

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 HONEYPOTŮ

DIPLOMOVÁ PRÁCE

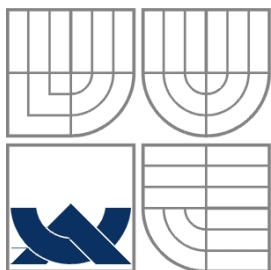
MASTER'S THESIS

AUTOR PRÁCE

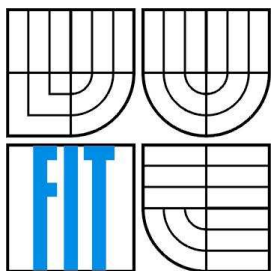
AUTHOR

Bc. JOSEF GALETKA

BRNO 2010



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 HONEYPOTŮ

NETWORK ATTACK ANALYSIS USING HONEYPOTS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JOSEF GALETKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL DROZD

BRNO 2010

Abstrakt

Práce se zabývá bezpečností počítačových sítí za využití technologie honeypotů, jakožto nástroje účelných nástrah pro útočníky. Blíže popisuje základní myšlenky a výhody a nevýhody tohoto konceptu. Zaměřuje se na honeypot s nízkou interakce Honeyd, jeho funkčnost a možná rozšíření. V rámci praktické části je zde popsán princip implementace rozšiřujících servisních skriptů Honeyd v podobě simulace chování červa Conficker. Dále se zabývá vytvořením automatizovaného skriptu analýzy, zpracování útoků a vyhodnocení dat, získaných v průběhu skutečného nasazení honeypotu v síti Internet.

Abstract

This text deals with computer network security using honeypot technology, as a tool of intentional trap for attackers. It closely describes basic thoughts, together with advantages and disadvantages of this concept. The main aim is a low interaction honeypot Honeyd, its functionality and possible extensional features. As a practical part of the text there is a description of principles of implementation Honeyd service scripts, which are represented as a simulation of behavior of computer worm Conficker. Further it describes creation of automated script used for analysis and processing of gathered data, captured during actual deployment of Honeyd in Internet network.

Klíčová slova

Honeypot, Honeyd, síťová bezpečnost, analýza útoků, malware, servisní skript, Conficker, automatizované zpracování

Keywords

Honeypot, Honeyd, network security, attack analysis, malware, service script, Conficker, automated process

Citace

Galetka Josef: Analýza síťových útoků pomocí honeypotů, diplomová práce, Brno, FIT VUT v Brně, 2010

Zadání diplomové práce

Řešitel: **Galetka Josef, Bc.**

Obor: Informační systémy

Téma: **Analýza síťových útoků pomocí honeypotů**
Network Attack Analysis Using Honeypots

Kategorie: Bezpečnost

Pokyny:

1. Seznamte se s nástroji typu honeypot.
2. Vyberte vhodné implementace známých nástrojů typu honeypot a detailně prostudujte jejich vlastnosti.
3. Po dohodě s vedoucím navrhnete sadu testů pro analýzu dat ze síťových útoků v lokální síti a na internetu. Jejich výsledky uložte do vhodné datové struktury.
4. Proveďte základní analytické operace nad nalezenými útoky (definujte vlastní metriky) a pokuste se na jejich základě vytvořit komplexnější analýzu průběhu útoku.
5. Výsledky řešení diskutujte a navrhnete možná rozšíření.

Literatura:

- N. Provos, T. Holz. Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Addison-Wesley, 2007. 440 p. ISBN 978-0321336323.
- C. H. Malin, E. Casey. J. M. Aquilina. Malware Forensics: Investigating and Analyzing Malicious Code. Syngress, 2008. 592 p. ISBN 978-1597492683.
- The Honeynet project. www.honeynet.org/project [cit. 2009-10-01].

