VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

APLIKACE PRO ZAMEZENÍ POUŽÍVÁNÍ NELEGÁLNÍHO A NEŽÁDOUCÍHO SOFTWARU

APPLICATION FOR PREVENTION OF ILLEGAL AND UNWANTED SOFTWARE USING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAKUB POPOVSKÝ
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN ROUPEC, Ph.D.
SUPERVISOR

BRNO 2012

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2011/2012

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Jakub Popovský

který/která studuje v **bakalářském studijním programu**

obor: **Aplikovaná informatika a řízení (3902R001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Aplikace pro zamezení používání nelegálního a nežádoucího softwaru

v anglickém jazyce:

Application for prevention of illegal and unwanted software using

Stručná charakteristika problematiky úkolu:

Aplikace bude zabraňovat spouštění nežádoucího a/nebo nelegálního software v lokální počítačové síti. Aplikace musí být co nejméně náročná na systémové zdroje počítače.

Cíle bakalářské práce:

Aplikace bude řešit následující problémy:
- vyhledání nežádoucího spuštěného SW a jeho ukončení,
- sběr dat o pokusech používat nežádouucí SW a jejich centrální zpracování,
- ochrana programu "agenta" proti ukončení,
- spouštění "Agenta" při přihlášení uživatele,
- distribuce aktualizovaného seznamu povoleného SW k "agentům" prostřednictvím počítačové sítě.

Seznam odborné literatury:

Chalupa, R.: 1001 tipů a triků pro Visual C++, Computer Press, 2003.
Nagel, Ch. a kol.: C# 2008 Programujeme profesionálně. Computer Press, 2009.
MSDN Library

Vedoucí bakalářské práce: Ing. Jan Roupec, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2011/12.

V Brně, dne 24.2.2012

L.S.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ
DĚKANÁT

Ing. Jan Roupec, Ph.D.
Ředitel ústavu

prof. RNDr. Miroslav Doupovec, CSc.
Děkan

ABSTRAKT

Cílem této bakalářské práce je vyvinout aplikaci pro MS Windows, která bude zabraňovat spouštění nežádoucího a / nebo nelegálního softwaru v lokální počítačové síti. Aplikace musí být co nejméně náročná na systémové zdroje počítače.

ABSTRACT

The aim of this thesis is to develop an application for MS Windows, that will prevent using of unwanted and / or illegal software in the local computer network. Application must be as least demanding on system resources as possible.

KLÍČOVÁ SLOVA

.NET Frameworks, Aplikace, C#, C++, Hákování, Knihovna, Nelegální, Nežádoucí, MySQL, Proces, Software, Vývoj, Whitelist, Windows, XML, Zabezpečení

KEYWORDS

.NET Frameworks, Application, C#, C++, Development, Hook, Illegal, Library, MySQL, Process, Security, Software, Unwanted, Whitelist, Windows, XML

PROHLÁŠENÍ O ORIGINALITĚ

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, pod vedením mého vedoucího. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Brně dne 22.5.2012

.....

BIBLIOGRAFICKÁ CITACE

POPOVSKÝ, J. *Aplikace pro zamezení používání nelegálního a nežádoucího softwaru.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2012. 37 s. Vedoucí bakalářské práce Ing. Jan Roupec, Ph.D.

PODĚKOVÁNÍ

Rád bych poděkoval mému vedoucímu práce, panu Ing. Janu Roupci, Ph.D. a Střední odborné škole průmyslové Edvarda Beneše a Střednímu odbornému učilišti, Břeclav, nábř. Komenského 1 za podporu této práce.

Obsah:

1 ÚVOD

1.1 Cíl práce

Cílem této práce je vytvořit aplikaci pro blokování nežádoucího a nelegálního softwaru na principu whitelist. V rámci této práce je postupně diskutováno několik možných řešení každé části takové aplikace, jejich vhodnost a důvod volby jednoho z nich.

1.2 Podpora a zázemí

Tato práce a aplikace, která je jejím cílem, byla a je vyvíjena ve spolupráci se Střední odbornou školou průmyslovou Edvarda Beneše a Středním odborným učilištěm, Břeclav, nábř. Komenského 1 (dále jen SOŠP Břeclav). Tato škola laskavě zapůjčila své počítače a žáky pro testování vyvíjené aplikace.

Obr. 1 SOŠP Břeclav (hlavní budova)

1.3 Kódové označení aplikace

V rámci této práce je vyvíjená aplikace pro zamezení používání nelegálního a nežádoucího softwaru pojmenována kódovým označením ValSec. Toto kódové označení je použito čistě z potřeby pojmenovat vyvíjenou aplikaci, aby bylo v textu zřejmé, že se jedná právě o tuto aplikaci.

2 ROZBOR PROBLÉMU

Aplikaci pro zamezení užívání nelegálního a nežádoucího softwaru je potřeba použít tam, kde není možné efektivně postihnout uživatele za „zavlečení“ a používání nežádoucí aplikace, ale zároveň mu nemůžeme zcela zabránit.

Jedná se tedy především o internetové kavárny, školy a jiná veřejná místa, kde nemůžeme zakázat užívání USB zařízení ani ukládání na lokální disk z důvodu potřeby práce s daty. Omezení práv uživatele (zakázání instalování aplikací, zákaz zápisu na lokální disk) tento problém neřeší, pouze znesnadňuje použití některého nežádoucího softwaru.

Cílem tedy je, aby uživatel na stanici dělal to, co dělat má (například procházel internet), měl možnost si získaná data odnést (použití USB zařízení a elektronické pošty) ale bylo mu zabráněno ve spouštění ostatních (nežádoucích) aplikací.

2.1 Situace na SOŠP Břeclav

SOŠP Břeclav je střední škola s vhodným prostředím pro nasazení a uplatnění aplikace pro zamezení užívání nelegálního a nežádoucího softwaru. Jako každá moderní škola je vybavena řadou počítačových učeben, kde jsou žáci vyučováni práci s internetem, daty a základy programování. SOŠP Břeclav se potýká s užíváním žáky „zavlečenými“ aplikacemi, zejména hrami. Žákům není možno zakázat používání USB zařízení a elektronické pošty (z důvodů výuky) a tak dochází k „zavlečení“ nežádoucích aplikací. Přítomnost těchto nežádoucích aplikací na pevných discích stanic, síťových discích jednotlivých studentů a tříd nebo na USB zařízeních žáků jim umožňuje nevěnovat se výuce a rušit své kolegy studenty.

Učitelé nezvládají zároveň přednášet látku a kontrolovat práci studentů. Škola nemá finance na vytvoření kamerového systému, zakoupení sledovacího softwaru (Master eye) ani dostatek personálu na neustálé čištění uživatelských stanic.

Nasazení aplikace pro zamezení používání nelegálního a nežádoucího softwaru je tedy jedinou možností, jak zaručit že se studenti budou věnovat probírané látce a úkolům namísto hraní her. Tato aplikace sice nezabrání „zavlečení“ nežádoucích dat, ale zabrání jejich používání.

Problémem whitelist přístupu je výuka programování, kde žáci vytváří vlastní spustitelný software. Proto je potřeba, aby aplikaci pro zamezení užívání nelegálního a nežádoucího softwaru bylo možno kdykoliv, rychle a snadno dočasně deaktivovat.

Jiné problémy, spojené především s internetem, jako jsou například P2P sítě, zahlcení sítě streamovaným videem, hraní online browserových her, prohlížení pornografie a možnost napadení sítě viry je řešeno pomocí nejrůznějších aplikací na centrálním serveru (Linux).

2.2 Stručné shrnutí požadavků

Aplikace musí splňovat následující požadavky:

- Blokování nebo ukončování procesů na principu whitelist.
- Aplikace musí být kompaktní a zbytečně nezatěžovat uživatelské stanice.
- Možnost rychle kontrolu vypnout a zapnout.
- Centrální sběr dat a řízení v rámci domény.

2.3 Prostředí, pro které je aplikace určena

Prostředí, pro které je aplikace ValSec vyvíjena a ve kterém byla její funkčnost testována, obsahuje stanice s operačními systémy Windows XP SP3 a Windows 7 SP1. Tyto stanice jsou přihlášeny do domény a řízeny systémovou politikou. Stanice jsou plně aktualizovány.

