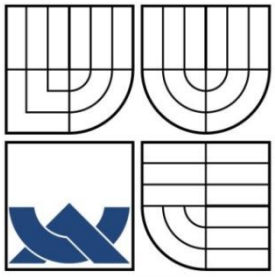
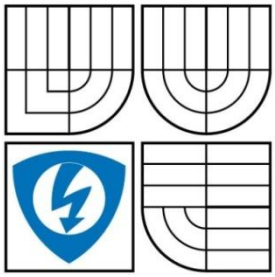


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

ODHALOVÁNÍ ROOTKITŮ A DETEKCE SPYWARU

UNCOVERING OF ROOTKITS AND DETECTION OF SPYWARE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — STANISLAV JURAS
AUTHOR

VEDOUCÍ PRÁCE — ING. MICHAL POLÍVKA
SUPERVISOR

BRNO 2008

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Stanislav Juras
Bytem: Řadová 878/17, 691 45, Podivín
Narozen/a (datum a místo): 7. 8. 1986, Hustopeče

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 244/53, 602 00, Brno
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Ing. Kamil Vrba, Csc.

(dále jen „nabyvatel“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - □ disertační práce
   - □ diplomová práce
   - ☑ bakalářská práce
   - □ jiná práce, jejíž druh je specifikován jako .....

   (dále jen VŠKP nebo dílo)

Název VŠKP: Odhalování rootkitů a detekce spywaru
Vedoucí/ školitel VŠKP: Ing. Michal Polívka
Ústav: Ústav telekomunikací
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

- ☑ tištěné formě – počet exemplářů 1
- ☑ elektronické formě – počet exemplářů 1

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

4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

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

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

Článek 3
Závěrečná ustanovení

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

V Brně dne: …….

…….. ……

Nabyvatel Autor

Anotace

Bakalářská práce nastiňuje problematiku odhalování rootkitů a detekci spywaru. Popisuje základní druhy známého spywaru a rootkitů. U spywaru se jedná převážně o popis činnosti jednotlivých druhů. V případě rootkitů se jedná hlavně o popis módů a způsobu jejich infekce. Jsou zde také nastíněny pokusy o legální použití rootkitů. Dále jsou zde shrnuty základní metody detekce spywaru a rootkitů, které se dnes běžně používají v různých detekčních programech.

Součástí práce je také praktická realizace (program) jedné z metod detekce spywaru. Program je navržen tak, aby byl schopen odhalit jednoduchý vzorek spywaru, který je uložený v jeho databázi. Jako metoda pro detekci je zvolena metoda detekce podle vzorku kódu. Pomocí uživatelského prostředí programu je možné si vybrat jednotkou, která má být testována.

Klíčová slova

Spyware, rootkit, sběr informací, hookování, patchování, uživatelský režim, režim jádra, detekce spywaru, detekce rootkitů, IDS.

Abstract

Bachelor’s thesis is about uncovering of rootkit and detection of spyware. It describes the basic types of known spyware and rootkits. Section dealing with spyware is especially about a description of each species. In case of rootkit the thesis is mainly about description of modes and the manner of their infection. There are also outlined attempts to use legal rootkit. In other case there are summarized the basic methods of rootkit and spyware detection, which are commonly used in various detection programs.

The second part of thesis is practical implementation (the program) of one of the methods of spyware detection. The program is designed to be able to detect a simple pattern of spyware, which is stored in its database. The program uses the file signature detection. It contains also the graphical user interface, where is possible to choose a unit that user want to test.

Key words

Spyware, rootkit, gathering information, hooking, patching, user mode, kernel mode, detection of spyware, detection of rootkit, IDS.

JURAS, S. *Odhalování rootkitů a detekce spywaru.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 57 s. Vedoucí bakalářské práce Ing. Michal Polívka.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „ODHALOVÁNÍ ROOTKITŮ A DETEKCE SPYWARU“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne …… ……..

Stanislav Juras

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Ing. Michalu Polívkovi za pedagogickou a odbornou pomoc a také za cenné rady při zpracování bakalářské práce.

V Brně dne…… ……
Stanislav Juras

Obsah:

Seznam obrázků:

1. Úvod

Internet je obrovské místo a na nezkušeného uživatele zde mohou číhat mnohá nebezpečí. Běžný uživatel poznává Internet prostřednictvím různých služeb. Asi nejběžnější službou je prezentace jedinců, případně skupin, prostřednictvím takzvaných WWW stránek. Samozřejmě se hojně využívají i jiné služby, například komunikace po internetu (emaily, konference, telefonování atd.), sdílení a přenos dat a spousty dalších. Dalo by se říct, že se Internet stává nepřehlednou změtí všech možných služeb a jiných věcí.

Jsou snahy vnést do tohoto světa Internetu pořádek. Za tímto účelem vznikly/vznikají různé standardy od různých organizací, které se snaží tento svět zpřehlednit, zjednodušit a jinak přizpůsobit k porozumění neznalých uživatelů. Na druhou stranu jsou tu i tací, kterým tato nepřehlednost a anonymita vyhovuje. Tito lidé, nebo skupiny lidí, se snaží napadat nepřipravené uživatele (respektive společnosti). Nepřipravené na to, že je někdo může odposlouchávat, špehovat, krást data, případně je jinak poškozovat. Tito „útočníci“ to dělají buď pro zábavu a radost z cizího utrpení nebo za jiným účelem (např. sebe obohacení), pravý důvod znají jen oni sami. Ke své činnosti využívají různých nástrojů. Ať už se jedná o různé druhy spywaru, trojských koní, virů a v poslední době i rootkitů.

Vzniká tedy potřeba se proti těmto hrozbám nějakým způsobem bránit. Musí se vymýšlet neustále nové sofistikované postupy na jejich odhalování. Samozřejmě i útočníci se snaží nezůstat pozadu a neustále své produkty upravují, vylepšují. To vše vyústilo v neustálý, nekompromisní a nekončící boj mezi útočníky a „obránci“. Jedni se snaží najít nové způsoby infiltrace a druzí se snaží vytvořit dokonalou obranu. Dá se říci, že obě strany jsou více či méně úspěšné a nelze tedy s jistotou určit, která strana je na tom lépe.

Pro běžného řadového uživatele počítače to znamená dávat si pozor při „brouzdání“ internetem. Nebezpečí může číhat prakticky všude, při otvírání elektronické pošty, při stahování dat/aplikací nebo jen při běžném prohlížení internetových stránek.

2. Spyware

2.1 Definice spywaru

Obecně můžeme spyware (spy = špion) definovat jako aplikaci, která se do počítače dostane ať už nechtěnou náhodou nebo jako součást jiných aplikací (podrobnosti viz kapitola 2.1.1). Všeobecně se spyware dostal do podvědomí jako aplikace, která má za úkol narušit soukromí uživatele. Dokáže shromažďovat statistické informace (navštívené stránky, používané programy apod.), případně i jinak citlivé informace – jako jsou například hesla, piny platebních karet atd. Takto nashromážděná data jsou následně poslána autorovi této aplikace a ten je použije dle svého uvážení.

Spyware lze tedy chápat jako převážně nechtěnou aplikaci, která se skrytě dostane do systému a následně shromažďuje a odesílá data o daném uživateli.

2.1.1 Sběr informací

Sběr informací je převažující činnost, kterou v dnešní době spyware vykonává.

Spyware může být součástí shareware, free programů jako cena za bezplatné využívání. Výjimečně může být součástí i placených programů, ale to není již tak časté. Nehledě na způsob přenosu, má spyware obvykle za úkol sbírat statistické informace, jako jsou např. informace o navštívených internetových stránkách, používaných programech, nákupních zvycích atd. Tvůrci se hájí tím, že chtějí dnešnímu uživateli nabízet jen to, o co má konkrétně zájem. Pokud budeme uvažovat např. tzv. adware (viz kapitola 2.2.1), mohli by tito tvůrci říct, že chtějí uživateli směřovat jen reklamu, která je v okruhu jeho zájmu, šlo by tedy o tzv. „cílenou reklamu“. Tato činnost je z právního hlediska legální.

Na druhou stranu může spyware sbírat statistické informace mnohem citlivějšího rázu, jako například uživatelská jména a hesla, piny kreditních karet anebo dokonce může sloužit jako zadní vrátka (= backdoor) do uživatelova systému. Tato činnost je jiným extrémem a ukazuje na nelegálnost toho programového kódu.

Hranice mezi „legálním“ a „nelegálním“ spywarem je velice tenká a nikdo nemůže s přesností říct, kdy jde o legální a kdy o nelegální činnost. Pokud budeme opět uvažovat např. adware, mohlo by se říct, že velká většina dnešních běžných uživatelů má k jakékoliv reklamě negativní postoj a legální spyware tedy zcela odmítá. Informace o přítomnosti spywaru by měli být součástí licenčních ujednání, které musí uživatel před začátkem instalace programu odsouhlasit, bohužel mnoho uživatelů si tato ujednání nepročte celé nebo vůbec.

2.2 Kategorie spyware

K upřesnění by se dalo říci, že se jedná o produkty, které pracují na bázi spyware nebo alespoň splňují některou z jeho vlastností. Čerpáno je zejména z pramenů [10], [20], [26].

2.2.1 Adware

Adware (advertising-supported software) je označení pro produkty, které znepříjemňující práci na počítači pomocí ať už vyžádané nebo nevyžádané reklamy. Nepatří mezi nebezpečné, ale úspěšně se mu daří zpomalovat prohlížení webových stránek a celkově činnost s počítačem spojenou. Děje se tak díky neustále vyskakujícím (pop-up) oknům zobrazujících reklamu. Dalším projevem může být změna domovské stránky v prohlížeči.

Adware může být součástí některých programů výměnou za bezplatné využívání nadstandardních funkcí, které v původní verzi nejsou k dispozici. Tvůrci programů skutečnost přítomnosti adwaru obvykle zahrnují do licenčních ujednání, které si je možné přečíst před instalaci produktu, například firmou Microsoft používaná tzv. „EULA“ (End User Licence Agreement = Licenční smlouva s koncovým uživatelem).

2.2.2 Dialer

Dialer postihuje majitele modemových připojení. Jde o program, který mění druh připojení k Internetu. Myšleno jako změna běžně používaného telefonního čísla pro Internetové připojení na číslo se zvláštní tarifikací (až několik desítek korun za hodinu).

V určitých případech se může jednat o legální činnost, například pokud tímto způsobem uživatel platí za nějaké služby. Za hranicí legálnosti je situace, jakmile je informace o přesměrování na dražší linku v nějakém cizím jazyce. Nelegální případy jsou samozřejmě také všechny, u kterých dochází k přesměrování bez vědomí uživatele.

2.2.3 Keylogger

Mohou být buď hardwarové, nebo softwarové. O spyware se jedná v případě softwarových keyloggerů. Nejsou přímo nebezpečné pro počítač. Jejich úkolem je zachytávat veškeré stisknuté klávesy na klávesnici. Výstup této činnosti může být například textový, ten se následně přeposílá tvůrci nebo původci této nákazy. Jsou využívány především k získávání hesel.

2.2.4 Backdoor

Jak již napovídá název „backdoor“ (= zadní vrátka) jedná se o program, který umožňuje vzdálený přístup do počítače. Útočník pomocí tohoto programu může obejít autentizaci a skrytě tak pracovat na vzdáleném počítači.

Princip programu je na bázi komunikace klient-server. Na vzdáleném počítači musí být nainstalována serverová část programu, potom může klient - útočník klást různé požadavky na server a ten mu je vyplní.

2.2.5 Malware

Vznikl složením slov Malicious (= zákeřný, škodlivý) software. Dalo by se ale říci, že se spíše jedná o takové všeobecné označení pro veškeré škodlivé softwarové produkty, které mohou počítač potkat. Tímto termínem se spíše označuje záměr nežli produkt.

3. Rootkyty

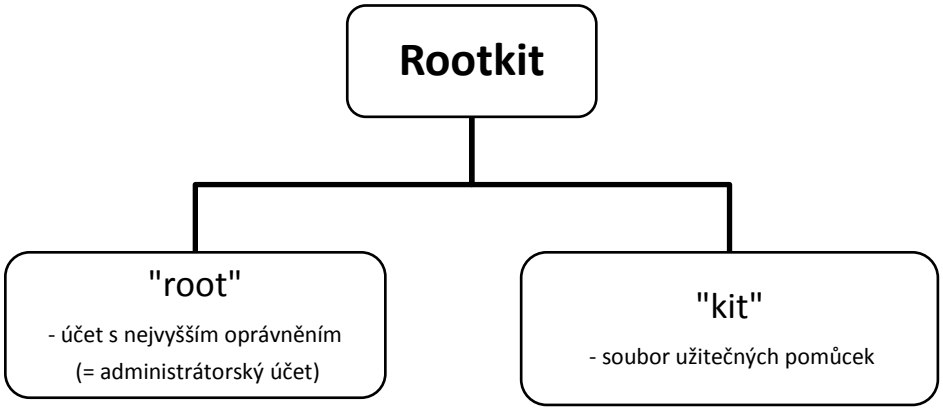
Aby mohl útočník pokračovat ve využívání jím napadeného počítače, měl by na něj nainstalovat backdoor. Potom bude moci například kontrolovat činnost napadeného počítače, případně do jeho činnosti i zasáhnout.

Mnohem jednodušší metodou pro zajištění utajení činnosti na počítači, je za využití rootkitů. V kapitole je čerpáno zejména z pramenů [3], [18], [25].

