VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

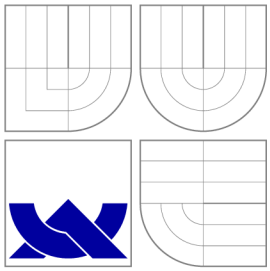
ANALÝZA SÍŤOVÝCH ÚTOKŮ POMOCÍ NÁSTROJE HONEYD

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

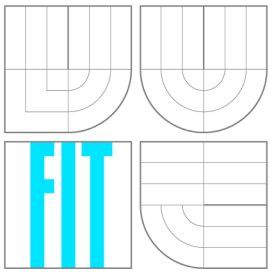
AUTOR PRÁCE
AUTHOR

Bc. JAN KOHOUTEK

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ANALÝZA SÍŤOVÝCH ÚTOKŮ POMOCÍ NÁSTROJE HONEYD

NETWORK ATTACK ANALYSIS USING HONEYD TOOL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JAN KOHOUTEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL DROZD
SUPERVISOR

BRNO 2009

Abstrakt

Analýza síťových útoků pomocí nástroje Honeyd. Testování nasazení opensource honeypotů WinHoneyd a LaBrea. Popis a řešení problémů v souvislosti na použitý operační systém. Zachycení síťového útoku pomocí nástrojů pro zachytávání paketů. Analýza a zpracování zachycených dat.

Abstract

Network attack analysis using honeyd tool. Opensource honeypots WinHoneyd and LaBrea deployment testing. Description and solving problems conected with operating system. Capture of network attack with packet sniffer. Captured data analyzing and procesing

Klíčová slova

Analýza, Síťové útoky, Honeypot, Honeyd, WinHoneyd, LaBrea, Metriky

Keywords

Analysis, Network attack, Honeypot, Honeyd, WinHoneyD, LaBrea, Metrics

Citace

Jan Kohoutek: Analýza síťových útoků pomocí nástroje Honeyd, diplomová práce, Brno, FIT VUT v Brně, 2009

Analýza síťových útoků pomocí nástroje Honeyd

Prohlášení

Prohlašuji, že jsem tento diplomový projekt vypracoval samostatně pod vedením pana Ing.Michala Drozda

.....
Jan Kohoutek
24. května 2010

Obsah

Kapitola 1

Úvod

Tento dokument vznikl jako textová část diplomové práce, na Fakultě informačních technologií, Vysokého učení technického v Brně.

První kapitola, která následuje za tímto úvodem, je nazvaná Uvedení do problematiky. Je zaměřena na nastínění základních skutečností, které s touto prací souvisí. To je především zamyšlení se nad současným stavem, za kterým následuje popis možných řešení ke zlepšení této situace. Poslední částí úvodní kapitoly je pak motivace a shrnutí všech hlavních cílů celého projektu.

Následuje kapitola Typy síťových útoků. Jejím účelem je popis jednotlivých síťových útoků, kterých se práce týká. Z široké škály současných hrozeb jsou vybrány převážně ty, na které by měla tato práce zaměřit, případně které s ní nějakým způsobem souvisí.

V kapitole nazvané Systémy ochrany proti síťovému útoku, jsou obecně charakterizovány dva hlavní přístupy současných systémů pro síťovou bezpečnost. Dále jsou pak nastíněny přístupy k detekování útoků.

Na tuto kapitolu navazuje kapitola následující - Systémy klamných cílů. Jsou v ní popsány a vysvětleny základní pojmy spojené s těmito systémy, je uvedeno a popsáno jejich základní členění, jejich výhody a nevýhody. Zároveň jsou zmíněna i možná rizika, které jejich použití může přinést.

Další kapitola Použité honeypoty, je již orientována spíše prakticky. Jsou v ní shrnuty a popsány zkušenosti a znalosti získané při testování honeypotů. Stejně tak jsou popsány problémy, které s provozem souvisí, aby bylo možné tyto poznatky využít při případném dalším výzkumu těchto zařízení. Dle upřesnění zadání byly testovány opensource honeypoty určené pro systémy Windows.

Při praktické práci s honeypoty byly zjištěny některé obecné vlastnosti, které se této problematiky týkají, či ji přímo ovlivňují. Ty jsou pak podrobněji rozepsány v kapitole Opensource honeypoty v prostředí Windows.

Prakticky zaměřená je i kapitola Analýza síťového útoku. Jejím cílem je zachycení, rozbor a zpracování dat získaných ze síťového útoku. Získaná data se analyzují definovanými metrikami a tyto výsledky jsou pak ukládány do vytvořené datové struktury.

V kapitole Další použité technologie jsou popsány technologie, které se během průběhu práce používaly, nicméně jejich důležitost nebyla tak vysoká jako u technologií a programů popsaných přímo v předcházejících kapitolách.

Poslední kapitola je Závěr, kde je celá práce shrnuta a zhodnocena, jsou popsány její přínosy a možné pokračování.

Z hlediska návaznosti na semestrální projekt byly použity některé jeho části. V semestrálním projektu se jedná o kapitolu Uvedení do problematiky, která byla převzata, doplněna

a v diplomové práci figuruje pod stejným názvem. Dále pak kapitola Typy síťových útoků, která byla převzata a uvedena opět pod stejným názvem. Kapitola Co je to honeypot byla převzata, doplněna a použita jako součást kapitoly 5 (Systémy klamných cílů) diplomové práce. Z poslední kapitoly semestrálního projektu nazvané Použité technologie, bylo převzata část pojednávající o Honeyd a použita v kapitole 5 (Systémy klamných cílů). Zbytek této kapitoly se pak použil v kapitole Další použité technologie diplomové práce.

Kapitola 2

Uvedení do problematiky

V této kapitole jsou zmíněny a rozebrány základní pojmy a motivace, nezbytné pro správné pochopení celého tématu. Dle pokynů pro psaní diplomové práce se jedná se o současný stav řešené problematiky, nastínění možných řešení a stanovení cílů, které bude práce sledovat.

2.1 Současný stav

S rozvojem Internetu jako celosvětově rozšířeného komunikačního média jdou ruku v ruce také negativní stránky tohoto fenoménu. Jedná se především o bezpečnost, či spíše nebezpečnost. Se zapojením se do celosvětové sítě, se každému uživateli naskýtá nejen možnost přístupu k informacím a komunikaci, ale zároveň se i vystavuje riziku možného útoku.

K tomu, aby vůbec došlo k napadení systému, je nutné splnění několika faktorů. Prvním z nich je vlastní existence nějaké hrozby, v tomto případě útočníka. Při dnešním rozšíření internetu a frekvencí jeho užívání více než miliardou uživatelů, je existence uživatele, který nemá zcela čisté úmysly, či přímo cíleně chce někoho napadnout, v podstatě jistá. U každého útočníka je rozdílná především úroveň znalostí. To přímo odpovídá jejich nebezpečnosti, útočník s vysokými znalostmi je vždy nebezpečnější než útočník s nízkými znalostmi. Důležitá je rovněž motivace. V případě finančního ohodnocení lze očekávat mnohem větší snahu o napadení systému, než při pouhé osobní motivaci - touze či potřebě prezentovat či dokazovat si svoje schopnosti. Zároveň bude mít finančně motivovaný útočník pravděpodobně vyšší cíle, než je například podomácku provozovaný server, na který se naopak zaměří útočník bez finanční motivace.

Druhým nezbytným faktorem je pak přítomnost slabého místa v systému. Pokud se útočníkovi podaří takové místo najít, je pak jen otázkou času, kdy ho zneužije k proniknutí do systému. Existence takových míst je zase dána problematikou vývoje softwaru. Jedná se především o chyby při návrhu systému nebo aplikace, kdy se například vybere slabý nebo jinak nevhodný způsob zabezpečení, při implementaci, kde je běžným případem chyba typu buffer-overflow nebo chyby při používání systému, kam spadá například heslo uložené na nezabezpečeném místě. Určující je také cena. Často se stává, že se o slabém místě ví, ale jeho oprava by byla příliš nákladná, případně kvůli vysokým pořizovacím a provozním nákladům není použito odpovídající zabezpečení. Tyto příčiny jsou dnes i přes veškeré snahy o kvalitní zabezpečení velmi časté, v mnoha případech dochází i k jejich kombinacím. To vše pak nahrává potenciálním útočníkům.

2.2 Možná řešení

Při hledání řešení těchto problémů je nutné zaměřit se na všechny aspekty, které je vyvolávají. Z hlediska existence hrozby je asi nereálné počítat s naprostou eliminací počítačové kriminality. Preventivní faktor nebude mít zřejmě v této oblasti takový vliv. Pachatelé elektronické kriminality si jsou u svého počínání velmi dobře vědomi jeho nebezpečnosti a škodlivosti. Ovliňujícím je však represivní faktor, který je ale závislý na legislativě, která mnohdy tyto oblasti buď nepokrývá vůbec nebo pouze okrajově a nejednoznačně. Její další zpřesnění a především rozšíření, které by přineslo přísnější a jasnější tresty by mohlo mít pozitivní vliv na množství potenciálních útočníků - jejich určité množství by vyšší tresty nejspíše odradily. Nezbytné je zaměřit se také na subjekty, které z těchto činností těží a poskytují útočníkům již zmiňovanou finanční motivaci. Například rozesílání spamu přestane být výhodné, pokud se za něj přestane platit.

Z hlediska zlepšení situace v existenci slabých míst se však řešení mohou jevit ještě těžší. Slabá místa způsobená nevhodným návrhem nebo chybnou implementací lze vyřešit zkvalitněním práce, dodržováním standardů a důslednějším testováním. Problémy zjištěné zpětně, již po vydání systému či aplikace, se jinak než vydáním záplaty zřejmě ani řešit nedají. Zde však vyvstává zřejmě klíčový problém. Existence záplaty, aktualizace či zveřejnění problému v žádném případě neznamená jeho vyřešení. K tomu je nutné, aby tato oprava byla uvedena i do praxe, tedy aby tuto opravu začali používat i samotní uživatelé, což však rozhodně není pravidlem a častěji je tomu spíše naopak. Problémy způsobené chybou při užívání, tedy vytvořením slabého místa samotným uživatelem, se odstraňují daleko hůře a je otázkou zda vůbec nějaké smysluplné řešení existuje. Pokud se i při existenci volně dostupných antivirových programů a firewallů opakují případy infikování nezabezpečeného systému, je problém zcela jednoznačně buď v nedostatečných znalostech uživatele nebo jeho naprostém nezájmu o tuto problematiku. Často se takové zabezpečení provádí až když je již pozdě, klasicky až po ztrátě důležitých dat, což pro uživatele znamená spouštěcí moment, kdy se o bezpečnost začíná zajímat. Svým způsobem je alespoň tento pokrok pozitivní, nicméně povědomí o nutnosti zabezpečení systému by mělo být u dnešních uživatelů rozhodně vyšší. Oproti snižování počtu hrozeb, kdy se jako účinné jeví aplikace represivních opatření, u slabých míst by se mělo jednat spíše o prevenci.

2.3 Motivace k projektu

Jednoznačnou podmínkou k nápravě situace slabých míst je mechanismus jejich oprav. Ten vyžaduje kromě dostatečné flexibility a schopnosti rychle reagovat, především informace o existenci takového slabého místa, či spíše způsobu jeho zneužití. Vhodnou technikou pro získání těchto informací je pak analýza útoků, ze které zjistíme, na která slabá místa a jakým způsobem se útočí a podle těchto informací pak vytvoříme adekvátní doplněk zabezpečení. Složení útoků, jejich způsoby a techniky se mění v závislosti na možnostech slabých míst. Proto je důležité takovéto analýzy provádět pokud možno průběžně.

2.4 Cíl práce

Cílem této práce je zanalyzování aktuálního stavu síťových útoků. První část práce spočívá v seznámení se s prostředím, které se pro tuto činnost používá, pochopením jeho principů a postupů. V další části je pak nezbytné sestavit dostatečné množství různých testů, ze

kterých se získá, pokud možno co nejširší, škála dat. Tyto data bude nutné uložit do vhodné struktury, ideálně databáze nebo datového skladu. Poslední část pak spočívá v analýze a prozkoumání získaných dat a prezentování výsledků.

Část diplomové práce vyčleněná jako semestrální projekt, by se tak měla věnovat právě seznámení se s prostředím a analýzou dané problematiky. To zahrnuje jednak rozbor různých druhů síťových útoků a dále pak popis nástrojů pro jejich detekování a analýzu.

Hlavní část práce by pak měla být zaměřená na využití nástrojů typu honeypot na systémech Windows. Jde především o použitelnost opensource implementací, které by bylo možné začlenit do většího systému automatizované obrany proti útokům. Z hlediska využitelnosti znalostí pro takovýto systém, se však škála dat získatelná z dostupných honeypotů zdála býti relativně malá. Proto další část práce spočívala ve využití programů odposlouchávající datový tok na síťovém rozhraní, jejichž data bylo nutné zpracovat. Toto zpracování spočívá ve vytvoření vhodných metrik pro zpracování získaných dat. Z těchto výsledků by pak v optimálním případě mělo být možné detekovat případné narušení.

Kapitola 3

Typy síťových útoků

Tato kapitola je zaměřena na hlavní typy síťových útoků, které se v dnešní době vyskytují či se dříve vyskytovaly.

3.1 Přímé útoky

V této oblasti jsou popsány techniky používané při útoku od konkrétního útočníka na konkrétní systém. Ten obvykle probíhá v několika fázích - v první útočník provádí zjišťuje informace o cíli útoku v rámci sítě. Jedná se například o odposlech síťového provozu, nebo dotazy Whois, kterými mapuje strukturu sítě, případně se může pokusit i o takzvané sociální útoky. V tomto případě se snaží pomocí vhodně zvolené manipulace nebo předstírání získat informace od samotných uživatelů. Druhou fází je scanování jednotlivých nalezených cílů. Jeho účelem je nalezení systémů a služeb, které na nich běží. K tomu často slouží scanovací programy, klasicky například port scanner, který hledá v systému otevřené porty, ke kterým je možné se připojit. Stejně tak se může zaměřit na protokoly nebo služby, případně se pokusit odposlouchávat přímo jednotlivé pakety. V další fázi se pak snaží o získání přístupu do nalezeného systému. K tomu často slouží veřejné informace o záplatách slabých míst konkrétních programů, které řeší známé bezpečnostní díry. Velmi často se stává, že aktuální záplaty nejsou ještě aplikovány, používají se starší neošetřené verze z důvodu kompatibility atd. Tyto informace je pak pro zkušenějšího útočníka snadné využít, pro méně zkušeného je pak alternativou použití již ověřených postupů. Často využívané jsou metody buffer overflow či útok na heslo. Pokud se útočníkovi podaří do systému získat přístup, je pravděpodobné, že se pokusí vytvořit v něm možnost opětovného přístupu. K tomu slouží takzvaná zadní vrátka. Jejich vytvoření může spočívat v instalaci určitého programu, například Trojského koně a podobně. Poslední činností pak bývá zahlazení stop, které svojí činností v systému vytvořil. [1]

3.2 Malware

Do češtiny se tento výraz běžně překládá jako „škodlivý software“, což tuto kategorii poměrně přesně popisuje. Patří sem typy útoků, které již mají značnou míru automatizace. Zjednodušeně řečeno „na druhé straně nemusí nikdo sedět“. Klasickým a nejstarším představitelem je počítačový vir, který se často nepřesně používá k označení celé této kategorie.

Jeho hlavním znakem je jednak provádění zmíněné škodlivé činnosti, ale především schopnost šíření sebe sama. Vir potřebuje hostitelský program který napadne, ze kterého

bude vykonávat svoji činnost a dále se šířit, jakmile se hostitelský program spustí. Z těchto podmínek tak vyplývají požadavky na jeho šíření. Pro to je potřeba co největší množství počítačů se stejným operačním systémem, které mají možnost si mezi sebou vyměňovat spustitelné soubory, tedy prostředí s vysokou konektivitou. [3] Tuto roli velmi vhodně splňuje prostředí Internetu. V době před jeho nástupem bylo téměř jediným způsobem šíření výměna dat na přenosných médiích. Již několikrát zmíněná škodlivá činnost je pro šíření viru rovněž klíčová, resp. její vhodné vybalancování. Pokud vir vykonává určitou destruktivní činnost, která je jasně patrná, bude pravděpodobně velmi brzy odhalen. Proto se autoři virů zaměřují především na způsob replikace a šíření, který musí být co nejefektivnější. Často se také škodlivá činnost „zpožďuje“, aby měl vir dost času nejprve se rozšířit. V současné době se již od klasických virů ustupuje, omezující je zejména nutnost existence hostitelského programu, který se musí spustit.

Mnohem větší potenciál má proto dnes spíše červ, který je na rozdíl od viru samostatný program, který hostitelský program nepotřebuje. Šíření v tomto případě probíhá ze systému na systém a stejně jako v případě viru je proto nezbytná vysoká konektivita. [3] Právě z toho důvodu se tento typ malwaru objevuje až s rozšířením internetu. Stejně tak se ovšem vyskytují červi, kteří jsou schopni se velmi efektivně šířit přes přenosná média - dnes velmi populární a rozšířené flash disky. Často je také infikace červem pouze první fází viz dále problematika botnetů.