V testovacím prostředí je používán antivirový software od společnosti AVG [1]. To nám umožňuje testovat odezvu antivirových softwarů na běh vyvíjené aplikace (zda nebude považována za virus).

K nasazení aplikace je možno použít přístupu pomocí účtu administrátora, nastavení systémové politiky a systémového registru. Ke svému běhu je ale již používat nemůže.

2.4 Dostupná řešení

Ve chvíli tvorby této práce se na trhu nachází několik placených i freeware aplikací, které se zabývají blokováním procesů. Z placených například Application Control od společnosti Lumension [2]. Z freeware aplikací pak Process Blocker od společnosti Softros Systems, Inc [3].

Nevýhodou placených aplikací je především jejich značná cena. Freeware aplikací pak jejich častá nespolehlivost, nesplnění všech požadavků a nedostatek nebo neexistence zákaznické podpory.

3 POSTUP ŘEŠENÍ

Tato kapitola se zabývá samotným vývojem aplikace ValSec. Je rozdělena podle problémů, které musí aplikace ValSec řešit a částí jejího vývoje. Jednotlivé podkapitoly jsou:

1. Volba programovacích nástrojů
2. Získání seznamu běžících procesů
3. Identifikace nežádoucího procesu
4. Ukončení nežádoucího procesu
5. Načtení a uložení nastavení
6. Serverová část aplikace
7. Uživatelské rozhraní
8. Ochrana aplikace proti ukončení uživatelem
9. Spouštění a ukončování aplikace

3.1 Volba programovacích nástrojů

Ještě před řešením jednotlivých částí aplikace, možností a případných problémů, je potřeba zvolit si vhodné programovací nástroje. Tato volba totiž omezuje možnosti použití nejrůznějších knihoven a tříd.

3.1.1 Platforma .NET

Platforma .NET Framework je vývojová platforma navržená společností Microsoft pro jednoduchý vývoj aplikací. Rozhraní .NET Framework je instalováno jako součást instalace operačních systémů řady Windows Server 2003 a novější. Prostředí .NET Framework je také součástí aktualizací systémů řady Windows XP.

Platforma .NET Framework obsahuje několik programovacích jazyků a velkou databázi použitelných knihoven a tříd. Je také široce používána programátorskou veřejností a podporována velkým množstvím tutoriálů a výukových textů.

Tato vývojová platforma umožňuje vyvíjet aplikace nezávisle na cílovém systému a dokonce i hardwaru (je možno v ní vyvíjet například aplikace pro mobilní telefony i osobní počítače). Dále umožňuje kombinovat v rámci jednoho projektů několik programovacích jazyků.

Díky své široké podpoře je platformu .NET vhodné použit pro vývoj aplikace ValSec. Většina problémů řešených při vývoji se totiž dá najít předřešena v MSDN (Microsoft Developer Network) nebo v rámci vyhledávání na Internetu.

Více o platformě .NET je uvedeno například v [4].

3.1.2 Volba programovacího jazyka

Protože platforma .NET nabízí mnoho běžných programovacích jazyků (např. C, C++, C#, Basic, J) a tvorba v prostředí .NET je na volbě jazyka nezávislá, je volba programovacího jazyka spíše otázkou osobních preferencí a vkusu, než důležité a omezující rozhodnutí které ovlivní celý projekt.

Aplikace ValSec je napsána v jazyku C#, s výjimkou hákovací knihovny (je probráno dále), která je napsána v jazyku C++.

3.2 Získání seznamu běžících procesů

Prvním krokem k vytvoření aplikace pro správu nežádoucích aplikací je získání seznamu běžících procesů.

3.2.1 Co je to proces a jak s nimi systém Windows zachází

Zjednodušeně řečeno, je proces běžící aplikací (programem). Proces je umístěn v systémové paměti a obsahuje nejen kód programu, ale i příslušná (měnící se) data. Jeden program může být rozdělen do několika běžících a na sobě často nezávislých procesů.

Moderní počítačové systémy umožňují multitasking, což je zdánlivý současný běh několika procesů najednou. Ve skutečnosti se procesy (velmi rychle) střídají v přístupu k procesoru a jiným systémovým zdrojům.

Toto vedlo k potřebě zavést Správu procesů, která musí zajišťovat, že data jednotlivých procesů budou zachována a nebudou moct být (náhodně, neúmyslně) ovlivněna chodem jiného procesu, a řídit přístup procesů k procesoru a jiným systémovým zdrojům.

3.2.2 Třída pro práci s procesy

Dle [5] je ve vývojovém prostředí .NET možno pro práci s procesy použít třídu *System.Diagnostics.Process*. S její pomocí je možno diagnostikovat běžící procesy, ukončovat je, nebo vytvářet nové.

Mezi nejdůležitější vlastnosti třídy *System.Diagnostics.Process* patří především proměnné:

- *IntPtr Handle* (Handle procesu v nativním kódu)
- *int Id* (Identifikátor daného procesu)
- *string ProcessName* (Název procesu)

Výše uvedené proměnné jsou jen několik málo vlastností této třídy. Třída *System.Diagnostics.Process* obsahuje například také mnoho vlastností pro diagnostiku užití procesoru a paměti.

3.2.3 Samotné získání seznamu procesů

Třída *System.Diagnostics.Process* má několik metod pro získávání procesů:

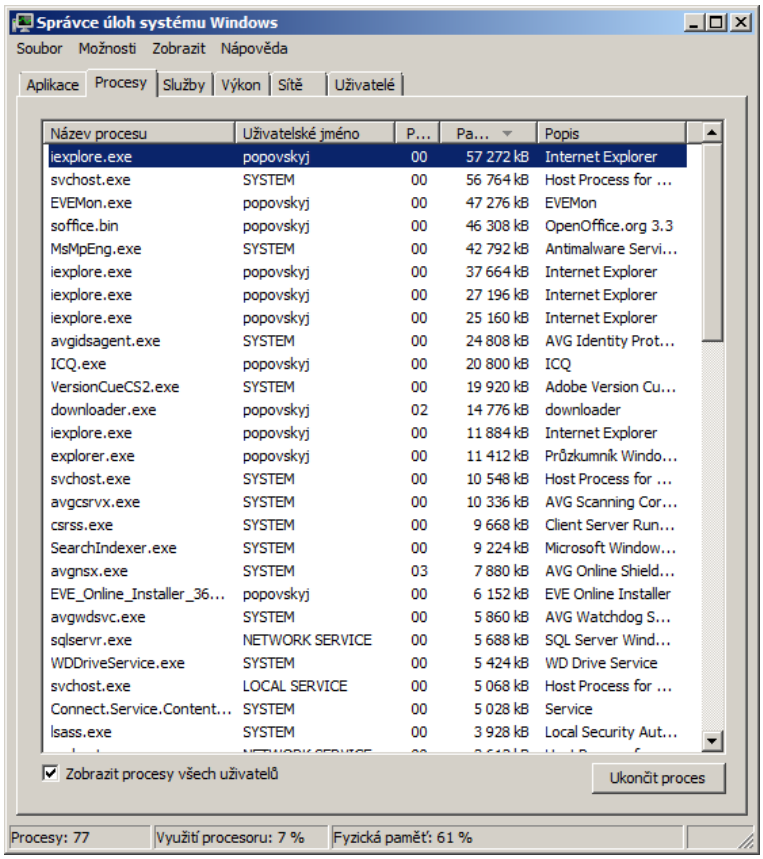
- *public static Process[] GetProcesses();* (Vrátí pole všech procesů běžících na lokálním počítači.)
- *public static Process[] GetProcesses(string machineName);* (Vrátí pole všech procesů běžících na určitém počítači.)
- *public static Process GetProcessById(int processId);* (Vrátí proces s daným Id na lokálním počítači.)
- *public static Process GetProcessById(int processId, string machineName);* (Vrátí proces s daným Id na určitém počítači.)
- *public static Process[] GetProcessesByName(string processName);* (Vrátí pole procesů s daným názvem na lokálním počítači.)
- *public static Process[] GetProcessesByName(string processName, string machineName);* (Vrátí pole procesů s daným názvem na určitém počítači.)