3.1 Definice rootkitu

Z laického pohledu by se dal rootkit popsat jako program (viz obr. 1), který se snaží skrýt jak svou činnost, tak i pro útočníka nežádoucí aplikaci. Největší nebezpečí tedy spočívá v tom, že tato aplikace zamaskuje působení škodlivého softwaru před uživatelem. Ten si potom není vědom, že se stal obětí útoku a nepodnikne tedy potřebné kroky k odstranění této infekce.

Rootkit se tedy dá na(ne)štěstí instalovat pouze na úspěšně napadený systém. Důležitým faktorem obrany je tedy samotná prevence (viz podkapitola 4.3).



Obr. 1: K vysvětlení pojmu rootkit.

3.2 Módy rootkitu v operačním systému

V operačním systému Windows existují dva možné režimy – režim jádra a uživatelský režim. Zatímco v uživatelském režimu jsou práva k práci v systému jen omezená, v režimu jádra jsou práva prakticky neomezená. Většina aplikací běží v uživatelském režimu, v režimu jádra běží veškeré systémové procesy, ovladače zařízení. Proces běžící v režimu jádra může změnit nebo

dokonce zničit datové struktury v paměti. V uživatelském režimu tyto činnosti provádět nejdou, lze ale přistupovat k adresovému prostoru daného procesu (k instrukcím a datům daného programu) [3].

Rootkit v uživatelském režimu může být buď jako samostatný program, nebo může být součástí jiného programu. Kdežto v režimu jádra má rootkit větší možnosti působení v systému nežli v uživatelském režimu.

Ať už je tedy naprogramován jakkoliv, jeho účelem je skrývat aplikace, soubory případně adresáře, klíče v registrech nebo procesy. Většinou ale slouží k filtrování, úpravě informací, které posílá operační systém uživateli.

Pokud bude chtít například uživatel provést výpis jistého adresáře, rootkit zachytí volání aplikace, odchytí posílaný výpis od operačního systému, pozmění jej podle sebe a pozměněný nebo zcela jiný výsledek odešle uživateli/aplikaci.

3.3 Typy rootkitů

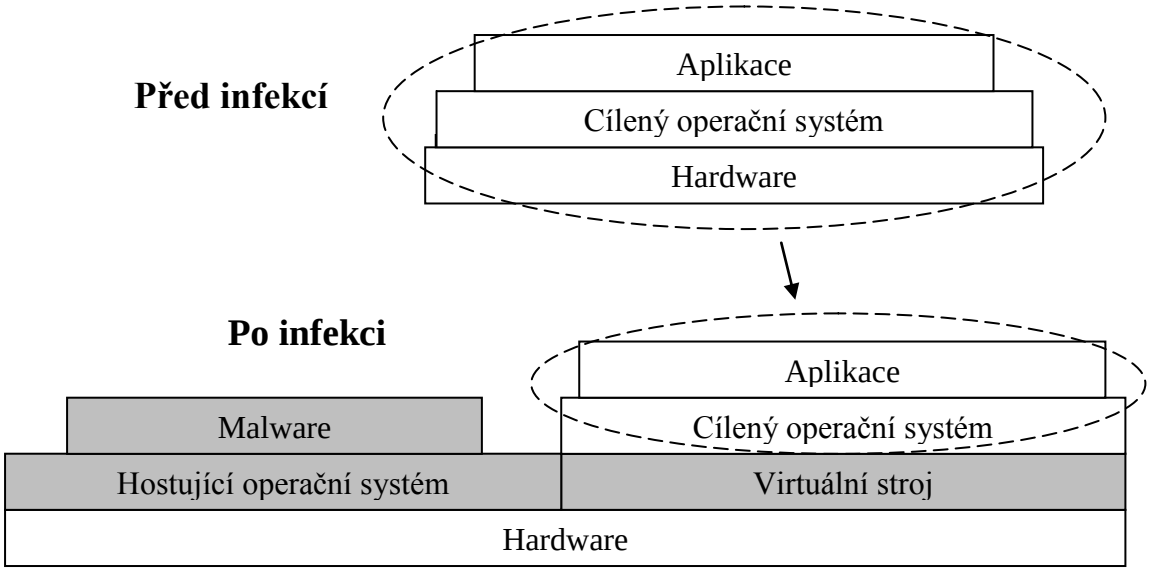
Ve skutečnosti ale existují celkem čtyři různé druhy rootkitů, rozdělených podle toho, na které úrovni systému pracují [18].

3.3.1 Virtuální rootkity (=VMBR)

VMBR (Virtual Machine Based Rootkit) pracuje na té nejnižší úrovni, na které jsou rootkity schopny pracovat, a proto patří k velmi obtížně zjistitelným. Tyto rootkity upraví pořadí při bootovaní počítače. Nejdříve se spustí BIOS, který dále spustí inicializaci rootkitu, místo toho aby přímo spustil operační systém. Jakmile se rootkit nahraje do paměti, spustí původní operační systém, ale pouze v režimu virtuálního počítače. Uživatel tedy bude pracovat s právy hosta (guest). Tím si rootkit zajistí odchytávání veškerých hardwarových volání od operačního sytému (viz obr. 2).

Jak již bylo psáno výše, takto zavedený rootkit je obtížně zjistitelný, pokud totiž uživatel nemá ponětí o tom, že jeho počítač byl infikován, tak nemůže ani podniknout nezbytné kroky k odstranění této infekce. Jedním z nejtypičtějších příznaků infekce tímto rootkitem je zpomalení výkonu počítače.

Příkladem takového typu rootkitu je „SubVirt“ [12], který byl vyvinut firmou Microsoft ve spolupráci s Michiganskou univerzitou.



Obr. 2: Ukázka infekce cíleného operačního systému VMBR (šedé části).

3.3.2 Režim jádra

Jde o rootkity, které pracují v režimu jádra. Přidávají další kód a/nebo nahradí část kódu jádra, aby pomohly ke skrytí backdooru v počítačovém systému. Toho je obyčejně dosaženo přidáním nového kódu do jádra systému přes ovladače k různým zařízením (v OS Windows) nebo modulů jádra (v Linuxu). Infikování na této úrovni není lehké dosáhnout, ale jakmile se jednou podaří, rootkit může vykonávat prakticky jakoukoliv aktivitu, aniž by byl detekován. Nejedná se pouze o kompromitování zabezpečení systému počítače, ale může to mít i drastický dopad na stabilitu systému a jeho budoucí rozšiřitelnost.

Bez pomoci specializovaného softwaru je tyto rootkity velmi těžké odhalit [18], [25].

3.3.3 Rootkity v DLL knihovnách

Používají metody jako je „patchování“[1], „hookování“[2] a nahrazování systémových volání upravenou verzí, která neprozradí skrytou činnost útočníka. Tyto funkce mohou ovlivnit chování legitimních programů, tím že je nechají vykonat další přídavné funkce, které nejsou

[1] Patch je malá část programu, která slouží k vylepšení programu samotného nebo k nahrazení jeho poškozené části.

[2] Pokud budeme uvažovat sled událostí vykonávané nějakým programem. Hookování se dá chápat tak, že se, v našem případě rootkit, „zahákne“ někde mezi tento sled příkazů a sám se stane jeho součástí. Po zavolání tohoto programu se v určitou chvíli vykonají tedy i příkazy obsažené v rootkitu.

v jiných případech povoleny. Například otevření nového spojení a vysílání důvěrných dat za použití oprávnění onoho legitimního programu.

3.3.4 Aplikační rootkity

Rootkity na této úrovni mohou nahrazovat části kódu aplikací trojským koněm nebo jiným malwarem. Mohou také nahradit činnost programu činností jiného škodlivého softwaru (metodě se říká „hijackig“).

Metody, které slouží pro úpravu chodu aplikace, mohou být - patchování, hookování, vkládání kódu atd.

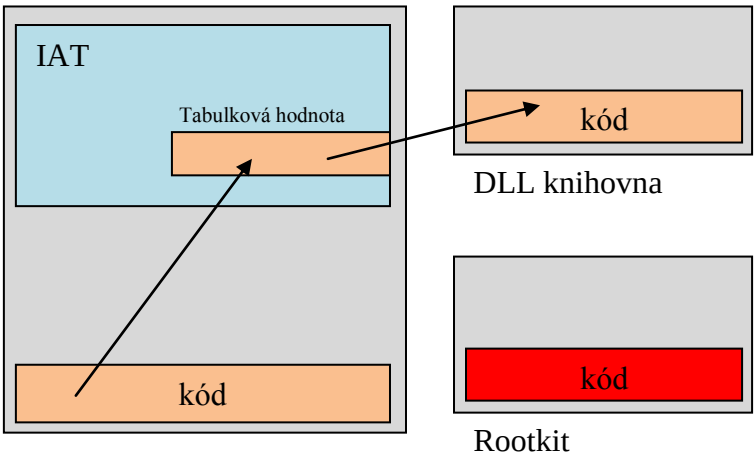
3.4 Druhy hookování

3.4.1 IAT hooking

IAT (Import Address Table) je jedna z částí spustitelného souboru. Je to tabulka ukazatelů na funkce nacházející se v různých DLL (Dynamic Link Library) knihovnách. Mohou teoreticky nastat dvě situace, program má vlastní funkce a další funkce DLL knihoven nepotřebuje (téměř nenastává), nebo se jedná o komplexnější program a bude tedy potřebovat IAT.

IAT se využívá při spouštění programu, ten totiž dopředu neví, kde se nalézají DLL knihovny obsahující potřebné funkce. Umístění DLL totiž není statické, jakmile se vytvoří nová verze knihovny, může se stávající poloha změnit. Během nahrávání spustitelného souboru prochází loader[3] tuto oblast a hledá potřebné funkce v různých knihovnách. Poté ukládá tyto adresy funkcí zpět do IAT daného programu (viz obr. 3).

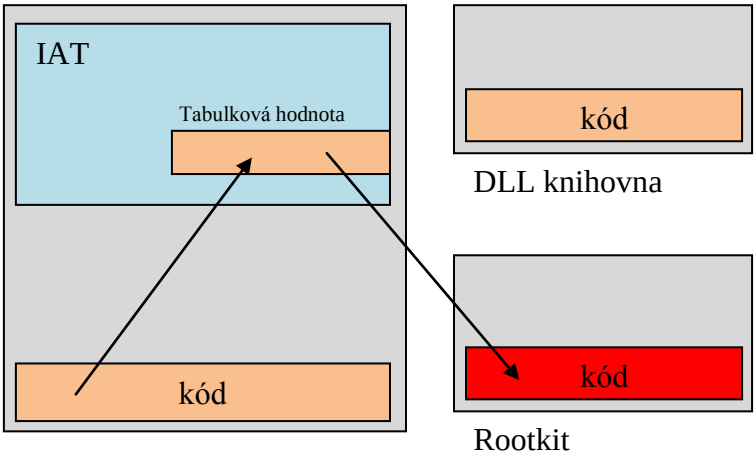
[3] Aplikace systému Windows, která prochází jednotlivé DLL knihovny a „nahrává“ ukazatele na potřebnou funkci do IAT daného programu.



Obr. 3: Správná funkce IAT [3].

Úpravou adres v IAT může útočník přesměrovat původní volání na volání kódu rootkitu. Rootkit může potom sám zavolat tuto funkci a její výsledky upravit podle sebe (viz obr. 4).

Rootkit nemusí napadat pouze IAT, může pracovat v režimu jádra. Jako takový může přímo nahrazovat adresy systémových funkcí za škodlivý kód. Tato technika je z hlediska napadeného velmi nebezpečná a těžce odhalitelná. Rootkit nenapadá jednotlivé aplikace, ale přímo jádro, což má za následek ovlivnění funkčnosti celého sytému [3], [9], [11].



Obr. 4: IAT infikované rootkitem [3].

Inline function hooking

Tento druh hookování je dokonalejší než předchozí, nedochází zde totiž k pouhému nahrazení ukazatele, ale dochází přímo k modifikaci dané funkce. Rootkit si odebere několik prvních

bytů dané funkce a sám se na toto místo nahraje[4]. Aby nedošlo k narušení běhu dané funkce a aby se ztížila možnost detekce, rootkit těch několik prvních bytů funkce zachová – funkce se tedy tváří, že pracuje, tak jak má.

S touto technikou se můžeme setkat i u Windows firmy Microsoft[5]. Díky této technice se může provést aktualizace operačního systému, aniž bychom ho museli restartovat. To je výhodné pro uživatele, protože nemusí ztrácet čas restartováním systému. Firma Microsoft uvedla, že se díky tomuto způsobu sníží celkový počet restartů přibližně o 53% [15].

