2006/2007

Zadání diplomové práce

Řešitel: **Hanuš Petr, Bc.**
Obor: Informační systémy
Téma: **Diagnostika a HW/SW audit v počítačové síti**
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se s požadavky konkrétní firmy na sledování a diagnostiku PC a serverů. Dále se seznamte s možnostmi tvorby HW/SW auditů na těchto serverech.
2. Nastudujte možnosti získávání specifických HW/SW parametrů počítače pro tvorbu diagnostiky a auditů.
3. Na základě výsledků analýzy proveďte návrh systému, který bude běžet na daném počítači a průběžně bude kontrolovat jeho vybrané HW/SW parametry a výsledky bude vhodným způsobem zasílat do centrální databáze. Zaměřte se na univerzálnost použití, v rámci vybraného operačního systému.
4. Navržený systém implementujte. Při implementaci se snažte používat bezpečný přístup, tak, aby získávaná/zasílaná data nebylo možné podvrhnout, případně zneužít.
5. Implementujte monitoring a statistiky nad centrální DB.
6. Funkčnost systému ověřte v praxi.
7. Diskutujte získané znalosti a možnosti dalšího rozšíření, zejména pro různé verze vybraného operačního systému.

Literatura:

- Dle doporučení vedoucího práce

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 - 3.

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

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% 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 paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Očenášek Pavel, Ing.**, UIFS FIT VUT
Datum zadání: 28. února 2006
Datum odevzdání: 22. května 2007

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

L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

Abstrakt

Tento projekt se zabývá návrhem a implementací systému na diagnostiku PC, tvorbu softwarových auditů, hardwarových auditů a informačního systému na ukládání a zpracování získávaných dat. Systém je navržen pro operační systém Microsoft Windows. Část systému, označena jako klient, je implementována v programovacím jazyce C++. K získávání diagnostických dat je použito rozhraní WMI. Informační systém je tvořen databázovým serverem MySQL a skriptovacím jazykem PHP.

Klíčová slova

Diagnostika PC, softwarový audit, hardwarový audit, WMI, MySQL, PHP.

Abstract

This project deals with the PC diagnostics, creation of SW and HW audits and development of information system for storage and analysis of data gained. The system is designed for Microsoft Windows operation system. Part of the system called Client is developed in C/C++ programming language. To gather diagnostic data, the WMI interface is used. The web part of information system is created using PHP scripting language and database MySQL server.

Keywords

Diagnostic PC, SW Audit, HW Audit, WMI, MySQL, PHP.

Citace

Hanuš Petr: Diagnostika SW/HW audit v počítačové síti, Brno, 2007, diplomová práce, FIT VUT v Brně.

Diagnostika a HW/SW audit v počítačové síti

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Pavla Očenáška.
Další informace mi poskytl pan Ing. Igor Šimkovský, ředitel společnosti The Best Network Solution.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Petr Hanuš
20.05.2007

Poděkování

Chtěl bych poděkovat panu Ing. Igoru Šimkovskému za pravidelné konzultace a praktické rady, které mi poskytoval. Dále panu Ing. Očenáškovi za jeho rady a vedení projektu.

Obsah

1 Úvod

Tato práce se zabývá návrhem a implementací systému na tvorbu HW/SW auditu a diagnostik. Tyto audity a diagnostiky jsou zasílány do centrálního serveru, kde jsou ukládány a zpracovány.

V kapitole nazvané *Požadavky firmy* shrnu všechny požadavky společnosti The Best Network Solution. V kapitole *Analýza požadavků* provedu analýzu požadavku a vyhodnotím možnost jejich splnitelnost. V kapitole nazvané *Teoretický rozbor* provedu teoretický rozbor možnosti získávání informací, plynoucích z požadavku firmy a rozbor použitých technologií v rámci celého systému.V kapitole nazvané *Architektura systému* popíši základní architekturu celého systému a základní principy interakce jednotlivých komponent celého systému. Dále zde popíši jednotlivé komponenty celého systému a princip komunikace mezi klientem a serverem. V kapitole nazvané *Klientská část systému* popíši detalně princip a implementaci klientské části systému. V kapitole nazvané *Serverová část systému* popíši detalně princip a implementaci serverové části systému. V poslední kapitole, která se nazává *Závěr,* shrnu celý projekt z hlediska dosažených výsledků a možnosti pokračování projektu.

2 Požadavky firmy

Požadavky firmy na systém vycházejí z několikaleté zkušenosti společnosti The Best Network Solution v oblasti správy počítačových sítí. Požadavky jsem projednával s ředitelem společnosti, panem Ing. Igorem Šimkovským. Požadavky na systém jsou kladeny tak, aby byl systém schopen zachytit nežádoucí stavy počítače a včas tyto nežádoucí stavy promítl do informačního systému, který bude k dispozici technikům společnosti The Best Network Solution.

K zachycení a předpovídání havarijních stavů počítače bude sloužit část systému, kterou budu nazývat Diagnostik. Tato komponenta systému bude v pravidelných časových intervalech sledovat počítač a kontrolovat jeho základní parametry, které jsou dány požadavky firmy.

Další důležitou součástí systému bude program provádějící softwarový audit. Tuto součást systému budu nazývat SW auditor. Hlavní funkcí SW auditoru bude zjišťování nainstalovaných programů. SW auditor pomůže společnosti zjistit nainstalovaný nežádoucí, případně nelegální software v počítači. Pokud takový stav nastane, společnost The Best Network Solution bude informovat společnost o nainstalovaném nežádoucím, případně nelegálním programu v počítači. Společnost BNS se tak zbaví odpovědnosti za nainstalovaný nelegální software v počítači.

Podobnou funkci jako má SW auditor, bude mít i HW auditor. HW auditor bude zjišťovat hardwarovou konfiguraci počítače. Pokud dojde ke změně v konfiguraci počítače, bude tato změna automaticky promítnuta do informačního systému. Tato část systému slouží k odhalení záměny hardwarové komponenty v počítači. Zmíněné komponenty patří do klientské části celého systému.

Další komponenta je z hlediska architektury celého systému server. Na tento server jsou posílány informace z klientů. Tuto komponentu bude tvořit informační systém zpracovávající získané hodnoty a aplikace, který bude zajišťovat importování informací do databázového systému.

2.1 Obecné požadavky na systém

Základní požadavek je kladen na operační systém, na kterém bude systém fungovat. Vzhledem k tomu, že se společnost specializuje na platformu Windows od společnosti Microsoft, tak systém musí fungovat na těchto operačních systémech:

- Windows 2000
- Windows XP
- Windows server 2003

2.2 Požadavky na diagnostiku

Jeden z nejčastějších problémů, se kterými se společnost setkává, je ztráta dat vlivem poškození pevných disků. Tento problém je velice závažný, protože společnost zodpovídá za bezproblémový chod serveru a počítačů. Časové limity pro znovuobnovení systému po havárii jsou velice přísné. Obnovení serverů je časově, a tudíž i finančně nákladné. Na serverech jsou zpravidla závislé celé pobočky firem, chod emailové komunikace, ukládání dat na sdílených discích a podobně.

Hlavním úkolem komponenty diagnostika je sledování funkčnosti pevných disků. Disky jsou na serverech zpravidla spojeny do diskových polí. Tyto pole sice umožňují rekonstrukci dat, ale nejednou se společnosti stalo, že pole nebylo možné obnovit.

Další požadavek je kladen na analyzování zaplnění pevných disků. Ne zřídka se stává, že je pevný disk počítače zaplněn a jeho výkon je uměle degradován.

Dalším požadavkem na diagnostiku je monitorování zatížení procesoru a zaplnění operačních pamětí. Tento požadavek je vytvořen, aby monitoroval případné nadměrné zatížení počítače a apeloval na rozšíření hardwarové konfigurace. Společnost tak může zákazníkovy ukázat, že je práce

na počítači pomalá vlivem nedostatečné hardwarové konfigurace a prezentovat tak nutnost zakoupení nového hardware.

2.3 Požadavky na SW audit

Hlavní požadavek na SW audit je hlídání nainstalovaného softwaru v počítači. SW audit by měl zjistit jména nainstalovaných programů, včetně jejich verzí. U operačních systémů Windows zjišťuje jejich název a licenční číslo.

2.4 Požadavky na HW audit

Hlavní požadavek na HW audit je získání informací o hardwarové konfiguraci počítače. Společnost tak zjistí veškerou změnu hardwaru v počítači a může informovat firmy o neoprávněných záměnách hardwaru.

HW auditor by měl sledovat tento hardware:

- Procesory
- Paměťové moduly
- Pevné disky
- CD/DVD mechaniky
- Grafické karty
- Síťové karty
- Zvukové karty
- Tiskárny

2.5 Požadavky na informační systém

Informační systém by měl být tvořen databázovým systémem MySQL. Dynamické stánky by měly být naprogramovány ve skriptovacím jazyce PHP. Stránky budou provozovány na webovém serveru IIS od společnosti Microsoft. IS by měl umožňovat automatické importování zasílaných dat.

2.6 Požadavky na komunikaci mezi klientem a serverem

Komunikace mezi klientem a serverem by měla být zajištěna pomocí protokolu FTP. Zasílaný soubor by neměl být posílán v otevřené podobě, aby nemohlo docházet k jeho podvržení. Klient by měl být navrhnut tak, aby byl schopen posílat data v sitích, kde je pevné připojení k Internetu.

2.7 Shrnutí požadavků na systém

Obecné požadavky

- Operační systém Windows 2000, Windows XP, Windows server 2003

Požadavky na diagnostiku (sledování parametrů počítače)

- Sledování funkčnosti pevných disků
- Sledování zaplnění pevných disků
- Sledování vytížení procesoru
- Sledování vytížení operačních pamětí

Požadavky na SW audit (sledování nainstalovaného SW a operačního systému)

- Název programu a jejich verze.
- Název operačního systému a jeho licenční číslo.

Požadavky na HW auditor (sledovaní nainstalovaného HW)

- Procesory
- Paměťové moduly
- Pevné disky
- CD/DVD mechaniky
- Grafické karty
- Síťové karty
- Zvukové karty
- Tiskárny

Požadavky na informační systém (použité technologie a vlastnosti IS)

- Úložiště dat – MySQL
- Webové rozhraní v PHP
- Automatické importování auditů

Požadavky na komunikaci mezi klientem a serverem

- K zasílání bude sloužit datový soubor
- Datový soubor nebude v otevřené podobě

3 Analýza požadavků

Všechny požadavky na systém jsem konzultoval s panem Ing. Igorem Šimkovským. Při jejich analýze jsem se zaměřoval na jejich splnění z hlediska realizace. Požadavky jsem hodnotil jako splnitelné.

Sledování pevných disků

Co se týká splnitelnosti požadavků ohledně sledování pevných disků, tak mnou navržený systém bude využívat technologii označenou jako S.M.A.R.T. Tato technologie umožňuje monitorovat stav pevného disku. Omezení tohoto požadavku je dáno podporou této technologie v pevném disku. Jde o technologii, která je implementována v logice pevného disku. Každý pevný disk disponuje pouze podmnožinou S.M.A.R.T atributů. Mocnost této podmnožiny je dána výrobcem pevného disku.

Zjišťování informací o počítači

Ke zjišťování informací o hardwaru a softwaru bude program využívat rozhraní WMI, které poskytuje operační systém Microsoft Windows. I zde mohou nastat různá omezení. Daný hardware například neumí poskytnou dané informace operačnímu systému.

Počítačové komponenty jsou obecně složitá zařízení a není tedy lehké na nich provádět diagnostiku nějakým univerzálním způsobem. Zpravidla na to existují specializované programy dodávané výrobcem. Dalo by se napsat, že neexistují universální přístupy zjišťování informací. Já jsem se rozhodl používat rozhraní WMI. Učinil jsem tak pro jeho univerzálnost, která může být vykoupena ne-stoprocentní funkčností na některých počítačích.

Informační systém

Co se týká požadavků na implementaci informačního systému, tak jsem nenašel žádný aspekt, který by mohl způsobovat nějaké problémy.

Informační systém je postaven na PHP a MySQL. PHP i MySQL jsou plně funkční na webovém serveru společnosti The Best Network Solution.

Komunikace

Klient bude schopen komunikovat přes pevné připojení do Internetu. Zde se použijí knihovny aplikačního rozhraní operačního systému Windows pro zajištění komunikace do sítě Internet.

Všechny počítače, které patří do množiny potenciálně sledovaných PC, disponují pevným a rychlým připojením do sítě Internet.

Shrnutí

Sledování HW a SW je poměrně problematické a obtížné. Je tedy pravděpodobné, že se na některých počítačích vyskytnou nečekané problémy. Jako například, že se nepodaří komunikace s rozhraním WMI. Co se týče kompatibility programu k nižším verzím operačních systémů než je Windows 2000, tak by to neměl být problém. Rozhraní WMI lze dodatečně nainstalovat, pokud není součástí OS. Tento instalační balíček existuje pouze pro operační systém Windows 98 a Windows 95. Pro nižší verze operačních systémů instalační balíček neexistuje. Přikládám ho jako přílohu k projektu na CD. Je v adresáři pomocné nástroje/WmiCore. Přesný odkaz na internetové stránky je v literatuře [7].

4 Teoretický rozbor

V této kapitole uvedu technologie a principy, které jsou využívany v rámci tohoto projektu.

4.1 S.M.A.R.T

Technologie S.M.A.R.T umožňuje sledování vybraných parametrů disku. Tyto parametry jsou definovány výrobci pevných disků. Pomocí těchto parametrů lze odhadnout spolehlivost pevného disku. Pevný disk však tuto technologii musí podporovat. Dá se ale konstatovat, že současné pevné disky touto technologií zpravidla disponují.

4.1.1 Spolehlivost disku – S.M.A.R.T atributy

V této kapitole jsem čerpal ze stránek www.pctuning.cz, přesný odkaz na článek uvádím v seznamu literatury [1].

Prvním parametrem, který o spolehlivosti disku něco vypovídá, je MTBF (Mean Time Between Failures), tedy střední doba mezi chybami. Jedná se o teoretickou hodnotu, která nám říká, jaký čas obvykle uplyne mezi dvěma poruchami disku. U dnešních disků je běžná hodnota 1 000 000 hodin. Často se MTBF zaměňuje s MTTF (Mean Time To Failure), tedy se střední dobou do poruchy disku. Výše zmíněných 1 000 000 hodin odpovídá 114 rokům. Jistě všichni dobře víme, že není problém, aby se pevný disk porouchal po prvním měsíci provozu. Pokud se tak stane, teoreticky by podle výpočtu MTBF neměla nastat porucha dalších 114 let. Přesněji to vyjádří tento vzorec:

$$MTBF = E(t) = \int_0^{\infty} xf(x)dx$$

Vzorec: 4.1.1.1

Vzhledem k tomu, že pevný disk je celkem složité zařízení, ve kterém nesmí docházet k chybám, je v něm implementována technologie, která sama pevný disk hlídá, dokáže předpovědět poruchu a vrací systému informace o stavu pevného disku. Nazývá se S.M.A.R.T (Self Monitoring Analysis and Reporting Technology). Bohužel, S.M.A.R.T není prozatím součástí operačního systému, který by s přehledem mohl v některých situacích uživatele informovat o tom, že se blíží porucha pevného disku.

Existuje však značný počet programů, které dokáží využívat technologii S.M.A.R.T a alarmovat nestandardní stavy pevného disku. Tyto programy snižují tak významnou pravděpodobnost ztráty dat, že lze předpokládat jejich nemalé zpoplatnění.

4.1.2 Atributy technologie S.M.A.R.T

Výčet důležitých atributů, které S.M.A.R.T hlídá a poskytuje:

GMR Head Amplitude - Vzdálenost hlaviček od plotny disku

Pokud dojde k tomu, že se hlavička dotkne točící se záznamové vrstvy, dojde k trvalému poškození povrchu. Toto je zřejmě nejhorší závada pevného disku, která může nastat. S.M.A.R.T hlídá správnou výšku hlaviček od plotny.

Error Correction Code - Počet použití ECC

Při čtení z magnetických médií dochází k občasným chybám. Zkonstruovat velmi spolehlivý disk by sice bylo možné, ale takové zařízení by bylo dost pomalé. Obecně můžeme říci, že čím spolehlivější zařízení zkonstruujeme, tím výrazněji roste jeho cena. Proto se konstruují takové pevné disky, které produkují akceptovatelné množství opravitelných chyb. Díky ECC disk dokáže dopočítat chybějící informaci. Tato procedura však disk zpomaluje, tudíž se ECC používá v omezené míře. S.M.A.R.T kontroluje také počet těchto chyb. Pokud toto číslo rychle narůstá, životnost disku se blíží ke svému konci.

Spin Up Time - Čas potřebný k roztočení ploten

S.M.A.R.T kontroluje čas potřebný k roztočení ploten. Pokud se toto číslo změní, znamená to, že by mohl být motorek poškozený.

Temperature - Teplota disku

Vzhledem k tomu, že v pevném disku je mnoho pohyblivých částí, dochází k mechanickému zahřívání. Zjištění teploty je důležité kvůli teplotní kalibraci TCAL, ale také proto, že zvýšená teplota může znamenat problémy s motorkem nebo ložisky disku.

Raw Read Error Rate – Počet chyb při čtení

Některé disky sčítají neopravitelné chyby, jiné sčítají všechny chyby. Kumulovaná hodnota udává počet chyb za život disku

Start/Stop Count - Počet roztočení plotny

Důležitá je kumulovaná hexadecimální hodnota RAW.

Reallocated Sector Count - Počet přemapovaných sektorů

Pokud dochází k chybám stále ve stejném sektoru, označí S.M.A.R.T tento sektor jako vadný. Starší disky přestaly takové sektory prostě používat. Nové disky mají v zásobě tzv. zásobní sektory, kam data z vadného přesunou. Toto probíhá na úrovni logiky disku, operační systém o tomto nerozhoduje.

Power On Hours Count - Počet hodin provozu disku

Počet hodin, kdy byl disk v provozu.

Spin Retry Count - Počet opětovných pokusů o roztočení ploten

Čím méně, tím lépe. Pokud takové pokusy nebyly, je to vůbec nejlepší varianta.

Power Cycle Count - Počet vypnutí a zapnutí pevného disku

Pokud tento počet není stejný jako Start/Stop Count, znamená to, že se do tohoto nepočítají různé spánkové režimy jako např. STR.

Throughput Performance - Výkon v propustnosti

Pokud se tato hodnota rychle snižuje, je to špatné.

Seek Time Performance - Výkon disku při seekování

Pokud tato hodnota klesá, je pravděpodobné, že na mechanickém systému disku se objevila závada.

Recalibration Retries - Počet pokusů o rekalibraci

Vysoký počet těchto pokusů indikuje mechanickou závadu.

Soft Read Error Rate - Počet SW chyb při čtení

Jde o počet softwarových chyb při čtení. Nejedná se tedy o chybu mechanickou, resp. hardwarovou.

G-Sense Error Rate - Počet chyb, které vznikly v důsledku otřesů

Tuto hodnotu je dobré sledovat především u notebooků.

Power-Off Retract Count - Počet vypnutí pevného disku

Počet vypnutí pevného disku.

Current Pending Sector Count - Počet sektorů, které čekají na přemapování

Počet sektorů, které čekají na přemapování. Důležitá je kumulativní hodnota.

Uncorrectable Sector Count - Počet neopravitelných sektorů

Počet neopravitelných sektorů.

Ultra DMA CRC Error Count - Počet chyb v CRC součtu

Počet chyb v CRC součtu, které proběhly v režimu Ultra DMA, tedy při přímém přístupu do operační paměti.

Write Error Rate (Multi Zone Error Rate) – Součet chyb

Součet chyb vzniklých při zápisu na disk.

Disk Shift

Disk Shift je ochrana proti otřesům, která zajišťuje, aby se hlavička nedotkla točící se plotny. S.M.A.R.T archivuje počet zákroků této technologie.

Loaded Hours

Čas, po který hlavičky disku opravdu pracovaly.

Load/Unload Retry Count

Počet pokusů o nastavení hlaviček.

Load Fiction

Počet konfliktů hlaviček a plotny.

Torque Amplification Count

Počet pokusů o vyrovnání rychlosti ploten.

S.M.A.R.T funguje tedy tak, že monitoruje pevný disk a poskytuje informace o jeho stavu. Na softwaru je, aby hlídal tyto hodnoty a případně alarmoval.

4.2 WMI

Při studování problému týkajícího se zjišťování hlubších informací o hardwaru a softwaru jsem se zaměřil na nějaký obecný mechanizmus zjišťování informací. Je sice mnoho diagnostických programů, ale ty jsou zpravidla směrovány na konkrétní hardware od konkrétního výrobce. Například sledování otáček ventilátoru na procesoru. Není problém tuto informaci zjišťovat pomocí software dodávaného výrobcem, takovýto přístup a software pro tento projekt není perspektivní. Základní stavební kámen, na kterém bude postavena diagnostika počítače je rozhraní WMI.

Windows Management Instrumentation (WMI) je dostupná a rozšiřitelná správní infrastruktura, která je součástí Windows 2000 a vyšších verzí. Jako implementace Web-based Enterprise Managementu (WBEM) převzatá z Distributed Management Task Force (DMTF), obsahuje sadu objektů pro správu počítačových systémů, operačních systémů a aplikací na daném spravovaném systému. Pomocí sady objektových tříd převzatých z modelu CIM, mohou aplikace a skripty vidět a měnit nastavení a přebírat události z objektů. WMI například předkládá bohaté informace o operačním systému (kolik je spuštěných úloh, stav jednotlivých služeb, využití procesoru, atd.) a publikuje je do jediného výstupu, který je dostupný pomocí standardních skriptovacích jazyků [2].

Vlastnosti WMI:

- ***Jednotné skriptovací API***
  Všechny spravované objekty jsou definovány pod běžným objektovým rámcem založeným na objektovém modelu CIM. K přístupu k informacím z různých zdrojů (jako Win32 API, protokol událostí Windows NT, registr, ovladače zařízení, SNMP a samozřejmě Active Directory) mohou skripty využívat jediné API (WMI). Ve skriptovacích jazycích WMI můžete také psát skripty pro ASP a HTML stránky.

- ***Vzdálenou správu***
  Objekty spravované pomocí WMI jsou dle své definice lokálně i vzdáleně dostupné aplikacím a skriptům. Ke správě vzdálených objektů není zapotřebí žádná dodatečná práce.

- ***Navigaci***
  Aplikace a skripty mohou zkoumat informace o systému. Můžete zjišťovat vztah mezi příbuznými objekty a zkoumat, jak jedna spravovaná entita ovlivňuje jinou.

- ***Možnost dotazování***
  WMI spravuje data jako relační databázi, takže data můžete filtrovat pomocí SQL dotazů.

- ***Účinný sběr a publikace událostí***
  Pro prakticky každou změnu ve spravovaných objektech, dokonce i těch, které interně události nepodporují, můžete požadovat spuštění určité události.

Základní WMI třídy

Třída WMI:	*Popis:*
Win32 Classes	Hlavní třídy pro práci s operačním systémem Windows.
WMI Registry Classes	Třídy určené pro manipulaci s registry.
WMI System Classes	Třídy poskytující základní funkčnost WMI
IPMI Classes	Třídy, které dodávají data z IPMI (Intelligent Platform Management Interface) poskytovatele, když je přístupný vhodný BMC (Baseboard Management Controller) hardware.
Monitor Display Classes	Třídy poskytující informace o zobrazovací jednotce.
MSFT Classes	Třídy poskytující prostředky pro manipulaci s osobitými vlastnostmi operačního systému. Obsahují WMI Troubleshooting Classes.
CIM Classes	Třídy určené na vývoj vlastních WMI tříd.
Standard Consumer Classes	Sada WMI událostí určená na spouštění akcí s podmínkou přijatou libovolnou událostí.
MSMCA Classes	Třídy poskytující prostředky na manipulaci a zobrazování systémových událostí.
WMI C++ Classes	Kompletní přehled WMI C++ tříd.

Získávání informací z WMI

Nejprve musíme vybrat jmenný prostor, například CIM V2. Potom vybereme třídu, dle toho jaké informace chceme zjišťovat. Nad touto třídou provádíme SQL dotazy. Třída se tváří jako relační databáze. WMI poskytuje rozhraní pro programovací jazyk C++ a další programovací a skriptovací jazyky. V tomto projektu se přistupuje k WMI pomocí programovacího jazyka C++.

Při testování získávání informací je vhodné používat nějaký nástroj, pomocí kterého lze procházet jmenné prostory WMI a jejich třídy. Já jsem používal program Scriptomatic od společnosti Microsoft, který je poskytován zdarma. Tento program neumožňuje pouze procházení tříd, ale i generování skriptů do různých skriptovacích jazyků. Program Scriptomatic přikládám s projektem. Je umístěn na CD v adresáři pomocné nástroje/scriptomatic. V literatuře [3] uvádím přesný odkaz na internetové stránky.

Třídy, které jsou využívány v projektu

Zde uvádím, jak získat potřebné parametry z WMI. V prvním sloupci je parametr, který chci získat. V prostředním sloupci je jmenný prostor, ve kterém je třída s potřebnou informací. V posledním sloupci je WQL dotaz, pomocí kterého se potřebná informace získá. Podle dotazu jsem vidět ve které třídě je informace uložena.

Dotazy fungují obdobně jako například u databázového systému. Pokud WMI informaci nemá, nevrátí žádný řádek. U procesorů je například zase možné, že řádků vrátí více v závislosti na počtu procesorů v počítači.