Ostatní znaky jsou podobné jako u viru, červ napadne systém, na kterém provádí svoji škodlivou činnost, a zároveň se snaží rozšířit na co nejvíce dalších systémů v okolí. Právě rychlost šíření je u červů velmi vysoká, často se jedná o stovky tisíc infekcí během prvního měsíce. „Nejúspěšnější“ červi ale dosahují i mnohem vyšších čísel, jako například červ ILOVEYOU z roku 2000, u kterého bylo už za první týden ohlášeno 50 milionů infekcí. [2] Takové množství už je ale opravdu výjimečné - viz například statistika z listopadu 2009, ve které můžeme rovněž vidět nejčastější malware - červy, viry a Trojské koně. [7]

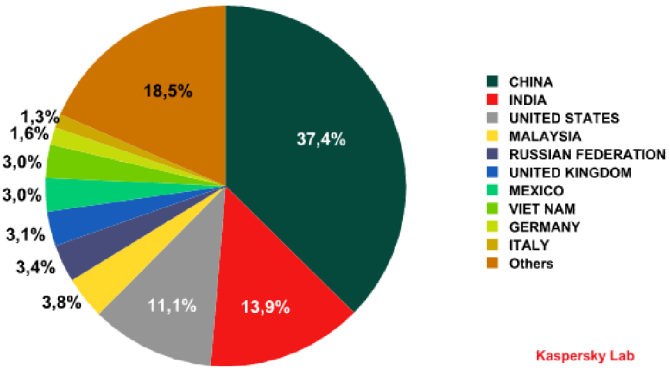
Position	Change in position	Name	Number of infected computers
1	0	Net-Worm.Win32.Kido.ir	330305
2	New	Net-Worm.Win32.Kido.iq	174351
3	-1	Net-Worm.Win32.Kido.ih	145332
4	0	Virus.Win32.Sality.aa	128737
5	0	Worm.Win32.FlyStudio.cu	93848
6	-3	not-a-virus:AdWare.Win32.Boran.z	84825
7	-1	Trojan-Downloader.Win32.VB.eql	63287
8	9	Trojan-Downloader.WMA.GetCodec.s	48426
9	1	Virus.Win32.Virut.ce	47812
10	-3	Virus.Win32.Induc.a	46252

Obrázek 3.1: Přehled nejčastějších ifekcí malwarem za listopad 2009 [7]

Dalším zástupcem této kategorie je právě Trojský kůň. Svůj název dostal právě podle dřevěného koně z Trojské války. Cílem tohoto programu je provádění určité skryté činnosti v napadeném systému, obvykle to bývá například hledání či odposlouchávání hesel nebo vytváření takzvaných zadních vrátek - potenciálních slabých míst, které mohou být při

útoku na daný systém využity. Na rozdíl od virů a červů se však sám nerozšiřuje.

Na závěr části o malwaru je přiložen graf zachycující rozložení jeho původu, podle země, ze které přišel. Dlouhodobě se na prvním místě nachází Čína, dále pak Spojené státy a Indie, které se mění na dalších místech podle různých období.



Obrázek 3.2: Malware podle původu za listopad 2009 [7]

3.3 SPAM

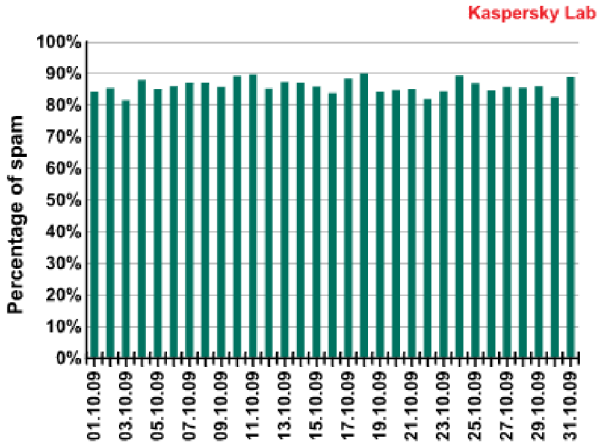
Spam, neboli nevyžádaná pošta, jde téměř ruku v ruce s rozvojem internetu a se kterou se téměř jistě setkal snad každý uživatel internetu. Přesto, že se může na první pohled tento problém zdát jako relativně málo nebezpečný, jedná se o věc poměrně závažnou. Podle mnohých analýz až 85% veškeré elektronické pošty představuje právě spam. [8] I přesto, že emailů obsahující nějaký nebezpečný malware je „pouze“ kolem 1%, je naprosto alarmující, že více než 8 z 10ti emailů je pouze zbytečná zátěž serverů, linek a času uživatelů. Zachycení a rozpoznání spamu je často obtížné. Používají se blacklisty tj. databáze ip adres ze kterých přichází spam, stejně tak se používají filtry, které hledají spamové řetězce přímo uvnitř emailů. Žádná metoda však nemá stoprocentní účinek - pro rozesílání spamu se využívají především počítače či servery napadené malwarem, který se nějakým způsobem šíří. Stejně tak se rozšiřuje i množství potenciálních ip adres ze kterých spam může přicházet. Metoda filtrování podle obsahu má zase tu nevýhodu, že filtry musí být nastaveny dostatečně citlivě, aby odhalily spam, ale zároveň aby nenarušovaly běžnou komunikaci.

Nejčastější spamy se z hlediska obsahu téměř pravidelně opakují. Nabídky na výhodné dárky a výhry, reklamy propagující medikamenty, nabídky obchodů, loterie a mnohé další. Tematicky jsou spamy často zaměřené na období kolem svátků (vánoce, halloween) případně sledují některé aktuální trendy (léky proti chřipkové epidemii a podobně).

Na obrázku je pak statistika spamu za říjen 2009, množství nevyžádané pošty osciluje mezi 80ti až 90ti procenty z celkového množství veškeré odeslané elektronické pošty.

3.4 Útoky s účelem zisku

Výše zmíněný Trojský kůň už postupně přechází z oblasti, kdy škodlivost malwaru spočívá v určité záškodnické činnosti, do činnosti prováděné přímo za účelem zisku. Do této oblasti



Obrázek 3.3: Podíl spamu na celkovém množství elektronické pošty za říjen 2009 [7]

spadá například spyware, což je souhrnné označení pro malware, jehož činnost spočívá v monitorování uživatelů. Sleduje se například které internetové stránky uživatel navštěvuje, o jaké oblasti se zajímá a podle toho se snaží zobrazovat specifické reklamy, případně ovlivňovat výsledky vyhledávačů. Přesto, že se nejedná o vyloženě destruktivní činnost, má často za následek zpomalení procesů, padání prohlížeče a podobně.

Mnohem nebezpečnější jsou pak takzvané botnety, tedy sítě botů - napadených počítačů, které je možné určitým způsobem dálkově ovládat. To je často způsobeno infekcí určitého druhu červa nebo viru, ten pak vytvoří vhodné prostředí - obvykle se maskuje jako některá ze služeb, poslouchá a čeká, až přijde jeho čas. Největší nebezpečnost totiž spočívá ne v jednom takto ovládaném systému, ale v celé síti, tvořené často tisícem nebo statisícem botů. Asi nejběžnější využití botnetu je pak pro rozesílání spamu, z čehož plynou pro vlastníka botnetu velmi tučné zisky. Další možností je DDoS útok, tedy distribuovaný útok typu DoS, který je veden najednou z velkého množství počítačů, čímž se rapidně zvyšují jeho šance na úspěch. V obou případech se však jedná o jednorázovou záležitost, po úspěšném rozeslání spamu nebo útoku bývá botnet odhalen a zničen.

Dnes už dosti archaické jsou pak například dialery, které se hojně vyskytovaly v souvislosti s přístupem na Internet pomocí vytáčeného připojení. Jejich princip byl takový, že namísto používaného připojení vytáčely předem daná vysoce zpoplatněná čísla. Při dnešním rozšíření kabelového internetu se však tato technika vytratila.

Některé techniky je pak poměrně obtížné zařadit, jako například phising, který je založený spíše na neznalosti a nepozornosti uživatelů. Jeho hlavní rozvoj nastal především s rozvojem internetového bankovnictví. Princip je poměrně jednoduchý. Na předem vytipované adresy dorazí podvržený email, odkazující například na podvržený web bankovní společnosti nebo služby, kde je uživatel vyzván k zadání vstupních údajů o jeho účtu. Při šikovně podvrženém systému a dostatečně velkém množství potencionálních obětí, kterým je takový email rozeslán, má pak útočník šanci na získání přístupu k cizímu kontu. Protože jsou použity skutečné přístupové údaje, které běžně používala oběť, je tento útok prakticky neodhalitelný a nejlepší obranou proti němu, je obezřetné nakládání s přihlašovacími údaji ze strany uživatele.

Kapitola 4

Systémy ochrany proti síťovému útoku

Obecně pak můžeme shrnout tři hlavní cíle, které musí ochrana systému splňovat. Prvním je důvěrnost (confidentiality), která představuje zaručení, že se k datům, zdrojům nebo službám dostane pouze uživatel, který k tomu má oprávnění. Spadá sem především ochrana před neoprávněným přístupem k souborům, ochrana uživatelských účtů a hesel a podobně. Druhým cílem je integrita (integrity). Ta představuje ochranu proti neoprávněné modifikaci, kterou může způsobit například škodlivý software. Posledním bodem je pak dostupnost (availability), která zajišťuje ochranu proti neoprávněnému odepření přístupu. Klasickým případem je obrana před DoS útokem, kdy dochází k záměrnému přetížení, s cílem způsobit nedostupnost dat. [3]

4.1 NIDS a NIPS

Různá bezpečnostní zařízení lze rozčlenit do dvou hlavních skupin. Jedná se o pasivní a aktivní systémy. První skupina se označuje pojmem NIDS (Network Intrusion Detection System). Jak vyplývá z názvu, tato zařízení pouze sledují a monitorují systém a aktivity uživatelů, které na něm probíhají. Pokud zjistí některé znaky neoprávněné aktivity, provádí pouze upozornění, že k něčemu takovému došlo. Další opatření už je tedy na operátorovi takového systému. Toto řešení je proto použitelné spíše ve větších organizacích, které mají dostatek prostředků. Dohled nad takovýmto systémem má často celý tým, který se o zabezpečení stará. Svým způsobem se to může zdát jako značná nevýhoda, protože není možné, aby u podobného zařízení byl neustále někdo přítomen. Do jisté míry je však ale součinnost lidského faktoru v této oblasti téměř nezbytná, jak uvidíme dále.

Druhou skupinou jsou NIPS (Network Intrusion Prevention System), které už na rozdíl od první skupiny, poskytují kromě detekce i obranná opatření. Aktivní mechanismus spočívá ve třech fázích. První představuje samotná detekce útoku, která je podobná jako v případě NIDS. Dalším krokem je ovšem aktivní reakce proti zjištěnému útoku a přerušení jeho provádění. To může být například resetování spojení, zahození obdržených paketů a podobně. Třetím krokem je pak snaha o zamezení zopakování podobné události. Toho se docílí například přidáním pravidla pro firewall, který bude takovouto aktivitu blokovat. Tento přístup má na první pohled poměrně značnou výhodu, všechny tyto činnosti mohou probíhat v podstatě automatizovaně. V tom se však skrývá i jeho hlavní nevýhoda, může se velmi lehce stát, že dojde k takzvanému falešnému pozitivnímu poplachu. Tak se ozna-

čuje situace, kdy obranný systém spustí alarm při běžné aktivitě. Často se tak nevyhneme nápravě, kterou už zautomatizovat nelze.

Obecně se může zdát, že detekce a prevence jsou v tomto případě téměř totožné technologie. Ve skutečnosti se však značně odlišují jak v samotném návrhu tak i funkci. Z pohled jednotlivých paketů se v případě prevence (tedy NIPS) při nalezení podezřelého paketu do systému jednoduše nevpustí. Je jednoduše zahozen. V případě detekce (NIDS) je takovýto paket analyzován a na základě výsledku se vytvoří příslušná odezva.

Dále bychom mohli aplikovat další dělení, například podle druhu umístění v síti, kde můžeme vyčlenit detekci pouze na jednom počítači/stanici, případně detekci pomocí senzorů na vstupních bodech sítě. Podobných členění bychom zřejmě našli ještě více, proto jsou zde uvedeny pouze tyto základní typy a jejich principy.

Mezi ochrannými systémy nelze jednoznačně určit, který ze způsobů je lepší. Tak jako v mnoha případech má každý své výhody a nevýhody a je tedy nutné zvážit i ostatní okolnosti a podmínky nasazení. Žádný z těchto systémů není stoprocentně spolehlivý, v případě chybného detekování nežádoucí aktivity se generuje nežádoucí falešný poplach. Případně může nastat i horší alternativa - systém nezareaguje na skutečný útok. Hlavní těžiště celé této problematiky tedy spočívá především ve správné detekci útoku. Právě různým typům a technikám detekce se věnuje následující podkapitola.

4.2 Detekce útoku

Jednotlivé útoky často nesou množství podobných znaků, jsou vedeny na stejné slabé místo systému, stejnou metodou, stejnou skupinou útočníků atd. Pro efektivní zabezpečení je proto velmi důležité tyto znaky rozpoznat, zanalyzovat a na základě takovýchto znalostí pak zefektivnit současné zabezpečení nebo vytvořit zabezpečení nové. Různých druhů detekce existuje celá řada, proto jsou zde opět uvedeny ty nejzákladnější.

Velmi často používaný přístup je detekce známých vzorů (signature-based detection). Tato metoda spočívá ve hledání předem známých vzorů v datovém toku, nebo aktivit na síti či počítači. Jakmile je takový vzorek detekován, systém spustí alarm. Principiálně se jedná o poměrně jednoduchý způsob, jeho nevýhodou však je, že je potřeba již předem znát způsoby útoku a ty pak detekovat. To je velmi užitečné, například pro odhalení DoS útoku (denial of service - znepřístupnění služby, například zahlcením serveru). Problém ale může nastat, pokud útočník rozloží svoje aktivity do delšího časového období, aby nemohly být jednoduše rozpoznány. Stejně tak, pokud dojde ke špatnému rozpoznání, nebo pokud se použije nový způsob útoku, který tento systém nezná. S tímto přístupem detekce proto souvisí i problematika aktuálnosti hledaných vzorů. Ty musí být průběžně aktualizovány, aby bylo možné detekovat i nové útoky. Je tedy nezbytné řešit i otázku, kde tyto nové postupy objevit. Jedno z možných řešení je připravit již předem na útočníka past, na které zjistíme, jaké techniky útočník použije. Z této pasti pak získáme potřebné znalosti o novém nebo i aktuálním způsobu útoku, které pak můžeme vhodně použít k vylepšení zabezpečení, doplnění či zaktualizování hledaných vzorů a podobně.

Druhým častým řešením je statistická analýza, která spočívá v hledání a detekci anomálií, které se vyskytují v síťovém provozu na určitém počítači/stanici dle momentálně aktivního uživatele. K tomu je nejprve potřeba vytvořit určitý „normální“ profil, tedy činnosti, které uživatel či skupina uživatelů běžně dělá, a hledat odchylky, které mohou značit přítomnost škodlivého softwaru či jiného napadení. Kvalita takového zabezpečení je přímo úměrná kvalitě vytvořeného profilu. Přesto však nemůže být stoprocentní. Uživatel se může

omylem od takového profilu odchýlit a nevhodně nastavený systém pak může generovat falešný poplach.

Další technikou je například kontrola obsahu protokolů, kde se systém zaměřuje na odchylky příkazů, které se používají. Podle jejich definic v RFC je pak možné detekovat nesrovnalosti. Taková kontrola je však často na úkor rychlosti spojení, je nutné udržovat si určité stavové informace a problematická jsou taktéž šifrovaná spojení. Přesto se jedná o spolehlivý prostředek například proti DoS útoku, nebo dříve proti takzvanému Ping of death, kdy byl místo paketu o velikosti 56 bajtů podvržen paket o velikosti 65 535 bajtů.

Kapitola 5

Systémy klamných cílů

Tato kapitola pojednává o použití určitého typu pasti nastražené na útočníka, díky které lze odhalit techniku útoku, kterou se pokusil použít. V minulé kapitole byla právě tato metoda detekce zmíněna.

5.1 Systémy honeypot

Doslovný překlad slova honeypot by byl nejspíše „hrneček s medem“. Tento název již může mnohé evokovat a napovědět o funkci tohoto zařízení. Zjednodušeně se skutečně jedná o určitý druh návnady, na kterou chceme přilákat případné zvědavce a zloděje, a zjistit, jak budou při dobývání se k této návnadě postupovat. Honeypot tedy představuje počítačový systém, který chceme nechat scanovat, napadnout či jiným způsobem ohrozit. Všechny tyto aktivity se monitorují a zaznamenávají. Další možnou aplikací pak může být i například nastražení honeypotu s fiktivními ale na první pohled uvěřitelnými daty vedle skutečného systému. Útočník se pak může spokojit s nabouráním se do honeypotu a získáním dat z něj, přičemž skutečný systém nechá na pokoji. Je však otázkou, jestli je tento přístup správný, zda se zkušený útočník nechá takto odradit. Hlavním využitím a účelem honeypotů je tak zejména analýza.

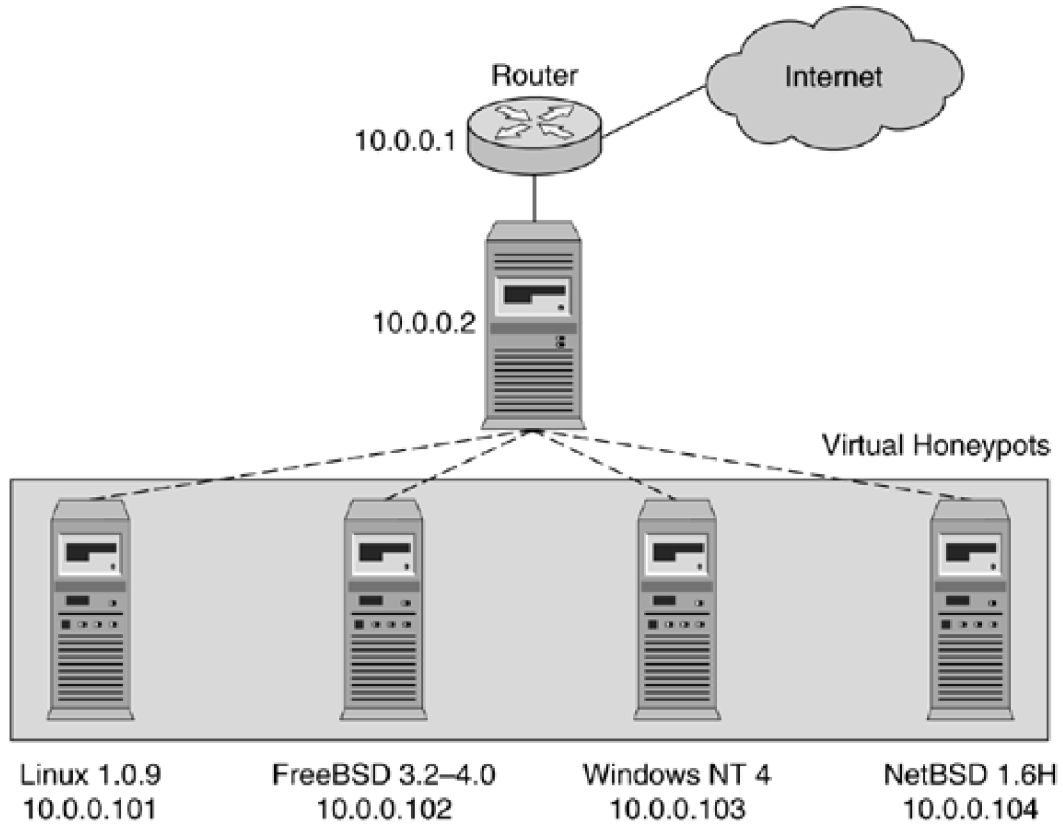
Vývoj a použití těchto nástrojů můžeme časově zařadit přibližně do roku 2000, tedy již poměrně starší záležitost, z hlediska rychlosti vývoje v oboru IT. Za uplynulých 10 let prošly systémy honeypot poměrně značným vývojem, vzniklo několik různých skupin, do kterých se honeypoty dělí. Prvním hlavním hlediskem je dělení na fyzické a virtuální, dalším je pak dělení na tzv. High-interaction a Low-interaction honeypoty.

