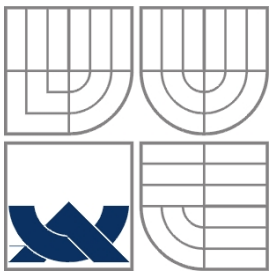
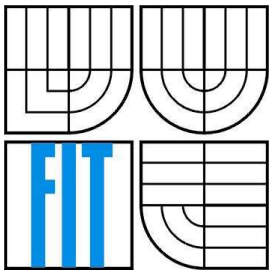


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

MONITORING A ANALÝZA UŽIVATELŮ SYSTÉMEM DLP

MONITORING AND ANALYSIS OF USERS USING DLP SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL PANDOŠČÁK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL DROZD

BRNO 2011

Abstrakt

Cílem této diplomové práce je obeznámit se s problematikou monitorování a analýzy uživatelů a procesů systémem DLP (Data Loss Prevention – prevence ztráty dat), definování možností útoků uvnitř a z vnějšku organizace, popis hlavních částí DLP systému, spravování podnikových politik, monitorování aktivit uživatelů a klasifikování obsahu informací. Práce vysvětluje rozdíl mezi kontextovou analýzou a analýzou obsahu dat a popisuje jejich jednotlivé techniky. Přibližuje monitorování sítě a koncových zařízení a popisuje možné chování uživatele a procesu při běžných činnostech, které mohou způsobit únik dat. V závěru jsme pomocí získaných informací vytvořili návrh a vyvinuli aplikaci endpoint agenta, který slouží k monitoringu aktivit procesu na koncové stanici.

Abstract

The purpose of this masters thesis is to study issues of monitoring and analysis of users using DLP (Data Loss Prevention) system, the definition of internal and external attacks, the description of the main parts of the DLP system, managing of politic, monitoring user activities and classifying the data content. This paper explains the difference between contextual and content analysis and describes their techniques. It shows the fundamentals of network and endpoint monitoring and describes the process and users activities which may cause a data leakage. Lastly, we have developed endpoint protection agent who serves to the monitoring activities at a terminal station.

Klíčové slova

prevence ztráty dat, kontextová analýza, analýza obsahu, monitorování, únik dat, minifiltr ovládač

Keywords

data loss prevention, context analysis, content analysis, monitoring, data leakage, minifilter driver

Citácia

Bc. Pandoščák Michal: Monitoring a analýza uživatelů systémem DLP, diplomová práce, Brno, FIT VUT v Brne, 2011

Monitoring a analýza užívateľov systémom DLP

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením Ing. Michala Drozda.
Ďalšie informácie mi poskytol Ing. Petr Javora.
Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

……
Michal Pandoščák
25.máj 2011

Poďakovanie

Na tomto mieste by som chcel poďakovať Ing. Michalovi Drozdovi, Ing. Petrovi Javorovi
a Ing. Ondrejovi Pandoščákovi za vedenie a poskytnutú pomoc.

Obsah

1 Úvod

Každá z dnešných organizácií disponuje množstvom citlivých informácií. Napriek času a peniazom vynaloženými na to, aby sa tieto informácie vytvorené zamestnancami, klientmi alebo inými obchodnými spolupracujúcimi dostali pod kontrolu, sa iba pár organizáciám podarilo zistiť, kde sú ich citlivé informácie uložené a ako sú zabezpečené. Iné sa spoliehajú na to, že ich dáta sú uložené na serveri, kde sa nedostane nik, iba zamestnanci, ktorí s nimi pracujú.

Organizácie neohrozujú iba útoky zvonku, ale aj vnútroorganizačné úniky, a to ani nie preto, že by si nevedeli ochrániť citlivé dáta na známych miestach, ale preto, že vedia iba veľmi málo o tom, kde sa nachádza ich obsah, ako je chránený a ako sa s ním zaobchádza.

Jedna z najsľubnejších techník, ktorá sa snaží takýmto organizáciám pomôcť znížiť riziko straty dát, sa nazýva Prevencia straty dát (*Data Loss Prevention - DLP*). DLP je produkt, ktorý na základe podnikových politík identifikuje, monitoruje a ochraňuje dáta v pokoji (*data at rest*), v pohybe (*data in motion*) a používané dáta (*data in use*), a to ich hĺbkovou analýzou obsahu.

Načo to všetko?

Predstavte si, že viete o každom umiestnení citlivých informácií a taktiež viete, kto má prístup k týmto dátam. Môžete sledovať celý úložný systém a dostávať informácie o tom, že sa citlivý obsah presunul na nechránené miesto, alebo iba o tom, že sa zmenili jeho prístupové práva. V niektorých prípadoch môžeme ochrániť obsah tým, že ho vložíme do karantény alebo zašifrujeme.

Tak ako sa každým dňom zvyšuje cena dát, takisto sa deň za dňom zvyšuje snaha ľudí o ich odcudzenie. Zvyčajne sa firmy dozvedajú o tom, že dáta boli ukradnuté až po tom, čo bol ich obsah odhalený verejnosti, napríklad na internete. Po zverejnení ukradnutých dát, už nie je žiadna možnosť ich znova utajiť, preto je ich ochrana v tomto štádiu nemožná a vedenie firmy nebude hlavne zaujímať, ako sa dáta stratili, ale to, prečo ich bezpečnostný technik už dávno nevyriešil tento problém.

Ak má už nejaká organizácia v dnešnej dobe implementovaný DLP systém, tak to je pravdepodobne dôsledkom jednej z týchto príčin:

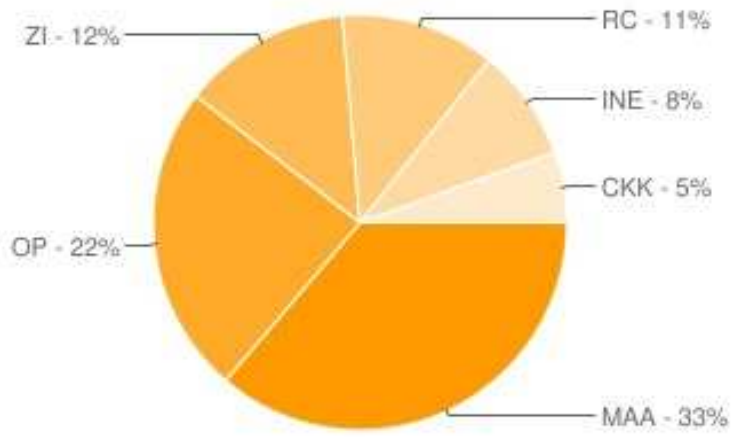
1. Prikazuje im to legislatíva.
2. Majú šikovného bezpečnostného správcu.
3. V ich organizácii sa už objavil únik dát.

2 Možnosti odcudzenia dát

2.1 Čo je únik dát?

Únik dát je neautorizovaný prenos dát alebo informácií z vnútra organizácie k vonkajšiemu príjemcovi. Prenos môže byť buď elektronický, alebo fyzický, pri ktorom napríklad zamestnanec vynesie vytlačené informácie. Na tomto mieste môžeme poznamenať, že neautorizovaný prenos nemusí ihneď znamenať, že zamestnanec konal úmyselne alebo zlomyseľne, ale aj takýto prenos informácií je obzvlášť nebezpečný. Podľa údajov z [1] až 73% všetkých únikov dát z vnútra organizácie je neúmyselných.

Typy dát, ktoré sa najčastejšie kradnú, sú: čísla občianskych preukazov (OP), mená a adresy (MAA), čísla kreditných kariet (CKK), rodné čísla (RC), zdravotné informácie (ZI) a iné (INE). Ich rozloženie si môžete pozrieť na obrázku 1 [1].



Obrázok 1: Pomer typov kradnutých dát za rok 2010 [1].

V ďalších kapitolách budeme používať pojem zamestnanec, pod týmto pojmom si predstavíme užívateľa s prístupom k citlivým podnikovým informáciám.

2.2 Úmyselné vnútroorganizačné úniky

Hoci predložené údaje ukazujú, že hlavnou hrozbou pre vnútroorganizačné úniky sú neúmyselné akcie, aj napriek tomu si organizácie musia dávať pozor na neoprávnené a úmyselné vynášanie dát vlastnými zamestnancami. Existuje mnoho spôsobov, ako dostať informácie von z organizácie, môžeme spomenúť najčastejšie, ako napríklad podnikový alebo webový email, IM, FTP, SSH, P2P[1], prenosné disky, USB[2] zariadenia, webové úložisko, vytlačené dáta a iné.

[1] Skratky IM, FTP, SSH, P2P budú vysvetlené v ďalších kapitolách.

[2] USB (Universal Serial Bus) je univerzálna sériová zbernica a zariadenia, ktoré ju používajú sa nazývajú USB zariadenia.

Dôvody pre vykonanie takejto činnosti môžu byť rôzne, ale ide najmä o priemyselnú špionáž, finančnú odmenu alebo najväčšiu hrozbu zamestnávateľa, ktorou je nespokojný zamestnanec.

Ako príklad môžeme uviesť nespokojného bezpečnostného technika, ktorého firma prepustí. Ešte skôr, ako sa mu stihnú odobrať všetky práva a zamedzí sa mu prístup k systému, zamestnanec si skopíruje všetky citlivé informácie a tie používa ako svoje know-how v ďalšej práci alebo ich po častiach predá konkurenčným firmám.

2.3 Neúmyselné vnútroorganizačné úniky

Ako sme už spomenuli, drvivá väčšina únikov dát je neúmyselná. Niektorí zamestnanci si ani sami neuvedomujú, že ich činy mali za následok únik informácií. V tejto oblasti sa majú organizácie ešte čo zdokonaľovať. Hlavným problémom je slabá počítačová zdatnosť zamestnancov. Riešením by nemalo byť zakazovanie rôznych stránok alebo nosičov dát. V tomto prípade by boli lepším riešením rôzne školenia o bezpečnosti a správe aplikácií.

Neúmyselným únikom je napríklad, ak zamestnanec pripojí ako prílohu emailu nesprávny súbor a ten následne odošle.

2.4 Metódy úniku dát z vnútra organizácie

V týchto metódach uvažujeme, že zamestnanec úmyselne alebo neúmyselne spôsobí únik citlivých firemných informácií. Na obrázku 2 si môžete prezrieť najčastejšie formy úniku, ktoré budú detailnejšie popísané v nasledujúcich kapitolách.

Obrázok 2: Metódy uniku dát zvnútra organizácie [3].

2.4.1 Podnikový email

Emailoví klienti, ako napríklad Mozilla Thunderbird, The Bat!, Microsoft Outlook atď., sú základom každej firemnej komunikácie. Zamestnanec s dostatočnou motiváciou môže poslať súbor s citlivými údajmi ako prílohu emailu niekomu neautorizovanému. Tento súbor môže dokonca zašifrovať alebo jeho obsah môže vložiť do iného súboru, aby tým zamaskoval jeho prítomnosť. Inou možnosťou by bolo skopírovať obsahu súboru a vložiť ho ako text do emailovej správy.

2.4.2 Webový email

Webový email používa skoro každý internetový užívateľ. Známe sú napríklad Gmail, Yahoo, Hotmail a pod. Aj tu sa objavuje príležitosť pre zamestnanca, ako odcudziť dáta. Rovnako ako u podnikového emailu môžeme pripojiť súbor ako prílohu správy alebo môžeme vložiť jeho obsah do tela správy a odoslať. Webový email komunikuje prostredníctvom HTTP alebo HTTPS[3], ktoré sú na portoch 80 a 443 a vo väčšine organizácií sú tieto porty povolené na firewalle a takto sa zamestnancom dovoľuje používanie ich služieb.

2.4.3 Škodlivé webové stránky

Webové stránky so škodlivým obsahom predstavujú riziko, že sa užívateľov počítač nakazí nejakým škodlivým softvérom, a to už aj otvorením danej stránky. Využije pri tom zistenú zraniteľnosť prehliadača, prostredníctvom ktorej sa zmocní počítača.

2.4.4 Instant Messaging

Veľa organizácií dovoľuje svojim zamestnancom používať Instant Messaging (IM) na ich firemných počítačoch. Do tejto kategórie spadajú aplikácie ako AOL, Jabber, Gtalk, ICQ, MSN Messenger, Skype a iné. Skoro s každým IM je možné posielať súbory alebo poslať obsah súboru ako správu inému užívateľovi. Prostredníctvom IM je možné na daný počítač nainštalovať škodlivý softvér takým spôsobom, že príde správa a v nej hypertextový odkaz na nejakú stránku so škodlivým obsahom. Počítač sa po nainštalovaní malwareu stáva bezmocným proti cielenému útoku zvonku.

2.4.5 Posielanie dát prostredníctvom FTP

File transfer protocol (FTP) je protokol na prenos súborov. V tomto prípade zamestnanec buď na svoj počítač nainštaluje FTP klienta a pomocou neho bude odcudzovať dáta, alebo ak má prístup k

[3] HTTP (Hyper Transfer Protocol) je internetový protokol určený pre posielanie hypertextových dokumentov. HTTPS je jeho bezpečnejšia verzia, ktorá umožňuje prenášané dáta šifrovať a tým ich chráni pred odpočúvaním a možným narušením.

serveru, tak na ňom môže nainštalovať a nastaviť si FTP server, aby mohol pokojne z domu pristupovať k informáciám, ktoré potrebuje.

2.4.6 SSL VPN

Secure socket layer virtual private network (SSL VPN) je technológia, pomocou ktorej si môžeme vytvoriť šifrovaný tunel medzi dvoma koncovými stanicami, a tak získať vzdialený prístup k firemným počítačom. Keďže je tento tunel šifrovaný, predstavuje menšiu hrozbu pre zamestnanca, že bude daný únik odhalený.

2.4.7 Zdieľanie dát na sieťach typu P2P

Peer to peer (P2P) je počítačová sieť, pomocou ktorej jej užívatelia zdieľajú dáta. Je to značná hrozba pre každú organizáciu, pretože akonáhle sa dostane súbor na sieť, môže sa veľmi rýchlo rozšíriť.

2.4.8 Nosiče dát

Pod pojmom nosič dát rozumieme externé disky, flash karty, CD a DVD. Všetky tieto veci sú v dnešnej dobe lacné a každý počítač na ne dokáže zapisovať a čítať z nich (pravdaže ak ma príslušné rozhranie). Maximálna veľkosť USB kľúča na portáli alza.cz je 256 GB. Na takýto kľúč by sme boli schopní uložiť státisíce kancelárskych dokumentov, skopírovaných databáz a pod.

Zamestnancovi potom stačí iba pripojiť kľúč a za pár dní môže mať doma všetky dostupné firemné dáta. Rovnakú hrozbu predstavujú aj rôzne prehrávače, ktoré taktiež môžu mať veľkosť aj niekoľko desiatok GB, a predsa zamestnávateľ nevyhodí zamestnanca za to, že si do práce priniesol svoj súkromný prehrávač.

Fyzické rozmery USB kľúčov majú za následok mnoho neúmyselných únikov dát. Príkladom je Slovenská armáda, ktorej zamestnancom sa podarilo v roku 2010 stratiť trikrát USB kľúč s citlivými informáciami. Americká armáda zakázala používanie USB kľúčov, z dôvodu šírenia vírusov a červov v jej sieťach.

2.4.9 Nedostatočné alebo nesprávne nastavenie práv súborov

V prípade, že prístupové práva k zložkám alebo ich súborom sú zle nastavené, naskytuje sa príležitosť, aby zamestnanec iba skopíroval súbory z nejakého sieťového disku na svoj počítač. Následne by mohol postupovať vyššie spomenutou metódou.

2.4.10 Straty a nálezy

Straty notebookov, USB kľúčov či firemných publikácii nie sú v dnešnej dobe zhonu ničím neobvyklým. Horšie je, keď stratená vec obsahovala tajné informácie, ktoré by boli prospešné

konkurenčnej firme. V tomto prípade najlepší spôsob, ako si ochrániť stratené dáta, je šifrovanie celého disku.

2.4.11 Fyzické vynášanie informácií

Ak už organizácia zaviedla rôzne bezpečnostné opatrenia, aby zamestnanci nemohli odcudzovať citlivé dáta nahrávaním vecí na USB zariadenia, vypaľovaním na CD nosiče alebo posielaním súborov po sieti, nastáva situácia, keď zamestnanci začínajú fyzicky vynášať veci (disky, tlačené dokumenty, počítače a pod.). Je jednoduché si vložiť do tašky pár vytlačených alebo oskenovaných strán projektovej dokumentácie.

2.4.12 Iné úniky

Existuje pár spôsobov, ktoré prakticky umožňujú ukradnúť dáta bez toho, aby v systéme ostala nejaká zmienka o modifikácii či presune dokumentu. Jedným z nich je, ak zamestnanec použije fotoaparát, ktorý je v dnešnej dobe súčasťou skoro každého mobilného zariadenia a pomocou neho si vyfotí daný dokument stranu po strane. Pri absencii fotoaparátu si zamestnanec prahnúci po informáciách vystačí isto aj s perom a papierom.

Ďalšou možnosťou je používanie Copy-Paste[4] a Print Screen[5] citlivých informácií. Tie si môžu zamestnanci ukladať do nezabezpečených súborov a takto ich vykrádať.

2.5 Metódy úniku dát zvonka organizácie

Do kategórie metód úniku dát zvonka organizácie patria krádeže fyzických vecí, škodlivý softvér, preberanie odpadkov, sociálne inžinierstvo a tiež ak sa útočník pokúša ukradnúť dáta prostredníctvom nejakej chyby v systéme. Nepredpokladáme, že útočník je zamestnancom firmy.

2.5.1 Krádež fyzických vecí

Podľa údajov z [1] sa v roku 2010 v USA ukradlo vyše sedem miliónov záznamov o ľuďoch, ktoré obsahovali čísla ich kreditných kariet, občianskych preukazov, zdravotné záznamy alebo iné citlivé údaje. Informácie o týchto krádežiach boli zverejnené, ale kto vie, koľko je ďalších nezverejnených prípadov a koľko ich je inde po celom svete. Kradnú sa najmä notebooky, pevné disky, CD a DVD nosiče, tlačené dokumenty a záložné pásky.

[4] Copy-Paste metóda je označenie presúvania obsahu jedného súboru do druhého.

[5] Print Screen označuje ukladanie aktuálneho vzhľadu obrazovky ako obrázku.

2.5.2 Škodlivý softvér

Škodlivý softvér je softvér, ktorý sa snaží odcudziť citlivé údaje, vykonávať rôzne podvody alebo posielať nevyžiadanú poštu z napadnutého počítača.

Do tejto skupiny spadajú všetky typy infiltrácií ako trójske kone, vírusy, spyware, adware, červy, rootkity a podobne.

Spoločnosť BitDefender nedávno varovala pred nebezpečenstvom nového špionážneho trójskeho koňa s označením Trojan.Spy.YEK. Tento kôň je popisovaný ako vážny nepriateľ, ktorý môže byť použitý pre potreby priemyselnej špionáže. Obsahuje funkcie backdoor a jeho činnosť spočíva vo vyhľadávaní kritických dát a cenných súborov, ktoré obsahujú citlivé informácie (hlavne firemné). S útočníkom komunikuje pomocou prehliadača Internet Explorer tak, že najprv ho injektuje a potom posiela informácie o celom systéme. Firewall si myslí, že ide o bežnú komunikáciu na porte 80. Tento odsek bol prevzatý z [2].

2.5.3 Útočník

Útočník je človek, ktorý má detailné znalosti o počítačoch, fungovaní systému, softvéri a programovaní. Využíva získané informácie o slabo zabezpečených miestach alebo bezpečnostných medzerách ku kriminálnym účelom alebo pre osobný prospech.

Jednou z najväčších krádeži bola krádež 130 miliónov čísel kreditných kariet zo spoločnosti Heartland Payment Systems, ktorá je piatou najväčšou spoločnosťou na spracúvanie platieb [1].

2.5.4 Preberanie odpadkov

Áno, aj to je spôsob, ako spoločnosť môže prísť o citlivé informácie. Stáva sa to pri vyhadzovaní tlačených informácií, ktoré sa správne neskartujú alebo inak nezničia. Tu nastáva riziko, že sa môžu dostať k niekomu, kto by tieto informácie vedel zneužiť. Podobné prípady môžu nastať pri CD, DVD nosičoch alebo dokonca počítačoch, ktoré spoločnosti predávajú.

2.5.5 Sociálne inžinierstvo

Sociálne inžinierstvo je spôsob manipulovania s ľuďmi za účelom prevedenia nejakej akcie alebo získania určitej informácie. Všetky techniky sociálneho inžinierstva sú založené na špecifických spôsoboch ľudského rozhodovania známych ako kognitívne chyby úsudku. Tieto chyby úsudku sú založené na nedokonalosti ľudského mozgu [3].

Najznámejšie sú:

- Pretexting je vytváranie a udržovanie vymysleného príbehu s cieľom presvedčiť obeť k výkonu nejakej operácie alebo k získaniu potrebnej informácie. Útočník pritom spája klamlivý príbeh a kúsok pravdivej informácie, ktorú sa mu pred tým podarilo získať.

Útok pretextingom je prevádzaný prostredníctvom telefónu, Instant Message alebo inou možnou komunikáciou v organizácii.

- Phishing je činnosť, pri ktorej sa útočník snaží získať citlivé údaje. Jeho prvým krokom je vytvorenie presnej kópie nejakého portálu, kde sa užívateľ prihlasuje. Ďalším krokom je rozposlanie emailov s podobným obsahom, ako "Pre zvýšenie ochrany si prosím Vás pravidelne obmieňajte heslo.", v tele správy sa bude nachádzať aj prípadný odkaz na daný portál. Ak sa užívateľ prihlási, útočník získa jeho meno a heslo.

3 Popis DLP

Prevencia straty dát (*Data loss prevention - DLP*) je súbor bezpečnostných technológií slúžiaci k ochrane pred chybami, ako je neúmyselný únik dát spôsobený zamestnancami firmy. DLP taktiež slúži pri ochrane pred úmyselnými únikmi dát spôsobených zamestnancami, ktorí si chcú privlastniť dáta pre ich osobný zisk. V neposlednej rade sa DLP zaoberá aj únikmi spôsobenými vonkajšími útokmi. V skratke môžeme DLP označiť ako dozorcu, ktorý sa stará o uplatňovanie a dodržiavanie firemných politík.