V tabulce 4.2.1 předkládám dotazy, které jsou využívány v rámci projektu.

Třídy a WQL dotazy:

Parametr:	*Namespace:*	*Wmi dotaz – WQL:*
Jméno počítače	CIM V2	Select Name from Win32_ComputerSystem
Doména	CIM V2	Select Domain from Win32_ComputerSystem
Operační systém	CIM V2	Select Caption from Win32_ComputerSystem
Sériové číslo	CIM V2	Select SerialNumber from Win32_ComputerSystem
Název programu	CIM V2	Select Name from Win32_Product
Verze programu	CIM V2	Select Version from Win32_Product
Typ procesoru	CIM V2	Select Name from Win32_ComputerSystem
SN procesoru	CIM V2	Select ProcessorId from Win32_ComputerSystem
RAM paměť kapacita	CIM V2	Select Capacity from Win32_PhysicalMemory
RAM paměť SN	CIM V2	Select SerialNumber from Win32_PhysicalMemory
HDD model	CIM V2	Select Model from Win32_PhysicalMemory
HDD velikost	CIM V2	Select Size from Win32_PhysicalMemory
HDD SN	CIM V2	Select SeialNumber from Win32_PhysicalMedia
CD/DVD	CIM V2	Select Name from Win32_CDROMDrive
Grafická karta	CIM V2	Select Caption from Win32_DisplayConfiguration
Síťová karta	CIM V2	Select Description from Win32_NetworkAdapterConfiguration where IPEnabled = True
Zvuková karta	CIM V2	Select Name from Win32_SoundDevice
Tiskárna	CIM V2	Select Name from Win32_Printer where Network=false
S.M.A.R.T status	WMI	Select vendorSpecific from MSStorageDriver_FailurePredictData
Zatížení procesoru v procentech	WMI	Select percentage from ProcessorPerformance
Zatížení operační paměti	CIM V2	Select AvaliableMBytes from Win32_PerfRawData_PerfOS_Memory

Tabulka 4.2.1
WMI třídy a WQL dotazy

4.3 PHP

PHP je skriptovací programovací jazyk, určený především pro programování dynamických internetových stránek. Nejčastěji se začleňuje přímo do struktury jazyka HTML, což je velmi výhodné pro tvorbu webových aplikací [4].

PHP skripty jsou prováděny na straně serveru k uživateli je přenášen až výsledek jejich činnosti. Syntaxe jazyka kombinuje hned několik programovacích jazyků (Perl, C, Pascal a Java). PHP je nezávislý na platformě, skripty fungují bez úprav na mnoha různých operačních systémech. Obsahuje rozsáhlé knihovny funkcí pro zpracování textu, grafiky, práci se soubory, přístup k většině databázových serverů (mj. MySQL, ODBC, Oracle, PostgreSQL), podporu celé řady internetových protokolů (HTTP, SMTP, SNMP, FTP, IMAP, POP3, LDAP, …).

V systému je PHP použito pro generování dynamických webových stránek informačního systému.

4.4 MySQL

MySQL je multiplatformní databázový systém. Komunikace s ní probíhá, jak už název napovídá, pomocí jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními.

Pro svou snadnou implementovatelnost (lze jej instalovat na Linux, MS Windows, ale i další operační systémy), výkon a především díky tomu, že se jedná o volně šiřitelný software, má vysoký podíl na v současné době používaných databázích. Velmi oblíbená a často nasazovaná je kombinace MySQL, PHP a Apache jako základní software webového serveru. MySQL byla od počátku optimalizována především na rychlost, a to i za cenu některých zjednodušení: má jen jednoduché způsoby zálohování a až donedávna nepodporovala pohledy, triggery, a uložené procedury. MySQL je v projektu použito jako datové úložiště.

I když nové verze MySQL poskytují triggery a uložené procedury, tak v projektu nejsou využívány, protože na firemním serveru je nainstalovaný MySQL nižší verze, která tyto novinky nepodporuje.

4.5 SQL

SQL je standardizovaný dotazovací jazyk, používaný pro práci s daty v relačních databázích. SQL je zkratka anglických slov Structured Query Language (strukturovaný dotazovací jazyk) [5].

SQL příkazy se zpravidla dělí do čtyřech základních skupin:

Příkazy pro manipulaci s daty

- Select - vybírá data z databáze, umožňuje výběr podmnožiny a řazení dat
- Insert - vkládá do databáze nová data
- Update - mění data v databázi (editace)
- Delete - odstraňuje data (záznamy) z databáze

Příkazy pro definici dat

- Create - vytváření nových objektů
- After - změny existujících objektů
- Drop - odstraňování objektů

Příkazy pro řízení dat

- Grant - příkaz pro přidělení oprávnění uživateli k určitým objektům
- Revoke - příkaz pro odnětí práv uživateli
- Begin - zahájení transakce
- Commit - potvrzení transakce
- Rollback - zrušení transakce, návrat do původního stavu

Ostatní příkazy

- Do této skupiny patří příkazy pro správu databáze

Dotazovací jazyk SQL je používán v informačním systému pro dolování dat z databáze a k získávání informací z WMI tříd. U WMI se tento jazyk nazývá WQL, ale jeho syntaxe je ve většině případů shodná se SQL. Osobně jsem nenašel žádný případ, kde by se WQL rozcházelo se SQL.

4.6 HTTP a HTTPS

HTTP (Hyper Text Transfer Protocol) je Internetový protokol určený původně pro výměnu hypertextových dokumentů ve formátu HTML. Tento protokol je spolu s elektronickou poštou tím nejvíce používaným a zasloužil se o obrovský rozmach internetu v posledních letech. HTTP používá jako některé další aplikace tzv. jednotný lokátor prostředků (URL, Uniform Resource Locator), který specifikuje jednoznačné umístění nějakého zdroje v Internetu [6].

K protokolu HTTP existuje také jeho bezpečnější verze HTTPS, která umožňuje přenášená data šifrovat a tím chránit před odposlechem či jiným narušením. Protokol funguje způsobem dotaz-odpověď. Uživatel (pomocí programu, obvykle internetového prohlížeče) pošle serveru dotaz ve formě čistého textu obsahujícího označení požadovaného dokumentu, informace o schopnostech prohlížeče apod. Server odpoví pomocí několika řádků textu popisujících výsledek dotazu (zda se dokument podařilo najít, jakého typu dokument je atd.), za kterými následují data samotného požadovaného dokumentu.

HTTP a HTTPS protokol je využíván pro komunikaci mezi webovým prohlížečem a webovým rozhraním informačního systému.

4.7 FTP

FTP (File Transfer Protocol) je protokol aplikační vrstvy z rodiny TCP/IP. Je určen pro přenos souborů mezi počítači, na kterých mohou běžet velmi rozdílné operační systémy.

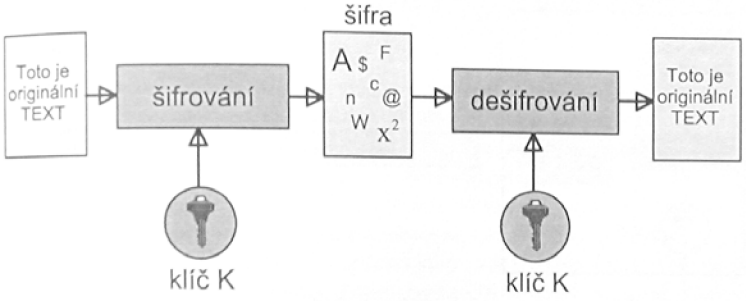
Tento protokol je využíván k přenosu dat mezi klientem (sledované PC) a serverem (informační systém).

4.8 Šifrování

Kryptografie neboli šifrování je nauka o metodách utajování smyslu zpráv převodem do podoby, která je čitelná jen se speciální znalostí. V projektu jsem použil symetrický šifrovací algoritmus Rijndael.

4.8.1 Symetrický šifrovací algoritmus

Symetrický algoritmus je nejpoužívanější typ šifrovacího algoritmu. Symetrický se nazývá proto, že se stejný šifrovací klíč používá k šifrování i dešifrování. Příjemce i odesilatel musí tedy mít naprosto stejný klíč. Na obrázku 4.8.1 je zobrazen základní princip symetrických šifrovacích algoritmů. Obrázek je z knihy Ochrana dat v informačních systémech [8].



Obrázek 4.8.1[7]
Blokové schéma symetrického šifrování a dešifrování

4.8.2 Šifrovací algoritmus Rijndael

Pro popis tohoto algoritmu jsem čerpal z časopisu Chip. Přesnou referenci na článek a jeho autora uvádím v seznamu literatury pod číslem [9].

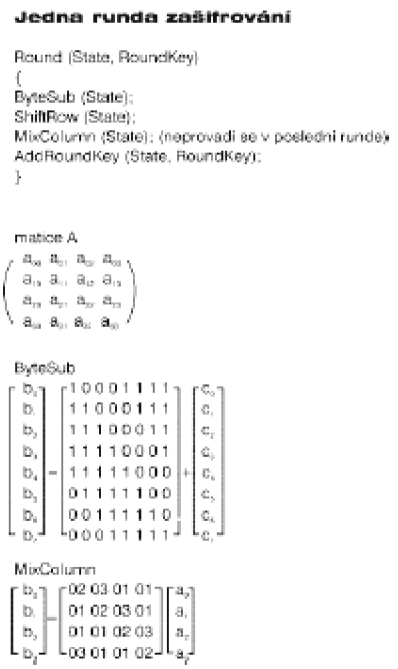
Rijndael je jedním z pěti kandidátů na Advanced Encryption Standard (AES). Ačkoliv šifra Rijndael podporuje i větší bloky, pro AES je délka vstupního a výstupního bloku definována jako 128 bitů. Délka klíče je volitelně 128, 192 a 256 bitů, což je 4, 6 nebo 8 32bitových slov.

Rijndael je velmi flexibilní, i když jeho popis uvedeme v bajtech, lze jej elegantně zapsat i v 32bitových slovech. Návrh je přímočarý a za základ jsou použity operace v různých algebraických strukturách. Pracuje se s prvky Galoisova tělesa GF (2^8) a s polynomy, jejichž koeficienty jsou prvky z GF (2^8). Příslušné operace s nimi lze provádět buď tabulkově nebo výpočtem přímo, což je v prvním případě výhodné pro implementaci softwarovou a v druhém případě pro hardwarovou. Bajtově orientovaný návrh také umožňuje optimalizovat programový kód pro různé mikroprocesory. Pro operace zašifrování a odšifrování sice není možné využít úplně totožný hardware (jako tomu bylo u šifry MARS), značnou část jeho prvků však použít lze.

Než přistoupíme k základním operacím, budou zde vysvětleny nejnutnější pojmy. Prvky v Galoisově tělese GF (2^8) mají 8bitů ($B_7,\ldots B_0$), nereprezentují však bajty, nýbrž polynomy ($b_7x^7+\ldots+b_1x^1+b_0$). Násobení těchto prvků je proto zavedeno nikoli jako násobení bajtů, ale jako násobení jim odpovídajících polynomů, a to modulo M (x)=$x^8+x^4+x^3+x^1+1$. Takže například ´57´ (v apostrofech píšeme běžné hexadecimální vyjádření bitů ($b_7,\ldots,b_0$) krát ´83´ je rovno ´C1´, neboť ($x^6 + x^4 + x^2 + x^1 + 1$) * ($x^7 + x^1 + 1$) = ($x^7 + x^6 + 1$) mod m(x).

Postup při šifrování

Rijndael pracuje v rundách. Jejich počet Nr = 10, 12 a 14 je určen podle toho, jak dlouhý je šifrovací klíč, a odpovídá hodnotám Nk = 4, 6 a 8. Pro delší klíč se tedy použije více rund. Před operací zašifrování (nebo v jejím průběhu, tzv. „on-the-fly“) se vypočítá 4 + Nr * 4 rundovních klíčů (32bitových slov). První čtyři se „naxorují“ na otevřený text (tzv. „whitening“). Potom proběhne Nr rund a v každé z nich se použijí čtyři rundovní klíče. Na počátku se 16 bajtů otevřeného textu naplní postupně po sloupcích (tj. shora dolů a zleva doprava) do matice bajtů a na ně se ve stejném pořadí postupně „xoruje“ 16 bajtů tvořících první čtyři rundovní klíče.



Obrázek 4.8.2.1
Jedna runda zašifrování

Po té proběhne Nr rund podle pseudokódu na obrázku 4.8.2.1, kde „State“ znamená stav matice A. Připomeňme, že prvky matice A jsou sice bajty, ale při násobení jsou chápány jako prvky GF (2^8). „Sčítání“ těchto prvků (při operaci MixColumn) je běžná operace XOR. Výsledný šifrovací text se opět vybírá po sloupcích z matice A.

Hlavní transformace

Všechny rundy jsou stejné, až na poslední, kde je malá změna – neprovádí se operace mixování MixColumn. Nyní k jednotlivým operacím z obrázku 4.8.2.1. ByteSub je bajtová substituce (a→b), kterou aplikujeme na každý bajt a_{ij} matice A. Nejprve vypočíteme multiplikativní inverzi prvku a, tj. $c = a^{-1} \bmod m$ (x) a poté bajt c transformujeme na b substitucí S podle obrázku 4.8.2.1. Substituci nemusíme počítat podle tohoto vzorce, ale můžeme si ji uložit jako pevnou tabulku.

ShiftRow vykoná v matici A cyklickou rotaci jejich prvků v jednotlivých řádcích doleva, a to tak, že první řádek ponechá beze změny, druhý rotuje o jednu pozici, třetí o dvě a čtvrtý o tři pozice.

MixCoulmn zesložití prvky v rámci každého sloupce matice A. Vstupem této transformace jsou všechny prvky daného sloupce (na obrázku je označen a) a výstupem jejich nové hodnoty (b).

Nakonec se operací AddRoundKey na prvky matice A (opět po sloupcích) „naxorují“ po řadě jednotlivé bajty čtyř rundovních klíčů, které jsou nařadě. A to je celé.

Odšifrování probíhá trochu jinak než zašifrování, ale využívá jeho stavební prvky.

Zpracování klíče

Šifrovací klíč key (viz obr. 4.8.2.2) o Nk 32bitových slovech (4, 6 nebo 8) se naplní na počátek pomocného pole 32bitových slov W [0… Nk-1]. Toto pole se poté expanduje tak, že každé nové W je vypočítáno jako W [i] = W [i – Nk] XOR temp, kde temp je W [i -1] nebo jeho modifikace - viz. obrázek 4.8.2.2. Při modifikaci se využívá operace cyklického posuvu bajtů slova temp o jeden doprava (RotByte), dále nám známé substituce bajtů SubByte, a to aplikované na každý bajt proměnné temp, a pole konstant Const[].

Expanze klíče

```
for i = Nk to 4*Nr + 3 do
{
  temp = W[i - 1];
  if (i mod Nk = 0)
    temp = SubByte(RotByte(temp)) ⊕ Const[i / Nk];
  if ((i mod Nk = 4) AND (Nk = 8))
    temp = SubByte(temp);
  W[i] = W[i - Nk] ⊕ temp;
}
```

Obrázek 4.8.2.2
Expanze klíče

Bezpečnost

Autoři šifry dokazují skvělé vlastnosti stavebních bloků schématu i odolnost vůči lineární a diferenciální kryptoanalýze. Protože schéma pro šifrování a dešifrování se v hardware liší, není tu riziko slabých klíčů. Ekvivalence klíčů (což je případ, kdy různé šifrovací klíče dávají stejné sady rundovních klíčů) brání podle autorů nelineární expanzi.

Shrnutí

U šifry Rijndael je ceněn její průzračný návrh, založený na různých algebraických operacích. Šifra je flexibilní při realizaci na různých typech procesorů s velmi malými nároky na paměť i velikost kódu a přitom vykazuje ještě dostatečnou rychlost. Je vhodná i pro paralelní zpracování a je odolná proti fyzickým typům útoku.

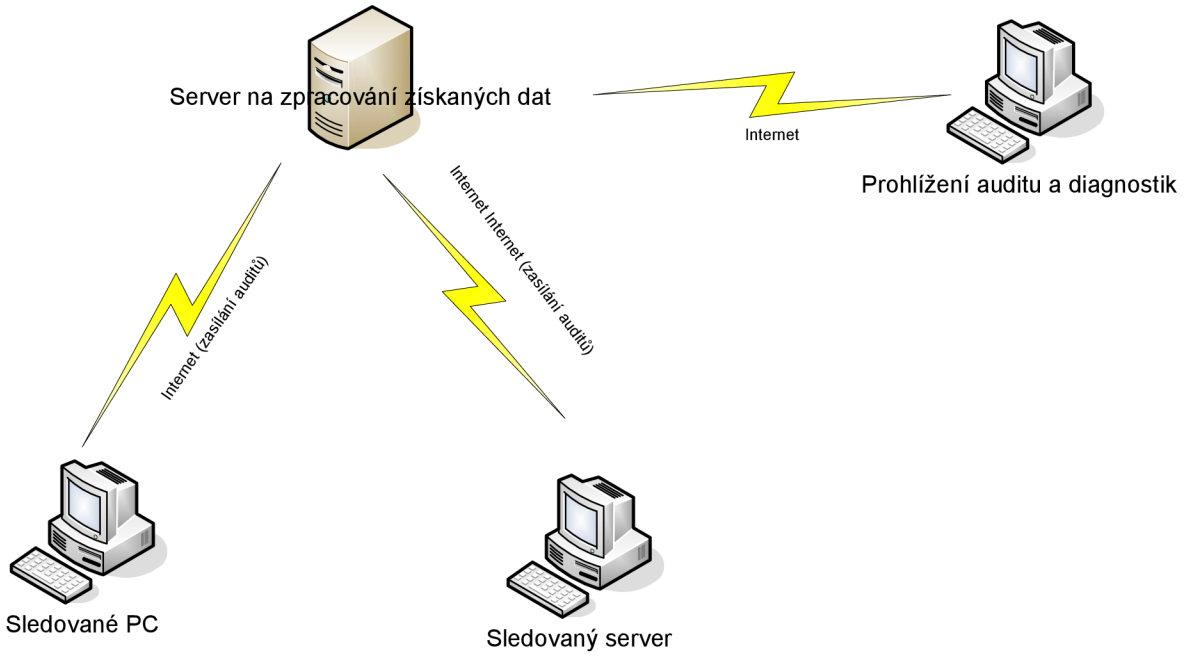
5 Architektura systému

V této kapitole bych chtěl představit základní komponenty celého systému, jejich komunikaci a kooperaci.

5.1 Základní architektura systému

Srdcem celého systému je server na ukládání a zpracování získaných dat. Tomuto serveru jsou posílány informace ze sledovaných počítačů a serverů.

Server, na kterém jsou ukládány audity a diagnostiky, poskytuje webové rozhraní přes které je možné prohlížení a analyzování získaných informací.



Obrázek 5.1.1
Základní architektura systému

5.2 Architektura klienta

Klientem jsem označil tu část systému, která je spouštěna na sledovaném počítači a v definovaných intervalech posílá data na server. Klient se skládá ze tří základních komponent. Jsou to komponenty:

- Skener
- Komponenta na instalaci skeneru
- Komponenta na odinstalovaní skeneru

První komponenta je samotný skener, který provádí skenování sledovaných parametrů počítače a jejich následné zasílání na server. Druhou komponentou je instalátor, pomocí něhož je možné nainstalovat skener. Třetí komponentou je komponenta na odinstalování klienta.

5.2.1 Komponenta skener

Tato komponenta je nejsložitější v celém systému. V tomto odstavci popíši její architekturu.

Skener se skládá z těchto komponent:

- Komponenta pro logování skeneru
- Komponenta zajišťující komunikaci s WMI
- Komponenta pro práci se soubory
- Komponenta pro získávání S.M.A.R.T atributů
- Komponenta pro kódování dat
- Komponenta pro FTP komunikaci
- Komponenta pro práci z registry
- Komponenta pro skenování parametrů počítače
- Komponenta pro řízení celé aplikace nazvaná jádro

Srdcem celého skeneru je jádro, které využívá další komponenty k dosažení svého cíle. Druhou nejpodstatnější komponentou je komponenta na skenování parametrů počítače. Tato komponenta využívá komponenty WMI a SMART k získávání informací o počítači.

Komponenta pro logování skeneru

Tato komponenta slouží k logování skeneru. Provádí logování do dvou souborů. Do prvního souboru loguje základní informace o běhu skeneru, jako jsou například informace o spouštění a ukončení skeneru. Do druhého souboru se provádí logování informací o běhu jádra.

Komponenta na komunikaci s WMI

Tato komponenta zajišťuje komunikaci s rozhraním WMI. Umožňuje klást WQL dotazy a zpracovávat data, která jsou vrácena rozhraním WMI.

Komponenta pro práci se soubory

Tato komponenta zajišťuje základní funkce pro práci se soubory.

Komponenta pro získávání S.M.A.R.T atributů

Tato komponenta zajišťuje získávání S.M.A.R.T atributů ze softwarového ovladače v operačním systému Windows.

Komponenta pro kódování dat

Tato komponenta zajišťuje šifrování datových souborů.

Komponenta pro FTP komunikaci

Tato komponenta zajišťuje posílání datových souborů přes FTP protokol.

Komponenta pro práci s registry

Tato komponenta zajišťuje práci s registry v operačním systému Windows.

Komponenta pro skenování parametrů počítače

Tato komponenta využívá další komponenty pro skenování parametrů počítače.

Komponenta pro řízení aplikace nazvaná jádro

Tato komponenta řídí celý skener. Zajišťuje spouštění jednotlivých operací na základě nastavených časových intervalů. Řídí operace skenování jednotlivých parametrů, komprimaci dat, kódování dat a jejich následné zasílání na server.

5.2.2 Komponenta na instalaci skeneru

Tato komponenta zajišťuje instalaci skeneru. Nejprve na základě souboru, který vygeneruje informační systém, vygeneruje konfigurační soubor pro skener. Poté se zapíše do registrů operačního systému hodnota, která zajišťuje spouštění skeneru po startu počítače.

5.2.3 Komponenta na odinstalování skeneru

Tato komponenta odstraňuje zapsanou hodnotu v registrech operačního systému. Pokud skener nenajde v registrech hodnotu, tak se ukončí.

5.2.4 Princip klienta

Po startu klienta se provede test na inicializaci WMI rozhraní. Pokud je možné WMI rozhraní inicializovat, tak se spustí jádro skeneru. V opačném případě se skener ukončí.

Po spuštění jádra se provede jeho inicializace. Inicializace se skládá z dekódování konfiguračního souboru a nastavení jádra do příslušného módu. Pokud je inicializace úspěšná, spustí se jádro.

Jádro nejprve otestuje, zda je aplikace nainstalovaná. Pokud ano, tak se v pravidelných intervalech kontroluje nastavení jednotlivých časovačů a podle nich se spouští skenování jednotlivých parametrů a odesílání dat na server. Data se ukládají do dočasného souboru. Na tento soubor je aplikováno zakódování. Soubor je odesílán na server, kde je dále zpracováván.

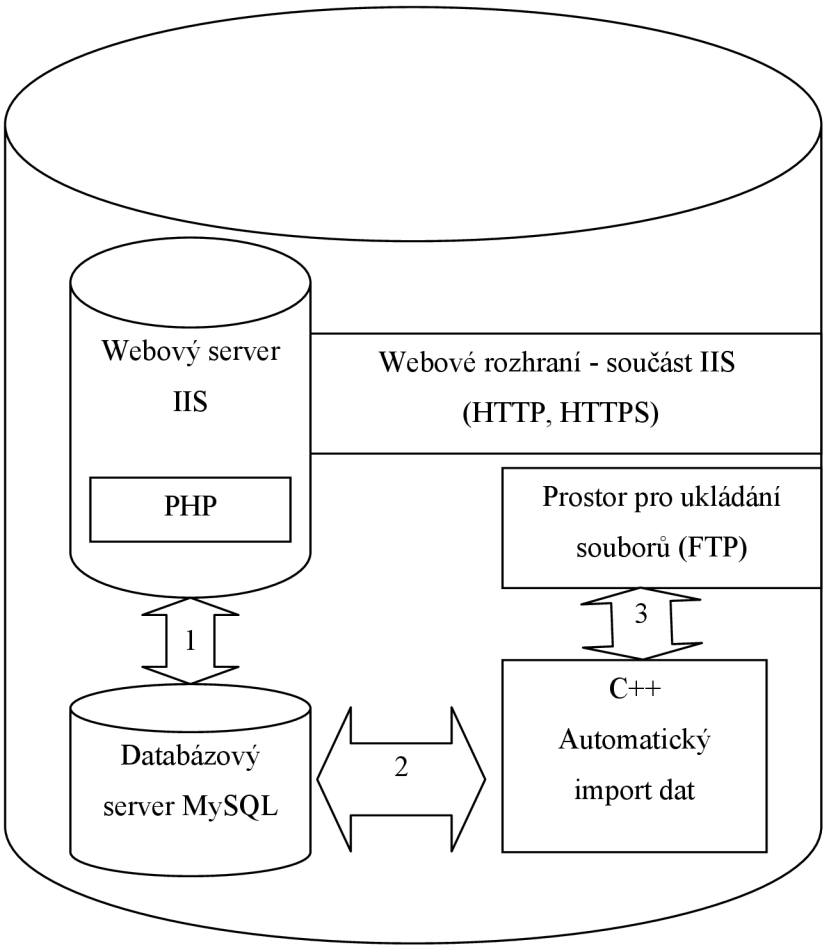
Jádro je také vybaveno mechanizmem pro odchytávání zpráv od operačního systému. Když se vypíná operační systém, tak je skeneru posílána zpráva, na kterou skener reaguje svým ukončením.