Při obhajobě semestrální části diplomového projektu je požadováno:

- 1. a 2. bod zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese

<http://www.fit.vutbr.cz/info/szz/>

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Drozd Michal, Ing., UITS FIT VUT**

Datum zadání: 21. září 2009

Datum odevzdání: 26. května 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
602 00 Brno, Božetěchova 2

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Analýza síťových útoků pomocí honeypotů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Michala Drozda. Další informace mi poskytl Ing. Petr Chmelař.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....

Josef Galetka

16. 5. 2010

Poděkování

Chtěl bych poděkovat Ing. Michalu Drozdovi a Ing. Petru Chmelařovi za poskytnuté informace, týkající se trendů síťové bezpečnosti a užití honeypotů. Současně jim děkuji za veškeré informace a pomoc, kterou mi poskytli při tvorbě této práce. Tímto chci poděkovat i panu Tomášovi Vobrubovi, který pomohl při nasazení a konfiguraci Honeyd na vyhrazeném serveru a společnosti AEC, která umožnila umístění tohoto serveru v rámci své sítě. Děkuji také své přítelkyni Vendule Pšeničkové za trpělivost a podporu.

© Josef Galetka, 2010

Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Obsah.....	1
1 Úvod.....	4
1.1 Stanovení cílů práce.....	5
1.2 Struktura práce.....	5
2 Malware a jeho současné trendy	7
2.1 Typy malware	7
2.1.1 Viry	7
2.1.2 Počítačové červi	7
2.1.3 Trojské koně	8
2.1.4 Rootkity	8
2.1.5 Spyware	8
2.2 Současné trendy malware	8
2.3 Conficker	9
2.3.1 Zranitelnost MS08-067	10
3 Technologie honeypotů.....	12
3.1 Úvod do problematiky honeypotů	12
3.2 Důvody nasazení.....	12
3.3 Přínos a význam honeypotů.....	13
3.3.1 Zero-day malware	14
3.4 Interakce honeypotů.....	15
3.5 Další typologie honeypotů.....	16
3.5.1 Fyzické a virtuální honeypoty.....	16
3.5.2 Bezdrátové honeypoty	16
3.5.3 Klientské honeypoty	16
3.6 Legální aspekty	17
4 Honeypoty v praxi.....	19
4.1 Honeynety (sítě honeypotů).....	19
4.2 Honeypoty s nízkou mírou interakce	20
4.3 Honeypoty s vysokou mírou interakce	23
4.4 Volba honeypotu pro praktickou část	25
5 Honeyd.....	26
5.1 Přehled konstrukčního řešení Honeyd	27
5.2 Emulované služby.....	28
5.3 Základní konfigurace	29

5.4	Evidence uskutečněných aktivit.....	31
5.5	Rozšiřující a podpůrné nástroje Honeyd.....	33
5.5.1	Honeydstats.....	33
5.5.2	Honeydctl.....	34
5.5.3	Honeycomb.....	34
5.5.4	Honeydsum.....	35
6	Emulace Confickeru v Honeyd.....	36
6.1	Detekce zranitelnosti nástrojem nmap.....	36
6.2	Postup tvorby servisního skriptu.....	37
7	Automatizovaný nástroj detekce útoků.....	42
7.1	Požadavky automatizace a výstupů.....	42
7.2	Podpůrné implementační nástroje.....	42
7.2.1	Perl.....	42
7.2.2	Tcpdump.....	43
7.2.3	Tcpslice.....	43
7.2.4	Tshark.....	44
7.2.5	Tcptrace.....	44
7.2.6	Snort.....	44
7.2.7	MySQL.....	44
7.3	Omezující prvky použitých nástrojů.....	45
7.4	Popis implementace skriptu.....	46
7.5	Výstupní metriky.....	47
7.5.1	Čas počátku spojení (<i>timeStart</i> , <i>timeStartMsecs</i>).....	49
7.5.2	Čas konce spojení (<i>timeEnd</i> , <i>timeEndMsecs</i>).....	49
7.5.3	Délka trvání spojení (<i>duration</i> , <i>durationMsecs</i>).....	49
7.5.4	Komunikační protokol (<i>proto</i>).....	50
7.5.5	Zdrojová IP adresa (<i>srcIp</i>).....	52
7.5.6	Zdrojový port (<i>srcPort</i>).....	52
7.5.7	Cílová IP adresa (<i>dstIp</i>).....	52
7.5.8	Cílový port (<i>dstPort</i>).....	53
7.5.9	Stav spojení (<i>connComplete</i>).....	53
7.5.10	Celkový počet paketů (<i>pckCountTotal</i>).....	53
7.5.11	Množství unikátních bytů (<i>pckBytesUnique</i>).....	54
7.5.12	Počet datových paketů (<i>pckCountData</i>).....	54
7.5.13	Počet opakované zaslaných paketů (<i>pckCountRexmt</i>).....	54
7.5.14	Počet out-of-order paketů (<i>pckCountOutorder</i>).....	54
7.5.15	Maximální velikost segmentu (<i>mssReq</i>).....	54