5.2 Fyzické a logické honeypoty

Jak již bylo zmíněno v předchozím odstavci, honeypot je počítačový systém. Tento systém můžeme asi nejjednodušeji vytvořit jako samostatné PC, které „obětujeme“, necháme na ně provádět útoky a ty monitorujeme. Toto řešení popisuje fyzické honeypoty - jedná se o skutečné stroje připojené k Internetu, každý s vlastní IP adresou. [4] Často také platí, že fyzické honeypoty jsou zároveň honeypoty s vysokou interakcí (co je to High-interaction honeypot viz dále), nicméně to nemusí být pravidlem. Logické honeypoty jsou virtuální stroje, běžící na jiném fyzickém stroji, ze kterého simulují jednotlivé další systémy. Obvykle jich může být velké množství. [4] U logických honeypotů je častá návaznost na nízkou interakci (co je to Low-interaction honeypot viz dále), nicméně to opět nemusí být pravidlem.

Oba tyto přístupy mají své výhody a nevýhody. Fyzický honeypot působí věrohodně, jako opravdový systém, kterým ve svém principu také je. Jeho vytvoření a provoz jsou však problematičtější a náročnější, proto se jich většinou používá omezené množství, řádově několik kusů, případně desítek kusů. Při nevhodně nastaveném systému se však může stát, že získaná data budou podrobná a kvalitní, ale jejich množství bude malé, protože jejich nízký počet přiláká pouze nízký počet útočníků. V tom mají naopak výhodu logické honeypoty, kterých můžeme vytvořit na jednom fyzickém stroji i několik tisíc. Pro vytvoření rozsáhlejší sítě se proto používají především logické honeypoty, protože vytvoření sítě fyzických honeypotů v rozsahu několika stovek vlastních IP adres je příliš nákladná a složitá záležitost. Z hlediska „důvěryhodnosti“ je u logických honeypotů nezbytné zajistit, že případný útočník nezjistí, že se jedná pouze o virtuální stroj. V praxi se pak oba přístupy kombinují, používají se jak fyzické, tak logické honeypoty, čímž se využívají hlavní výhody obou přístupů.

Schéma virtuálního honeypotu, který simuluje několik systémů, je na následujícím obrázku.



Obrázek 5.1: Schéma virtuálního honeypotu
[4]

5.3 High-interaction a Low-interaction honeypoty

Neboli honeypoty s vysokou a nízkou interakcí. Rozdíl mezi nimi je právě v možnostech, či spíše rozsahu služeb, které poskytují. Nízká interakce v tomto případě znamená, že honeypot

poskytuje pouze určité služby. [4] To je do jisté míry omezující, protože zkušený útočník může toto omezení odhalit, nicméně naprosto dostačující pro útoky červů nebo předem připravenými skripty, které hledají předem vybrané slabé místo systému. Tímto způsobem je možné zjistit konkrétní cíle útoku, případně najít nový neznámý postup. Jednoznačnou výhodou je snadná instalace a údržba, stejně tak zaměření se na konkrétní druh útoku - nejčastěji se používají pro detekci spamu, červů, Trojských koňů atd.

Oproti tomu honeypoty s vysokou interakcí, poskytují celý systém. Klasicky to může být například počítač s routerem, který však sám o sobě nevyvíjí žádnou aktivitu. [4] Jakákoliv známka síťové činnosti, tj. například scan portů, či pokus o přístup k tomuto systému jsou pak automaticky brány jako neoprávněné a můžou se ihned zaznamenat. Výhodou takovýchto systému je právě jejich komplexnost, můžeme sledovat počínání útočníka v celém systému. Stejně tak je to i jeho nevýhodou, protože přes takovýto systém by se útočníkovi mohlo podařit napadnou i další systémy, které již honeypoty nejsou. Zároveň pak velké množství dat, které o daném útoku získáme vyžaduje delší analýzu, neboť je potřeba hledat souvislosti v globálnějším rozsahu.

Při výběru mezi nasazením high a low interaction honeypotů záleží především na zaměření celého projektu. Pokud je naším cílem zkoumat techniku útoku na počítač nebo počítačový systém jako celek, je vhodné či spíše nezbytné použít honeypoty s vysokou interakcí. Pro zaměření se na určitý druh útoků je zase vhodnější analyzovat útoky přímo na danou službu nebo určitou metodou.

5.4 Rizika použití honeypotu

Stejně jako každý systém, ani systém honeypot není zcela dokonalý. Kromě již zmíněné možnosti u fyzických a High-interaction honeypotů, kdy se útočníkovi může podařit napadnout i další systémy, které již jako honeypoty nefungují, je tu i jeden obecný aspekt. Aby honeypot správně fungoval, je nutné jej vystavit útokům, tj. nechat útočníka, aby se do systému zkoušel dostat. Právě to mu ale paradoxně může napomoci, protože honeypot použije jako odrazový můstek - vyzkouší si na něm techniky, které pak použije jinde, na skutečném systému, který má stejné slabé místo, jaké simulujeme na honeypotu.

Je otázka jak moc velkou míru viny za napadení takového systému nese zmíněný honeypot, či přímo člověk, který ho spravuje. Tato skutečnost je už závislá na právním systému konkrétní země, ve které se honeypot používá. V případě striktního právního systému by mohl být honeypot využitý při útoku na skutečný systém chápán jako napomáhání v trestné činnosti či dokonce sledování aktivity útočníka bráno jako narušení soukromí. Kupříkladu v německém právním řádu je používání honeypotů ošetřeno ve prospěch jejich provozovatele. Co samozřejmě není možné, tak oplácet útočníkovi, který napadl honeypot, stejnou mincí, tj. útokem na jeho systém.

Kapitola 6

Použité honeypoty

Cílem této kapitoly, je prezentovat znalosti a zkušenosti získané při testování jednotlivých honeypotů, které by bylo možné dále využít. Podle zadání byla diplomové práce zaměřena na opensource honeypoty pod systémem Windows. Během tohoto testování vyvstala celá řada obtíží, které práci komplikovaly, což je i jeden z důvodů, proč je zde popsáno více honeypotů. Primární zaměření bylo na honeypot Honeyd, kvůli obtížím s jeho použitím se však zaměřila pozornost i na další opensource honeypot Labrea. Přesto, že se nakonec podařilo úspěšně zprovoznit verzi Honeyd pro Windows, je zde prezentován i obecnější úvod a získané zkušenosti s druhým honeypotem.

6.1 Honeyd

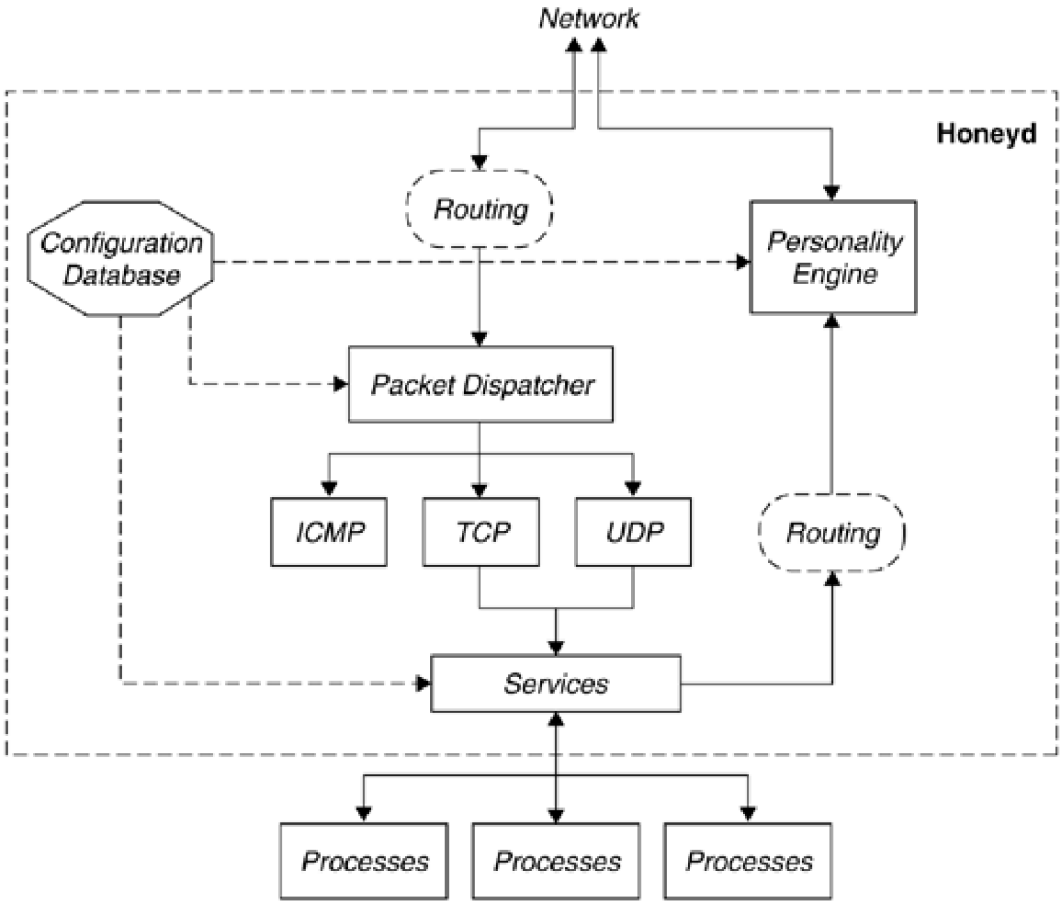
Za tímto názvem se skrývá nástroj spadající do kategorie virtuálních a low-interaction honeypotů. Z hlediska operačního systému se jedná o démona, který umožňuje na jednom systému simulovat několik (i tisíce) dalších systémů, které navenek vypadají jako samostatný systém. Podle nastavení to může být v podstatě libovolný operační systém, který je ale přístupný pouze přes síťové rozhraní. Přes toto rozhraní pak lze na simulovaném systému přistupovat ke službám jako je např. http, ftp a další. Stejně tak je možné tímto nástrojem vytvořit celou síť (honeynet) obsahující virtuální routery a další prvky. Takto vytvořená síť se pak snaží vypadat co nejvíce skutečně, u virtuálního routeru se generuje zpoždění atd.

Schéma samotného systému je na následujícím obrázku. Veškerá síťová komunikace je sledována v jednotce Packet Dispatcher. Jednotlivé služby si pak tyto příchozí pakety rozebírají, podle příslušných nastavení na ně reagují a odesílají zpět odpovědi.

Zdrojové kódy Honeyd jsou na internetu volně ke stažení, stačí je tedy pouze přeložit a program nainstalovat. Stejně tak je k dispozici i předkompilovaná verze pro Linux. Dalším krokem je pak umožnění průchodu příchozích paketů k honeypotu, kterého můžeme dosáhnout například přesměrováním IP. Při této možnosti však může docházet k velké duplikaci paketů, proto je vhodnější zároveň nastavit firewall, tak aby honeypot na pakety odpovídal, nicméně operační systém samotný je ignoroval. [4]

6.1.1 WinHoneyd - Problémy a omezení

Honeyd je primárně zaměřeno na systémy Linux nebo FreeBSD, především z důvodů flexibility, které poskytuje Unixové prostředí. Existuje však i verze portovaná na operační



Obrázek 6.1: Schéma činnosti honeypotu
[4]

systém Windows. Tato varianta je pak označována jako WinHoneyd, nicméně její použití v současných systémech je značně problematické.

Jádro problému spočívá v omezení tzv. Raw sockets, což jsou sokety, které umožňují odesílání a přijímání paketů přímo aplikací. Tato technologie je součástí API knihoven Berkeley sockets, které se staly de facto standardem pro abstrakci síťových soketů. Jeho implementace je tak obsažena v jádře systému Linux, Unix nebo BSD. Podpora těchto technologií byla pak přijata i v systémech Windows, kde dostala název Winsock. Původní verze systému Windows XP, vydaná v roce 2001, tedy obsahovala plnou podporu raw sockets. Nicméně po objevení možných problémů (např. s konektivitou), které tato implementace může způsobovat, přistoupil Microsoft k omezení této funkce. Blokování této vlastnosti způsobuje hotfix MS05-019 z roku 2005, který je neodstranitelný. Tato skutečnost proto používání WinHoneyd poměrně komplikuje, v současné době na nejvíce rozšířených operačních systémech Windows - XP se service packem 2 nebo 3, Vista případně 7 nelze WinHoneyd bez zásahu do systému používat. Pro testovací účely byl nicméně použit virtuální stroj se systémem Windows XP bez zmíněné hotfixu.

Zároveň jsem se obrátil na autory WinHoneyd, firmu netVigilance, kteří tuto skutečnost potvrdili, WinHoneyd se jim podařilo úspěšně otestovat pouze na systému Windows

XP používající service pack 1 případně bez něj. Zároveň přiznávají, že po téměř třech letech vývoje s cílem vytvořit snadno použitelnou portaci honeypotu Honeyd jakýkoliv další vývoj programu prakticky skončil. Pro potřeby firmy stále používají Honeyd, ovšem jeho původní linuxovou formu. Obdobně reagoval i autor knihy Honeypots for Windows Roger A. Grimes, který rovněž s používáním WinHoneyd skončil. Rovněž zmiňuje problémy, které se objevovaly již při psaní knihy a dále se prohlubovaly v následujících letech.

6.1.2 WinHoneyd - Instalace a konfigurace

I přes výše zmíněné problémy je stále WinHoneyd volně k dispozici jako open source. Z internetu je tedy možné stáhnout jak zdrojové kódy, tak již zkompilovanou verzi. Poslední verze programu je 1.5c, z roku 2008, k dispozici na stránkách firmy netVigilance - http://www.netvigilance.com/WinHoneyd. Zároveň je potřeba nainstalovat program WinPcap, což je opět portovaná verze linuxového pcap. Ten slouží k zachytávání paketů na síti a v Linuxu, Unixu i BSD je opět častou součástí jádra. Pro použití ve Windows je program portován a je třeba ho nainstalovat. Doporučená verze 3.1 je ke stažení v archivu http://www.winpcap.org/archive/.

Při použití starších verzí WinHoneyd je potřeba i další portovaný program WinArpd (port démona arpd), který slouží k zajištění dostatečného rozsahu adres pro honeypot. Od verzí WinHoneyd 1.0 a novějších pak tento program již potřeba není.

6.1.3 WinHoneyd - Práce s programem

Honeyd lze spustit s několika parametry, podle důležitosti se jedná o tyto následující:

- -f configfile: Parametr udává cestu ke konfiguračnímu souboru, který určuje, které systémy bude simulovat, které služby na nich poběží a další nastavení.
- -i interface: Určuje síťové rozhraní, které se pro honeypot použije. Jako výchozí se používání první, stejně tak lze naslouchat i na více rozhraních.
- -d: Spuštění v debuger módu, ve kterém je možné ladit jednotlivé konfigurace.
- -l logfile: Přepínačem l umožníme Honeyd aby zaznamenával přenosy paketů do log souboru, který určíme v parametru logfile.
- -s servicelog: Obdobné jako předchozí přepínač. V tomto případě se ale zapisují informace z jednotlivých simulovaných služeb.
- -p fingerprints: Tímto parametrem se určuje cesta k datovým souborů s otisky pro Nmap - scanovací nástroj který honeypotu umožní odhalit operační systém se kterým komunikuje. Pokud máme nainstalovanou novější verzi Nmap než je v honeypotu, použijeme právě tento parametr.
- -0 p0f-file: Cesta k databázi pasivních otisků, sloužících stejně jako v předchozím případu k identifikaci systému který s honeypotem komunikuje.
- -x xprob: Tímto parametrem se určí cesta k databázi otisků typu Xprobe, podobně jako Nmap slouží k identifikaci vzdáleného systému, využívá však jinou metodu určování.

- -a assoc: Cesta k asociovaným datům Nmap a Xprobe, v takovém přístupu se kombinují výhody obou nástrojů.

Hlavními soubory, které se pro úspěšné spuštění používají, jsou tedy konfigurační a logovací soubor. V konfiguračním souboru nastavujeme jednotlivé systémy, které chceme simulovat, v logovacím souboru sledujeme výsledky tj. jednotlivá připojení k těmto systémům.