5.3 Architektura Serveru

Serverem jsem označil tu část systému, na kterou jsou zasílány informace od klientů. Server se stará o jejich zpracování a ukládání.

Informační systém slouží k ukládání a diagnostice získaných informací. Jako úložiště dat slouží databázový systém MySQL. Pro zobrazování dynamických webových stránek a generování statistik bude sloužit skriptovací jazyk PHP. Pro automatické vkládání dat do datového úložiště je používán programovací jazyk C++.

Schéma architektury serveru



Obrázek 5.3.1
Architektura serverové části

Popis rozhraní a komunikace

- ***Webové rozhraní***
  Jde o webové rozhraní, které poskytuje webový server, konkrétně webový server IIS od společnosti Microsoft. Přes toto rozhraní je možné zobrazovat webové stránky zobrazující audity, diagnostiky.

- ***Webový server***
  Jde o webový server IIS od společnosti Microsoft.

- ***PHP***
  Jde o skriptovací jazyk, který je nainstalovaný na webovém serveru IIS jako CGI skript. Přes skriptovací jazyk PHP je umožněno připojení k databázovému serveru MySQL a generování dynamických webových stránek.

- ***Databázový server MySQL***
  Databázový server k ukládání dat.

- ***Automatický import dat***
  Jde o program, který v pravidelných intervalech kontroluje obsah adresáře, kam se ukládají datové soubory, ve kterých jsou uloženy informace o auditech.

- ***Prostor na ukládání souborů***
  Jde o vyhrazený diskový prostor pro nahrávání souborů přes protokol FTP.

- ***Komunikace 1***
  Komunikace mezi PHP a MySQL.

- ***Komunikace 2***
  Komunikace mezi MySQL a C++. Tento program komunikuje s databází MySQL přes rozhraní API, které poskytuje databázový server.

- ***Komunikace 3***
  Program zpracuje soubor a pokud je zpracování úspěšné, nahraje jeho obsah do databáze.

5.4 Komunikace

Komunikace mezi klientem a serverem je prováděna pomocí nahrávání souborů na server. Klient v pravidelných intervalech nahrává informace na server. Server poté tyto soubory zpracuje a naimportuje do informačního systému. Na straně klienta je soubor zašifrován. Na straně serveru dešifrován. Šifrovací mechanizmus je pomocí sdíleného klíče. K šifrování je použit algoritmus Rijndael. Nároky na šifru z hlediska prolomení jsou malé, je to dáno cílovou skupinou uživatelů. Ale i tak jsem použil moderní šifrovací mechanizmus.

5.5 Informační systém serveru

Informační systém je postaven na skriptovacím jazyce PHP a databázovém systému MySQL.
Tato kombinace je v praxi velmi často využívána.

5.6 Import informací do IS

O automatický import informací do IS se stará program v jazyce C++. Tento program cyklicky kontroluje diskový prostor pro soubory, které zasílá klient. Pokud se objeví nový soubor, tak tento soubor dešifruje, dekomprimuje a zpracuje. Nahraje jeho obsah do databáze. Program provádí kontroly souborů z hlediska bezpečnosti. To znamená zejména strukturu a typ souboru.

5.7 Nastavení systému

Aby mohla klientská a serverová část komunikovat, tak se musí zajistit potřebná nastavení. Systém jsem navrhl tak, že se nejprve v IS musí založit nový počítač. Na základě každého založeného počítače jde stáhnout z IS konfigurační soubor, který obsahuje potřebná data, jako je například heslo pro šifrování a dešifrování nebo identifikační číslo.

Po stáhnutí konfiguračního souboru se tento soubor nahraje do adresáře skeneru. Po spuštění instalátoru se tento soubor nahraje a na základě zvolených parametrů se vygeneruje konfigurační soubor pro skener. Pokud je instalace korektní, tak se skener zapíše do registrů počítače.

Skener pracuje dle nastavení v konfiguračním souboru. Skenuje, šifruje a posílá data. Když na FTP server dorazí datový soubor, je tento soubor dešifrován na základě hesla, které je uloženo v informačním systému. Heslo se identifikuje podle názvu souboru. Každý skenovaný počítač má unikátní jméno datového souboru. Po dešifrování je prováděna kontrola podle identifikátoru v datovém souboru a identifikátoru uloženém v databázi.

6 Klientská část systému

Jak jsem již zmínil v minulé kapitole, klientská část systému se skládá ze tří základních komponent. Skeneru, instanční komponentě a odinstalační komponentě.V této kapitole detailně popíši jejich princip, funkci a implementaci.

6.1 Skener

Hlavní princip skeneru je na vývojovém diagramu, který je zobrazen na obrázku 6.1.1.1. Celý program pracuje v cyklu. V definovaných intervalech testuje jednotlivé časovače a dle jejich nastavení provádí danou činnost. Protože program pracuje v cyklu, tak musí být možné korektně ho ukončit. K ukončení programu vede pět možných cest:

- ***Specifická konfigurace skeneru.***
  Skener se sám ukončí, pokud je nastavena taková konfigurace, že již není co skenovat. Například pokud je skenování parametrů nastaveno na skenování po startu a ostatní parametry jsou vypnuté, není důvod, aby skener běžel a zabíral tak systémové prostředky počítače.

- ***Skener dostane od operačního systému žádost na ukončení.***
  Skener je vybaven sledovací částí, která odchytává zprávy od operačního systému. Pokud přijde zpráva na ukončení programu, tak se program korektně ukončí. Operační systém čeká několik sekund po odeslání zprávy na ukončení programu. Pokud se program sám neukončí, tak je ukončen operačním systémem. Mnou navržený program testuje zprávy vždy na začátku cyklu, tím je zabezpečeno, že se program neukončí v nějaké rozpracované situaci.

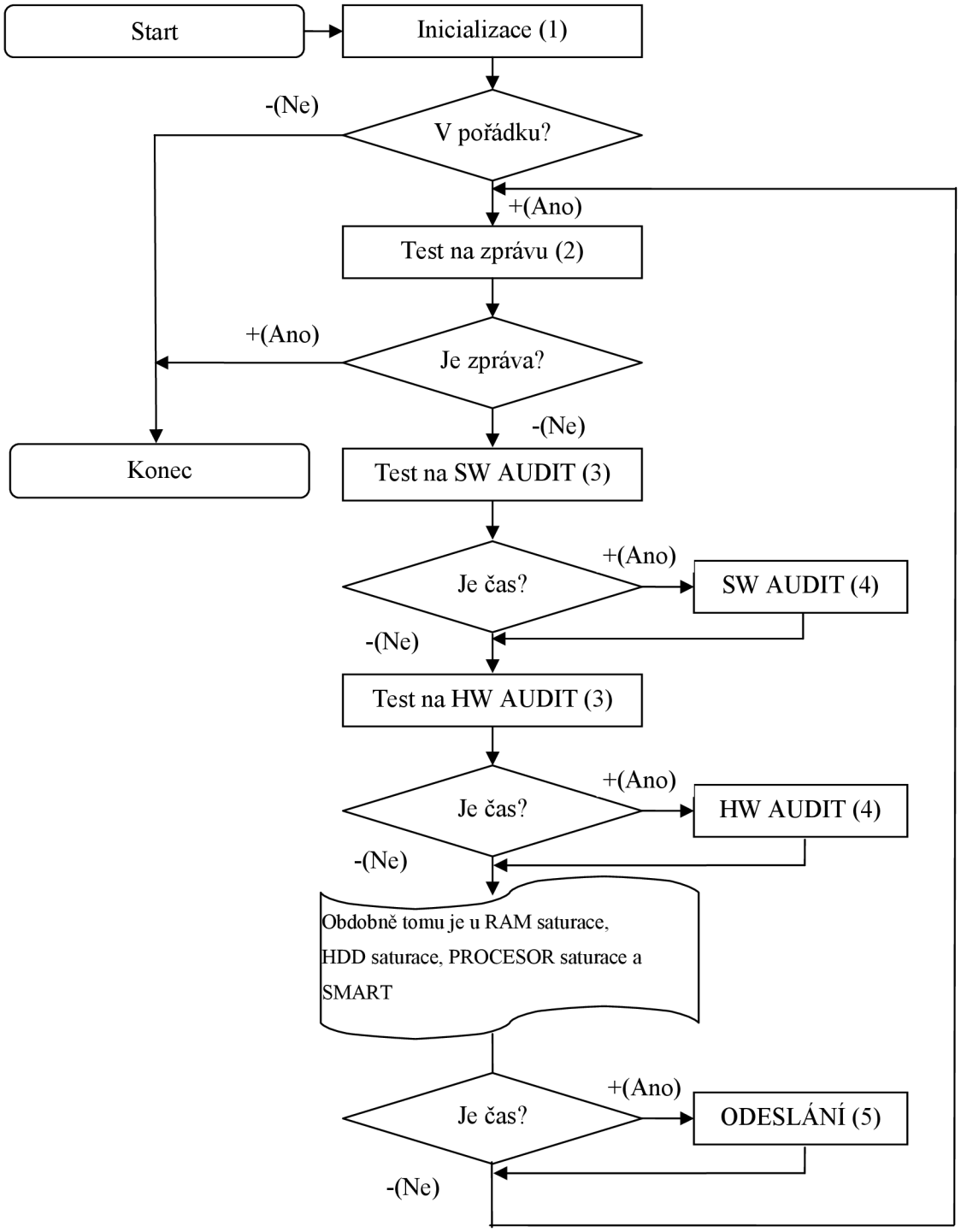
- ***Program není nainstalován.***
  Program obsahuje mechanizmus, kterým se provádí kontrola, zda je program nainstalován. Pokud program není nainstalován, tak se ukončí a do logovacího souboru vypíše hlášení, že se má program nejdříve nainstalovat. Toto se kontroluje nejen při startu programu, ale při každém cyklu. Pokud je program odinstalován, tak se automaticky ukončí.

- ***Program má špatné konfigurační údaje***
  Pokud se programu nepovede nastavit jádro vlivem špatných konfiguračních údajů, tak je program ukončen.

- *Vlivem chyby*
  Pokud nastane chyba při běhu programu nebo se například nepovede inicializovat rozhraní WMI.

6.1.1 Vývojový diagram skeneru

V tomto vývojovém diagramu je zachycena funkce skeneru, respektive jeho jádra. Program nejprve provede inicializaci (1). Pokud je v pořádku, tak program pokračuje, jinak se ukončí. Dále se testuje na zprávu od operačního systému (2). Pokud není zpráva, tak se pokračuje. Dále se provádí testování jednotlivých časovačů (3) a případně vykonání požadované funkce (4). Na konci cyklu se testuje odesílací časovač, pokud je podmínka splněna datový soubor se odešle (5).



Obrázek 6.1.1
Vývojový diagram skeneru

6.1.2 Implementace skeneru

V žádném popisu, pokud to není vyloženě důležité, nepopisuji konstruktor a destruktor. V konstruktoru provádím pouze inicializaci atributu třídy. A v destruktoru provádím kontrolu, zda jsou prostředky uvolněny a pokud ne, provádím jejich uvolňování. U každé třídy uvádím metody, které jsou významné. Rozhraní všech tříd je možné nastudovat ve zdrojových kódech programu.

Tato komponenta je nejsložitějším prvkem celého systému. Spolupracuje zde nejvíce tříd. Skener se skládá z těchto tříd:

- ***CWmi***
  Tato třída slouží jako rozhraní k WMI.

- ***CSMART***
  Tato třída slouží na získávání S.M.A.R.T atributů.

- ***SCanner***
  Tato třída provádí skenování údajů. Využívá pro to třídy CWmi a CSMART.

- ***CRijndael***
  Tato třída provádí šifrování datových souborů.

- ***CMyFTP***
  Tato třída slouží na zasílání dat na server pomocí protokolu FTP.

- ***Cfile***
  Tato třída slouží pro práci se soubory na pevném disku počítače.

- ***CLog***
  Tato třída slouží pro logování.

- ***Ckernel***
  Tato třída zajišťuje řízení celého skeneru.

- ***CInstaller***
  Tato třída se používá pro instalaci skeneru. Zde se však využívá pouze její jedné metody a tj. testování instalace skeneru.

6.1.2.1 CWmi

Před implementací této třídy jsem hledal nějaké hotové řešení pro komunikaci s rozhraním WMI. Žádnou knihovnu, která by odpovídala mým požadavkům jsem nenašel. Rozhodl jsem se, že celou knihovnu vytvořím.

Rozhraní třídy je implementováno v hlavičkovém souboru *wmi.h*. Rozhraní třídy poskytuje celkem pět privátních metod a dvě metody veřejné. Zde se zaměřím na detailní vysvětlení pouze u některých metod. Ostatní metody pouze nastíním. Při implementaci této třídy jsem využíval webových stránek společnosti Microsoft, přesný odkaz uvádím v literatuře [11]. Na stránkách má sice Microsoft ukázkové fragmenty programu, ale tyto fragmenty nejsou ošetřeny vůči neočekávaným situacím. Například neexistenci zadaného jmenného prostoru. Při implementaci třídy pro práci s WMI jsem byl velice opatrný co se týče ošetření přidělování a uvolňování paměti.

```
private:
      int mInitCOM();
      int mSetCOMSecLevel();
      int mInitLocatorWMI();
      int mSelectWmiNamespace(string wmiNamespace);

public:
      int mInitWmiInterface();
      int mWmiQuery(
            string wmiNamespace,
            string mujDotaz,
            string wmiName,
            vector<string> * result,
            unsigned int typeQuery
      );
```

Metody mInitCom, mSetComSecLevel a mInitLocatorWMI slouží k inicializaci WMI rozhraní a všechny informace o nich jsou na stránkách Microsftu [11]. Ostatní metody popíši detailněji v dalším textu.

mSelectWmiNamespace

```
int mSelectWmiNamespace(string wmiNamespace);
```

Tato metoda provádí výběr jmenného prostoru. Jako vstupní parametr má jmenný prostor a vrací buď 0, pokud je výběr v pořádku nebo kladné číslo, které indikuje příslušný kód chyby.

Metoda je naprogramovaná tak, že si pamatuje aktuální wmi namespace. Pokud se žádá o změnu a je již daný namespace vybraný, tak se znova výběr neprovádí. Výběr namespace se provádí pomocí funkce *ConnectServer*, jejíž přesná definice je na stránkách Microsoft. Přesný odkaz uvádím v literatuře [12].

mWmiQuery

```
int mWmiQuery(    string wmiNamespace, string mujDotaz,string wmiName,
                  vector<string> * result, unsigned int typeQuery);
```

Tato metoda provádí WQL dotazy. Jako vstup metody je název jmenného prostoru, wmi dotaz, sloupec který nás zajímá, typ výsledku dotazu a ukazatel na vektor, do kterého se ukládají data.

Metoda nejprve otestuje inicializaci WMI rozhraní. Pokud není inicializováno, tak se zavolá inicializace. Poté se provede test na výběr zadaného WMI. Pokud je rozdílný, tak se zavolá metoda *mSelectWmiNamespace*. Poté se provede WQL dotaz pomocí funkce *ExecQuery*, jejíž přesná definice je na stránkách Microsoft [13]. Následně je WMI dotaz zpracován. Podle toho v jakém formátu je očekáván výsledek dotazu, zda je to číslo nebo řetězec, se provede převod výsledku. Typ výsledku očekávaného dotazu se nastavuje pomocí *typeQuery*. Toto jsem zavedl, protože jsem se setkal s dotazy, které vracely hodnoty v různých formátech. WMI toto umožňuje. Nicméně docházelo buď k tomu, že výsledek byl nula, nebo k zamrzání programu. Dlouho jsem nemohl přijít na to co je špatně. Nakonec jsem WMI rozhraní odladil tak, aby bylo bezpečné a nezamrzalo, zkrátka v případě chyby vrací 1 nebo prázdný vektor, záleží na druhu chyby. Když jsem studoval tento problém, tak jsem nenašel žádný program, či fragment kódu, který by řešil složitější WQL dotazy. Všude byly ukázaný pouze jednoduché dotazy, které jsou triviální a bezproblémové.

6.1.2.2 CSMART

Tato třída zajišťuje zjišťování S.M.A.R.T atributů. Nejprve jsem toto chtěl udělat pomocí rozhraní WMI. Tam jsem se ale setkal se dvěma závažnými problémy. WMI po dotazu na S.M.A.R.T atributy vrací jakýsi vektor, ve kterém jsou všechny dostupné atributy zobrazeny. Nikde jsem ale nenašel, jak jsou jednotlivé atributy ve vektoru umístěny. Na stránkách společnosti Microsoft je napsáno, že by

umístění atributů ve vektoru měly poskytovat výrobci disků. Mně se tyto informace nepodařilo najít. Tak toto byl první problém a druhý spočíval v samotném zjišťování atributů. Testoval jsem to na třech různých počítačích. Na jednom počítači WMI vrátilo pouze jeden vektor, i když v PC byly nainstalovány dva disky. Na druhém počítači se podařilo vektor zjistit bezproblémově a na třetím nešel vektor zjistit vůbec. Nabízí se otázka, zda dané disky podporovaly technologii S.M.A.R.T. Odpověď je ano. Disponování technologie S.M.A.R.T jsem testoval programem Aktive SMART 2.51, který je v demoverzi k dispozici. Pro mé účely testování postačil. Nyní přede mnou byla otázka jakým směrem se vydat. Sledování S.M.A.R.T přes WMI se ukázalo jako špatná a neperspektivní cesta.

Hledal jsem tedy jiné možnosti. Nakonec jsem se rozhodl, že použiji funkce API operačního systému. Na stránkách, které uvádím v literatuře [14], jsem našel program, který provádí celkovou analýzu disku. V tomto programu jsem vyhledal funkce, které autor používá pro analyzování S.M.A.R.T atributu. Vyhledal jsem tyto funkce na stránkách společnosti Microsoft a nastudoval je. Poté co jsem třídu dokončil, tak jsem jí opět otestoval na již zmíněných počítačích a úspěšnost byla stoprocentní.

Rozhraní třídy je implementováno v hlavičkovém souboru *smart.h*. Nejvýznamnější metodou je metoda scannHdd, která provádí samotné získávání S.M.A.R.T atributů.

scannHdd

```
int scannHdd(vector<ssSMART> * mySmart);
```

Tato metoda naplní zadaný vektor S.M.A.R.T atributy. Metoda nejprve použije funkci *CreateFile*, aby získala proměnnou typu HANDLE pro přístup k objektu. Odkaz na tuto funkci uvádím v literatuře [15]. Tato funkce se používá pro vytvoření a otevření souborů na disku případně otevření COM portu. Pokud se podaří přístup k objektu, použije se funkce *DeviceIoControl*, aby se získaly potřebné informace. Odkaz na tuto funkci uvádím v literatuře [16]. Funkci musí být předány naplněné struktury, podle kterých vrací požadované informace.

6.1.2.3 CScanner

Tato třída se stará o získávání skenování všech parametrů. Využívá k tomu třídy CSMART a CWmi. Rozhraní třídy je definováno v hlavičkovém souboru *scanner.h*. Všechny metody mají jako vstupní parametr jméno dočasného souboru, do kterého se ukládají data. Metody využívají ke zjišťování informací rozhraní WMI.

```
int mScannOS(string filename);
int mScannSW(string filename);
int mScannRAM(string filename);
int mScannHDD(string filename);
int mScannProcessor(string filename);
int mScannCD(string filename);
int mScanGraphicCard(string filename);
int mScanLAN(string filename);
int mScanSOUND(string filename);
int mScanPrinter(string filename);
int RAMSaturation(string filename);
int HDDSaturation(string filename);
int ProcessorSaturation(string filename);
int SMART(string filename);
```

mScannOS

Zjišťování jména operačního systému:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `Select * from Win32_OperatingSystem, Caption`
Typ: `Řetezec`

Zjišťování sériového čísla operačního systému:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `Select * from Win32_OperatingSystem, SerialNumber`
Typ: `Řetezec`

mScannSW

Zjišťování názvu aplikací:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `Select * from Win32_Product, Caption`
Typ: `Řetezec`

Zjišťování verze aplikací:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `Select * from Win32_Product, SerialNumber`
Typ: `Řetezec`

mScannRAM

Zjišťování velikosti operační paměti:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `Select * from Win32_PhysicalMemory, Capacity`
Typ: `Řetezec`

Zjišťování sériového čísla modulu paměti:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `Select * from Win32_PhysicalMemory, SerialNumber`
Typ: `Řetezec`

mScannHDD

Zjišťování velikosti disku:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `Select * from Win32_DiskDrive, Size`
Typ: `Řetezec`

Zjišťování modelu disku:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `Select * from Win32_DiskDrive, Model`
Typ: `Řetezec`

mScannProcessor

Zjišťování modelu procesoru:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `SELECT * FROM Win32_Processor, Caption`
Typ: `Řetezec`

Zjišťování sériového čísla procesoru:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `SELECT * FROM Win32_Processor, ProcessorId`
Typ: `Řetezec`

mScannCD

Zjišťování CD mechanik:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `SELECT * FROM Win32_CDROMDrive, Caption`
Typ: `Řetezec`

mScanGraphicCard

Zjišťování grafických karet:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `SELECT * FROM Win32_DisplayConfiguration, Caption`
Typ: `Řetezec`

mScanLAN

Zjišťování síťových karet:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `SELECT * FROM Win32_NetworkAdapterConfiguration where IPEnabled = True, Description`
Typ: `Řetezec`

mScanSOUND

Zjišťování zvukových karet:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `SELECT * FROM Win32_SoundDevice, Name`
Typ: `Řetezec`

mScanPrinter

Zjišťování tiskáren:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `Select * from Win32_Printer where Network=false, Name`
Typ: `Řetezec`

RAMSaturation

Zjišťování kapacity paměti:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `Select * from Win32_PhysicalMemory, Capacity`
Typ: `Řetezec`

Zjišťování zaplnění paměti:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `SELECT * FROM Win32_PerfFormattedData_PerfOS_Memory, AvailableMBytes`
Typ: `Řetezec`

Zde se musí zjistit kapacita všech modulů pamětí, aby se získala celková kapacita paměti, kterou má k dispozici systém.

HDDSaturation

Zjišťování volného prostoru disku:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `SELECT * FROM Win32_PerfRawData_PerfDisk_LogicalDisk WHERE Name != '_Total', FreeMegabytes`
Typ: `Číslo`

Zjišťování názvu disku:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `SELECT * FROM Win32_PerfRawData_PerfDisk_LogicalDisk WHERE Name != '_Total', Name`
Typ: `Řetězec`

Zjišťování kolik je volných procent:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `SELECT * FROM Win32_PerfFormattedData_PerfDisk_LogicalDisk, PercentFreeSpace`
Typ: `Řetězec`

ProcessorSaturation

Zjišťování zatížení procesoru:
Namespace: `ROOT\\CIMV2`
WQL, jméno: `SELECT * FROM Win32_Processor, LoadPercentage`
Typ: `Číslo`

SMART

Zde se využije třída CSMART ke zjištění S.M.A.R.T atributů.

6.1.2.4 CRijndael

V projektu jsem měl zabezpečit data proti jejich zneužití. K tomuto účelu jsem využil šifrovací mechanismus Rijndael. Před implementací jsem se rozhodoval, zda budu nějaký algoritmus implementovat sám nebo zda použiji nějakou hotovou knihovnu. Nakonec jsem se rozhodl použít hotové zdrojové kódy. Na stránkách www.codeproject.com jsem nalezl hotové kódy a použil jsem je ve svém projektu. Přesný odkaz na tyto kódy uvádím v literatuře [17]. Rozhraní třídy je definováno v hlavičkovém souboru Rijndael.h.

6.1.2.5 Cfile

Tato třída slouží pro práci se soubory na pevném disku počítače. Její rozhraní je implementováno v hlavičkovém souboru *myFile.h*. Zde jsem použil standardní funkce pro práci se soubory. Jediné, co bych zde měl asi zmínit jsou metody na zápis vektorů do souboru. Protože v projektu se hodně pracuje s vektorovými daty, tak jsem vytvořil metody na zápis jednoho, dvou a třech vektorů do souboru. Vysvětlím to na jednoduchém příkladě, kdy se mají zapisovat informace o pevném disku. První skenování zjistí jména všech disků, druhé skenování zjistí velikosti všech disků. Pokud je v počítači instalováno více disků, mohutnost vektorů bude závislá na počtu disků. Do datového souboru musím zapsat data tak, aby byl dodržen přesný formát dat. Při zápisu vektoru se testuje, zda mají stejnou mohutnost, pokud ano, není se zápisem problém. Může však nastat situace, že nelze zjistit velikosti pevných disků, potom jsou mohutnosti rozdílně. Můj program to řeší tak, že místo, kde by měly být velikosti disku, nahrazuje speciálním znakem. Tak je zajištěna konzistence dat.

6.1.2.6 Clog

Tato třída slouží na logování informací do souboru. Její rozhraní je implementováno v hlavičkovém souboru *log.h*.