3.2.4 Příklad získání seznamu procesů v .NET C#

Seznam běžících procesů (a tím příslušné Handle a Id procesu) můžeme v C# snadno získat například následovně:

private Process[] JedouciProcesy = System.Diagnostics.Process.GetProcesses();

Správnost získaného seznamu můžeme snadno ověřit porovnáním proměnných ProcessName například se seznamem procesů programu Správce úloh systému Windows (taskmgr) nebo jakýmkoliv jiným programem pro správu procesů.



Obr. 2 Seznam běžících procesů v programu Správce úloh (taskmgr)

3.3 Identifikace nežádoucího procesu

Protože je aplikace ValSec založena na principu whitelist (seznam povolených položek, neboli „co není povoleno je zakázáno“), je během běhu aplikace každý proces kontrolován vůči tomuto listu.

Aplikace ValSec používá tři hlavní seznamy procesů. Jedná se o:

- Seznam povolených procesů (náš whitelist obsahující všechny povolené procesy)
- Seznam běžících procesů (je získán v každém cyklu od systému)
- Seznam ověřených procesů (procesy, které byly ověřeny jako povolené nebo získaly dočasnou povolenku)

3.3.1 Seznam povolených procesů

Seznam povolených procesů je načten z pevného disku při spuštění aplikace a obsahuje *ProcessName* (název povoleného procesu) a *FileName* (cesta k .exe souboru procesu). Tyto informace dostačují k identifikaci povoleného procesu.

Bezpečnostní díra aplikace: Uživatel by mohl podvrhnout svůj (nežádoucí) program za jeden z povolených, pokud by jej přepsal. Tomuto se ale dá v doméně snadno zabránit pomocí systémové politiky.

3.3.2 Seznam běžících procesů

Seznam běžících procesů je získán v každém cyklu programu pomocí funkce *System.Diagnostics.Process.GetProcesses()* . Její použití bylo popsáno v kapitole 3.2.

3.3.3 Seznam ověřených procesů

Seznam ověřených procesů obsahuje informace o procesu z Seznamu běžících procesů, který byl ověřen vůči Seznamu povolených procesů a získal úplnou nebo dočasnou povolenku. Proces je identifikován pomocí proměnných *Id* (unikátní identifikátor), *ProcessName* (název povoleného procesu) a *FileName* (cesta k .exe souboru procesu). Další použité proměnné jsou *DocasnaPovolenka* (byla procesu vystavena pouze dočasná povolenka?) a *ZbyvaDocasnePovolenky* (obsahuje počet kontrol, po kterých vyprší dočasná povolenka).

3.3.4 Získání jména vlastníka procesu

Velmi efektivním filtrem procesů je zjistit, komu proces patří. Značná část (systémových) procesů totiž běží pod uživateli NETWORK SERVICE, LOCAL SERVICE a SYSTEM. Procesy těchto uživatelů není potřeba kontrolovat. Navíc, v rámci bezpečnosti, je dobré nekontrolovat procesy uživatele Administrator, aby bylo možné se k uživatelské stanici přihlásit v případě chybné funkce aplikace.

Dle [6] existují dva způsoby, jak získat jméno uživatele, který proces vlastní. Jsou to:

1. Použít Windows Management Instrumentation (WMI)
2. Použít Win32 API

První metoda je snadnější a její zdrojový kód je podstatně kratší. Její provedení ale trvá výrazně déle než u druhé metody (u několika procesů se často dostáváme již do řádu sekund). Druhá metoda je sice pracnější, ale její provedení je podstatně rychlejší. Obě metody vyžadují znalost *Handle* procesu. Zdrojový kód obou metod je dostupný v [6].

V aplikaci ValSec je použita 2. metoda.

Bezpečnostní díra aplikace: Uživatel by mohl podvrhnout vlastníka svého (nežádoucího) procesu za jeden z povolených vlastníků. K tomu by ale potřeboval znát heslo daného uživatele. V našem případě uživatele Administrator. Pokud by ale neoprávněný uživatel získal toto heslo, bude možnost podvrhnout vlastníka procesu nejmenším z možných problémů.

3.3.5 Výjimky

Do aplikace je vhodné zadat napevno některé výjimky, které zajistí možnosti alespoň minimálního chodu počítače v případě poruchy chodu aplikace (jako je například vymazání obsahu whitelistu). Jedná se především o již zmíněné procesy uživatelů NETWORK SERVICE, LOCAL SERVICE, SYSTEM a Administrator. Další vhodné výjimky jsou:

- taskmgr (cesta: C:\windows\system32\taskmgr.exe ; Jedná se o program pro správu úloh.)
- dllhost (Cestu není možno zjistit; Jedná se o COM+ hosting proces, který používá mnoho aplikací. Pokusy o ukončení těchto procesů by mohly vést k nestabilitě aplikací nebo celého systému.)

3.3.6 Postup ověření procesů

Cyklus ověřování procesů funguje v aplikaci ValSec následovně:

1. Získáme od systému Seznam běžících procesů.
2. Každý proces ověříme, podle jeho *Id*, jestli už nebyl zkontrolován, vůči Seznamu ověřených procesů.
3. Procesy, které ještě nebyly zkontrolovány ověříme vůči Seznamu povolených procesů. Pokud ověření selže (stává se z důvodu nemožnosti získat *FileName* u procesů, které se teprve spouští) je procesu vystavena dočasná povolenka. Procesy, které byly ověřeny vůči Seznamu povolených procesů jsou zapsány do Seznamu ověřených procesů.
4. U procesů, které zbyly, zjistíme jméno uživatele, kterému patří, a pokud se jedná o systémový proces nebo proces uživatele Administrator, je přidán do Seznamu ověřených procesů.
5. Zbývající procesy jsou identifikovány jako nežádoucí a tedy označeny k ukončení. Pokud má proces dočasnou povolenku, je její hodnota snížena o 1. Pokud dosáhne nuly, je program také označen za nežádoucí a určen k ukončení.

Tento systém ověření procesů se snaží co nejméně používat Získání jména vlastníka procesu, protože se jedná o časově náročnější kód, než ověřování vůči whitelistu.

Je důležité poznamenat, že Seznam ověřených procesů a Seznam povolených procesů se mezi jednotlivými cykly aplikace nenulují. Zatímco Seznam povolených procesů zůstává během chodu aplikace neměnný, do Seznamu ověřených procesů jsou postupně přidávány další a další ověřené procesy. Je proto potřeba na něm provádět údržbu a procesy, které byly ověřeny a již byly (uživatelem) ukončeny, z něj odstranit.

Poznámka: Během vývoje aplikace ValSec bylo zjištěno, že operační systémy Windows jsou case-sensitive (citlivé na malá a velká písmena) ve jménech procesů. Pro zjednodušení a zamezení případným problémům převádí aplikace ValSec všechny názvy na lcase (malá písmena).

3.4 Ukončení nežádoucího procesu

Nyní, když aplikace identifikovala nežádoucí proces, je potřeba jej ukončit. K dispozici je několik metod. Mezi nejběžnější z nich patří:

- *Process.CloseMainWindow();*
- *Process.Kill();*
- *Process.Close();*
- *Process.Dispose();*

Všechny čtyři metody jsou součástí třídy *System.Diagnostics*. Jednotlivé metody jsou dále probrány podrobněji.

3.4.1 Metoda *CloseMainWindow*

Tato metoda zavře proces, který má uživatelské rozhraní, zasláním ukončovací zprávy jeho hlavnímu oknu. Což ve správně navrženém programu způsobí ukončení všech dceřiným oken a nakonec celého programu. Tuto metodu je možné použít pouze pro lokálně běžící procesy. Není možné ji použít k ukončování procesů na jiných počítačích.

Tato metoda vrací hodnotu *true*, pokud byla správa odeslána a nebo *false*, pokud se odeslání nezdařilo (cílový proces nemá uživatelské rozhraní).