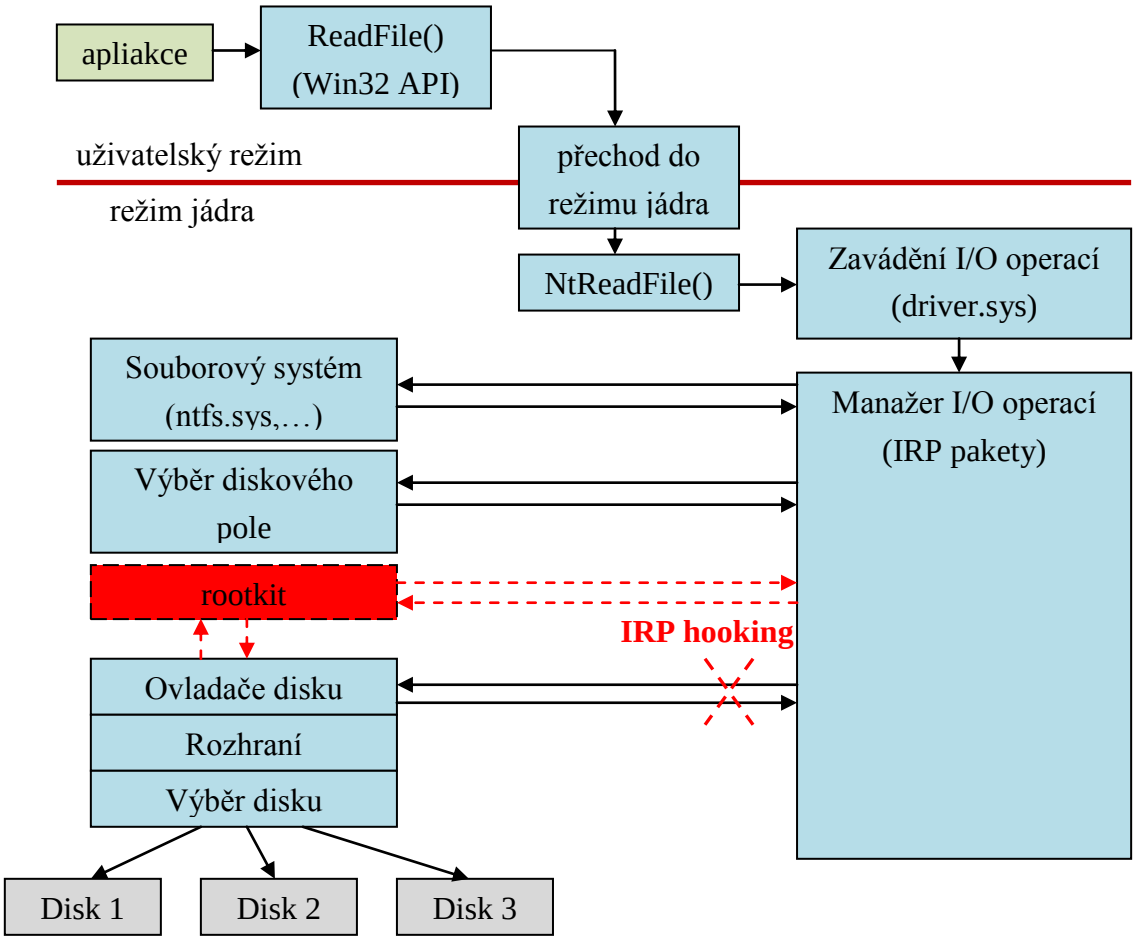
3.4.2 IRP hooking

Každý ovladač zařízení se dá chápat jako tabulka odkazů na funkce, pomocí kterých se obsluhují různá I/O zařízení. Tyto funkce jsou získány buď od systému, nebo od klientské aplikace. Komunikace s ovladačem je ve většině případů realizována pomocí IRP (I/O Request Packet).

V případě infekce je některá hodnota v tabulce nahrazena kódem rootkitu. Místo požadované operace se vykoná kód samotného rootkitu (obr. 5 – plnými čarami je naznačen normální průběh, čerchovanými čarami po infekci rootkitem) [13]. Dokonalejší variantou je, podobně jako u inline hookingu, že si rootkit nejprve zazálohuje původní hodnotu z tabulky, vykoná původní žádost, ale následně pozmění výsledky [1], [3].

[4] Přesněji řečeno, je zde nahrán odkaz na rootkit ve formě nepodmíněného skoku.

[5] Microsoft nazývá tuto techniku „hot patching“.



Obr. 5: Příklad čtení souboru z disku počítače infikovaného rootkitem - metoda IRP hook.

3.5 Pokusy o legální rootkit

3.5.1 Sony rootkit

Jedná se o příklad využití rootkitů v komerční sféře. Firma Sony ke konci roku 2005 uvolnila spolu s prodávanými CD nosiči i program na přehrávání hudby od MediaMax. Tento program na přehrávání se spustil automaticky po vložení CD do mechaniky. Nikdo však tehdy netušil, že se jim kvůli spuštění přehrávače do systému dostane i rootkit. Tento rootkit umožnil vypálení jen určitého počtu kopií tohoto média a umožňoval přehrávání hudby pouze pomocí dodávaného přehrávače.

Tento DRM (Digital Rights Management) rootkit odhalil až o rok později odborník na Windows Mark Russinovich (viz lit. [17], [19]) při testování svého programu na odhalování

rootkitů „RootkitRevealeru“[6]. Po spuštění programu na svém počítači objevil řadu skrytých souborů a klíčů v registru, všechny podezřelé řádky začínaly na řetězec sys. Zjistil, že ovladač „Aeries.sys“ ukrývá činnost těchto spuštěných procesů. Dalším pátráním se dostal k tomu, že za to může DRM od firmy First 4 Internet s názvem XCP (nyní už The Amergence Group).

Aby přišel na to, co to přesně dělá, nechal si toto CD přehrát v mechanice. Po spuštění hudebního přehrávače z CD si všiml, že program sysDRMServer.exe, který byl skrytý, začal zatěžovat výkon procesoru. Po zavření přehrávače tento program však pokračoval v činnosti, ale již ne s takovou zátěží procesoru. Dalším zkoumáním zjistil, že tento program každé dvě sekundy zjišťuje informace o dvou systémových souborech na disku. Rozhodl se tyto soubory odstranit jak fyzicky z disku tak i z registrů. Po restartu počítače ale zjistil, že mu zmizely všechny optické mechaniky. Dalším šetření se dopátral k tomu, že za to mohl jeden jím smazaný soubor syscreater, který má za úkol inicializaci ovladačů optických mechanik při startu systému.

Firma Sony se této kauze bránila dlouho, neustále se hájila tím, že tento rootkit není pro uživatele nijak nebezpečný. Svědčí o tom i výrok firmy Sony [7]:

> “Většina lidí vůbec netuší, co to rootkit je, takže proč by se o to měli zajímat?[7]”

Firma Sony kauzu prohrála a jako odškodné vyplatila každému poškozenému uživateli něco kolem 100$.

3.5.2 USB flash disky od Sony

Počátkem září roku 2007 objevili bezpečnostní experti z firmy F-Secure na internetu další rootkit - aplikace používaná firmou Sony [22]. Tentokráte se jedná o USB flash disky MicroVault řady USM-F, které disponují čtečkou otisků prstů. Rootkit se nachází v softwaru na přiloženém CD.

Rootkit si vytvoří skrytou složku pod \WINDOWS, která je tak skrytá, že se do ní normální cestou přes průzkumníka Windows nedostaneme. Pokud známe jméno složky, tak již není problém se do tohoto umístění dostat. Dále již rootkit „neškodí“, problém je ale v tom, že dané místo je „neviditelné“ pro antivirové programy. Problém je tedy v tom, že se na dané umístění může nahrát škodlivý software, který může skrytě a nerušeně provádět svou činnost. Sony tuto mezeru v bezpečnosti doposud neopravila.

---

[6] Tento program se snaží odhalit soubory na disku a záznamy v registru, které nejsou vidět v operačním systému – jsou před Windows API schovány.

[7] Výrok Thomase Hesse, prezidenta divize pro globální digitální obchod, Sony BMG.

Firma F-Secure oznámila, že stávající rootkit není tak nebezpečný jako ten předchozí (viz kapitola 3.5.1). Liší se hlavně v tom, že software pro USB zařízení si uživatel instaluje vědomě, zatímco přehrávač u hudebních CD byl instalován automaticky. Přestože je méně nebezpečný, stále může být zneužit potencionálními útočníky.

4. Detekce

V této části jsou prezentovány běžně používané metody detekce spywaru a rootkitů. Čerpáno je zejména z pramenů [14], [25], [26].

4.1 Techniky detekce spywaru

4.1.1 Shoda názvů souborů

Nejjednodušší metodou detekce je pomocí shody názvů. Jak název metody napovídá, detekční program skenuje disk a hledá názvy souboru shodujících se s doposud známými názvy spywaru. Soubory pozitivní na výskyt spywaru jsou potom označeny k vymazání.

Tato forma detekce funguje, ale má velké nevýhody. Ve snaze zmařit úsilí detekujícího softwaru, spywarové společnosti změní název souboru nebo používají strategii náhodného pojmenování souborů. Jakmile se jednou název souboru změní, program na detekci již není schopen takto upravený spyware rozpoznat.

Další problém je, že programy na detekci nejsou schopny rozlišit rozdíl mezi spywarem a regulérním programem. Například, předpokládejme, že známá spywarová aplikace má soubor pojmenovaný example.dll, který je s ní spojen. Jakmile by regulérní program používal také soubor pojmenovaný example.dll, program na detekci by ho označil k vymazání, bez ohledu s jakou aplikací je spojen.

4.1.2 Shoda vlastností souboru

Další metodou detekce spywaru je porovnávání vlastností souboru s vlastnostmi známého spywaru v databázi. Tato metoda se obvykle používá spolu s předchozím typem detekce a vytváří tak o trochu spolehlivější řešení detekce spywaru.

Jakmile program pro detekci označí soubor pomocí metody shody názvů, zkontroluje potom ještě vlastnosti souboru, jako jsou velikost, vydavatel a verzi souboru. Pokud se alespoň jedna nebo více vlastností shoduje s vlastnostmi uvedených v databázi, je soubor označen pro odstranění.

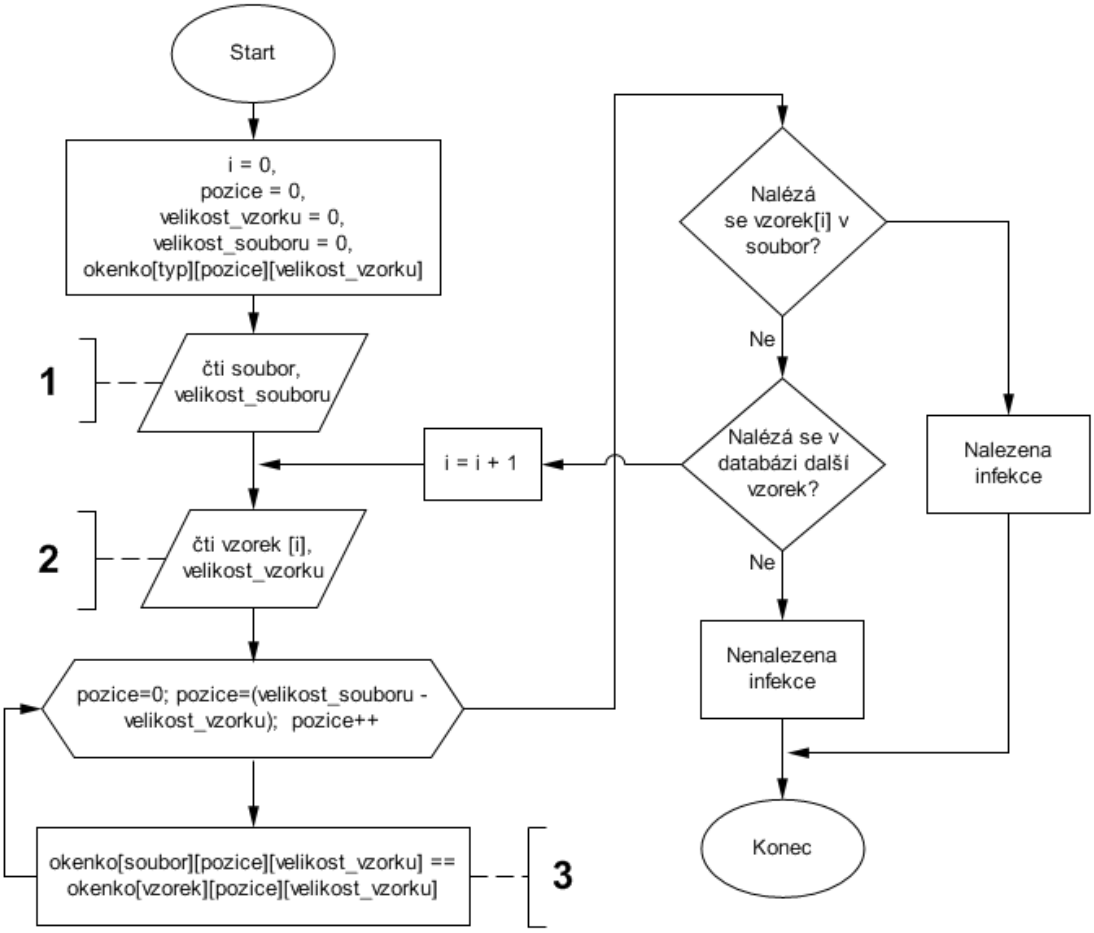
Kombinace shody názvů a porovnání vlastností dělá detekční software mnohem výkonnějším. Autoři spywaru jsou schopni tuto formu detekce snadno obcházet a to přejmenováváním názvu souboru, změnou názvu vydavatele, jednoduchou úpravou velikosti

souboru nebo nahráním nové verze. Naštěstí se pomocí této metody detekce spywaru zmenší pravděpodobnost chybného označení regulérního souboru.

4.1.3 Shoda podle vzorku kódu v souboru

Metody jako jsou Shoda názvů a Shoda vlastností souboru se, zjednodušeně řečeno, dívají pouze na „obal“ kolem samotného kódu souboru. Jedinou možností, jak zjistit s čím se potýkáme, je podívat se „dovnitř“ toho obalu. Při hledání spywaru program pro detekci prohledává „vnitřní“ část souboru a hledá tam jisté znaky/vzorky shodné s těmi, které má uložené v databázi. Jakmile se najde znak, který se shoduje s tím v databázi, soubor je označen pro odstranění.

Podrobnější pohled na použitý algoritmus poskytuje vývojový diagram na obr. 6.



Obr. 6: Algoritmus metody detekce pomocí vzorku kódu.

