

## **Záložka: Internetové útoky**

### **Název: Rootkit – postrach každého administrátora**

**Perex:** *Domníváte se po ověření všech logů a kontrolních součtů, že je váš počítač v pořádku? A víte jistě, že to, co vidíte, neprošlo nejprve „cenzurou“ chytrého rootkitu? Rootkitu si totiž většinou všimnete až když hacker udělá chybu a něco přestane fungovat...*

Získání práva administrátora systému je nejvyšší metou hackera. Lze jí dosáhnout řadou způsobů. Může se jednat o nedostatečně záplatovaný systém, 0-day exploit (viz **box 1**) nebo jen na chvíli přístupnou neuzamknutou konzolu administrátora. V každém případě útočník usiluje o to, aby po sobě v napadeném počítači zanechal co nejméně stop, avšak typicky si chce si ponechat možnost opakovaného budoucího přístupu do něj. Často je takový počítač zneužit k nejruznějším nelegálním či jinak škodlivým aktivitám.

#### **Co je to rootkit?**

Wikipedia [34] definuje *rootkit* jako sadu nástrojů používaných hackerem po napadení počítačového systému pro udržení přístupu k systému a jeho využití ke škodlivým aktivitám.

Samozřejmou snahou útočníka je skrytí svých aktivit před zrakem skutečného administrátora systému a právě na to se hodí rootkit. Ten mu umožní provést všechny potřebné operace „na jedno kliknutí“ a v kombinaci s automatickým zkoušením exploitů může postup útočníka vypadat tak, že večer spustí program a ráno si prohlédne seznam ovládnutých strojů.

#### **Box 1**

##### **Co je exploit?**

Žádný program není bez chyb. Dokud o chybě nikdo neví, nic se neděje, jakmile je však chyba objevena, může být zneužita. Program, který slouží ke zneužití nějaké chyby, nazýváme exploit. Dopad případného zneužití exploitu závisí na konkrétní chybě, může například způsobit ukončení aplikace a tím výpadku její dostupnosti nebo díky chybě v programu může útočník získat práva, která mu nepatří. Jakmile je nalezená chyba zveřejněna (resp. dozví se o ní tvůrce programu), pracuje se více či méně usilovně na odstranění chyby a vydání opravené verze programu. Obvykle není třeba instalovat opravený program celý znovu, ale – podle rozsahu chyby – se aplikuje pouze tzv. bezpečnostní oprava, záplata (anglicky patch), která obsahuje pouze změněnou část.

Exploit, který zneužívá chybu, pro niž ještě nebyla vydána záplata (neboť chyba objevená hackery dosud nebyla oznámena tvůrci programu nebo tvůrce programu zatím nestihl vydat záplatu), se nazývá 0-day exploit. Doba, která uplyne od zveřejnění chyby do vydání záplaty bývá označována jako okno ohrožení (anglicky window of exposure).

## **Unixové systémy**

Útok, při kterém útočník získá administrátorská práva, je obvykle doprovázen řadou záznamů v logovacích souborech pořízených příslušnými systémovými mechanismy. Prvním krokem útočníka proto bude likvidace těchto usvědčujících záznamů. V unixových systémech se jedná především o záznamy syslogu, wtmp, utmp a lastlog (viz **box 2**). Ke smazání lze použít jednu ze známých utilitek wted, zapper nebo z2, které mažou nebo alespoň nulují záznamy. Vynulované záznamy sice nejsou standardními utilitami operačního systému zobrazovány, specializované programy (pro detekci manipulace se záznamy) si takových záznamů všímají a informují o nich správce. Proto je (pro útočníka) bezpečnější problematické záznamy úplně smazat. Pro administrátora to pak znamená, že detekce „děr“ v záznamech není úplně spolehlivá. Ačkoliv úprava logovacích záznamů není hlavní náplní činnosti rootkitů, jsou tyto obvykle vybaveny nástroji na mazání záznamů, a detektory rootkitů naopak utilitami na vyhledávání „děr“ v logovaných datech.