Důležité: Tato metoda nenutí cílový proces, aby se ukončil. Pouze jej k ukončení vyzývá. Proces může tuto výzvu ignorovat. Dalším velkým nedostatkem této metody je potřeba uživatelského rozhraní, kterému by mohla být správa zaslána. Posledním nedostatkem je, že metoda nevrací informaci o tom, jestli byl proces ukončen nebo jestli se ukončuje, ale pouze o tom, jestli byla zpráva odeslána.

Více o této metodě obsahuje [7].

3.4.2 Metoda *Kill*

Tato metoda přinutí proces k ukončení. Tato metoda vyvolá nenormální ukončení procesu a tak může dojít ke ztrátě dat. Kill je jedinou metodou pro ukončení procesů bez uživatelského rozhraní. Tuto metodu je možné použít pouze pro lokálně běžící procesy. Není možné ji použít k ukončování procesů na jiných počítačích.

Tato metoda nemá návratovou hodnotu.

Důležité: Tato metoda je vhodnou metodou pro násilné a okamžité ukončování procesů bez ohledu na jejich data.

Více o této metodě obsahuje [8].

3.4.3 Metoda *Close*

Tato metoda uvolní všechny systémové zdroje spojené s cílovou komponentou (procesem). Tuto metodu je možné použít pouze pro lokálně běžící procesy. Není možné ji použít k ukončování procesů na jiných počítačích.

Tato metoda nemá návratovou hodnotu.

Důležité: Tato metoda, i když se tak může zdát z jejího názvu, neslouží k ukončení procesu a není možné s její pomocí proces ukončit. Slouží pouze k uklizení přidělených systémových zdrojů při ukončování procesu. Je proto vhodné ji zavolat po ukončení procesu pomocí jiné metody.

Více o této metodě obsahuje [9].

3.4.4 Metoda *Dispose*

Tato metoda uvolní všechny systémové zdroje spojené s cílovou komponentou (procesem) a zanechá ji v nepoužitelném stavu. Tuto metodu je možné použít pouze pro lokálně běžící procesy. Není možné ji použít k ukončování procesů na jiných počítačích.

Tato metoda nemá návratovou hodnotu.

Tato metoda je poděděna z třídy *System.ComponentModel*.

Důležité: Přestože tato metoda slouží pouze k uklizení přidělených systémových zdrojů při ukončování procesu, použití této metody na běžící proces způsobí jeho pád. I přestože se nejedná o vhodnou metodu pro ukončování procesů, je ji možno použít k jeho poškození a tím pádem ukončení.

Více o této metodě obsahuje [10].

3.4.5 Volba metody

I když se může zdát použití metody *Kill* jako nejvhodnější, je důležité si uvědomit, že v případě použití této metody nemá cílový proces téměř žádnou šanci uložit svá data a může také dokonce dojít k poškození souborů na pevném disku.

Proto je vhodné zvážit použití metody *CloseMainWindow*. Ta sice nezaručuje ukončení cílového procesu a navíc nedokáže ukončit procesy bez uživatelského rozhraní, ale zaručuje bezpečné a spolehlivé ukončení práce procesu a uložení jeho dat.

Po ukončení procesu je vhodné použít metodu *Close*, která zaručí, že všechny systémové prostředky procesu byly uvolněny.

V případě, že je proces velmi odolný ukončení a metoda *Kill* neuspěla, je možné použít metodu *Dispose*, jejímž nejspíše neúmyslným účinkem je nestabilita procesu a jeho pád. Tato metoda by ale měla být použita pouze v opravdu krajních případech.

Poznámka: Během testování aplikace bylo zjištěno, že procesy, které se teprve spouští jsou odolné metodě *Kill*. Není ale vhodné je ukončovat pomocí metody *Dispose*. Vhodnější je počkat, až se proces načte do paměti a spustí, a až pak jej ukončit.

Správný postup užití metod by tedy měl být:

1. Použít metodu *CloseMainWindow*.
2. Pokud metoda neuspěla (proces nemá uživatelské rozhraní) nebo pokud se proces odmítá ukončit, použít metodu *Kill*.
3. Pokud jedna z předchozích metod uspěla, použít metodu *Close*.
4. Pokud ani metoda *Kill* neuspěla, použít metodu *Dispose*.

Aplikace ValSec používá metody *Kill* a *Dispose*. Testováním bylo zjištěno, že použití těchto metod nezpůsobí nestabilitu systému a není potřeba brát ohled na data nežádoucích aplikací. Garbage collector systému Windows je dostatečně robustní, aby zvládl uklidit „nepořádek“ který zůstane po užití těchto metod.

Možnost vylepšení: I přesto, že je tento přístup dostačující, není vhodné systému přidělávat práci se správou systémových zdrojů. V budoucnu by tedy bylo vhodné tuto část aplikace ValSec vylepšit, aby používala i metodu *CloseMainWindow*.

3.5 Načtení a uložení nastavení

Whitelist, a případné další nastavení aplikace, je potřeba při jejím spuštění načíst. I přestože bude dále probráno spojení se serverovou částí, není vhodné načítat whitelist při každém spuštění aplikace ze serveru. Je vhodnější mít jej uložen lokálně na pevném disku stanice a aktualizovat jej pouze v případě potřeby. Takto nebudeme zbytečně zatěžovat provoz sítě a server.

To tedy znamená, že whitelist a nastavení není potřeba pouze načítat, ale také uložit po stažení aktualizace od serverové části.

3.5.1 Načtení a uložení obsahu souboru pomocí třídy *System.IO*

Jedním z nejjednodušších způsobů, jak načíst obsah souboru v prostředí .NET je použít třídu *System.IO*. Tato třída obsahuje základní nástroje pro práci se soubory a jejich obsahem. Pro nás nejvhodnějším nástrojem bude načíst soubor po řádcích.

Načtení řádků pomocí této metody může v C# vypadat například takto:

```
string[] lines = System.IO.File.ReadAllLines(@"C:\test.txt");
```

A ukládání řádků například takto:

```
string[] lines = {"první radek", "druhy radek", "treti radek"};
System.IO.File.WriteAllLines(@"C:\test.txt", lines);
```

Tato metoda je vhodná pro načítání a ukládání stejných záznamů do jednoho souboru (jako je například whitelist, který obsahuje velké množství záznamů se stejnou strukturou). Není vhodný pro načítání a ukládání různých druhů záznamů (různé druhy proměnných, jako je například nastavení aplikace).

Více o této metodě například v [11].

3.5.2 Načtení a uložení obsahu souboru pomocí XML

XML je zkratka z anglického eXtensible Markup Language, v překladu rozšiřitelný značkovací jazyk. XML je formát, který slouží k strukturalizaci dat. Svou strukturou je velmi podobný HTML (oboje používá tagy a atributy). Na rozdíl od HTML, XML ale nespecifikuje k čemu jednotlivý atribut nebo tag slouží (to určí programátor), pouze specifikuje strukturu dokumentu. Také je důležité zmínit, že oproti HTML, XML je velmi citlivé na syntaxi a zapomenutí párové značky nebo jejího uzavření vede k zneplatnění celého dokumentu.

Nevýhodou XML jsou velké nároky na místo při ukládání. XML tedy není vhodné použít k ukládání velkého množství dat. Je ale velmi vhodné pro načítání a ukládání různých druhů záznamů (různé druhy proměnných, jako je například nastavení aplikace).

Načtení uzlů XML dokumentu (zde je z načteného souboru vyhledán atribut *hodnota* uzlu *HASH*) může v C# vypadat například takto:

```
XmlDocument xml = new XmlDocument();
xml.Load(@"C:/test.xml"); //Načteme XML soubor
XmlElement koren = xml.DocumentElement; //Hlavní uzel
foreach (XmlNode uzel in koren) //Projedeme všechny uzly v dokumentu
{
        if (uzel.Name == "HASH") Promenna = uzel.Attributes["hodnota"].Value;
}
```

A uložení celého dokumentu (který obsahuje uzel *CasVytvoreniZnacky* s atributem *hodnota* uvnitř uzlu *Znacka*) například takto:

```
using (XmlWriter writer = XmlWriter.Create(@"C:/test.xml"))
{
        writer.WriteStartDocument();
                writer.WriteStartElement("Znacka");
                        writer.WriteStartElement("CasVytvoreniZnacky");
                                writer.WriteAttributeString("hodnota", SoucasnyCas);
                        writer.WriteEndElement();
                writer.WriteEndElement();
        writer.WriteEndDocument();
}
```

Více o této metodě například v [12].