Pojem prevencia straty dát je niekedy nahradzovaný pojmami prevencia a detekcia úniku informácií (*Information Leak Detection and Prevention – ILDP* [8]), prevencia úniku dát (*Information Leak Prevention – ILP* [9]) alebo monitoring a filtrovanie obsahu (*Content Monitoring and filtering – CMF* [10]).

3.1 Časti DLP

Systém DLP môžeme rozdeliť podľa viacerých kritérií. Z môjho pohľadu je hlavné rozdelenie podľa jeho životného cyklu. Tu patrí vytváranie politík, klasifikovanie dát, monitorovanie dát a tréning zamestnancov. Pri pravidelnom sledovaní a upravovaní týchto častí sa zvyšuje úspešnosť prevencie straty dát.

3.1.1 Politiky, štandardy a procedúry

Firemné politiky, štandardy a procedúry sú základom pre efektívnu DLP stratégiu. Zaručujú, že firemné dáta spadajú pod určitý stupeň ochrany primeraný ich dôležitosti. Je dôležité, aby sa tieto prvky pravidelne upravovali, a tak stále zvyšovali bezpečnosť dát.

Vytvorenie politík je štartovacím bodom pred tým, ako si spoločnosť vytvorí štandardy a procedúry, ktoré dovoľujú zvýšiť bezpečnosť a efektivitu ich firemného DLP riešenia. Štandardy sú akcie, pravidlá a predpisy vyvinuté na to, aby bolo používanie politík zrozumiteľnejšie a zmysluplnejšie. Procedúry objasňujú špecifikácie toho, ako politiky a štandardy budú skutočne implementované v prevádzkovom prostredí.

3.1.2 Klasifikácia dát

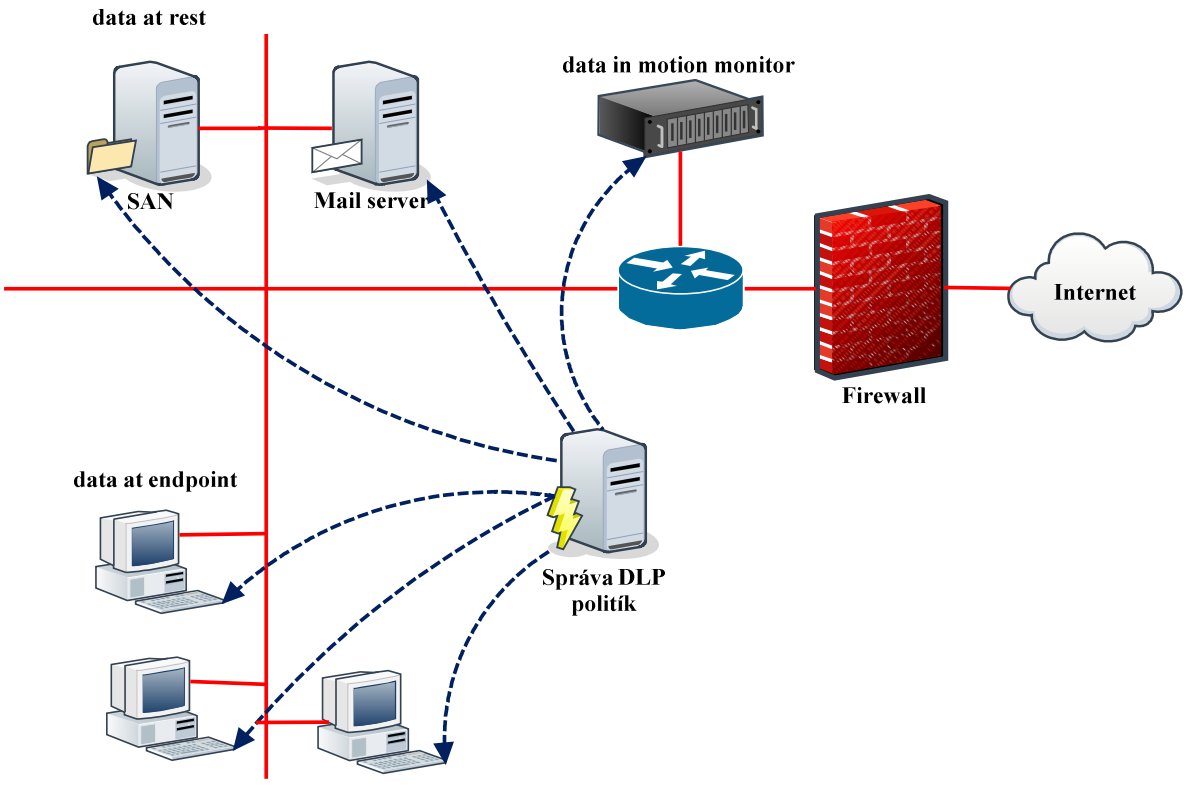
Klasifikácia dát je proces triedenia obsahu informácií na základe ich ceny a citlivosti s ohľadom na danú organizáciu. Klasifikáciou určujeme vhodnú výšku priority pre majetok a zdroje organizácie, čo nám v konečnom dôsledku pomôže určiť stupeň ochrany pre daný systém, na ktorom sa dáta nachádzajú.

Dáta by mali byť kategorizované podľa potreby utajenia[6], a to na verejné, dôverné, tajné či privátne. Po klasifikácii dát by malo nasledovať skenovanie všetkých dátových úložísk, aby sme sa dozvedeli, kde sa nachádzajú naše citlivé dáta.

3.1.3 Monitoring dát

Jednou z hlavných súčastí DLP systému je monitoring dát. Tento monitoring prehľadáva obsah dát a na základe dopredu definovaných politík je schopný aplikovať potrebnú procedúru (vytvoriť log alebo správu, premiestniť, zašifrovať, zabrániť užívateľovi prístup).

Dáta rozdeľujeme do troch skupín, a to na práve používané dáta (*data in use*), dáta uložené na serveroch (*data at rest*) a prenášané dáta (*data in motion*).



Obrázok 3: Rozdelenie dát na typy podľa systému DLP.

Data at rest je pojem označujúci dáta, ktoré sú uložené niekde na úložisku. Tento pojem nezahŕňa dáta, ktoré sú pravidelne sťahované zo serverov alebo sú tam umiestnené iba dočasne. Data at rest sú dáta, ktoré sú archívmi alebo referenciami súborov, ktoré sa menia zriedka alebo skoro nikdy. Ako príklad môžeme uviesť súbory uložené na externých zálohovacích diskoch alebo súbory uložené na SAN[7]. Dáta tohto typu sú hlavným cieľom útočníkov.

[6] V iných prípadoch sa kategórie môžu vytvárať podľa potrieb firmy.

[7] SAN (Storage Area Nerwork) je dedikovaná dátová sieť, ktorá slúži pre pripojenie externých zariadení k servrom.

Data in motion je pojem označujúci dáta všetkého druhu, ktoré sa premiestňujú počítačovou sieťou z jedného miesta na druhé. Do tejto kategórie by sme mohli zaradiť napríklad informácie posielané prostredníctvom emailu (ako príloha alebo obsah), FTP alebo P2P.

Data in use sú všetky dáta, ktoré nepatria do data at rest, čiže dáta, s ktorými sa aktívne pracuje alebo manipuluje. Do tejto skupiny patria dáta, ktoré otvárame, tlačíme, ukladáme, mažeme. Ak sú tieto dáta na koncovej stanici (notebook, desktop), potom ich označujeme ako dáta na koncovej stanici (*data at endpoint*). Vtedy k nim zahŕňame všetky dáta na koncovej stanici a dáta na USB, PDA alebo iných vymeniteľných médiách.

DLP systém by sme mohli rozdeliť z hľadiska monitorovania na dve časti. Prvá časť sa zaoberá monitorovaním dát prechádzajúcich sieťou (*data in motion*) a druhá sa zaoberá monitorovaním dát na úložných miestach (*data at rest, data at endpoint*).

3.1.4 Tréning zamestnancov

Aby sme dosiahli efektívnosti DLP riešenia, musíme spolupracovať so zamestnancami firmy. Zamestnanci musia pochopiť, prečo niektoré ich aktivity sú neprijateľné a mohli by uškodiť spoločnosti.

Veď predsa nie všetky priestupky voči politikám sú vykonané so zlým úmyslom. Napríklad ak chce zamestnanec pracovať doma a pošle si email na jeho súkromnú a možno aj slabšie zabezpečenú schránku. Hoci jeho úmysel bol dobrý, napriek tomu jeho akcia nebola. Priebežný tréning prispeje k uvedomeniu si svojho správania a poskytne zamestnancom informácie, ako správne zaobchádzať s citlivými údajmi.

3.2 Obsah dát vs. kontext dát

Skôr ako začneme monitorovať dáta musíme si ujasniť, či budeme monitorovať, kontext alebo obsah dát. Jednou z hlavných schopností DLP systému je uvedomovanie si obsahu dát. Pomocou tejto schopnosti hĺbkovo analyzuje dáta za použitia rôznych techník a veľmi sa odlišuje od analýzy kontextu. Predstavme si obsah ako list a kontext ako obálku. V kontexte je obsiahnutý odosielateľ, adresát, hlavička a podobne. Obsah predstavuje vnútro obálky.

Uvedomenie si obsahu zahŕňa nahliadanie do kontajnerov (paketov alebo súborov) a analýzu ich obsahu. Veľkou výhodou používania kontextovej analýzy je to, že nie sme na ňu striktne viazaní. Ak chceme ochraňovať hoci aj kúsok dát, tak to musíme byť schopní spraviť všade a nie iba vo vnútri citlivých kontajnerov. Takto ochraňujeme dáta a nie obálku, v ktorej sa nachádzajú.

3.2.1 Kontextová analýza

Na začiatku vývoja DLP bola kontextová analýza veľmi jednoduchá, zvyčajne sa kontrolovala iba hlavička emailu alebo metadáta súboru. Odvtedy ku kontextovej analýze pribudli dôležité vyhodnocovacie faktory ako:

- Vlastníctvo a práva súboru.
- Používanie šifrovaných formátov súborov alebo protokolov.
- Úlohy užívateľa a podniku.
- Špecifické webové služby, ako webmail alebo sociálne siete.
- Webové adresy.
- Informácie o USB zariadeniach, ako výrobca alebo model.

3.2.2 Analýza obsahu

V prvom kroku kontextová analýza zachytí obálku, otvorí ju, rozpársuje kontext a nahliadne do nej. V prípade emailu, ak sa v ňom nachádza iba čistý text, je to jednoduché, ale ak je jeho obsahom binárny súbor, tak sa jeho obsah stáva zložitejším na pochopenie. DLP systém používa na vyriešenie tohto problému tzv. *cracking* súborov.

Cracking súborov je technológia, ktorá sa používa na čítanie a pochopenie súboru, hoci je jeho obsah uložený pod viacerými vrstvami. Napríklad, ak je tabuľka z Excelu vložená do dokumentu Word a ten je zozipovaný. Cracker musí najprv rozbaliť súbor, prečítať dokument Word, analyzovať ho, nájsť Excel tabuľku, tú prečítať a zanalyzovať ju. Iné riešenia môžu byť ešte komplexnejšie, ako napríklad súbor CAD[8] zabudovaný v .pdf súbore.

V prípade, keď sa už nájde obsah, tak na jeho analýzu môžeme použiť techniky spomínané v nasledujúcich podkapitolách, aby sme boli schopní odhaliť prípadné porušenie firemných politik. Každá z nich má svoje slabé a silné stránky.

3.2.3 Regulárne výrazy

Regulárne výrazy sú najčastejšie používaná technika DLP systému na analýzu obsahu. Analyzuje obsah pomocou špecifických pravidiel, ako napríklad, že 16-ciferné číslo zodpovedá popisu čísla kreditnej karty. Vo väčšine DLP riešení sa tieto základné regulárne výrazy rozširujú o prídavnú analýzu. Príkladom môže byť, ak sa meno nachádza v blízkosti adresy a tá zas v blízkosti čísla kreditnej karty.

Najčastejšie sa používa ako prvý filter, pomocou ktorého sme schopní ľahko detekovať časti štruktúrovaných dát, ako napríklad čísla občianskych preukazov alebo platobných kariet. Silnou

[8] Computer-aided design CAD súbor obsahuje 2D alebo 3D model.

stránkou tejto techniky je, že sa spravidla dajú vytvoriť veľmi rýchlo, následne sa jednoducho upravujú a sú ľahko pochopiteľné.

Slabou stránkou tejto techniky je náchylnosť k chybnému vyhodnoteniu. Navyše poskytuje veľmi malú ochranu pri neštruktúrovanom obsahu.

3.2.4 Databáza odtlačkov

Niekedy sa nazýva aj *Exact Data Matching*, je technika, pomocou ktorej porovnávame obsah s presným popisom citlivých dát z databázy. Ako príklad môžeme uviesť politiku, ktorá sa zameria iba na čísla kreditných kariet klientov a bude ignorovať zamestnancov, ktorí budú môcť napríklad nakupovať cez internet.

Táto technika sa využíva pri analyzovaní štruktúrovaných dát. Jej výhodou je to, že jej chybovosť označenia sa blíži k nule. Dovoľuje nám ochrániť citlivé dáta a pritom ignoruje iné alebo podobné dáta. Jej hlavnou nevýhodou je jej obsiahla databáza vzoriek.

3.2.5 Presné zhodovanie súborov

Pri tejto technike sa z každého citlivého súboru spraví hash a monitoruje sa, či nejaký súbor odpovedá danému odtlačku. Niekto považuje túto techniku za kontextovú analýzu, keďže obsah samotného súboru nie je analyzovaný.

Táto technika je vhodná pri analyzovaní masmediálnych alebo iných binárnych súborov, kde by analýza textu nebola úspešná. Sú to súbory ako fotky, video a audio nahrávky a iné. Výhodou tejto technológie je, že ju môžeme použiť na hocijaký typ súboru a jej chybovosť bude nízka za predpokladu dostatočne veľkého hash.

Nevýhodou je, že sa jej dá ľahko vyhnúť a to zmenením alebo pridaním pár bitov súboru. Taktiež nie je vhodná na súbory, ktoré sa často upravujú, ako dokumenty alebo editované masmediálne súbory.

3.2.6 Čiastočné zhodovanie súborov

Táto technika vyhľadáva časť alebo celý obsah chráneného obsahu. Mohli by sme vytvoriť politiku, ktorá bude ochraňovať citlivý dokument a DLP dohliadne na to, či sa niekde tento obsah objaví, buď to ako celok, alebo pár viet z neho. Napríklad by sme mohli vytvoriť novú politiku, ktorá bude ochraňovať dokumentáciu nového produktu. DLP oznámi každé porušenie tejto politiky aj keby sa zamestnanec pokúšal poslať (napr. pomocou IM) hoci iba odsek z nej.

Väčšina riešení je založená na cyklickom (alebo prekrývajúcom sa) hashovaní, pri ktorom sa vytvorí hash z časti obsahu, ofset a preddefinovaný počet znakov, a potom ďalší a ďalší až do konca dokumentu. Analyzovaný dokument prechádza rovnakým postupom a kontroluje, či sa zhoduje hash.

Túto techniku používame pri ochrane textových dokumentov, ako sú napríklad zdrojové kódy. Výhodou tejto techniky je, že dokáže ochrániť neštruktúrované dáta a má nízku chybovosť. Niekto by sa mohol odvážiť povedať, že má nulovú chybovosť, ale za predpokladu, že kontrolujeme vetu po vete, tak každá bežná veta by mohla vyvolať výstrahu porušenia politiky. Ďalšou výhodou je, že nepotrebuje analyzovať celý obsah veľkého dokumentu, ale postačí jej iba jeho časť, aby odhalila priestupok.

3.2.7 Statická analýza

Statická analýza využíva strojové učenie a iné statické techniky na analýzu tela obsahu a zisťovanie porušenia politík v obsahu, ktorý zodpovedá chránenému obsahu. Do tejto skupiny spadajú rôzne techniky statickej analýzy, ktoré sa líšia v spôsobe implementácie a vo výkonnosti. Niektoré sa podobajú na techniky používane pri blokovaní spamu.

Používa sa na analýzu neštruktúrovaného obsahu, kde použitie deterministickej techniky, ako napríklad techniky čiastočného zhodovania, by bolo neefektívne. Dokáže pracovať aj s nejasným obsahom, kde nemusíme byť schopní izolovať presné podklady k vytvoreniu politiky. Jeho značnou nevýhodou je veľká chybovosť.

3.2.8 Slovníky a kategorizovanie

Slovníková technika používa slovníky, ktoré obsahujú zoznam slov a fráz špecifických pre danú organizáciu. Slovníkom môže byť napríklad XML[9] dokument, ktorý obsahuje daný zoznam napríklad pre finančné alebo zdravotnícke organizácie. Pri prehľadávaní obsahu sa porovnávajú jednotlivé slová a ak sa zhodujú, vyvolá sa chyba porušenia politík.

Kategorizovanie je technika, ktorá používa prednastavené kategórie pravidiel a slovníkov pre bežné typy citlivých dát. Výhodou je jednoduchosť konfigurácie. Kategórie politík môžu tvoriť základ pre väčšie a sofistikovanejšie firemné politiky.

3.3 Spôsoby monitorovania

Ako sme už spomenuli v kapitole 3.1.3, poznáme tri typy dát – Data in motion, data at rest a data at endpoint. Hlavnými časťami DLP systému v oblasti monitorovania je monitorovanie siete a monitorovanie koncových zariadení. V ďalších kapitolách si ich popíšeme.

3.3.1 Monitorovanie siete

Väčšina organizácií ako prvý produkt z rady DLP nástrojov implementuje práve monitorovanie siete, ktoré im poskytuje ochranu spravovaných aj nespravovaných systémov.

[9] XML (Extensible Markup Language – rozšíriteľný značkovací jazyk) slúži k štrukturalizácii dát.

Srdcom väčšiny DLP systémov je pasívny monitor siete. Tento monitor sa typicky umiestňuje v blízkosti *gateway*[10] alebo SPAN[11]. Zachytáva pakety, zoraďuje ich a vykonáva analýzu obsahu v reálnom čase. Hlavnou požiadavkou užívateľov je, aby tento monitor neobmedzoval priepustnosť ich siete. Niektoré riešenia DLP používajú radšej monitor na sledovanie preddefinovaných kombinácií portov a protokolov, ako na sledovanie každej komunikácie iba na základe jej obsahu. Na vedomie musíme brať aj fakt, že výkonnosť je nepriamo úmerná počtu vytvorených politik. V prípade, že vytvoríme väčšie množstvo politík založených na čiastočnom porovnávaní dokumentov alebo databáze odtlačkov (obe sú založené na porovnávaní s uloženým hashom), určite sa stane, že výkon systému nám bude klesať, a preto potrebujeme vybalansovať alebo rozdeliť záťaž na viac tokov. Príkladom toho môže byť presunutie emailovej komunikácie na server, ktorého politiky sú priamo nastavené na problematiku emailov.

DLP systém je navrhovaný tak, aby blokoval komunikáciu. Samotné blokovanie nie je najjednoduchšie, a to hlavne z toho dôvodu, že potrebujeme povoliť autorizovanú komunikáciu a zablokovať neautorizovanú a to všetko musí byť vykonané v reálnom čase. Preto musíme filtrovať pomocou mostu, proxy alebo *TCP*[12] *poisoning*.

Most je zariadenie s dvoma sieťovými kartami, ktoré vykonáva analýzu obsahu za behu. Ak sa mu niečo v komunikácii nebude páčiť, preruší ju. Spojovanie pomocou mostu nie je najvhodnejšie riešenie, pretože nemusí zastaviť každý únik hneď na jeho začiatku.

Proxy je špeciálne určený pre určitú aplikáciu alebo protokol. Pred tým, ako pošle ďalej komunikáciu, ukladá si ju do vyrovnávacej pamäte, kde je možné vykonať hlbšiu analýzu jej obsahu. Ak zariadenie dokáže revervovať SSL[13], potom môže taktiež nazerať do šifrovanej komunikácie (napr. Gmail, Hotmail, Facebook a pod.).

TCP poisoning je metóda, pri ktorej sa monitoruje komunikácia a ak odhalí niečo nepovolené, injektujeme TCP reset paket, ktorý zruší danú komunikáciu. Toto funguje na každom protokole, ale výsledok nie je príliš efektívny. Niektoré protokoly sa aj napriek resetu paketu budú snažiť obnoviť komunikáciu.

3.3.2 Monitorovanie koncových zariadení

Ako sme sa už zmienili implementácia DLP systému sa začína na sieti, a to z toho dôvodu, aby sme rýchlo a efektívne pokryli všetky zariadenia v danej sieti. Toto riešenie, ale nie je úplné. Neochraňuje dáta v prípade, ak si niekto odnesie notebook, ani keď si niekto nakopíruje dáta na prenosný disk.

[10] Gateway je uzol, ktorý spojuje dve siete.

[11] SPAN (Switched Port Analyzer) je zapojený medzi dvoma alebo viacerými prvkami siete a analyzuje ním prechádzajúci tok sieťou.

[12] TCP (Transmission Control Protocol) je internetový protokol.

[13] SSL (Secure Socket Layer) je kryptografický protokol. Používa sa v aplikáciách rôzneho druhu.

Aby sme sa presunuli od prevencie sieťového úniku ku kompletnej ochrane obsahu, musíme implementovať ochranu aj na koncové zariadenia (desktop, notebook, SAN). Daný softvér nazývame endpoint agent.

Pridanie endpoint agenta do DLP riešení nám poskytuje nielen schopnosť prehľadávať lokálne uložené dáta, ale aj schopnosť uchrániť systém v prípade, že už nie je pripojený k sieti. Agenti pri monitorovaní a ochrane dát sú obmedzení vlastnosťami daného stroja. V prípade, že máme vytvorenú politiku „Ochraňuj všetkých desať miliónov čísel kreditných kariet z databázy“, môžeme ju nahradiť jednoduchšou politikou ako napríklad „Ochráň všetky čísla kreditných kariet“.