7.5.16	Maximální doba nečinnosti mezi pakety (<i>idletimeMax</i>).....	56
7.5.17	Počet zachycených RTT vzorků (<i>rttSamplesCount</i>).....	56
7.5.18	Hodnota RTT z TCP 3-Way Hand-Shake (<i>rtt3whs</i>).....	56
7.5.19	Počet full size segment RTT vzorků (<i>rttFullSamplesCount</i>).....	57
7.5.20	Průměrná přenosová rychlost (<i>throughput</i>)	57
7.5.21	Procentuální vyjádření ztrátovosti paketů (<i>packetloss</i>)	57
7.5.22	Odhadovaná verze OS zdrojové stanice (<i>os</i>)	57
7.5.23	Hrubý datový tok spojení (<i>data</i>).....	58
7.5.24	Výstupní detekce nástroje snort (<i>snort</i>).....	58
7.6	Struktura uložení metrik v databázi	59
7.7	Zhodnocení implementace	60
8	Sběr a vyhodnocení získaných dat	61
8.1	Konfigurace a požadavky systému	61
8.2	Statistické výsledky sběru.....	63
8.2.1	Počty spojení dle typu protokolu	63
8.2.2	Počet spojení na jednotlivé honeypoty	64
8.2.3	Rozložení spojení v průběhu dne.....	64
8.2.4	Operační systémy útočníků.....	65
8.2.5	Cílové porty zájmu útočníků.....	65
8.2.6	Ostatní statistiky	67
8.3	Vyhodnocení získaných výsledků	67
9	Závěr	69

1 Úvod

V nedávných letech došlo k růstu bezpečnostních incidentů v síti Internet, které celosvětově způsobily jejím uživatelům značné problémy. Velkou roli na tom hraje rozšíření počítačových virů, červů a dalšího nežádoucího software. S pokrokem v IT a rozmachem Internetu dochází ruku v ruce i k růstu počítačové kriminality a značných ekonomických škod, které tím vznikají.

V rámci udržitelnosti bezpečnosti počítačových systému a zamezení možného globálního kolapsu je potřeba hrozbám předcházet a vytvářet dostatečně schopné nástroje detekce škodlivých kódů a systémy odpovídající reakce. Jako jedna ze schopných technologií boje proti novým hrozbám se jeví honeypoty. Honeypoty jsou specializované počítačové systémy sloužící jako účelné návnady pro přilákání útočníků a odhalování dosud neznámých počítačových hrozeb a technik bezpečnostních útoků.

Tato práce se zabývá možnostmi automatizovaného sledování a vyhodnocování síťové komunikace a potenciálních útoků za využití honeypotů. Práce je zaměřena na popis jejich technologie, způsoby využití, výhodami a nevýhodami a dalšími elementy spjatými s tímto tématem. Zaměříme se na soudobé hrozby a trendy v oblasti počítačové bezpečnosti a nežádoucích programů označovaných pojmem malware.