Základní příkazy, které při sestavení konfiguračního souboru potřebujeme jsou především tyto :

- create: Vytváření nového vzoru pro virtuální stroj, za příkazem následuje název vzoru. Příkaz má tvar create `nazev_vzoru`.
- set : Tímto příkazem nastavujeme vlastnosti vybraného vzoru. Vlastností je několik, obecně slouží k větší věrohodnosti při přístupu k honeypotu. K základním patří vlastnost personality, která určuje jaký systém je simulován, dále pak například uptime, která udává jak dlouho již systém běží, nebo ethernet, který specifikuje MAC adresu. Příkaz má tvar set `nazev_vzoru` personality „`nazev_systemu`“'.
- add : Nastavuje u vybraného vzoru simulaci a sledování služeb. Do konfiguračního souboru je nutné zadat typ použitého protokolu (tcp nebo udp) a číslo portu, na kterém daná služba komunikuje. Dále je možné spustit skript, který danou službu simuluje, pokud není specifikován, jsou sledovány pouze příchozí pakety. Příkaz má tvar add `nazev_vzoru` tcp port `cislo_portu`.
- bind : Tento příkaz slouží k připojení vybraného vzoru na zadanou IP adresu, čímž vzniká na této adrese virtuální stroj. Příkaz má tvar bind `IP_adresa nazev_vzoru`.

Konfigurační příkazy jsou ve WinHoneyd stejné, jako v původní verzi pro Linux. Jejich kompletní seznam a popis je pak například zde - http://linux.die.net/man/8/honeyd.

Pro účely této diplomové práce byl vytvořen následující konfigurační soubor.

```
create WinXP1
set WinXP1 personality ,,Microsoft Windows XP SP1‘‘
set WinXP1 uptime 1794614
set WinXP1 default tcp action reset
set WinXP1 default udp action reset
set WinXP1 default icmp action open
add WinXP1 tcp port 20 open           # ftp - data
add WinXP1 tcp port 21 open           # ftp - control
add WinXP1 tcp port 22 open           # ssh
add WinXP1 udp port 22 open           # ssh
add WinXP1 tcp port 23 open           # telnet
add WinXP1 tcp port 25 open           # smtp
add WinXP1 tcp port 80 open           # http
add WinXP1 udp port 80 open           # http
add WinXP1 tcp port 110 open          # pop3
add WinXP1 tcp port 135 open          # Microsoft EPMAP
add WinXP1 udp port 135 open          # Microsoft EPMAP
add WinXP1 udp port 137 open          # NetBIOS NetBIOS Name Service
```

```
add WinXP1 udp port 138 open          # NetBIOS NetBIOS Datagram Service
add WinXP1 udp port 139 open          # NetBIOS NetBIOS Session Service
add WinXP1 tcp port 143 open          # imap
add WinXP1 udp port 143 open          # imap
add WinXP1 tcp port 445 open          # Microsoft-DS Active Directory
add WinXP1 udp port 445 open          # Microsoft-DS SMB file sharing
add WinXP1 tcp port 3128 open         # HTTP used by Web caches
set WinXP1 ethernet ,,3com‘‘

bind 192.168.1.1 WinXP1
```

Tato konfigurace vytvoří virtuální stroj `WinXP1`, který se bude identifikovat jako systém `Microsoft Windows XP Professional SP1` a dále hlásit že běží již `1794614` vteřin. Výchozí nastavení protokolů je pak upraveno pro konkrétní služby (příslušné porty jsou otevřeny), v poznámce je uvedeno o jakou službu se jedná. Na konec proběhne nastavení MAC adresy, která se vybere tak, aby odpovídala výrobci karet 3com a tato konfigurace se naváže na adresu `192.168.1.1` .

Obdobně lze vytvořit i virtuální routery a tím i celou část virtuální sítě.

6.1.4 WinHoneyd - Výstup z programu

Jak bylo zmíněno již výše, hlavní část výstupů tvoří log soubor se záznamy jednotlivých přístupů k honeypotu.

Zápis v logovacím souboru vypadá například takto :

Datum	Prot.	T	Zdroj		Cíl		Info	Systém
			IP	Port	IP	Port		
2005-04-02-15:35:15	tcp(6)	S	10.3.6.139	1827	10.1.2.124	3128		[Windows XP SP1]
2005-04-02-15:35:56	tcp(6)	-	10.3.6.139	4378	10.1.2.84	8080:	48 S	[Windows XP SP1]
2005-04-02-15:36:11	tcp(6)	-	10.4.7.196	2671	10.1.2.175	2380:	40 RA	[FreeBSD 5.0-5.1]
2005-04-02-15:39:47	tcp(6)	E	10.3.6.139	1827	10.1.2.123	3128:	9950 240	
2005-04-02-15:40:18	icmp (1)	-	10.3.5.182		10.1.3.99:		11(0): 56	

Obrázek 6.2: Příklad logovacího souboru honeypotu
[4]

První sloupec udává čas, kdy daný paket dorazil na honeypot, druhý pak určuje použitý protokol. Ve třetím sloupci může být S, pokud se jednalo o začátek nového spojení, E znamená ukončení spojení nebo - pokud paket nepatří k žádnému připojení. Následuje dvojice sloupců určující ip adresu odesílatele a port ze kterého paket přišel, druhá dvojice pak udává ip adresu cíle (ip adresu na které běží příslušný virtuální stroj) a cílový port. Další sloupec pak udává dodatečné informace o připojení nebo paketu. V případě ukončení připojení udává celkovou velikost přijatých a odeslaných paketů. Pokud se jedná o paket, který nepatří žádnému spojení, je zde velikost paketu a příslušný příznak, který je v hlavičce,

dle použitého protokolu. V posledním sloupci je pak operační systém identifikovaný na základě rozpoznaného otisku.

Druhým výstupem jsou pak logy jednotlivých služeb, které na příslušném virtuálním stroji běží. Formát výstupu je velmi podobný jako v případě paketů. V prvním sloupci je zaznamenaný čas, druhý udává protokol. Následuje dvojice ip adresa odesílatele + port a za dále pak ip adresa cíle + port. Poslední sloupec pak udává požadavek který daná služba zaznamenala, její formát je závislý na skriptu, kterým je reprezentována. V příkladu je například zaznamenán výpis ze simulovaného proxy serveru, přes který se někdo snaží anonymně připojit k mail serveru a odeslat z něj email.

Datum	Prot.	Zdroj		Cíl		Data
		IP	Port	IP	Port	
2005-04-10-00:56:48	tcp(6)	10.3.23.14	3259	10.1.3.222	3128:	CONNECT 10.4.228.113:25 HTTP/1.0
2005-04-10-00:59:12	tcp(6)	10.3.23.14	3343	10.1.3.124	8000:	CONNECT 10.5.167.5:25 HTTP/1.0
2005-04-10-01:04:20	tcp(6)	10.3.23.14	4116	10.1.3.209	3128:	some@net.em ⟶ schan@net.em

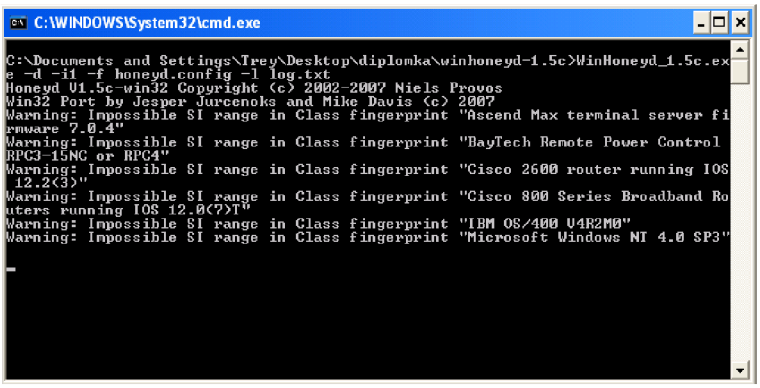
Obrázek 6.3: Příklad logovacího souboru služeb
[4]

6.1.5 WinHoneyd - Dosažené výsledky

Jak již bylo zmíněno, použitelnost WinHoneyd je silně limitovaná. Výsledky testování jsou proto poměrně neuspokojivé. Výše zmiňovaný hotfix, který ovlivňuje funkčnost aplikace, zasahuje úplně stejně i do komunikace sdílené NAT sítě, kterou lze vytvořit pomocí virtualizační aplikace jako je například použitý Virtual PC, případně obdobné jako např. VMware. Na virtuálním stroji pak dostáváme stejné výsledky jako na skutečném systému s blokováním raw soketů, WinHoneyd zůstane zablokováno při snaze otevřít si raw socket, nereaguje na signál pro ukončení a nevytváří ani logovací soubor. Tato situace je velmi nepříjemná. Jde především o to, že nalezení příčiny takového chování u honeypotu, může být pro člověka, který není s tímto prostředím a problematikou obeznámen velmi zdlouhavé. Rozpoznat, zda se jedná o nevhodnou konfiguraci honeypotu, problémy s nastavením sítě či zcela jiný problém tak zabere dosti času. Jak vypadá WinHoneyd uvízlé právě při snaze vytvářet raw socket je zachyceno na obrázku 6.4 na straně 24.

Pokud virtualizaci spustíme pouze jako lokální síť, do které umístíme dva počítače (jeden hostitelský, na kterém běží WinHoneyd, ze druhého se na něj dotazujeme), budou dosažené výsledky již mnohem lepší. Konfigurace propojení sítě je znázorněno na obrázku 6.5 na straně 25.

V tomto případě se již WinHoneyd může korektně spustit, což poznáme jednoduše díky přítomnosti vytvořeného logovacího souboru a stejně tak reakcí na signál ukončení, na který odpoví. Z druhého počítače spustíme programy pro scanování, například Nmap a Netscan. Netscan necháme hledat v rozsahu 169.254.30.19-169.254.30.20. Nalezne jedno zařízení, ke které můžeme zkusit připojit přes různé služby (ftp, http, telnet). Simulace služby spočívá pouze v otevřeném portu, proto na jednotlivé pokusy nedostaneme odpovídající odezvu. Všechny tyto žádosti jsou ale na honeypotu logovány. Program Nmap pak provádí klasický



Obrázek 6.4: WinHoneyd - Blokování raw soketů.

scan portů, postupně prochází všechny možné porty na scanovaném počítači a zjišťuje, zda je port otevřený či nikoliv. Průběh scanu je zachycen na obrázku 6.6. V horní části ve výpisu příkazové řádky je výpis z Nmapu, v dolní pak okno programu Netscan.

Jakmile dokončíme scan, můžeme se přesunout na počítač s honeypotem a zkontrolovat výsledky v logovacím souboru. Ten je zachycen na obrázku 6.7 na straně 6.7.

WinHoneyd v něm úspěšně identifikoval příchozí komunikaci jako Nmap scan. Na jednotlivých řádcích - viz dále, vidíme, že příchozí IP adresa odpovídá adrese počítače ze kterého jsme scan prováděli, cílová adresa je testovaný honeypot.

```
2010-04-21-15:47:25.1410 tcp(6) - 169.254.90.204 45862 169.254.30.19 424:
40 S [NMAP syn scan 1]
2010-04-21-15:47:25.1410 tcp(6) - 169.254.90.204 45862 169.254.30.19 593:
40 S [NMAP syn scan 1]
```

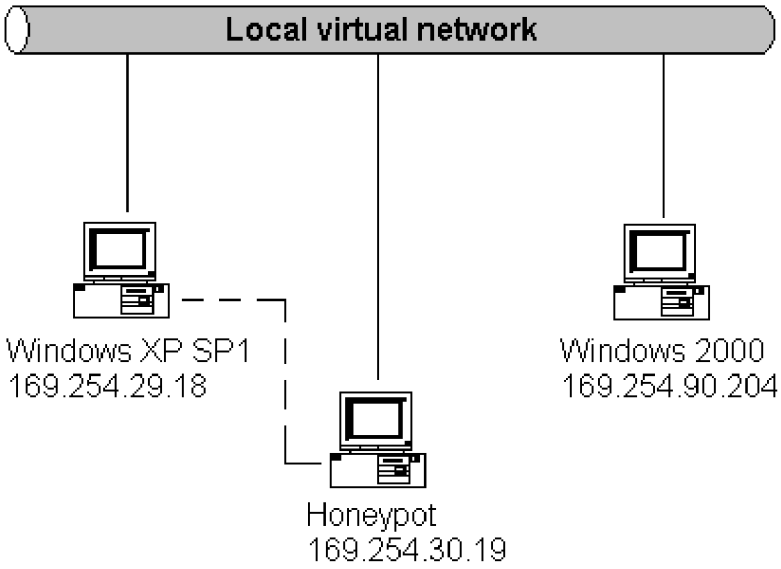
Předcházející záznamy zachycují pokusy o připojení programem Netscan. Například na následujících řádcích je zachycena komunikace přes Telnet, kterou poznáme podle použitého cílového portu. Podle času můžeme celou komunikace dobře sledovat - Netscan odešle žádost o spojení a když na ni nedostane odpověd, následuje ukončení spojení.

```
2010-04-21-15:46:12.9310 tcp(6) S 169.254.90.204 1040 169.254.30.19 23
[Windows XP SP1]
2010-04-21-15:46:16.9570 tcp(6) E 169.254.90.204 1040 169.254.30.19 23:
0 0
```

Tohoto výsledku, přestože není nikterak průlomový, se bohužel nepodařilo dosáhnout hned. Jak již bylo uvedeno, samotné spuštění tohoto honeypotu na systému Windows, je závislé na velkém množství faktorů, počínaje použitým operačním systémem, nastavením sítě a programovým vybavením (jako je např WinPcap) konče. V porovnání s operačním systémem Linux, kde se Honeyd často objevuje jako součást distribuce, je pak spuštění na systémech Windows jednoznačně pracnější a náročnější.

6.1.6 WinHoneyd - Shrnutí

Pro úspěšné nasazení WinHoneyd na sběr dat z Internetu, je tedy hlavním problémem sestavení vhodného prostředí, kde může honeypot fungovat. Možností by mohlo být buďto



Obrázek 6.5: WinHoneyd - Schéma umístění honeypotu.

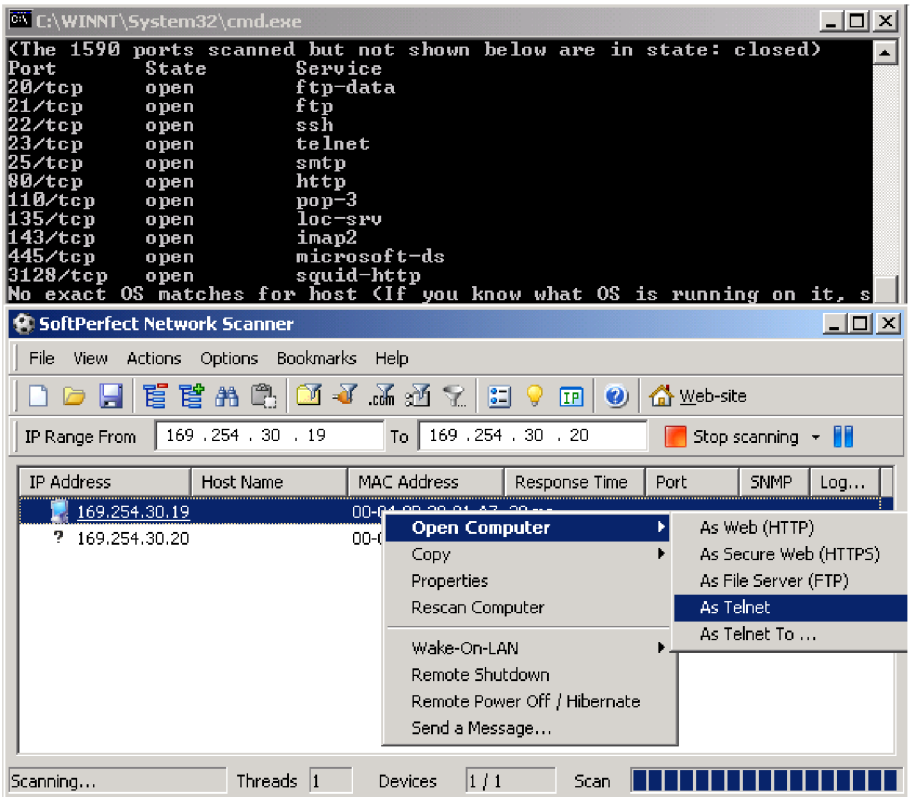
použití Windows 2003 Server, kde podle vyjádření Microsoftu je možné používat ke komunikaci raw sockety. Při komunikaci s lidmi z firma netVigilance, která jsou autorem poslední verze WinHoneyd, se však o této variantě zmiňovali rovněž pesimisticky. Zmíněný systém při psaní této práce nebyl k dispozici, proto tyto skutečnosti nemohu plně ověřit.

Použití systému Windows XP nepřipadá příliš v úvahu. Aby bylo možné WinHoneyd úspěšně použít, je nezbytná nepřítomnost výše zmíněného hotfixu, potažmo Service Packu 2. V tom případě ale značně utrpí bezpečnost, protože právě Service Pack 2 přidává množství nových prvků pro zabezpečení systému.

Podle oficiálních zdrojů, které Microsoft vydal k problematice raw soketů [10], by mělo být možné použít systém Windows 2000. Při použití starších verzí bylo možné, při testování na virtuální síti, tento systém úspěšně použít, ale stejně jako v případě Windows XP, byla záměrně použita starší verze systému, z doby před zmíněným hotfixem. Přesto, že by ani v novějších verzích neměly být raw sokety limitovány, nebyla tato skutečnost ověřena.

Z hlediska současných systému, jako jsou Windows Vista a Windows 7 platí stejná omezení jako pro Windows XP se Service packem 2. [10]

Druhou alternativou je proto využít virtualizační prostředí se sdílenou sítí na operačním systému, který raw sokety povoluje, což může být třeba i Linux. Tato možnost mi ale připadá pro reálné použití poměrně nešikovná. Skutečnost, že na operačním systému Linux simulujeme virtuální OS s prostředím Windows, abychom v něj mohli používat program portovaný právě z Linuxu je už téměř bizarní.