6.1.2.7 CKernel

Tato třída řídí celý skener. Její rozhraní je implantováno v hlavičkovém souboru *kernel.h*. Vzhledem k tomu, že je tato třída z hlediska funkce skeneru nejdůležitější, tak popíši všechny její metody.

```
private:
      bool mSetNextTime(
            unsigned int * myTime,
            BYTE hours,
            BYTE times,
            BYTE seconds
      );
      bool mIsTime(unsigned int myTime);
      int  mInitialization(void);

public:
      cKernel(void);
      ~cKernel(void);
      int mTestWmiInterface(void);
      int mStart(void);
```

cKernel

```
cKernel(void);
```

Zde popisuji konstruktor, protože jsem měl problémy s implementací. V konstruktoru jsem prováděl inicializaci některých částí skeneru. Při testování skener fungoval bez problémů. Když jsem však vytvářel instalační komponentu a testoval spouštění skeneru po startu počítače, skener nebyl stabilní. Nakonec jsem zjistil, že se program nestandardně ukončuje při vytváření instance třídy cKernel. Jak víme, nejprve se volá konstruktor třídy. Proto jsem se zaměřil na toto místo v programu. Nakonec byl závěr takový, že jsem přesunul některé inicializace z konstruktoru do inicializační metody. Protože mě toto zarazilo, tak jsem hledal nějakou literaturu, kde by o tomto problému bylo napsáno více. V knize C++ 101 programovacích technik (v literatuře [18]), jsem se dočetl, že by se v konstruktoru měly provádět pouze základní inicializační operace, složitější operace mohou způsobovat zmíněný problém.

mInitialization

```
int  mInitialization(void);
```

V této metodě se provádí některé inicializace a nastavení všech parametrů jádra.

mIsTime

```
bool mIsTime(unsigned int myTime);
```

Tato metoda slouží ke zjištění, zda je čas na provedení další operace. Metoda vrací *true* pokud je čas a *false* pokud čas není. Metoda využívá strukturu *time_t,* ve které je uložen systémový čas. Přesný odkaz na definici funkce uvádím v literatuře [19].

mSetNextTime

```
bool mSetNextTime(unsigned int * myTime, BYTE hours, BYTE times,
                  BYTE seconds
);
```

Tato metoda nastavuje časovač, jehož ukazatel se předá do parametru myTime na čas dalšího cyklu. Čas dalšího cyklu je závislý od hodnot, které se zadají do ostatních parametrů metody.

mTestWmiInterface

```
int mTestWmiInterface(void);
```

Tato metoda provádí testování inicializace rozhraní WMI.

mStart

```
int mStart(void);
```

Tato metoda spouští jádro. Jádro funguje dle vývojového diagramu, který je zobrazený na obrázku 6.1.1. Zde bych chtěl vysvětlit princip mechanizmu na odchytávání zpráv od systému. Na stránkách společnosti Microsoft jsem hledal nějaký vhodný způsob, jak bych toto mohl zajistit. Zjistil jsem, že operační systém Windows poskytuje více funkcí na odchytávání zpráv. Jeden druh je blokující a druhý neblokující. Blokující čeká dokud nedorazí zpráva od systému. Neblokující nečeká, pouze kontroluje buffer, do kterého chodí zprávy od operačního systému. Já jsem ve své implementaci použil neblokující odchytávání zpráv. Funkce, kterou využívám, se jmenuje *PeekMessage* a přesný odkaz uvádím v literatuře [20].

6.1.2.8 CInstaller

Ve skeneru se využívána pouze jedna metoda třídy CInstaller. Je to metoda isInstall, která vrací true, pokud je program zaregistrovaný v registrech operačního systému. Přesný popis této třídy uvedu v kapitole Komponenta na instalaci skeneru.

6.1.2.9 CMyFTP

Při návrhu jsem se rozhodoval mezi dvěma variantami. První varianta spočívala v použití externího programu na FTP komunikaci, který poskytuje operační systém Microsoft Windows. Tato varianta měla výhodu v tom, že program byl již hotový a stačilo by pouze vytvořit jednoduchý skript, který by zasílal datový soubor na server. Velká nevýhoda externího volání programu je pomalejší rychlost a nemožnost zpětně kontrolovat a programově reagovat na případné potíže se zasíláním dat. Další velkou nevýhodou je velmi malá bezpečnost. Druhá varianta spočívala v implementaci algoritmu zasílaní dat přímo do skeneru. Tato varianta má výhodu v tom, že program má zpětnou vazbu, má přehled o připojení k Internetu a o tom, zda datový soubor dorazil na server. Bezpečnost tohoto řešení je mnohem lepší, než u první varianty. Jedinou nevýhodou je, že se musí algoritmus zasílaní dat implementovat. Rozhodl jsem se pro druhou variantu.

Nejprve jsem hledal na Internetu nějakou knihovnu, kde by již byla implementována komunikace FTP protokolu. Zjistil jsem, že takových knihoven je moc, ale zpravidla to byly knihovny, které řešily kompletní FTP komunikaci nebo jejich použití bylo podmíněno finančním zaplacením. Vzhledem k tomu, že v projektu potřebuji pouze zasílání dat na FTP server a ne celé obsáhlé knihovny s FTP komunikací, tak jsem se nakonec rozhodl pro vlastní implementaci algoritmu. Třída, která řeší zasílání souborů na FTP server se jmenuje *CmyFtp* a její rozhraní je implementováno v hlavičkovém souboru *myFtp.h*.

Rozhraní třídy

Rozhraní třídy je implementováno v hlavičkovém souboru *myFtp.h*. Rozhraní třídy poskytuje celkem dvě metody:

```
bool isInternetConnection(void);
int sendFile(
      string serverName,
      unsigned int port,
      string user,
      string password,
      string localFile,
      string serverFile
      );
```

Metoda *isInternetConnection* slouží k testování připojení do sítě Internet. Tato metoda vrací 0 pokud existuje spojení. Pokud spojení neexistuje vrací hodnotu 1. Tato metoda je privátní metoda třídy.

Metoda *sendFile* slouží k zaslání souboru na FTP server. Tato metoda má jako vstupní parametry:

- serverName – jméno nebo IP adresa FTP serveru
- port – číslo portu, na kterém běží FTP server (zpravidla port 21)
- user – uživatelské jméno pro připojení na server
- password – uživatelské heslo pro připojení na server
- localFile – název, respektive cesta k souboru na lokálním počítači
- serverFile - název, respektive cesta k souboru na FTP serveru

Jako výstupní parametr vrací hodnotu, která indikuje úspěšnost provedení. Pokud je vše v pořádku, tak metoda vrací hodnotu 0, jinak vrací hodnotu větší než 0.

Implementace metod třídy

Funkce, které jsem použil pro implementaci FTP komunikace, jsou implementovány v knihovně Wininet.lib, hlavičkový soubor se jmenuje Wininet.h. Všechny informace o této knihovně jsou popsány na stránkách společnosti Microsoft [21].

Test připojení do sítě Internet

K testování spojení slouží tato metoda:

```
bool isInternetConnection(void)
```

Pro testování připojení do Internetu jsem použil funkci *InternetGetConnectedState*, jejíž definice je na stránkách společnosti Microsoft [22]. Funkce vrací true, pokud existuje spojení do sítě Internet a do jednoho ze svých parametrů ukládá typ spojení, například lan nebo modem.

Připojení a zaslání souboru na FTP server

K připojení a zasílání dat slouží tato metoda:

```
int sendFile(
      string serverName,
      unsigned int port,
      string user,
      string password,
      string localFile,
      string serverFile
      );
```

Metoda nejprve otestuje vstupní parametry. Pokud jsou všechny parametry vyplněny, tak otestuje spojení pomocí privátní metody *isInternetConnection*, pokud existuje spojení, tak otevře internetové spojení. Pomocí funkce *InternetOpen*, jejíž definice je na stránkách společnosti Microsoft [23]. Metoda vrací handle typu HINTERNET na internetové spojení. Pokud se spojení nepovede, handle má hodnotu false.

Po otevření spojení je volána funkce InternetConnect, jejíž definice je na stránkách společnosti Microsoft [24]. Jako vstupní parametr funkce jsou handle na internetové spojení, název serveru, port služby serveru, uživatelské jméno, uživatelské heslo, příznak zda jde o FTP nebo http přenos a příznak pasivního spojení. Funkce vrací handle typu HINTERNET. Pokud je handle roven false, spojení se nepovedlo. Po navázání spojení s FTP serverem je volána funkce FtpPutFile, jejíž definice je na stránkách společnosti Microsoft [25]. Jako vstupní parametry funkce jsou handle na spojení s FTP serverem, jméno souboru na lokálním počítači, jméno souboru na FTP serveru, příznak zda jde o binární nebo textová data a příznak zda se má soubor na FTP serveru, pokud existuje, přepsat. Pokud přenos souboru neproběhl, tak funkce vrací false, pokud přenos proběhl, tak funkce vrací true. Na serveru je poté soubor dále zpracováván.

6.2 Komponenta na instalaci skeneru

Instalátor funguje tak, že nastaví konfigurační soubor pro skener a zapíše potřebné hodnoty do registrů operačního systému. Konfigurační soubor pro skener je v šifrované podobě. Nelze tedy poznat na jaký server se budou posílat data ani jaké je uživatelské heslo nebo jméno pro přístup k FTP serveru.

Celý instalátor se skládá z těchto tříd:

- ***CMenu***
  Tato třída slouží pro zobrazování menu na obrazovce. Vypisuje pouze texty na obrazovku a reaguje na dotazy uživatele.

- ***CFile***
  Tato třída slouží pro práci se soubory a byla již popsána.

- ***CInstaller***
  Tato třída obsahuje metody na instalaci skeneru.

- ***CRegistry***
  Tato třída obsahuje metody pro práci s registry.

- ***CRijndael***
  Tato třída slouží na šifrování dat, byla již popsána.

Popíši pouze třídy CInstaller a CRegistry, ostatní již byly popsány, nebo nejsou důležité.

6.2.1.1 CInstaller a CRegistry

Třída CInstaller využívá třídu CRegistry pro práci s registry. Třídu CRegistry jsem použil již hotovou a odkaz na jejího autora uvádím v literatuře [26]. Třída CRegistry využívá třídu CString, odkaz na jejího autora uvádím v literatuře [29]. Rozhraní třídy CInstaller je implementováno v hlavičkovém souboru *installer.h*.

Třída CInstaller obsahuje tyto metody:

```
bool Install(TCHAR * fileName);
bool Uninstall();
bool IsInstalled();
```

Install

```
bool Install(TCHAR * fileName);
```

Protože se má program spouštět po startu počítače, hledal jsem nějaké řešení tohoto problému. V podstatě mě napadli dvě řešení. Buď vytvořit zástupce a tohoto zástupce vložit do programů po spuštění, nebo zapsat cesty programu do registrů. Druhá varianta je jistě lepší. Pro běžného uživatele je tato informace neviditelná. Hledal jsem tady nějaký článek, ve kterém by bylo popsáno, do jakého registru se má zapsat cesta k programu. Takový článek jsem našel na webových stránkách www.builder.cz. Přesný odkaz uvádím v literatuře [27]. Protože jsem chtěl, aby byl program spouštěn pouze pod uživateli, kde je nainstalován, musí se hodnota zapsat do registru:

HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Bylo by možné zajistit i spouštění pod všemi uživateli, ale to by neplnilo svůj účel. Každý uživatel počítače má svůj účet a za tento účet je zodpovědný.

A nyní k samotné metodě. Metoda Install nastaví do příslušného registru cestu na zadaný soubor. Metoda nejprve zjistí aktuální adresář a přidá k němu jméno souboru. Pro tuto informaci vytvoří klíč v registrech a zapíše ji tam. Tím je zajištěno, že se program spouští po startu.

Uninstall

Tato metoda odebere, pokud existuje, hodnotu z registrů operačního systému.

IsInstalled

Tato metoda provádí kontrolu registrů a pokud je program zaregistrován, tak vrací true.

6.3 Komponenta na odinstalaci skeneru

Tato komponenta je velice jednoduchá. Využívá třídu install na odinstalování aplikace. Třída CInstall již byla popsána v předešlé kapitole.

7 Serverová část

Serverová část systému je tvořena dvěmi hlavními komponentami. První komponentu tvoří program, implementovaný v jazyce C++, který cyklicky provádí kontrolu FTP prostoru na přítomnost nových datových souborů. Pokud se na FTP objeví nový datový soubor, tak jej program zpracuje a příslušná data uloží do informačního systému, respektive do datového úložiště. Druhou komponentu tvoří IS implementovaný ve skriptovacím jazyce PHP. Jako datové úložiště složí databázový systém MySQL.

Při návrhu serverové části systému jsem se rozhodoval mezi dvěmi možnostmi řešení. První možnost spočívala v implementaci celé serverové části ve skriptovacím jazyce PHP. Druhá možnost spočívala v rozdělení na dva celky a každý celek řešit pomocí jiné technologie. První komponentu, který zpracovává zasílané soubory, implementovat v jazyce C++ a druhou komponentu, informační systém, implementovat pomocí skriptovacího jazyka PHP.

Zamýšlel jsem se nad výhodami jednotlivých řešení a hledal jsem možné knihovny na realizaci serverové části.

U první varianty, kdy je celý systém implementován v jednom jazyce je výhodné, že se nemusí řešit komunikace mezi PHP a MySQL. PHP má v sobě potřebné knihovny na komunikaci s databázovým úložištěm. Další výhodou této varianty je použití pouze dvou technologií. Co se týká nevýhod, tak je to jistě slabá podpora operací v rámci operačního systému, jako jsou například diskové operace. PHP je zkrátka skriptovací jazyk a je určen pro tvorbu dynamických webových stránek. Při implementaci by mohl například vzniknout problém při nastavení pravidelného spouštění skriptu na zpracovávání datových souborů.

U druhé varianty je výhoda, že C++ je programovací jazyk, pomocí kterého lze efektně využívat funkce aplikačního rozhraní operačního systému Windows. Nevýhoda je, že se musí zajistit komunikace mezi C++ a databázovým systémem MySQL. Komunikace mezi MySQL a PHP je bezproblémová, zatímco komunikace mezi C++ a MySQL není tak přímá a jednoduchá. Tento problém lze vyřešit pomocí aplikačního rozhraní, které poskytuje databázový systém MySQL. Nevýhoda této varianty řešení je, že se musí skloubit tři technologie: C++, MySQL a PHP.

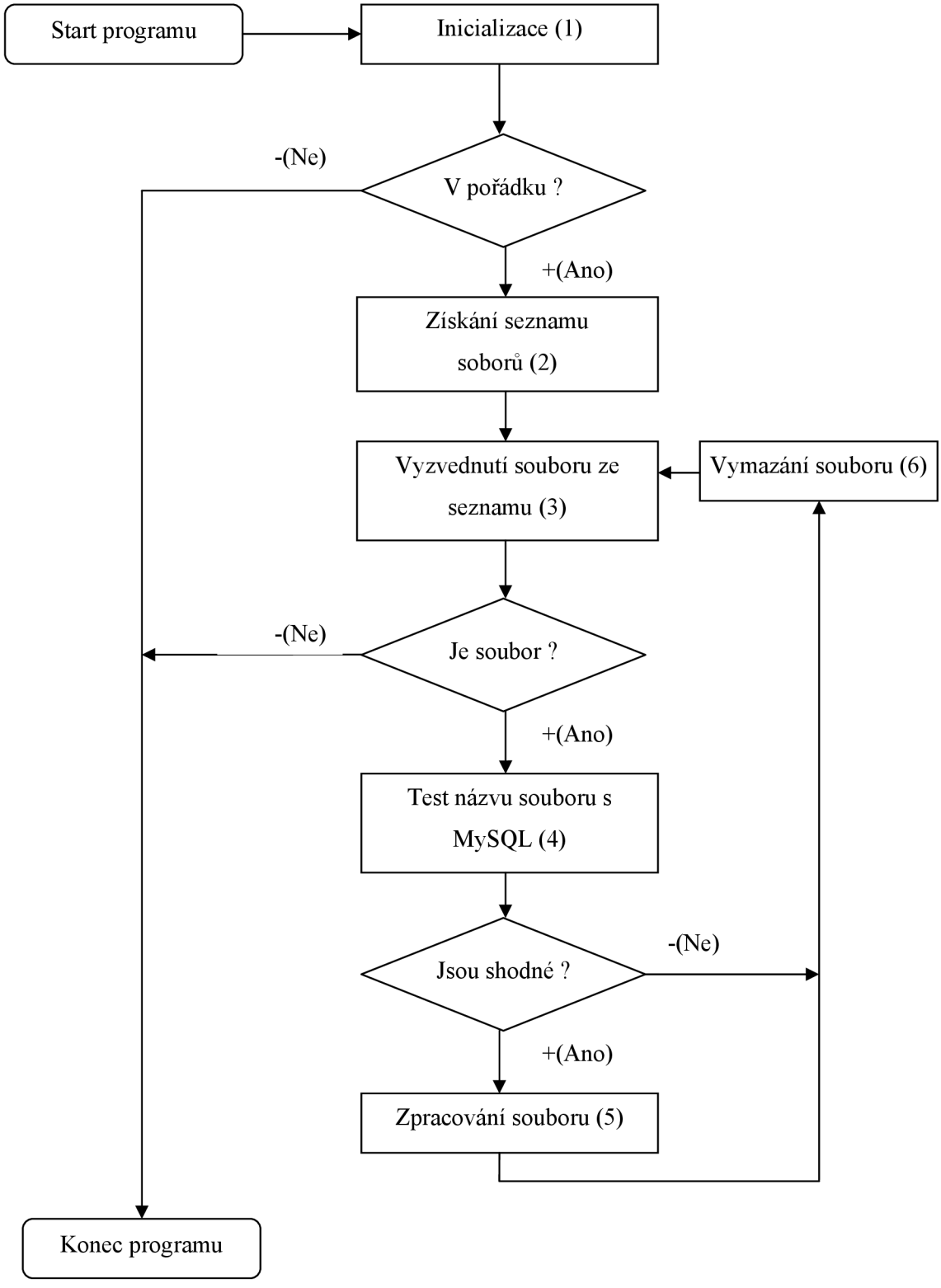
Nakonec jsem si vybral druhou možnost. Implementace v jazyce C++ je bezpečnější a poskytuje daleko větší možnosti v rámci operačního systému. Skriptovací jazyk PHP jsem využil pouze pro generování internetových stránek.

7.1 Zpracování datových souborů - parser

K implementaci programu na zpracovávání datových souborů jsem využil programovací jazyk C++. Program je spouštěn operačním systémem v přesně definovaných intervalech. Program si nejprve z konfiguračního souboru načte příslušné parametry. Dle nich provádí testovaní FTP prostoru na daném místě a daného souboru. Program si vytvoří seznam všech souborů na FTP prostoru, které odpovídají masce a mohly by být produkty klientských skenerů. Nezpracovává tedy všechny soubory, je to ohledně bezpečnosti. Program postupně prochází seznam souborů a zpracovává každý soubor. Před zpracováváním souboru se nejprve provede jeho dešifrování. Po dešifrování se provádí syntaktická kontrola souboru. Pokud soubor neodpovídá, tak se dále nezpracovává a je z FTP prostoru smazán. Pokud je syntaktická kontrola v pořádku, tak se začnou ukládat informace do databázového úložiště. Všechny řádky, které jsou vkládány do databázového úložiště jsou kontrolovány na nebezpečné znaky jako je například středník, který by mohl způsobit potíže. Tyto nebezpečné znaky jsou nahrazovány znakem mezera. Po zpracování datového souboru se přejde ke zpracování dalšího datového souboru v seznamu. Pokud již není žádný soubor v seznamu, tak se program ukončí.

7.1.1 Vývojový diagram parseru

Vývojový diagram zpracování souboru



Obrázek 7.1.1.1
Diagram parseru

Význam jednotlivých kroků diagramu

- ***Inicializace (1)***
  Inicializace se skládá z načtení konfiguračních údajů z konfiguračního souboru. Konfigurační soubor obsahuje:
  - Absolutní cestu do adresáře, kam se ukládají soubory od klientských skenerů.
  - Název MySQL serveru, zpravidla to bývá localhost.
  - Uživatelské jméno pro přístup k MySQL.
  - Uživatelské heslo pro přístup k MySQL.
  - Číslo portu na kterém je MySQL.
  - Název databáze.

  Tyto informace slouží k lokalizaci datových souborů a pro komunikaci s aplikačním rozhraním databázového serveru MySQL.

- ***Získání seznamu souborů (2)***
  Na základě inicializace si program vytvoří seznam souborů, které jsou na FTP prostoru a odpovídají dané masce. Masku jsem určil jako audit*.dat. Kde hvězdička reprezentuje libovolný počet libovolných znaků. Pokud se na FTP prostoru vyskytují nějaké další soubory, které nevyhovují masce, jsou ignorovány. Mohou sloužit pro analýzu možného proniknutí.

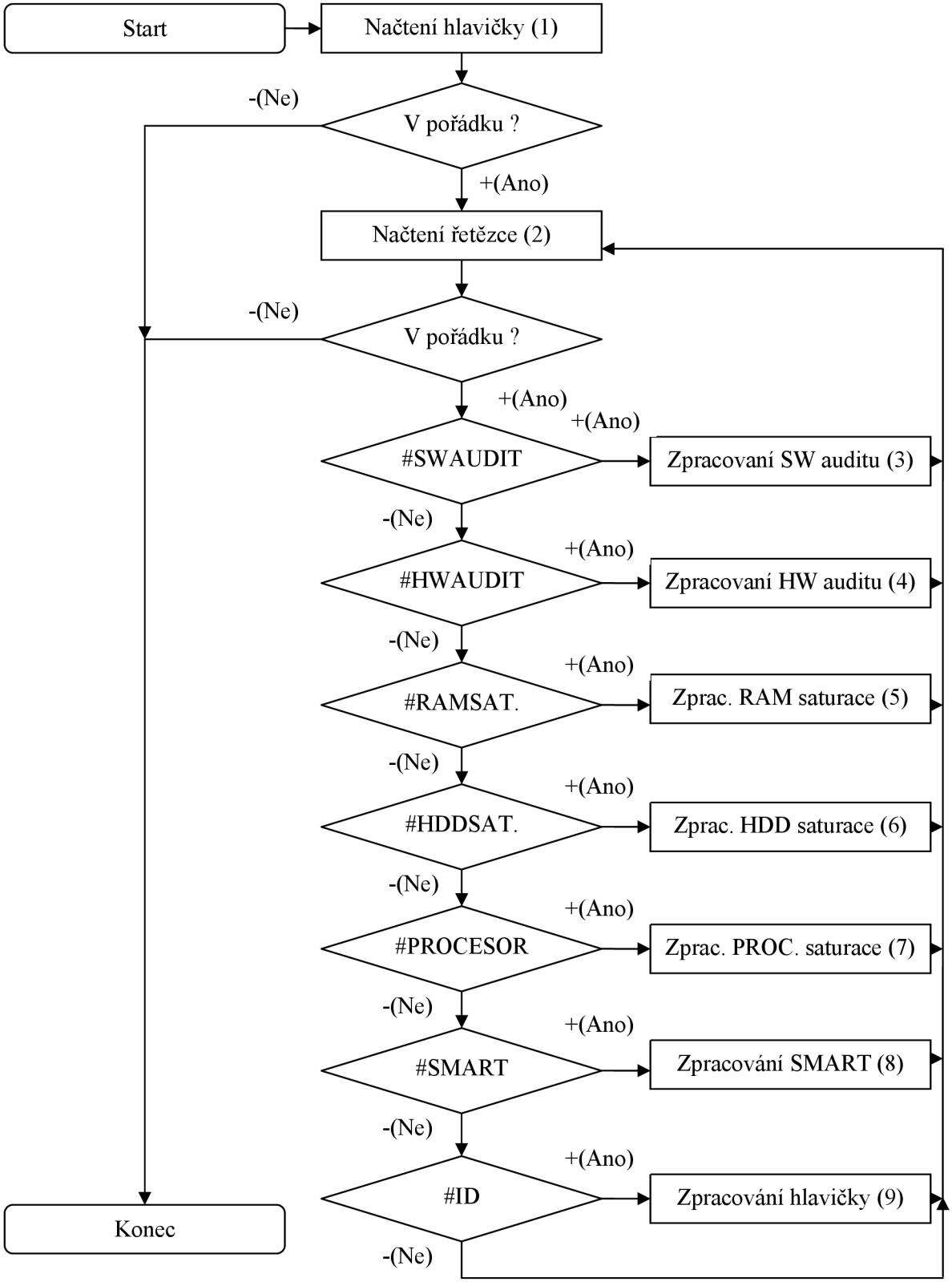
- ***Vyzvednutí souboru ze seznamu (3)***
  Jde o získání názvu, respektive absolutní cesty k dalšímu datovému souboru. Pokud není možné žádný další soubor ze seznamu vyzvednout, program se ukončí.

- ***Test názvu souboru s MySQL (4)***
  V tomto kroku se ještě více zužuje seznam potenciálních souborů. Název daného souborů se porovnává s názvy uloženými v datovém úložišti. Pokud název souboru není uložen v databázovém úložišti, soubor je odmítnut a vymazává se. Pokud je název uložen, tak je na jeho základě získáno ID, které musí odpovídat uloženému ID v souboru. V tuto chvíli ale ještě toto ID ze souboru není známo, protože je soubor zašifrován. A jako další parametr se získá heslo, pomocí kterého se provede dešifrování soubor.

- ***Zpracování souboru (5)***
  V tomto kroku se provede dešifrování souboru. Zkontroluje se hlavička souboru a jeho ID. Pokud je vše v pořádku, tak se provádí parsování a nahrávání dat do datového úložiště. Všechna data se kontrolují na výskyt nebezpečných znaků.