Jako ukázkový představitel nám poslouží honeypot s nízkou mírou interakce Honeyd, přičemž střed zájmu bude soustředěn na jeho funkčnost a možnosti reálného nasazení, s ohledem na schopnost představovat zajímavý cíl zájmu útočníků s využitím emulovaných služeb. Součástí práce bude vytvoření automatizovaného nástroje, který bude vyhodnocovat uskutečněná spojení a analyzovat jejich obsah. Na základě vyhodnocení budou generovány výstupní metriky, které budou sloužit k popisu a identifikaci možného útoku. Dojde k nasazení honeypotu v síti Internet a sběru dat, která v závěrečné fázi poslouží pro podrobnější vyhodnocení získaných poznatků.

1.1 Stanovení cílů práce

K hlavním cílům, které byly stanoveny jako očekávaný výstup této práce patří zejména:

- Analýza trendů škodlivého software, jeho kategorizace a zaměření na červa Conficker.
- Seznámení se s technologií honeypotů jako účelných nástrah pro detekci a analýzu síťových útoků. Poskytnout přehled o jejich hlavních funkčních možnostech, přínosech, typologii, výhodách a nevýhodách využití této technologie.
- Uvést přehled dostupných řešení, jejich hlavní parametry a možné způsoby nasazení v praxi.
- Blíže se seznámit s honeypotem Honeyd. Analyzovat požadavky spojené s nasazením honeypotu, možnosti konfigurace a škálu nabízených služeb, zhodnotit možnosti využití jako součást bezpečnostních síťových mechanismů dnešní doby.
- Vytvořit automatizovaný nástroj, který bude sbírat a vyhodnocovat informace o průběhu prováděných spojení a útoků na honeypot, ověřit možné využití dostupných podpůrných prostředků sběru a analýzy dat. Na základě získaných údajů generovat výstupní metriky pro budoucí zpracování a evidenci.
- Provést nasazení honeypotu do sítě Internet, kde bude sbírat informace o současných hrozbách a nebezpečích, ze strany útočníků. Za pomoci vytvořeného automatizovaného nástroje analýzy zpracovat a provést vyhodnocení nad získanými daty.

1.2 Struktura práce

První kapitolu tvoří úvod a identifikace očekávaných cílů. Ve druhé kapitole se blíže seznámíme s pojmem malware, typologií škodlivého software a zaměříme se na současný stav počítačové kriminality.

Kapitola třetí nás blíže seznámí s celkovým konceptem technologie honeypot. Ujasníme si, čím je specifická, jaké přináší inovace vůči jiným bezpečnostním mechanismům a v čem jsou její největší klady a úskalí.

Čtvrtá kapitola nás provede dostupnými řešeními honeypotů dle dělení podle dané typologie. Uvedeme si příznačné znaky jednotlivých produktů a jejich hlavní rozdíly oproti konkurenčním řešením.

Kapitola pátá popisuje honeypot s nízkou interakce Honeyd, který bude jako stěžejní část využit pro praktickou část práce. Seznámíme se se samotným funkčním schématem honeypotu, jeho komponentami, funkcími a možnostmi rozšíření, které skýtá.

V kapitole šesté, jako jedné z praktických částí spjatých s diplomovou prací, je popsána jedna ze současných bezpečnostních hrozeb v podobě červa Conficker, který se šířil po Internetu a napadl velkou řadu počítačů. Zaměříme se na vývoj servisního skriptu Honeyd, který nám umožní emulovat odezvy systému napadeného právě tímto virem.

Kapitola sedmá popisuje principy a cíle automatizovaného nástroje pro sběr a analýzu dat síťových útoků. Jedná se o nástroj, který bude vyhodnocovat uskutečněná spojení s dostupnými honeypoty, provádět rozbor zachycených dat a vytvářet výstupní metriky popisující bezpečnostní incident pro možnost hlubšího zkoumání a vyhodnocení. Je zde popsán princip implementace tohoto nástroje, využití podpůrné programy, jejich omezení a možnosti výsledného výstupu v podobě metrik.