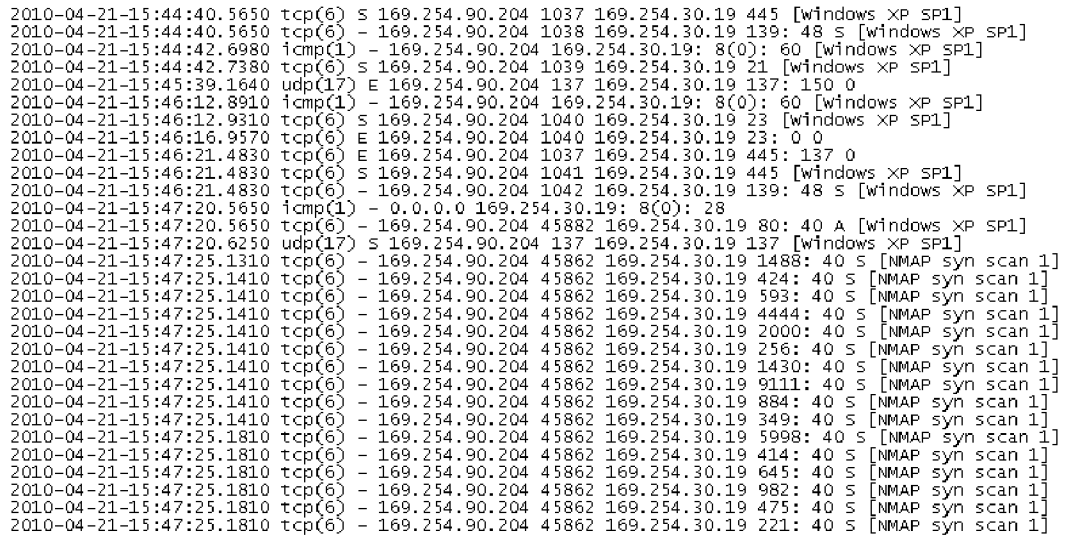
Obrázek 6.6: WinHoneyd - Scanování honeypotu.

6.2 LaBrea

Protože výsledky práce s WinHoneyd nebyly zdaleka uspokojivé, a v mnohém zaostávaly za možnostmi a kompaktností jeho použití v Linuxu, přišlo na řadu testování dalších alternativ honeypotu. Přestože se nakonec WinHoneyd podařilo úspěšně otestovat alespoň na lokální síti, chtěl bych zde uvést poznatky zjištěné při práci s honeypotem LaBrea.

Stejně jako předchozí WinHoneyd je i LaBrea šířen jako opensource. Na rozdíl od Honeyd, které je klasickým honeypotem se pro LaBrea často používá pojem tarpit, v doslovném překladu „jáma s dehtem“. Jeho cílem je zpomalení a zdržení příchozí komunikace, kterou monitoruje. Prioritní zaměření je tak na červy, automatické scanování systému a podobné aktivity. Vytvořen byl jako odpověď na červa Code Red z roku 2001, přičemž se v počátku jednalo pouze o Pythonovský skript, běžící na Unixovém stroji. [11] Rychle se však přeformoval do podoby samostatné aplikace a dočkal se portu na platformu Windows.

Princip fungování LaBrea je vcelku jednoduchý. Program naslouchá ARP žádosti, které vysílá například scannovací program, a které přicházejí na nepoužívané IP adresy. Na tyto žádosti odpovídá, čímž obdrží od scanneru iniciační SYN paket a jako odpověď odesílá SYN/ACK. Přestože se prakticky žádné skutečné spojení neustanoví, scanner je přesvědčen, že tří-cestý handshake je proveden úspěšně a začne vysílat data. Ty ale k cíli nikdy nedorazí a jakmile vyprší časový limit pro odpověď, začne se scanner, pod dojmem toho, že komunikuje se skutečným systémem, snažit o opětovné odeslání dat. Tímto způsobem dochází k požadovanému zdržení, které evokuje právě analogii s dehtovou jámou, do které se útočník zachytí.



Obrázek 6.7: WinHoneyd - Logování scanu.

6.2.1 LaBrea - Problémy a omezení

V dalších verzích je přidána další funkčnost, možnost odpovědi na příchozí data a jejich logování. Je pak možné mluvit o plnohodnotném honeypotu, nikoliv pouze o tarpitu. Všechny tyto možnosti jsou dostupné v aktuální verzi, která byla v době psaní této práce k dispozici. Zároveň však sám autor upozorňuje na jeden z podstatných problémů - nemožnost používat LaBrea na systému Windows XP, podle knihy Honeypots for Windows [5] platí stejné omezení i pro Windows 2003 server. Pro testovací účely byl proto použit virtuální stroj se systémem Windows 2000, stejně jako u předchozího honeypotu bez aktualizací, které by mohly způsobovat konflikty a problémy.

6.2.2 LaBrea - Instalace a konfigurace

Zmíněné problémy vypadají na první pohled velmi podobně jako u WinHoneyd. Samotný program získáme snadno z Internetu, autor jej distribuoval na opensource serveru sourceforge - http://sourceforge.net/projects/labrea/files/. Poslední spustitelná verze 2.5.1 jak pro systém Linux tak Windows pochází z 30.10.2003, tedy již poměrně staršího data.

Zároveň je potřeba nainstalovat program WinPcap, který slouží k zachytávání paketů (http://www.winpcap.org/, stejně jako u WinHoneyd) a navíc pak knihovnu LibnetNT. Ta je opět portem Linuxové knihovny Libnet, která slouží k implementaci funkcionality paketů na nižších vrstvách. [12] Ke stažení je například zde -

http://members.fortunecity.com/sektorsecurity/downloads/links/libnetnt.zip.

6.2.3 LaBrea - Práce s programem

V možnostech parametrů, se kterými je tento program možné spustit, má LaBrea ještě větší rozsah než WinHoneyd. Základní příkazy, jsou obdobné.

6.2.4 LaBrea - Dosažené výsledky

Při spuštění honeypotu s doporučenými parametry, se kterými by měl program správně pracovat, ale nastal problém. Program vrátil chybu oznamující nemožnost vytvoření linkového rozhraní. Pokud toho chybové hlášení vyhledáme ve zdrojových kódech, dostaneme se k souboru lbio.c, který tvoří součást balíčku Labrea. K prohlédnutí je např. zde - http://labrea.sourcearchive.com/documentation/2.5-stable-3/lbio_8c-source.html. Příslušná problematická pasáž je pak níže.

```
/* open link interface for raw packet output */
if ((io.eth = eth_open(io.ifent->intf_name)) == NULL ) {
  warnx(,,*** Couldn't open libdnet link interface‘‘);
  util_clean_exit(1);
}
```

Jak je z této části patrné, problém je opět spojen s raw sokety, v tomto případě dojde k chybě při jeho vytváření, patrně opět ze stejných důvodů, jako u výše uvedeného WinHoneyd. Výskyt této chyby je však zvláštní, v závislosti na použitém systému by tyto problémy nastat neměly.

Kapitola 7

Opensource honeypoty v prostředí Windows

Tato kapitola je věnována problematice raw soketů, které jsou hlavním problémem při používání honeypotů pod současnými systémy Windows. Přesto, že oba použité honeypoty jsou uzpůsobeny pro chod pod operačními systémy Windows, během jejich testování se vyskytlo poměrně značné množství problémů, které jejich chod omezovaly či přímo znemožňovaly.

7.1 Problematika raw soketů

Tento problém se u obou dvou testovaných honeypotů vyskytoval zřejmě nejčastěji. Jak již bylo zmíněno výše, raw sokety umožňují odesílání a přijímání paketů přímo aplikací. Je přes ně možné přistupovat přímo k celému paketu a to jak pro čtení, tak i pro zápis. U obyčejných soketů se o obsluhu příchozího či odchozího paketu stará operační systém. Pokud uvažujeme například TCP paket, který obsahuje hlavičku, je u příchozího paketu nejprve tato hlavička odebrána a pak operační systém předá data z paketu příslušné aplikaci, pro který je paket určen. V případě odchozího paketu je naopak k datům, které operační systém dostane od aplikace, hlavička přidána a paket je odeslán. Pokud jsou použity raw sokety, dochází svým způsobem k obcházení operačního systému, protože aplikace dostává přímo celý paket, který dorazí na síťové rozhraní. Z hlediska síťové bezpečnosti je tato technologie, jak už to často bývá, dvousečná. Může sloužit jak ke zvyšování bezpečnosti, v našem případě například pomocí honeypotů, ale stejně tak je možné ji zneužít ve škodlivém softwaru, jako jsou Trojští koně, červi nebo viry.

Z hlediska podpory této technologie v operačních systémech je situace následující. Operační systémy Unix a jemu podobné, tj. například Linux nebo BSD, umožňují použití raw soketů již dlouhou dobu. Všechny tyto implementace v zásadě vychází z návrhu Berkeley Sockets, který vznikl již v 80. letech. V systémech Windows je situace mnohem nepřehlednější. Ve starších systémech jako Windows 95 nebyly raw sokety implementovány vůbec. V dalších implementacích WinSock, což je zkratka pro Windows Socket API, se pak již tato možnost objevuje, nicméně nemá příliš dlouhého trvání. Přestože v systémech Windows XP a Windows 2000 bylo původně možné raw sokety používat, na základě některých možných problémů a bezpečnostních hledisek však byla tato funkce omezena. Toto omezení přichází, jak už bylo dříve zmíněno, jako součást Service Packu 2, případně samostatně v hotfixu MS05-019 z roku 2005. Důvodů, které k tomu Microsoft vedly, je několik. Jedním z nich může být například možnost zneužití odchozího UDP datagramu s podvrženou IP, který

se odešle právě přes raw soket, jako DoS útoku. [10] Stejně tak by mohl raw soket, který se váže na síťovou a transportní vrstvu, narušit konektivitu některých dalších služeb nebo aplikací.

7.2 Řešení problému

Pokud chceme používat raw sokety ve Windows, je tedy nezbytné vědět, zda konkrétní systém tuto možnost vůbec umožňuje. Z hlediska použití honeypotu pak vycházím čistě z vlastního testování na jednotlivých systémech a informací od jejich autorů (firma NetVigilance) a autorů publikací o honeypotech (Roger A. Grimes). Jako referenční honeypot jsem vybral WinHoneyd a podle jednotlivých zkušeností s různými operačními systémy a jejich verzemi, případně přavzatými informacemi, sestavil následující přehled.

1. Otestováno fungující WinHoneyd
    - Windows 2000 SP0
    - Windows 2000 SP1 - dle tvrzení Rogera A. Grimese
    - Windows XP SP0
    - Windows XP SP1 - dle tvrzení NetVigilance
2. Otestováno nefungující WinHoneyd
    - Windows XP SP2
    - Windows XP SP3 - dle tvrzení NetVigilance
3. Nejspíše nefungující WinHoneyd
    - Windows Vista - dle tvrzení NetVigilance
    - Windows 7 - dle tvrzení NetVigilance
    - Windows 2003 Server* - dle tvrzení NetVigilance

* U Windows 2003 Server je ze strany Microsoftu uváděna podpora raw soketů, nicméně autoři WinHoneyd se k funkčnosti honeypotu na tomto systému staví spíše skepticky.

Ze strany Microsoftu je pak uváděno ještě jedno možné řešení, které spočívá v implementaci vlastního ovladače TDI (Transport Driver Interface) pro specifický síťový protokol spolupracující s WinSock. Takový protokol by se pak přidal do seznamu podporovaných ve WinSock katalogu. [10] Takový ovladač by pak běžel v režimu jádra a mohl přistupovat k síťovému adaptéru. Bohužel, tato implementace rozhodně není nic triviálního, což by mohla být také jedna z možností, proč s tímto řešením ještě autoři opensource honeypotů nepřišli. Naopak lze usuzovat, že komerční honeypoty pro Windows, jako například KFSensor, který mi byl několikrát doporučen, právě toto řešení používají.

Při řešení této diplomové práce se jako dostatečné ukázalo použití starších systémů Windows. Pro lokální testování na virtuálních strojích ale není absence příslušných service packů zdaleka tak kritická, jako tomu může být u skutečného stroje s přístupem do Internetu.

Kapitola 8

Analýza síťového útoku

Další podstatnou částí této diplomové práce, je analyzování zachyceného útoku. Z důvodů obtíží spojených se spouštěním honeypotů, se tato analýza prováděla pouze v rámci lokálního síťového rozhraní. Opět byly použity virtuální stroje z minulé kapitoly, které sloužily pro testování honeypotů.

Jeden virtuální stroj představoval útočníka, na druhém bylo třeba útok zachytávat. Pro funkci zachytávání útoku existuje vcelku široká nabídka takzvaných packet sniffers - programů určených pro odposlouchávání a zachytávání paketů ze síťového rozhraní. Na druhém PC, které představuje útočníka, je pak třeba použít některý z programů, určených pro testování zranitelnosti systému.

Jakmile je útok zachycen, je potřeba data, která jsou z něj získána určitým způsobem zpracovat. Výstupy, které produkují programy zachytávající útok, jsou v textové formě. To ale pro další zpracování není příliš vhodné. Získaná data je proto potřeba nejprve pročistit, vybrat z nich pouze ty, které se týkají útoku. Z nich je pak třeba na základě definovaných metrik získat informace, které tento útok určitým způsobem popisují. Výsledky z metrik je pak potřeba uložit tak, aby k nim bylo možno rychle přistupovat, tedy například do datové struktury.

8.1 WinDump

Pod tímto názvem se skrývá další z řady aplikací původně vyvinutých na systémech Linux, která se dočkala portu i na operační systém Windows. Původní verze nese název TcpDump, pro operační systém Windows je pak pozměněn na WinDump. To je však pravděpodobně jediná zásadní změna, ostatní vlastnosti, jako je možnost spouštění z příkazové řádky a stejně tak parametry pro konfiguraci zůstávají stejné jako u původní verze.

8.1.1 WinDump - Problémy a omezení

Pro spuštění této aplikace je nezbytná přítomnost WinPcap, které již bylo zmíněno dříve v souvislosti se spuštěním honeypotu. Z hlediska použitého operačního systému by neměly vznikat žádné problémy, WinDump je spustitelný pod drtivou většinou operačních systémů Windows za posledních několik let (Windows 95, 98, ME, NT, 2000, XP, 2003) a tudíž by nemělo docházet k problémům podobným těm, které vznikly při testování honeypotu. Z hlediska nejnovějších systémů by za použití posledních rozhraní WinPcap měl být program funkční i na systému Windows Vista.

8.1.2 WinDump - Instalace a konfigurace

Stejně jako předchozí použité programy, i WinDump lze díky vývoji pod BSD licencí získat z internetu. Ke stažení je například zde - http://www.winpcap.org/windump/. Instalace není nutná, program je již přímo spustitelný, při použití aktuálního rozhraní WinPcap tak při instalaci nenastaly žádné problémy ani komplikace.

8.1.3 WinDump - Práce s programem

Stejně jako ostatní předchozí nástroje, i WinDump používá širokou škálu parametrů, kterými můžeme nastavit jeho chování při sběru paketů. Pro jednoduchost jsou uvedeny pouze ty, které byly při práci použity, kompletní seznam je k dispozici na manuálových stránkách WinDumpu, resp. TcpDumpu.

- -D : Při zadání tohoto parametru vypíše WinDump seznam nalezených rozhraní, ze kterých může zachytávat pakety. Rozhraní je identifikováno jednak číslem a dále pak názvem. V operačních systémech Windows 2000 a novějších je však název tvořen velmi dlouhým a složitým řetězcem, proto je vhodnější použití číselného označení.
- -i interface: Specifikuje síťové rozhraní, na kterém program naslouchá a ze kterého sbírá pakety. Jak bylo uveden výše, jako interface lze použít číselné označení i název. Pokud tento parametr není specifikován, použije se síťové rozhraní s nejnižším číslem v seznamu nalezených rozhraní.
- -n : Parametr udávající, že IP adresy se nebudou překládat na jména.
- -v : Přidává další údaje do výstupu, jako například Time To Live, celkovou délku paketu, kontrolní součet nebo identifikaci.
- -tttt : Na každý řádek, tj. ke každému zachycenému paketu, přidává časové razítko. To je ve formátu rok-měsíc-den hodiny:minuty:vteřiny. Razítko tedy vypadá například takto - 2010-05-06 15:40:32.442776
- -w file : Parametr nastavuje výstup do souboru zadaného řetězcem file namísto výpisu na standardní výstup na obrazovku. Zápis je v hexadecimálním tvaru.
- -r file : Nastavuje čtení ze zadaného vstupního souboru zadaného řetězcem file, který byl předtím programem vytvořen. Pokud soubor není zadán, je použit standardní výstup.
- -x : Zadáním tohoto parametru se do výstupu přidávají data obsažená v paketu v hexadecimálním tvaru. Vypisují se tak pouze data, hlavičky paketu jsou osekány.
- Dále pak může být jako parametr zadán výraz, který dále specifikuje pakety, které se mají vybrat. Z možných výrazů je použit příkaz dst, který specifikuje výběr pouze na cílovou adresu, na kterou pakety přicházejí. Výraz tedy vypadá např. takto : dst 169.254.30.19

Kompletní seznam všech možných parametrů a výrazů, které lze aplikovat lze najít v manuálových stránkách WinDumpu, případně TcpDumpu.

Konkrétní postup při použití programu WinDumpu byl v této práci následující. Nejprve spustíme sbírání paketů z vybraného rozhraní a uložíme je do souboru. Tam se uloží kompletní pakety v hexadecimálním tvaru. Pokud budeme potřebovat vybírat jiné informace,

bude možné použít dříve vytvořené soubory, pouze použijeme jiné parametry na výběr ze souboru. Syntaxe tedy vypadá například následovně :

```
windump -i 1 -w pakety.txt
```

Druhým krokem je pak výběr informací, které chceme analyzovat. Zde už použijeme předchozí uložené pakety, ze kterých vybereme požadované informace. Použijeme parametry popsané výše. Chceme tedy nepřekládat IP adresy na jména, uvádět rozšířené údaje u každého paketu, vkládat časová razítka a přidávat data, která jsou v paketu posílaná. Ze záznamů navíc vybereme pouze ty, kde jako cílová nebo zdrojová adresa figuruje IP adresa počítače, na který se útočí. Tyto výsledky pak místo výpisu na obrazovku uložíme do souboru.