Hlavnými vlastnosťami sú:

- Vyhľadávanie obsahu – prehľadávanie uložených dát na možné porušenie firemných politík.
- Ochrana súborového systému – monitorovanie a posilnenie operácií so súborom. Toto sa používa pri zabránení zápisu obsahu na USB zariadenie. Obsahuje aj nástroje na dešifrovanie alebo aplikáciu DRM[14].
- Ochrana siete – monitoruje a posilňuje sieťové operácie. V prípade, že sa koncové zariadenie nachádza mimo firemnej siete, môže endpoint agent poskytnúť podobnú ochranu ako gateway DLP. Na väčšine koncových zariadení sa stará o tlač a faxovanie prostredníctvom siete.
- Ochrana GUI[15]/jadra – obecnejšia kategória, ktorá obsahuje rôzne scenáre, ako napríklad Copy-Paste, Print Screean a pod.

[14] DRM (Digital Rights Management) je pojem označujúci technické metódy, ktorých účelom je kontrolovanie a obmedzovanie používania obsahu digitálnych médií.

[15] GUI (Graphical User Interface) v preklade grafické užívateľské rozhranie napr. okno aplikácie.

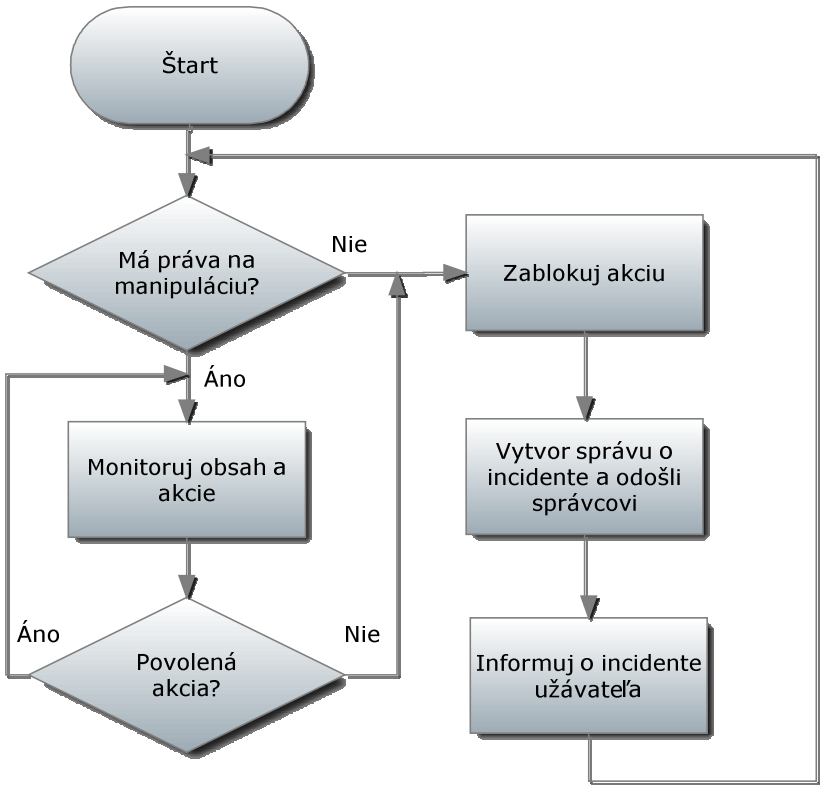
4 Popis správania užívateľov a procesov

V tejto kapitole popíšeme pár modelových situácií správania užívateľov. Ukážeme na nich, aké incidenty môžu spôsobiť a ako by sa k danému incidentu mal postaviť DLP systém. Predpokladáme bežný operačný systém, na ktorom beží endpoint agent.

4.1 Správanie užívateľa

Ako prvý modelový prípad si uvedieme správanie úradníka, ktorého hlavnou pracovnou náplňou je vytváranie a upravovanie dokumentov. Tieto dokumenty majú citlivý obsah (zdravotné záznamy, výplatné pásky a pod.) a DLP systém má nastavenú politiku tak, aby tento obsah chránil pred vložením do IM správy alebo na web.

Úradník počas pracovnej doby komunikuje prostredníctvom sociálnej siete so svojimi známymi a popri tom pomocou Copy-Paste presúva informácie medzi dokumentmi alebo databázami. Pri chvíľke nepozornosti vloží obsah schránky[16] do nesprávneho textového poľa a odošle. Informácie sú zverejnené na internete. V prípade, že sa tieto informácie niekde nakešujú, sú nenávratne prezradené.



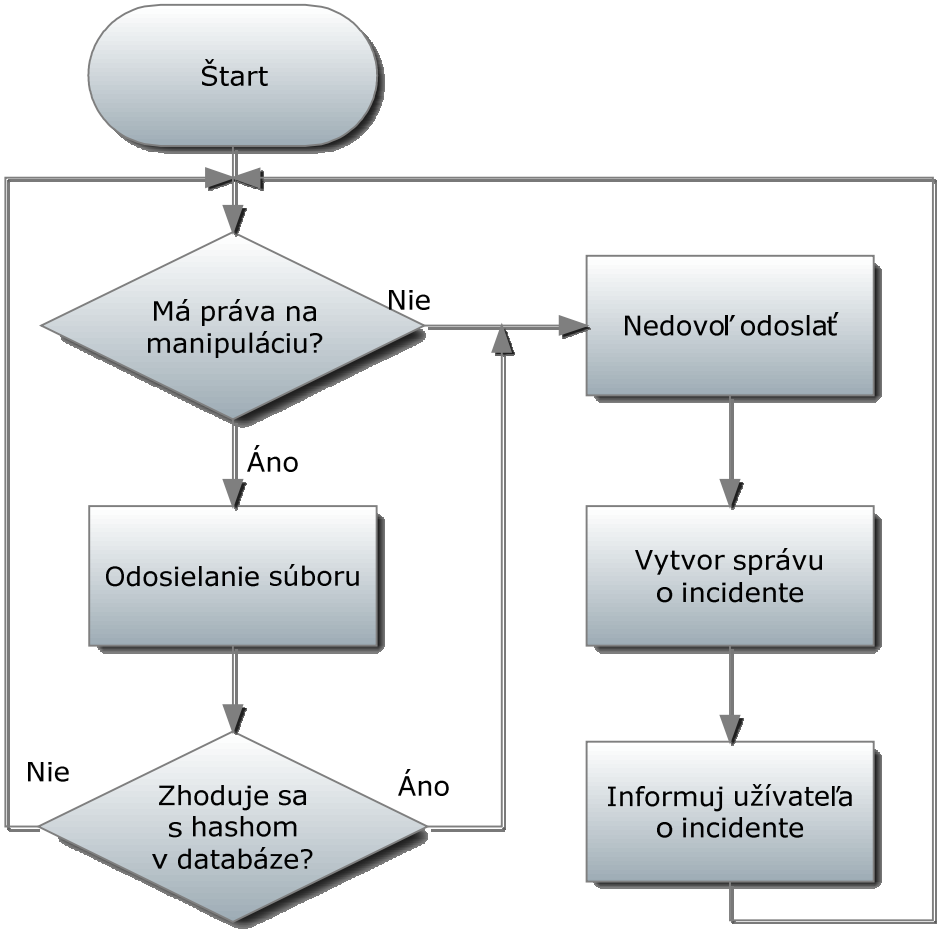
Obrázok 4: Schéma monitoringu.

[16] Schránka je spravovaná operačným systémom a je určená k prechodnému uchovávaniu rozličných dát.

Takýto záver DLP systém nedovolí. Ten aktívne monitoruje obsah schránky a miesto, kam sa užívateľ snaží vložiť jej obsah. Monitor zistí, že sa presúvajú citlivé dáta na nepovolené umiestnenie, odhalí porušenie firemných politík a nedovolí užívateľovi vykonať danú akciu. Vytvorí správu o priestupku a odošle ju správcovi DLP. V poslednom kroku informuje užívateľa, že spôsobil možný únik citlivého obsahu.

4.2 Správanie procesu

Za proces v tomto prípade považujeme škodlivý softvér, ktorý má v úmysle vykonať nejaký únik dát. Predpokladajme situáciu, kde si užívateľ nastaví politiku k zložke, ktorá obsahuje jeho privátne fotky. DLP systém si vytvorí z každej fotky v danej zložke hash a uloží si ho do databázy odtlačkov. Politika zakazuje posielanie súborov prostredníctvom webu.



Obrázok 5: Diagram chovania procesu.

V tejto modelovej situácií uvažujme, že operačný systém obsahuje trójskeho koňa s bežnou funkcionalitou popísanou v kapitole 2.5.2. Trójan sa pokúsi odoslať získané dáta. Operačný systém nevidí rozdiel medzi trójskym koňom a bežným užívateľom, pretože trójsky kôň vykonáva svoje

akcie s právami užívateľa. Ak má užívateľ povolenú manipuláciu so súborom, ako napríklad otváranie a kopírovanie, potom to všetko má povolené aj trójsky kôň. Ten môže prekopírovať fotky na iné nechránené miesto na disku, a potom sa ich pokúsi poslať von z počítača. Táto komunikácia vyvolá porušenie zadefinovaných politík, lebo pri porovnávaní posielaného súboru sa zhodoval jeho hash s iným v databáze. Následne vytvorí správu o incidente a oboznámi užívateľa o danom porušení.

Táto ochrana upozorňuje na zero-day[17] útoky alebo na nezabezpečenú chybu systému.

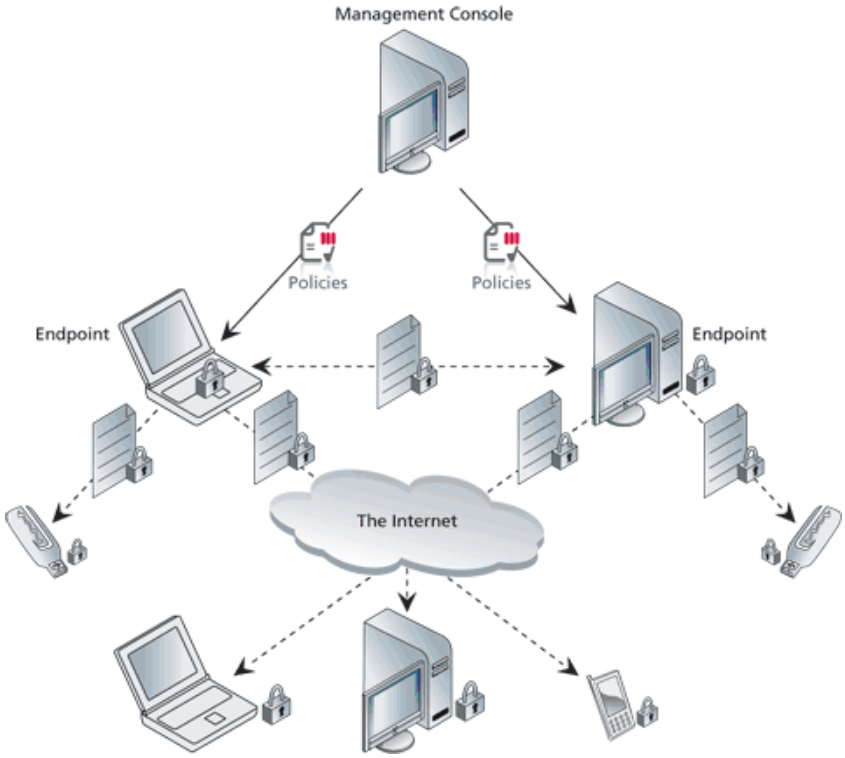
[17] Zero-day útok je označenie útoku, ktorý sa snaží v počítači využiť chybu softvéru, ktorá ešte nie je všeobecne známa.

5 Možnosti monitoringu užívateľov a procesov

V súčasnosti existuje na trhu viacero produktov na monitoring užívateľov systémom DLP od výrobcov ako Symantec [33], McAfee [34], Websense [35], RSA [36], Trustwave [37] a pod. Ich produkty sa líšia nielen v spôsobe monitoringu, ale aj v tom, aké aktivity monitorujú.

Väčšinou sa DLP systém skladá z agentského softvéru a správcovského serveru. Práve pomocou serveru môžu všetky zodpovedné osoby centrálne uplatňovať bezpečnostnú politiku v praxi. Ochrana zo serveru putuje na koncové stanice podnikovej siete, jednotlivé stanice sú nepretržite monitorované a na nich prebiehajúca manipulácia s dátami. Ich prenos alebo používanie podlieha obmedzeniam prednastaveným v bezpečnostnej politike. Je potrebné vybrať tie dáta, ktoré si vyžadujú zvláštnu ochranu a pozornosť. Vybrať môžeme celé stanice alebo iba jednotlivé priečinky adresára, súbory alebo iba zložky určitého typu a ako posledné im nastavíme príslušné práva. Každý takto vybraný objekt získa svoje vlastné označenie. Softvérový agent pri hocijakej následnej manipulácii so súborom na konkrétnej stanici v sieti kontroluje toto označenie a následne umožní užívateľovi iba také akcie, ktoré spoločnosť vopred definovala pre daný súbor a užívateľa.

Ak sa užívateľ pokúsi manipulovať s dátami, na ktoré nemá dostatočné práva, bude o tom patrične oboznámený. Upozornený nebude iba užívateľ, ale aj osoba zodpovedná za bezpečnosť systému.



Obrázok 6: Schéma prepojenia a komunikácie systému DLP [23].

Výhodou je práca offline. V prípade, že koncová stanica je odpojená od siete, nedochádza k vypnutiu ochrany. V agentovi je totiž uložená posledná verzia firemnej bezpečnostnej politiky, takže ochrana prebieha neustále, aj keď si užívateľ odnesie dôležité dáta na notebooku domov a nie je pripojený do firemnej siete.

V nasledujúcich kapitolách popíšeme významnejšie vlastnosti komerčných aplikácií.

5.1 Symantec – Insight

Symantec nazval technológiu, pomocou ktorej vyhľadáva a klasifikuje dáta, Insight (*nahliadnutie*). Táto technika ako jediná používa vektorové strojové učenie, aby znížila čas a prostriedky potrebné na vývoj nových politík, ktoré budú ochraňovať neštruktúrované dáta. Na vytvorenie politiky jej postačuje aj malé množstvo odtlačkov neštruktúrovaných súborov. Ďalšou vlastnosťou Insight je zistenie vlastníka súboru. V prípade, že pri prehľadávaní data at rest alebo data at endpoint nájde citlivý súbor, oznámi jeho top[18] piatim vlastníkom, že umiestnenie ich súboru porušuje firemné politiky [7].

5.2 Websense – PreciseID

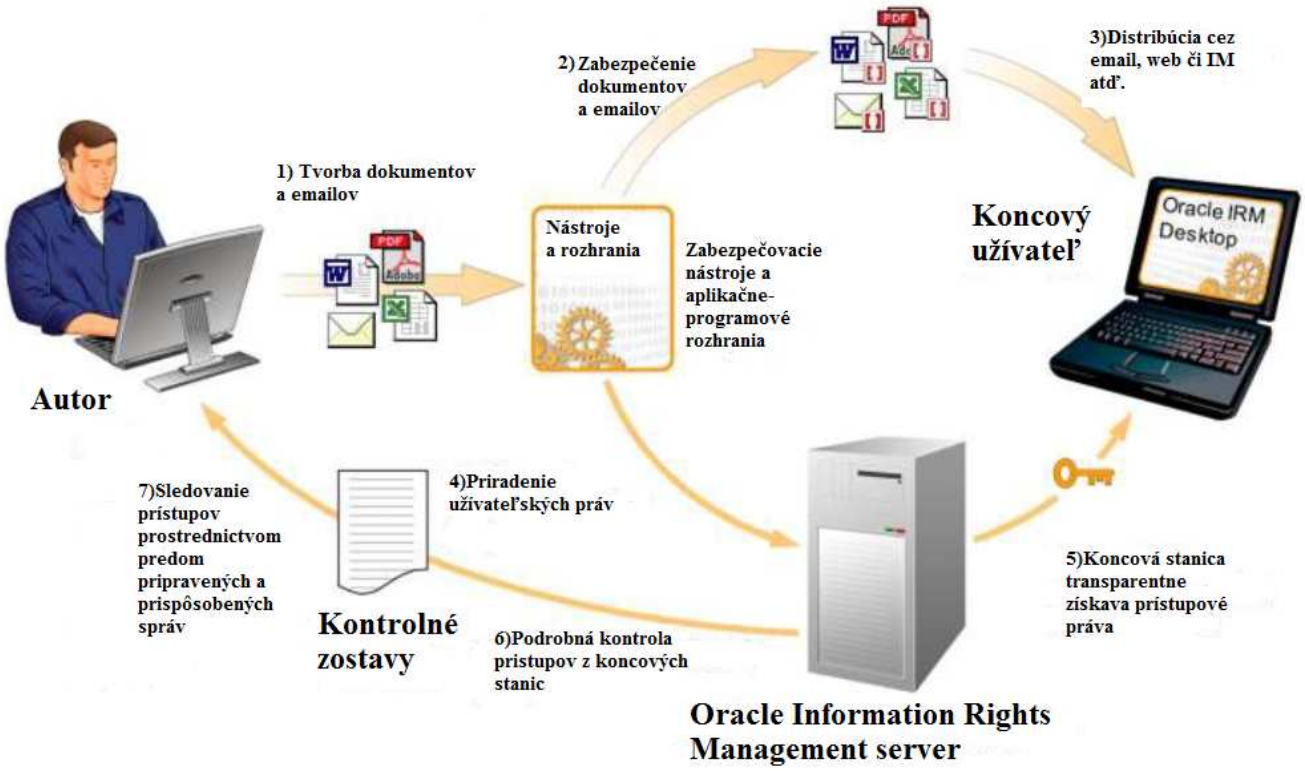
PreciseID je technológia, ktorá umožňuje presnejšiu identifikáciu a klasifikáciu obsahu viac ako 370 rôznych typov súborov a formátov, od zdrojového kódu cez CAD výkresy, Verilog kód a mnohé iné, hoci by bol obsah vyňatý a vložený do iného súboru. Táto technológia využíva viaceré metódy klasifikácie, ako databázu odtlačkov, regulárne výrazy, slovníky, čiastočné a celkové zhodovanie, statickú analýzu a spracovanie prirodzeného jazyka k tomu, aby pomohla organizáciám odhaliť ich aktíva a posilniť ich politiky. Spracovanie prirodzeného jazyka zvyšuje presnosť detekcie a klasifikácie PreciseID. Pôvodne bol vyvinutý pre potreby izraelskej armády, preto spĺňa vysoké bezpečnostné požiadavky [6].

5.3 Oracle IRM

Je systém na zabezpečenie a sledovanie dôležitých dokumentov, a to aj mimo firemnej siete. Využíva kryptografické algoritmy AES a RSA spolu s protimanipulačným softvérom a ochranou proti snímaniu obrazovky.

[18] Top piatim môžeme chápať, ako posledných päť vlastníkov, ktorí s daným súborom manipulovalo, alebo prvých piatich, ktorí sú v zozname jeho vlastníkov.

Oracle IRM[19] sa používa na šifrovanie (zapečatenie) súboru tak, aby iba oprávnené osoby mohli dočasne dešifrovať a používať daný súbor. Vďaka jeho transparentnosti môže autorizovaný užívateľ používať zapečatenú informáciu v štandardnom (operačnom) prostredí. Klientský softvér, pomocou ktorého dešifrujeme dokument, zároveň chráni a sleduje zapečatené informácie, aj keď sú používané a tak zabraňuje užívateľovi, aby získal čisté nešifrované dáta. Schému zabezpečenia si môžete pozrieť na obrázku 7 [5].



Obrázok 7: Schéma popisujúca fungovanie zabezpečenia dokumentov pomocou Oracle IRM [11].

5.4 Magic Quadrant

Spoločnosť Gartner Inc. [12] vytvára každoročne hodnotenia rôznych produktov rovnakého typu, ktoré sa nazýva Magic Quadrant. Výrobcov v ňom hodnotia podľa dvoch kritérií: *úplnosť vízie* a *schopnosť vykonávať*. Používa rôzne kvalifikátory pre tieto kritéria, pomocou ktorých MQ hodnotí účastníkov testu. Gartner nezdieľa získané informácie s účastníkmi. Podľa hodnotenia oboch kritérií vytvára graf rozdelený do štyroch kvadrantov:

- Vodcovia (*Leaders*), ktorí mali vysoké skóre v oboch kritériách. Typicky to bývajú veľké spoločnosti, ktoré si vybudovali podnik s víziou a potenciálom rásť.

[19] IRM (Information rights Management) je pojem označujúci technológiu, ktorá ochraňuje citlivé dáta pred neautorizovaným prístupom.

- Vyzývatelia (Challengers), tí majú vyššie skóre v kritériu schopnosti vykonávať. Typicky to bývajú väčšie spoločnosti s minimálnymi plánmi do budúcnosti.
- Vizionári (Visionares), tí majú vyššie skóre v kritériu úplnosti vízie. Bývajú to menšie spoločnosti, ktoré pomaly ale isto odhaľujú svoj potenciál.
- Drobní hráči (Niche players), títo majú nízke skóre v oboch kritériách. Bývajú to nováčikovia na trhu.