V osmé kapitole zhodnotíme dosažené výsledky a provedeme rozbor zpracovaných dat, která byla získána v rámci nasazení Honeyd do produkční sítě v rámci Internetu. Poslední devátá kapitola tvoří závěrečné shrnutí práce.

2 Malware a jeho současné trendy

Hlavním cílem práce je analýza síťových útoků, zmapování aktuálních trendů a možnost automatizovaného vyhodnocení údajů o dostupných hrozbách. V této kapitole bude popsáno, jakým směrem jsou orientovány nynější počítačové útoky, uvedena kategorizace malware se zaměřením na hlavní představitele současné scény.

2.1 Typy malware

Malware je zkratkou anglického malicious software. Tímto termínem označujeme veškerý škodlivý software, který je vytvořen za účelem narušení bezpečnosti a průniku do počítačového systému bez souhlasu a vědomí uživatele. Jedná se o nejobecnější pojem používaný v počítačové terminologii, který označuje všechny formy nežádoucího, nebezpečného a škodlivého kódu, který je vytvářen s úmyslem neoprávněného získání přístupu k počítači, krádeži nebo poškození dat, které je většinou prováděno za účelem vytvoření škody nebo vlastního obohacení útočníka. S ohledem na to, že existuje velká škála možného členění malware do kategorií, přičemž mezi nimi neexistují přesně ohraničené hranice, zmíníme pouze některé ze základních skupin.

2.1.1 Viry

Počítačový virus je program, který umožňuje vlastní kopírování sebe sama a infikování počítače. Pojmem virus bývají často laickou veřejností ne zcela správně označováni i zástupci ostatních kategorií malware, jako jsou červi, trojské koně a podobně. Za účelem vlastní replikace musí mít virus možnost spouštění a zápisu do paměti. Z tohoto důvodu se včleňuje do obsahu spustitelných souborů a tváří se jako jejich legitimní součást. Jakmile dojde ke spuštění hostitelského souboru, je současně s ním aktivován i virus.

2.1.2 Počítačové červi

Počítačový červ (ang. worm) je program s vlastní schopností replikace. Hlavním rozdílem oproti virům je fakt, že nepotřebuje žádné další soubory, aby se mohl šířit. Ve většině případů nepotřebuje ani prvotní uživatelskou akci k vlastnímu spuštění, ale využívá zranitelností v systému a přidružených službách. S využitím sítě rozesílá svoje kopie do ostatních nechráněných a dostatečně nezabezpečených uzlů. K šíření často používá email, IRC, ICQ kanály a další. Červi téměř vždy způsobují škody, alespoň co se týče neoprávněného plýtvání přenosového pásma komunikační linky, kdežto viry nejčastěji narušují a modifikují soubory v počítači.

2.1.3 Trojské koně

V širším pojetí chápeme trojský kůň jako program, který pod záminkou určité funkcionality naláká uživatele k jeho spuštění, přičemž ukrývá zranitelný kód. Ten okamžitě začne provádět svou činnost, což vede k nežádoucím a neočekávaným stavům systému, jako je mazání souborů a instalace dalších škodlivých programů. Trojské koně neboli trojany upravují chování systému a umožňují vzdálené ovládání stanice útočníkem. Často slouží jako přenašeče (ang. droppers) a napomáhají dalšímu šíření červů a podobně.

2.1.4 Rootkity

Hlavní známkou rootkitů je používání pokročilých technik, které spočívají modifikaci součástí operačního systému, přičemž hlavním cílem je snaha o maximální ukrytí a omezení možné detekce. Rootkity se snaží maskovat před uživatelem a zakrývat vlastní běžící procesy a příslušné soubory na disku. Rootkitem byla původně označována sada nástrojů, které byly používány při ručních typech útoků na Unixové systémy, kde cílem bylo získání administrátorských práv.