Použijeme tedy například následující syntaxi :

```
windump -n -v -tttt -x -r pakety.txt dst 169.254.29.18 or src 169.254.29.18
> ms06_40_in_out.txt
```

Záznam o jednom zachyceném paketu v souboru output.txt pak vypadá takto :

```
2010-05-13 10:53:29.619774 IP (tos 0x0, ttl 128, id 12197, offset 0,
flags [DF], proto: TCP (6), length: 40) 169.254.90.204.1100 >
169.254.29.18.445: ., cksum 0x40a5 (correct), ack 1 win 17520
0x0000:  4500 0028 2fa5 4000 8006 ff4f a9fe 5acc
0x0010:  a9fe 1d12 044c 01bd 7af1 a530 dcaf 5c09
0x0020:  5010 4470 40a5 0000 0000 0000 0000
```

8.2 Vytvoření útoku

Pro analyzování útoku v podmínkách lokální sítě je nezbytné jeho vytvoření. Možných útoků existuje celá řada. Pří výběru pak bylo jedno z hlavních kritérií, aby byl útok pokud možno dobře zdokumentovaný a existovaly jeho testovací implementace. Protože celá práce je zaměřená na systémy Windows, byl proto vybrán útok, jehož cílem je služba tohoto systému. Těmto kritériím vyhovuje například útok na zranitelnost MS06_40, který je popsán dále.

8.2.1 Útok na zranitelnost MS06_40

Tento útok spočívá ve zneužití zranitelnosti ve službě Server, konkrétně ve funkci CanonicalizePathName() modulu NetApi32. [14] Tato zranitelnost spočívá v možnosti útoku typu buffer overflow, který má následující princip. Na začátku je existence neošetřeného bufferu (implementační chyba). Při zápisu většího množství dat, než jaká je velikost bufferu, tak logicky dojde k přetečení bufferu do další části paměti. Pokud se ovšem toto přetečení provede cíleně a data, kterými se buffer přeplní jsou podle tohoto záměru upravena, je možné jimi přepsat část paměti, kde je uložena návratová adresa z podprogramu. Jestliže na tuto adresu pak útočník umístí svůj vlastní škodlivý kód a zajistí tak i jeho provedení. Výsledkem útoku je tak vzdálené spuštění kódu. Závažnost této zranitelnosti je označena nejvyšší úrovní, tedy jako kritická a to v systémech Windows 2000, Windows XP Service Pack 1, Windows XP Service Pack 2, Windows Server 2003, a Windows Server 2003 Service Pack 1. [13]

Podle zachycených dat můžeme útok rozdělit na několik fází. První spočívá v jakémsi scanu - ověření, zda na portu 4444 není spuštěná nějaká aplikace. Pokud je port volný může se při útoku dále využít a právě z toho důvodu se tato dostupnost hned na začátku ověřuje. Útočník tak zahájí spojení paketem typu SYN směrovaným na port 4444 na sledovaném pc. Pokud by na portu již nějaká aplikace naslouchala, jako odpověď by se odeslal paket typu SYN. Protože ale na tomto portu nic prozatím není, je odeslán paket RST, který znamená odmítnutí (reset) tohoto spojení. Pro útočníka to zároveň znamená zjištění, že tento port je volný.

Hlavička prvního SYN paketu vypadá takto.

```
2010-05-13 10:53:29.610006 IP (tos 0x0, ttl 128, id 12195, offset 0,
flags [DF], proto: TCP (6), length: 48) 169.254.90.204.1099 >
169.254.29.18.4444: S, cksum 0x39db (correct),
2062593233:2062593233(0) win 16384 <mss 1460,nop,nop,sackOK>
```

Jako odpověď se odešle RST paket.

```
2010-05-13 10:53:29.610091 IP (tos 0x0, ttl 128, id 2358, offset 0,
flags [none], proto: TCP (6), length: 40) 169.254.29.18.4444 >
169.254.90.204.1099: R, cksum 0xa68b (correct),
0:0(0) ack 2062593234 win 0
```

Pak už může začít samotný útok. Další pakety už jsou cíleny na port 445, kde naslouchá služba SMB. Po navázání komunikace začne útočník posílat datové pakety určené pro tuto službu. Pakety které jsou pro ni určené obsahují příslušnou hlavičku, která obsahuje identifikaci jejího protokolu. Hodnota prvních 4 bajtů tohoto protokolu je 0xFF, 'SMB', v hexadecimálním tvaru tedy ff:53:4d:42, což taky můžeme v zachycených paketech najít. Tato komunikace pokračuje tak dlouho dokud v službě nenastane buffer overflow.

Zde je ukázka části této komunikace. Příchozí paket pro službu SMB vypadá například takto.

```
2010-05-13 10:53:29.619899 IP (tos 0x0, ttl 128, id 12199, offset 0,
flags [DF], proto: TCP (6), length: 128) 169.254.90.204.1100 >
169.254.29.18.445: P 1:89(88) ack 1 win 65535
0x0000:  4500 0080 2fa7 4000 8006 fef5 a9fe 5acc
0x0010:  a9fe 1d12 044c 01bd 7af1 a530 dcaf 5c09
0x0020:  5018 ffff 1a47 0000 0000 0054 ff53 4d42
0x0030:  7200 0000 0018 0128 0000 0000 0000 0000
0x0040:  0000 0000 0000 9e54 0000 f900 0031 0002
0x0050:  4c41
```

Obdobně vypadá i odpověď, opět se jedná o posílání dat protokolu SMB.

```
2010-05-13 10:53:29.620346 IP (tos 0x0, ttl 128, id 2360, offset 0,
flags [DF], proto: TCP (6), length: 129) 169.254.29.18.445 >
169.254.90.204.1100: P 1:90(89) ack 89 win 64152
0x0000:  4500 0081 0938 4000 8006 2564 a9fe 1d12
0x0010:  a9fe 5acc 01bd 044c dcaf 5c09 7af1 a588
0x0020:  5018 fa98 9d0c 0000 0000 0055 ff53 4d42
0x0030:  7200 0000 0098 0128 0000 0000 0000 0000
0x0040:  0000 0000 0000 9e54 0000 f900 1103 0003
0x0050:  0a00
```

Jakmile dojde ke zneužití výše popsaná zranitelnosti, útočník se pokusí opět připojit k portu 4444. Pokud byl útok úspěšný tak se právě na tomto portu vytvořila vzdálená konzole. Při pokusu o komunikaci se tak jako odpověď odešle paket typu SYN a vytvoří se spojení. Na port jsou pak přijímány řídící pakety pro vzdálenou konzoli, ty jsou prováděny a jako odpověď jsou odeslány pakety obsahující výstup z konzole.

Tento paket například obsahuje informaci o aktuálním adresáři.

```
2010-05-13 10:53:40.531106 IP (tos 0x0, ttl 128, id 2387, offset 0,
flags [DF], proto: TCP (6), length: 53) 169.254.29.18.4444 >
169.254.90.204.1099: P, cksum 0x9b38 (correct),
111:124(13) ack 251 win 63990
0x0000:  4500 0035 0953 4000 8006 2595 a9fe 1d12
0x0010:  a9fe 5acc 115c 044b dcb1 6412 7af0 adcc
0x0020:  5018 f9f6 9b38 0000 0d0a 433a 5c57 494e
0x0030:  444f 5753 3e
```

Část `0d0a 433a 5c57 494e 444f 5753 3e` převedená do ascii znamená `C:\WINDOWS>`.

Vytvoření a práce s konzolí je poslední a finální částí útoku. Z hlediska detekce se už jedná o druhotný efekt, protože možností, jak postupovat po prolomení zabezpečení, je celá řada. Proto je potřeba se zaměřit na detekci činností a situací které jí předchází.

8.2.2 Provedení útoku

K samotnému provedení takového útoku v podmínkách lokální virtuální sítě je třeba použít odpovídající nástroj. Takovýto program je označován jako nástroj pro penetrační testování. Ten pak umožňuje spouštění specifických exploitů - tedy možných druhů zneužití určité zranitelnosti. Podstatnou částí je také výběr takzvaného payloadu - kódu, nebo obecně činnosti, která se bude provádět po úspěšném zneužití dané zranitelnosti. Takovéto systémy jsou vyvíjeny například i jako opensource. Z hlediska zneužitelnosti takového prostředí ke skutečnému útoku pak autoři také často dodávají, že veškeré informace a nástroje, které v rámci takového projektu poskytují, slouží pouze k legálnímu výzkumu a testování.

8.3 Metriky pro popis útoku

Prvotní členění a výběr dat nastává již při výstupu z programu WinDump. Z kompletního odposlechnutého provozu na síti se vyberou pouze pakety, které mají souvislost se sledovaným systémem. Toho je docíleno, jak již bylo zmíněno výše, pomocí parametrů src a dst, pro výběr paketů přijatých a odeslaných na zkoumané IP adrese.

Pro zpracování dalších dat, jejichž objem může mít stále objem několika stovek paketů, jsou použity skripty. Použitý skriptovací jazyk bylo možné vybrat libovolně. Z důvodu předchozí krátké zkušenosti s jazykem Python, byl vybrán právě tento jazyk. Konkrétně pak verze 2.6, která je portována i pro prostředí Windows. Prostředí pak bylo dále doplněno o modul MySQLdb, který umožňuje přistupovat k MySQL databázi. Celkem byly vytvořeny tři hlavní skripty.

- dump_export.py - slouží k prvotnímu zpracování souboru, který je výstupem z programu WinDump.
- metrika.py - skript obsahuje metriky, které slouží k popsání útoku

- db_insert.py - finální skript pro připojení se k databázi, do které jsou výsledky metrik vloženy

Pro práci z databází jsou pak určeny kratší skripty db_create.py, který slouží pro vytvoření databáze a tabulek, a db_clear, který je naopak určen k jejich smazání.

Jednotlivé skripty budou podrobněji popsány dále, pro jednodušší obsluhu jsou navíc přidány dávkové soubory db_create.bat, který na adrese localhost vytvoří novou databázi dip podle schématu definovaného v podkapitole Datová struktura -reference-. Dávkou db_insert.bat se pak provede analýza výstupu z programu WinDump a vložení výsledků do databáze. Pro smazání databáze je pak určena poslední dávka - db_clear.bat

8.3.1 dump_export.py

Jak již bylo zmíněno výše, tento skript je určen pro pro prvotní zpracování dat z programu WinDump. Skript má jeden vstupní parametr -i, kterým lze zadat vstupní soubor, který chceme zpracovat. Konkrétní použití tedy vypadá například následovně.

```
python dump_export.py -i  ms06_40_in_out.txt
```

Ve výstupu WinDumpu jsou obsaženo více různých druhů paketů. Je proto nutné použít více výstupních souborů a do každého vkládat pouze jeden druh. Proto jsou použity tři výstupní soubory, pro pakety IP protokolu (TCP a UDP), ICMP protokolu a protokolu ARP. Výstupní soubory jsou pak pojmenovány jako ip_output.txt, icmp_output.txt a arp_output.txt.

Informace obsažené ve výstupu z programu WinDump je pak dále nutné zpracovat. Skript proto jednoprůchodově prochází po řádcích vstupní soubor a získává z něj informace. Některé nejsou z hlediska analýzy tak podstatné, některé je třeba upravit do vhodného formátu. Stejně tak se liší data získaná z různých druhů paketů. Nejobjemnější a zřejmě i nejdůležitější jsou informace o paketech IP protokolu. U nich jsou vybírány tyto položky.

- Datum - datum přijmutí/odeslání paketu
- Čas - čas přijmutí/odeslání paketu
- Protokol - použitý protokol, v tomto případě IP
- TTL - hodnota time to live
- ID - id paketu
- Protokol - tcp případně udp protokol
- Délka - délka paketu
- Zdrojová IP - IP adresa odkud byl paket odeslán
- Zdrojový port - port ze kterého byl paket odeslán
- Cílová IP - IP adresa na kterou byl paket odeslán
- Cílový port - port na který byl paket odeslán
- Flag - příznak určující typ paketu, např. S pro SYN paket atd.

- Délka dat/checksum - u řídících paketů výsledek kontrolního součtu, u datových paketů pak množství dat v paketu v bytech
- Data - Data obsažená v datové části paketu

Tyto informace se pak ukládají do souboru ip_output.txt kde je z důvodu dalšího zpracování umístěn každý záznam na jednom řádku, jednotlivé hodnoty jsou odděleny mezerami.

Jeden záznam pak vypadá například takto.

```
2010-05-13   10:53:29.679491   IP   128   2368   TCP   91   169.254.29.18
445   169.254.90.204   1100   P   51   4500005b0940400080062582a9fe1d12a9
fe5acc01bd044cdcaf5fb57af1aabd5018f563682c00000000002fff534d422f000000009
80120000000000000000000000000000089e540108f90006ff002f0000
```

Po úspěšném zpracování celého souboru vypíše skript informaci o počtu zpracovaných záznamů a rozložení dat z hlediska výstupních souborů.

8.3.2 metrika.py

Získaná data je dále nutné zanalyzovat. K tomuto účelu jsou pro navrženy metriky, které popisují jejich vlastnosti. Tyto vlastnosti jsou získány, či dále odvozeny podle dané metriky, z předchozích výsledků. Skript s metrikami je možné spustit se čtyřmi parametry, jsou to tyto následující.

- -i input_file - tímto parametrem zadáme skriptu vstupní soubor zadaný dle řetězce input_file. Výchozí hodnotou je ip_output.txt.
- -s similar - při výběru dat, která se budou ukládat do databáze je možné specifikovat míru podobnosti. Hodnota similar se zadává celočíselně a udává procentuální hodnotu. Konkrétněji je tato položka popsána dále. Pokud parametr není zadán je použita hodnota 90.
- -m mod - při zpracování vstupního souboru je možné nastavit mód zpracování, v závislosti na formátu dat, které jsou v něm uložené. Výchozí hodnota mod je řetězec IP, další možné hodnoty jsou arp nebo ICMP.
- -host ip_adresa - tímto parametrem se specifikuje IP adresa pc, které má být bráno jako cíl útoku. Pokud parametr není zadán je použita hodnota 169.254.29.18.

Konkrétní volání skriptu tedy vypadá například následovně.

```
python metrika.py -i ip\_output.txt -s 82 -m IP -host 169.254.29.18
```

Metriky, které jsou ve skriptu implementované, jsou pak zaměřeny na více různých oblastí. Jedná se jak o analýzu z hlediska času, tak získání informací z vlastností paketů a jejich datových částí.

Datum a čas útoku

Tento údaj má spíše statistickou vlastnost, slouží především v zařazení útoku do určitého období. Pokud by se jednalo o útok prováděný za účasti člověka, bylo by možné odvodit z času například časové pásmo, nicméně takováto analýza by byla velmi zkreslená. Zajímavější by tak mohlo být hledání určitých pravidelností v datech a časech útoku. Například při útoku snažícím se každou noc několik dní po sobě prolomit heslo k ftp přístupu a podobně. Výstup této metriky je řetězec obsahující datum a čas ve formátu rok-měsíc-den hodiny:minuty:vteřiny.

Délka útoku

Pro určení typu nebo závažnosti útoku však může poměrně značnou roli hrát délka útoku. Tu metrika určuje jako rozdíl mezi prvním a posledním paketem, kde jako zdrojová nebo cílová IP adresa figuruje IP adresa útočníka. Podle délky útoku lze odvodit zda se jedná o krátkodobou aktivitu, jako například scan, automatizovaný skript nebo krátký útok. Dlouhá délka útoku pak může indikovat například snahu útočníka najít nějaké konkrétní údaje, krádež dat a podobně. Výstup této metriky je řetězec obsahující čas ve formátu hodiny:minuty:vteřiny.

Časové intervaly mezi pakety

Podle rozestupů mezi jednotlivými pakety lze poměrně jednoduše vypozorovat, zda útok provádí předem připravený zautomatizovaný skript, či zda je přítomen člověk. Při této metrice se proto zkoumá časová mezera mezi jednotlivými přijatými pakety. Tato skutečnost je dobře patrná při porovnání paketů ze začátku zachyceného útoku, kdy byl útok prováděn automatizovaně. Jakmile byla vytvořena vzdálená konzole, se kterou se již pracoval člověk, časové prodlevy mezi jednotlivými pakety rapidně vzrostly. Takovouto statistiku může ovlivnit nastavení časové prodlevy ve skriptu, mezi jednotlivými příkazy resp. pakety. Výstup této metriky je hodnota boolean, 1 značí přítomnost operátora, 0 skript.

IP adresa

Zdrojová IP adresa, ze které přichází pakety je jedna ze základních specifikací útoku. Udává odkud byl útok veden a tím pádem poskytuje nejrychlejší možné řešení problému - zablokování této adresy. Tím je zamezeno útočníkovi v dalším přístupu. Vlastníkem zjištěné IP adresy může být buď přímo útočník, mnohem spíše se ale jedná o infikovaný počítač, který je pro útok použit. V každém případě je ale zablokování adresy ze které útok přišel nezbytné, aby bylo zabráněno dalšímu možnému útoku nebo jeho pokračování. S velkou pravděpodobností pak může stejný nebo podobný útok nastat v krátké době znovu, za použití jiné IP adresy. Výstup této metriky je řetězec, obsahující IP adresu v klasickém formátu xxx.xxx.xxx.xxx.

TTL

Hodnota TTL, tedy time to live, poskytuje další zajímavé informace o útoku. Tento údaj má každý odeslaný paket. Hodnota TTL se při odeslání paketu nastaví v závislosti na použitém operačním systému na určitou hodnotu. Například pro systém Microsoft Windows XP je to 128. Při průchodu směrovačem je tato hodnota snížena o 1 a jakmile TTL klesna až na nulu, je paket zahozen. Pokud k zahození paketu dojde, směrovač, který paket zahodil,

o tom informuje ICMP zprávou. Díky tomu je tak možné získat kompletní cestu, kterou paket urazil. Stejně tak můžeme určit i operační systém, ze kterého paket přišel. Výchozí hodnotu TTL získáme jako součet TTL, které je v paketu, který jsme zachytili a počtu směrovačů přes které prošel.