3.5.3 Volba způsobu načtení a uložení

Načtení a uložení obsahu souboru pomocí třídy *System.IO* je snadné použít a je vhodné pro načítání a ukládání velkého množství dat stejné struktury. Whitelist je proto vhodné načítat a ukládat pomocí této metody.

Načtení a uložení obsahu souboru pomocí XML je pracnější na použití, ale je vhodné pro načítání a ukládání dat různých typů. Tuto metodu je proto vhodné použít k načítání a ukládání nastavení aplikace.

Aplikace ValSec používá načtení a uložení obsahu souboru pomocí třídy *System.IO* pro načítání a ukládání whitelistu a načtení a uložení obsahu souboru pomocí XML pro načítání a ukládání nastavení aplikace.

Možnost vylepšení: I přesto, že je načtení a uložení obsahu souboru pomocí třídy *System.IO* dostačující metodou pro načítání a ukládání whitelistu, bylo by vhodné v budoucnu sjednotit oba soubory (s whitelistem a nastavením) do jednoho souboru ve formátu XML, který může snadno obsahovat obě části.

Možnost vylepšení: Namísto XML je možno použít i nějaký jiný značkovací jazyk, jako je například YAML [13], který je úspornější z hlediska místa, snazší na čtení (human friendly) a méně citlivý na dodržení syntaxe (může být chápáno jako výhoda i nevýhoda).

3.6 Serverová část aplikace

Součástí požadavků na vyvíjenou aplikaci je potřeba rychlého centrálního řízení, centrální shromažďování dat a distribuce aktualizací whitelistu. Tyto potřeby zajišťuje serverová část aplikace. V této kapitole jsou navrhnuty tři způsoby řešení serverové části aplikace:

1. Soubory na síťovém disku
2. MySQL databáze
3. WCF serverová aplikace

3.6.1 Soubory na síťovém disku

Nejspíše nejjednodušším způsobem, jak řešit potřebu sdílených serverových dat, je nasdílet síťový disk, ke kterému bude mít aplikace ValSec přístup. Disk můžeme samozřejmě opatřit přístupovým heslem a tak zajistit, že uživatelé nebudou mít k disku přístup.

Na síťovém disku je pak možno umístit soubory s příslušnými whitelisty pro jednotlivé skupiny aplikace ValSec (například pro jednotlivé učebny). Podobně můžeme vytvořit soubory pro zápis záznamů a soubory, které budou obsahovat nastavení aplikace ValSec, a tak ji řídit.

Tento přístup se může zdát být poměrně jednoduchý a snadno proveditelný. Má ale jednu velmi podstatnou nevýhodu. V jednu chvíli může mít soubor otevřený k zápisu jen jedna aplikace, a tak může snadno dojít k zbytečným konfliktům, nutnosti čekat a možnosti vzniku chyb.

3.6.2 MySQL databáze

MySQL je multiplatformní databázový systém. Do MySQL lze ukládat různá data (proměnné, texty, obrázky, aj.), s nimiž lze dále snadno pracovat (třídit, řadit, filtrovat apod.). Tyto data jsou v MySQL databázi rozděleny do tabulek (uloženy v buňkách).

Platforma .NET může s MySQL databází komunikovat pomocí tzv. MySQL konektorů (podrobněji o MySQL konektorech dále).

Propojení aplikace s MySQL databází je poměrně snadné a jeho největší výhodou je snadná tvorba internetového uživatelského rozhraní, přes které je pak možno obsluhovat nastavení, spravovat whitelisty a řídit chod aplikace ValSec. MySQL také řeší současný přístup více uživatelů k databázi.

3.6.3 WCF serverová aplikace

WCF je zkratka z anglického Windows Communication Foundation, což znamená komunikační základy Windows. WCF je framework pro programování aplikací poskytujících služby. V podstatě se jedná o framework pro vytváření aplikací typu server-klient, kdy klient volá vzdáleně funkce serverové části. WCF poskytuje snadný způsob takové komunikace.

Samotná serverová část musí být připojena k nějaké databázi (například MySQL), protože každé spojení klientské části vytvoří novou (vlastní) instanci serverové části.

Více o WCF například v [14].

3.6.4 Volba serverové části aplikace

Použít soubory na síťovém disku je ve skutečnosti pracnější, než by se mohlo zdát, protože je nutno řešit problémy se současnými pokusy o přístup a čekání až bude soubor přístupný. Toto řešení tedy není vhodné.

WCF serverová aplikace je velmi robustním řešením, které umožňuje poskytovat vzdáleně služby (například vzdálené volání funkcí). Pro aplikaci ValSec toto ale není potřeba a zbytečně bychom programovaly funkce, která nám může poskytnout přímo MySQL databáze.

Nejvhodnějším řešením serverové části aplikace ValSec je tedy přímé spojení s MySQL databází.

3.6.5 Komunikace platformy .NET s MySQL

Jak již bylo zmíněno, aby mohla aplikace vyvíjená na platformě .NET komunikovat s MySQL databází, je potřeba do ní nejdříve doinstalovat MySQL konektor. Ten je poskytován zdarma, a je jej možné stáhnout například na [15].

Po nainstalování MySQL konektoru a připojení příslušných knihoven se již můžeme snadno spojit s MySQL databází. V C# může takové spojení vypadat například takto:

```
MySqlCommand cmd = new MySqlCommand();
cmd.Connection = new MySqlConnection("Database=test;DataSource=192.180.0.1;UserId=
Uzivatel;Password=heslo");
cmd.Connection.Open();
cmd.CommandText = "SELECT * FROM `TabulkaPC`";
MySqlDataReader read = cmd.ExecuteReader(); //vykoná dotaz a vrátí do read
while (read.Read())
{
        //zde můžeme pracovat s daty získanými z databáze
}
read.Close();
cmd.Connection.Close();
```

V MySQL databázi pak můžeme snadno vytvořit tabulky pro sběr informací (kam budou instance aplikace ValSec zapisovat pokusy uživatelů o spuštění nepovolených aplikací), tabulky pro ovládání instancí aplikace ValSec (více dále) a tabulky s jednotlivými whitelisty. Aplikace ValSec si pak může snadno zjistit pomocí SQL dotazů jestli je potřeba lokální whitelist aktualizovat.

Poznámka: Spojení s MySQL databází je potřeba zabezpečit uživatelským jménem a heslem. Jméno a heslo je možno buď zadat přímo do zdrojového kódu aplikace nebo jej zakódované uložit do souboru s nastavením. Rozhodně jej není bezpečné ukládat nezakódované.

Detailní rozbor komunikace s databází pomocí SQL dotazů je možno najít například v [16].

3.6.6 Zjištění IP adresy stanice

Protože je aplikace ValSec nasazena v doméně, kde jsou jednotlivým uživatelským stanicím přidělovány IP adresy DHCP serverem na základě rezervací (na základě MAC adres), je možno každou stanici identifikovat podle její IP adresy. Aplikace ValSec se tedy může identifikovat právě IP adresou své stanice.

V MySQL databázi pak můžeme uložit nastavení pro každou stanici podle její IP adresy. Aby pak mohla aplikace ValSec zjistit, který záznam je určen právě pro danou instanci, musí nejdříve zjistit IP adresu své stanice.

Zjištění IP adresy může v C# vypadat například takto:

```
System.Net.IPAddress[] ListIPAdres = System.Net.Dns.GetHostAddresses(System.Net.Dns.
GetHostName());
```

Všimněte si, že návratová hodnota je několik adres, ne jen jedna adresa. To proto, že stanice může mít více síťových karet, a tedy i více IP adres. Je potřeba vyzkoušet je vůči MySQL databázi všechny.

3.7 Uživatelské rozhraní

Uživatelské rozhraní aplikace ValSec je rozděleno do dvou částí.