V následujícím textu je jednoduše popsán postup algoritmu detekce pomocí vzorku kódu z obr. 6. Nejdříve je potřeba načíst do paměti podezřelý soubor (odpovídá poznámce 1

na obr. 6), načtený soubor by mohl vypadat asi jako na obr. 7. Poté se zjistí velikost daného souboru a uloží se do příslušné proměnné.

... A1 43 5C FB 10 4E AD 25 40 7A 22 A1 3C 9C 82 FE 8B 20 ...

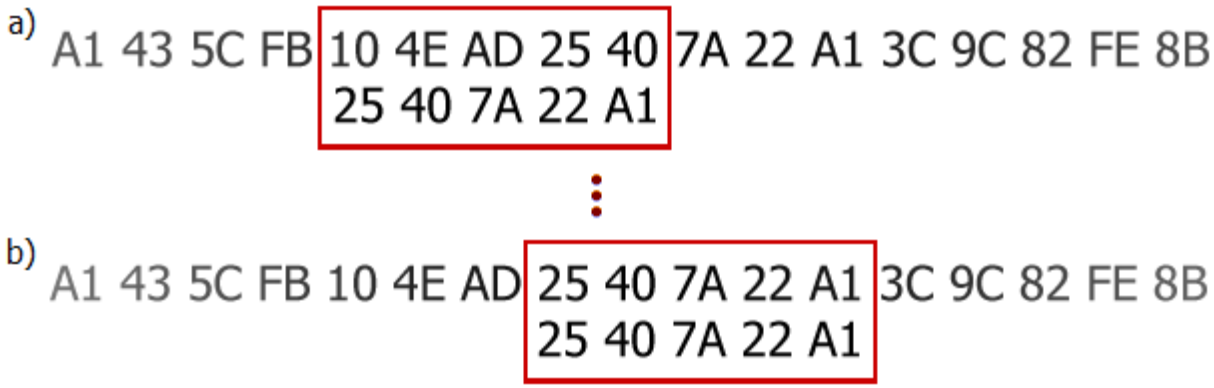
Obr. 7: Ukázka načteného souboru.

Analogicky k postupu načtení souboru se provede načtení (viz obr. 8)[8] a zjištění délky vzorku spywaru (odpovídá poznámce 2 z obr. 6).

25 40 7A 22 A1

Obr. 8: Ukázka načteného vzorku spywaru.

Jakmile je načtený soubor i vzorek spywaru, začne se porovnávat (tomuto odpovídá část č. 3 na obr. 6). Nejprve se vytvoří tzv. „okénko“, které má stejnou velikost jako vzorek daného spywaru. Ze souboru se tedy vezme tolik bytů, jaká je velikost okénka. Takto získaná část kódu se porovná se vzorkem spywaru, jestliže se kódy neshodují – nebyla nalezena infekce, okénko se posune o jednu pozici doprava a opět proběhne testování, jakmile je nalezena shoda těchto dvou kódu – byla nalezena infekce (viz například obr. 9).



Obr. 9: Demonstrace použití „okénka“, a) nic nenalezeno, b) nalezena infekce.

Jakmile se projde celý soubor, aplikace se podívá do databáze spywaru, zda se zde nenachází ještě další vzorky, jestliže ano, celý proces testování se opakuje, jinak se aplikace ukončí.

Přestože autoři spywaru mohou lehce změnit název souboru, případně pozměnit jeho vlastnosti, je mnohem složitější upravit samotný program. Shoda podle vzorku kódu je tedy spolehlivá metoda a hodně známých programů pro detekci spywaru ji používá.

[8] pro větší přehlednost je vzorek spywaru zkrácen

4.1.4 Heuristická analýza

Heuristická analýza pracuje na podobném principu jako při vyhledávání určitého vzorku v kódu programu. Rozdíl je v tom, že hledá instrukce nebo příkazy, které za normálních okolností nejsou součástí programu, jako například příkaz na mazání věcí na disku. Tato metoda se převážně používá na detekci malwaru a jiných aplikací podobného typu.

Heuristická analýza je dobrou metodou pro odhalování spywaru, ale jednoduchý heuristický systém odhaluje pouze škodlivé kódy nikoliv cookies nebo adware. Modernější programy pro detekci sypwaru kombinují heuristickou analýzu spolu s dalšími metodami uvedenými dříve.

4.1.5 Prohledávání registrů

Stejně jako většina běžných a neškodných aplikací i spyware upravuje během své instalace systémové registry. Časem mohou tyto hodnoty způsobit zmatek v registrech a zpomalit počítač. Registry se také mohou stát nepoužitelnými. Prakticky všechny detekční programy hledají v registrech jakoukoliv stopu po spywaru. Dosahují toho pomocí porovnávání vzorků z registrů se vzorky uloženými v aplikační databázi detekčního programu.

4.2 Detekce rootkitů

Rootkity se za posledních pár let vyvinuly a stávají se mnohem komplikovanějšími. O to větší nebezpečí hrozí, jakmile se spojí se spywarem. Čím jsou tedy rootkity složitější, tím se samozřejmě komplikuje jejich detekce - odstranění. Navíc, pokud je počítač jednou napaden, nikdy už mu nelze plně důvěřovat. Existuje několik různě úspěšných technik, které lze k odhalení rootkitů použít.

Většina metod je podobná již popsaným metodám detekce spywaru (viz kapitola 4.1).

4.2.1 Shoda s referenční kopií systému

Jedna z teoreticky nejúčinnějších metod detekce rootkitů je využití neinfikované kopie systému. Podezřelý počítač se po zapnutí nechá nabootovat z neifikované kopie systému. Na této verzi systému se spustí aplikace, která bude procházet systém a porovnávat jednotlivé soubory. Podezřelá verze systému bude brána jako data a rootkit bude zcela neúčinný, nebude tedy moct skrýt svou přítomnost. Tato metoda potřebuje neinfikovanou referenční kopii systému, což by mohl být problém. Systémy totiž nejsou neměnné, ale dochází v nich k neustálým změnám. Proto realizace této metody by byla velmi komplikovaná.

4.2.2 Prohledávání disku z bootovacích médií

Přestože je předchozí metoda pro své nároky obtížně realizovatelná, myšlenka prohledávání neaktivního systému má něco do sebe. Mít neustále aktuální neinfikovanou referenční kopii systému je náročné. Objevila se tedy myšlenka mít médium (CD, USB flash disk, …), které obsahuje pouze nejnutnější věci potřebné pro nabootování. Navíc by mělo ještě obsahovat nějakou aplikaci, pomocí které se bude prohledávat systém.

Počítač se nechá nabootovat například z CD nebo USB flash disku, poté se spustí aplikace, která bude prohledávat podezřelý systém. Rootkit ukrytý v počítači bude neaktivní a nebude tedy moct ukrýt svou přítomnost nebo se jinak zamaskovat. Schopnost odhalovat rootkity závisí pouze na spuštěné aplikaci na bootovacím médiu.

4.2.3 Cross-view detection

Tato metoda využívá faktu, že neexistuje pouze jediná cesta jak získat požadované informace. Při použití této metody jsou skryté soubory identifikovány díky porovnání výsledků, které jsou získány pomocí různých přístupů k datům. Metoda porovnává výstup API funkcí (vysoká úroveň) s výstupem získaných pomocí metod používajících velmi nízkou úroveň. Například pokud budeme chtít najít ukryté soubory, metoda nejdříve použije funkce Windows API, aby získala informace o daném souboru ze systému. Následně použije nízko-úrovňové metody k provedení analýzy souborového systému. Metoda potom porovnává získané informace s informacemi získaných pomocí API, díky tomu je schopná odkrýt soubory, které jsou běžně pro úroveň API skryté [5].

Tato metoda má však nevýhodu, není schopná detekovat rootkity, které pracují na stejně nízké nebo dokonce nižší úrovni než použitá metoda. I přes tuto nevýhodu patří tato metoda k těm úspěšnějším, nepoužívá totiž vyhledávání na základě specifických znaků nebo podle informací z databáze.

4.2.4 Další detekční metody

Další metody, které se běžně používají, pracují uvnitř napadeného systému.

Jedním z možných postupů detekce je vyhledávání stop v systému, které po sobě rootkity zanechávají. Z jistého hlediska probíhá tato detekce obdobně jako u spywaru nebo u virů. Odhalování probíhá na základě databáze vzorků, podle které jsou identifikovány infikované soubory. Na rozdíl od spywaru nebo virů nemůže být tato technika příliš účinná. Jednak jsou detekční programy schopny odhalit pouze známé rootkity, jednak je obtížné prohledávat soubory, které bývají pomocí rootkitů skryty.

Předchozí metodu lze skombinovat s některou vyspělejší metodou jako je například heuristická analýza. Opět je zde jistá podobnost s detekcí spywaru. Programy pro detekci, které pracují na principu heuristické analýzy spolu s porovnáváním vzorků, mohou odhalit doposud neznámé rootkity. Nástroje, které tato metoda poskytuje, jsou schopny odhalit i různé nesrovnalosti v systému případně odchylky od běžného chování.

4.2.5 Kernel patch protection

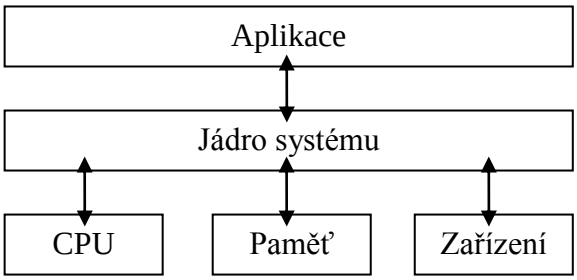
Z předchozích kapitol lze pochopit, že největší nebezpečí číhá od rootkitů, které napadají přímo jádro systému, manipulují s ním nebo v něm provádí různé změny. Tyto zásahy do jádra však mívají za následek problémy s jeho stabilitou, spolehlivostí a výkonem. S přechodem na 64-bitovou architekturu je možné zvýšit zabezpečení jádra.

Nové 64-bitová Windows Vista podporují technologii KPP (Kernel Patch Protection), nazývanou též PatchGuard. Tato technologie zabraňuje neoprávněným zásahům do jádra, jeho modifikace atd. KPP neustále hlídá, zda nedochází k neoprávněné (neautorizované) úpravě jádra systému. Pokud by k tomu došlo a nedokázala by tomu zabránit, bylo by zahájeno okamžité vypnutí tohoto systému. Výrazně tak dochází ke zvýšení bezpečnosti a stability jádra operačního systému.

Občas je ale nutné nějakou tu modifikaci jádra provést, může jít o ovladače zařízení, různé zabezpečovací produkty atd. Proto zavedla firma Microsoft systém, který bude využívat podepsaných ovladačů. Tyto ovladače jsou testovány a následně schváleny přímo Microsoftem a 64-bitové Visty, respektive 64-bitové XP Professional, nedovolí instalaci nepodepsaných ovladačů. Ideální by bylo tyto změny aplikovat i na stávající 32-bitové operační systémy, ale to by mělo za následek nemalé problémy s kompatibilitou již vydaného hardwaru. Proto se zavedla tato technika až pro 64-bitové operační systémy, na ty ještě neexistuje takové množství softwaru.

Přestože tato technologie nezaručí absolutní bezpečnost systému, je to jistě krok kupředu. Bude to mít za následek i zvýšení stability systému, protože většina havárií systému je způsobena použitím nekvalitních ovladačů.

Na druhou stranu objevují se i názory, které říkají, že tuto metodu lze překonat. Objevují se stížnosti některých antivirových firem, které říkají, že jejich antivirové programy nebudou schopny stoprocentně pracovat za použití této obrany jádra [15].



Obr. 10: Jádro jako prostředník mezi softwarem a hardwarem.

4.3 Detekce na síťovém rozhraní

Spoléhat se pouze na detekční programy a firewall počítače může být pro někoho postačující, ale v dnešním světě sítí je to již nedostatečné. Je tedy vhodné mezi již používané vrstvy ochrany vložit ještě jednu vrstvu, která bude detekovat pokusy o průnik přes síťové rozhraní.

Karta síťového rozhraní může teoreticky pracovat ve dvou módech, kdy zachytává:

- pakety, které jsou určeny právě pro daný počítač (určeno IP adresou, MAC adresou)
- všechny pakety, které „vidí“ na přenosovém médiu

Detekce pracuje na principu kontrolování provozu na síťovém rozhraní. Je tedy vhodné, aby síťová karta pracovala ve druhém z uvedených režimů [8].

Jedna z možností kontroly provozu spočívá v porovnávání paketů s již známými vzorky podezřelých kódů, které jsou uloženy v lokální databázi. Toto chování je tedy velmi podobné principu popsanému v kapitole 4.1.3. Porovnávají se byty v celém paketu nebo pouze v jejich hlavičkách (rychlejší způsob). Problémem ovšem může být velikost databáze, případně nemožnost detekování nových typů útoků [2].

Další možností je kontrolovat příchozí pakety v souvislosti s odchozím provozem počítače – zda je příchozí paket reakcí na požadavek zaslaný počítačem apod. Kontrolovat se dá i na základě „povahy“ případně množství příchozích paketů. Velké množství příchozích paketů, které jsou nasměrovaných na různé porty (tzv. skenování portů), může naznačovat, že se někdo snaží zjistit slabiny daného počítače. Případně velké množství příchozích ICMP[9] paketů, aniž bychom například použili příkaz *ping*, naznačuje, že se někdo pokouší o zahlcení naší sítě (tzv. DoS[10] útok) apod.