## Box 2

### Co je wtmp, utmp, lastlog a syslog?

Neinteraktivní, trvale běžící programy (v unixových systémech nazývaných démony) většinou nevypisují informace o událostech uživateli (administrátorovi) přímo, ale prostřednictvím systému zpracování logů. U unixových systémů se nejčastěji jedná o program syslog, který přijímá údaje od jiných programů a na základě své konfigurace dále zpracovává. To může zahrnovat ukládání záznamů do souborů (typicky do adresáře /var/log), odesílání na jiný počítač, ev. i výpis na konzolu. Možnost pozdější modifikace záznamů závisí na způsobu zpracování záznamů: pokud se jedná o ukládání do souboru, může obvykle administrátor (nebo hacker s jeho právy) soubor modifikovat i později, při odesílání záznamů na jiný stroj je pro modifikaci nutný přístup k tomuto stroji.

Kromě údajů o běhu aplikací ukládá operační systém také údaje o přihlašování uživatelů. Nejčastěji se jedná o soubory lastlog (pro každého uživatele je udržován záznam o jeho posledním přihlášení), wtmp (každé přihlášení přidává jeden záznam) a utmp (seznam aktuálně přihlášených uživatelů). Tyto soubory se také obvykle nacházejí v adresáři /var/log.

Další krokem útočníka bývá umožnění následného snadného přístupu k napadenému počítači (tj. instalace zadních vrátek, tzv. backdoor). Zde má útočník řadu možností od jednoduchého přidání uživatele s UID 0 (viz **box 3**) nebo navázání (anglicky bind) rootovského shellu na určitý port (o principech komunikace klient – server blíže **box 4**), přes úpravu přihlašovacích procedur (změna programu login, konfigurace r\* služeb apod.) až po speciality typu zadní vrátka přístupná pouze z určitých IP adres a komunikující přes běžně používaný port (např. 80) bez vlivu na funkčnost aplikace normálně navázané k tomuto portu, tj. http [16]. Kromě přihlášení ze sítě je obvykle umožněna i eskalace privilegií běžného uživatele systému, kdy po zadání speciálního parametru (a případně hesla) některému běžnému SUID (např. chsh) programu (který je takto útočníkem změněn) je například možné přímo získat rootovský shell. Instalace zadních vrátek bývá standardní součástí rootkitů, často jde o úpravu ssh serveru spočívající v umožnění přístupu útočníka po zadání speciálního hesla.

### Box 3

#### Pár unixových pojmů

**UID:** Ačkoliv uživatelé mají svá uživatelská jména, pomocí kterých se přihlašují k systému, počítači se lépe pracuje s čísly, a proto je s každým uživatelským jménem svázáno i numerický uživatelský identifikátor (tzv. UID). Mezi UID má zvláštní postavení hodnota 0. Toto UID je vyhrazeno pro správce (administrátora, superuživatele, root-a) a jsou s ním svázána speciální práva (někdy se lze setkat s výrokem že „práva takového uživatele nejsou prověřována“). Obvykle je UID 0 přiřazeno uživatelskému jménu root, není to však podmínkou a účet administrátora se může jmenovat jinak. Může dokonce existovat několik uživatelských jmen se stejným UID včetně UID 0. Pokud uživatel spustí program, má tento práva daného uživatele (tzv. vlastníka procesu).

**SUID:** Protože pro některé úkony, např. pro přístup do zvláště chráněných souborů obsahujících hashe hesel (což je nutné např. při změně hesla) uživatelům nestačí jejich běžná práva, byla zavedena určitá výjimka při získávání přístupových práv. Pokud je u nějakého souboru nastaven tzv. SUID (zkratka z *set-UID*) bit, pak program v souboru uložený nepoběží s právy uživatele, který jej spustil, ale s právy uživatele, kterému patří onen soubor. Protože není snadné napsat program tak, aby bezpečně prováděl právě to co má a to za všech okolností, které mu spouštějící uživatel připravil, bývají SUID programy častým cílem útočníků.

**r\* služby:** V dobách, kdy Internet ještě býval bezpečný, se autentizace často odvíjela pouze od IP adresy příchozího spojení. Na tomto principu byla založena řada služeb, které (po patřičném nakonfigurování) umožňovaly bez další autentizace heslem přihlášení na vzdálený stroj (rsh, rlogin), kopírování souborů (rcp) a spouštění programů (rexec).

**ifconfig:** Příkaz pro konfiguraci síťových rozhraní (zkratka z interface configuration).

**ls:** Příkaz pro výpis obsahu adresáře (obdoba dosového dir).

**echo:** Příkaz pro opis argumentů, přičemž argumenty jsou před vypsáním expandovány příkazovým interpretem (např. \* je expandována na seznam souborů v pracovním adresáři); (nelze-li, jsou argumenty jen opsány).