2.1.5 Spyware

Spyware je typem malware, který je instalován v počítači a jeho hlavním úkolem je sběr informací o uživateli bez jeho vědomí. K jeho známým zástupcům patří tzv. keyloggers, které zaznamenávají kompletní posloupnosti stisknutých kláves. Spyware sbírá a odesílá citlivé informace, které mohou být údaji o platebních kartách, přístupových heslech k účtům a další citlivá data. Často slouží k vyhodnocování chování uživatele v rámci Internetu, jeho navštívených stránek, vyhledávacích frází a jiné. Tyto informace jsou pak používány pro potřeby reklamy a různého narušování soukromí.

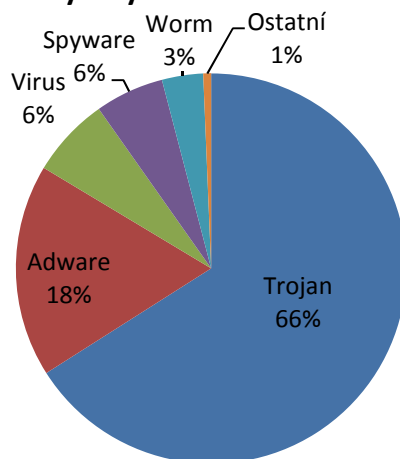
2.2 Současné trendy malware

Rok 2009 byl přelomovým rokem v historii malware a počítačových zločinů. Byly tím položeny základy, od kterých se předpokládá, že bude nadále směřován rozvoj a množství šířeného malware. V letech 2007-2008 téměř došlo k vymizení nekomerčního malware, přičemž hlavní proud tvořily trojské koně kradoucí data se zaměřením na počítačové hráče, jejich hesla a účty. V posledních 3-4 letech se Čína stala hlavním původcem malware. Čínští útočníci prokázali schopnosti vytvářet obrovské množství škodlivých kódů, přičemž společnosti zabývající se vývojem antivirových řešení museli vložit většinu snahy a prostředků právě do detekce a eliminace malware pocházející z Číny.

Statistiky vytvářené různými společnostmi hovoří o razantním růstu nových hrozeb. Společnost Kaspersky uvádí, že identifikovala 2 miliony unikátních vzorků malware v letech 1992-2007, avšak v samotném roce 2008 zaznamenala 15 milionů jedinečných vzorků [29]. V roce 2009 dosáhla hranice 33.9 milionů.

Rok 2009 je charakterizován komplexností škodlivých programů na bázi rootkitů, globálními epidemiemi šíření, virem Conficker, webovými útoky, botnety a útoky na sociální sítě. V tomto roce bylo očekáváno navýšení epidemií, přičemž tato předpověď byla korektní a tento trend je očekáván i do budoucna. K nejvýraznějším představitelům patří Sinowal¹, TDSS² a Clampi³.

Rozložení zachycených vzorků malware v roce 2009



Obrázek 1 - Rozložení zachycených vzorků malware v roce 2009 dle PandaLabs [zdroj http://www.pandasecurity.com/img/enc/Annual_Report_PandaLabs_2009.pdf]

2.3 Conficker

Conficker je bezesporu hlavní a nejpočetnější zdroj epidemiálního šíření malware v roce 2009. Conficker (známý též pod jmény Kido nebo Downadup) je počítačový červ šířící se pomocí chyby operačního systému Microsoft Windows. Napadá počítače několika známými způsoby - útoky silou na síťová hesla, skrze přenosová média a zranitelnost Windows MS08-067. Každá z infikovaných stanic se pak stává částí botnetu (sítě infikovaných počítačů), která je ovladatelná na dálku a využitelná pro téměř libovolnou úlohu.