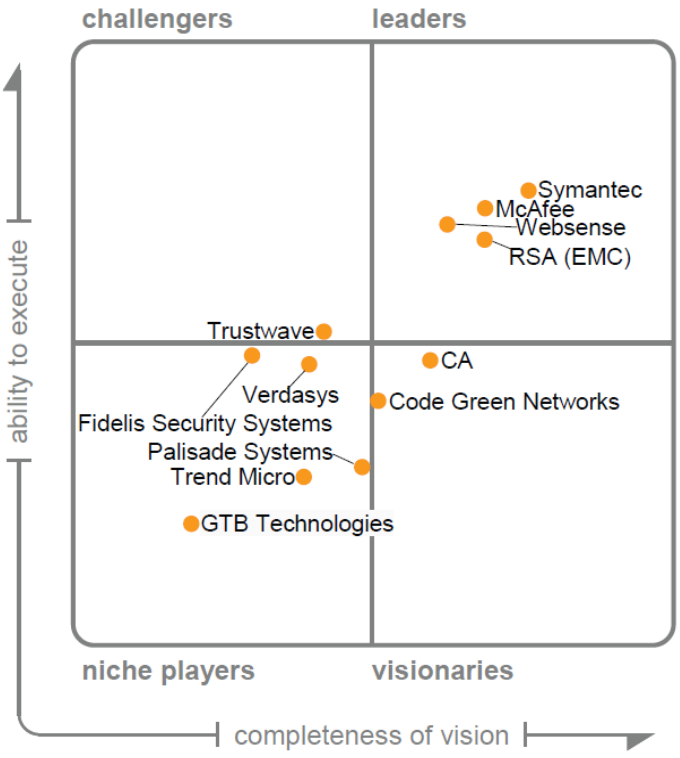
5.4.1 Výsledky Magic Quadrant pre DLP systémy

Vodcovia ukázali, že dobre chápu potreby klientov a poskytli im komplexné riešenia všetkých troch problémov (siete, vyhľadávania, koncové zariadenia), a to priamo alebo úzkou spoluprácou s klientmi. Poskytli inovujúce nápady a riešenia, ktoré ešte musia plne začleniť do prevádzky, aby stopercentne uspokojili potreby trhu.

Vyhodnocovacie kritéria	Váha
Pochopenie trhu	štandardná
Trhová stratégia	štandardná
Stratégia predaja	štandardná
Stratégia ponuky produktu	vysoká
Inovácia	štandardná
Grafická stratégia	štandardná

Vyhodnocovacie kritéria	Váha
Produkt/Služby	vysoká
Predaj/Cena	vysoká
Reakcia na požiadavky trhu	štandardná
Hodnotenie užívateľov	vysoká
Operácie	vysoká

Tabuľka 1 Vľavo sú vyhodnocovacie kritéria pre úplnosť vízie, v pravo pre schopnosť vykonávať [4].



Obrázok 8: Vyhodnotenie Magic Quadrantu v roku 2010 v oblasti systémov DLP [4].

Trustwave sa ako jediný ocitol v kvadrante vyzývateľov nielen vďaka tomu, že jeho produkt má dobré schopnosti, ale najmä preto, že jeho obchodný model nie je zameraný iba na DLP problém.

Vo vizionárskom kvadrante sa ocitli CA a Code Green Networks. CA sa ocitla opäť, ako v roku 2009 vo vizionárskom kvadrante aj napriek tomu, že odvtedy spravila pár významných krokov, ale ešte stále jej chýbajú niektoré kľúčové funkcie. Code Green Networks je kvalitný produkt najmä pre malé a stredné podniky, ale nájde si uplatnenie aj u pár väčších podnikov.

V kvadrante drobných hráčov sa pre rok 2010 objavilo až päť výrobcov. GTB Technologies a Palisade Systems sú malé rozvíjajúce sa firmy, ale pomaly sa doťahujú na konkurenciu. Verdasys a Fidelis Security System poskytujú najsľubnejšie technológie pre koncové zariadenia a siete, ale chýbajú im komplexné riešenia, ktoré požadujú veľké podniky [4].

6 Návrh aplikácie

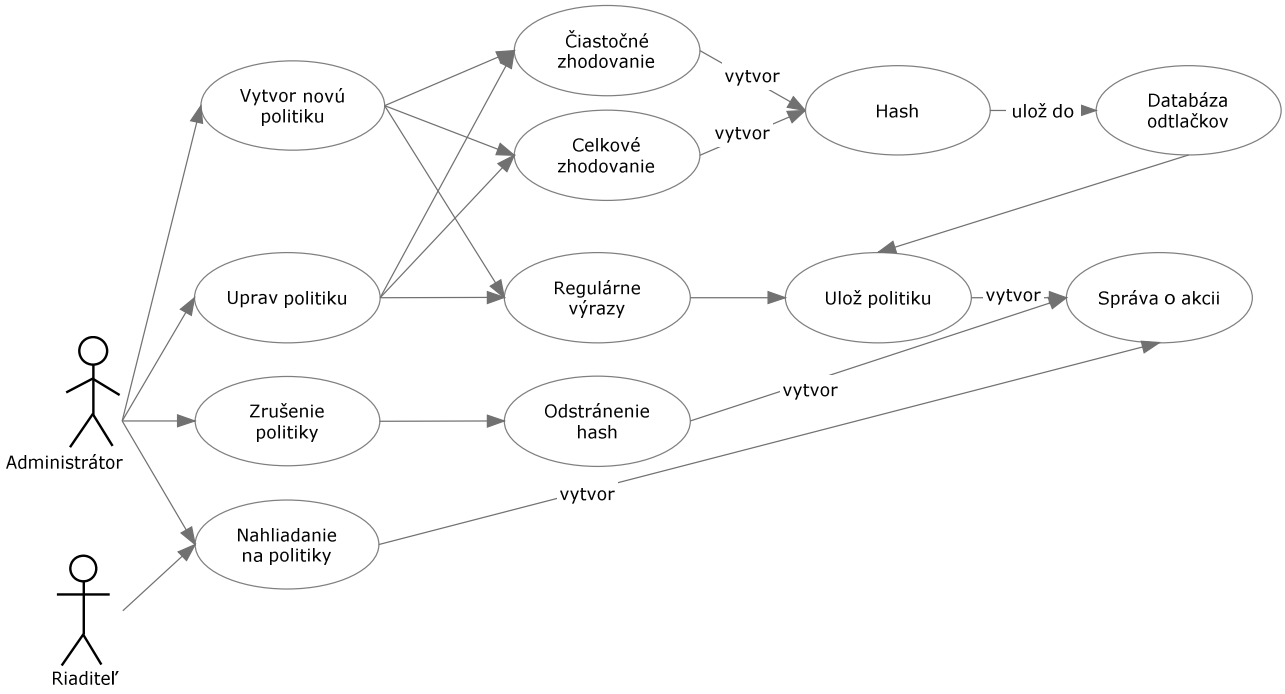
Cieľom tejto kapitoly je špecifikovať požiadavky na aplikáciu, ktorá by bola schopná ochrániť koncovú stanicu pred únikmi citlivých dát. Jedná sa teda o návrh endpoint agenta, ktorý by mal vykonávať monitoring (sledovať, detekovať priestupky a klasifikovať ich), vytvárať správy o incidentoch, uchovávať správy o incidente a spravovať politiky (vytvárať, pozmeňovať a rušiť). Navrhovaná aplikácia nebude predstavovať plne funkčný DLP systém.

Aplikácia je navrhovaná pre operačný systém MS Windows a predpokladá nasledujúce hierarchické usporiadanie užívateľov (čím nižšie číslo, tým má užívateľ väčšie oprávnenia pri zaobchádzaní s endpoint agentom):

1. Administrátor DLP systému.
2. Riaditeľ.
3. Úradník.

Úradník ma spravidla viac obmedzení pri manipulácii so súborom ako riaditeľ.

6.1 Správa politík



Obrázok 9: Správa politík.

Správu politík má na starosti administrátor DLP systému (viď obr. 9). On je aj zodpovedný za správny návrh a úpravu politík. Jeho možné akcie sú: vytvor politiku, uprav politiku, zruš politiku a nahliadanie do politík. Ďalším aktérom je riaditeľ, jeho jedinou akciou je nahliadanie do politík bez možnosti úpravy. Úradník nemá žiadne povolené akcie spojené so správou politík.

Pri vytváraní politiky sa budeme rozhodovať, pomocou akej techniky ochránime daný obsah. Ak je obsah v textovej forme, potom naň môžeme použiť techniku regulárnych výrazov. V prípade, že je obsah v binárnej (alebo textovej) forme, tak použijeme techniku čiastočného alebo celkového zhodovania. Atribúty politiky sú:

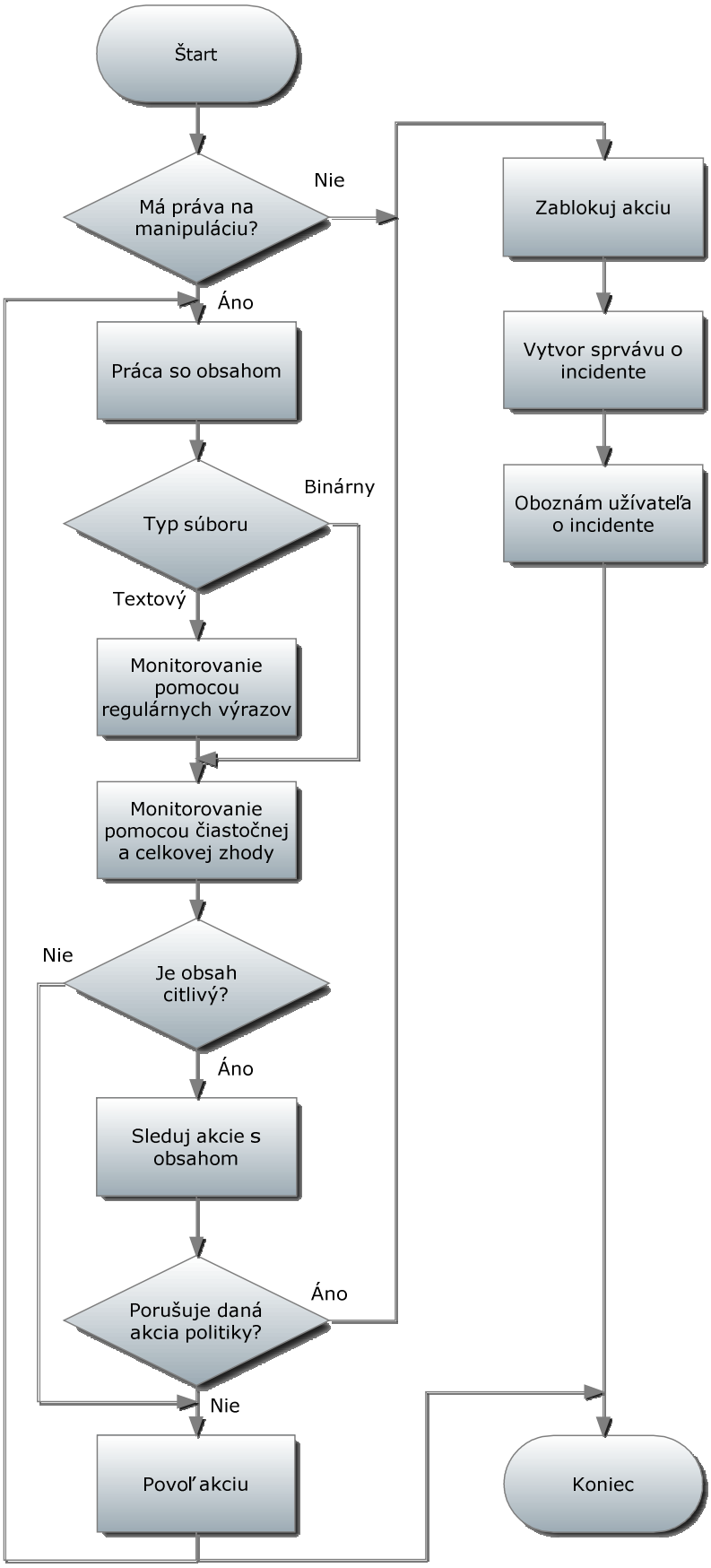
- Cesta – k zložke, k súboru.
- Použité pravidlo – buď to jedno, alebo kombinácia dostupných techník.

Úprava politík je dôležitým úkonom správy. Politiky upravujeme v prípade, že ich nasadenie je neefektívne alebo neúčinné. V takomto prípade môžeme politiku celkom zrušiť alebo ju môžeme potrebne upraviť.

6.2 Monitorovanie akcií

Monitorovaním zisťujeme, či nejaká akcia neporušuje definované politiky. Monitor pri tom sleduje viaceré okolnosti. V prvom rade zistí, či má užívateľ k danému obsahu prístup. Ak nie, vytvorí správu o incidente a oboznámi užívateľa, že porušil politiky. Inak povolí užívateľovi manipulovať s obsahom. Monitor pomocou regulárnych výrazov a čiastočného alebo celkového zhodovania zisťuje, či je alebo nie je manipulované s citlivým obsahom. Ak sa nakladá s obsahom v rozpore s politikami, napríklad presúvanie dát do nezabezpečeného súboru, monitor okamžite zablokuje dané vloženie a patrične to oznámi. V prípadoch, keď obsah (citlivý alebo nie) a manipulácia s ním sú v súlade s politikami, monitor vyhodnotí celok ako bezpečný a povolí ho. Spôsob monitorovania je detailnejšie popísaný na obr. 10.

Monitor nám taktiež zaručuje kontrolu umiestnenia citlivých dát. Súbory sa niekedy nepozornosťou alebo nedbalosťou môžu dostať mimo kontrolovaný priestor. Prehľadávaním súboru po súbore monitorom zistíme také anomálie a podnikneme potrebné kroky k ochrane daných dát. Ideálnym riešením daného problému je premiestnenie súboru do dočasnej ochrannej zložky a oboznámenie vlastníka súboru. V niektorých prípadoch by bolo lepšie, aby sa daný súbor zmazal alebo aby sa zašifroval jeho obsah.



Obrázok 10: Monitorovanie činnosti.

6.3 Správa incidentov

Pod pojmom správa incidentov si môžeme predstaviť databázu, ktorá uchováva informácie o každom priestupku proti politikám. Každá správa by mala obsahovať tieto základné údaje (obr. 11.):

- Kto vykonal incident, aby bolo možné vyvodiť dôsledky.
- Kedy sa daný incident stal, aby sme mohli zistiť, či nejaký užívateľ si každý štvrťrok posiela „omylom“ časti firemného know-how.
- Aká politika bola porušená, aby sme v prípade chybného návrhu a implementácie politiky boli schopní zistiť, o ktorú politiku sa jedná.
- Aká akcia bola vykonaná.

K databáze ma prístup iba bezpečnostný technik, ktorý je spôsobilý riešiť dané problémy.

Správa o incidente
KTO: Juraj Pap
KEDY: 12:34 12.január 2011
POLITIKA: Vytváranie kopií citlivých dát
AKCIA: Zamedzenie akcie, oboznámenie úžívateľa

Obrázok 11: Príklad správy o incidente vkladanej do logu.

7 Ovládače (Drivers)

Cieľom tejto kapitoly je oboznámiť čitateľa s princípmi vývoja ovládačov. V nasledujúcich kapitolách si najprv predstavíme hlavný model ovládačov, z ktorého sa odvíjajú všetky návrhy implementácií ovládačov. Vysvetlíme si, ako postupujú vstupno-výstupné požiadavky a ako na ne reaguje aktuálny ovládač. Predstavíme si špeciálny typ ovládača zvaného minifilter, jeho vlastnosti a spôsob filtrovania požiadaviek.

7.1 Windows Driver Model

Pre správnu komunikáciu a ovládanie periférneho zariadenia postačí použiť iba jeden ovládač. V praxi sa to takýmto spôsobom nerobí. Fyzické zariadenia (disky, tlačiarne, klávesnice, zobrazovacie jednotky a pod.) sú občas obsluhované celou skupinou zreťazených ovládačov, ktorá sa nazýva *driver stack* (zásobník ovládačov).

Každý ovládač v zásobníku vykonáva časť práce potrebnej k podpore danej vstupno-výstupnej (ďalej I/O *input/output*) operácie. Základná motivácia pre tento vrstevný prístup je založená na potrebe redukovať redundantný kód, a to zavedením deľby práce. Niektoré periférne zariadenia môžu zdieľať rovnaké ovládače zariadení a I/O zbernicu. Riešením by mohlo byť, ak by sme implementovali opakovane ten istý kód pre každý ovládač zariadenia, ale to by bola strata času, energie a prostriedkov. Lepším riešením je modularizovať funkcionalitu do jednoznačných komponentov, čoho následkom bude maximalizácia možnosti opätovného použitia daného modulu. Tento spôsob podporuje aj fakt, že ovládače v móde jadra sú podľa Windows driver model [14] (WDM) rozdelené do troch základných tried:

- Ovládače funkčnosti (*function drivers*).
- Ovládače zberníc (*bus drivers*).
- Filtrovacie ovládače (*filter drivers*).

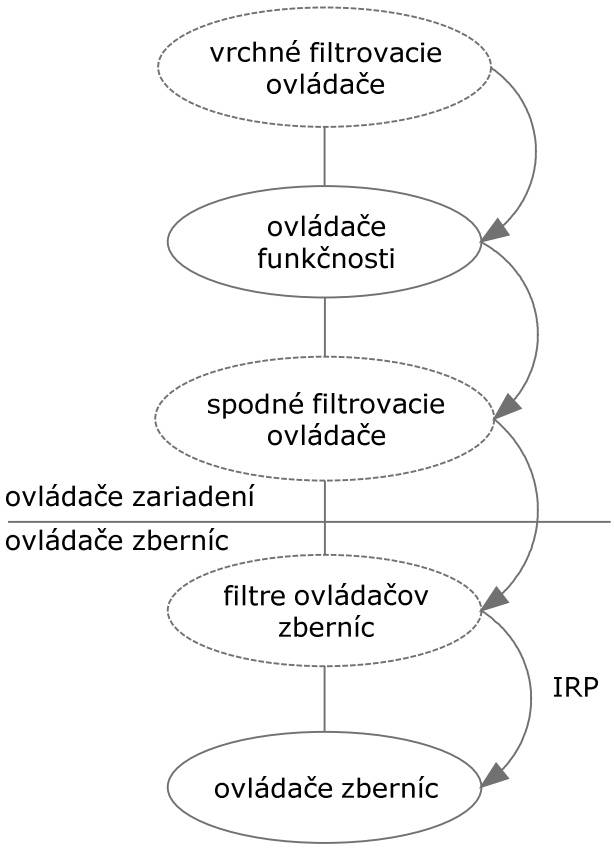
Ovládače funkčnosti sú brané ako primárne ovládače pre dané zariadenie. Ako primárne ich označujeme preto, lebo vykonávajú drvivú väčšinu obsluhy I/O požiadaviek. Tento druh ovládača prevezme volanie Windows API, ktoré bolo vytvorené napr. klientskou aplikáciou a preloží ho do skupiny I/O príkazov, ktoré môžu byť ďalej poslané k zariadeniu. Ovládače funkčnosti sú zvyčajne v zásobníku ovládačov umiestnené niekde v strede medzi filtrovacími ovládačmi a ovládačmi zberníc.

Ovládače zberníc implementujú špecifickú funkcionalitu pre dané hardvérové rozhranie (napr. USB, PCI, SCSI a pod.). Sú to v podstate nízkoúrovňové ovládače funkčnosti pre daný typ zbernice. Ovládače zberníc sú stále umiestnené na dne zásobníka ovládačov.

Filtrovacie ovládače sami o sebe neovládajú hardvér. Namiesto toho prerušujú a modifikujú informácie, ktoré cez ne prechádzajú. Filtrovacie ovládače môžu byť použité napríklad pre šifrovanie

dát pri zápise na disk a následné dešifrovanie pri čítaní dát z disku. Tieto ovládače môžeme rozdeliť v závislosti od ich pozície vzhľadom k ovládačom funkčnosti na vrchné a spodné filtre. Ovládače funkčnosti a zberníc sú často implementované ako pár ovládač-minifilter ovládač alebo obecne ako pár trieda-minitrieda ovládača.

Trieda ovládača poskytuje podporu operácií pre obecný hardvér (žiaden špecifický) na základe jeho typu (klávesnica, monitor a pod.). Systém Windows poskytuje sadu tried ovládačov. Ovládače minitried sú poskytované výrobcom daného zariadenia. Tie podporujú jeho špecifické operácie.



Obrázok 12: Zreťazenie ovládačov v zásobníku ovládačov a predávanie IRP paketu.

7.2 Paket I/O požiadavky

Ovládače v zásobníku ovládačov predávajú informácie ovládačom pod nimi, v závislosti na obsahu IRP (I/O Request Packet – paket I/O požiadavky). Celý proces začína, keď užívateľská aplikácia pošle I/O požiadavku. Windows I/O manažér zabalí túto požiadavku do IRP a umiestni objekt zariadenia na vrchol zásobníku zariadení a použije ho k nasmerovaniu IRP do vhodnej odbavovacej rutiny najvrchnejšieho ovládača daného zariadenia. Ak daný ovládač dokáže obslúžiť danú požiadavku, ukončí I/O požiadavku a vráti IRP I/O manažérovi. Ukončenie si vysvetlíme neskôr, zatiaľ chápme ukončenie I/O požiadavky ako to, že zásobník ovládačov vykonal to, čo po ňom bolo požadované alebo sa o to aspoň pokúsil.

Ak sa vrchnému ovládaču nepodarilo uspokojiť požiadavku, potom spraví iba to, čo môže a následne zistí, ktorý ovládač je ďalší v rade pod ním pomocou volania *IoCallDriver()*. Požiada I/O manažéra, aby poslal IRP k ďalšiemu ovládaču (obr. 12). Táto séria krokov sa opakuje pre ďalší a ďalší ovládač, až kým sa neukončí I/O požiadavka. Poznamenajme, že požiadavka sa ukončí vždy. Ak narazí na najspodnejší ovládač, ten ju už nemá kam ďalej poslať, a tak ju musí ukončiť.

Aby sme lepšie pochopili fungovanie vrstvenia ovládačov, pozrime sa bližšie na štruktúru IRP. Oficiálna dokumentácia Windows Driver Kit [16] popisuje IRP štruktúru ako „čiastočne zahmlenú“, čo v preklade znamená, že nám o nej nepovedia viac, než sú ochotní. Nezdokumentované položky sú vraj rezervované a používané iba I/O manažérom.

```
typedef struct _IRP {
  PMDL  Type;
  ULONG  Flags;
  union {
    struct _IRP  *MasterIrp;
    PVOID  SystemBuffer;
  } AssociatedIrp;
  IO_STATUS_BLOCK  IoStatus;
  KPROCESSOR_MODE  RequestorMode;
  BOOLEAN PendingReturned;
  BOOLEAN  Cancel;
  KIRQL  CancelIrql;
  PDRIVER_CANCEL  CancelRoutine;
  PVOID UserBuffer;
  union {
    struct {
    union {
      KDEVICE_QUEUE_ENTRY DeviceQueueEntry;
      struct {
        PVOID  DriverContext[4];
      };
    };
    PETHREAD  Thread;
    LIST_ENTRY  ListEntry;
    } Overlay;
  } Tail;
} IRP, *PIRP;
```

Najpodstatnejšia časť tejto štruktúry je jej typ. Každý IRP, ktorý I/O manažér pošle dole zásobníkom, má pridelenú tzv. hlavnú funkciu (*major function*), ktorá je označená ako *IRP_MJ_xxx*. Kódy týchto funkcií hovoria ovládaču, akú operáciu by mal vykonať, aby uspokojil I/O požiadavku. Tri najčastejšie sa vyskytujúce typy IRP sú:

- *IRP_MJ_CREATE*.
- *IRP_MJ_READ*.
- *IRP_MJ_WRITE*.