**ps:** Příkaz pro výpis procesů.

### Box 4

#### Princip klient – server komunikace

Klient-server je vztah mezi dvěma počítačovými programy, kdy jeden program (klient) posílá druhému programu (serveru) své požadavky a ten je plní. Příkladem může být žádost o určitý soubor.

Princip klient-server je často používán např. v Internetu. Jsou na něm založeny mimo jiné protokoly http, ftp, smtp a pop3. Serverem je program, který se naváže na určitý port a naslouchá na něm. Klient se k tomuto portu připojí a sdělí serveru svůj požadavek (například kterou webovou stránku si chce stáhnout). Běžné protokoly mívají přidělen svůj port (well known port), který je použit, pokud není uvedeno jinak (např. protokol https používá implicitně port 443).

Pojďme se však podívat na hlavní část rootkitů, jíž je kód (program), který se snaží zakrýt všechny kompromitující aktivity útočníka a nechat administrátora v domněnku, že všechno je

v pořádku, server nebyl úspěšně napaden a žádná podezřelá aktivita není nebo nebyla vyvíjena.

## **Trojský kůň**

První verze rootkitů se snažily zakrýt svoji přítomnost nahrazením základních utilit administrátora jinými programy – trojskými koni. Například program ls pro zobrazení obsahu adresáře byl nahrazen upravenou verzí, která se chovala zdánlivě stejně jako originál, ale nezobrazovala některé soubory útočníka (a rootkitu samotného). Obdobně program ps pro získání seznamu spuštěných procesů skrýval procesy útočníka. Tímto způsobem upravených utilit bývá celá řada; aby iluze, že všechno je v pořádku, byla co nejdokonalejší. Například Rootkit IV obsahuje upravené verze utilit ls, find, du, ps, top, pidof, netstat, killall, ifconfig, crontab, tcpd a syslogd. Administrátor, který používá tyto podvržené utility, nemusí registrovat aktivity útočníka po dlouhou dobu. Podezření obvykle pojme, až když něco přestane pracovat tak, jak byl zvyklý.

Ačkoliv výhodou rootkitů ve formě trojských koní je platformní nezávislost v tom smyslu, že jsou k dispozici ve zdrojovém kódu a útočník je kompiluje až na cílovém systému, je toto zároveň i nevýhodou. Podstatné rozdíly mezi parametry příkazové řádky jsou nejen mezi jednotlivými unixovými systémy, ale i mezi jednotlivými distribucemi jednoho systému a to i u tak běžných programů jako je ls nebo ps.

Pokud útočník udělá chybu a použije nesprávnou verzi trojské utility pro daný systém, může se stát, že chování utility bude odlišné, což upozorní administrátora, eventuálně i obyčejné uživatele. Existuje celá řada dalších možností odhalení útočníka, neboť útočník nemůže upravit všechny potenciálně používané nástroje a dříve či později může administrátor na nesrovnalosti přijít. Oblíbeným trikem může být použití programu dd (který obvykle nebývá nahrazován trojskou verzí), rozdíl mezi výsledkem „ls“ a „echo \*“ či použití souborového manažeru mc.

## **Modifikace jádra**

O řád lepší úroveň ukrytí mají rootkity nové generace, které přímo upravují jádro operačního systému. Princip spočívá v tom, že místo úpravy uživatelských programů typu ls ke skrytí souborů, procesů apod. se rovnou upraví volání jádra tak, že všechny uživatelské programy (včetně těch spuštěných administrátorem) se již dostanou k „cenzorovaným“ informacím. V žádném z moderních operačních systémů nemají uživatelské programy (programy běžící v uživatelském prostoru) přímý přístup k hardwaru a musí žádat operační systém o zprostředkování formou definovaného systémového volání (tzv. Application Program Interface – API). Změnou funkcí tohoto API (např. volání open – otevření souboru, read – čtení ze souboru, getdents – čtení adresáře) je možné lehce ukrýt soubory, procesy, síťová spojení apod. před všemi procesy v systému. Náročnost změny chování jádra závisí na konkrétním systému (administrátorská práva jsou vyžadována v každém případě).