Prijate pakety

Důležité jsou rovněž informace o počtech příchozích paketů a právě na tyto statistiky je tato metrika zaměřena. Obsahuje několik částí, z nichž je každá zaměřená na určitou část této statistiky, jak je popsáno dále. Základním údajem z hlediska přijatých paketů je rozhodně jejich počet. Dále je vhodné rozlišit, které pakety byly datové a které byly přijaty pouze v rámci vytváření či ověřování spojení. Datové pakety obsahují, jak vyplývá z jejich názvu, data, která jsou přenášena mezi počítači. Právě tyto pakety často obsahují škodlivý kód. Pro kompletní analýzu je dobré určit i celkové množství dat v bytech, které bylo v datových paketech přijato. Podle určení paketů, můžeme dále rozlišovat dvě fáze útoku. Nejprve přicházejí pakety, které jsou směřovány na určitou službu, s cílem zneužít určitý druh zranitelnosti. Jakmile se toto podaří, začne probíhat vlastní škodlivá činnost. Zatímco zranitelnost existuje na systémových službách, škodlivá činnost je prováděna na jiné části systému. Tomu odpovídají i použité porty. Proto je další členění právě z hlediska portů. Zjišťuje se, kolik přijatých paketů z celkového počtu bylo cíleno na systémové služby (použitý port je menší než 1024). Stejně jako u přijatých paketů celkově tak i zde zjišťujeme počet datových paketů a stejně tak i jejich objem v bytech.

Odeslane pakety

Stejně jako u přijatých paketů jsou tyto statistiky zajímavé i u paketů odeslaných. Metriky u odeslaných paketů proto sledují stejné údaje jako v případě přijatých. Sleduje se celkový počet, kolik paketů z tohoto množství bylo datových a jaký byl jejich celkový objem v bytech. Stejně jako u příchozích paketů se sleduje kdy jsou využívány systémové služby, v tomto případě se počítají ty pakety, které ze sledovaného počítače odcházejí z portu menšího než 1024. U nich opět zjišťujeme, kolik z nich bylo datových a jejich objem v bytech.

Dvojice portů

Zároveň s IP adresou je port další základní specifikací. Z hlediska Sledovaného počítače udává místo nebo konkrétně službu, která na daném portu běží, na které se útok zaměřil. Na druhé straně pak udává port, který používá útočník. Výstup této metriky je celočíselná dvojice, port sledovaného počítače - port útočníka.

Identické datové pakety

Zkoumání unikátnosti příchozí paketů může rovněž poskytovat zajímavé výsledky. V případě, že se útočník snaží o zahlcení nebo přetížení systému, může dojít právě k této situaci. Jedním z příznaků takového útoku pak často bývá právě výskyt opakujících se paketů. Výstup této metriky je celočíselná hodnota, počet identických datových paketů.

Data popisující útok

Protože celkový objem paketů v jednom útoku může být značný, není možné ukládat všechny, které jsou zachyceny. Prvotní omezení tak spočívá ve výběru paketů zaměřených

na služby, tedy takových, u kterých je cílový port na sledovaném počítači nižší než 1024. Dále se pak provede další selekce, ukládají se pouze pakety, které jsou v rámci celého zachyceného přenosu unikátní. Podle parametru určujícího míru shody paketů (viz parametry při spuštění skriptu) se pak vyberou jen ty, které jsou svým způsobem unikátní. Při průchodu jednotlivými pakety se vždy zkontroluje seznam, zda už v něm podobná data nejsou. Pokud je podobnost menší než zadaný parametr, paket se přeskočí. Výstup této metriky je řetězec, obsahující datovou část paketu v hexadecimálním tvaru.

Výsledky z metrik jsou jednak vypsány na standardní výstup, zároveň jsou i uloženy do pomocných souborů, aby mohly být následně vloženy do databáze. K tomuto účelu se vytvoří textové soubory db_insert.txt, který obsahuje statistiky přijatých a odeslaných paketů, IP adresu, TTL a podobně. V dalším souboru db_insert_data.txt jsou pak uložena data popisující útok v hexadecimálním tvaru. Poslední soubor je pak db_insert_ports.txt, kam se ukládají dvojice použitých portů.

Jako vedlejší produkt při tvorbě metrik pro analýzu konkrétního útoku vznikla sada metrik pro analýzu kompletní síťové aktivity. Stejně jako u ostatních vytvořených součástí se i zde jedná o skripty v jazyce Python. Každý z nich je umístěna v samostatném souboru pojmenovaném metrika1.py až metrika5.py. Jejich princip je však takový, že z kompletního datového toku vyberou pouze pakety dle určité specifikace, např. z nejčastěji se vyskytující IP adresy.

8.3.3 db_insert.py

Třetí skript slouží ke vložení výsledků metrik do databázové struktury. V závislosti na použité databázi používá rozšíření MySQLdb, modul pro Python sloužící k připojení a práci s databází MySQL. Celá tato operace je záměrně umístěna v samostatném skriptu, protože právě toto rozšíření je vázáno na konkrétní použitou databázi. Pokud by bylo třeba používat jinou databázi, je možné nahradit pouze tento jeden skript.

Funkce toho skriptu je jednoduchá. Po otevření tří souborů s výsledky metrik prochází každý soubor a načítá data, která mají předem definovaný tvar. Ta jsou pak doplněna do sql dotazů a vložena do databáze. Po úspěšném vložení je vypsána informace o počtu vložených záznamů.

8.4 Výsledky analýzy

Při postupu dle uvedených příkladů použití programu WinDump a dále pak skriptů pro zpracování dumpu a aplikaci metrik dostaneme následující výsledek.

```
Datum utoku 2010-05-13 10:53:29.610006
Utok trval 0:1:7.460333 (h:m:s)
110 paketu prislo v rozmezi mensim nez 0.5 vteriny = skript
11 paketu prislo v rozmezi vetsim nez 0.5 vteriny = operator

IP: 169.254.90.204
TTL: 128

Prijate pakety
prijato celkem                : 68
```

```
prijatych paketu datovych     : 22
celkem prijatych bytu dat.    : 4409
pakety cilene na syst. sluzby : 26
datove pakety c. n. s.        : 21
bytu                          : 4169

Odeslane pakety
odeslano celkem               : 53
odeslanych paketu datovych    : 27
celkem odeslanych bytu dat.   : 6018
pakety cilene na syst. sluzby : 23
datove pakety c. n. s.        : 19
bytu                          : 2261

Byly pouzity tyto dvojice portu:
4444 - 1099
445 - 1100

Identickych prijimanych dat paketu 0
Prijimanych dat paketu c.n.s se shodou 82.0% : 14
Data popisujici utok:
450000302fa340008006ff49a9fe5acca9fe1d12044b115c7af0acd1000000007002400039
db0000020405b401010402

450000282fa540008006ff4fa9fe5acca9fe1d12044c01bd7af1a530dcaf5c095010447040
a50000000000000000

450000802fa740008006fef5a9fe5acca9fe1d12044c01bd7af1a530dcaf5c095018ffff1a
47000000000054ff534d427200000000180128000000000000000000000000009e540000
f900003100024c41

450001392fa940008006fe3aa9fe5acca9fe1d12044c01bd7af1a645dcaf5db75018fe51d0
4a00000000010dff534d427300000000180128000000000000000000000000009e540008
f9000cff000000df

450000872fac40008006fee9a9fe5acca9fe1d12044c01bd7af1a810dcaf5e6c5018fd9cb8
f000000000005bff534d42a2000000001801200000000000000000000000089e540108
f90018ff00000000

450001a62fb140008006fdc5a9fe5acca9fe1d12044c01bd7af1ab39dcaf601b5018fbed01
3e00000000017aff534d422f0000000018012000000000000000000000089e540108
f9000eff00000001

450000282fc040008006ff34a9fe5acca9fe1d12044c01bd7af1b579dcaf65055011fd746e
5a0000000000000000
```

Datum a čas útoku zde jednoznačně identifikuje okamžik kdy útok začal, tak jak bylo v metrikách navrženo. Z délky útoku, který se blíží jedné minutě pak můžeme usuzovat, že šlo

o krátký předem přichystaný útok. To odpovídá i skutečnosti, protože útok vznikal simulovaně a byl předem připravený, jak z části samotného zneužití zranitelnosti tak i následné činnosti po prolomení zabezpečení. Dle intervalů mezi pakety pak metrika určila kombinaci skriptu a člověka. I tato metrika potvrzuje skutečnost, protože ke zneužití zranitelnosti byl použit automatizovaný program, kdežto další činnost, která probíhala prostřednictvím vytvořené vzdálené konzole, již byla prováděna člověkem. Ta spočívala v průchodu adresářovou strukturou a vytvoření textového souboru na napadeném počítači.

Uvedená IP adresa odpovídá adrese druhého virtuálního počítače, který byl pro útok použit. Ze zjištěné hodnoty TTL můžeme jednak odvodit, že mezi útočníkem a sledovaným počítačem nebyl žádný směrovač, protože hodnota 128 je výchozí hodnota TTL nastavená systémy Windows XP nebo 2000.

Zajímavé jsou rovněž statistiky získané z počtu přijatých a odeslaných paketů. Celkový poměr paketů je téměř vyrovnaný - 68 přijatých oproti 53 odeslaným, což je však dáno samotným způsobem komunikace založeným na principu otázka - odpověď. Z hlediska datových paketů dokonce převažuje množství odeslaných, což je však způsobeno činností v poslední fázi útoku, kdy sledovaný počítač odesílá datové pakety vzdálené konzole. To ovlivňuje i množství dat v bytech, které je u odeslaných paketů opět vyšší. Pokud se ovšem podíváme na pakety, které jsou zaměřené na služby, tj. které byly na sledovaném počítači přijaty nebo odeslány z portu menšího než 1024, zjistíme, že tyto poměry již spíše odpovídají útoku typu buffer overflow. Přestože počty odeslaných a přijatých paketů jsou opět téměř vyrovnané a stejně tak počty datových paketů, při porovnání velikosti dat jasně převažují data přijatá.

Dvojice použitých portů zaznamenaly dvě různé aktivity. První z nich je komunikace mezi portem 4444 na sledovaném počítači a 1099 na počítači útočníka. To představuje jednak úvodní ověření dostupnosti portu 4444, aby právě na něm mohla vzniknout vzdálená konzole. Přes ni jsou pak v poslední fázi útoku přenášena data příkazů pro napadený počítač. Druhá dvojice portů 445 a 1100 představuje samotné zneužití zranitelnosti služby SMB (Server Message Block), která na tomto portu naslouchá.

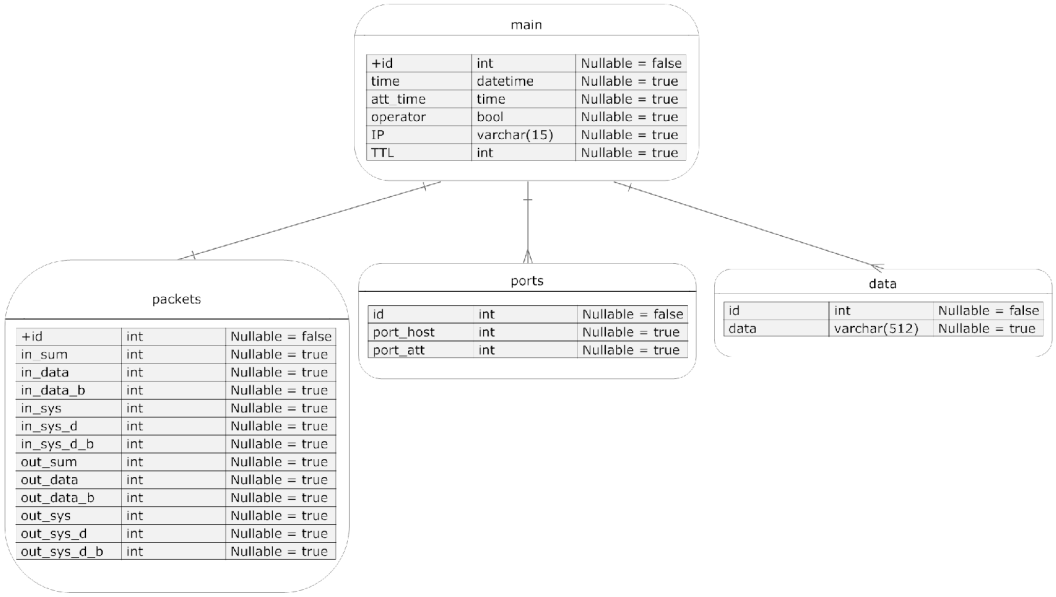
V celém datovém toku se pak nenašly žádné identické pakety, z čehož lze usuzovat, že se nejedná o útok zaměřený na znepřístupnění služby, který by právě výskyt totožných paketů indikoval. Z hlediska podobnosti příchozích paketů, které byly směřovány na systémovou službu, jich pak bylo 14 vyřazeno z důvodu podobnosti dle nastaveného kritéria. Na konci jsou pak uvedeny data z paketů, které požadovaným kritériím odpovídají a budou tudíž vloženy do databáze.

8.5 Datová struktura

Jak již bylo zmíněno, získaná data je nutné ukládat ve formě, ve které se s nimi bude moct dobře pracovat. Textové soubory jsou pro další analýzu těžkopádné, proto je mnohem vhodnější použít datovou strukturu, konkrétně relační databázi.

8.5.1 Datová struktura - návrh

Při návrhu struktury databáze se postupovalo dle dat, která je třeba ukládat. Pro hlavní zařazení dat, slouží tabulka main, ve které jsou základní údaje, podle kterých lze útok zařadit. To jsou čas útoku (pole time), IP adresa útočníka (pole IP) a TTL (pole TTL). K těm jsou pak přidány další údaje s nimi spojené - délka útoku (att_time) a přítomnost operátora



Obrázek 8.1: Návrh datové struktury

(operator). Jako primární klíč je použit identifikátor id. Právě přes tento identifikátor se v ostatních tabulkách vytváří vazby na tabulku main, pomocí cizího klíče.

V tabulce packets se ukládají statistiky o přijatých a odeslaných paketech. U přijatých paketů se ukládá jejich celkové množství (in_sum), počtu přijatých paketů, které byly datové (in_data) a velikosti těchto datových paketů v bytech (in_data_b). Dále jsou to pak pakety cílené na systémové služby. U nich se ukládá opět počet (in_sys), počet datových paketů v tomto množství (in_sys_d) a velikost dat v nich obsažených v bytech (in_sys_d_b). Obdobně jsou pak ukládány i informace o odeslaných paketech (pole mají v názvu out namísto in). Tyto statistiky paketů jsou pro každý útok specifické, každému útoku tak odpovídá jeden záznam. Přes cizí klíč id je tabulka navázána na předcházející tabulku main, což odpovídá vazbě 1:1.

Třetí tabulka ports obsahuje informace o použitých portech. Ty jsou vždy ve dvojici, v pořadí port na sledovaném počítači (pole port_host) a pak port na počítači útočníka (port_att). Pro navázání na tabulku main je opět použit cizí klíč id, typ vazby je 1:N.

Poslední tabulka je pojmenována data. Do ní se vkládají datové části vybraných paketů, které daný útok určitým způsobem popisují (pole data). Přes cizí klíč id se záznam naváže na tabulku main. Každému paketu tak odpovídá jeden záznam, vazba na tabulku main je tak 1:N.

Toto řešení je znázorněno v diagramu na obrázku 8.1 na straně 43.

Kapitola 9

Další použité technologie

V této kapitole jsou zmíněny další nástroje, které byly při řešení diplomové práce použity, ale jejich důležitost není taková, jako u nástrojů popsaných přímo v kapitole věnované příslušné problematice. Zároveň se jedná o nástroje, u kterých existují další možné varianty, které je možné použít. Výjimku tvoří nástroj Systrace, který z důvodů, které jsou u něj uvedeny použit nebyl.

9.1 Systrace

Užitečným nástrojem, který se rovněž zabývá otázkou bezpečnosti, je Systrace. Umožňuje sledovat a především blokovat přístup aplikací k systému. To spočívá v nastavování a vynucování příslušných zásad přístupu k systémovým voláním. Ty je možné sledovat a za běhu přidávat nové, pokud například některá aplikace začne provádět něco, co se nám nezamlouvá. Stejně tak lze monitorovat i provoz sítě. Původně tento program vznikl pro prostředí BSD, stejně tak je však dostupný na systémech Linux a Mac OS. [6] Pro prostředí Windows ale portovaná verze neexistuje. Přesto, že je je tento program uveden v zadání diplomové práce, pro její konkrétní zaměření, tj. provoz honeypotu na operačním systému Windows, ho není možné použít. Při provozu honeypotu na lokální virtuální síti však nemá toto omezení vážnější vliv.

9.2 Microsoft Virtual PC

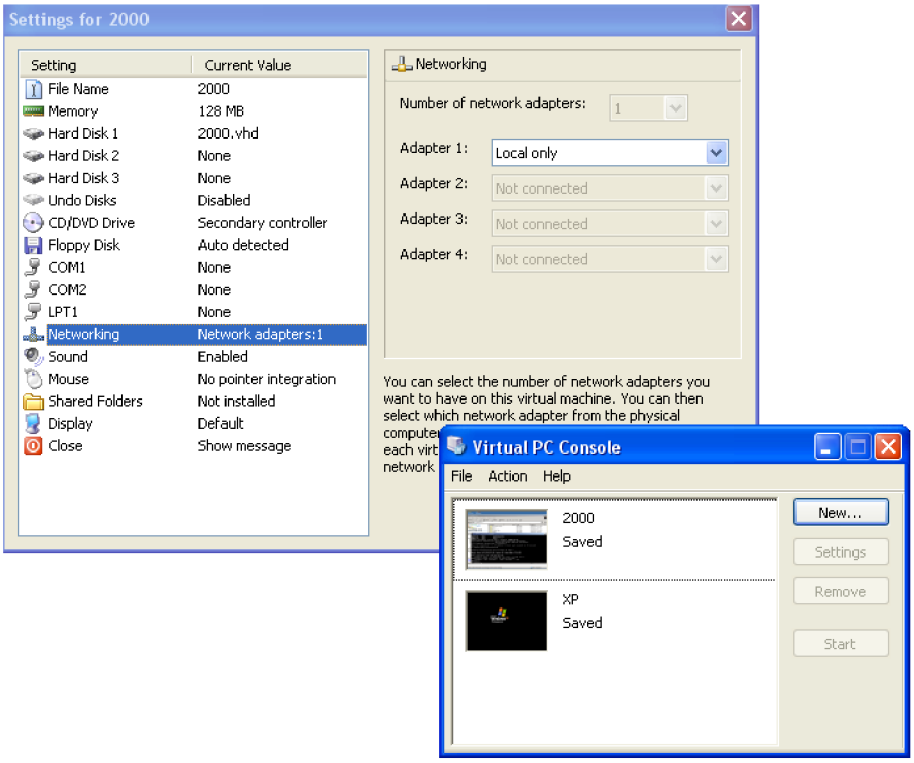
Jak bylo uvedeno v kapitole o použitých honeypotech, pro správnou funkci těchto programů, je nezbytné specifické prostředí. Při testování bylo nezbytné vyzkoušet takových prostředí několik. Z hlediska operačního systému jsou jeho změny na skutečném fyzickém PC zdlouhavé a problematické, pro testovací účely je proto mnohem vhodnější použít prostředí virtuální. Při jeho použití můžeme používat několik různých operačních systémů na jednom fyzickém PC, což je pro potřeby testování programu (v našem případě to byl honeypot) na různých operačních systémech, ideální.