[9] Internet Control Message Protokol – protokol používaný příkazem *ping*

[10] Denial of Service = odmítnutí služby – síťové rozhraní je zahlceno tak, že není schopno propouštět regulérní provoz

Výše popsané techniky by se daly popsat jako pasivní – tzv. IDS[11], obecně vzato mají tyto techniky za úkol pouze upozornit na přicházející nebo právě probíhající útok. Jestliže by měl systém podniknout i kroky k zastavení přicházející hrozby (například zablokování zranitelného portu, případně celého síťového rozhraní atd.), jednalo by se již o aktivní řešení – tzv. IPS[12].

Za základní nástroje, které by se daly používat k „ruční“ kontrole provozu, by se mohly považovat například analyzátory síťového provozu[13].

4.4 Prevence

Úkol první linie obrany je vždy v zabránění vstupu jakéhokoliv rootkitu, spywaru a jiného škodlivého softwaru do počítače. Je proto dobré dodržovat, alespoň pár základních pravidel:

- používat personální firewall, který bude chránit proti neautorizovaným vstupům
- využívat antivirové a antispywarové řešení pro počítač
- zapojení nějakého druhu síťového „štítu“
- pravidelně kontrolovat aktualizace pro veškeré nainstalované softwarové produkty (obzvláště pro antivirové a antispywarové programy)
- stahovat nejnovější záplaty systému
- používat silná hesla

Přestože jsou bezpečnostní programy neustále vylepšovány, stejně tak se zlepšují i programy, proti kterým bojují. Využíváním těchto základních pravidel včetně zdravého rozumu, je krok správným směrem z hlediska zabezpečení systému.

[11] Intrusion Detection System = systémy detekce průniku

[12] Intrusion Prevention System = systém prevence průniku

[13] volně šiřitelný je např. program Wireshark (dříve Ethereal), dostupný z www.wireshark.org

5. Realizace – program Tester

Správně by se mělo psát, že se jedná o program „Tester 2.0“[14]. Ten vznikl na základě původního programu „Tester“, který testoval pouze jeden vybraný soubor.

Oproti první verzi programu nabízí druhá tyto výhody:

- zrychlení prohledávání souboru
- ve výpisu se spolu s cestou k infikovanému souboru objeví i informace o nákaze
- rozšířené ovládání během testování (pozastavení apod.)
- možnost testování celého disku (datové jednotky)

5.1 Vývojový diagram

Program Tester slouží k prohledání zvolené jednotky (ať už harddisku, CD nebo flash disku) na výskyt infekce. V tomto případě se jedná převážně o spyware, ale jakmile bychom doplnili jeho databázi i o vzorky virů, tak by mohl sloužit i jako primitivní antivirový program[15]. Program pracuje na principu vyhledávání určitého vzorku kódu uvnitř testovaných souborů (podrobněji viz kapitola 4.1.3). Vzorky kódů se porovnají a v případě shody jsou podniknuty příslušné kroky. Vzorek kódu může představovat specifickou instrukci či příkaz, který je charakteristický pro danou infekci.

Tato metoda není tak sofistikovaná jako například heuristická analýza, ale na druhou stranu není ani tak jednoduchá jako je například vyhledávání pomocí shody názvů. Autoři spywaru mohou totiž lehce změnit název souboru popřípadě jeho vlastnosti, ale je pro ně mnohem náročnější pozměnit samotné instrukce programu. Proto metoda, vyhledávání podle vzorku kódu, patří mezi ty spolehlivější a hodně známých detekčních programů ji využívá.

Vývojový diagram programu Tester viz příloha 1. Je v něm možné poznat, jak program pracuje. Některé realizace algoritmů jsou popsány v následující kapitole.

[14] dále v textu se bude názvem „Tester“ myslet program „Tester 2.0“

[15] primitivní proto, že reálné antivirové programy kombinují mnohem více vyhledávacích funkcí a navíc poskytují i různé způsoby rezidentní ochrany

5.2 Důležité algoritmy programu

Program Tester byl vytvořen v programovacím prostředí Microsoft Visual Studio 2005 Professional Edition v jazyce C#. Je navržen tak, aby jeho ovládání bylo co nejjednodušší a pokud možno co nejvíce intuitivní, ale hlavně aby splňovalo svůj účel. Pro informace o programování v jazyce C# viz [4], [6], [16].

Kvůli větší přehlednosti v programovém kódu jsou v něm vytvořeny navíc dvě třídy (*Soubory* a *Hledani*). Pro lepší orientaci v následujícím textu, je níže uvedena struktura programu (třídy a jejich nejdůležitější metody):

- Soubory
  - hledej
  - DejSoubor
- Hledani
  - SpustTest
  - nactiXML

5.2.1 Třída *Soubory*

Tato třída slouží, jak už její název napovídá, k práci s adresáři a soubory. Přesněji by se dalo říct, že obstarává prohledávání disku. Obsahuje metody, díky kterým se získávají postupně cesty ke všem souborům na vybrané jednotce. Jiné metody slouží k předávání potřebných informací třídě *Hledani* (např. cesty k souborům, informace o stavu zásobníků apod.).

Metoda „hledej“

Asi nejdůležitější ve třídě *Soubory* je metoda *hledej* (nejpodstatnější část algoritmu je uvedena na obr. 11). Tato metoda má za úkol prohledávat celý disk a nalezené soubory uložit do příslušné paměti[16].

Metoda přebírá parametr `cesta`, což je umístění, které se má prohledávat na výskyt souborů, například při startu programu se jedná o písmeno námi vybrané jednotky. Metoda nám neposkytuje žádné návratové hodnoty (ani není potřeba), toho dosáhneme klíčovým slovem `void` v definici této metody.

[16] pro upřesnění: nejedná se o soubory jako takové, ale pouze o plnou (absolutní) cestu k těmto souborům

```
private void hledej(string cesta)
{
      < zdrojový kód zkrácen >

      DirectoryInfo dir = new DirectoryInfo(cesta);
      foreach (FileInfo f in dir.GetFiles("*.*"))
      {
            soubory.Enqueue(f.FullName);
      }
      foreach (DirectoryInfo g in dir.GetDirectories())
      {
            adresare.Push(g.FullName);
      }
}
```

Obr. 11: Metoda „hledej“ třídy *Soubory*.

Pomocí příkazu `dir.GetFiles("*.*")` v prvním cyklu `foreach` prohledáváme umístění. Veškeré nalezené soubory (respektive jejich absolutní cesty) uložíme příkazem `soubory.Enqueue()` do paměti k tomu určené – do paměti `soubory`. Ve druhém cyklu `foreach` necháme totéž proběhnout ještě jednou, tentokráte ale pro adresáře, a veškeré nálezy se uloží pomocí `adresare.Push()` do paměti `adresare`.

Metoda „DejSoubor“

Hlavním úkolem této metody je obstarávání pravidelného přísunu souborů třídě *Hledani*.

Zdrojový kód je k nahlédnutí na obr. 12. Metoda nejdříve otestuje, zda je v paměti `soubory` alespoň jeden záznam. Pokud ano, tak tento záznam předá třídě *Hledani* a následně si jeho záznam smaže. Jestliže je ale paměť `soubory` prázdná, tak otestuje, zda se v paměti `adresare` nachází nějaký záznam. Jakmile se zde nachází alespoň jeden záznam, metoda zavolá metodu `hledej` s parametrem umístění adresáře a následně proběhne prohledání adresáře (viz popis metody `hledej`), adresář se poté vymaže z paměti.

Jestliže není v paměti `soubory` ani v paměti `adresare` nějaký záznam, metoda vrátí prázdný řetězec, což je znamení k ukončení celého testování.

```
public string DejSoubor()
{
     if (soubory.Count == 0)
     {
          if (this.adresare.Count == 0)
               return String.Empty;
          hledej(adresare.Pop());
          return this.DejSoubor();
     }
     else
     {
          return soubory.Dequeue();
     }
}
```

Obr. 12: Metoda „DejSoubor“ ze třídy *Soubory*.

5.2.2 Třída *Hledani*

Zjednodušeně by se dalo říci, že tato třída má na starosti vyhledávání vzorků spywaru uvnitř souboru. Pracuje tak, že od třídy *Soubory*, pomocí metody `DejSoubor`, přebírá cesty k souborům, které pak následně otestuje.

Metoda „SpustTest“

Dalo by se říci, že tato metoda zastřešuje celý program. Zavoláním metody `SpustTest` se vlastně spustí samotný proces testování – prohledává se vybraný soubor na výskyt vzorků z databáze.

Zpočátku si nadefinujeme proměnou `soub` ze třídy `Soubory`, díky této proměnné můžeme přistupovat k metodám, které nám tato třída poskytuje (část zdrojového kódu na obr. 13). Jelikož má tato metoda obsahuje volání na všechny důležité funkce/metody je do ní začleněno i načítání databáze spywaru do paměti (podrobněji viz metoda `nactiXML`). Příkazem `soub.Disk = this.jednotka` získáme počáteční lokaci[17] určenou k prohledání. Dále následuje definování proměných a jejich počátečního stavu:

- `stav` – určuje nám stav testu (true = nalezena infekce, false = nenalezena)
- `info` – informace o dané infekci
- `nazev` – zde se ukládá cesta k testovanému souboru

[17] písmeno jednotky zadané uživatelem

Cyklus `do-while` bude probíhat tak dlouho, dokud bude v paměti `soubory` a v paměti `adresare` alespoň nějaký záznam (tuto podmínku obstarává metoda `prazdno` ze třídy `Soubory`). Jeden běh tohoto cyklu obstará prohledání právě jednoho souboru na výskyt vzorků z databáze. Uvnitř cyklu proběhne zpočátku přiřazení hodnot proměným, do proměnné:

- `nazev` a `aktualniSoubor` – cesta k právě testovanému souboru
- `stav` – výsledek vyhledávání vzorku uvnitř souboru (buď true nebo false)
- `pocetOtestovanych` – hodnota se inkrementuje o jedna (indikuje počet otestovaných souborů)

Pokud je nalezena infekce (`stav == true`), provede se záznam do příslušných promněných a pamětí. Do paměti `infikovane` se uloží cesta k infikovanému souboru. Pomocí proměnné `infikovaneSoubory` se v programu vypisují informace o infekci (umístění souboru + pokud jsou v databázi informace o nákaze, tak i tyto se zobrazí).

```
public void SpustTest()
{
     Soubory soub = new Soubory();
     soub.Disk = this.jednotka;
     this.nactiXML();
     bool stav = false;
     string info = String.Empty;
     string nazev = String.Empty;
     do
     {
          nazev = soub.DejSoubor();
          stav = this.Skenuj(nazev);
          aktualniSoubor = nazev;
          pocetOtestovanych++;
          if (stav == true)
          {
               infikovane.Add(nazev);
               info = vzorkyInfo[cisloVzorku];
               info = vzorkyInfo[cisloVzorku];
               infikovaneSoubory = nazev + " \t " + info +
          "\r\n" + infikovaneSoubory;
     }
     while (!soub.prazdno());
     < zdrojový kód upraven >
}
```

Obr. 13: Metoda „SpustTest“ třídy *Hledani.*

Metoda „nactiXML“

Pomocí této metody je program schopen načítat soubor, který obsahuje databázi spywaru. Soubor s databází je ve formátu XML. Podrobnější informace viz kapitola 5.4.

Z obr. 14 je patrné, že nejprve si nadeklarujeme proměnou typu `XmlTextReader`, která má za parametr umístění souboru (v našem případě databázi se vzorky). V cyklu `while` probíhá samotné čtení souboru. Pomocí první podmínky uložíme informace o vzorcích spywaru do paměti `vzorkyInfo`. Ve druhé podmínce již probíhá samotné ukládání vzorků do paměti `vzorky`.

Jak napovídá deklarace, tato metoda nám po vykonání kódu navrací hodnotu *true* nebo *false.* Metoda nám navrátí *true*, jestliže byl do paměti `vzorky` uložen alespoň jeden záznam, ve druhém případě je návratová hodnota *false.*

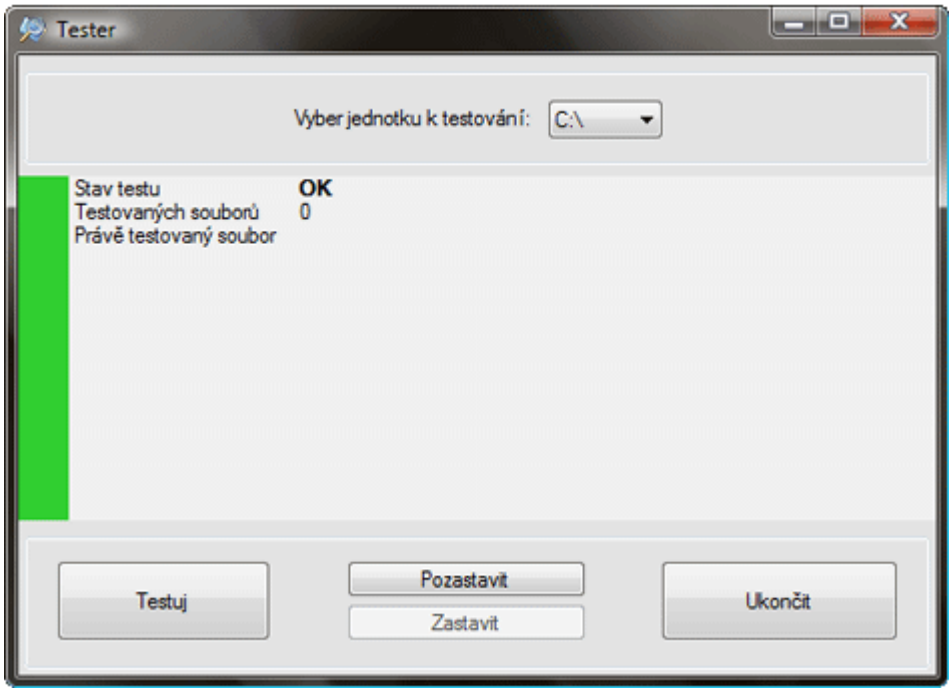
```
public bool nactiXML()
{
     XmlTextReader xmlReader = new
XmlTextReader("databazeSpywaru.xml");
     while (xmlReader.Read())
     {
          if (xmlReader.NodeType != XmlNodeType.XmlDeclaration)
          {
               while (xmlReader.MoveToNextAttribute())
               {
                    vzorkyInfo.Add(xmlReader.Value);
               }
          }
          if (xmlReader.NodeType == XmlNodeType.Text)
               vzorky.Add(encoding.GetBytes(xmlReader.Value))
     }
     if (vzorky == null) return false;
     else return true;

     < zdrojový kód zkrácen >
}
```

Obr. 14: Metoda „nactiXML“ ze třídy *Hledani.*

5.3 Popis grafického rozhraní

Jak vypadá program Tester ihned po startu, je vidět na obr. 15. Okno programu je opticky rozděleno na tři části.