V případě operačního systému Linux do verze jeho jádra 2.4. s aktivovaným modulárním systémem je úprava hračkou [4, 10] (jádra do verze 2.4. exportují symbol s tabulkou systémových volání `sys_call_table` a vložení nového modulu je nejjednodušší forma modifikace jádra), od verze 2.5. již tabulka systémových volání není exportována (zřejmě právě z důvodu jejího zneužívání), lze ji však rychle najít v kódu obsluhy přerušení 0x80 (systémového volání) [11, 12]. Linuxové jádro bez modulárního systému lze modifikovat přímým zápisem do paměti (přes speciální zařízení `/dev/kmem`) [11, 12]. Rootkity

modifikující jádro existují i pro operační systémy Solaris (Sparc i Intel), OpenBSD a FreeBSD, linuxové rootkity pro 32bitovou architekturu Intel jsou však zdaleka nejrozšířenější.

### **Jak se bránit?**

Předně je potřeba zdůraznit, že úloha rootkitu začíná až po získání administrátorských oprávnění, takže je třeba úsilí zaměřit především takovým směrem, aby útočník tato práva nezískal. 100% bezpečnost však neexistuje, a proto je nutné prevenci doplňovat detekcí.

Detekce starších trojských rootkitů je relativně snadná. Protože útočník zaměňuje binární verze některých utilit za svoje verze, stačí kontrolovat změny těchto klíčových utilit programy typu tripwire nebo „rpm –V“. Samozřejmě pak musí být jak samotný program pro výpočet hashů tak i vypočítané hashe programů zabezpečeny z hlediska integrity.

Detekovat rootkity modifikující jádro operačního systému je možné například pomocí testování vedlejších efektů upravených systémových volání [17], kontrolou neobvyklých příznaků procesů (rootkity je používají pro interní potřeby), kontrolou integrity tabulky systémových volání či procházením paměti a hledáním známých vzorů daného rootkitu (např. podpis autora), struktur zavedených modulů (pro moduly, které se později skryly vypojením z lineárního seznamu), struktur procesů (obdobně pro skryté procesy) apod. Řada rootkitů například zakrývá promiskuitní režim síťového rozhraní (protože jej používá k odposlouchávání síťového provozu na odchyťování nešifrovaných hesel); jednoduchý test spočívá v uvedení rozhraní do promiskuitního režimu (např. spuštěním programu tcpdump) a zkontrolováním, zda příkaz `ifconfig` tento režim korektně zobrazí či nikoliv.

Existují rovněž specializované programy pro vyhledávání rootkitů. Mezi takové programy patří například rkhunter [25] a chkrootkit [18], které jsou zaměřeny především na trojské koně. Protože se své činnosti spoléhají na jádro, jsou oba dva na moderní rootkity krátké. Paměť jádra přímo analyzují například detektory kstat [26] a rdetect [27], které k paměti přistupují pomocí čtení zařízení `/dev/kmem` a jsou tak závislé na správné činnosti systémových volání `open` a `read` s tímto zařízením – zde se rootkity zatím neangažují, to se však v budoucnosti může změnit. Pro FreeBSD existuje podobný LKM Detector (`sec_lkm`) [28].

Jakýkoliv detektor je však více či méně závislý na spolupráci jádra. Pokud rootkit ví, co detektor při své práci sleduje, může mu vždy podstrčit nesprávné hodnoty nebo jej zcela deaktivovat (jde tedy v principu o to, jestli máte aktuálnější rootkit či detektor rootkitů). Můžeme sledovat jasný trend, že i detekce a obrana vůči modifikaci jádra rootkitem se stěhuje do kódu jádra (například speciálních modulů). Pro opravdu spolehlivou detekci je však nutné opustit (potenciálně) modifikované jádro, nabootovat jádro čisté a zjistit, jaký je pravý stav věcí.

Vraťme se ale na chvíli opět k prevenci. Protože aktivní modulární systém (LKM) je nejjednodušším způsobem modifikace jádra, znamená jeho vypnutí eliminaci celé řady rootkitů, které jsou na této vlastnosti jádra závislé. Dalším způsobem obrany může být omezení manipulace s pamětí pomocí zařízení `/dev/kmem`. Existují úpravy jádra, které nepovolují u zařízení `/dev/kmem` čtení ani zápis. Například Linuxová distribuce Fedora má tuto úpravu ve svých jádrech zahrnutu automaticky. (Potom ale samozřejmě nefungují ani detektory rootkitu závislé na čtení z `/dev/kmem`.) Existuje také celá řada drobných modifikací jádra, které ztěžují útočníkům zavedení rootkitu, např. St. Michael [23], tripwire pro tabulku systémových volání [6] apod.