První část, je uživatelské rozhraní samotné aplikace ValSec, které neobsahuje žádné ovládací prvky a slouží pouze k informování uživatele o chodu aplikace.

Druhou částí je uživatelské rozhraní serverové části, kde je možno aktualizovat whitelist, nastavení aplikace a vypínat a zapínat kontrolu aplikace (tedy v případě potřeby pozastavit kontrolu a tak umožnit spuštění jakéhokoliv procesu).

3.7.1 Zprávy pro uživatele

I když nemá uživatel žádnou (legální) možnost, jak ovlivnit chod aplikace ValSec, a tedy by nebylo žádné uživatelské rozhraní potřeba, je vhodné jej o chodu aplikace a krocích, které podnikla informovat.

Informovat uživatele o chodu aplikace má jednak odstrašující účinek, a jednak zabraňuje matení uživatelů, kteří by bez příslušné zprávy nechápali, proč byla jejich aplikace nečekaně ukončena.

Příklady zpráv pro uživatele jsou uvedeny na Obr. 3 a Obr. 4 .

Zpráva je pro uživatele zobrazena 10 sekund. Okno je vynuceně zobrazeno na popředí (aby se nezobrazilo pod oknem nějaké jiné aplikace) a je možno jej zavřít kliknutím levým tlačítkem myši.

3.7.2 Zjištění jména přihlášeného uživatele

Velmi účinnou zbraní v boji proti nepořádným a neposlušným uživatelům je připomenout jim, že práce na počítači není anonymní. Mnoho dnešních uživatelů trpí falešnou představou, že na počítači (a především internetu) mohou dělat a psát cokoliv je napadne bez nebezpečí nějakého postihu. Samozřejmě, že takové uživatele je možno dohledat pomocí doby přístupu, IP adresy a podobně.

Je proto vhodné, aby si aplikace zjistila a používala jméno přihlášeného uživatele ve svých zprávách, a tak mu dala najevo, že víme, kdo je za daný pokus o spuštění nežádoucí aplikace zodpovědný.

Získání jména přihlášeného uživatele je velmi snadné, protože je uloženo v systémové proměnné *Environment.UserName*.

Poznámka: Dalším velmi vhodným nástrojem, který uživatele rychle odradí od pokusů o spouštění nežádoucích aplikací je zmínit ve zprávě pro uživatele, že jeho pokus o spuštění nežádoucí aplikace byl zaznamenán. A to i kdybychom tuto funkci nepoužívali.

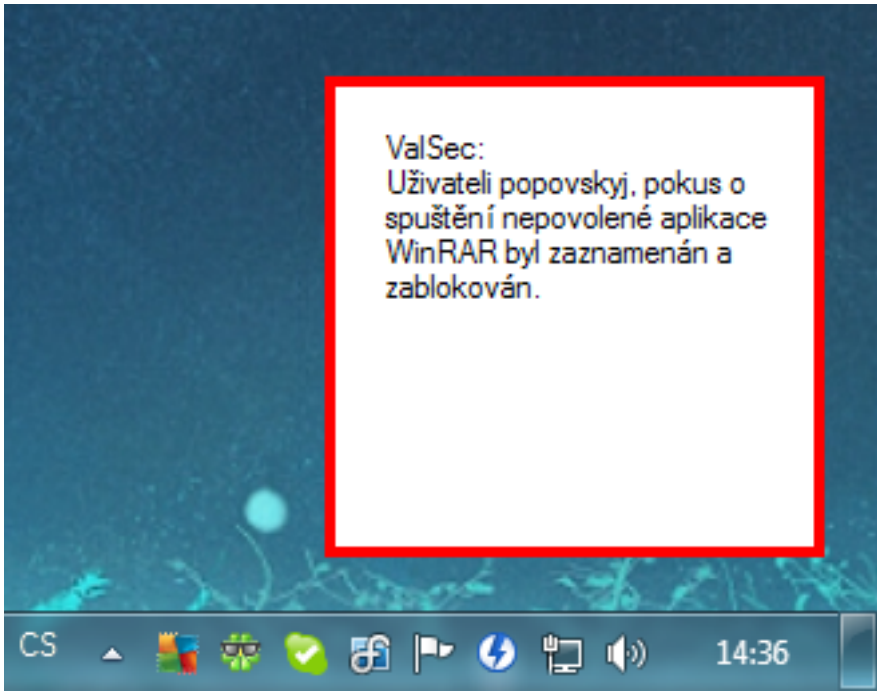
3.7.3 Serverová část aplikace

Veškeré nastavení aplikace ValSec je možno pouze z její serverové části nebo přímým zápisem hodnot do souboru s nastavením (který je chráněn hash klíčem, a tedy není snadné tyto hodnoty přímo měnit).

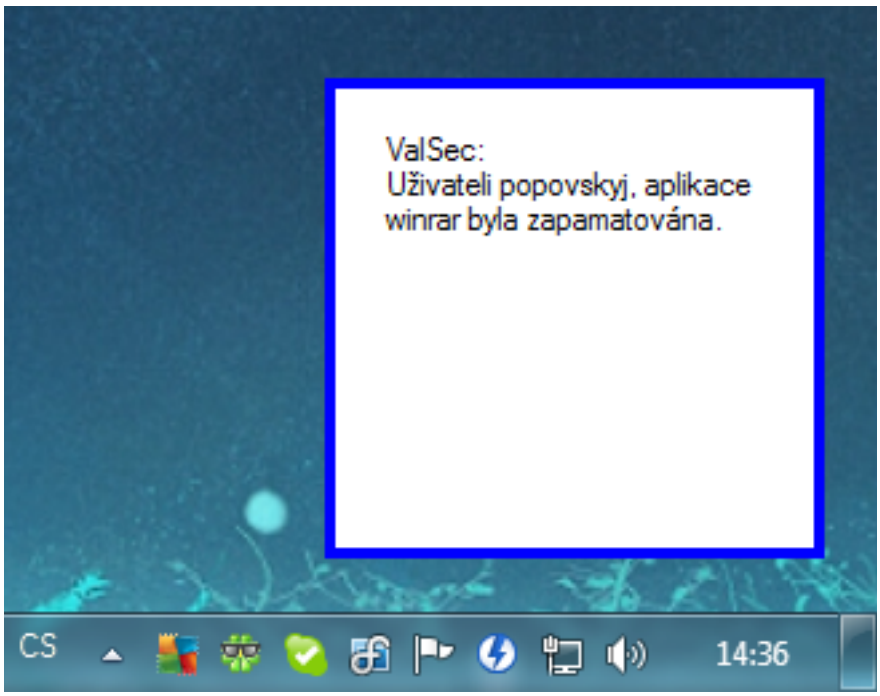
Protože serverová část aplikace ValSec je zajištěna MySQL databází, je možno využít jeden z široké nabídky softwarů pro správu MySQL databáze. Toto řešení uživatelského rozhraní je velmi jednoduché a rychlé. Bohužel není vhodné pro běžného uživatele (v našem případě učitel, který potřebuje rychle a snadno vypnout nebo zapnout kontrolu aplikace ValSec).

Aplikace ValSec používá pro správu MySQL databáze uživatelské rozhraní phpMyAdmin verze 2.9.0.3.

Možnost vylepšení: V budoucnu by bylo vhodné vytvořit internetové uživatelské rozhraní, kde by mohli učitelé po přihlášení zapínat a vypínat kontrolování aplikace ValSec. Také by bylo vhodné změnit způsob vypínání ze zap/vyp na dočasné vypnutí, kdy se po určité době kontrolování pomocí aplikace ValSec samo znovu zapne.



Obr. 3 Zpráva pro uživatele při pokusu o spuštění nepovoleného procesu



Obr. 4 Informativní zpráva pro uživatele (v tomto případě přidání procesu do whitelistu)

3.8 Ochrana aplikace proti ukončení uživatelem

Velmi důležitým krokem vývoje aplikace ValSec je její zabezpečení proti nechtěnému ukončení uživatelem. Toto zabezpečení spočívá v několika krocích, které jsou dále probrány.