IRP_MJ_CREATE sa posiela pri otváraní a vytváraní nových súborov. *IRP_MJ_READ* a *IRP_MJ_WRITE* sa posielajú po otvorení súboru a označujú čítanie a zápis dát.

Pri vytváraní IRP I/O manažér naalokuje prídavný úložný priestor za IRP hlavičkou pre každý ovládač v zásobníku ovládačov o veľkosti štruktúry *IO_STACK_LOCATION* [17]. Táto štruktúra sa líši podľa typu IRP. Zvyčajne obsahuje ukazovateľ na odbavovaciu rutinu v ovládači, ktorú I/O

manažér zavolá, a takisto informácie, ktoré budú predané volajúcej rutine. Taktiež obsahuje ukazovateľ na objekt zariadenia, ktorému je daný IRP priradený.

7.3 Preposielanie IRP

Keď odbavovacia rutina ovládača príjme IRP, zvyčajne získa hodnoty parametrov z jej *IO_STACK_LOCATION*. Po získaní si s dátami naloží podľa vlastných potrieb. Ak na konci svojej funkčnosti táto rutina plánuje preposlanie IRP k ďalšiemu ovládaču v zásobníku, tak musí:

1. Naplniť *IO_STACK_LOCATION* štruktúru pre ďalší ovládač potrebnými parametrami.
2. Zaregistrovať si ukončovaciu rutinu (tento krok je voliteľný).
3. Poslať IRP k ďalšiemu ovládaču v rade.
4. Vrátiť návratovú hodnotu.

Je viacero spôsobov ako nastaviť potrebnú štruktúru. Jedným z nich je, ak nepoužívame momentálnu štruktúru na nič špeciálne a chceli by sme ju jednoducho predať do ďalšieho ovládača, zavoláme rutinu *IoSkipCurrentIrpLocation()*, ktorá to vykoná [15].

7.4 Ukončenie IRP

IRP sa nemôže donekonečna preposielať. Nakoniec musí byť ukončený. Je to podstata toho, že IRP hľadá svoje ukončenie. Ak IRP dosiahne na najspodnejší ovládač v zásobníku, ten ho musí ukončiť, pretože ho už nemá kam poslať. Bez ohľadu na to, či sa táto akcia skončí úspechom alebo neúspechom, I/O požiadavka je na konci.

Z technického hľadiska sa však ešte nekončí. Hlavne ak I/O manažér začne svoj ukončovací proces pre daný IRP. Po ukončení IRP I/O manažér prechádza štruktúru IO_STACK_LOCATION počnúc aktuálnym ovládačom a hľadá, či má predchádzajúci ovládač zaregistrovanú ukončovaciu rutinu. Ak nie je nič zaregistrované, I/O manažér sa presunie na IO_STACK_LOCATION vyššieho ovládača a znova ho prehľadáva. Takto postupuje, až kým nedosiahne vrchol zásobníka ovládačov pre dané zariadenie. V prípade, že si ovládač počas spracovania IRP zaregistroval ukončovaciu rutinu, I/O manažér ju vykoná a presunie sa znova bližšie k vrcholu zásobníka.

IRP je najpodstatnejšou časťou systému vrátane I/O volaní. Prechádza dvomi cestami. Pri prvej zostupuje dole po zásobníku ovládačov. Zastavuje sa iba v ovládačoch, ktoré majú potrebnú registrovanú rutinu. Po dokončení tejto cesty je IRP ukončený a smer v zásobníku sa obráti. Tentokrát IRP zastavuje iba v ovládačoch, ktoré majú registrovanú ukončovaciu rutinu.

7.5 FS minifilter

Minifilter je ovládač, ktorý filtruje požiadavky smerujúce k a od zariadenia, presnejšie k úložným zariadeniam (FS – *filesystem* –súborový systém). Po predchádzajúcich kapitolách sa vynára otázka, prečo vlastne vytvárať nejaký minifilter, keď možno vytvoriť filtrovací ovládač. Hlavnými výhodami minifiltra sú:

- Je jednoduchší na vývoj. Môžeme vytvoriť spoľahlivejší ovládač s menším úsilím.
- Dynamicky ho môžeme načítať a odstrániť, a taktiež dynamicky pripojiť a odpojiť od zariadenia.
- Pripojiť ho môžeme na presne definované miesto v zásobníku ovládačov.
- Správa kontextu: rýchle, čisté a spoľahlivé kontexty pre objekt súboru, tok dát a pod.
- Skupina úžitkových rutín vrátane podpory získavania mien a ich ukladanie pre ďalší efektívny prístup. Ďalej môžeme spomenúť rutiny pre komunikáciu medzi minifiltrom a jeho obsluhou v užívateľskom móde.
- Podpora nerekurzívnych I/O požiadaviek, takže požiadavky vygenerované minifiltrom budú viditeľné iba ovládačom pod ním v zásobníku a súborovým systémom.
- Filtruje iba vybrané operácie, na rozdiel od jeho predchodcu (*legacy model*). Tento musel mať pre každú požiadavku, ktorá ním prechádzala, odbavovaciu rutinu.

7.5.1 Základné pojmy

- Filter – ovládač, ktorý vykonáva určitý druh filtrovania operácií.
- Zväzok (*volume*) – objekt reprezentujúci logický zväzok daného súborového systému.
- Inštancia (*instance*) – inštancia filtra na zväzku na určitej úrovni. Jednému zväzku môže byť priradených aj viac inštancií minifiltra.
- Súbor – pomenovaný objekt dát, ktorý je uložený na disku a môže sa skladať z viacerých tokov.
- Tok (*stream*) – reprezentuje fyzický tok dát súboru.
- Návratové dáta (*CallbackData*) – štruktúra, ktorá obsahuje informácie o operácii. Je to ekvivalent k IRP.

7.5.2 Inštalácia minifiltra

Minifiltre sa inštalujú pomocou tzv. INF[20] súboru. INF súbor označuje, aké inštancie daný minifilter podporuje. Každé označenie inštancie musí obsahovať aj hodnotu pozície, na ktorú sa umiestni, identifikáciu triedy (GUID[21]), do ktorej patrí a taktiež množinu príznakov.

Príznaky určujú, či minifilter potrebuje automatické priradenie k zväzku. Ak áno, tak pre každý zväzok v systéme dostane minifilter notifikáciu o tom, že sa môže naň pripojiť.

Tabuľka 2 Hodnoty pozícií a GUID definované spoločnosťou Microsoft [18].

Poradie načítania	Pozícia skupiny	GUID triedy
Filter	420000-429999	<bez GUID>
FSFilter Top	400000-409999	<bez GUID>
FSFilter Activity Monitor	360000-389999	{b86dff51-a31e-4bac-b3cf-e8cfe75c9fc2}
FSFilter Undelete	340000-349999	{fe8f1572-c67a-48c0-bbac-0b5c6d66cafb}
FSFilter Anti-Virus	320000-329999	{b1d1a169-c54f-4379-81db-bee7d88d7454}
FSFilter Replication	300000-309999	{48d3ebc4-4cf8-48ff-b869-9c68ad42eb9f}
FSFilter Continuous Backup	280000-289999	{71aa14f8-6fad-4622-ad77-92bb9d7e6947}
FSFilter Content Screener	260000-269999	{3e3f0674-c83c-4558-bb26-9820e1eba5c5}
FSFilter Quota Management	240000-249999	{8503c911-a6c7-4919-8f79-5028f5866b0c}
FSFilter System Recovery	220000-229999	<bez GUID>
FSFilter Cluster File System	200000-209999	{cdcf0939-b75b-4630-bf76-80f7ba655884}
FSFilter HSM	180000-189999	{d546500a-2aeb-45f6-9482-f4b1799c3177}
FSFilter Imaging	170000-174999	<FSFilter Compression GUID>
FSFilter Compression	160000-169999	{f3586baf-b5aa-49b5-8d6c-0569284c639f}
FSFilter Encryption	140000-149999	{a0a701c0-a511-42ff-aa6c-06dc0395576f}
FSFilter Virtualization	130000-139999	<nedefinované GUID>
FSFilter Physical Quota management	120000-129999	{6a0a8e78-bba6-4fc4-a709-1e33cd09d67e}
FSFilter Open File	100000-109999	{f8ecafa6-66d1-41a5-899b-66585d7216b7}
FSFilter Security Enhancer	80000-89999	{d02bc3da-0c8e-4945-9bd5-f1883c226c8c}
FSFilter Copy Protection	60000-69999	{89786ff1-9c12-402f-9c9e-17753c7f4375}
FSFilter Bottom	40000-49999	<nedefinované GUID>
FSFilter System	20000-29999	{5d1b9aaa-01e2-46af-849f-272b3f324c46}

Hodnotu pozície si môže každá inštancia určiť dynamicky na požadovanú úroveň, a to kedykoľvek počas chodu minifiltra prostredníctvom volania *FilterAttacheAtAttitude()*.

[20] INF súbor je súbor obsahujúci nastavovacie informácie. Operačný systém Windows používa tento typ súborov k inštalácií softvéru alebo ovládačov.

[21] GUID (Globaly Unique Indentificator)– jedinečný globálny identifikátor, pozostáva z 32 znakov hexadecimálnej sústavy. Jeho maximálna hodnota presahuje 2 trilióny, a to z toho dôvodu, aby sa znížila možnosť priradenia dvoch rovnakých GUID.

7.5.3 Registrácia minifiltra

FS minifiltre sú ovládače na úrovni jadra, preto musia obsahovať funkciu nazývanú *DriverEntry()*, ktorá je prvou zavolanou funkciou v momente, keď je ovládač načítaný. Rutina *DriverEntry()* je niečo podobné ako *main()* v jazyku C. Väčšina minifiltrov v nej volá *FltRegisterFilter()*.

FltRegisterFilter() má ako vstupný parameter štruktúru *FLT_REGISTRATION* [19], ktorá obsahuje rutinu na odstránenie minifiltra, notifikácie spätných volaní inštancie, zoznam s ukazovateľmi na kontextové funkcie spätných volaní a zoznam ukazovateľov na spätné volania operácií súborového systému. Vo väčšine prípadov obsahuje každý zoznam iba pár záznamov.

7.5.4 Inicializácia filtrovania

Po tom, ako sa minifilter zaregistruje, by mal začať s filtrovaním pomocou volania *FltStartFiltering()*. Nie je potrebné, aby sme ho volali z *DriverEntry()*, ale väčšina minifltrov je takto implementovaná. Táto funkcia rozpošle notifikácie, čoho následkom bude pripojenie minifiltra k potrebným zväzkom a štart filtrovania I/O požiadaviek.

Správca filtrov prechádza zoznamom inštancií minifiltra. Každej je priradená hodnota pozície. Hodnoty pozícií pre komerčné minifiltre vydáva Microsoft a môžeme ich nájsť v [20]. Čím vyššie číslo hodnoty pozície inštancia obsahuje, tým bude umiestnená vyššie v zásobníku pre daný zväzok.

Pre vývojárov je poskytnutých pár hodnôt, ktoré ich nepodpísané[22] ovládače môžu používať.

7.5.5 Notifikácie inštancie

Ide o množinu spätných volaní, ktoré informujú minifilter v prípade, že jeho inštancia bola vytvorená alebo zrušená. Minifilter môže prostredníctvom týchto volaní kontrolovať pripájanie alebo odpájanie jeho inštancie od zväzku.

Nastavovacia rutina *InstanceSetupCallback()* je volaná v nasledujúcich prípadoch:

- Ak sa načíta minifilter, tak raz pre každý zväzok súborového systému.
- Ak je pripojený nový zväzok.
- Pri volaní nasledujúcich funkcií *FltAttach(), FltAttachVolume(), FltAttachAtAltitude(), FltAttachVolumeAtAttitude()*.

V priebehu spracovania tejto rutiny sa minifilter rozhodne, či má k zväzku pripojiť svoju inštanciu alebo nie.

InstanceQueryTeardown() rutina je volaná iba v prípade, že je inštancia odpájaná manuálne. To môžeme vyvolať pomocou *FltDetachVolume()* alebo *FltDetach()*. V prípade, že ovládač neposkytuje túto rutinu, nepodporuje ani manuálne odstránenie. Ak *InstanceQueryTeardown()* vráti úspešný návratový kód, potom filter manažér začne s odstraňovaním danej inštancie.

[22] Ako podpísané ovládače sa označujú tie, ktoré boli preverené a schválené spoločnosťou Microsoft [39].

7.5.6 Vytvorenie komunikačného portu

Pre vytváranie bezpečnej a viackanálovej komunikácie sa používa objekt komunikačného portu minifiltra. Je určený pre komunikáciu medzi módom jadra a užívateľským módom v oboch smeroch. Komunikácia v rámci módu jadra nie je podporovaná. Port je pomenovaný objekt so svojím bezpečnostným deskriptorom. Manažér filtrov vytvorí tento objekt pri inicializácii minifiltra pred tým, než bude hociktorá inštancia minifiltra načítaná. Tento port môžu vytvoriť iba ovládače, ktoré sú v móde jadra volaním *FltCreateCommunicationPort()*.

```
NTSTATUS
FltCreateCommunicationPort(
    IN PFLT_FILTER Filter,
    OUT PHANDLE PortHandle,
    IN POBJECT_ATTRIBUTES ObjectAttributes,
    IN PVOID ServerPortCookie OPTIONAL,
    IN PFLT_CONNECT_NOTIFY ConnectNotifyCallback,
    IN PFLT_DISCONNECT_NOTIFY DisconnectNotifyCallback,
    IN PFLT_MESSAGE_NOTIFY MessageNotifyCallback,
    IN ULONG MaxConnections
    );
```

Filter je nástrojom pre objekt minifiltra, ktorý vytvára tento objekt komunikačného portu.

PortHandle získame po úspešnom volaní funkcie *FltCreateCommunicationPort()*.

Parameter *ObjectAttributes* definuje rovnomennú štruktúru, ktorá je potrebná na inicializáciu mena, pomocných príznakov a bezpečnostného deskriptora pre práve vznikajúci komunikačný port.

ServerPortCookie je kontext, ktorý minifilter môže priradiť k portu a tento kontext je nejasný pre manažéra filtrov. Všetky notifikácie o pripojení k či odpojení od minifiltra sa predajú tomuto „cookie“.

V prípade, že sa proces v užívateľskom móde hocikedy pokúsi otvoriť port, tak sa zavolá *ConnectNotifyCallback()* rutina. Minifilter týmto nadobúda schopnosť zabrániť nežiaducemu pripojeniu.

Ak sa na strane užívateľského módu uzavrie port, zavolá sa *DisconnectNotifyCallback()*.

MessageNotifyCallback() je volaný ,ak ovládač príjme správu.

Maximálny počet pripojení, ktorý je povolený pre daný port, je určený pomocou *MaxConnections*.

V prípade, že minifilter úspešne vytvoril komunikačný port, ihneď na ňom začne načúvať a v tejto činnosti pokračuje, až kým nie je uzavretý.

7.5.7 Pripájanie sa z užívateľského módu na minifilter

Na strane aplikácie v užívateľskom móde sa nachádza funkcia na otvorenie komunikačného portu. V prípade úspešného spojenia sa v minifiltri zavolá rutina *ConnectNotify()*, aby ho oboznámila o vytvorení spojenia, ktoré sa bude využívať na posielanie správ medzi módom jadra a užívateľským módom. Prototyp funkcie pre pripojenie užívateľskej aplikácie je nasledovný:

```
HRESULT
FilterConnectCommunicationPort(
    IN LPWSTR lpPortName,
    IN DWORD dwOptions,
    IN LPVOID lpContext,
    IN WORD wSizeOfContext,
    IN LPSECURITY_ATTRIBUTES lpSecurityAttributes,
    OUT HANDLE *hPort
     );
```

lpPortName je ukazovateľ na reťazec znakov, ktorý špecifikuje meno portu, ku ktorému sa pripája. Toto meno musí byť totožné s menom použitým pri vytváraní portu v minifiltri. Musí začínať znakom '\', čo znamená, že ide o hlavný adresár.

Parameter *dwOptions* sa nepoužíva.

lpContext je ukazovateľ na nejasný parameter, ktorý bude predaný rutine minifiltra *ConnectNotify()*. To sa môže použiť pri autentizácii aplikácie, ktorá sa snaží vytvoriť komunikačný kanál. *wSizeOfContext* určuje veľkosť *lpContext* v bajtoch.

lpSecurityAttributes určuje bezpečnostné atribúty definované pre užívateľskú stranu.

Aby bol proces schopný rozpoznať informáciu o prijatí správy, je potrebné vytvoriť tzv. notifikátor ukončenia prenosu správy. Ten indikuje procesu prijatú správu pripravenú na spracovanie. Notifikátor vytvoríme volaním:

```
HANDLE WINAPI
CreateIoCompletionPort(
     IN      HANDLE PortHandle,
     IN_OPT  HANDLE ExistingCompletionPort,
     IN      ULONG_PTR CompletionKey,
     IN      DWORD NumberOfConcurrentThreads
     );
```

PortHandle je ukazovateľ na komunikačný port.

ExistingCompletitionPort je ukazovateľ na už vytvorený notifikátor ukončenia. Používa sa v prípadoch, ak chceme navýšiť počet komunikačných vlákien.

CompletionKey je užívateľom nastavený kľúč, ktorý sa posiela spolu s každou notifikáciou.

Hodnota *NumberOfconcurrentThreads* určuje maximálny počet konkurenčných vlákien, ktoré je notifikátor schopný obslúžiť.

7.5.8 Prerušenie komunikácie

O prerušenie komunikácie sa v užívateľskom móde stará *CloseHandle()* a v móde jadra *ZwClose()*. Následne sa zavolá rutina minifiltra *DisconnectNotify()*, ktorá sa postará o celkové ukončenie spojenia. Minifilter by mal zvyčajne ukončovať spojenie v tejto rutine. V inom prípade sa musí ošetriť prípad dvojnásobného uzavretia portu.

Minifilter môže zavolať *ZwClose()* kedykoľvek, to nezruší už vytvorené spojenia, ale postará sa iba o to, že sa už žiadne iné spojenia nebudú vytvárať.

7.5.9 Získavanie mien súborov

Manažér filtrov poskytuje knižnicu rutín potrebných pre získanie mena objektu v momentálne vykonávanej operácii z parametrov operácie alebo pýtaním sa systému. Pre zvýšenie výkonu si každé získané meno uloží do svojej cache pamäte a minifiltrom požadujúcim meno objektu predá iba ukazovateľ naň. Manažér filtrov vracia požadované mená v štruktúre *FLT_FILE_NAME*, aby zabránil zbytočnému kopírovaniu mien v prípade ich potreby. Táto štruktúra je iba odkaz a je možné, že ju medzi sebou zdieľa viacej minifiltrov. Iba manažér filtrov by mal meniť dáta v tejto štruktúre.

Pre získanie mena súboru pre aktuálnu operáciu musí minifilter zavolať nasledujúcu rutinu:

```
NTSTATUS
FLTAPI
FltGetFileNameInformation (
    IN PFLT_CALLBACK_DATA CallbackData,
    IN FLT_FILE_NAME_FORMAT NameFormat,
    IN FLT_FILE_NAME_QUERY_METHOD QueryMethod,
    OUT PFLT_FILE_NAME_INFORMATION *FileNameInformation
    );
```

CallbackData je štruktúra *FLT_CALLBACK_DATA* pre operáciu na súbore, ktorého meno minifilter požaduje.

NameFormat môže byť:

- *FLT_FILE_NAME_NORMALIZED_FORMAT* – meno požadované týmto formátom obsahuje plnú cestu k súboru vrátane mena zväzku. Všetky krátke názvy (krátke názvy sú pozostatok MS-Dos, kde meno súboru alebo zložky je v tvare 8.3) sú rozšírené na ich plnohodnotné názvy.
- *FLT_FILE_NAME_OPENED_FORMAT* – meno pod týmto formátom obsahuje plnú cestu k súboru vrátane mena zväzku, ale meno je rovnaké, aké použil jeho žiadateľ pri jeho otváraní, preto môže obsahovať krátke názvy pre každý komponent v ceste k súboru.
- *FLT_FILE_NAME_SHORT_FORMAT* - meno pod týmto formátom obsahuje krátky názov iba koncového komponentu cesty. Zvyšok cesty sa neposkytuje.

QueryMethod môže byť:

- *FLT_FILTER_NAME_QUERY_DEFAULT* – pri tomto spôsobe sa manažér filtrov najprv pozrie, či dané meno nemá uložené v cache pamäti. Ak nie, tak ho získa požiadavkou na súborový systém.
- *FLT_FILE_NAME_QUERY_CACHE_ONLY* – s týmto označením prehľadá iba cache pamäť. Ak ho v nej nájde, vráti ho, inak vráti chybový status *STATUS_CACHE_MISS*
- FLT_FILE_NAME_QUERY_FILE_SYSTEM_ONLY – pri tomto označení pošle iba požiadavku súborovému systému, ktorý mu zistí meno daného súboru.

Meno je vrátené ako parameter *FileNameInformation*. Táto štruktúra je množina reťazcov, ktoré zdieľajú jedno pole. Jednotlivé reťazce označujú rôzne časti mena súboru. Ak minifilter ukončí prácu s menom súboru, musí ju uvoľniť.