BSD systémy ztěžují práci útočníkům zavedením bezpečnostních úrovní, které neumožňují manipulaci s jádrem bez rebootu systému. I tato ochrana však není neprůstředná [2].

## Microsoft Windows

Jak je to s rootkity pod operačním systémem Windows? Ačkoliv je slovo rootkit přímo odvozeno od slova root, což, jak již víme, je administrátor systému UNIX, není princip rootkitu omezený pouze na operační systémy typu UNIX.

Rootkity pro operační systém Windows také prošly vývojem od jednodušších ke složitějším. První generace rootkitů ve formě trojských koní systémových utilit se pod Windows příliš neprosadila, druhá generace spočívající v modifikaci aplikace v paměti nebo některých částí operačního systému (např. přilinkování jiných funkcí či úprava tabulky systémových volání) je již rozšířena zcela běžně. Třetí generace rootkitů jde ještě hlouběji do operačního systému a modifikuje přímo dynamické struktury jádra bez volání funkcí jádra (Direct Kernel Object Manipulation -- DKOM). Čtvrtá generace rootkitů navíc umí díky trikům se správou paměti skrýt svůj kód v paměti před detektory rootkitů [13].

Pojďme se podívat, jakým způsobem může rootkit pod Windows například skrývat soubory [14]: po spuštění příkazu „dir“ se využijí funkce FindFirstFile a FindNextFile v knihovně kernel32.dll. Při linkování knihovny se používá importní adresní tabulka, jejíž manipulací je možné docílit změny funkčnosti daných funkcí (takto skrývá soubory např. rootkit Mersting [29]). Jiné rootkity (např. Vanquish [20]) přímo mění volání API v paměti procesu tak, aby se volal kód rootkitu. Funkce rozhraní Win32 FindFirst(Next)File při své činnosti volají funkci NtQueryDirectoryFile z knihovny ntdll.dll a zde mají rootkity další možnost modifikace funkčnosti (například rootkit Hacker Defender [21]).

NtQueryDirectoryFile slouží ke spuštění systémového volání a zde se tak dostáváme z kódu běžícího v uživatelském režimu ke kódu jádra. V jádře je nejprve použita tabulka systémových volání (System Service Dispatch Table -- SSDT); tu ke skrývání souborů modifikuje například ovladač, který je součástí keyloggeru ProBot [22]. Dále jsou využity služby správce vstupně výstupních operací a ovladačů souborových systémů, kam je také možné vložit specializované ovladače, jejichž funkcí bude skrývat některé soubory (zde typicky působí ovladače, které jsou součástí komerčních skrývacích programů jako např. Hide Folders XP [19]).

Hledání škodlivého kódu (programů) mají pod Windows na starost především antivirové programy. Ty procházejí procesy v paměti a soubory na disku a hledají v nich známé (a pomocí heuristik i neznámé) vzory. Protože však rootkity skrývají svůj kód v paměti i na disku před uživatelskými procesy (jakými jsou i antiviry), klasické antivirové programy rootkity obvykle nenajdou. Pro hledání rootkitů v operačním systému Windows existuje celá řada drobných utilitek, které s větší či menší úspěšností hledají známé projevy rootkitů (změněné tabulky systémových volání, seznamy procesů apod.). Mezi takové programky patří například Kernel Hidden Process/Module Checker [7], F-secure blacklight ([obrázek 1](#) ukazuje detekci rootkitu Hacker Defender) [30], Patchfinder [31], VICE [32]. V blízké budoucnosti zřejmě bude tato činnost přímo součástí antivirových programů.

## Odhalení vlastní zbraní

S rostoucí rafinovaností rootkitů je jejich detekce stále náročnější. Jestliže útočník upraví jádro, jsou možnosti jeho odhalení pomocí programů běžících v uživatelském režimu jen velmi omezené. Skutečnost, že rootkit něco ukrývá (ať už jsou to soubory, adresáře, položky registrů, procesy či jiné struktury operačního systému), je však možné lehce obrátit proti němu a využít toho při detekci [15]. Pro srovnání potřebujeme nestranný pohled na systém.