- Horní část, ve které si uživatel může vybrat jednotku k testování
- Prostřední část, ta slouží k informování uživatele o průběhu testu
- Spodní část poskytuje možnosti k ovládání programu



Obr. 15: Program Tester, aplikace ihned po spuštění.

5.3.1 Pole pro volbu jednotky

Jedná se horní část okna programu. V tomto poli si může uživatel zvolit jednotku, kterou chce testovat. Nabídka jednotek je vysouvací a uživatelem nemůže být editována, program je totiž uzpůsoben tomu, aby po svém spuštění vždy načetl aktuální výběr jednotek. Ve výběru tak může uživatel dostat ať už všechny své pevné disky, případně mechaniky, tak i třeba různá aktuálně připojená paměťová média.

5.3.2 Pole pro informování o stavu testu

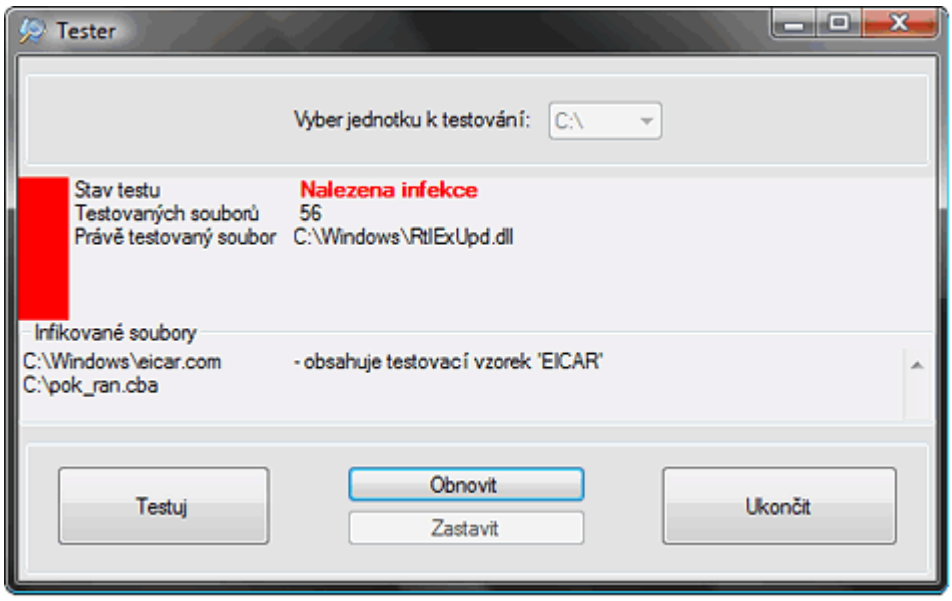
Prostřední část okna programu je rozdělena na tři části (viz obr. 16), pouze dvě jsou zpočátku viditelné. Třetí část se objeví poté, co program nalezne infekci.

Levá část obsahuje proužek, který nás informuje o tom, zda byla, či nebyla nalezna infekce. Počáteční barva proužku je zelená (viz obr. 15), tato barva také indikuje stav, který nám oznamuje, že zatím nebylo nic nalezeno. Pokud bude programem nalezena infekce, proužek změní svou barvu na červenou a do konce testování již takový zůstane (viz obr. 16).

Pravá část nám poskytuje mnohem podrobnější informace:

- *Stav testu* – poskytuje vesměs stejné informace, jako barevný proužek (zda byla či nebyla nalezena infekce)
- *Testovaných souborů* – slouží jako čítač, ukazuje, kolik souborů již bylo otestováno
- *Právě testovaný soubor* – zde se ukazuje absolutní cesta k právě testovanému souboru

Třetí část, pojmenovaná „Infikované soubory“, zobrazuje záznamy o infikovaných souborech – cestu k souboru a informace o daném vzorku. Jestliže v databázi spywaru není u vzorku uveden žádný popis, zobrazí se pouze cesta k souboru. Příklad je vidět na obr. 16, kde byly zatím nalezeny dva infikované soubory, z toho u souboru „eicar.com“ byl v databázi uveden popis, kdežto u souboru „pok_ran.cba“ nikoliv.



Obr. 16: Program Tester, nalezeny infikované soubory.

5.3.3 Pole pro ovládání programu – tlačítka

Jedná se o dolní část aplikace, kde se nachází celkem čtyři tlačítka. Jejich funkce se v průběhu běhu programu mění v závislosti na tom, v jakém je program momentálně stavu. Celkem mohou nastat čtyři stavy:

1. start programu
2. probíhá testování
3. testování je pozastaveno
4. testování bylo zastaveno nebo program ukončil testování (tento stav je podobný tomu prvnímu)

Start programu

Ihned po naběhnutí aplikace jsou aktivní tlačítka: *Testuj*, *Pozastavit* a *Ukončit*, tlačítko *Zastavit* je zatím neaktivní. Kliknutím na tlačítko:

- *Testuj* – program začne prohledávat zvolenou jednotku na výskyt spywaru
- *Pozastavit* – přestože je tlačítko aktivní, po kliknutí na něj se, v této fázi programu, nic nestane
- *Ukončit* – aplikace se ukončí

Probíhá testování

Program je ve stavu, kdy probíhá testování. Nyní jsou aktivní všechna tlačítka (Testuj, Pozastavit, Zastavit, Ukončit). Po kliknutí na tlačítko:

- *Testuj* – nic se nestane, jen nám program oznámí, že testování již probíhá
- *Pozastavit* – program pozastaví testování a zároveň se tlačítko změní na *Obnovit*
- *Zastavit* – testování je zastaveno, pokud byly nalezeny nějaké infikované soubory, výpis zůstane viditelný
- *Ukončit* – aplikace se ukončí

Testování pozastaveno

Tento stav nastane, jakmile se během spuštěného testování zmáčkne tlačítko *Pozastavit* (např. obr. 16). To ihned změní svůj název na *Obnovit*. Všechna ostatní tlačítka zůstanou aktivní, kromě tlačítka *Zastavit*. Aktivní zůstanou i doposud zjištěné výsledky.

Význam tlačítek *Testuj* a *Ukončit* je stejný jako ve stavu „probíhá testování“. Po kliknutí na tlačítko *Obnovit*, začne aplikace opět testovat od místa, kde přestala (tlačítko opět změní název na *Pozastavit*).

Testování zastaveno nebo program testování již dokončil

Ať už byl program zastaven pomocí tlačítka *Zastavit*, nebo již sám skončil s testováním, výsledku testu budou aktivní a to až do té doby, dokud se opět nespustí testování, nebo dokud

se neukončí aplikace. Aktivní jsou tlačítka stejné jako ve stavu „start programu“ (aktivní jsou tlačítka *Testuj*, *Pozastavit*, *Ukončit*). Po kliknutí na tlačítko:

- *Testuj* – program vymaže dosavadní výsledky a započne nové testování
- *Pozastavit* – nic se nestane, výsledky z předchozího testu zůstávají vypsány
- *Ukončit* – aplikace se ukončí

Jestliže program dokončil testování sám (nikoliv tlačítkem *Zastavit*), zobrazí se nové okno aplikace, které bude informovat o dosažených výsledcích.

5.4 Databáze spywaru

Databáze spywaru je napsána ve značkovacím jazyce XML. Tento jazyk je všeobecně využíván pro přenos dat mezi aplikacemi. Využívá se hlavně díky volnosti, se kterou se může struktura dokumentu vytvářet, u XML totiž není podstatný vzhled dokumentu, ale hlavně jeho obsah. Díky tomu, že se data vkládají do struktury, se dokument může libovolně transformovat na jiné typy. Pro popis struktury se používá jazyk DTD. Podrobněji o XML a DTD například v pramenu [23].

Dokument se dá později různě formátovat a převádět (např. vytvářet vzhled) pomocí rodiny jazyků XSL.

Na obr. 17 je naznačená struktura databáze spywaru[18], tak jak je použito v souboru databazeSpywaru.xml.

```
<ELEMENT! databaze (vzorek)>
<ELEMENT! vzorek (#PCDATA)>
<ATTLIST! vzorek
          info CDATA>
```

Obr. 17: Definice struktury databáze spywaru.

Konkrétní příklad záznamu vzorku spywaru v souboru „databazeSpywaru.xml“, je vidět na obr. 18 (pro větší přehlednost je vzorek zmenšen). Mezi tagy `<vzorek>` a `</vzorek>` je vepsán vzorek spywaru, v tagu `<vzorek>` je vložen atribut `info`, ve kterém jsou uloženy informace o příslušném vzorku.

[18] takhle by vypadal obsah souboru DTD

```
<databaze>
     <vzorek info ="- vzorek 'EICAR'">
              X5O!P%@AP[4\PZX54(P^)7CC)7H+H*
     </vzorek>
</databaze>
```

Obr. 18: Příklad záznamu v databázi spywaru.

5.5 Práce s programem

5.5.1 Návod pro práci s programem

Tento návod shrnuje v kostce instalaci a návod k používání programu pro běžného uživatele.

Postup při instalaci

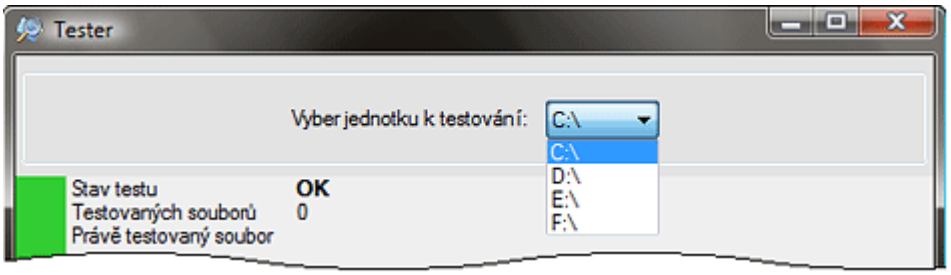
1) Nejprve nalezneme instalační soubor programu Tester na přiloženém CD (umístění viz příloha 2).
2) Spustíme instalaci.
3) Objeví se úvodní obrazovka průvodce instalací, dáme NEXT.
4) Tlačítkem BROWSE si můžeme vybrat, kam se má program nainstalovat. Také můžeme zvolit, komu má být program na počítači přístupný, pak dáme NEXT.
5) Kliknutím na NEXT spustíme instalaci.
6) Tlačítkem CLOSE dokončíme instalaci.

Návod k použití

Program spustíme souborem „Tester 2.0.exe“, který se nachází buď v defaultně nastaveném umístění[19], nebo na jiném místě zvoleném při instalaci. Jakmile je program spuštěn, naskytne se pohled jako na obr. 15.

1) Nejdříve si vybereme jednotku, která má být otestována (obr. 19).
2) Jakmile je vybrána jednotka, může se spustit samotné testování pomocí tlačítka *Testuj*.
3) Program může být kdykoliv během testování pozastaven (poté obnoven), nebo také ukončen.

[19] C:\Program Files\Tester 2.0\



Obr. 19: Program Tester, výběr jednotky k testování.

V případě, že program nalezne infekci, objeví se v okně aplikace informace o nákaze (obr. 16).

5.5.2 Návod k odzkoušení programu

Vývoj a testování programu probíhalo na operačním systému Microsoft Windows Vista. Na operačním systému Windows XP byl pouze odzkoušen.

Příprava

Před samotným spuštěním testování je nutné buď vypnout rezidentní ochranu antivirového programu, nebo tento antivirový program raději vypnout úplně. Databázový soubor „databazeSpywaru.xml“ totiž obsahuje, mimo jiných, i vzorek „eicar“, na který by antivirový program mohl reagovat a znemožnit tak odzkoušení programu.

Testovací vzorek „eicar“ je vytvořen institutem EICAR (European Institute for Computer Antivirus Research)[20], tento vzorek běžně slouží pro testování antivirových programů [21]. Všechny antivirové programy by jej tedy měly odhalit, případně zabránit v jakékoliv manipulaci s tímto souborem. Soubor je ale zcela neškodný a není třeba se ničeho obávat.

Testování

Aplikaci nainstalujeme stejným způsobem, který je popsán v kapitole 5.5.1.

Na přiloženém médiu jsou umístěny, mimo jiné, také soubory „eicar.com“ a „pok_ran.cba“ (umístění viz příloha 2). Na ty by měl program Tester reagovat jako na infikované. Program Tester provádí testování souborů vždy od rootovského adresáře, poté

[20] překlad: Evropský Institut pro Výzkum Počítačových Antivirů

pokračuje s testováním v adresáři, který je až na posledním místě (myšleno v abecedě)[21]. Je tedy vhodné umístit tyto soubory tak, aby se nemuselo dlouho čekat na výsledky.