8 Implementácia

V tejto kapitole popisujeme implementáciu projektu a jeho základné komponenty. Projekt pozostáva z dvoch častí. Hlavná časť je File System[23] minifilter, ktorý filtruje I/O požiadavky smerujúce zo systému k súborovému systému. Druhá časť predstavuje užívateľskú aplikáciu pozostávajúcu z GUI, logiky rozhodovania a správy databáz. Tento projekt nie je plne funkčný DLP systém. Implementuje iba jeho časť monitorovania procesov. Zmyslom tejto práce je priblížiť čitateľovi možný návrh a realizáciu monitorovacej časti DLP systému. Minifilter je navrhovaný pre lokálne použitie a neberie v úvahu sieťové disky.

Implementácia tohto systému sa líši od návrhu aplikácie, pretože som pri návrhu nepredpokladal dopad času a náročnosti vývoja tejto aplikácie [13]. Hlavným rozdielom je, že sa nekontroluje obsah súborov, ale iba ich umiestnenie.

8.1 FS minifilter

Minifilter ovládač som implementoval v jazyku C vo vývojovom prostredí WDK, ktoré bližšie vysvetlím v ďalšej kapitole. Bolo by možné implementovať tento ovládač aj v jazyku C++ za použitia kompilátoru zvaného Super-C, ale Microsoft neodporúča (ale nezakazuje ho) vývoj v C++, pre prípad možnej nekompatibility niektorých C++ knižníc, ktoré sú vytvárané výhradne pre použitie v užívateľskom móde, a iných neočakávaných chýb [25].

Úlohou minifiltra je filtrovať I/O požiadavky, ktoré si zaregistroval u manažéra. Konkrétne sa jedná o *IRP_MJ_CREATE*, *IRP_MJ_READ* a *IRP_MJ_SET_INFORMATION*[24].

Pri implementácií minifiltra som vychádzal z modelu ovládačov spomenutého v kapitole 7.V nasledujúcich kapitolách popíšem spôsob implementácie a inštalácie minifiltra.

8.1.1 Instable File System Kit

Pri vývoji tohto ovládača som použil softvérovú sadu Windows Driver Kit(WDK) [28], ktorá v sebe zahŕňa príslušnú dokumentáciu, príklady, vývojové prostredie a nástroje pre vývoj ovládačov. Pre vývoj filtrovacích ovládačov (minifiltrov) sa používa jej modul Instable File System Kit [29].

Instable File System Kit je aplikačné rozhranie (API) pre IBM OS/2 a Microsoft Windows, ktoré dovoľuje operačnému systému rozoznať a načítať ovládače pre súborový systém. Pomocou

[23] Ďalej budem pre File System (súborový systém) používať skratku FS.

[24] Požiadavky typu *IRP_MJ_SET_INFORMATION* sú posielané v prípadoch, keď sa presúva súbor alebo keď sa mení jeho meno.

tohto modulu výrobcovia antivírusov, šifrovacích aplikácií a iných nástrojov na manipuláciu s dátami vyvíjajú svoje programy. Pre viac informácií odporúčam webové stránky MSDN[25].

8.1.2 Testovanie ovládača pomocou PREfast

Pri testovaní ovládača som použil „PREfast pre ovládače“ (PFD[26]), ktoré je rozšírením PREfast. Je to nástroj na statické overovanie v čase kompilácie, ktorý detekuje chyby, ktoré neobjavil kompilátor alebo sa nemusia vyskytnúť pri testovaní. Detekuje bežné programovacie chyby v programoch jazyka C a C++ a je taktiež navrhnutý na detekciu chýb v kóde ovládačov v móde jadra. Je integrovaný do vývojového prostredia WDK pre Windows 7.

PFD podporuje veľké množstvo anotácií, ktoré sú nad rámec bežného PREfast, vrátane anotácií pre rôzne pamäťové úniky a prísnejšiu typovú kontrolu.

PFD je vhodné použiť hneď na začiatku vývoja ovládačov, čo zabezpečí hladký priebeh jeho vývoja.

8.1.3 INF súbor

Ako som už spomenul INF súbor slúži na inštaláciu ovládačov. Obsahuje popis inštancie minifiltra, jej meno, hodnotu pozície v zásobníku a príslušnú triedu. Pri vytváraní inštalačného súboru som vychádzal z [23], ktorý popisuje vytváranie inštalačného INF súboru pre minifiltre.

Pri uvažovaní o správnom umiestnení minifiltra v zásobníku ovládačov bolo treba zvážiť jeho hlavnú funkciu. Tento minifilter má blokovať neoprávnený prístup k súborom, čím zvyšuje ich bezpečnosť. Takýto minifilter spadá do triedy *FSFilter Security Enhancer*[27]. Táto trieda má vyhradené hodnoty pozícií a GUID spomenuté v tab. 2.

GUID minifiltra musí byť totožné s GUID danej triedy a hodnota pozície sa musí pohybovať v jej rozsahu. Hodnotu pozície si môže vývojár zvoliť sám alebo môže o ňu požiadať priamo spoločnosť Microsoft vyplnením webového dotazníku na [24]. V tomto projekte som si hodnotu pozície zvolil.

Pri inštalácii som schopný nastaviť, kedy sa bude ovládač spúšťať. Na výber sú tri možnosti:

- Pri bootovaní.
- Pri štarte systému.
- Na požiadanie užívateľa.

V prípade tohto minifiltra je potrebné, aby sa spúšťal pri štarte systému a aby bol schopný hneď reagovať na I/O požiadavky.

---

[25] MSDN (Microsoft Developer Network) [27] je oficiálna centrála pre programátorov, softvérových inžinierov a dizajnérov, ktorí sa zaoberajú s produktmi pre operačný systém Windows.

[26] PREfast for drivers PFD [30].

[27] FSFilter Security Enhancer je trieda minifiltrov definovaná v [18] zabezpečujúcich zvýšenie ochrany dát.

8.1.4 Inicializácia minifiltra

Prvým krokom pri inicializácii každého ovládača, čiže aj tohto, je volanie jeho počiatočnej rutiny *DriverEntry()*. Touto rutinou sa minifilter zaregistruje u manažéra filtrov a inicializuje si v nej všetky globálne štruktúry. V tejto rutine vytvorí bezpečný port pre komunikáciu s užívateľským prostredím. Na tento port sa neskôr môže pripojiť iba aplikácia, ktorá pozná názov portu a má patričné administrátorské oprávnenia.

Po úspešnom vytvorení a otvorení portu sa spustí minifilter, ktorému manažér filtrov posiela notifikačné správy o prípadných zmenách alebo akciách.

Manažér filtrov potom pre každý zväzok v systéme zistí, či mu má byť priradená inštancia minifiltra. Keďže minifilter je navrhovaný pre lokálne použitie, tak v rutine *ScannerInstanceSetup()* zistí, či nie je daný zväzok náhodou sieťový disk. Táto rutina je taktiež volaná v prípade, že je do systému pridaný nový súborový zväzok. V prípade kladného výsledku manažér zaistí, aby bola danému zväzku pridaná inštancia na požadovanú úroveň.

V tomto momente je všetko pripravené na správnu funkciu minifiltra a čaká sa na pripojenie užívateľskej aplikácie. V prípade úspešného pripojenia na port si minifilter uchová PID procesu (identifikáciu procesu), ktorý sa na neho pripojil, pre prídavnú kontrolu.

8.1.5 Komunikácia s aplikáciou

V nasledujúcich kapitolách popíšem obslužné rutiny pre registrované typy požiadaviek. Pre ich lepšie pochopenie predstavím najprv ich hlavnú výkonnú jednotku. Je ňou funkcia *ScannerScanFileInUserMode()*, ktorá zabezpečuje informácie potrebné pre komunikáciu s užívateľskou aplikáciou. Pomocou nej minifilter naalokuje potrebné miesto pre štruktúru *FILE_INFO*, ktorá bude obsahovať meno procesu a súboru a identifikáciu akcie[28]. Táto štruktúra tvorí hlavnú časť správy posielanej užívateľskej aplikácii.

Následne sa zistí meno procesu, ktorý požiadavku vygeneroval a vloží sa do štruktúry spolu s menom otváraného súboru a identifikáciou akcie.

Pre zvýšenie „user frendly“ vlastnosti celkového projektu som musel pridať jednu kontrolu pred odoslaním správy užívateľskej aplikácii. Minifilter obsahuje definované prehliadače súborov (Explorer.exe, TOTALCMD.exe). Tieto uľahčujú užívateľovi pohodlnejšiu prácu so súbormi a uvažuje sa s nimi iba v prípade akcie otvárania súboru. Povolenie tejto akcie im umožňuje zobrazenie mena súboru. Takže pred odoslaním správy sa skontroluje, či sa meno procesu nezhoduje s názvami prehliadačov

[28] Akcie môžu byť: otváranie, kopírovanie a premenovávanie súboru.

Komunikáciu medzi minifiltrom a aplikáciou zabezpečuje funkcia *FltSendMessage*:

```
NTSTATUS FltSendMessage(
  __in       PFLT_FILTER Filter,
  __in       PFLT_PORT *ClientPort,
  __in       PVOID SenderBuffer,
  __in       ULONG SenderBufferLength,
  __out_opt  PVOID ReplyBuffer,
  __inout    PULONG ReplyLength,
  __in_opt   PLARGE_INTEGER Timeout
);
```

Kde *Filter* je ukazovateľ na minifilter, *ClientPort* je ukazovateľ na ukazovateľ na komunikačný port, *ScannerBuffer* je ukazovateľ na alokované pole, obsahujúce správu, ktorú posielame. *SenderBufferLength* je veľkosť naalokovaného poľa. *ReplyBuffer* predstavuje pole, v ktorom sa vráti odpoveď a *ReplayLength* je dĺžka tejto odpovede. Hodnota *Timeout* určuje čas (v stovkách nanosekúnd), po ktorý má minifilter čakať na odpoveď. V tomto prípade je táto hodnota nastavená na *NULL*, čo značí, že minifilter bude čakať do nekonečna alebo až kým nedostane odpoveď.

V odpovedi sa nachádza informácia o povolení alebo zakázaní požiadavky, podľa ktorej sa minifilter rozhoduje.

8.1.6 Filtrovanie IRP – otváranie súborov

Po dlhšom zisťovaní a uvažovaní som dospel k tomu, že ak chcem, aby bol minifilter schopný detekcie otvorenia súboru, musím filtrovať IRP typu *IRP_MJ_CREATE*, ktoré sa vyskytujú pri vytváraní a otváraní súboru. Preto si pri inicializácii minifilter musí zaregistrovať pre tento typ správy jej obslužnú rutinu, ktorú som nazval *ScannerPreCreate()*. Táto rutina je volaná v prípade, že užívateľské API vytvorí I/O požiadavku na otvorenie súboru, čiže reaguje na požiadavku smerujúcu k zariadeniu a rozhoduje o jej povolení či zamietnutí. Pri registrácii som nedefinoval žiadnu ukončovaciu rutinu pre tento typ požiadavky.

Bolo by možné vytvoriť minifilter, ktorý by disponoval obidvomi rutinami. V smere od užívateľského API k zväzku by som zistil, či sa jedná o sledovanú akciu. V kladnom prípade by si minifilter zaregistroval iba ukončovaciu rutinu. V opačnom smere by sa pri volaní ukončovacej rutiny vykonali potrebné akcie a rozhodlo by sa o povolení I/O požiadavky.

Výhodou môjho rozhodnutia je to, že všetky informácie, ktoré potrebujem, mám k dispozícii už pri posielaní požiadavky k zväzku. Zaujíma ma hlavne meno otváraného súboru a procesu, ktorý sa snaží tento súbor otvoriť.

Ak by som to zisťoval v ukončovacej rutine, mal by som k dispozícii aj obsah súboru. Ten je pre mňa nepodstatnou vecou a musel by som zaistiť jeho správne uvoľnenie v prípade zákazu požiadavky. V úvahu by som ho bral iba vtedy, ak by sa kontroloval aj tento obsah súboru.

Pre ujasnenie som implementoval iba *ScannerPreCreate()* bez ukončovacej rutiny.

Ako prvé *ScannerPreCreate()* zistí, či je vytvorené komunikačné spojenie, keďže rozhodovacia logika sídli v užívateľskom móde. Bez jej pomoci sa nedokáže rozhodnúť, či má danú požiadavku povoliť, alebo nie. Ak nie je vytvorené spojenie, minifilter automaticky povoľuje všetky požiadavky.

V opačnom prípade si minifiler od manažéra vyžiada meno súboru aktuálnej požiadavky a nasleduje typová kontrola súboru. Pre zvýšenie výkonu minifiltra som sa rozhodol, ktoré typy súborov sa majú kontrolovať. Výber padol na .pdf, .txt, .doc, .jpg a .bmp, iné typy sa automaticky povolia. Názvy typov sú definované ako globálne pole reťazcov a pre koncového užívateľa sa môžu podľa potreby a požiadaviek upravovať vhodnou aktualizáciou minifiltra na iné typy súborov. Prídavnou kontrolou vykonávanou v tejto rutine je kontrola otváracieho procesu. Ak požiadavku vytvoril náš užívateľský proces, povolíme ju. Táto požiadavka sa vytvára pri pridávaní súboru do kontrolného zoznamu.

Pre rozhodnutie akcie predá rutina meno súboru funkcii *ScannerScanFileInUserMode()*, ktorá zabezpečuje komunikáciu s užívateľskou aplikáciou a čaká na rozhodnutie.

Ak užívateľská aplikácia zamietne akciu, minifilter pre danú požiadavku nastaví stav návratových hodnôt štruktúry *FLT_CALLBACK_DATA*, presnejšie jej podštruktúry *IO_STATUS_BLOCK*., a to položky:

- *Data->IoStatus.Status*, ktorej hodnota označuje návratový status nejakou návratovou hodnotou z hodnôt *NTSTATUS* [31]. V prípade zamietnutia nastavujem túto hodnotu na *STATUS_ACCESS_DENIED*, čo pre tvorcu požiadavky značí, že má zakázaný prístup k požadovanému súboru.
- *Data->IoStatus.Information*. V prípade úspechu požiadavky (*STATUS_SUCCESS*) obsahuje počet prenášaných bajtov. Pri zamietnutí ju nastavím na hodnotu nula, čo značí, že neobsahuje žiadne dáta.

Tento cyklus, počnúc volaním *ScannerPreCreate()* až po rozhodnutie požiadavky, sa stále opakuje pre daný typ požiadavky, až kým neukončíme program.

8.1.7 Filtrovanie IRP – kopírovanie a premenovávanie súborov

Aby som zvýšil bezpečnosť sledovaných súborov, musel som brať v úvahu možnosť, že niekto súbor premenuje alebo presunie na iné miesto na disku, čím by zrušil sledovanie daného súboru.

Pri premenovávaní systém vygeneruje požiadavku typu *IRP_MJ_SET_INFORMATION*, ktorá obsahuje súbor, ktorý sa má premenovať. Preto si musíme pre daný typ požiadavky registrovať u manažéra minifiltrov odbavovaciu rutinu bez ukončovacej rutiny z rovnakých dôvodov, ako pri *IRP_CREATE*. Nazval som ju *ScannerPreSetInformation()*.

Pri kopírovaní súboru proces, ktorý ho kopíruje, ho musí najprv prečítať, a preto vygeneruje požiadavku typu *IRP_MJ_READ*. Aj pre tento typ požiadaviek si minifilter musí zaregistrovať odbavovaciu rutinu bez ukončovacej rutiny. Pomenoval som ju *ScannerPreRead()*.

Funkcie *ScannerPreSetInformation()* a *ScannerPreRead()* sú podobné so *ScannerPreCreate()*. Pošle sa správa užívateľskej aplikácii, ktorá rozhodne o výsledku požiadavky. Funkcie sa líšia iba tým, že v posielanej správe sa nastaví, o akú požiadavku ide. To napomáha užívateľskej aplikácii pri rozhodovaní.

8.1.8 Zistenie mena procesu

Zisťovanie mena procesu bol jeden z najväčších problémov pri implementácii tohto minifiltra. Pomocou nedokumentovaných funkcií, ako napríklad *PsGetProcessImageFileName()*, som bol schopný zistiť iba štruktúru *EPROCESS*, obsahujúcu informácie o procese, ktorý vytvoril I/O požiadavku. Táto štruktúra obsahuje iba prvých 16 znakov cesty k danému procesu. Niekedy by cesta k procesu mohla byť aj dlhšia ako 16 znakov. Ak by sa cesty k dvom procesom zhodovali v prvých 16 znakoch, tak by sme ich nevedeli rozoznať a prípadne ani stanoviť, o aký proces sa jedná. To bol hlavný dôvod, prečo som pre zisťovanie mena procesu musel hľadať iný spôsob.

Prvým riešením bolo spojenie funkcií *GetProcessHandle()* a *GetObjectByHandle()*. Prvá mi vrátila ukazovateľ na proces a pomocou druhej som získal objekt procesu. Objekt obsahoval iba skrátený koncový názov vo formáte 8.3[29]. Tu by mohlo vzniknúť nebezpečenstvo zo strany nejakého škodlivého súboru, ktorý by mal síce odlišnú cestu k súboru, ale volal by sa rovnako ako proces, ktorý môže pristupovať ku kontrolovanému súboru.

Na fóre OSR Online[30] som sa nakoniec oboznámil s návrhom funkcie *GetProcessImageName()*, pomocou ktorej som schopný získať celú cestu k procesu. Táto funkcia sa snaží naplniť pole vstupného reťazca. Ak reťazec nemá naalokované potrebné miesto, tak mu nastaví iba jeho potrebnú veľkosť a ukončí sa stavom *STATUS_BUFFER_OVERFLOW*. Ak má reťazec pole o potrebnej veľkosti, vráti v ňom požadovanú cestu k procesu.

K tejto funkcii som vytvoril pomocnú funkciu *GetRequestorPath()*, ktorá volá dvakrát *GetProcessImageName()*. Pri prvom volaní sa zistí, aké veľké pole je potrebné pre danú cestu. Následne ju alokuje a nasleduje druhé volanie, ktorým získame reťazec obsahujúci cestu k procesu.

8.2 Užívateľská časť

Užívateľská časť v tomto projekte predstavuje jeho logickú časť. Jej hlavnou úlohou je komunikácia s minifiltrom a rozhodovanie o výsledku akcie. Z hľadiska modulov som ju rozdelil na päť častí:

1. Hlavná časť zaručuje vytvorenie spojenia medzi minifiltrom a aplikáciou, vytvorenie databáz, spustenie GUI a ukončenie programu.

[29] 8 prvých znakov názvu a 3 prvé znaky prípony napr. TOTALCMD.EXE.

[30] OSR Online je diskusné fórum pre vývojárov minifiltrov súborových systémov.

2. Správa databáz zabezpečuje pridávanie, odstraňovanie a hlavne predprípravu dát na zobrazenie.
3. Správa užívateľského prostredia obsahuje funkcie zabezpečujúce obsluhu GUI a správne zobrazenie informácie.
4. Správa reťazcov disponuje funkciami na úpravu posielaných správ.
5. Správa vlákna obsahuje obslužnú rutinu vlákna.

Užívateľskú aplikáciu som implementoval v jazyku C++.

8.2.1 Hlavná časť

Užívateľská aplikácia je bez minifiltra nefunkčná. Preto sa pri jej inicializácii aplikácia pokúsi pripojiť na dopredu definovaný port[31]. Pri neúspechu pripojenia sa aplikácia ukončí chybovou správou.

Keďže inštancie minifiltra sú priraďované každému lokálnemu zväzku na počítači, bolo by neefektívne, ak by sme obsluhovali správy minifiltra jedným vláknom, prípadne hlavným procesom. Preto je možné pri spúšťaní aplikácie definovať, koľko obslužných vlákien by sa malo o tieto správy starať. V prípade spustenia aplikácie bez parametrov je počet vlákien nastavený na hodnotu 2.

Aby aplikácia bola schopná vo vláknach komunikovať s minifiltrom, musí vytvoriť pre komunikačný port tzv. notifikátor ukončenia, ktorý sa stará o doručovanie správ vláknam.

Nasleduje vytvorenie spojenia s databázou jej tabuliek. Ich vytvorenie a obsluha budú vysvetlené neskôr.

Ak operácia pripojenia prebehla úspešne, sme schopní vytvoriť vlákna, ktoré budú komunikovať s minifiltrom. Každému vláknu sa predá štruktúra *SCANNER_THREAD_CONTEXT*, ktorej obsahom sú ukazovatele na komunikačný port, jeho ukončovací notifikátor a na databázu. Ďalej je vláknam naalokovaný priestor pre prijímanie správ.

Na konci hlavnej časti sa vytvorí dialógové okno. Okno slúži užívateľovi ako indikátor behu programu a jeho ukončením sa ukončí celá aplikácia vrátane komunikačných vlákien.

8.2.2 Správa databáz

Pri riešení problému ako spravovať politiky, ktorými sa tento projekt riadi, som mal na výber tri možnosti ukladania politík.

Prvou bolo použitie textového súboru, ktorý by obsahoval jednu alebo viac tabuliek. Hlavným problémom tohto riešenia je sprístupnenie požadovanej tabuľky viacerým vláknam, aby sa zachovala integrita celej databázy. Ďalšie problémy by mohli nastať napríklad pri návrhu alebo správe databázy.

Druhou možnosťou by bolo použitie MySQL servera [21] na správu databáz. Jeho nevýhodou je, že by som sa musel starať aj o správu servera.

[31] Pripájanie je bližšie objasnené v kapitole 7.5.7.

Treťou a mnou zvolenou metódou je použitie knižnice SQLite dostupnej na [32]. SQLite implementuje vstavaný, bezserverový, nekonfigurovateľný a transakčný SQL[32] databázový engine. Kód SQLite je verejný, čím sa stáva voľne použiteľným na akýkoľvek účel komerčný či súkromný. Engine, na rozdiel od ostatných SQL databáz, neobsahuje separátne procesy servera. SQLite číta a zapisuje priamo do súborov na disku. Kompletná SQL databáza s viacerými tabuľkami, indexmi a pohľadmi sa môže nachádzať v jednom súbore. Formát databázy je platformovo prenositeľný a môže byť kopírovaný aj medzi 32 a 64 bitovými operačnými systémami.