- ***Vymazáni souboru (7)***
  Jde o vymazávání analyzovaných, případně zpracovaných souborů.

Vývojový diagram zpracování souboru



Obrázek 7.1.1.2

Diagram zpracování datového souboru

Význam jednotlivých kroků diagramu.

Vývojový diagram předpokládá, že je již soubor dešifrován.

- ***Načtení hlavičky (1)***
  Načtení a kontrola hlavičky datového souboru. Na této hlavičce se provádí kontrola, pokud není výsledek kontroly pozitivní, soubor je odmítnut.

- ***Načtení řetězce (2)***
  V tomto kroku se načte řetězec z datového souboru. Tento řetězec se dále porovnává a kontroluje. Po hlavičce musí následovat jedna z těchto značek:
  - #SWAUDIT
  - #HWAUDIT
  - #RAMSATURACE
  - #HDDSATURACE
  - #PROCESORSATURACE
  - #SMART
  - #ID

  Na základě těchto značek je detekováno, co má parser zpracovávat a jaký má předpokládat formát dat.

- ***Zpracování SW auditu (3)***
  Zpracování SW auditu začíná značkou #SWAUDIT. SW audit obsahuje informace o nainstalovaných aplikacích a operačních systémech. Po značce #SWAUDIT následuje vždy datum provedeného skenování. Po datu může následovat jedna z těchto značek:
  - #APLIKACE
    Tato značka říká, že budou následovat informace o nainstalovaných aplikacích.

  - #OS
    Tato značka říká, že budou následovat informace o nainstalovaných operačních systémech.

  - #END.
    Tato značka oznamuje konec softwarového auditu.

  Pokud daný formát není splněn je ohlášena chyba.

- ***Zpracování HW auditu (4)***
  Zpracování HW auditu začíná značkou #HWAUDIT. HW audit obsahuje informace o nainstalovaném hardware. Po značce #HWAUDIT následuje vždy datum provedeného skenování. Po datu může následovat jedna z těchto značek:
  - #RAM
    Tato značka říká, že budou následovat informace o nainstalovaných modulech operačních pamětí.

  - #HDD
    Tato značka říká, že budou následovat informace o nainstalovaných pevných discích.

  - #PROCESOR
    Tato značka říká, že budou následovat informace o nainstalovaném procesoru.

- #CD
  Tato značka říká, že budou následovat informace o nainstalovaných optických mechanikách.

- #GRAFIKA
  Tato značka říká, že budou následovat informace o nainstalovaných grafických a multimediálních kartách.

- #LAN
  Tato značka říká, že budou následovat informace o nainstalovaných síťových kartách.

- #ZVUK
  Tato značka říká, že budou následovat informace o nainstalovaných zvukových kartách.

- #TISKARNA
  Tato značka říká, že budou následovat informace o nainstalovaných tiskárnách.

- #END
  Tato značka oznamuje konec hardwarového auditu.

Pokud daný formát není splněn je ohlášena chyba.

- ***Zpracování RAM saturace (5)***
  Po značce #RAM_SATURACE vždy následuje datum. Po datu následují informace o kapacitě operační paměti a kolik je v danou chvíli k dispozici. Po těchto informacích následuje značka #END, která oznamuje konec.

- ***Zpracování HDD saturace (6)***
  Po značce #HDD_SATURACE vždy následuje datum. Po datu následuje vždy název disku. Poté kolik je volného místa na disku a nakonec kolik je to procent. Z volného místa a informace kolik je to procent se dopočítává celková kapacita disku. Tento trošku krkolomný způsob je dán problematickým zjišťováním velikostí oddílu disku. Po těchto informacích následuje značka #END, která oznamuje konec.

- ***Zpracování PROCESOR saturace (7)***
  Po značce #PROCESOR_SATURACE vždy následuje datum. Po datu následuje zatížení procesoru v procentech. Po zatížení následuje značka #END, která oznamuje konec.

- ***Zpracování SMART (8)***
  Po značce #SMART následuje vždy datum. Po datu následuje název modelu disku. Po modelu disku mohou následovat atributy technologie S.M.A.R.T uvozené těmito značkami:

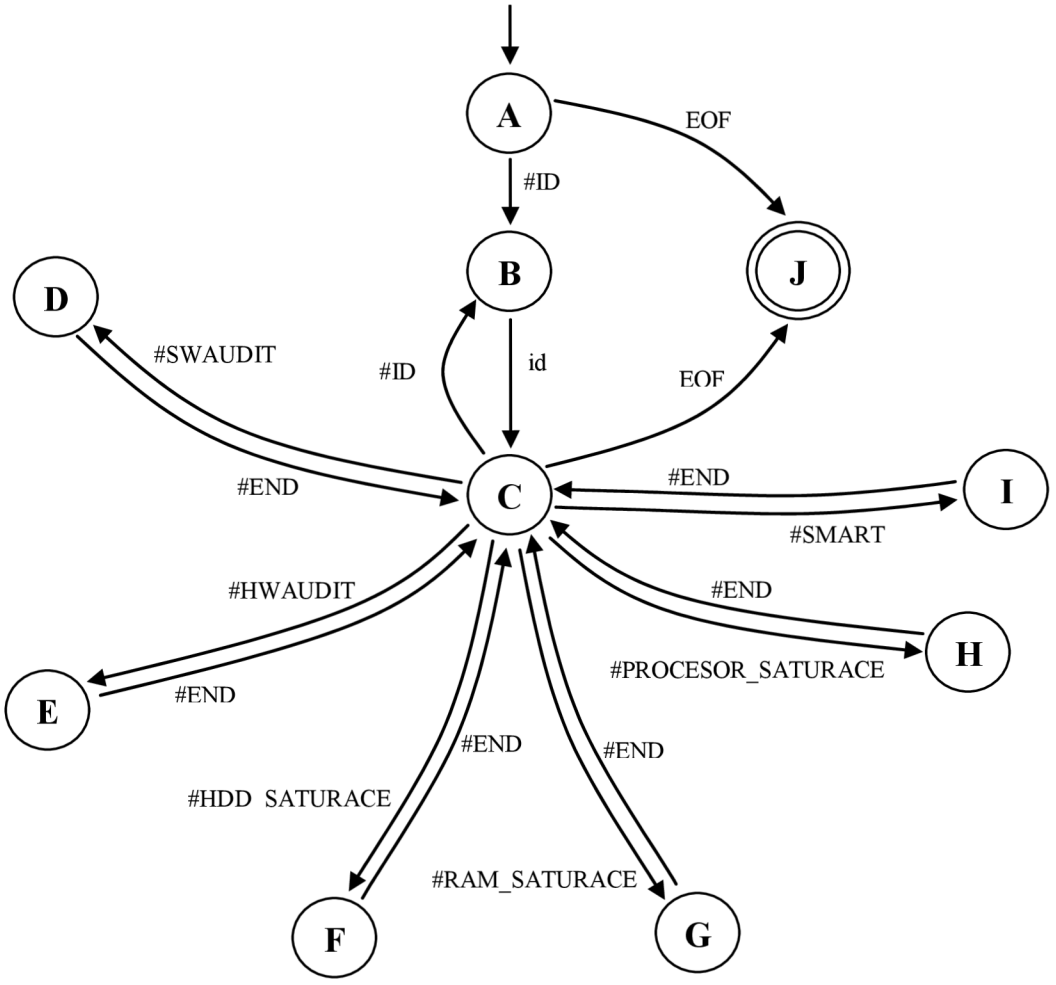
  - *#RER - RAW READ ERROR RATE.*
  - *#TPE – THROUGHPUT PERFORMANCE*
  - *#SUT – SPIN UP TIME*
  - *#SSC – START STOP COUNT*
  - *#RSC – RELLOCATED SECTOR COUNT*
  - *#RCM – READ CHANNEL MARGIN*
  - *#SER – SEEK ERROR RATE*
  - *#STP – SEEK TIME PERFORMANCE*
  - *#POHC – POWER ON HOURS COUNT*
  - *#SRC - SPIN RETRY COUNT*
  - *#CRC - CALIBRATION RETRY COUNT*
  - *#PCC - POWER CYCLE COUNT*

- *#SEER - SOFT READ ERROR RATE*
- *#TEM - TEMPERATURE*
- *#REC - RELLOCATION EVENT COUNT*
- *#CPS - CURRENT PENDING SECTOR COUNT*
- *#OFU - OFFLINE SCAN UNCORRECT*
- *#UAC - ULTRA ATA CRC ERROR COUNT*
- *#WER - WRITE ERROR RATE*

Význam všech těchto parametrů je vysvětlen v kapitole teoretický rozbor. Po každé z těchto značek následuje vektor, ve kterém jsou uložena data o příslušném atributu. Jako poslední značka je #END, která oznamuje konec S.M.A.R.T dat.

- ***Zpracování hlavičky (9)***
  Hlavička začíná atributem #ID. Po značce #ID následuje identifikační číslo, které je kontrolováno i číslem v databázi.

7.1.2 Konečný automat na zpracování souboru



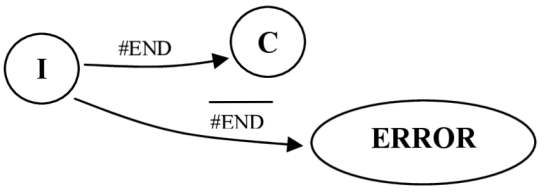
Obrázek 7.1.2.1
Konečný automat na zpracování datového souboru

Poznámky k obrázku 7.1.2.1:

EOF - značí konec souboru
A - stornovací stav
J - koncový stav

- STAVY D, E, F, G, H, I jsou rozpracovány na dalších diagramech

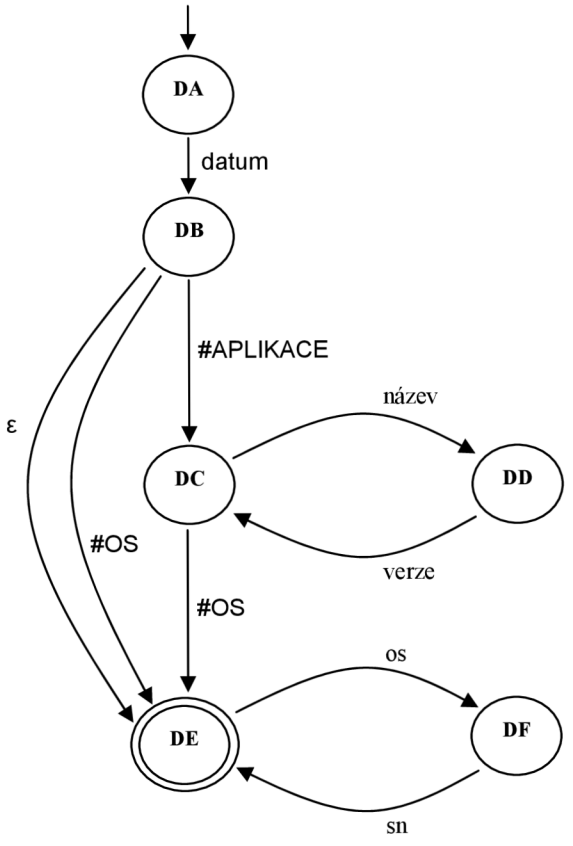
- Pokud konečný automat nemůže pokračovat a není v konečném stavu, je to bráno jako špatný formát datového souboru. Chybový stav zde není zobrazen kvůli přehlednosti. Nicméně platí například:



- V některých diagramech jsou použity **ε** hrany.

7.1.3 Konečný automat na zpracování SW auditu

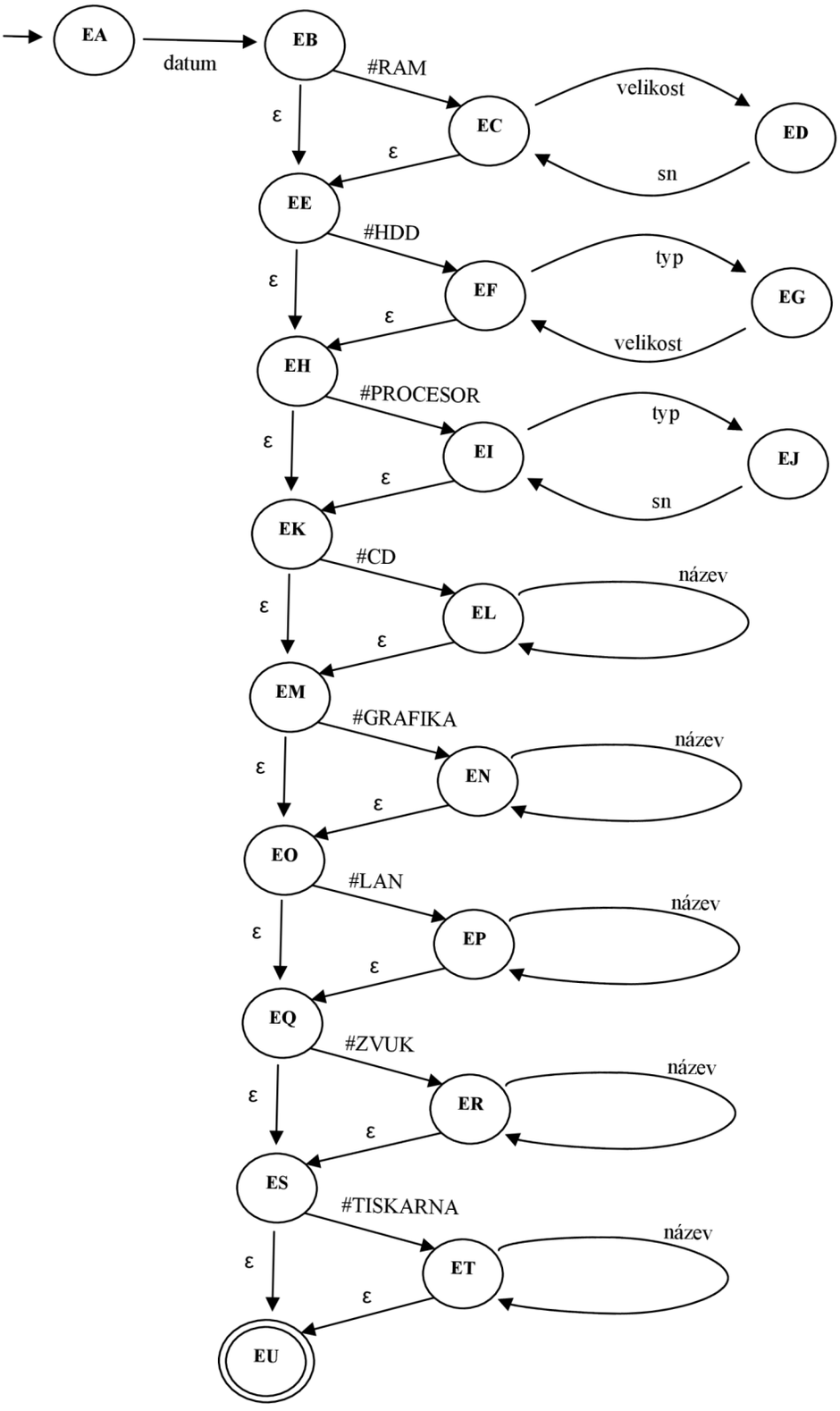
Rozpracování stavu D



Obrázek 7.1.3.1
Konečný automat na zpracování SW Auditu

7.1.4 Konečný automat na zpracování HW AUDITU

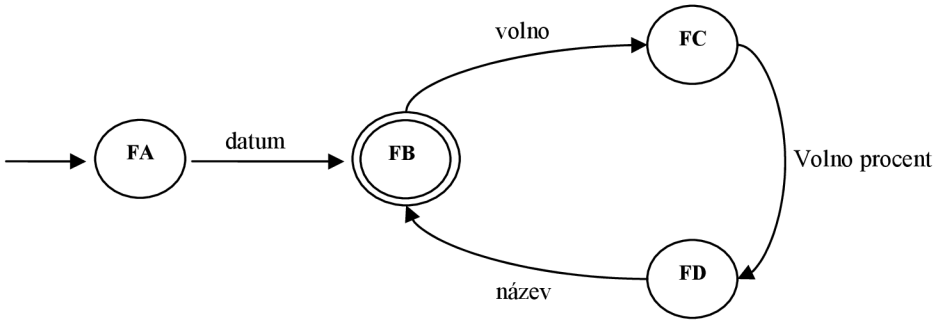
Rozpracování stavu E



Obrázek 7.1.4.1
Konečný automat na zpracování HW Auditu

7.1.5 Konečný automat na zpracování RAM Saturace

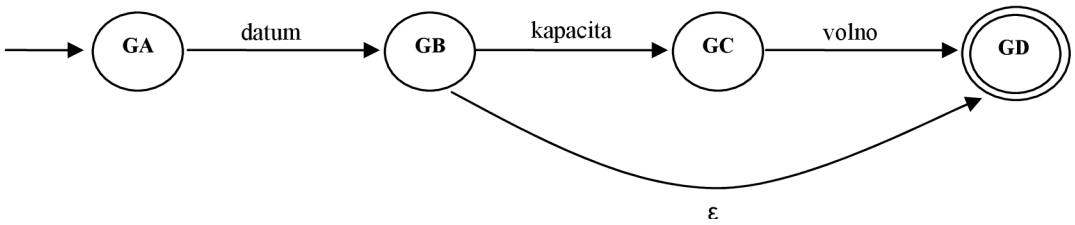
Rozpracování stavu F



Obrázek 7.1.5.1
Konečný automat na zpracování RAM Saturace

7.1.6 Konečný automat na zpracování HDD Saturace

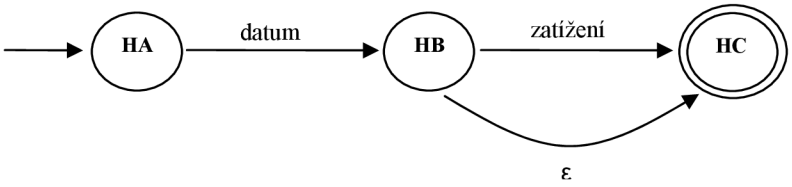
Rozpracování stavu G



Obrázek 7.1.6.1
Konečný automat na zpracování HDD Saturace

7.1.7 Konečný automat na zpracování PROCESOR Saturace

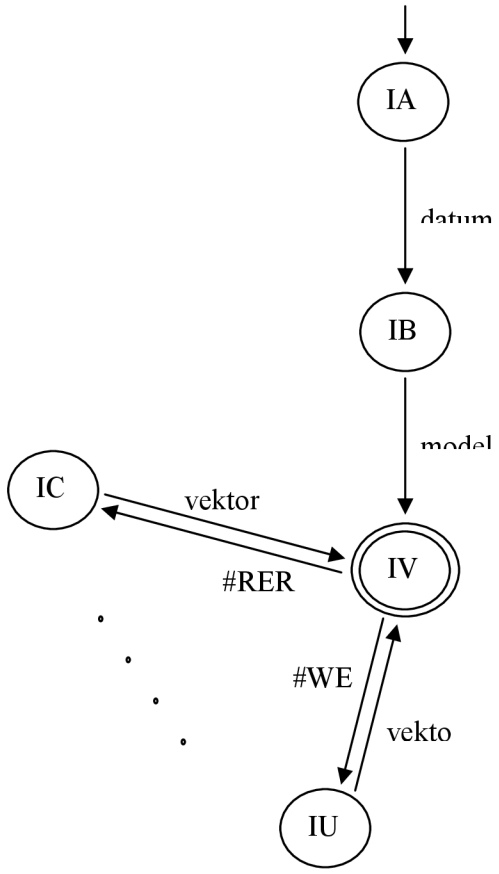
Rozpracování stavu H



Obrázek 7.1.6.1
Konečný automat na zpracování PROCESSOR Saturace

7.1.8 Konečný automat na zpracování S.M.A.R.T

Rozpracování stavu I



Obrázek 7.1.8.1
Konečný automat na zpracování S.M.A.R.T

Zde jsem zobrazil jenom parametry #RER a #WER, protože stejné schéma platí i pro ostatní parametry a diagram by byl nepřehledný.

7.2 Implementace parseru

Parser se skládá ze čtyř tříd, které mezi sebou komunikují. Jsou to třídy:

- ***CKernel***
  Tato třída provádí parsování a využívá třídy Cfile a CmySQL.

- ***CFile***
  Tato třída zajišťuje přístup k souborům na pevném disku. Již jsem jí popsal v kapitole implementace skeneru.

- ***CmySQL***
  Tato třída slouží jako rozhraní pro práci s databázovým systémem MySQL.

- ***CRijndael***
  Tato třída slouží pro šifrování a dešifrování. Její implementaci jsem již popsal v kapitole implementace skeneru.

7.2.1 Implementace třídy Ckernel

Rozhraní třídy je definováno v hlavičkovém souboru *kernel.h*. Rozhraní poskytuje celkem dvě metody:

```
unsigned int loadConfigFromFile(
     string fileName
);

unsigned int startKernel();
```

Metoda *loadConfigFromFile* slouží pro nahrání konfigurace z konfiguračního souboru. Má jediný vstupní parametr *fileName*, do kterého se předává název konfiguračního souboru. Název konfiguračního souboru je definován pomocí konstanty. Metoda vrací 0 v případě správného provedení, jinak vrací nezápornou hodnotu, která indikuje chybu a její druh.

Metoda *startKernel* slouží pro samotné parsování. Nemá vstupní parametr. Metoda vrací 0 v případe správného provedení, jinak vrací nezápornou hodnotu, která indikuje chybu a její druh.

Metoda loadConfigFromFile

```
unsigned int loadConfigFromFile(string fileName);
```

Metoda nejprve otevře konfigurační soubor. Pokud konfigurační soubor nejde otevřít, tak vrací příslušný kód chyby. Po otevření souboru se nahrávají konfigurační data. Pokud jsou všechny potřebné konfigurační parametry jádra načteny, tak se nastaví atribut třídy, který indikuje inicializaci jádra.

Metoda startKernel

```
unsigned int startKernel();
```

Metoda nejprve otestuje, zda je jádro inicializováno. Pokud není inicializováno, tak se vrací příslušný kód chyby. Tato situace by se dala řešit i tak, že pokud by nebylo jádro inicializováno, tak by spustila inicializace. To by se ale předával kód chyby z inicializace do metody startKernel a ta by musela tyto chybové kódy dále předávat. Zvolil jsem proto jednodušší variantu.

Metoda postupuje v práci podle vývojového diagramu, který jsem uvedl v kapitole vývojový diagram parseru. Pro práci s databází využívá třídu CmySQL.

7.2.2 Implementace třídy CmySQL

Před implementací této třídy jsem se snažil jít cestou nejmenšího odporu. Snažil jsem se najít již nějaké hotové řešení. Sice jsem nějaké třídy na práci s MySQL našel, ale žádná nevyhovovala mým požadavkům. Nakonec jsem tento problém vyřešil tak, že jsem si ze stránek www.mysql.com stáhnul kompletní instalaci MySQL. Tato instalace obsahuje i aplikační rozhraní pro komunikaci s databázovým serverem. Přesný odkaz na instalační balíček uvádím v literatuře [10]. Instalační balíček jsem umístil i na přiložené CD do adresáře Knihovny/MySQL. Při výběru instalačního

balíčku jsem dbal na jeho verzi. Chtěl jsem, aby verze instalačního balíčku byla shodná s verzí MySQL, která je nainstalována na webovém serveru společnosti The Best Network Solution.

Pomocí tohoto aplikačního rozhraní jsem vytvořil třídu, kterou využívám pro komunikaci s databází. Třída CmySQL poskytuje celkem čtyři metody:

```
unsigned int mysqlConnect(
      string severName,
      string user,
      string password,
      string databaseName,
      unsigned int port
);

bool mysqlCloseConnect();

bool mysqlSelect(
      vector<string> * data,
      string query
);

bool mysqlInsert(
      string query
);
```

Rozhraní třídy je definováno v hlavičkovém souboru *mySQL.h*.

mysqlConnect

```
unsigned int mysqlConnect(string severName,
      string user,
      string password,
      string databaseName,
      unsigned int port
);
```

Tato metoda slouží k navázání spojení s mySQL serverem. Jako vstupní parametry má název serveru, uživatelské jméno, uživatelské heslo, jméno databáze a port. Metoda využívá funkci aplikačního rozhraní MySQL pro připojení a vybrání databáze. Pokud se povede připojit k MySQL, tak metoda vrací 0 v opačném případě nezáporné celé číslo, které indikuje příslušný kód chyby.

mysqlCloseConnect

```
bool mysqlCloseConnect();
```

Tato metoda slouží k ukončení spojení s databází. Pokud je otevřené spojení, tak tato metoda ukončí spojení. V této metodě jsou využívány funkce, které poskuje aplikační rozhraní MySQL.

mysqlSelect

```
bool mysqlSelect(vector<string> * data,string query);
```

Tato metoda provádí dotazy typu SELECT. Jako vstupní parametry má ukazatel na vektor, který naplní příslušnými daty a queru, který specifikuje SQL dotaz. Metoda vrací 0 pokud proběhl dotaz v pořádku a 1 pokud databáze MySQL ohlásila chybu. V metodě jsou využívány funkce aplikačního rozhraní MySQL.

mysqlInsert

```
bool mysqlInsert(string query);
```

Tato metoda provádí dotazy typu INSERT. Jako vstupní parametr má řetězec, který specifikuje SQL. Metoda vrací 0 pokud proběhl dotaz typu INSERT v pořádku a 1 pokud databáze MySQL ohlásila chybu. V metodě jsou využívány funkce aplikačního rozhraní MySQL.