Jakmile je program nainstalován, infikované soubory umístěny, antivirový program deaktivován, může začít samotný proces testování. Testování se spustí tlačítkem „Testuj“. Poté se může třeba vyzkoušet pozastavení programu a jeho obnovení. Může se také zkusit, v libovolném stavu programu jej zastavit, případně ukončit, pokusit se spustit druhý souběžný test apod.

Proces testování by se dal přehledně shrnout do několika bodů:

1) Deaktivovat rezidentní ochranu antivirového programu, případně jej deaktivovat úplně.
2) Nainstalovat program Tester 2.0.
3) Vhodně umísit infikované soubory.
4) Spustit Tester 2.0, vybrat jednotku k testování a spustit.
5) Odzkoušet funkčnosti tlačítek.

Další námět na testování by mohl například tento:

a) Vytvoření vlastního záznamu v souboru „databazeSpywaru.xml“ (popis struktury databáze viz kapitola 5.4).
b) Umístění nového vzorku například do textového editoru a uložit pod libovolným názvem (s libovolnou koncovkou).
c) Zahájit proces testování (viz návod uvedený výše)

Nutno poznamenat, že soubor „databazeSpywaru.xml“ má atribut „-h“, to znamená, že se jedná o skrytý soubor. Je tedy nutné podniknout příslušné kroky, aby se do něj dalo dostat. Ať už zadáním jeho absolutní cesty do vyhledávače, nebo jej dát vyhledávat, případně povolit zobrazování skrytých souborů. Soubor je umístěn ve stejném umístění jako soubor „Tester 2.0.exe“[22]. Soubor je skrytý proto, aby jej běžný uživatel nemohl tak snadno smazat, případně zasahovat do jeho vnitřních dat.

5.6 Možná budoucí rozšíření programu

Zjednodušeně řečeno, program *Tester* pracuje tak, že si načte do paměti několik souborů a ty postupně otestuje na výskyt všech vzorků spywaru, které má uložené v databázi. Tento postup se opakuje tak dlouho, dokud se neprohledá celý disk (respektive vybraná jednotka). Je

[21] tzn., začne například složkou „Windows“ a skončí například ve složce „Antivirus“
[22] defaultně je to C:\Program Files\Tester 2.0\

navržen tak, že na případnou infekci pouze upozorní. Rozšíření oproti minulé verzi spočívá v možnosti výpisu informací o typu infekce.

Jako rozšíření do budoucna by mohla být rozšířená práce s infikovanými soubory. Například by se tento soubor mohl přesunout na jinou, k tomuto účelu určenou, lokaci na disku, případně by se rovnou tento infikovaný soubor mohl smazat.

Jak bylo zmíněno výše, program prohledává celou, námi určenou, jednotku. Do budoucna by se mohla rozšířit možnost výběru oblasti k testování. Uživatel nemusí zrovna chtít otestovat celý disk, může chtít prozkoumat třeba jen jednu složku. Druhý extrém může být ten, kdy uživatel bude chtít otestovat kompletně všechna datová úložiště.

Ve stávající podobě však program dostatečně demonstruje současné možnosti detekce škodlivého kódu, další úpravy by spíše zvyšovaly uživatelskou přívětivost programu

6. Závěr

Využívání internetu se stává čím dál tím více nebezpečné – hrozby v podobě virů, spywaru, rootkitů aj. Když už se rozhodneme využít některou z jeho služeb, měli bychom být připraveni. Příprava může spočívat ve věcech, jako jsou například: pravidelná aktualizace systému, využívání kvalitního firewallu, využívání programů pro odhalování spywaru, rootkitů, virů, …, ale hlavně používat zdravý rozum.

Z výše zmíněných způsobů prevence útoků na náš počítač jsou nemálo důležité metody pro odhalování rootkitů a spywaru. Každý typ je sám o sobě již dostatečně nebezpečný, ale jejich kombinací se vše ještě více zhorší. Proto je nutné neustále rozvíjet metody pro jejich odhalování.

Ze základních metod se používají detekce na základě shody názvů nebo podle shody specifických vlastností. Kvalitnější metody pracují na základě vyhledávání vzorku kódu uvnitř souboru. Mezi sofistikované metody patří například heuristická analýza, která je schopná odhalit i doposud neznámé druhy infekcí, je ale náročnější na výpočetní výkon procesoru. Tyto principy detekce se dají aplikovat jak na spyware tak i na rootkyty. Ze samotného principu fungování rootkitů je ale jasné, že jejich detekce bude komplikovanější než u spywaru. Proto existují metody jako je například cross-view detection, prohledávání disku z bootovacího média apod.

Výstupem této bakalářské práce je demonstrační program na detekci spywaru – program Tester verze 2.0. Obsahuje jednoduché uživatelské rozhraní, které umožňuje jednodušší ovládání a v případě infekce i informace o dané nákaze (cesta k souboru + informace o nalezeném vzorku). Program pracuje na principu vyhledávání vzorku kódu uvnitř vybraného souboru. Tato metoda není tak dokonalá jako např. heuristická analýza, je rychlejší a je také účinnější než metoda shody názvů apod. Hodně stávajících detekčních programů tuto metodu využívá. Veškeré vzorky jsou uloženy v souboru databáze vzorků, kam je lze v případě potřeby doplňovat. Databáze také obsahuje základní informace o vložených vzorcích. Oproti předchozí verzi programu (vytvořenou v semestrálním projektu) obsahuje tato několik vylepšení – umožňuje prohledávání celého disku najednou, prohledávání bylo zrychleno apod.

Seznam použitých zkratek:

API	Application Program Interface
BIOS	Basic Input Output System
CD	Compact Disk
DLL	Dynamic Link Library
DoS	Denial of Service
DRM	Digital Rights Management
DTD	Document Type Definition
EICAR	European Institute for Computer Antivirus Research
EULA	End User License Agreement (Licenční smlouva s koncovým uživatelem)
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
I/O	Input / Output
IAT	Input Address Table
IRP	Input / Output Request Packet
KPP	Kernel Patch Protection
OS	Operační systém
USB	Universal Serial Bus
VMBR	Virtual Machine Based Rootkit (rootkit pracující na bázi virtuálního stroje)
WWW	World Wide Web
XML	eXtensible Markup Language (rozšiřitelný značkovací jazyk)
XSL	eXtensible Stylesheet Language (rozšiřitelný stylový jazyk)

Seznam literatury a použitých zdrojů

[1] BILBY, Darren. *Low Down and Dirty: Anti-forensic Rootkits* [online]. 2006 , 2006 [cit. 2007-12-09]. Text v angličtině. Dostupný z WWW: <www.ruxcon.org.au/files/2006/anti_forensic_rootkits.ppt>.

[2] BRECHLEROVÁ, Dagmar, VESELÝ, Arnošt. *IDS - Systémy detekce průniku a jejich význam pro bezpečnost IS* [online]. [2003] [cit. 2008-04-26]. Dostupný z WWW: <http://www.fem.uniag.sk/uninfos2003/zbornik/brechlerova_vesely.pdf>.

[3] BUDAI, David. *Rootkity: skrytí přítomnosti v systému* [online]. [2007] [cit. 2007-12-09]. Text v češtině. Dostupný z WWW: <http://www.zive.cz/Titulni-strana/Rootkity-skryti-pritomnosti-v-systemu/sc-21-a-135438/default.aspx>. ISSN 1214-1887.

[4] *Builder* [online]. Grafika Publishing, s.r.o., c1997-2003 [cit. 2007-11-21]. Text v češtině. Dostupný z WWW: <http://www.builder.cz/>.

[5] BUTLER, James, SPARKS, Sherri. *Windows rootkits of 2005* [online]. SecurityFocus, c2005 , 2005-11-04 [cit. 2008-11-09]. Text v angličtině. Dostupný z WWW: <http://www.securityfocus.com/infocus/1850>.

[6] *C# Practical Learning* [online]. Verze 3.0. FunctionX, Inc., c2006-2008 [cit. 2008-04-12]. Text v angličtině. Dostupný z WWW: <http://www.functionx.com/csharp/index.htm>.

[7] DELAHUNTY, James. *Sony BMG\'s Thomas Hesse on the \'rootkit\' DRM* [online]. 2005 [cit. 2007-11-27]. Dostupný z WWW: <http://www.afterdawn.com/news/archive/7013.cfm>.

[8] ELSON, David. *SecurityFocus : Intrusion Detection, Theory and Practice* [online]. c2007 [cit. 2008-04-25]. Dostupný z WWW: <http://www.securityfocus.com/infocus/1203/>.

[9] FREILING, Felix C., SCHWITTAY, Bastian. *Towards Reliable Rootkit Detection* [online]. 2007, 10-Aug-2007 [cit. 2007-12-09]. Text v angličtině. Dostupný z WWW: <http://pi1.informatik.uni-mannheim.de/filepool/publications/imf2007-rootkit-detection.pdf>.

[10] HÁK, Igor. *Moderní počítačové viry*. [s.l.], [2005]. 110 s. Univerzita v Hradci Králové, Fakulta informatiky a managementu. Vedoucí bakalářské práce Doc. RNDr. Josef Zelenka, CSc. Dostupný z WWW: <http://www.viry.cz/viry.cz/kniha/kniha.pdf>.

[11] *IAT Function Hooking* [online]. c2003 [cit. 2007-12-09]. Text v angličtině. Dostupný z WWW: <http://sandsprite.com/CodeStuff/IAT_Hooking.html>.

[12] KING, Sam, et al. *SubVirt: Implementing malware with virtual machines* [online]. 2006 [cit. 2007-10-28]. Text v angličtině. Dostupný z WWW: <http://www.eecs.umich.edu/virtual/papers/king06.pdf>.

[13] *Low Down and Dirty: Anti-forensic Rootkits* [online]. Security-Assessment.com, 2006 [cit. 2007-10-23]. Text v angličtině. Dostupný z WWW: <http://www.security-assessment.com>. www.ruxcon.org.au/files/2006/anti_forensic_rootkits.ppt.

[14] *Methods of spyware detection* [online]. c2004 [cit. 2007-10-20]. Text v angličtině. Dostupný z WWW: <www.sunbelt-software.com/documents/battling_spyware_3.pdf>.

[15] *Microsoft reboot reduction initiative* [online]. c2007 [cit. 2007-12-09]. Text v angličtině. Dostupný z WWW: <http://technet2.microsoft.com/windowsserver/en/library/e55050fc-22c9-4984-9bae-b8b0527334721033.mspx?mfr=true>.

[16] *Microsoft Visual C#* [online]. FunctionX, Inc., c2006-2007, Last Update: Sunday, February 24, 2008 [cit. 2008-04-12]. Text v angličtině. Dostupný z WWW: <http://www.functionx.com/vcsharp/index.htm>.

[17] MURMAK, Petr. *DRM od SONY za hranicí slušnosti* [online]. CD-R server, c1998-2007, Úterý, 1. listopadu 2005 [cit. 2007-12-08]. Text v češtině. Dostupný z WWW: <http://www.cdr.cz/a/15736>. ISSN 1213-2225.

[18] *Rootkity: skrytí přítomnosti v systému* [online]. c2007 , 17. 4. 2007 [cit. 2007-10-20]. Text v češtině. Dostupný z WWW: <http://www.zive.cz/Titulni-strana/Rootkity-skryti-pritomnosti-v-systemu/Je-rootkit-program/sc-21-sr-1-a-135438-ch-53546/default.aspx>. ISSN 1214-1887.

[19] RUSSINOVICH, Mark . *Sony, Rootkits and Digital Rights Management Gone Too Far* [online]. 10/31/2005 [cit. 2007-12-09]. Text v angličtině. Dostupný z WWW: <http://blogs.technet.com/markrussinovich/archive/2005/10/31/sony-rootkits-and-digital-rights-management-gone-too-far.aspx>.

[20] *Spyware, ochrana proti spyware, doporučené a účinné antispyware řešení* [online]. Gesto Communications, c2007 [cit. 2007-10-28]. Text v češtině. Dostupný z WWW: <http://www.barracudanetworks.cz/barracuda-networks-o-spyware-ochrana-proti-spyware-doporucene-a-ucinne-antispyware-reseni.htm>.

[21] *Standardní testovací soubor EICAR* [online]. ALWIL Software a.s., c1998-2008 [cit. 2008-04-11]. Text v češtině. Dostupný z WWW: <http://www.avast.cz/cze/eicar-antivirus-test-file.html>. Http://www.eicar.org/download/eicar.com.

[22] VAŠEK, Václav. *Sony opět používá techniku rootkit* [online]. CD-R server, c1998-2007, Úterý, 28. srpna 2007 [cit. 2007-12-08]. Text v češtině. Dostupný z WWW: <http://www.cdr.cz/a/22235>. ISSN 1213-2225.

[23] *Wikipedia : Extensible Markup Language* [online]. Upravené vydání. 2001- , naposledy editováno 18. 3. 2008 v 19:33 [cit. 2008-04-22]. Text v češtině. Dostupný z WWW: <http://cs.wikipedia.org/wiki/XML#Syntaxe_XML>.

[24] *Wikipedia : Flowchart* [online]. Upravené vydání. [2007-] , last modified 05:40, 10 December 2007 [cit. 2007-12-10]. Text v angličtině. Dostupný z WWW: <http://en.wikipedia.org/wiki/Flowchart>.