Nástrojů pro vytvoření virtuálního počítače je celá řada. Jak již bylo zmíněno, mezi nejrozšířenější patří VMware, Microsoft Virtual PC, nebo z linuxových nástrojů např. Linux VKM. Z hlediska dostupnosti bylo použito Microsoft Virtual PC, které lze pro školní účely získat v rámci projektu MSDN AA.

Instalace programu je klasická, v samotném uživatelském rozhraní pak můžeme spravovat jednotlivé virtuální stroje. Při vytvoření nového stroje je pak nezbytné přidělit mu

určité množství paměti a diskového prostoru, pro vytvoření virtuálního disku, na se kterým bude stroj pracovat. Instalace operačního systému na takový stroj pak spočívá v přidělení fyzické mechaniky cd/dvd, přes kterou provedeme instalaci systému na přidělený virtuální oddíl. Z dalšího nastavení můžeme vytvořenému virtuálnímu PC přidělit i fyzickou disketovou mechaniku, COM nebo LPT port a síťové rozhraní. Právě nastavení sítě je z hlediska této práce nejpodstatnější, možná nastavení jsou dvě. Prvním z nich je použití sdílené NAT sítě s fyzickým PC, na kterém Virtual PC běží. Druhou možností je použití pouze lokální sítě v rámci Virtual PC, ve které budou pouze virtuální počítače.

Pro jednoduchý přenos souborů je pak možné doinstalovat Virtual Machine Network Services pro snadné přenášení souborů mezi virtuálním a fyzickým pc, na kterém Virtual PC běží.



Obrázek 9.1: Microsoft VirtualPC

9.3 Python

Python je interpretovaný programovací jazyk, který je využitelný pro širokou škálu aplikací. Společně s jazyky Perl nebo Ruby bývá navíc často označován jako jazyk skriptovací. Jednou z jeho nesporných výhod je nezávislost na platformě, k dispozici jsou verze pro všechny hlavní operační systémy současnosti - Windows, Linux/Unix i Mac. Zároveň je vyvíjen jako opensource, je tedy volně šiřitelný a distribuovatelný. [15]

Konkrétní verze použitá v tomto projektu byla 2.6, která je k dispozici ke stažení na oficiálních stránkách http://www.python.org/download/. Po nainstalování je vhodné doplnit systémovou proměnnou PATH o cestu k interpretu tohoto jazyka, aby jej bylo možné jednoduše používat z příkazové řádky.

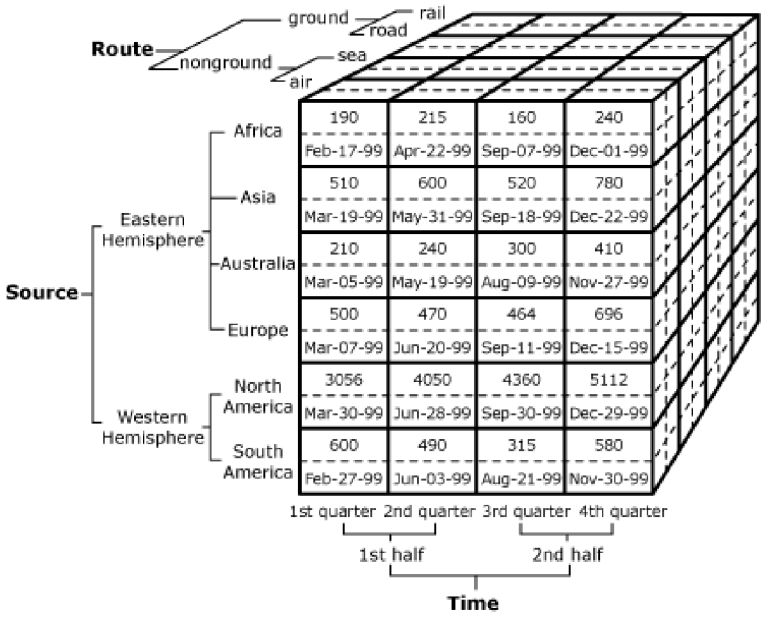
Důvody k použití tohoto jazyka byly čistě praktické, z důvodu předchozí krátké zkušenosti s tímto jazykem. Z hlediska nároků ze strany této práce je však možné použít i jiný obdobný nástroj, například Perl, Ruby, případně C nebo C++.

9.4 Datové sklady

Tato technologie představuje specifický druh databáze, primárně určený k podpoře rozhodování. Na rozdíl od klasických relačních databází se v datovém skladu uchovává řádově mnohem větší množství dat, čímž vyplývá i zaměření, které se na rozdíl od každodenních operací nebo zpracování transakcí, které známe u relačních databází, přesunulo k modelování a analýze dat. Stejně tak se podstatně rozšířil časový horizont, u datových skladů klidně i na 5-10 let. Nejčastější operací je tím pádem čtení a ukládání, na které jsou datové sklady optimalizovány.

Důležitá je především kvalita uchovávaných dat - před vložením se data čistí. Čištění znamená hledání a odstranění případných nekonzistencí, jako jsou například chybějící údaje, odchýlené hodnoty, nejasná či chybná zadání a podobně. Stejně tak se v datovém skladu často spojují heterogenní data, z několika různých zdrojů. Nejčastějším způsobem uložení je pak forma datové kostky (viz obrázek), kde je možné pracovat s jednotlivými rovinami a tím vybírat jen určitá data, která nás zajímají.

Nasazení a použití datových skladů je nicméně často náročné a vyžaduje komplexnější návrh. Pro potřeby této práce se proto ukázalo jako dostačující použití relační databáze.



Obrázek 9.2: Příklad datové kostky
[9]

9.5 MySQL

Jedním z řady dnes používaných relačních databází je MySQL. Celý systém je vyvíjen jako opensource a je opět multiplatformní. Tato kombinace z něj dělá v dnešní době jednu z nejčastěji používaných databází.

Kromě možnosti stažení samostatné databáze je mnohem praktičtější použít již připravený balík označený jako WAMP, což znamená Windows, Apache, MySQL, PHP. Přestože pro účely této práce není celá tato sestava potřeba, obsahuje další užitečné nástroje, jako například phpMyAdmin, určený pro správu databáze.

MySQL bylo pro potřeby ukládání výsledků z metrik do datové struktury vybráno z široké nabídky různých relačních databází. Důvodů, které k tomu vedly, je více. Hlavním z nich je především kombinace s modulem pro Python nazvaným MySQLdb, který právě s touto relační databází pracuje. Volná dostupnost a flexibilita použití z hlediska operačního systému, dostatečné množství dobře zpracované dokumentace a předchozí zkušenosti jsou pak další faktory, které měly vliv na výběr použité databáze.

Kapitola 10

Závěr

Již během semestrálního projektu byl vytvořen úvod k této diplomové práci. Ten se zakládá v první části práce na úvaze, zhodnocení a popisu současného stavu problematiky síťových útoků, hledáním možných způsobů řešení a motivace pro tuto práci. Další část práce se zabývá problematikou a mapováním typů útoků, které jsou v dnešní době aktuální. Následující kapitola se již zabývá systémy obrany proti síťovému útoku. Po obecném rozdělení a popisu jednotlivých přístupů následuje část o systémech honeypot - tedy systému klamných cílů. Popisuje jejich základní rozdělení, popis činnosti, principy fungování a zároveň možná rizika vznikající při jejich použití. Všechny tyto části jsou zaměřeny spíše teoreticky a slouží k přiblížení a zmapování zkoumané a řešené problematiky.

Další kapitola a její podkapitoly se již zabývají konkrétními programy, na které je práce zaměřena. V první fázi se jedná především o seznámení se s jejich činností a používáním. Ze zadání práce vyplývá hlavní zaměření na nástroj Honeyd, kterému je věnováno více prostoru. Přesto, že je již staršího data, je stále vnímán jako velmi dobrý virtuální honeypot, především díky dostupnosti a široké škále nastavení. Podle upřesnění zadání se pak práce zaměřila na použití honeypotů na Windows. Proto je použita portovaná verze WinHoneyd, která je právě pro tento systém určena. Při používání tohoto nástroje však vzniklo množství problémů, především s použitelností a vůbec jeho spuštěním, v závislosti na použitém operačním systému. Z toho důvodu je v kapitole Použité honeypoty popsán i honeypot LaBrea. Jeho popis je záměrně strukturován stejně, jako v předchozím případě. Rovněž použití tohoto honeypotu se neobešlo bez komplikací. Nakonec se úspěšně podařilo spustit a rozběhnout původně testovaný WinHoneyd, nicméně dosažené výsledky u honeypotu LaBrea jsou zde rovněž uvedeny.

Veškeré tyto komplikace a problematika s použitím v závislosti na operačním systému bohužel zabránily v nasazení v reálném provozu. Úspěšně se tak WinHoneyd podařilo otestovat pouze v rámci lokální virtuální sítě. Toto testování bylo úspěšné, honeypot správně zaznamenával pokusy o příchozí komunikaci, scanování portů a další aktivity. Z důvodu existence zmíněných komplikací je dále popsána problematika kolem takzvaných raw soketů, které právě s těmito problémy na systémech Windows souvisí. Součástí kapitoly zabývající se opensource honeypoty na systémech Windows je rovněž shrnutí zjištěných chování a použitelnosti v závislosti na operačním systému. Stejně tak byli při řešení této části kontaktováni lidé, kteří se na vývoji WinHoneyd podíleli nebo podílejí, případně autoři knih, kde jsou tato zařízení popsána. Emaily, které od nich jako odpovědi na položené otázky dorazily, jsou uvedeny v příloze.

Po použití honeypotů bylo dalším krokem zachycení a analýza útoku z hlediska jednotlivých paketů, s použitím programů sledujících datový tok. Na již dříve použité virtuální

lokální síti se tak provedl útok pomocí programu pro penetrační testování, který byl na napadeném počítači monitorován. Jako prováděný útok bylo vybráno zneužití zranitelnosti MS06_40. To je podrobněji rozepsáno z pohledu jednotlivých fází a přenášených paketů. Pro potřeby monitorování byl pak použit program WinDump, jehož výsledky bylo nutné dále upravit a zpracovat. K tomuto účely byly vytvořeny skripty v jazyce Python, které se na získané výsledky postupně aplikují. Nejprve je zpracován výstup z programu WinDump, jsou vybrány pouze určité údaje a ty jsou uloženy ve vhodném textovém formátu. Druhým je pak skript obsahující metriky pro analýzu těchto dat. Prozatím probíhá veškeré zpracování a ukládání dat pouze na úrovni textových souborů. Ve třetím skriptu se proto již operuje s relační databází, do které se výsledky analýzy ukládají. V závěru kapitoly Analýza síťového útoku jsou pak zjištěné výsledky zhodnoceny.

V závěru práce jsou pak uvedeny nástroje, které byly při její tvorbě použity, ale jejichž důležitost nebyla zcela prioritní. Patří sem například programy pro vytvoření virtuálních počítačů, použitý skriptovací jazyk či použitá relační databáze. Použití této databáze a skriptovacího jazyka bylo svým způsobem propojené díky jejich návaznosti. Obecně je však možné použít i jiné technologie. Dále jsou zde uveden nástroj Systrace, který je určen pro využití v systému Linux a přesto, že je uveden v zadání práce, nebyl při jejím průběhu z důvodu nedostupnosti na systému Windows použit. Obdobně je tomu s datovými sklady, které by pro velký a komplexní systém dat ze zachycených útoků byly vhodné, nicméně pro použití v menším měřítku je dostačující relační databáze.

Z hlediska dalšího pokračování práce by mělo být možné využít znalosti o použití honeypotů na systémech Windows a tím se vyvarovat problémům, které během této práce nastaly. Zopakování testů, které během ní byly provedeny by tak mělo být, při dodržení postupů zde popsaných, již velmi rychlé. Na druhou stranu je však další využití těchto nástrojů na současných systémech dosti problematické, jak je popsáno v kapitole Open-source honeypoty v prostředí Windows. Stručně shrnuto, bez zásadních změn je nemožné tyto zkoumané systémy, které představují prakticky jediné volně dostupné implementace honeypotů určené pro Windows, používat na systému vydaném po roce 2005 případně s aktualizacemi novějšími než z tohoto roku. V návaznosti na další část, tedy na analýzu útoku je pak určitě vhodné doplnit další metriky a rozšířit možnosti zpracování a hlavně dalšího použití dat.

Pro sazbu práce bylo použito prostředí LaTeX. Psaní přímo v systému pro sazbu není příliš pohodlné, proto byl použit poměrně častý princip tvorby textu v textovém editoru, konkrétně Writer z kancelářského balíku OpenOffice, a jeho následné zkopírování a zformátování do LaTeXu. Původní dokument i zdrojové kódy jsou k dispozici na přiloženém cd. Šablony LaTeXu pak výsledný text lehce zkracují, z vlastností dokumentu v OpenOffice zjistíme počet znaků práce kolem 100 tisíc, což při normostraně 1800 znaků odpovídá rozsahu kolem 55 stran čistého textu bez obrázků.

Celkově můžeme tuto diplomovou práce prohlásit za úspěšnou, bylo postupováno podle jednotlivých bodů zadaní, které se, byť s problémy, podařilo vždy alespoň částečně splnit.

Literatura

[1] Allen Harper, C. E., Shon Harris: *Hacking - manuál hackera*. Grada, 2008, iSBN 978-8024713465.

[2] Barker, G.: Microsoft May Have Been Target of Lovebug. *The Age*, 2000.

[3] Hanáček, P.: *přednáška - Hrozby, slabá místa*. VUT FIT, [cit. 2010-01-03].

[4] Niels Provos, Thorsten Holz: *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addison-Wesley, 2007, iSBN 978-0321336323.

[5] Roger A. Grimes: *Honeypots for Windows*. Apress, 2005, iSBN 978-1590593356.

[6] WWW stránky: Interactive Policy Generation for System Calls. `http://www.citi.umich.edu/u/provos/systrace/`, [cit. 2009-12-21].

[7] WWW stránky: VirusList. `http://www.viruslist.com/`, [cit. 2009-12-21].

[8] WWW stránky: Email Metrics Program:The Network Operators Perspective. `http://www.maawg.org/about/MAAWG20072Q_Metrics_Report.pdf`, [cit. 2010-01-03].

[9] WWW stránky: Logical Architecture Overview (Analysis Services - Multidimensional Data). `http://technet.microsoft.com/en-us/library/ms174587.aspx`, [cit. 2010-01-03].

[10] WWW stránky: TCP/IP Raw Sockets. `http://msdn.microsoft.com/en-us/library/ms740548(VS.85).aspx`, [cit. 2010-05-14].

[11] WWW stránky: LaBrea-Intro History. `http://labrea.sourceforge.net/Intro-History.html`, [cit. 2010-05-17].

[12] WWW stránky: LibnetNT. `http://www.securityfocus.com/tools/1559`, [cit. 2010-05-17].

[13] WWW stránky: Microsoft Security Bulletin MS06-040. `http://www.microsoft.com/technet/security/bulletin/ms06-040.mspx`, [cit. 2010-05-18].

[14] WWW stránky: Microsoft Windows Remote Code Execution (Exploit, MS06-040). `http://www.securiteam.com/exploits/5DP0L00JFM.html`, [cit. 2010-05-18].

[15] WWW stránky: About Python. `http://www.python.org/about/`, [cit. 2010-05-21].

Dodatek A

Obsah CD

Skripty s implementovanými metrikami

Textové soubory s výstupy programů a metrik

Instalační soubory použitých programů

Zdrojové texty

Dodatek B

Emaily

B.1 Email od firmy netVigilance

Hello Jan,
    WinHoneyD will not work with XP SP2. Microsoft Patch MS05-019/ KB893066 prevents sending TCP via Raw Sockets (explanation from Microsoft). Microsoft recommended solution: 'If you frequently use tools that send packets over raw sockets, we suggest that you use Microsoft Windows Server 2003. Windows Server 2003 does not restrict traffic over raw sockets'. So WinHoneyd will NOT work on XP SP2 or SP3, I have only tested on windows XP SP0 and SP1! Assume Vista, 7, 2003 etc will not work.

    Honeyd in General and WinHoneyd in particular is very Hard to get to work. After literally 3 years of work on the project of making a good and stable-easy to use honeyd for windows (May 2005 to April 2008) we gave up and shelved it, we are now using Linux honeyd instead for production.

Make sure you visit and read these links
http://www.honeyd.org/general.php
http://www.honeyd.org/configuration.php
http://www.honeyd.org/faq.php

Regards,
netVigilance Support

B.2 Email od Rogera A. Grimese

Unfortunately, I've given up on making Honeyd work on Windows. The Honeyd development team has done little over the last 5 years to fix the bugs and get it working. The problems really started to develop during the writing of the book, but instead of getting better, they got worse. I only running Honeyd on Linux and BSD (I use OpenBSD, myself, but it's a bit tough to work with if you haven't worked with Linux/BSD before).

I'd also recommend using the demo copy of Kfsensor (www.keyfocus.net).

There is no better honeypot software on the market.

Thanks for writing.

Roger