7.3 Informační systém

Informační systém jsem implementoval pomocí skriptovacího jazyka PHP. Jako datové úložiště jsem použil databázový systém MySQL.

7.3.1 Uživatelské role v informačním systému

V informačním systému jsem navrhl dvě uživatelské role. První role je administrátor a druhá role je technik. Role se liší v možnosti prováděných operací. Uživatelská role technik může provádět pouze podmnožinu uživatelských operací role administrátor. Technikům jsou přiřazovány firmy. Diagram použití je v následující kapitole.

Role technik může provádět tyto činnosti:

- Procházet počítače ve svých firmách
- Zobrazovat SW audity na počítačích, které jsou registrovány v jeho firmách
- Zobrazovat HW audity na počítačích, které jsou registrovány v jeho firmách.
- Zobrazovat diagnostiky na počítačích, které jsou registrovány v jeho firmách
- Mazat audity na počítačích, které jsou registrovány v jeho firmách
- Do svých firem může registrovat nové počítače
- Mazat počítače ve svých firmách
- Editovat počítače ve svých firmách
- Odhlásit se ze systému

Role administrátor může provádět tyto operace:

- Procházet počítače ve firmách všech firmách
- Zobrazovat SW audity na všech počítačích
- Zobrazovat HW audity na všech počítačích
- Zobrazovat diagnostiky na všech počítačích
- Mazat audity na všech počítačích
- Do všech firem může registrovat nové počítače
- Mazat počítače ve všech firmách
- Editovat počítače ve všech firmách
- Registrovat nové uživatele
- Editovat uživatele a jejich role
- Mazat uživatele
- Registrovat nové firmy
- Editovat firmy
- Mazat firmy
- Přiřazovat firmy technikům
- Odhlásit se ze systému

7.3.2 Diagram použití

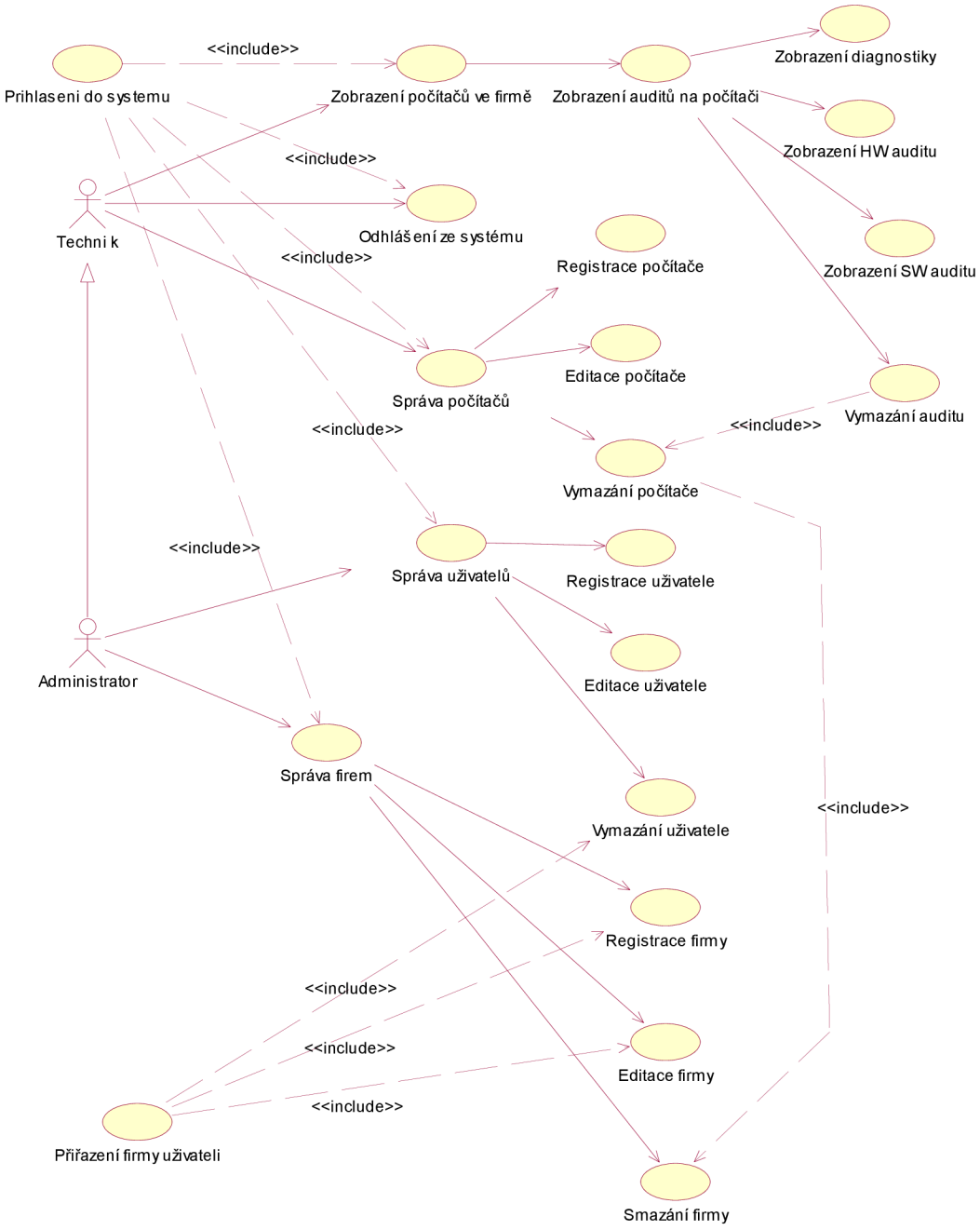


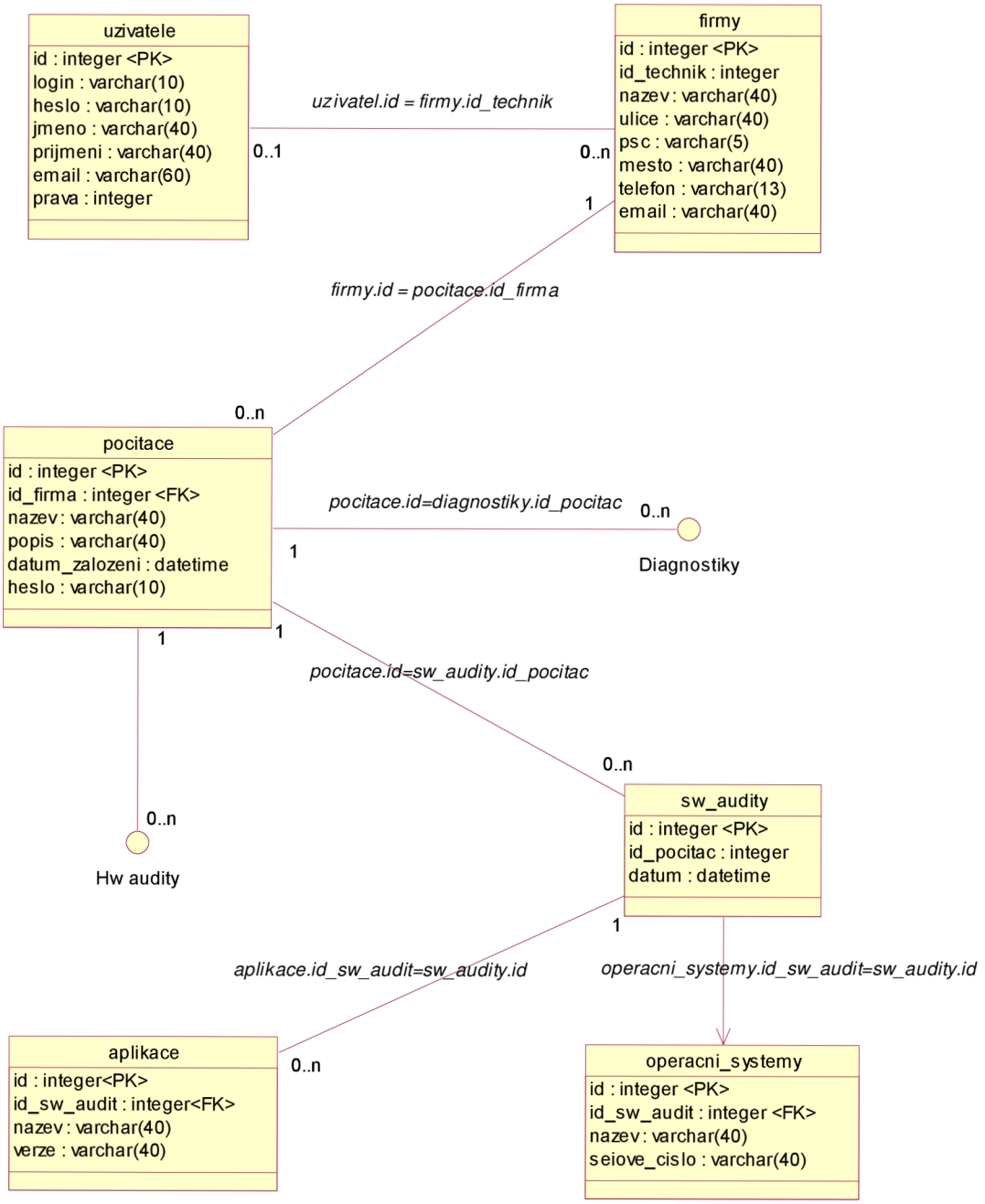
Diagram použití

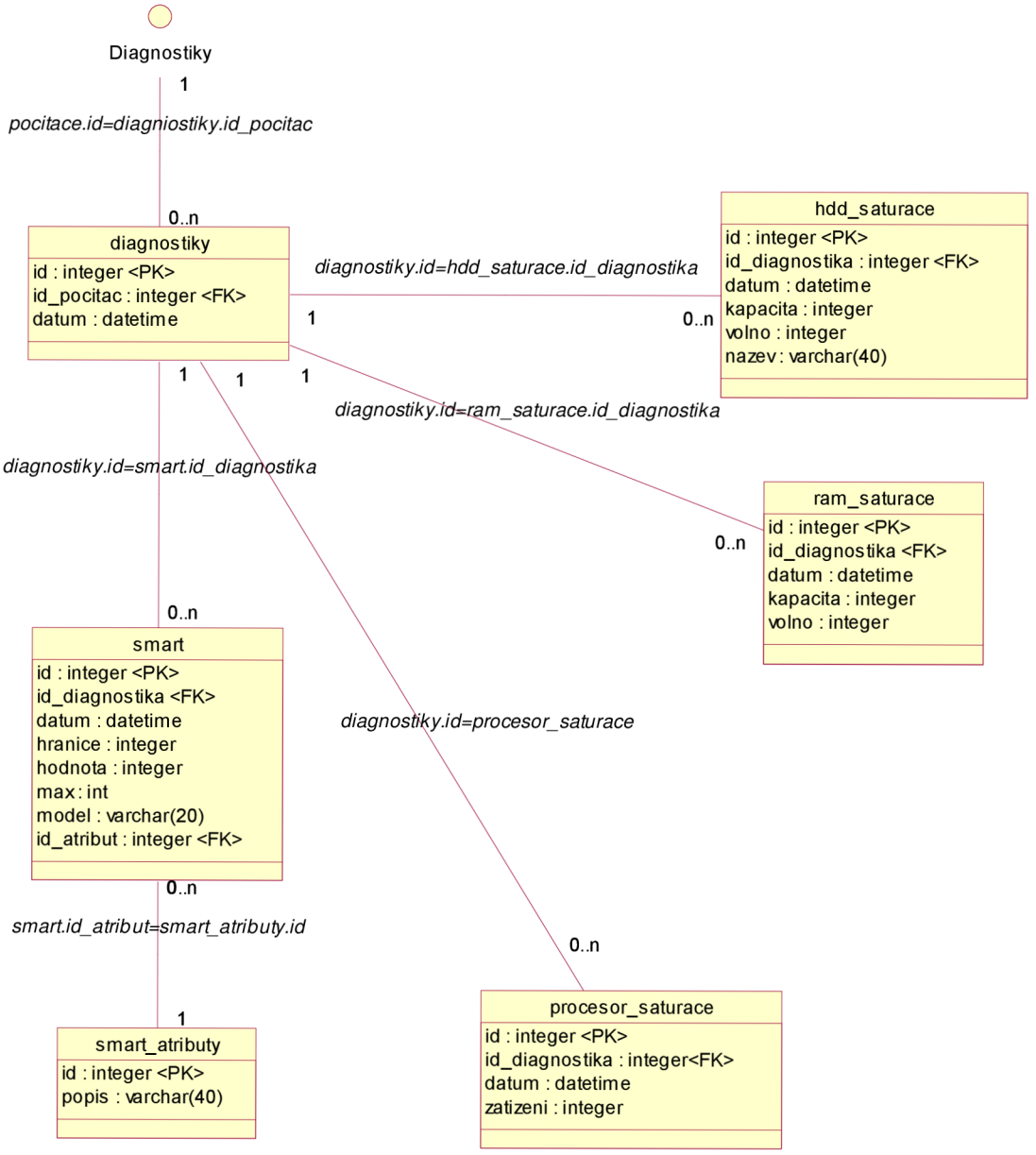
7.3.3 Návrh databázového schématu

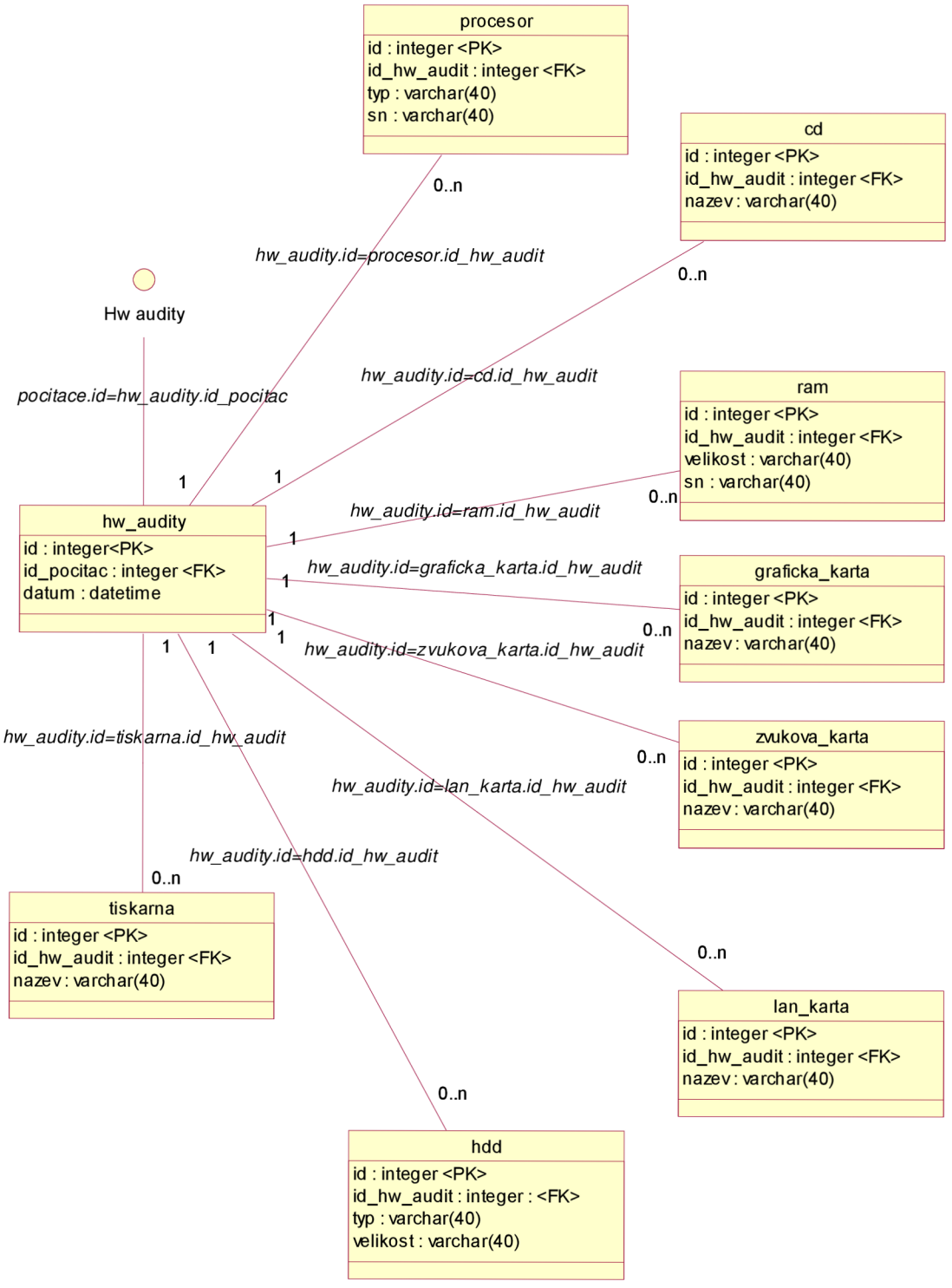
7.3.3.1 Relační schéma databáze

Při návrhu databázového schématu jsem se snažil, aby byly tabulky v první normální formě a aby v nich nedocházelo k redundanci a nekonzistenci dat. Protože schéma bylo moc veliké, tak jsem do

relačního schématu přidal dvě rozhraní, a to sice *Hw audity* a *Diagnostiky*. Tyto rozhraní zachycují relace mezi tabulkami, které jsou na různých stranách.




Diagnostiky
1
pocitace.id=diagniostiky.id_pocitac
0..n
diagnostiky
id : integer <PK>
id_pocitac : integer <FK>
datum : datetime
diagnostiky.id=hdd_saturace.id_diagnostika
1
0..n
hdd_saturace
id : integer <PK>
id_diagnostika : integer <FK>
datum : datetime
kapacita : integer
volno : integer
nazev : varchar(40)
1
1
1
diagnostiky.id=ram_saturace.id_diagnostika
diagnostiky.id=smart.id_diagnostika
0..n
ram_saturace
id : integer <PK>
id_diagnostika <FK>
datum : datetime
kapacita : integer
volno : integer
0..n
smart
id : integer <PK>
id_diagnostika <FK>
datum : datetime
hranice : integer
hodnota : integer
max : int
model : varchar(20)
id_atribut : integer <FK>
diagnostiky.id=procesor_saturace
0..n
smart.id_atribut=smart_atributy.id
0..n
1
smart_atributy
id : integer <PK>
popis : varchar(40)
procesor_saturace
id : integer <PK>
id_diagnostika : integer<FK>
datum : datetime
zatizeni : integer


procesor
id : integer <PK>
id_hw_audit : integer <FK>
typ : varchar(40)
sn : varchar(40)
cd
id : integer <PK>
id_hw_audit : integer <FK>
nazev : varchar(40)
Hw audity
pocitace.id=hw_audity.id_pocitac
hw_audity.id=procesor.id_hw_audit
hw_audity.id=cd.id_hw_audit
ram
id : integer <PK>
id_hw_audit : integer <FK>
velikost : varchar(40)
sn : varchar(40)
hw_audity
id : integer<PK>
id_pocitac : integer <FK>
datum : datetime
hw_audity.id=ram.id_hw_audit
hw_audity.id=graficka_karta.id_hw_audit
graficka_karta
id : integer <PK>
id_hw_audit : integer <FK>
nazev : varchar(40)
hw_audity.id=zvukova_karta.id_hw_audit
zvukova_karta
id : integer <PK>
id_hw_audit : integer <FK>
nazev : varchar(40)
hw_audity.id=tiskarna.id_hw_audit
hw_audity.id=lan_karta.id_hw_audit
hw_audity.id=hdd.id_hw_audit
tiskarna
id : integer <PK>
id_hw_audit : integer <FK>
nazev : varchar(40)
lan_karta
id : integer <PK>
id_hw_audit : integer <FK>
nazev : varchar(40)
hdd
id : integer <PK>
id_hw_audit : integer : <FK>
typ : varchar(40)
velikost : varchar(40)
0..n
1

7.3.3.2 Popis a význam tabulek

Aby byl jasný význam všech databázových tabulek, tak ke každé tabulce napíši její význam, případně zdůrazním některé sloupce, o kterých si myslím, že jsou důležité nebo jejich význam ne

- ***Uzivatele***
  Tabulka slouží k ukládání informací o uživatelích, respektive administrátorech a technicích. *Login* a *heslo* slouží pro autentizaci uživatele. *Jmeno* a *prijmeni* pro identifikaci uživatele v systému. *Email* slouží k zasílání emailových zpráv. *Prava* slouží rozlišení rolí v informačním systému.

- ***Firmy***
  Tabulka slouží k ukládání základních informací o firmě.

- ***Pocitace***
  Tabulka slouží k ukládání informací o konkrétním počítači. Každý počítat patří do nějaké firmy. Sloupec *heslo* slouží k ukládání hesel, podle kterých se šifrují datové soubory.

- ***Sw_audity***
  Tabulka souží k ukládání SW auditů prováděných na počítači. U každého auditu je zaznamenán datum a na jakém počítači byl audit proveden.

- ***Aplikace***
  Tabulka slouží k ukládání informací o nainstalovaných aplikacích na počítači. Tato tabulka se odkazuje do tabulky *sw_audity*.

- ***Operacni_systemy***
  Tabulka slouží k ukládání informací o nainstalovaných operačních systémech na počítači. Tato tabulka se odkazuje do tabulky *sw_audity*.

- ***Hw_audity***
  Tabulka slouží k ukládání HW auditů prováděných na počítači. U každého auditu je zaznamenán datum a na jakém počítači byl audit proveden.

- ***Cd***
  Tabulka slouží k ukládání informací o nainstalovaných CD/DVD mechanikách. Tabulka se odkazuje do tabulky *hw_audity*.

- ***Graficka_karta***
  Tabulka slouží k ukládání informací o nainstalovaných grafických kartách na počítači. Tabulka se odkazuje do tabulky *hw_audity*.

- ***Hdd***
  Tabulka slouží k ukládání informací o nainstalovaných pevných discích na počítači. Tabulka se odkazuje do tabulky *hw_audity*.

- ***Lan_karta***
  Tabulka slouží k ukládání informací o nainstalovaných síťových rozhraních na počítači. Tabulka se odkazuje do tabulky *hw_audity*.

- ***Procesor***
  Tabulka slouží k ukládání informací o procesorech v počítači. Tabulka se odkazuje do tabulky *hw_audity*.

- ***Ram***
  Tabulka slouží k ukládání informací o procesorech v počítači. Tabulka se odkazuje do tabulky *hw_audity*.

- ***Tiskarna***
  Tabulka slouží k ukládání informací o nainstalovaných tiskárnách v počítači. Tabulka se odkazuje do tabulky *hw_audity*.

- ***Zvuková karta***
  Tabulka slouží k ukládání informací o nainstalovaných zvukových zařízeních a multimediálních adapterech v počítači. Tabulka se odkazuje do tabulky *hw_audity*.

- ***Diagnostiky***
  Tabulka slouží k ukládání diagnostik prováděných na počítači. U každé diagnostiky je zaznamenán datum a na jakém počítači byla diagnostika provedena. Tabulka se odkazuje do tabulky *diagnostika*.

- ***Hdd_saturace***
  Tabulka slouží k ukládání informací o zaplnění pevných disků nainstalovaných v počítači. Ukládá se celková kapacita a volná kapacita disku. Tabulka se odkazuje do tabulky *diagnostika*.

- ***Procesor_saturace***
  Tabulka slouží k ukládání informací o vytíženosti procesorů nainstalovaných v počítači. V tabulce je ukládána hodnota zatížení v procentech. Tabulka se odkazuje do tabulky *diagnostika*.

- ***Ram_saturace***
  Tabulka slouží pro ukládání informací o vytíženosti operační paměti nainstalovaných v počítači. V tabulce je ukládána celková kapacita a dostupná kapacita operační paměti v daný okamžik. Tabulka se odkazuje do tabulky *diagnostika*.

- ***Smart***
  Tabulka slouží k ukládání S.M.A.R.T hodnot disku. Ukládá se hranice hodnoty, maximální hodnota a aktuální hodnota daného atributu. Ve sloupci *model* se ukládá na jakém disku bylo měřeno. Sloupec *id_atribut* odkazuje do tabulky *smart_atributy*, který dává informace o typu měřeného atributu. Tabulka se odkazuje do tabulky *diagnostika*.

- ***Smart_atributy***
  Tabulka slouží k ukládání informací o S.M.A.R.T atributech.

7.3.4 Bezpečnost informačního systému a udržování kontextu

Bezpečnost informačního systému je velice důležitou a neopomenutelnou záležitostí. Mnou navrhnutý informační systém obsahuje základní bezpečnostní prvky.

Jako první bezpečnostní prvek je používání šifrovaného protokolu https. Používání tohoto protokolu není dáno implementací systému, ale nastavením webového serveru.

Další bezpečnostní prvek je používání session ve skriptovacím jazyce PHP. Pokud uživatel úspěšně projde přihlašovací procedurou, tak se zaregistruje a nastaví proměnná typu session. Při nahrávání každé stránky je testováno zaregistrování příslušné proměnné session a její hodnota. Pokud je test negativní, tak je uživatel přesměrován na přihlašovací formulář.

Proměnná typu session slouží i k udržování kontextu mezi serverem a prohlížečem. Například pokud by byl uživatel delší dobu neaktivní, tak se zruší proměnná session a uživatel je automaticky odhlášen ze systému.