Při srovnávání sledujeme rozdíl v seznamu souborů, adresářů, záznamů registrů apod. z pohledu potenciálně modifikovaného jádra a z pohledu nestranného. Získat nestranný pohled na soubory (včetně souborů s registry) můžeme buďto přímým čtením fyzického zařízení (protože však budeme používat potenciálně nespolehlivé jádro, nemusí být výsledek vždy přesný – zatím využíváme toho, že rootkity toto chování neupravují) nebo zavedením čistého jádra (zde však díky určitému časovému odstavu může dojít k řadě legitimních změn).

Nabootování čistého jádra spočívá v případě Windows ve využití bootovacího CD s Windows PE [24], záchranného (případně i Live) CD v případě unixových systémů nebo vložení disku do jiného počítače s čistou (nemodifikovanou) verzí operačního systému. Pro získání seznamu souborů můžeme použít běžný příkaz `ls` či `dir` a s vytvořením rozdílového seznamu nám pomůže program `diff` (`windiff`). Microsoft Research již na výzkumu v této oblasti pracuje, samostatný produkt (nazvaný `GhostBuster` [14]) však zatím není volně k dispozici a funkčnost analýzy rozdílů bude zřejmě časem přidána do programu `Microsoft Antispyware`. Na podobném principu pracuje i program `RootKitRevealer` (ten porovnává výsledek vysokoúrovňového API s nízkoúrovňovým přístupem k disku) [33]. Ukázkou výstupu programu `RootKitRevealer` zobrazuje [obrázek 2](#).

Pro nestranný pohled do paměti nám však přebotování nepomůže a musíme využít speciální hardware. Potřebujeme přídavné zařízení, které umí přečíst obsah paměti pomocí DMA (direct memory access – přímý přístup do paměti) přímo z paměťových čipů bez pomoci procesoru a operačního systému [15, 9]. V takto získaném výsledku je pak možné vyhledávat vzory rootkitu či jiné anomálie.

## Závěr

Tímto příspěvkem jsem rozhodně nechtěl čtenáře DSM strašit, nakonec rootkity nejsou žádnou novinkou posledních dnů, na podobném principu úpravy systémových volání pracovala mimo jiné řada virů již pro operační systém MS-DOS. Je však vhodné nezapomenout, že štěstí přeje připraveným, proto je dobré znát základní principy jejich činnosti i metody obrany a při nezvyklém chování operačního systému si na rootkity vzpomenout. Nejjistější způsob detekce bohužel vyžaduje restart počítače, což je zvláště v případě produkčních serverů luxus, který není možné si dopřát příliš často, přesto je vhodné podle hesla „důvěřuj, ale prověřuj“ čas od času kontrolu provést. Zajímavý je také fakt, že v oblasti rootkitů je Česká i Slovenská republika velice aktivní (a to na obou stranách barikády, tedy jak v oblasti tvorby rootkitů, tak i tvorby detektorů rootkitů).