Microsoft vydal v říjnu 2008 aktualizaci označenou MS08-067 [20], která chybu v systémech Windows opravuje. I přes to se udává, že 1 z 10 počítačů je stále zranitelný [28]. První verze červa se

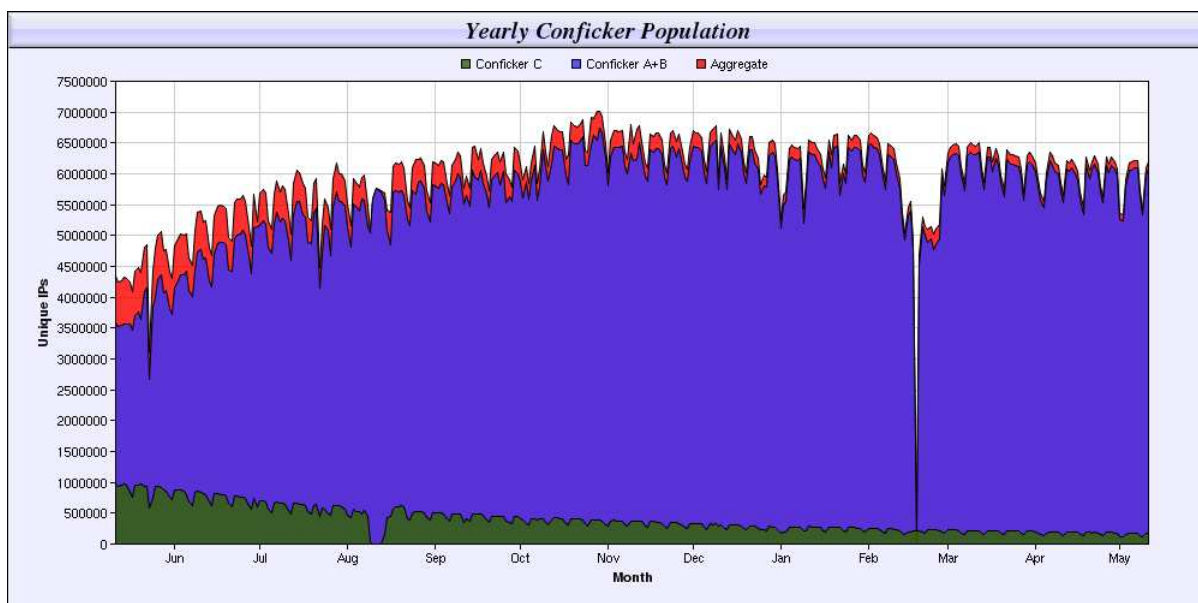
¹ Sinowal je představitelem bootkitu. Bootkit je jeden z nejpokročilejších typů malware, který je efektivně maskován a většina antivirových programů ho nebyla schopna zaznamenat. Vznikl v roce 2007, největšího rozmachu dosáhl v roce 2009. Jeho šíření probíhá skrze webové portály, často s obsahem pornografie a pirátského software. Kradе hesla a monitoruje klíčová slova zadávaná skrze webový prohlížeč, která pak odesílá na stranu útočníka.

² TDSS je implementován současně dvěma extrémně komplexními technologiemi - infikuje systémové ovladače Windows a vytváří svůj vlastní virtuální souborový systém, kde ukrývá vlastní soubory. Jedná se o první typ malware schopný proniknout do systému a pracovat na takové úrovni.

³ Clampi je trojský kůň, určený ke zcizení utajovaných uživatelských dat a umožnění vzdálenému přístupu k počítači.

začala šířit koncem roku 2008. Společnost Microsoft vypsalu odměnu \$250 000 na odhalení a dopadení autora viru, ovšem ten zůstává doposud skryt.