7.3.5 Návrh obrazovek informačního systému

7.3.5.1 Přihlašovací obrazovka informačního systému

K přihlášení do systému slouží jednoduchý přihlašovací formulář. Po zadání loginu a hesla je uživatel buď přihlášen do systému nebo je vyzván k opětovnému zadání loginu a hesla. Pokud login a heslo neodpovídá žádnému registrovanému uživateli v systému, je uživateli sděleno hlášení, že zadal nesprávné přihlašovací údaje.

Obrázek 7.3.5.1.1
Přihlašovací obrazovka informačního systému

7.3.5.2 Základní obrazovka informačního systému

Základní obrazovku informačního systému jsem navrhl tak, aby ovládání bylo jednoduché a přímé.

Obrázek 7.3.5.1.2
Přihlašovací obrazovka informačního systému

Hlavním ovládacím prvkem je horizontální menu. Z menu jsou dostupné všechny důležité funkce informačního systému.

Vpravo nahoře zobrazuje informační systém jméno přihlášené osoby a její roli v informačním systému. Pod menu je hlavní zobrazovací plocha. Vlevo nahoře v této ploše je vždy kategorie, ve které se zrovna uživatel pohybuje. Pod tímto nadpisem kategorie jsou zobrazovány stránky a ovládací prvky příslušných kategorií.

7.3.6 Implementace informačního systému

K implementaci jsem používal skriptovací jazyk PHP verze 4, který je dostupný na webovém serveru společnosti The Best Network Solution. Skriptovací jazyk PHP lze stáhnout na stránkách www.php.net.

Při implementaci informačního systému jsem nepoužíval objektový přístup. Programové kódy byly relativně jednoduché a neviděl jsem žádnou výhodu v objektově orientovaném přístupu. Znovupoužitelné fragmenty kódu jsem řešil funkcemi.

8 Závěr

Závěrem bych chtěl shrnout tento celý projekt. Napsat jeho komplexní zhodnocení, poznámky k implementaci a testování a zhodnotit jeho možné pokračování a získané zkušenosti.

8.1 Poznámky k implementaci

K implementaci jsem využíval Microsoft Visual Studio Professional Edition Version 8.0.50727.42, které mi poskytla společnost The Best Network Solution.

Při implementaci jsem se nevyhnul všem varováním od překladače. Některé varování byly způsobeny použitím jistých funkcí, které mohou způsobit havárii programu. Například funkce itoa, která provádí převod čísla na řetězec. U této funkce je problém s bufferem, který se jí předává. Pokud by buffer nebyl dostatečný, může dojít ke kolapsu programu. Tato funkce by šla nahradit novější funkcí, která si předává ještě jeden parametr navíc, který udává velikost bufferu. Tato funkce však nezaručuje zpětnou kompatibilitu. Toto je jeden z příkladů, na kterém jsem chtěl ukázat, že některá varování nebylo možné odstranit. Několik varování vychází i z použití knihoven, které jsou v projektu využívány.

Při implementování jsem často používal knihu Mistrovství v C++, přesnou referenci uvádím v literatuře [28]. Tato kniha mi poskytla mnoho praktických rad a informací.

8.2 Testování

Testování bylo velice obtížné. Zaprvé kvůli tomu, že tato aplikace je typu klient server a za druhé, protože aplikace nepoužívá žádné okno. Abych mohl testovat jednotlivé části systému, tak jsem se při implementaci snažil postupovat systematicky. Například nejprve jsem zprovoznil komunikaci mezi skenerem a IS bez šifrování a poté jsem teprve šifrování dodělal. Tento systematický přístup mi umožnil testování celé aplikace po částech. Ze začátku fungoval skener jako konzolová aplikace. Když byly všechny implementace téměř hotové, tak jsem skener předělal na okenní aplikaci. Protože toto okno nesmí být vidět, kvůli zvýšení bezpečnosti, tak jsem ho ve zdrojových kódech nenechal zobrazit. Tím je docíleno, že aplikace není normálním uživatelům viditelná.

K testování jsem využíval dva počítače. První počítač sloužil jako server a druhý jako sledované PC.

8.3 Pokračování projektu

Když jsem projekt začínal řešit, tak jsem studoval informace o podobných diagnostických programech. Většina těchto programů byla řešena nějakým vývojovým týmem a tyto programy měly zpravidla několika roční vývojový životní cyklus. Vzhledem k tomu, že je tento projekt specifický a navrhnutý pro potřeby společnost The Best Network Solution, tak porovnání s jinými aplikacemi podobného druhu je obtížné. Nenašel jsem žádný program podobný tomuto. Mnou navržený program provádí nejenom skenování parametrů počítače, ale řeší i jejich transport a zpracování v centrální databázi.

Vzhledem k tomu, že je tento projekt v celku rozsáhlý, tak se nabízí řada vylepšení, na kterých by mohl projekt pokračovat.

Mohou to být:

- Základní rozšíření aplikace vidím v automatizovaném informování příslušných techniků na vzniklé nestandardní stavy. Původně jsem tuto funkci chtěl implementovat do tohoto systému. Protože jsem strávil hodně času na implementaci technologie S.M.A.R.T, tuto funkci jsem již nestihl implementovat. Nicméně databázový návrh je pro tuto funkci připraven. Tento problém by šel řešit několika variantami. První varianta by byla v kontrolování dat při parsování. Tato varianta však není moc perspektivní, neumožňuje vytvořit zpětnou vazbu mezi technikem a problémem. Druhá varianta spočívá ve vytvoření skriptu, který by se spouštěl v pravidelných intervalech. Tento skript by analyzoval data. Pokud by se vyskytla chyba, technik by byl o chybě informován emailem. Musel by se přihlásit do IS a odsouhlasit, že o dané chybě ví. Tím by se na něho přesunula zodpovědnost vyřešení problému. Pokud by tak neučinil IS by posílal zprávy opakovaně. Pokud by technik nereagoval ani po urgencích, byl by informován administrátor systému.

- Automatizované generování reportů.

- Komponenta na zálohování dat v informačním systému.

- Rozšíření aplikace o více-vláknový režim. Využitý více vláken pro skenování jednotlivých parametrů je zde přímo nabízející. Tento program je řešen jako sekvenční cyklus a tudíž v něm dochází k časovým ztrátám. Nicméně tyto ztráty nejsou tak markantní, aby způsobovaly veliké časové ztráty.

- Rozšíření sortimentu sledovaných parametrů.

- Vytvoření univerzálního rozhraní pro sledování nových parametrů. Program by mohl disponovat inteligentním rozhraním, které by zajišťovalo sledování parametrů na základě WQL dotazů. Podobné rozhraní má program Samurize. Do tohoto rozhraní se vloží WQL dotaz a jeho popis a je možné sledovat daný parametr. Vytvořením takového universálního rozhraní bylo nelehké, ale ne neřešitelné. Musel by se přizpůsobit jak parser, tak databáze na straně serveru.

- V neposlední řade by se mohlo vylepšit uživatelské rozhraní instalátoru případně odinstalátoru.

8.4 Zhodnocení a výsledky

Podařilo se mi navrhnout a naimplementovat systém na sledování vybraných parametrů počítače a na jejich zpracování v centrální databázi. V projektu jsem byl nucen sloučit více technologií dohromady a zjistil jsem, že to není vůbec jednoduché. Při řešení projektu jsem nahlédnul do problematiky programování pod operačním systémem Microsoft Windows. Zejména používání API funkcí operačního systému. Velmi důležitou částí projektu bylo testování. Naučil jsem se, jak systematicky testovat jednotlivé části projektu, které spolu kooperují. Tento projekt byl pro mě z hlediska programování a návrhu architektury programu velice přínosný.

Literatura

[1] Helmich Jiří, www.pctuning.cz, Pevné disky – principy a technologie,19.04.2006 [online]. [cit. 21.duben 2007]
Dostupné na WWW:
<http://www.pctuning.cz/index.php?option=com_content&task=view&id=6815&Itemid=45&limit=1&limitstart=3>

[2] Ed Wilson, Microsoft Windows Scripting with WMI, Microsoft Press 2006, Redmond, Washington 98052-6399

[3] Scriptomatic Version 2.0, [online]. © Microsoft Corporation 2006
Dostupné na WWW:
<http://www.microsoft.com/technet/scriptcenter/tools/scripto2.mspx>

[4] Jakub Mach, PHP pro úplné začátečníky. Computer Press, Brno 2003. ISBN 80-7226-834-1.

[5] Luke Welling, Laura Thomson, MySQL Průvodce základy databázového systému. CP Books, Brno 2005. ISBN 80-251-0671-3.

[6] Jakub Mach, PHP pro úplné začátečníky. Computer Press, Brno 2003. ISBN 80-7226-834-1.

[7] WMI Core 1.5, [online]. © Microsoft Corporation 2000
Dostupné na WWW:
<http://www.microsoft.com/downloads/details.aspx?FamilyID=98a4c5ba-337b-4e92-8c18-a63847760ea5&DisplayLang=en>

[8] Luboš Dobda, Ochrana dat v informačních systémech. Grada 1998. ISBN 80-7169-479-7.

[9] Vlastimil Klíma, časopis Chip, Představujeme kandidáty na AES. Listopad 1999, str.64-65.

[10] Instalační balíček MySQL verze 4.1 [online]. © 1995-2007 MySQL AB.
Dostupné na WWW:
<http://dev.mysql.com/downloads/mysql/4.1.html>

[11] Microsoft [online]. © Microsoft Corporation 2007
[cit. 16.duben 2007]
Dostupné na WWW:
<http://msdn2.microsoft.com/en-us/library/aa394558.aspx>

[12] Microsoft [online]. © Microsoft Corporation 2007
[cit. 16.duben 2007]
Dostupné na WWW:
<http://msdn2.microsoft.com/en-us/library/aa391769.aspx>

[13] Microsoft [online]. © Microsoft Corporation 2007
[cit. 16.duben 2007]
Dostupné na WWW:
<http://msdn2.microsoft.com/en-us/library/aa392770.aspx>

[14] Self-Monitoring Analysis and Reporting Technology (SMART), Pavel Procházka
[cit. 20.duben 2007]
Dostupné na WWW:
<http://prochazka.d2.cz/index.php?clanek=download >

[15] Microsoft [online]. © Microsoft Corporation 2007
[cit. 20.duben 2007]
Dostupné na WWW:
< http://msdn2.microsoft.com/en-us/library/aa363858.aspx>

[16] Microsoft [online]. © Microsoft Corporation 2007
[cit. 26.duben 2007]
Dostupné na WWW:
<http://msdn2.microsoft.com/en-us/library/aa363216.aspx>

[17] A C++ Implementation of the Rijndael Encryption/Decryption method, George Anescu, 19.září 2001
[cit. 2.květen 2007]
Dostupné na WWW:
<http://www.codeproject.com/cpp/aes.asp>

[18] Herb Sutter, Andrei Alexandrescu, C++ 101 programovacích technik, Zoner Press Brno 2005, ISBN 80-86815-28-5.

[19] Microsoft [online]. © Microsoft Corporation 2007
[cit. 26.duben 2007]
Dostupné na WWW:
<http://msdn2.microsoft.com/en-us/library/1f4c8f33(VS.71).aspx>

[20] Microsoft [online]. © Microsoft Corporation 2007
[cit. 26.duben 2007]
Dostupné na WWW:
<http://msdn2.microsoft.com/en-us/library/ms644943.aspx>

[21] Microsoft [online]. © Microsoft Corporation 2007
[cit. 26.duben 2007]
<http://msdn2.microsoft.com/en-us/library/aa385473.aspx>

[22] Microsoft [online]. © Microsoft Corporation 2007
[cit. 26.duben 2007]
Dostupné na WWW:
http://msdn2.microsoft.com/en-us/library/aa384702.aspx>

[23] Microsoft [online]. © Microsoft Corporation 2007
[cit. 26.duben 2007]
Dostupné na WWW:<http://msdn2.microsoft.com/en-us/library/aa385096.aspx>

[24] Microsoft [online]. © Microsoft Corporation 2007
[cit. 26.duben 2007]
Dostupné na WWW:<http://msdn2.microsoft.com/en-us/library/aa923721.aspx>

[25] Microsoft [online]. © Microsoft Corporation 2007
[cit. 26.duben 2007]
Dostupné na WWW:<http://msdn2.microsoft.com/en-us/library/aa384170.aspx>

[26] Stuart Konen, Registry Wrapper Class.
[cit. 30.duben 2007]
Dostupné na WWW:< http://www.codeproject.com/system/CRegistry.asp>

[27] David Majda, Spuštění programu po startu Windows, 30.03.2001
[cit. 30.duben 2007]
Dostupné na WWW:< http://www.codeproject.com/system/CRegistry.asp>

[28] Stehen Prata, Mistrovství v C++, Computer Press 2004, ISBN – 80-251- 0098-7

[29] Joe O'Leary, knihovna CString, 07.06.2003
[cit. 30.duben 2007]
Dostupné na WWW:< http://www.joeo.net/code/StdString.zip >

Seznam příloh

Příloha A. Instalační manuál. Uložen na přiloženém CD v adresáři Manuály.

Příloha B. CD

Příloha C. Zdrojové texty. Uloženy na CD v adresáři Zdrojové texty

Příloha D. Zkompilované programy. Uloženy na CD v adresáři Programy

Příloha A. Instalační manuál

1 Instalace Informačního systému

Instalace systému je pro zkušeného administrátora. Instalace klienta již tak hluboké znalosti nevyžaduje. Systém je navrhnut tak, aby instalace klienta byla snadná a zvládli jí technici, kteří nemají tak hluboké znalosti s administrací systému.

Instalace předpokládá nainstalované PHP a MySQL. PHP verze 4.0 a vyšší a MySQL verze 4.0. U PHP se přepokládá funkční komunikace s databázovým serverem MySQL. Dále se předpokládá funkční FTP server na který se budou posílat data a možnost použití protokolu HTTPS. Protokol HTTPS je otázkou nastavení webového serveru.

Instalace databáze MySQL

Ve složce „Zdrojové kódy\Serverová část systému\Informační systém\SQL“ na přiloženém CD jsou zdrojové kódy pro vytvoření databáze. Import databáze je možné provést přes příkazová řádek nebo pomocí nějakého dostupného grafického rozhraní, například webové rozhraní PhpMyAdmin.

Po nainstalování databáze je v databázi vytvořen jeden uživatel, který má roli administrátor. Jeho přihlašovací údaje jsou **admin/admin**. Databáze, která je vytvořena se jmenuje audit.

K databázi audit je nutné **vytvořit uživatelský účet**, pře který přistupuje k databázi z webových stránek a parseru.

Instalace zdrojových kódů

Ve složce „Zdrojové kódy\Serverová část systému\Informační systém\WWW“ na přiloženém CD jsou zdrojové kódy webových stránek informačního systému. Tyto soubory se nahrají do adresáře na webovém serveru. V souboru **setup.php** je potřebné provést nastavení pro přístup k databázi a informace o FTP serveru.

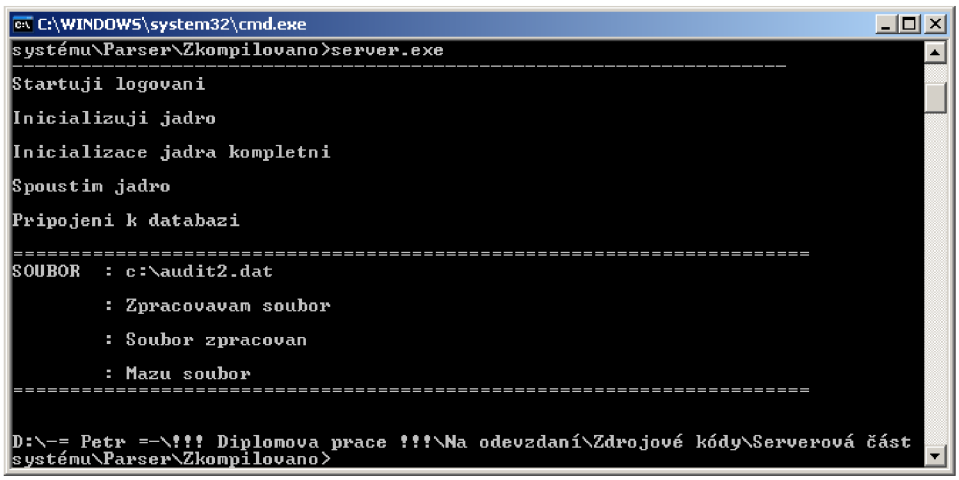
Nad adresářem **tmp** je potřeba nastavit potřebná práva, aby PHP mohlo v tom to adresáři pracovat se soubory. Vytvářet soubory, mazat soubory a editovat soubory.

Instalace parseru

Ve složce „"Zdrojové kódy\Serverová část systému\Parser\ "“ na přiloženém CD je umístěn parser. U tohoto parseru se musí nastavit konfigurační soubor. Jeho přesná struktura je popsaná v souboru ctime.txt u parseru. U parseru se musí dále nastavit jeho pravidelné spouštění. Dobu spouštění doporučuji cca 1 za 30 minut. Spouštění toho programu se musí nastavit v operačním systému. Windows server 2003 poskytuje nástroj na spouštění aplikací v danou dobu. Tento nástroj umožňuje i přidělení prostředků operačního systému.

V tento okamžik měla být celá serverová část kompletně nainstalovaná.

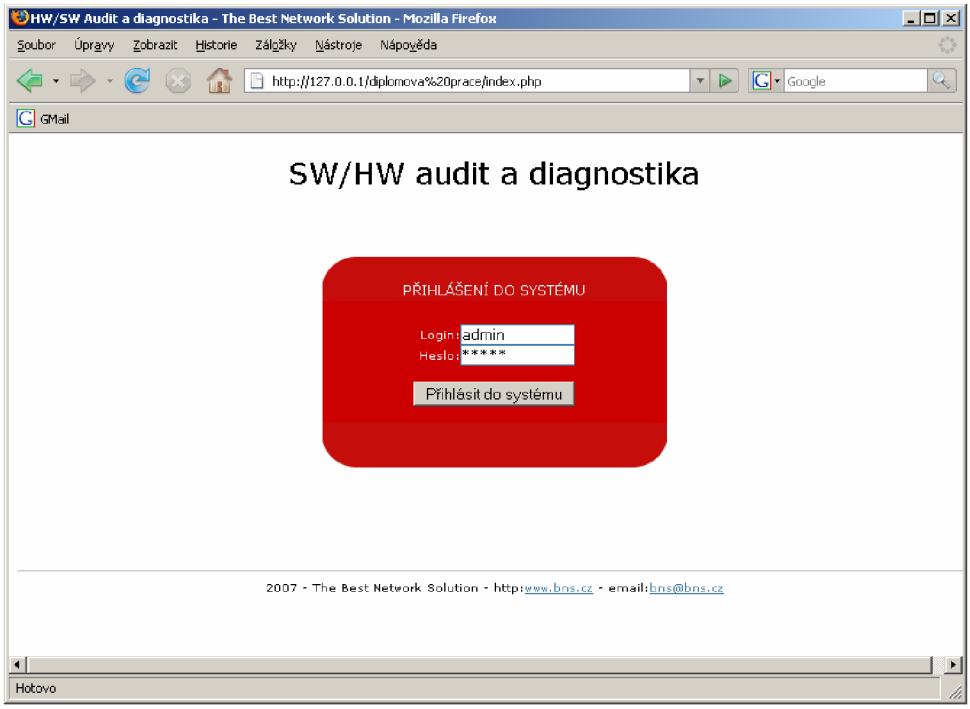
Tato obrazovka ukazuje spuštění parseru, který na FTP prostoru nalez datový soubor audit2.dat, provedl jeho zpracování a pak ho vymazal.



Parser si vytvoří seznam všech souborů na FTP prostoru a provádí jejich zpracování. Soubory, které neodpovídají danému formátu buď ignoruje nebo je smaže. Pokud má soubor jiný název než tato maska audit*.dat, tak se soubor ignoruje. Pokud není nalez žádný soubor na zpracování, tak se parser ukončí.

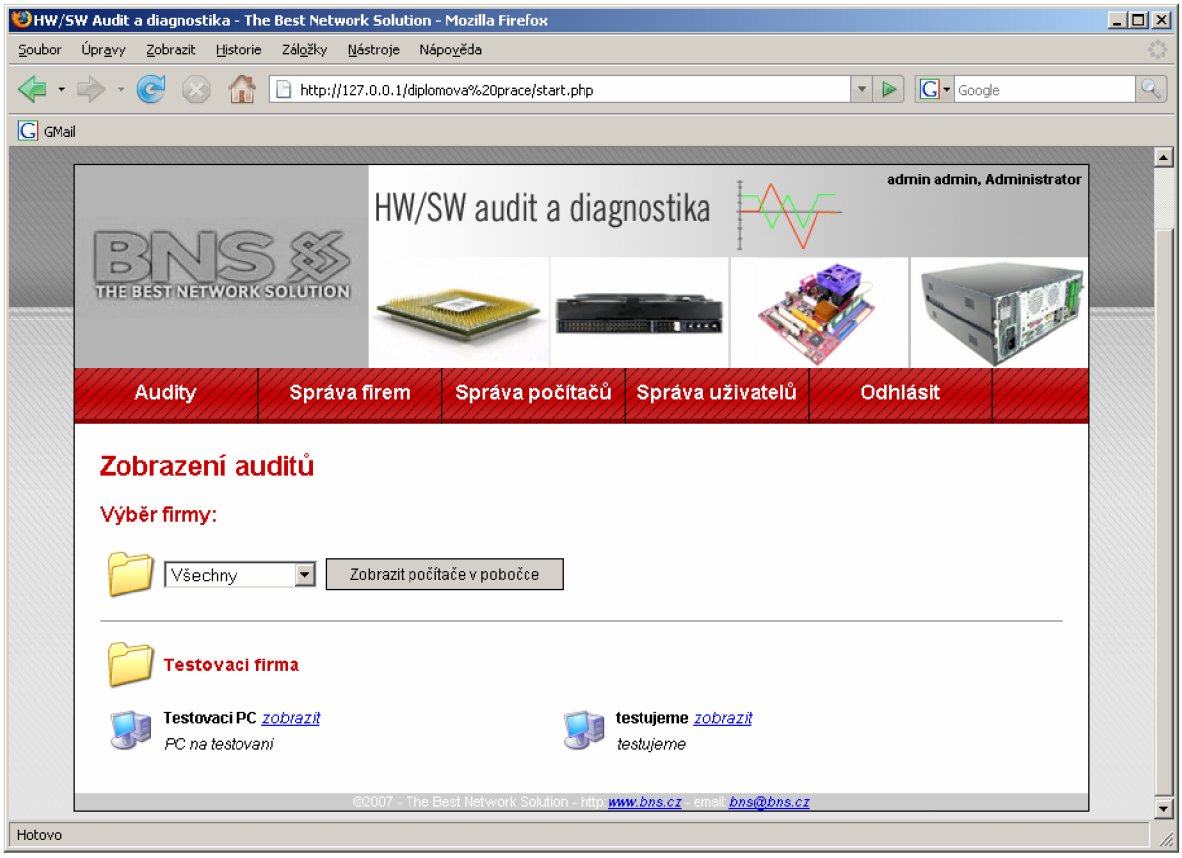
1.1 Přihlášení do systému

Přihlášení do systému se provádí pomocí stránky index.php. Je vhodné tuto stránku nastavit na webovém serveru jako výchozí.



Pokud proběhla instalace v pořádku, tak je možné se přihlásit pod uživatelským jménem a heslem admin/admin.

1.2 Menu

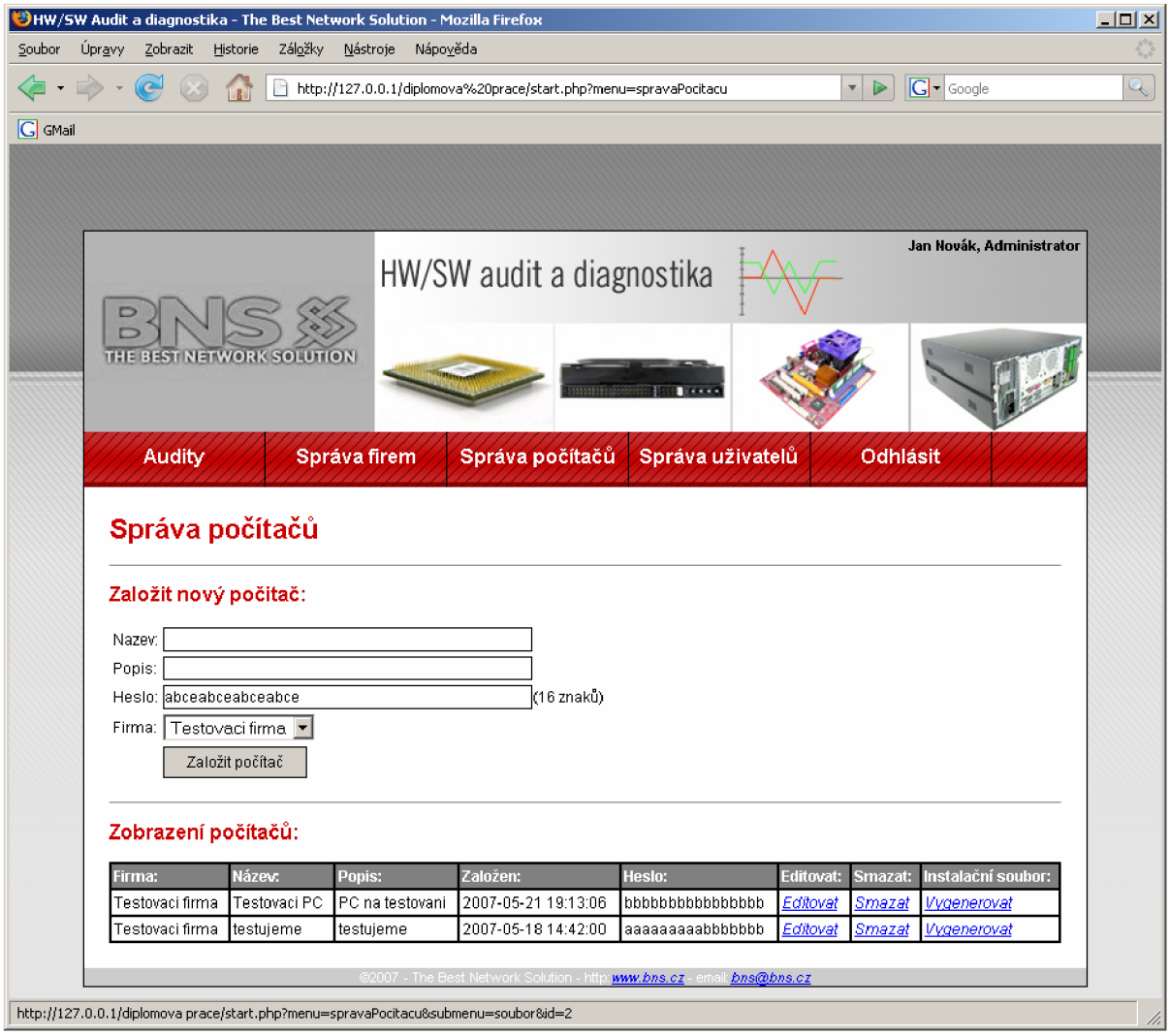


Základní menu programu poskytuje tyto volby:

- Audity – procházení auditu.
- Správu firem – spravování firem v systému.
- Správu počítačů – spravování počítačů.
- Správu uživatelů – spravování uživatelů.
- Odhlásit – odhlášení ze systému.