Ďalšou výhodou SQLite je, že podporuje prácu s viacerými vláknami. Môže byť v troch módoch:

1. Jednovláknový. V tomto móde sú znefunkčnené všetky mutexy a SQLite nie je vhodný pre viacej ako jedno vlákno.
2. Viacvláknový. V tomto móde sa môže SQLite používať vo viacerých vláknach, ale ku každej databáze môže byť pripojené iba jedno vlákno.
3. Sériový. V tomto móde sa môže SQLite bezpečne používať vo viacerých vláknach bez akýchkoľvek obmedzení.

Tento mód sa nastavuje v čase kompilácie. Implicitne je nastavený sériový mód.

8.2.3 Implementácia databázy

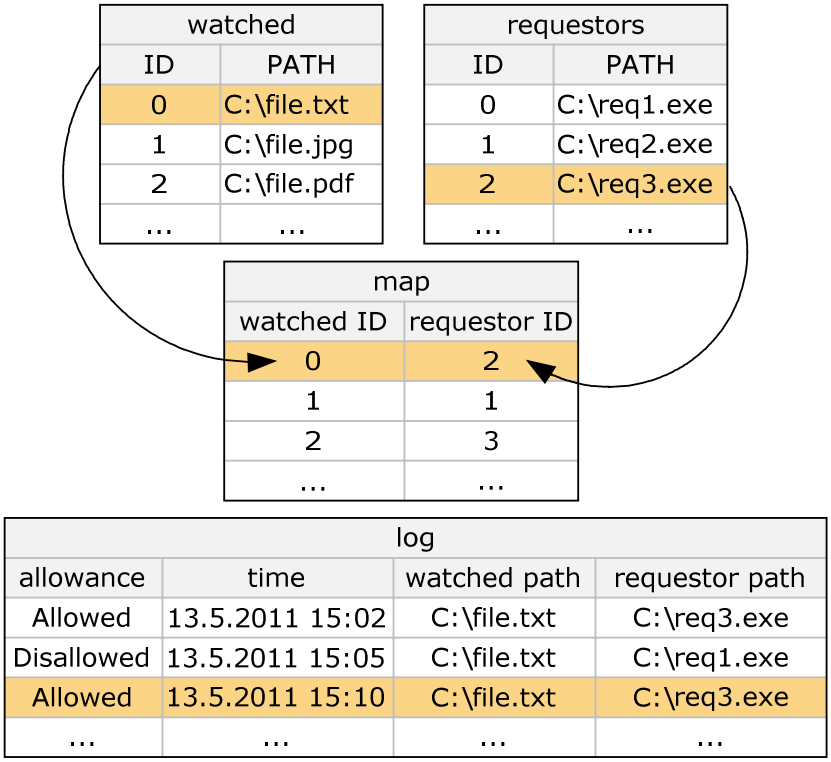
Ako som poznamenal v kapitole 8.1.5, minifilter posiela užívateľskej aplikácii informáciu o tom, ktorý súbor je otváraný a akým procesom. Z týchto informácií vyplýva, že potrebujeme jednu tabuľku na kontrolované súbory (*watched*) a druhú (*requestors*) na procesy (viď obr. 13), ktoré tieto súbory budú otvárať. Obe tieto tabuľky budú pozostávať z dvoch stĺpcov. Prvý bude obsahovať jedinečný identifikátor (ID) a druhý jedinečnú cestu k súboru či procesu. Jedinečnosť zaručuje, že nebudeme mať v databáze viackrát tú istú cestu alebo dve rôzne cesty nebudú mať ten istý identifikátor.

Aby sme sa vedeli rozhodnúť, ktorý proces ma právo otvoriť daný súbor, potrebujeme ďalšiu tabuľku (*map)*. Má taktiež dva stĺpce. Prvý obsahuje hodnotu ID súboru a druhý hodnotu ID procesu, ktorý daný súbor môže otvoriť.

Posledná tabuľka v databáze je tabuľka *log*. Obsahuje záznam o stave akcie, jej čase a o tom, aký proces otváral aký súbor.

Tabuľky sú znázornené na obr. 13. Je na ňom znázornený príklad otvárania súboru. Pre proces s ID = 2 a súbor s ID = 0 aplikácia nahliadne do tabuľky *map* a vyhľadá daný pár. Ak sa v nej nachádza požiadavku povolí a informáciu o nej vloží do logu (tabuľky *log*).

[32] SQL (Structured Query Language) je počítačový jazyk používaný pre prácu s dátami v databázach.



Obrázok 13: Príklad tabuliek databázy.

8.2.4 Užívateľské rozhranie – GUI

Užívateľské rozhranie slúži užívateľovi na ovládanie databázy. Umožňuje prehľadné pridávanie nových sledovaných súborov a ich otváracích procesov, ich upravovanie a sledovanie zachytených akcií v logu. Po vzore, že každá aplikácia by mala mať svoje meno, som pomenoval túto aplikáciu DaDy. Jej vzhľad je znázornený na obr. 14.

GUI je vytvárané pomocou Windows API, ktoré poskytuje prostredie pre vývoj grafického užívateľského prostredia s bežným ovládaním.

Užívateľské prostredie pozostáva z dvoch zoznamov a štyroch ovládacích tlačidiel. Zoznam *LIST_FILE* obsahuje výpis z tabuľky sledovaných súborov. *LIST_PROCESS* sa aktualizuje podľa práve označeného súboru v *LIST_FILE* a obsahuje zoznam procesov z tabuľky *requestors*, ktoré môžu daný súbor otvárať.

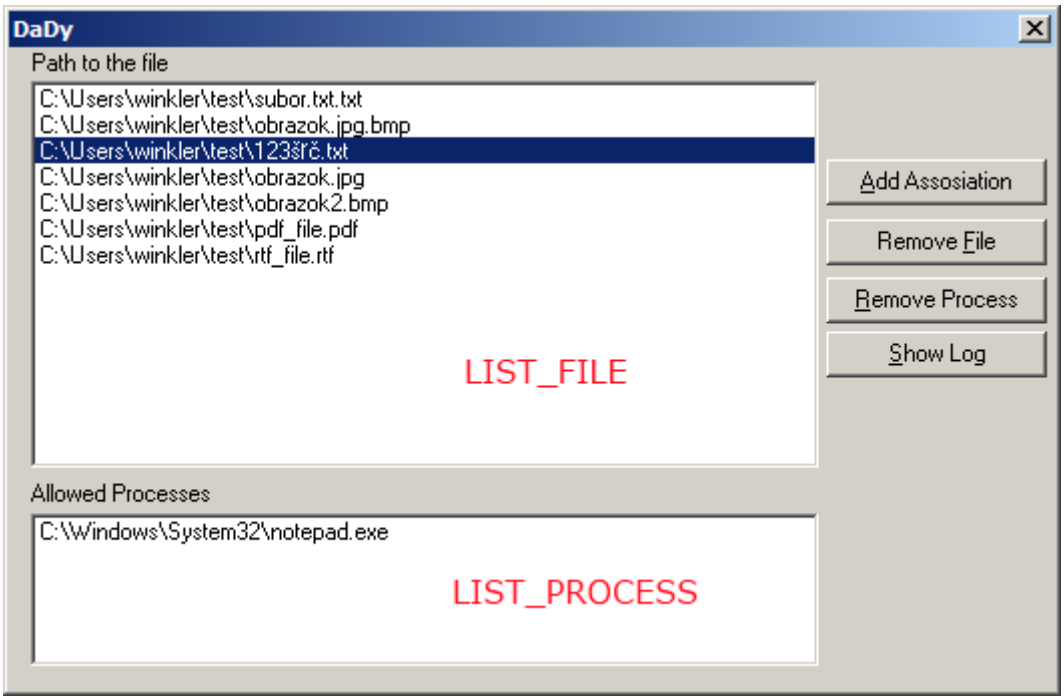
Po stlačení tlačidla *Add Assosiation* sa vytvorí nové dialógové okno na vybranie súboru požadovaného sledovaného súboru jeho otváracieho procesu. Po vybraní procesu a kontrolovaného súboru sa skontroluje, či už sú vložené v odpovedajúcich tabuľkách a aké v nich majú ID, ak nie, pridajú sa a zistí sa ich ID. Ak sa daný pár ID nevyskytuje v tabuľke *map*, tak sa do nej pridá.

Tlačidlo *Remove File* odstráni označený sledovaný súbor v *LIST_FILE* z tabuľky *watched* a odstránia sa všetky jeho asociácie v tabuľke *map*. Proces, ktorý otváral tento súbor sa ponechá v databáze pre možné opätovné použitie.

Pre aktivovanie tlačidla *Remove Process* musí byť označený súbor v *LIST_FILE* a proces v *LIST_PROCESS*. Daná asociácia sa odstráni v tabuľke *map* a obnoví sa zoznam *LIST_PROCESS*.

Môže nastať situácia, kedy pre sledovaný súbor nebude definovaný žiaden proces, ktorý ho môže otvárať. V takomto prípade bude tento proces blokovaný, až kým mu nepriradíme otvárací proces.

Tlačidlom *Show Log* vyvoláme nové dialógové okno, v ktorom sa zobrazí obsah tabuľky *log* v reverznom poradí. Toto okno obsahuje tlačidlo *Back*, ktoré ho ukončí, a *Refresh*, ktoré obnoví jeho obsah.



Obrázok 14: Vzhľad hlavného dialógového okna.

8.2.5 Správa vlákna

Správa vlákna obsahuje obslužnú rutinu vlákna. Pri spustení vlákna táto rutina obdrží štruktúru obsahujúcu ukazovatele komunikačného portu, jeho notifikátora ukončenia a databázy.

Telo vlákna je umiestnené v nekonečnej slučke, ktorá sa ukončí iba v prípadoch, ak sa ukončí hlavné dialógové okno, po jeho ukončení sa nastaví zdieľaná premenná *shouldExitThread*[33], alebo ak sa zruší komunikačný port.

Cyklus začína kontrolou notifikátora ukončenia. Vlákno čaká na notifikáciu po dobu 100 ms, po tejto dobe preruší túto činnosť a skontroluje, či sa má ukončiť. Ak nie, opäť sa vráti ku kontrole notifikátora. V prípade, že notifikátor oznámi príjem správy, vlákno príjme správu a vyberie z nej obsah štruktúry *OPENED_FILE* obsahujúcej cestu k otváranému súboru, jeho procesu a identifikáciu akcie.

[33] *shouldExitThread* indikuje pre vlákno, že hlavné dialógové okno sa už končilo a musí sa ukončiť aj vlákno.

Nasleduje volanie funkcie *CheckAllowance*, ktorá si zistí, či sa dané cesty nachádzajú v tabuľkách *watched* a *requestors*. V danom momente môžu nastať štyri prípady pre akcie otvárania a kopírovania súboru:

1. Obe cesty sa nenachádzajú v databáze[34].
2. Cesta súboru sa nevyskytuje v databáze a cesta procesu áno.
3. Cesta súboru sa vyskytuje v databáze, ale proces sa v nej nevyskytuje.
4. Obe cesty sa nachádzajú v databáze.

V prvých dvoch prípadoch je zjavné, že ak sa cesta k súboru nevyskytuje v databáze, tak nás tento súbor nezaujíma a túto akciu môžeme povoliť.

Tretí prípad nastáva, ak sa nepovolený proces snaží otvoriť kontrolovaný súbor. To porušuje bezpečnostnú politiku, a preto musí byť táto akcia zakázaná a zaznamenaná do logu.

V poslednom prípade, kde sa obe cesty objavujú v databáze, je potrebné zistiť, či má daný proces možnosť otvárať požadovaný súbor. Ak takéto oprávnenie má, danú akciu povolíme, inak ho zakážeme a zapíšeme do logu.

V prípade premenovávania súboru rozhoduje iba to, či sa premenovávaný prvok nachádza v tabuľke *watched*. Súborom nachádzajúcim sa v tej tabuľke je zakázané premenovávanie. V prípade pokusu o akciu premenovania je zaznamenaná do logu.

8.2.6 Správa reťazcov

V tejto práci som sa musel zamýšľať aj nad správnou reprezentáciou reťazcov. Išlo hlavne o prípady, ak som chcel zobraziť v GUI cestu k súboru či procesu alebo ak sa vkladali tieto cesty do databázy. Oba reťazce ciest sú v každej fáze prenosu rovnakého dátového typu, preto budem ďalej hovoriť o jednom, ale platí to pre oba.

Reťazec, ktorý sa zistí v minifiltri je obsiahnutý v štruktúre *UNICODE_STRING* presnejšie v jej položke *Bffer*, ktorý je dátového typu *WSTR*. Tento typ pozostáva zo znakov typu *WCHAR* (*wide-charakter*[35]). Obsah *Buffer* sa vloží do odpovedajúcej položky štruktúry *FILE_INFO*[36]. Reťazec v *Buffer* nie je ukončený *\0*. Preto ho po vložení do štruktúry musíme patrične ukončiť.

[34] V databáze sa myslí, či je cesta k súboru v tabuľke *watched* a procesu v tabuľke *requestors*.

[35] Veľkosť dátového typu wide charakter (WCHAR) je dvojnásobná oproti dátovému typu CHAR. *sizeof(WCHAR) =2x sizeof(CHAR)*. WCHAR je označenie pre jazyk C, v C++ je to wchar_t.

[36] *FILE_INFO* štruktúra je posielaná užívateľskej aplikácii.

Užívateľská aplikácia po prijatí správy musí upraviť reťazec do formy, ktorú pochopí užívateľ. Prijatý reťazec obsahuje cestu v tvare NT[37], ktorý sa používa v móde jadra. Tento tvar vyzerá napríklad takto:

```
\Device\HarddiskVolume2\Windows\system32\notepad.exe
```

Táto cesta bežnému užívateľovi nič nepovie, preto som ju musel preložiť na tvar DOS, ktorý vyzerá nasledovne:

```
C:\Windows\system32\notepad.exe
```

O prekladanie názvu cesty sa stará funkcia *FixPath()*, ktorá si z reťazca vyjme označenie zväzku v NT názve (označené červenou). Potom zistí všetky NT názvy zväzkov na disku a ich odpovedajúce DOS názvy a porovnáva, ktorý sa zhoduje s NT reťazcom. Pri zhode vymení NT názov za DOS názov (označené zelenou), čím ho upraví do užívateľovi zrozumiteľnej formy.

Tento modul ďalej obsahuje funkcie na prevod reťazca so znakmi typu *wchar_t* na reťazec so znakmi typu *char* a spať, to z dôvodu vkladania dát do databázy a ich vyberania pre prípadné zobrazenie. Obsahuje taktiež funkciu na prevod dát typu *int* na *string*, ktorá sa využíva vytváraní SQL požiadavky.

8.3 Testy

V tejto kapitole popíšem testy vykonávané na výslednom produkte. Prvé tri podkapitoly sa zaoberajú funkčnou stránkou projektu. To znamená, či mnou implementovaná časť DLP systému dokáže ochrániť sledované dáta.

Kapitola 8.3.4 sa zaoberá výkonnostnými testami projektu. Úlohou týchto testov je ukázať funkčnosť celého projektu a overiť jeho časovú náročnosť.

Testy prebiehali na počítači s parametrami:

- Veľkosť pamäte: 3,00 GB.
- Procesor: Intel Pentium Dual @ 2.17 GHz.
- Operačný systém: Windows 7 Professional 32-bit with Service Pack 1.

[37] Skratka NT spočiatku označovala procesor N-ten, na ktorej bol Windows NT vyvíjaný. Neskôr sa Bill Gates vyjadril, že označuje pojem *New Technologies* (nové technológie). V dnešnej dobe nemá žiaden špecifický význam.

8.3.1 Ochrana premenovania

Počiatočné informácie:

- V databáze sledovaných súborov je uložený súbor s názvom C:\subor.txt.
- Pomocou prehliadača súborov sa užívateľ snaží zmeniť jeho názov.

Dôsledky:

- Minifilter zachytí požiadavku o premenovaní súboru, ktorá je typu IRP_MJ_SET_INFORMATION a prepošle ju užívateľskej aplikácií.
- Aplikácia nahliadne do tabuľky sledovaných súborov a zistí, že daný súbor nemôže byť premenovaný. Zaznamená pokus o premenovanie do logu, požiadavku zamietne a oboznámi s tým minifilter.
- Minifilter nastaví patričné hodnoty a ukončí požiadavku.

Tejto ochrane podliehajú iba súbory typu, ktorý minifilter sleduje, a ktoré sú obsiahnuté v databáze.

8.3.2 Ochrana otvárania

Počiatočné informácie:

- V databáze sledovaných súborov je uložený súbor s názvom C:\subor.txt.
- V databáze povolených procesov je mu priradený iba proces s názvom C:\notepad.exe
- Užívateľ sa ho snaží poslať prostredníctvom e-mailu niekomu inému.

Dôsledky:

- Minifilter zachytí požiadavku o otvorení súboru typu IRP_MJ_CREATE a prepošle ju užívateľskej aplikácií.
- Aplikácia nahliadne do tabuľky sledovaných súborov a zistí, že pre daný súbor a proces[38], ktorý ho otvára, neexistuje v tabuľke *map* príslušná asociácia. To zaznamená, že ide o priestupok voči politike definovanej v databáze a pokus sa zaznamená do logu, požiadavku zamietne a oboznámi s tým minifilter.
- Minifilter nastaví patričné hodnoty a ukončí požiadavku.

Požiadavka by bola povolená, ak by užívateľ otváral súbor C:\subor.txt pomocou procesu C:\notepad.exe, ktorých asociácia je v tabuľke *map*. Táto akcia je taktiež poznamenaná do logu, pre prípadné vyšetrovanie.

[38] Proces môže byť obsiahnutý v tabuľke *requestors*.

8.3.3 Ochrana kopírovania

Počiatočné informácie:

- V databáze sledovaných súborov je uložený súbor s názvom C:\subor.txt.
- Pomocou prehliadača súborov sa ho užívateľ snaží skopírovať.
- Prehliadač a súbor nemajú potrebnú asociáciu v databáze.

Dôsledky:

- Minifilter zachytí požiadavku o premenovaní súboru typu IRP_MJ_READ a prepošle ju užívateľskej aplikácií.
- Aplikácia nahliadne do tabuľky sledovaných súborov a zistí, že prehliadač súborov nemôže čítať tento súbor. Zaznamená preto tento pokus do logu a požiadavku zamietne a oboznámi s tým minifilter.
- Minifilter nastaví patričné hodnoty a ukončí požiadavku.

Tejto ochrane podliehajú iba súbory typu, ktorý minifilter sleduje, a ktoré sú obsiahnuté v databáze.

8.3.4 Testy rýchlosti behu programu

Pod pojmom testy rýchlosti behu programu myslím čas vyriešenia I/O požiadavky, počnúc jej zachytením minifiltrom, ktorý pošle informácie o požiadavke užívateľskej aplikácii. Aplikácia pomocou informácii uložených v databáze rozhodne o výsledku požiadavky a ten prepošle späť minifiltru.

Pre meranie času som použil DebugView[39], ktorý má každý výpis označený časovou známkou. Na obrázku 15 môžeme vidieť v stĺpci *Time* čas v sekundách, ktorý uplynul od prvého výpisu. Stĺpec *DebugPrint* obsahuje ladiaci výpis.

Všetky ladiace výpisy sú z mnou implementovaného minifiltra a značia:

- Výpis obsahujúci *IRP_CREATE START* označuje začiatok sledovanej požiadavky.
- Výpis obsahujúci *process* označuje NT cestu k procesu otvárajúcemu súbor.
- Výpis obsahujúci *file* označuje názov otváraného súboru.

Výpis obsahujúci *IRP_CREATE disALLOW EXIT* označuje ukončenie požiadavky a jej rozhodnutie. *disALLOW* znamená, že požiadavka bola zakázaná, *ALLOW* znamená, že požiadavka bola povolená.

Pri meraní som použil štyri typy veľkostí databáz. Každá databáza obsahuje *X* sledovaných súborov, *X* procesov, ktoré môžu otvárať procesy. Každému súboru je priradený jeden proces, to znamená, že v tabuľke *map* je takisto *X* záznamov. Hodnota *X* bola 10, 100, 1000, 10000.

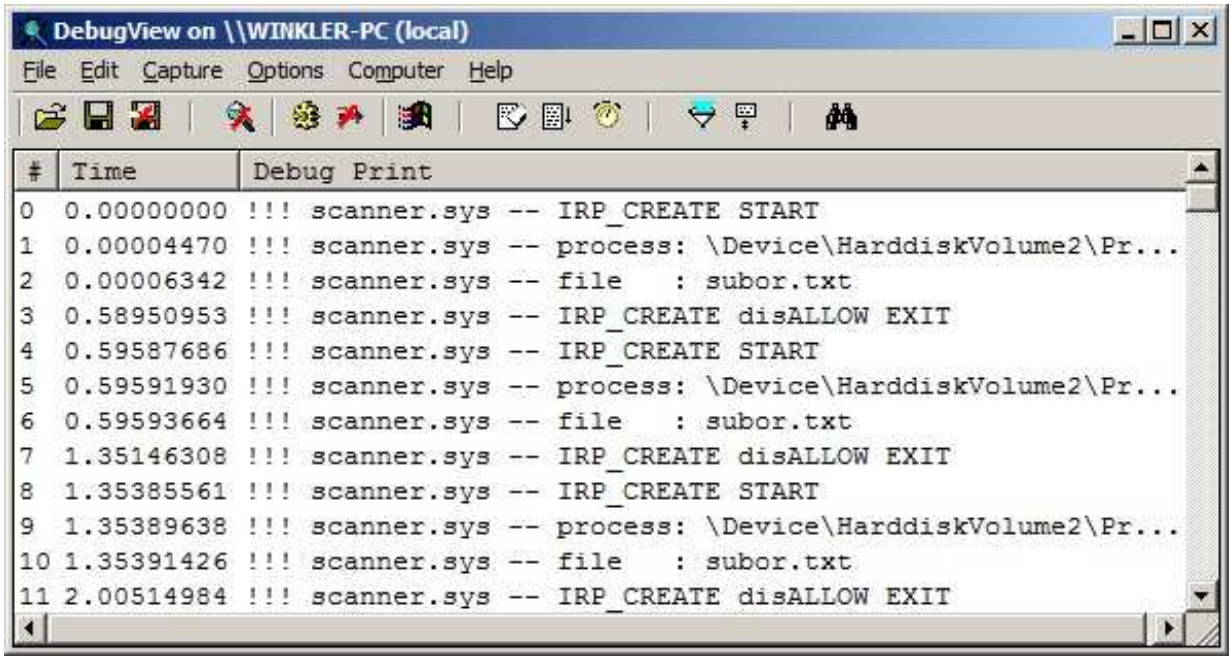
[39] DebugView je program, ktorý dovoľuje sledovať ladiace výpisy módu jadra a WIN32.