3.8.1 Vhodné „uživatelské rozhraní“

Prvním krokem, jak zabezpečit aplikaci proti nechtěnému ukončení, je navrhnout vhodné uživatelské rozhraní. Přesněji, odstranit z něj běžné prvky pro ukončení aplikace. Tím je myšleno malé „x“ v horním pravém rohu jakéhokoliv běžného okna. Funkci tohoto malého x můžeme potlačit uvnitř zdrojového kódu. Jednodušší a účinější je ale jej zcela odstranit nastavením proměnné *ControlBox* na hodnotu *false*.

Další důležitou proměnnou, kterou je potřeba nastavit na hodnotu *false* je proměnná *ShowInTaskbar*. Tím zaručíme, že se okno aplikace nebude zobrazovat v seznamu aplikací, kde by jej například v programu taskmgr bylo možno ukončit.

3.8.2 Vhodné jméno

Velmi snadným krokem, který značně ztíží snahy uživatelů o ukončení aplikace ValSec je „vhodně“ ji pojmenovat. To, že se aplikace ve svých zprávách identifikuje označením ValSec, totiž vůbec neznamená, že by se tak měla jmenovat. Aplikaci je proto vhodné pojmenovat například A10X15Y.exe, cleaner.exe, windump.exe nebo podobně. Důležité je, aby jméno mělo s aplikací co možná nejméně společného.

Podobně adresář, do kterého aplikaci umístíme, je vhodné zvolit někde hluboko uvnitř adresáře Windows a soubory s nastavením, whitelistem a hákovací knihovnou (je probráno dále) pojmenovat rovněž „vhodně“.

3.8.3 Odmítnutí ukončit se

Jak bylo probráno v kapitole 3.4.1, první metodou pro ukončení procesu je *CloseMainWindow*. Tato metoda vyzve aplikaci k ukončení. Proti této metodě se může aplikace snadno a jednoduše ubránit. Stačí zprávu odchytit a odmítnout se ukončit. To může v C# vypadat například takto:

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
        if (!Konec) { e.Cancel = true;}  //Pokud se nechceme ukončit, odpověz zamítnutím
}
```

3.8.4 Metoda *Kill* a aplikace typu Watchdog

Metodu *Kill* již není možno nijak odmítnout, nebo se proti ní ubránit. Aplikace se musí ukončit. Je však možno tuto metodu obejít použitím tzv. Watchdog aplikace.

Aplikace typu watchdog sleduje provoz jiné aplikace a v případě potřeby do něj zasahuje. V našem případě můžeme vytvořit aplikaci, která bude sledovat provoz aplikace ValSec a v případě jejího (náhlého) ukončení ji znovu spustit. Stejně tak aplikace ValSec pak musí sledovat provoz Watchdog aplikace a v případě potřeby ji spustit.

Slepá ulička vývoje: Aplikace Watchdog byla v rámci této práce vyvinuta a testována. Ukázala se být plně schopna sledovat a v případě potřeby znovu spustit aplikaci ValSec. Bohužel se objevil problém s funkcí Ukončit strom procesu aplikace taskmgr, proti kterému nebyla Watchdog aplikace schopna aplikaci ValSec ochránit.

3.8.5 Problém s funkcí Ukončit strom procesu programu taskmgr

Funkce Ukončit strom procesu programu taskmgr ukončí proces včetně všech jeho dceřiných procesů. To například znamená, že při ukončení procesu Watchdog se ukončí i aplikace ValSec a nebo naopak. Záleží, který z procesů je zrovna „matkou“ a který „dcerou“.

Tato funkce se ukázala být velkým problémem v zabezpečení aplikace ValSec, protože byla schopna ukončit v 50% případů chod aplikace ValSec včetně aplikace Watchdog. Bylo by tedy jen otázkou času, než by na to uživatelé přišli.

Slepé uličky vývoje: Jednou z možných řešení tohoto problému se zdálo být nějak přinutit systém Windows, aby přerušil dceřinný vztah mezi aplikacemi. Bylo odzkoušeno spustit aplikaci ValSec pomocí Watchdog aplikace pod jiným uživatelem; pod stejným uživatelem a pak ji přehlásit a dokonce vložit do řetězu spouštění třetí, prostřední aplikaci, která se pak ukončila. Bohužel žádná z těchto možností nebyla úspěšná.

Jediným možným řešením problému tedy je, namísto snahy „přežít“ funkci Ukončit strom procesu, zabránit uživateli v jejím použití.

3.8.6 Hákovací knihovna

Hákovaní (angl. Hook) je termín, který se vžil pro označování procesu, kdy je odchycena zpráva (od systému nebo aplikace) určená nějaké aplikaci nebo systému, zpracována (například pozměněna nebo zapsána) a až pak předána cílové aplikaci nebo systému.

Tohoto principu je možno využít ke skrytí aplikace ValSec před jinými aplikacemi a uživateli. Jak již bylo uvedeno v kapitole 3.2, k získání seznamu jedoucích procesů je použita funkce *System.Diagnostics.Process.GetProcesses*. Po zavolání této funkce vrátí systém aplikaci zprávu obsahující systémové informace o procesech. Tuto zprávu můžeme odchytit a aplikaci ValSec z ní odstranit.

Význam: Pokud aplikaci ValSec vhodně pojmenujeme a navíc ji skryjeme pomocí hákovací knihovny, bude pro uživatele prakticky nemožné zjistit její jméno a ukončit ji.

Varování: Hákovací knihovna se po jejím připojení k systému Windows načte ke každému spuštěnému procesu (jak hákovací knihovnu připojit je probráno dále). To znamená, že pokud se v jejím zdrojovém kódu vyskytne chyba (ať již chybná syntaxe nebo logická chyba), může dojít k pádu všech běžících procesů. Před připojením hákovací knihovny je tedy důležité důkladně ji zkontrolovat a testovat ji nejlépe na k tomu určené stanici, kde by nás případné poškození systému Windows nepřipravilo o důležitá data.

Příklad tvorby hákovací knihovny je velmi detailně probrán v [17].

Připojení hákovací knihovny ke všem procesům je probráno v kapitole 3.9.2.

Možnost rozšíření: Hákovací knihovna, která skrývá aplikaci ValSec, není digitálně podepsána. I když toto nevadí v operačních systémech, pro které je aplikace ValSec vyvíjena (Windows XP SP3 a Windows 7 SP1), do budoucna by bylo dobré ji digitálně podepsat. I když je zatím možno připojit k systému nepodepsanou hákovací knihovnu, v novějších verzích systému Windows již nemusí být nepodepsané hákovací knihovny podporovány.

3.8.7 Závěr problému ochrany aplikace

Během vývoje aplikace ValSec bylo zjištěno, že není možné vyvinout aplikaci, která bude „neukončitelná“ uživatelem. Můžeme ale uživateli tuto snahu ztížit natolik, že se ukončení aplikace stane téměř nemožné.

3.9 Spouštění a ukončování aplikace

Aplikace ValSec a její části jsou připojeny do systému Windows a spouštěny pomocí záznamů v systémovém registru. Navíc, protože je aplikace ValSec chráněna proti běžným způsobům ukončení, je potřeba definovat, kdy se má ukončit (nechceme, aby například zabraňovala odhlášení nebo vypnutí počítače).

3.9.1 Záznam pro aplikaci ValSec v systémovém registru

Aplikace ValSec se spouští při každém přihlášení uživatele. To zajistíme zápisem následující hodnoty do systémového registru:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Run]
"ValSec"="C:\\ValSec.exe"
```

Poznámka: Během spuštění aplikace ValSec se zjišťuje jméno přihlášeného uživatele a kontrolují se aktualizace whitelistu a nastavení aplikace ze serverové části (MySQL). Ty se zjišťují a stahují jen při spuštění aplikace. Žáci se totiž učí v kuse nejdéle 3 vyučovací hodiny (2h 15m + přestávky), což je dostatečně krátká doba. Toto se netýká detekce nastavení zapnutí a vypnutí kontroly, které probíhá každých 5 minut.

Možnost rozšíření: V budoucnu by bylo vhodné změnit četnost kontroly aktualizací na nějaký interval (například při spuštění a každou hodinu). I když se nepředpokládá častá aktualizace whitelistu a nastavení aplikace ValSec, bude rychlejší odezva instancí aplikace vhodnější.