1.3 Registrace nové firmy a počítače

Nová firma se registruje v sekci správa firem. Po registraci firmy je možné zaregistrovat počítač. Počítače se registrují v sekci správa počítačů. Po zaregistrování počítače je možné vygenerova instalační soubor.

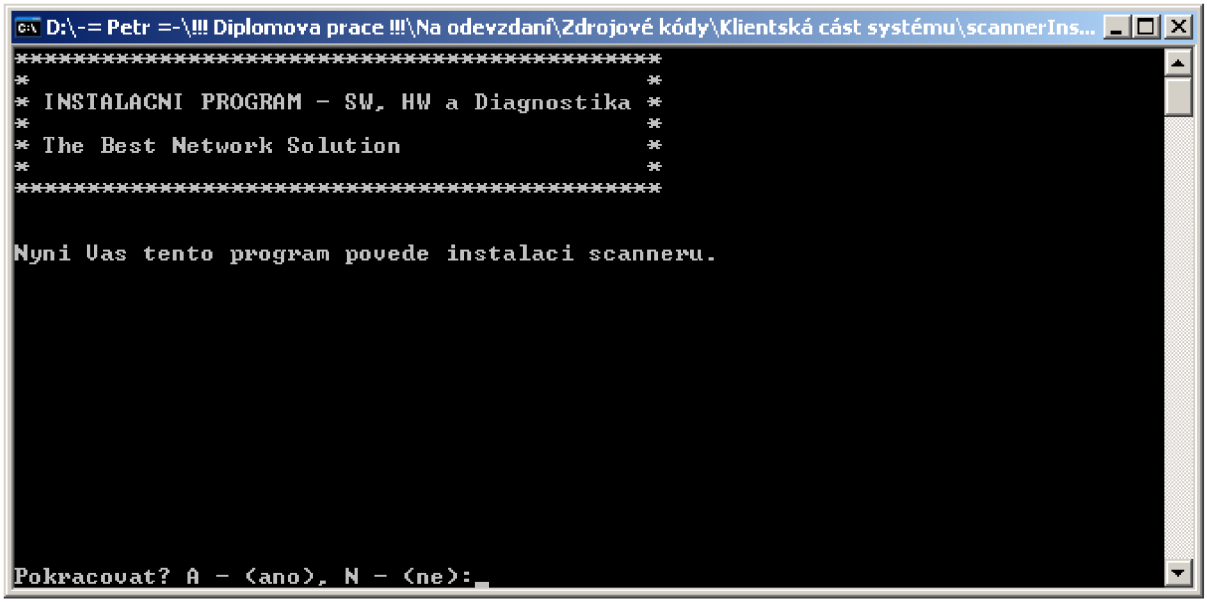


Tento instalační soubor slouží k nastavení lokálního klienta. Každý počítač může mít jiné heslo.

1.4 Nastavení klienta

Instalační soubor, který lze stáhnout z informačního systému je nutné nahrát do adresáře skeneru. Skener lze najít v adresáři Zdrojové kódy\Klientská část systému\ na přiloženém CD. Po vložení souboru do adresáře je nutné spustit soubor scannerInstall.exe, který provede nainstalovaní klienta. Tento instalační program je vytvořen jako průvodce, pomocí kterého je možné skener nastavit do požadovaného módu.

Instalační program

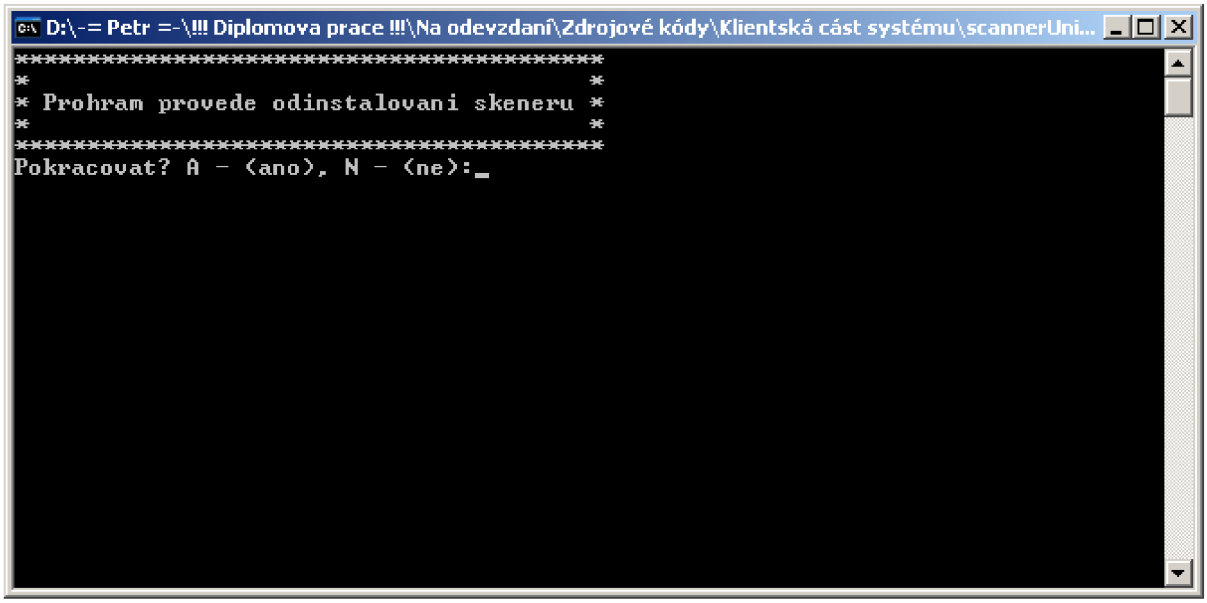


Po nainstalování klienta je nutné restartovat počítač. Klient naběhne automaticky se startem operačního systému. Po instalaci klienta je instalační soubor smazán.

Pro správnou komunikaci klienta a serveru je důležité **nastavit firewall**. V operačním systému Windows se toto provádí v sekci síťová připojení. Programu scannerClient.exe musí být dovolena internetová komunikace.

Odinstalační program

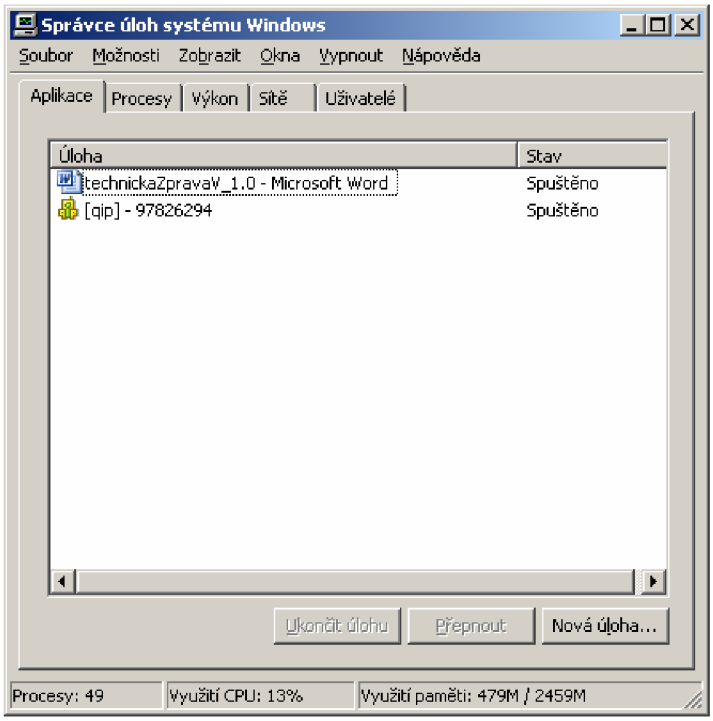
Pro odinstalování slouží soubor scannerUnstall.exe



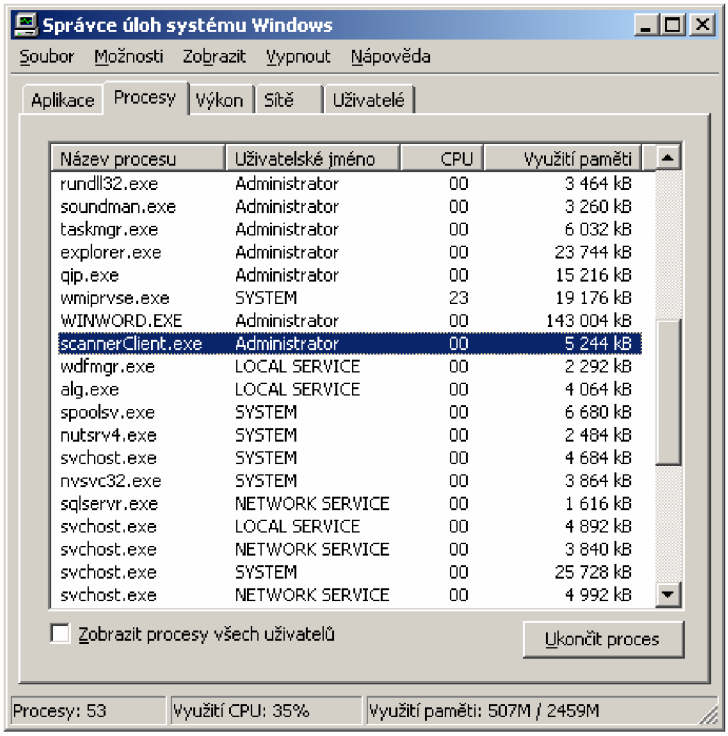
1.5 Viditelnost aplikace

Skener – Procesy a Aplikace

Mezi aplikacemi scanner není viditelný

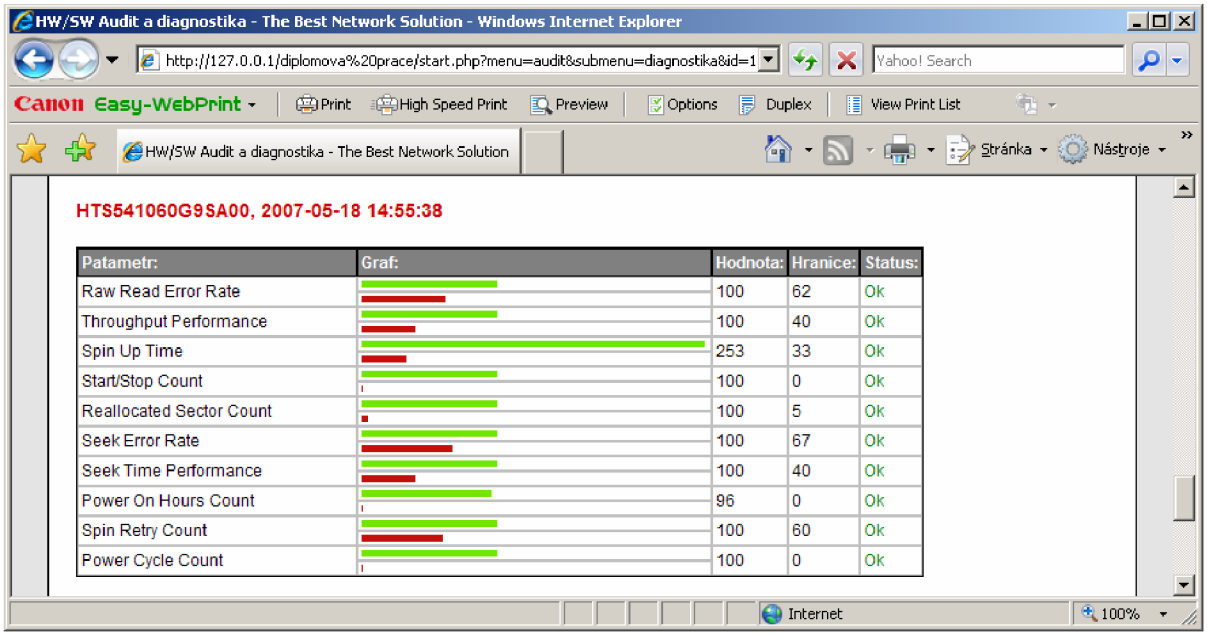


Mezi procesy ano



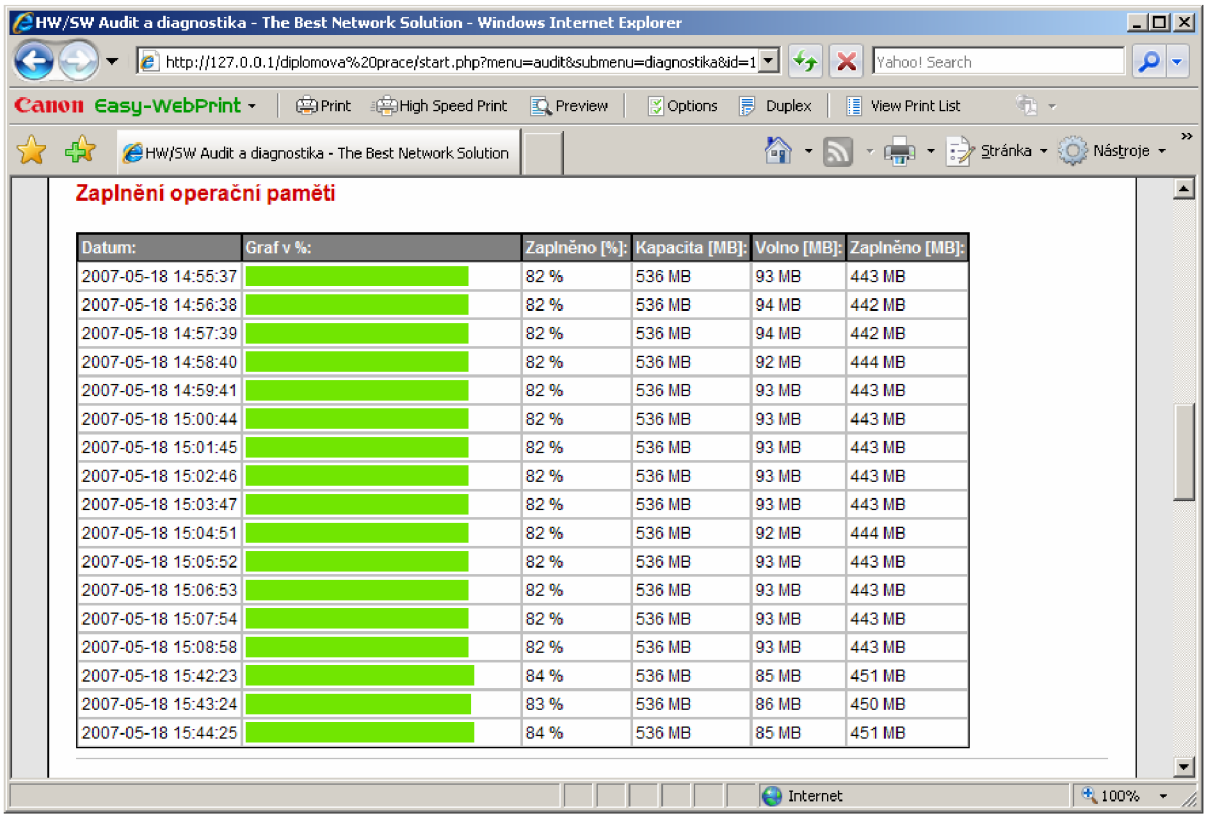
1.6 Ukázky grafů

Informační systém – Zobrazení S.M.A.R.T atributů



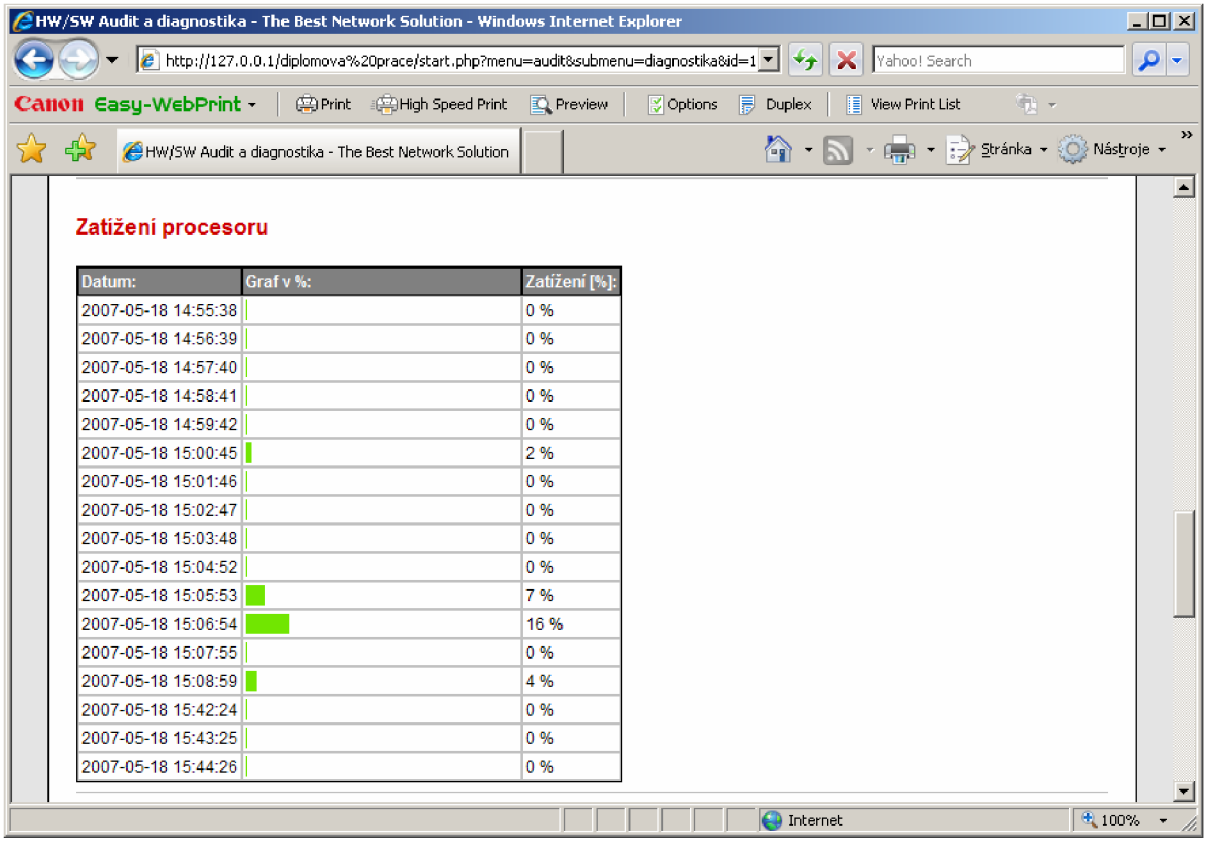
Patametr:	Graf:	Hodnota:	Hranice:	Status:
Raw Read Error Rate		100	62	Ok
Throughput Performance		100	40	Ok
Spin Up Time		253	33	Ok
Start/Stop Count		100	0	Ok
Reallocated Sector Count		100	5	Ok
Seek Error Rate		100	67	Ok
Seek Time Performance		100	40	Ok
Power On Hours Count		96	0	Ok
Spin Retry Count		100	60	Ok
Power Cycle Count		100	0	Ok

Informační systém – Graf zaplnění operačních pamětí

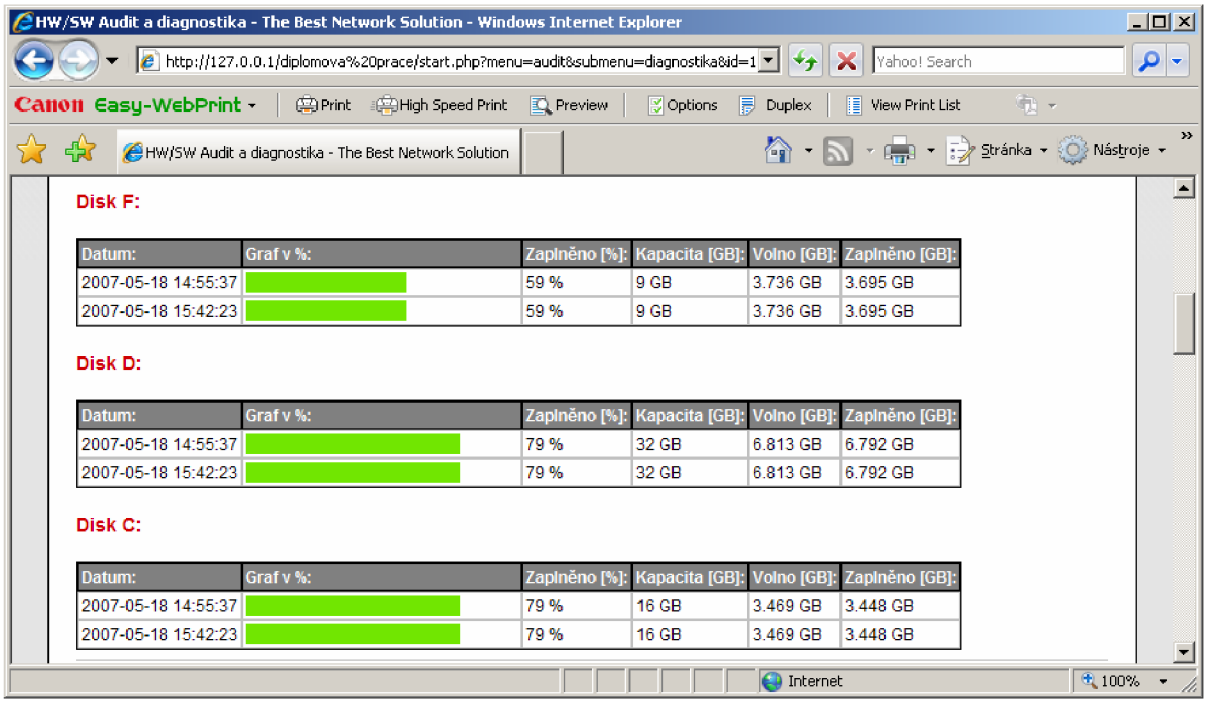


Datum:	Graf v %:	Zaplněno [%]:	Kapacita [MB]:	Volno [MB]:	Zaplněno [MB]:
2007-05-18 14:55:37		82 %	536 MB	93 MB	443 MB
2007-05-18 14:56:38		82 %	536 MB	94 MB	442 MB
2007-05-18 14:57:39		82 %	536 MB	94 MB	442 MB
2007-05-18 14:58:40		82 %	536 MB	92 MB	444 MB
2007-05-18 14:59:41		82 %	536 MB	93 MB	443 MB
2007-05-18 15:00:44		82 %	536 MB	93 MB	443 MB
2007-05-18 15:01:45		82 %	536 MB	93 MB	443 MB
2007-05-18 15:02:46		82 %	536 MB	93 MB	443 MB
2007-05-18 15:03:47		82 %	536 MB	93 MB	443 MB
2007-05-18 15:04:51		82 %	536 MB	92 MB	444 MB
2007-05-18 15:05:52		82 %	536 MB	93 MB	443 MB
2007-05-18 15:06:53		82 %	536 MB	93 MB	443 MB
2007-05-18 15:07:54		82 %	536 MB	93 MB	443 MB
2007-05-18 15:08:58		82 %	536 MB	93 MB	443 MB
2007-05-18 15:42:23		84 %	536 MB	85 MB	451 MB
2007-05-18 15:43:24		83 %	536 MB	86 MB	450 MB
2007-05-18 15:44:25		84 %	536 MB	85 MB	451 MB

Informační systém – Graf zatížení procesoru



Informační systém – Grafy zaplnění disků



1.7 Ukázky hlášení změn a chyb