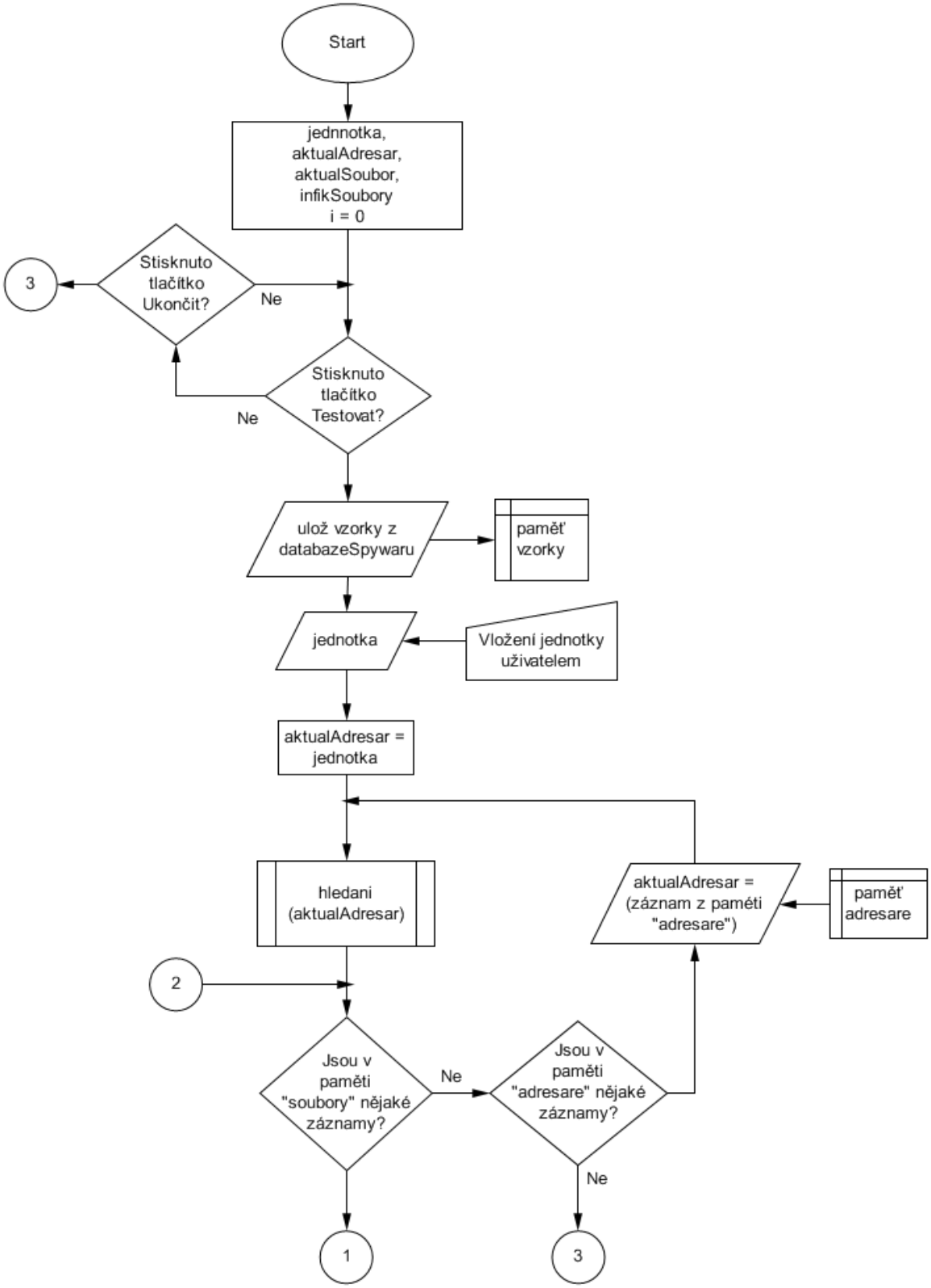
[25] *Wikipedia : Rootkit* [online]. Upravené vydání, [2007-] , last modified 21:51, 3 December 2007 [cit. 2007-12-08]. Text v angličtině. Dostupný z WWW: <http://en.wikipedia.org/wiki/Rootkit>

[26] *Wikipedia : Spyware* [online]. Upravené vydání, [2007-] , last modified 17:14, 4 December 2007 [cit. 2007-10-05]. Text v angličtině. Dostupný z WWW: <http://en.wikipedia.org/wiki/Spyware>

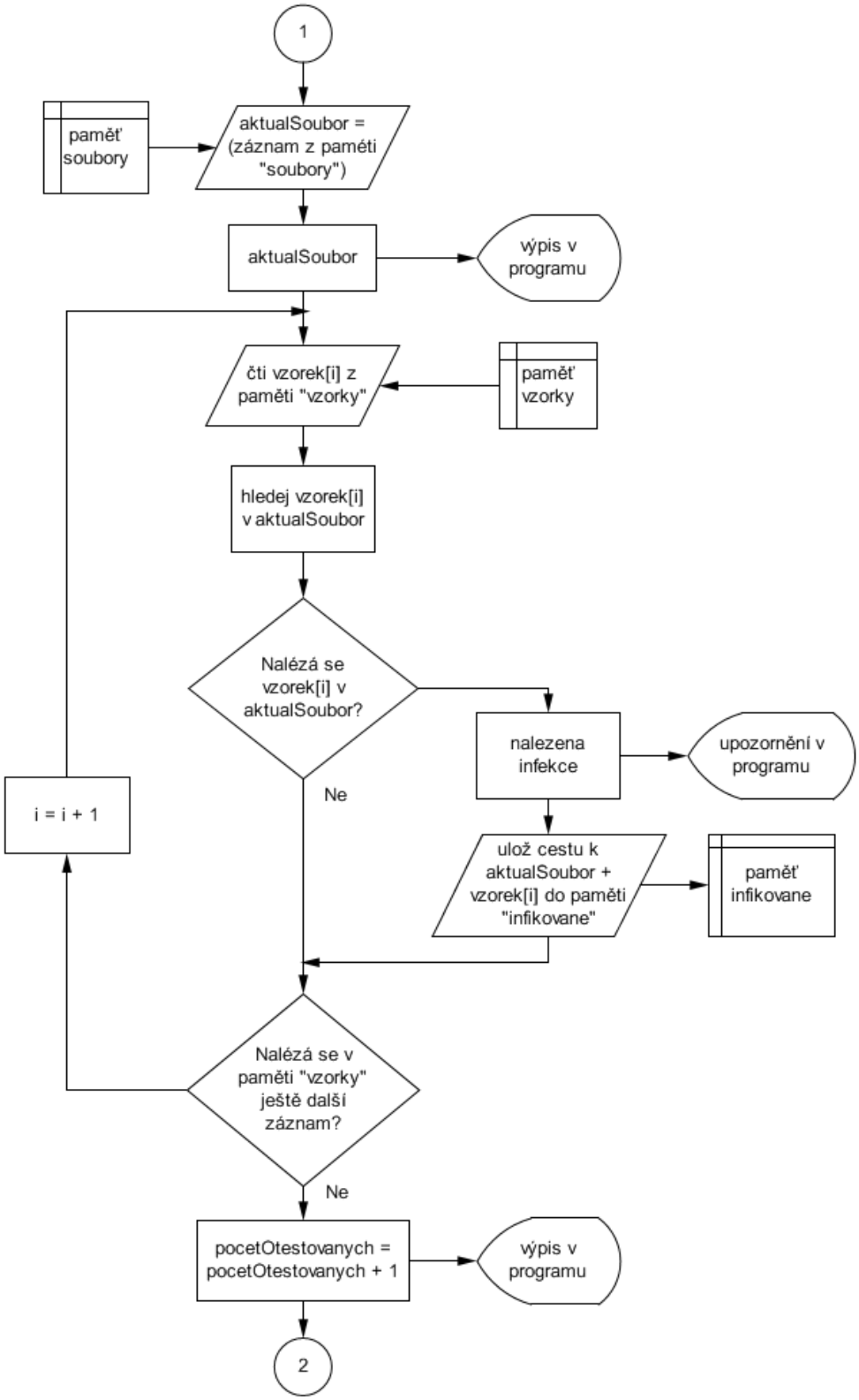
Seznam příloh

Příloha 1: Vývojový diagram programu Tester

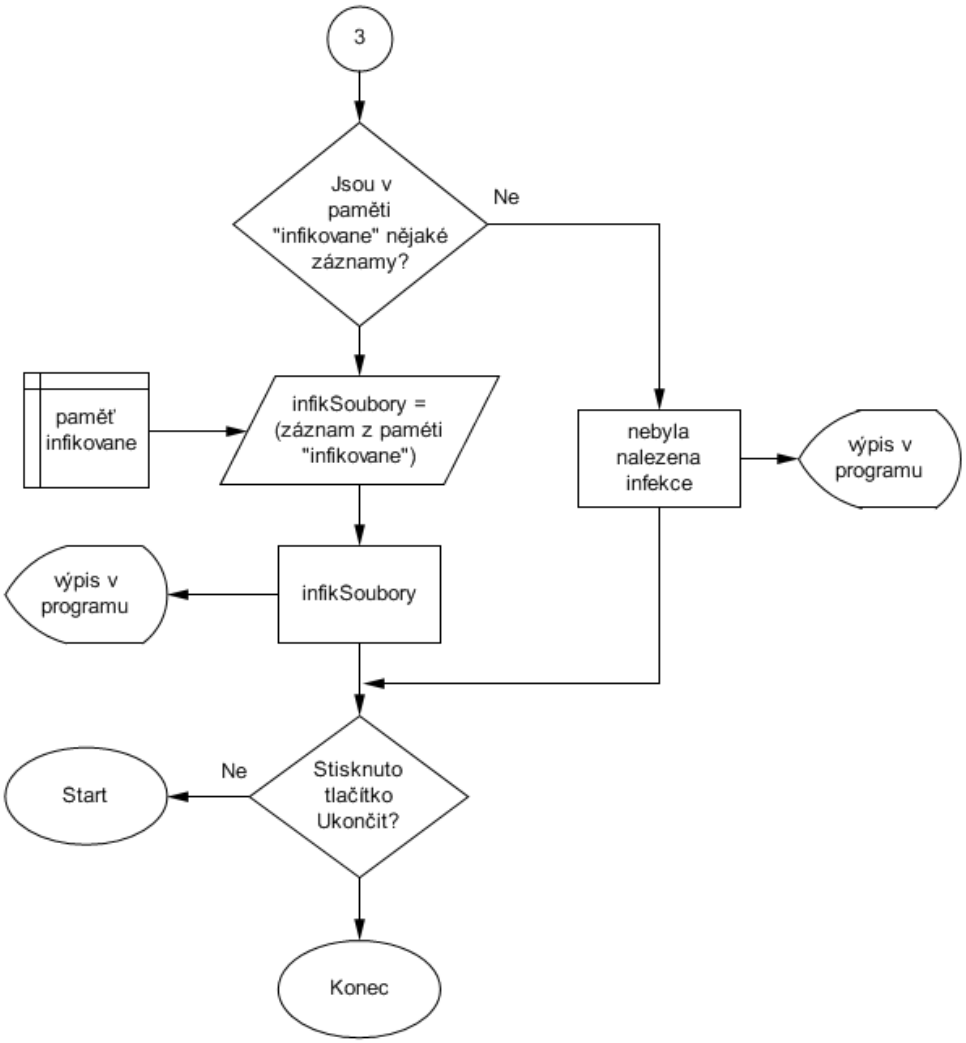
Příloha 1.A: Prohledávání disku



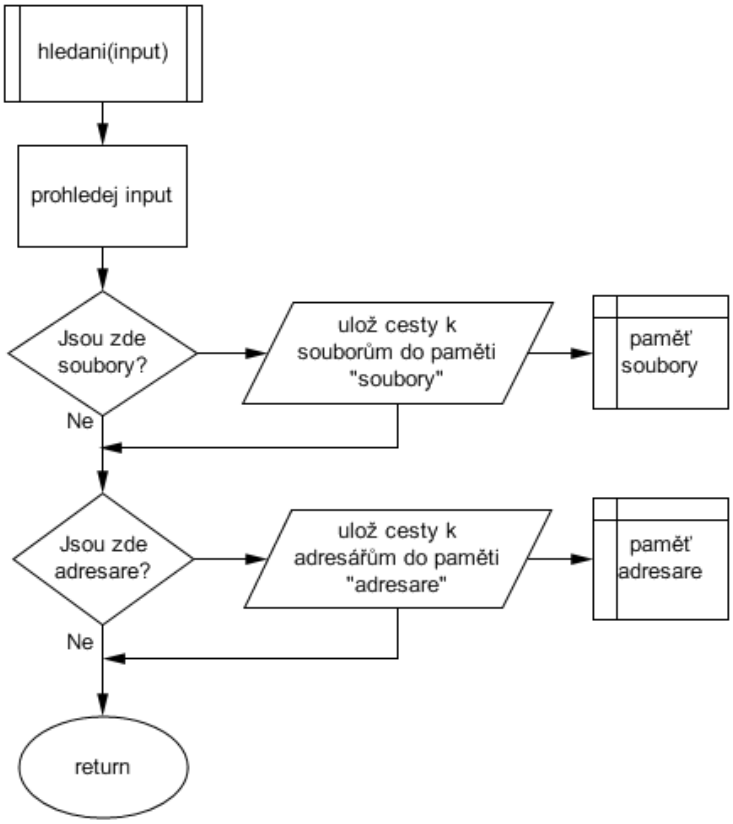
Příloha 1.B: Testování souboru



Příloha 1.C: Výpisy v programu



Příloha 1.D: Podproces hledani()



Příloha 2: Obsah přiloženého média