Zdeněk Říha  
zriha@fi.muni.cz

## Odkazy

- [1] BRUMLEY, D. Invisible intruders: rootkits in practice, ;login:, září 1999, <http://www.usenix.org/publications/login/1999-9/features/rootkits.html>.
- [2] EREN, S. Smashing The Kernel Stack For Fun And Profit, *Phrack*, roč. 0x0b, č. 0x3c, <http://www.phrack.org/show.php?p=60&a=6>.
- [3] „Holy\_Father“ (pseudonym). *How to become unseen on Windows NT*, <http://rootkit.host.sk/knowhow/hidingen.txt>.
- [4] HÝSEK, J. Uvod patchovani systemovych volani v Linuxu (LKM), *Blackhole*, <http://www.blackhole.sk/readme.php?id=166>.
- [5] JONES, A. R. A Review of Loadable Kernel Modules, *SANS Security Essentials*.
- [6] JONES, K. Loadable kernel modules, ;login:, roč. 26, č. 7, 2001, <http://www.usenix.org/publications/login/2001-11/pdfs/jones2.pdf>.
- [7] KEONG T. CH. *Win2K Kernel Hidden Process/Module Checker*, <http://www.security.org.sg/code/kproccheck.html>.
- [8] „OpioN“ (pseudonym), Kernel Rootkits Explained, *IT observer*, 26. březen 2003, <http://www.ebcvg.com/articles.php?id=124>.
- [9] PETRONI N. L. ET AL. Copilot – a Coprocessor-based Kernel Runtime Integrity Monitor, In *Proceeding from Usenix Security Symposium*, srpen 2004.
- [10] „plaguez“ (pseudonym), Weakening the Linux Kornel, *Phrack*, roč. 8, č. 52, 1998.
- [11] „sd“ (pseudonym), patchovani linuxoveho /dev/kmem, prielom #17, 27.2.2002, <http://hysteria.sk/prielom/>.
- [12] „sd, devik“ (pseudonymy), Linux on-the-fly kernel patching without LKM, *Phrack*, roč. 0xb, č. 0x3a, prosinec 2001.
- [13] SPARKS, S., HITLER, J. Raising The Bar For Windows Rootkit Detection, *Phrack*, roč. 0xb, č. 0x3d.
- [14] WANG Y. et al. Detecting Stealth Software with Strider GhostBuster, *Microsoft Research Technical Report*, MSR-TR-2005-25.
- [15] WANG Y. et al. Strider GhostBuster: Why It's A Bad Idea For Stealth Software To Hide Files, *Microsoft Research Technical Report*, MSR-TR-2004-71.
- [16] ZHOU J., QIAO L. *Backdoor and Linux LKM Rootkit - smashing the kernel at your own risk*, <http://eva.fit.vutbr.cz/~xhysek02/syscalls/020129lkm.htm>.
- [17] ŽILKA, R. *Analýza a detekce linuxových rootkitů*, bakalářská práce FI MU, 2005.
- [18] <http://www.chkrootkit.org/>.
- [19] <http://www.fspro.net/downloads.html>.
- [20] <http://www.rootkit.com/project.php?id=9>.
- [21] <http://rootkit.host.sk/>.
- [22] <http://www.nethunter.cc/index.php?id=14>.
- [23] <http://sourceforge.net/projects/stjude>.
- [24] <http://www.microsoft.com/licensing/programs/sa/support/winpe.mspx>.



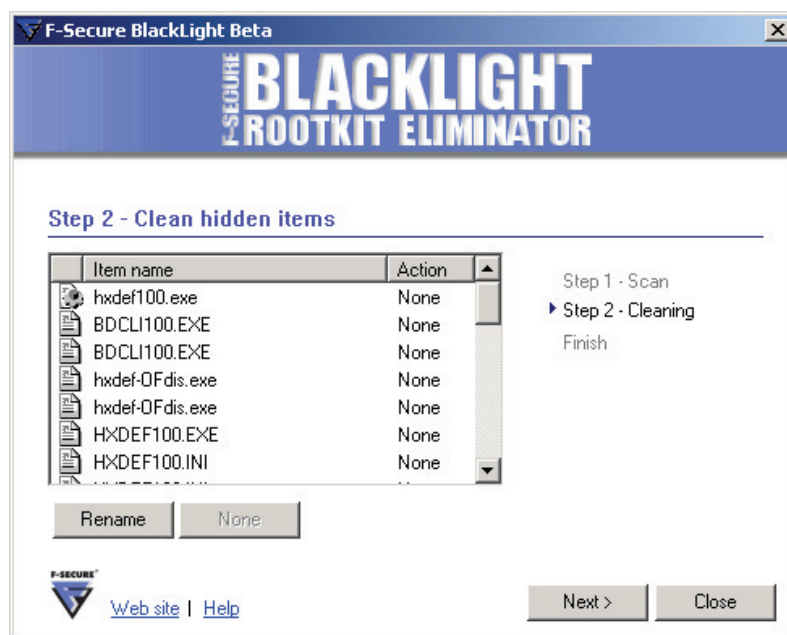
- [25] [http://www.rootkit.nl/projects/rootkit\\_hunter.html](http://www.rootkit.nl/projects/rootkit_hunter.html).
- [26] <http://www.s0ftpj.org/en/site.html>.
- [27] <http://www.stud.fit.vutbr.cz/~xhysek02/>.
- [28] <http://www.s0ftpj.org/tools/tools.html>.
- [29] <http://www3.ca.com/securityadvisor/virusinfo/virus.aspx?id=39113>.
- [30] <http://www.f-secure.com/blacklight/>.
- [31] <http://www.rootkit.com/project.php?id=15>.
- [32] <http://www.rootkit.com/project.php?id=20>.
- [33] <http://www.sysinternals.com/utilities/rootkitrevealer.html>.
- [34] <http://en.wikipedia.org/wiki/Rootkit>.