Najhorší prípad nastáva ak súbor a proces, ktorý ho otvára, majú najvyššie[40] ID a priradenie týchto identifikátorov je taktiež na poslednej pozícii tabuľky *map*.

Pre program to znamená, že ak potrebuje zistiť či sa súbor, proces a ich priradenie nachádzajú v databáze, musí prehľadať všetky tri tabuľky od začiatku až do konca. Každý test bral v úvahu tento najhorší prípad.

Hlavný problém, ktorý pri meraniach nastal, bol, že pri otváraní súboru, premenovávaní alebo kopírovaní sa posielal viackrát rovnaký typ správy s identickým obsahom. Na obr.15 môžeme vidieť prvé tri správy, ktoré sa posielali pri pokuse otvoriť *súbor.txt* nepovoleným procesom, konkrétne sa jednalo o *wordpad.exe*. Na obrázku sú znázornené iba prvé tri I/O požiadavky *wordpad.exe* ich presnejšie posiela štyri. Program na prezeranie súborov *mspaint.exe* posiela sedem požiadaviek a textový editor *notepad.exe* ich posiela 3. Je to individuálne od každého procesu a neprišiel som na možnosť to nejakým spôsobom ovplyvniť.

Pri testoch som meral stále iba prvú požiadavku, lebo je jedno či prejde minifiltrom jedna požiadavka, alebo ich prejde sedem. Čas prechodu všetkých identických požiadaviek je približne zhodný.



Obrázok 15: Príklad výpisu programu DebugView, pri meraní behu programu.

Vykonané testy sú:

1. Čas povolenia otvorenia sledovaného súboru.
2. Čas zakázania otvárania sledovaného súboru.
3. Čas otvárania nesledovaného[41] súboru.

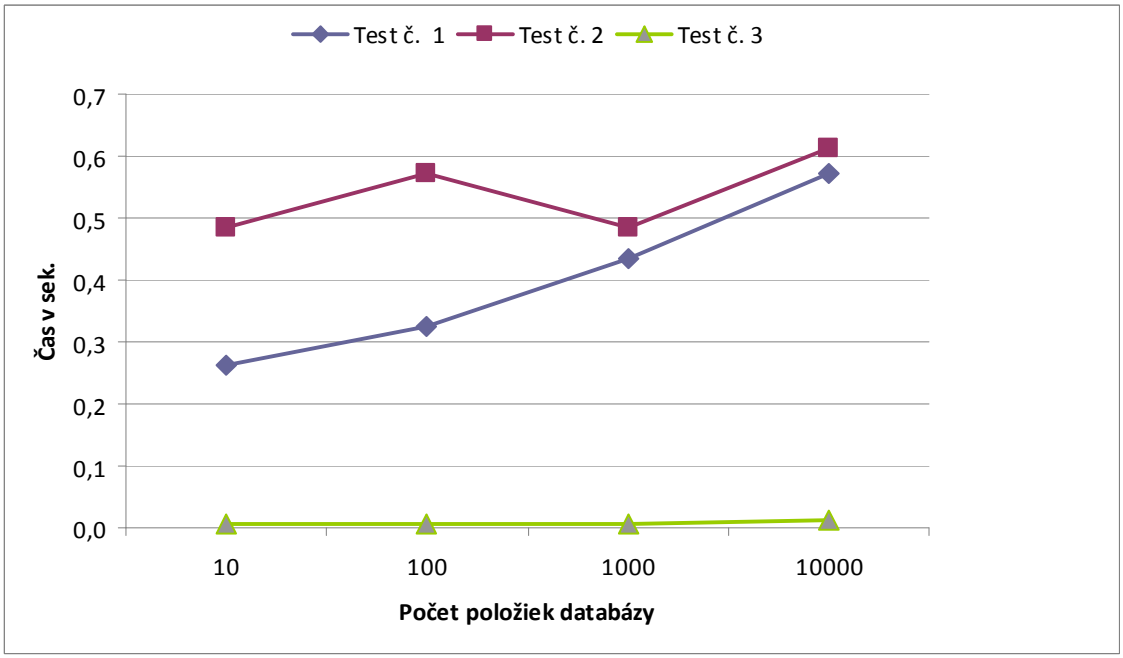
[40] Najvyšší index má posledný prvok v tabuľke.

[41] Nesledovaný súbor je taký, ktorý nie je v tabuľke *watched*.

Tabuľka 3: Priemerné výsledné hodnoty testov v sekundách.

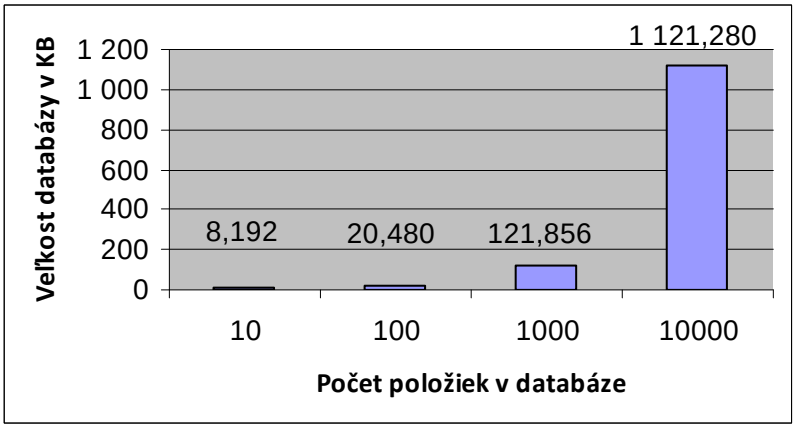
Test č. / X	10	100	1000	10000
1.	0,262544	0,323692	0,435216	0,571797
2.	0,483413	0,571797	0,485673	0,612182
3.	0,004892	0,005872	0,006368	0,012467

Prvé dva testy určujú výkonný čas, za ktorý sa program musí rozhodnúť, či povolí alebo zakáže akciu. Tretí test určuje čas, ktorý potrebuje I/O požiadavka nesledovaného súboru, aby prešla programom. Každý test bol vykonaný 20-krát a zo zistených hodnôt som vypočítal priemernú hodnotu. Výsledky sú znázornené v tabuľke 3 a na grafe 1.



Graf 1: Výsledné hodnoty testov v sekundách.

Pri behu programu som sledoval aj veľkosť využívanej pamäte. Hoci pri každom teste mala databáza inú veľkosť (graf 2), program zaberal stále rovnako veľké miesto v pamäti cca. 1 000 KB.



Graf 2: Veľkosť databáz v KB pre X prvkov v databáze.

8.3.5 Vyhodnotenie testov

Z vykonaných testov vyplýva, že program ovplyvňuje dĺžku doby výkonu I/O požiadavky, ktorá súvisí so sledovanými súbormi. Pri prehliadaní databáz bol najvyšší čas obsluhy požiadavky pri databáze s obsahom 10000 prvkov 0,7 s a priemerný čas bol 0,61 s. Tento čas je adekvátny zvýšeniu bezpečnosti požadovaných dát. Čas obsluhy je priamoúmerný počtu položiek v databáze.

Veľkým plusom tejto aplikácie je, že čas vybavenia požiadavky, ktorá nesúvisí s prvkami v databáze je veľmi rýchly (0,01 s pre 10000 položiek v databáze).

8.4 Možnosti ďalšieho rozšírenia

Aplikácia implementuje časť systému DLP, ktorá zabezpečuje bezpečnosť súborového systému na základe cesty k jednotlivým súborom. Výhodou tejto implementácie je, že aplikácia nemusí prehliadať obsah súboru, ale rozhodne sa iba podľa jeho umiestnenia v systéme. Týmto riešením som zaistil, že o povolení požiadavky sa rozhodne v rovnakom čase, či už ide o súbor veľkosti niekoľko KB alebo GB.

Sledovaný súbor je zabezpečený voči premenovaniu alebo prekopírovaniu na nesledované miesto. A všetky akcie prevádzané so sledovaným súborom sú zaznamenávané.

Jedným z vylepšení tejto aplikácie by mohlo byť odstránenie alebo potlačenie viacnásobného výskytu identickej požiadavky. Riešením by mohol byť napríklad zoznam posledných pár akcií a informácií o ich povolení, ktorý by vlastnil minifilter a pred posielaním požiadavky by nahliadol do tohto zoznamu, či tam nenájde danú dvojicu informácií. Toto riešenie by odstránilo posielanie duplicitných požiadaviek užívateľskej aplikácií.

Iným rozšírením by mohla byť kontrola obsahu súboru pomocou čiastočnej zhody alebo regulárnych výrazov.

9 Záver

Táto práca sa zaoberá problematikou monitorovania užívateľov a procesov pomocou systému DLP. Systém Data Loss Prevention (prevencia straty dát) sa snaží zabraňovať úniku citlivých dát z vnútra aj zvonku organizácie. Prevencia straty dát je v dnešnej dobe stále pomerne neznámy pojem. Ak by som mal DLP presnejšie definovať, povedal by som, že sa viac zameriava na problematiku úniku dát z vnútra organizácie, ale má dobré výsledky aj pri odhaľovaní nových tzv. zero-day útokov. Systém DLP neposkytuje 100% ochranu pred únikom dát, snaží sa hlavne predchádzať neúmyselnému zverejneniu utajovanej informácie (čísla kreditných kariet, zdravotné záznamy a i.).

Na začiatku tejto práce sme definovali, čo je únik a aké sú spôsoby jeho realizácie. Užívateľ s dostatočnými vedomosťami, schopnosťami, ale hlavne motiváciou (najčastejšie finančnou alebo pre obohatenie vlastného know-how) je schopný pomocou nemonitorovanej komunikácie alebo nezabezpečeného systému spôsobiť firme značné škody.

V tretej kapitole popisujem systém DLP a jeho hlavné časti, ktorými sú správa podnikových politík, klasifikácia dát a monitorovanie aktivít užívateľov. Popisujem v nej rozdiel medzi analýzou kontextu a obsahu. V jednoduchosti na príklade obálky s listom, kontextová analýza skúma obálku a analýza obsahu skúma jej vnútro, čiže list. V závere kapitoly rozdeľujem monitorovanie na dva celky, monitorovanie siete a koncových staníc. Oba typy sa navzájom dopĺňajú a spolu vytvárajú srdce systému.

V ďalších dvoch kapitolách som sa zaoberal popisom správania užívateľov a procesov a možných komerčných spôsobov ich monitorovania. Produkty, ktoré sú dostupné na trhu sa od seba veľmi nelíšia stavbou DLP systému, ich hlavnými odlišnosťami je spôsob analýzy obsahu a vytvárania pravidiel.

Po prvotnej analýze som sa zameral na aplikáciu DLP z pohľadu jej personálneho nasadenia na koncovú stanicu. Cieľom tejto aplikácie je chrániť dáta užívateľa pred ich neúmyselnou stratou alebo odcudzením škodlivým softvérom.

Naimplementoval som ovládač, presnejšie minifilter ovládač súborového systému, a užívateľské rozhranie na riadenie aplikácie. Minifilter filtruje vstupno-výstupné požiadavky smerujúce z užívateľského prostredia smerom k súborovému systému. Zisťuje informácie o požiadavke, ktoré preposiela užívateľskej aplikácii. Aplikácia disponuje databázou sledovaných súborov a procesov, ktoré môžu jednotlivé súbory otvárať. Taktiež obsahuje funkcionalitu na modifikovanie databázy. Prehľadaním databázy sledovaných súborov aplikácia rozhodne o povolení či zakázaní požiadavky.

Aplikácia ako celok (užívateľské prostredie aj minifilter) je schopná nadefinovať konkrétnemu súboru procesy, ktoré ho môžu otvárať a tým je zaručené, že ho iný proces neotvorí. V aplikácii je

implementovaná aj ochrana proti kopírovaniu sledovaných súborov a ochrana proti ich premenovávaniu.

Pri testoch bolo preukázané, že všetky tri ochrany sledovaných súborov sú funkčné a spoľahlivé. Ďalším testovaním som zistil, že minifilter takmer vôbec neovplyvňuje čas otvorenia súboru, ktorý nie je sledovaný a iba v minimálnej miere ovplyvňuje otvorenie sledovaného súboru. 0,01 s je priemerný čas, ktorým prejde požiadavka nesledovaného súboru minifiltrom a 0,6 s je priemerný čas, ktorý potrebuje minifilter spolu s užívateľskou aplikáciou, aby rozhodli o jej povolení. Tento výsledok by som zhodnotil ako veľmi pozitívny z dôvodu nízkeho zaťaženia systému.

Jedným z možných vylepšení implementovaného riešenia by mohla byť analýza obsahu čiastočnou zhodou alebo pomocou regulárnych výrazov. Vhodným zlepšením by bolo aj zefektívnenie vyhľadávania v databáze.

Táto aplikácia umožňuje docieliť, aby bežný užívateľ nebol schopný nepovolene narábať so sledovanými dátami. Užívateľ s potrebnými skúsenosťami a znalosťami by možno vedel obísť implementovanú ochranu. Prípadným ďalším zvýšením ochrany pred skúseným útočníkom by bolo implementovať iné prvky ochrany (heslá a pod.).

Literatúra

[1] *DataLossDB open security foundation*. [online]. aktualizované 2010-11-10 [cit. 2011-01-05]. DataLossDB. Dostupné na URL: <http://datalossdb.org>

[2] *Secit.sk*.[online]. aktualizované 2010-11-17 [cit. 2011-01-05]. Objavil sa nový nebezpečný Trójan. Dostupné na URL: <http://www.secit.sk/sk/content/objavil-sa-novy-nebezpecny-trojan>

[3] GRANGER, Sarah. *Symantec* [online]. 17. december 2001 [cit. 2011-01-19]. Social Engineering Fundamentals Part I: Hacker Tactics . Dostupné z WWW: <www.symantec.com/connect/articles/social-engineering-fundamentals-part-i-hacker-tactics>

[4] PROCTOR, Paul; OULLER, Eric. *Symantec* [online]. 2. jún 2010 [cit. 2011-01-19]. Quadrant for Content-Aware Data Loss Prevention . Dostupné z WWW: <http://www.symantec.com/content/en/us/about/media/industryanalysts/Gartner_Content_Aware_DLP_MQ_June2010.pdf>

[5] *Argentra*. [online]. [cit. 2011-01-05]. Oracle IRM. Dostupné na URL: <http://www.argentra.com/cms4/products/oracle/oracle-irm.html>

[6] *Websense* [online]. [cit. 2011-01-05]. Websense: PreciseID Technology. Dostupné na URL: <http://www.websense.com/content/PreciseID.aspx >

[7] *Symantec*. [online]. aktualizované 2010-02 [cit. 2011-01-05]. Symantec: Symantec Data Loss Prevention Insight. Dostupné na URL: <http://eval.symantec.com/mktginfo/enterprise/ fact_sheets/b-symc_dlp_data_insight_DS-21005381.en-us.pdf >

[8] CHEE, Wong Onn. Information Leakage, Detection, and Prevention. *The Global Voice Of Information Security*. December 2007, 12, s. 37-39.

[9] CHAPPLE, Mike. *Do information leak prevention products protect critical data?* [online]. 2005 [cit. 2011-05-13]. Dostupné z WWW: <http://searchsecurity.techtarget.com/answer/Do-information-leak-prevention-products-protect-critical-data>

[10] *Garter*. [online]. 2006 [cit. 2011-05-13]. Proofpoint Positioned in 'Challengers' Quadrant in Content Monitoring and Filtering Magic Quadrant. Dostupné z WWW: <http://goliath.ecnext.com/coms2/gi_0199-5276054/Proofpoint-Positioned-in-Challengers-Quadrant.html>

[11] *Oracle* [online]. 2007 [cit. 2011-05-13]. Oracle® Fusion Middleware Administrator's Guide for Oracle IRM Server. Dostupné z WWW: <http://download.oracle.com/docs/cd/E17904_01/admin.1111/e12321/isvaintroducingoirm.htm>

[12] *Garter* [online]. 2011 [cit. 2011-05-13]. Gartner.com. Dostupné z WWW: <http://www.gartner.com/technology/about.jsp>

[13] *OSR*. [online]. 6. august. 2010 [cit. 2011-05-13]. File Systems and Filters: A Specialty . Dostupné z WWW: <http://www.osronline.com/article.cfm?article=565>

[14] *MSDN* [online]. 15. apríl. 2002 [cit. 2011-05-13]. Windows Driver Model (WDM). Dostupné z WWW: http://msdn.microsoft.com/en-us/windows/hardware/gg463453

[15] *MSDN* [online]. 2. august 2006 [cit. 2011-05-13]. Handling IRPs: What Every Driver Writer Needs to Know. Dostupné z WWW: <http://msdn.microsoft.com/en-us/windows/hardware/gg487398>

[16] *MSDN* [online]. 5, marec. 2011 [cit. 2011-05-13]. IRP. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/ff550694%28v=vs.85%29.aspx>
[17] *OSR* [online]. 11. apríl 2003 [cit. 2011-05-13]. IO_STACK_LOCATION. Dostupné z WWW: <http://www.osronline.com/ddkx/kmarch/k112_49bm.htm>
[18] *MSDN* [online]. 21. jún 2007 [cit. 2011-05-13]. File System Minifilter Load Order Groups and Altitude Ranges. Dostupné z WWW: <http://msdn.microsoft.com/en-us/windows/hardware/gg462963>
[19] *MSDN* [online]. 28. september 2010 [cit. 2011-05-13]. FLT_REGISTRATION Structure. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/ff544811%28v=vs.85%29.aspx>
[20] *MSDN* [online]. 12. apríl 2010 [cit. 2011-05-13]. File System Minifilter Allocated Altitudes. Dostupné z WWW: <http://msdn.microsoft.com/en-us/windows/hardware/gg462961>
[21] *MySQL* [online]. 2011 [cit. 2011-05-15]. MySQL :: The world's most popular open source database. Dostupné z WWW: <http://www.mysql.com/>
[22] *Laptops Area* [online]. máj 2008 [cit. 2011-05-16]. McAfee Endpoint Encryption. Dostupné z WWW: <http://www.laptopsarena.com/mcafee-endpoint-encryption/>.
[23] *MSDN* [online]. 28. september 2010 [cit. 2011-05-16]. Creating an INF File for a Minifilter Driver. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/ff540045%28v=vs.85%29.aspx >
[24] *Microsoft connect* [online]. 2011 [cit. 2011-05-16]. Minifilter Altitude Allocation. Dostupné z WWW: <http://connect.microsoft.com/Survey/Survey.aspx?SurveyID=2361&SiteID=221>
[25] *OSRonline* [online]. 2006 [cit. 2011-05-16]. What's in a (Process) Name? Obtaining A Useful Name for the Executable Image in a Process . Dostupné z WWW: <http://www.osronline.com/article.cfm?article=472>
[26] *MSDN* [online]. 2007 [cit. 2011-05-16]. C++ for Kernel Mode Drivers: Pros and Cons. Dostupné z WWW: <http://msdn.microsoft.com/en-us/windows/hardware/gg487420>
[27] *MSDN* [online]. 2011 [cit. 2011-05-16]. MSDN. Dostupné z WWW: <http://msdn.microsoft.com>
[28] *MSDN* [online]. 2011 [cit. 2011-05-16]. About the Windows Driver Kit (WDK). Dostupné z WWW: <http://msdn.microsoft.com/en-us/windows/hardware/gg487428>
[29] *MSDN* [online]. 2011 [cit. 2011-05-16]. IFS Kit - Installable File System Kit. Dostupné z WWW: <http://msdn.microsoft.com/en-us/windows/hardware/gg463062>
[30] *MSDN* [online]. 5. marec 2011 [cit. 2011-05-18]. Understanding PREfast for Drivers. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/ff554010%28v=vs.85%29.aspx>
[31] *MSDN* [online]. 28. september 2010 [cit. 2011-05-18]. NTSTATUS values. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/cc704588%28v=prot.10%29.aspx>
[32] *SQLite* [online]. 2011 [cit. 2011-05-18]. SQLite. Dostupné z WWW: <http://www.sqlite.org/>
[33] *Symantec* [online]. 2011 [cit. 2011-05-19]. Data Loss Prevention. Dostupné z WWW: <http://www.symantec.com/en/uk/business/solutions/solutiondetail.jsp?solid=sol_info_risk_comp&solfid=sol_data_loss_prevention>
[34] *McAffe* [online]. 2011 [cit. 2011-05-19]. McAfee Host Data Loss Prevention. Dostupné z WWW: <http://www.mcafee.com/us/products/host-data-loss-prevention-dlp.aspx>
[35] *Websense* [online]. 2011 [cit. 2011-05-19]. Data security. Dostupné z WWW: <http://www.websense.com/content/data-security-solutions.aspx>

[36] *RSA* [online]. 2011 [cit. 2011-05-19]. RSA Data Loss Prevention. Dostupné z WWW: <http://www.rsa.com/node.aspx?id=3426>

[37] *Trustwave* [online]. 2011 [cit. 2011-05-19]. Data Loss Prevention. Dostupné z WWW: <https://www.trustwave.com/data-loss-prevention/>

[38] *Windows Sysinternals* [online]. 16. október 2008 [cit. 2011-05-20]. DebugView for Windows v4.76. Dostupné z WWW: <http://technet.microsoft.com/en-us/sysinternals/bb896647>

[39] *MSDN* [online]. 2011 [cit. 2011-05-21]. Driver Signing Requirements for Windows. Dostupné z WWW: <http://msdn.microsoft.com/en-us/windows/hardware/gg487317.aspx>

Prílohy

Príloha 1. CD so zdrojovými kódmi.