3.9.2 Záznam pro hákovací knihovnu v systémovém registru

Hákovací knihovnu, kterou jsme vytvořili pro skrytí aplikace ValSec, je potřeba připojit k systému Windows pomocí nastavení systémového registru:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows]
"AppInit_DLLs"="C:\\ValSecHook.dll"
"LoadAppInit_DLLs"=dword:00000001
"RequireSignedAppInit_DLLs"=dword:00000000
```

Více o nastavení hodnot registru pro připojení knihovny ke všem procesům můžete najít například v [18].

3.9.3 Odhlašování uživatele

Protože se aplikace ValSec spouští při přihlášení uživatele, je zřejmé, že je potřeba ji vypnout při jeho odhlášení. To, že se uživatel odhlašuje oznamuje systém všem aplikacím zasláním zprávy *WM_QUERYENDSESSION*. Tuto zprávu tedy potřebujeme zachytit a zareagovat na ni ukončením aplikace.

Rozšíříme metodu *WndProc* aplikace, aby se ukončila, když se chce uživatel odhlásit nebo vypnout PC. To může v C# vypadat například takto:

```
private static int WM_QUERYENDSESSION = 0x11;
protected override void WndProc(ref System.Windows.Forms.Message mesage)
{
        if (mesage.Msg == WM_QUERYENDSESSION)
        {
                Konec = true; //Konec je globální proměnná, která přeruší hlavní smyčku
        }
        base.WndProc(ref mesage);
}
```

3.9.4 Kontrola spuštění více aplikací

Výjimečně by se mohlo stát, že se spustí dvě a více instancí aplikace ValSec. Tento stav by mohl způsobit nepořádek v záznamech o provozu aplikace, chyby funkce aplikace (například při pokusu obou instancí uložit nové nastavení nebo whitelist) a další těžko předvídatelné chyby. Je proto důležité, aby aplikace ValSec obsahovala při spuštění kontrolu, jestli již není spuštěna starší instance.

V případě běžné aplikace, by stačilo získat seznam běžících procesů (viz. Kapitola 3.2) a zjistit, jestli už neběží proces se stejným jménem. Aplikace ValSec bohužel tento jednoduchý způsob použít nemůže. Jednak by pak bylo snadné podvrhnout aplikaci se stejným jménem a přinutit aplikaci ValSec k ukončení a také nemůžeme tento způsob použít, protože aplikace ValSec je ze seznamu běžících procesů skryta pomocí hákovací knihovny (viz. Kapitola 3.8).

Je proto potřeba využít značek v podobě malých souborů na pevném disku. Aplikace ValSec při svém spuštění vytvoří značku, do které zapíše čas svého spuštění. Pokud je pak během provozu aplikace značka přepsána novou (a hlavně platnou) značkou, znamená to, že byla spuštěna nová instance aplikace ValSec. V tuto chvíli se stará instance ukončí.

Poznámky: K tomuto systému se dospělo po několika předchozích verzích. Musí se ukončit stará instance, protože nová instance si nemůže ověřit, že stará značka je od stále platné a jedoucí instance. Také je tento přístup vhodný pro případné budoucí aktualizace souboru .exe aplikace ValSec (není implementováno ve stávající verzi), kdy stará instance může spustit novou verzi, a když se vytvořením značky ověří, že je funkční a v provozu, tak se ukončí.

Bezpečnostní díra aplikace: Uživatel by mohl podvrhnout značku o spuštění nové instance aplikace ValSec a tak přinutit starší (platnou a jedinou) instanci k ukončení. K tomu by ale musel najít, kam se značky ukládají a rozluštit hashovací algoritmus, který značky šifruje.

4 ZÁVĚR

V rámci této práce byly vyvinuta ve spolupráci se SOŠP Břeclav aplikace pro zamezení užívání nelegálního a nežádoucího softwaru v lokální počítačové síti, pod kódovým označením ValSec.

Tato aplikace byla vyvinuta ve vývojovém prostředí .NET framework od společnosti Microsoft. Aplikace byla testována a je v současné době nasazena a používána v počítačových učebnách Střední odborné školy průmyslové Edvarda Beneše a Středního odborného učiliště, Břeclav, nábř. Komenského 1.

Během vývoje aplikace bylo zjištěno a zkoumáno mnoho přínosných informací o tom, jak operační systémy řady Windows zachází s procesy aplikací a jaké jsou možnosti zabezpečení aplikace proti nežádoucímu ukončení.

SEZNAM POUŽITÉ LITERATURY

[1] AVG. *AVG Antivirus* [software]. [cit. 2012-05-30]. Dostupné z: http://www.avg.com

[2] LUMENSION SECURITY, INC. *Application Blocker* [software] [cit. 2012-05-30]. Dostupné z: http://www.lumension.com

[3] SOFTROS SYSTEMS, INC. *Process Blocker* [software]. [cit. 2012-05-30]. Dostupné z: http://www.processblocker.com/ .

[4] Vochozka. *.NET, stručná informace*. Zpravodaj ÚVT MU. ISSN 1212-0901, 2003, roč. XIII, č. 5, s. 15-17. Dostupné z: http://www.ics.muni.cz/bulletin/articles/281.html

[5] KOVÁŘ, Dušan. *Procesy* [online]. [cit. 2012-05-30]. Dostupné z: http://projektysipvz.gytool.cz/ProjektySIPVZ/Default.aspx?uid=494

[6] WARLIB. *How To Get Process Owner ID and Current User SID* [online]. 2006-07-15 [cit. 2012-05-30]. Dostupné z: http://www.codeproject.com/Articles/14828/How-To-Get-Process-Owner-ID-and-Current-User-SID

[7] O DONELL, Rory. *Process.CloseMainWindow Method* [online]. 2011-07-18 [cit. 2012-05-30]. Dostupné z: http://msdn.microsoft.com/en-us/library/system.diagnostics.process.closemainwindow.aspx

[8] *Process.Kill Method* [online]. [cit. 2012-05-30]. Dostupné z: http://msdn.microsoft.com/en-us/library/system.diagnostics.process.kill.aspx

[9] SOYUZ. *Process.Close Method* [online]. 2010-12-30 [cit. 2012-05-30]. Dostupné z: http://msdn.microsoft.com/en-us/library/system.diagnostics.process.close.aspx

[10] *Component::Dispose Method* [online]. [cit. 2012-05-30]. Dostupné z: http://msdn.microsoft.com/en-us/library/3cc9y48w.aspx

[11] LEE, Thomas. *File Class (system.IO)* [online]. 2010-07-18 [cit. 2012-05-30].Dostupné z: http://msdn.microsoft.com/en-us/library/system.io.file.aspx

[12] CHAND Mahesh. *Reading and Writing XML in C#* [online]. 2001-10-29 [cit. 2012-05-30]. Dostupné z: http://www.c-sharpcorner.com/UploadFile/mahesh/ReadWriteXMLTutMell li2111282005041517AM/ReadWriteXMLTutMellli21.aspx

[13] EVANS, Clark C. *The Official YAML Web Site* [online]. [cit. 2012-05-30]. Dostupné z: http://www.yaml.org/

[14] *What Is Windows Communication Foundation* [online]. [cit. 2012-05-30]. Dostupné z: http://msdn.microsoft.com/en-us/library/ms731082.aspx

[15] *Index of /MySQL/Downloads/Connector-Net* [online]. [cit. 2012-05-30]. Dostupné z: http://download.softagency.net/MySQL/Downloads/Connector-Net/

[16] SHARP, John. *Microsoft Visual C# 2010: krok za krokem*. 1. vyd. Brno: Computer Press, 2010. 696 s. ISBN 978-80-251-3147-3

[17] CHALUPA, Radek. *1001 tipů a triků: Visual C++*. 1. vyd. Brno: Computer Press, 2003. 434 s. ISBN 80-7226-842-2.

[18] *AppInit_DLLs in Windows 7 and Windows Server 2008 R2* [online]. [cit. 2012-05-30]. Dostupné z: http://msdn.microsoft.com/en-us/library/dd744762(v=VS.85).aspx