**Ing. Mgr. Zdeněk Říha, Ph.D.**

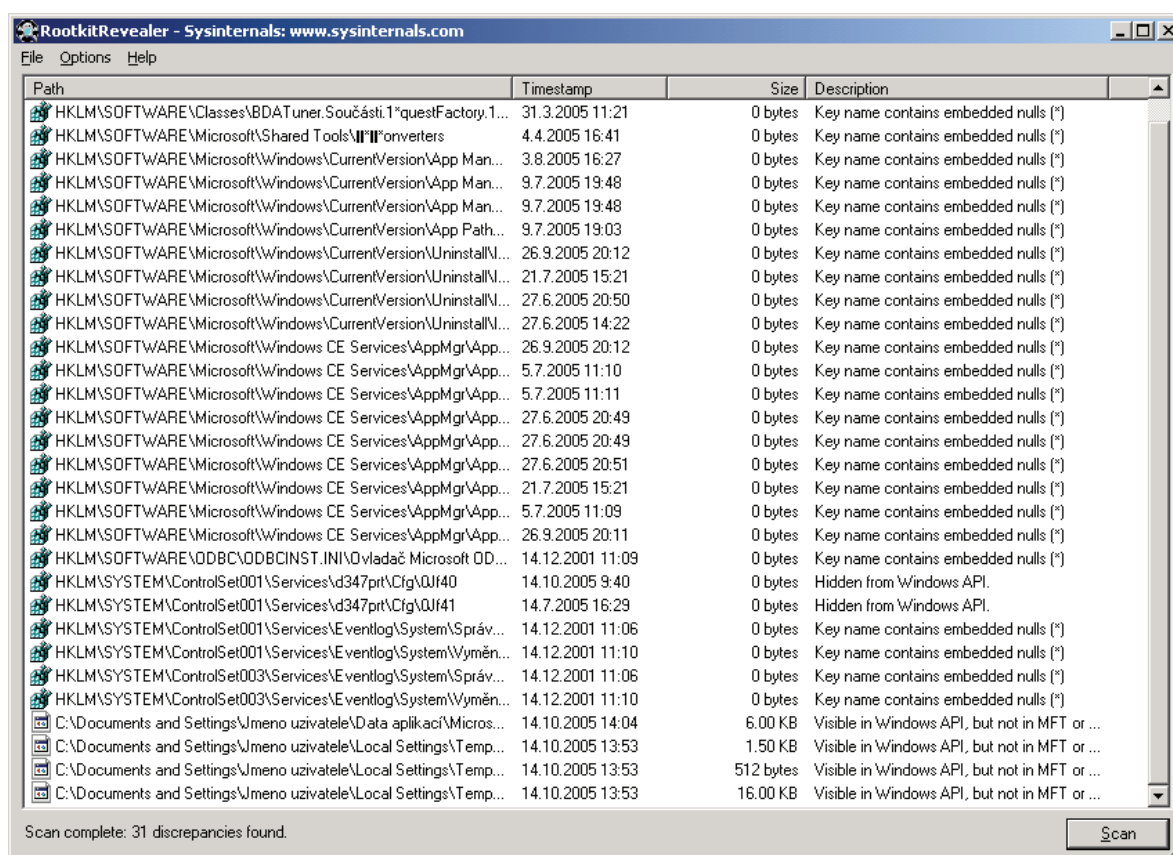
Odborný asistent na Fakultě Informatiky Masarykovy v Universitě v Brně. V současné době je na dlouhodobé stáži ve Společném výzkumném centru Evropské komise v italské Ispře.

**Management Summary**

Rootkity jsou mocnou zbraní hackerů při maskování jejich přístupu na napadený počítač. Při detekci stále rafinovanějších rootkitů je třeba jít stále blíže jádru operačního systému. Článek ukazuje principy rootkitů i možnosti jejich detekce.



Obr. 1: Rootkit Hacker Defender zachycený detektorem F-secure Blacklight.



Obr. 2: Ukázka výstupu programu RootkitRevealer, který hlásí hromadu falešných poplachů.