V lednu 2009 se počet napadených počítačů odhadoval na 9 milionů [21], odhad současného stavu je 7 milionů stanic. V průběhu jeho existence vzniklo několik variant Confickeru - Conficker.A až Conficker.E. V současné době existuje celá řada nástrojů detekujících jeho přítomnost a zajišťujících jeho následnou likvidaci. Mezi jednoduchý nástroj detekce patří Conficker Aye Chart [22]. Další informace o Confickeru lze nalézt například na [23, 24, 25]



Obrázek 2 - Počet napadených počítačů Confickerem v průběhu roku 2009/2010 [zdroj <http://www.shadowserver.org/wiki/uploads/Stats/conficker-population-year.png>]

2.3.1 Zranitelnost MS08-067

Jedná se o kritickou chybu ve Windows Server Service, kterou využívá pro napadení systému výše zmiňovaný červ Conficker a jiné škodlivé kódy. Server service umožňuje lokálním uživatelům sdílení zdrojů, jako jsou disky a tiskárny, takže k nim ostatní uživatelé sítě mohou přistupovat. Server service využívá Common Internet File System (CIFS) – SMB⁴ na protokolu TCP 445. Jde o chybu, která umožňuje vzdálené provádění kódu za využití RPC (remote procedure call) požadavků.

Pokud je exploit⁵ úspěšně proveden, zranitelnost MS08-067 umožňuje útočníkovi získat vzdálenou kontrolu nad počítačem se stejnými právy jako aktuálně přihlášený uživatel. V případě, že se jedná o uživatele s administrátorskými právy, to znamená možnost libovolného vytváření, úpravy a mazání souborů, instalaci programů a vytváření nových uživatelských účtů a podobně. Jakmile je

⁴ Protokoly SMB/CIFS jsou navrženy pro sdílení souborů a tiskáren. Původně navrženy IBM, později byl převzat Microsoftem a implementován pro TCP/IP komunikaci od systému Windows 2000 a výše. CIFS využívá klient/server programovací model, kde klient klade požadavky na server a ten mu příslušně odpovídá.

⁵ Exploit je programový kód nebo sekvence příkazů, která využívá bezpečnostních chyb a zranitelností v počítači či jiném zařízení, s účelem způsobit neočekávané chování. Tím bývá nejčastěji pokus získání neautorizovaného přístupu do systému, narušení bezpečnosti, odepření služeb a podobně.

červ v počítači, okamžitě zakazuje provoz služeb jako je wuauserv (automatické aktualizace Windows), WinDefend (Windows Defender), upravuje DNS⁶ záznamy a brání uživateli v přístupu na stránky antivirových společností a servery s bezpečnostní tematikou. Zamezuje tím bezpečnostní aktualizace a možnost případné detekce. Instaluje do počítače virus Waledac, rozesílá nevyžádanou poštu spolu s falešným antispywarem. Maže i body obnovy v rámci funkce "Obnova systému" a v registrech též navyšuje hodnotu TcpNumConnections, aby zvýšil množství současně otevřených spojení a tím i rychlost šíření na další PC v lokální síti i Internetu. Pak zahajuje další vlnu šíření - pro napadání systémů se zranitelností MS08-067 instaluje HTTP server, přes který hromadně zasílá speciálně upravené pakety (exploit) na další počítače a snaží se je ovládnout.

Pokusy o eliminaci botnetu byly o to obtížnější, že Conficker byl implementován za použití nejnovějších, sofistikovaných technik vytváření virů. Například jedna z modifikací červa byla navržena tak, aby se sama aktualizovala z více než 500 domén, přičemž ty byly náhodně vybírány ze seznamu více než 50 000 adres generovaných denně. Spojení na bázi peer-to-peer byly použity jako podpůrné kanály.

⁶ DNS neboli Domain name system/server je zjednodušeně řečeno nástroj překladu doménových jmen na IP adresy. IP adresa určuje jednoznačnou identifikaci počítače v síti, ale z důvodu lepší použitelnosti a zapamatovatelnosti pro uživatele jsou IP adresy asociovány s doménovými jmény, takže např. adresa www.vutbr.cz odpovídá 147.229.2.90

3 Technologie honeypotů

Tak jak bylo zmíněno v kapitole 2, růst počítačových hrozeb a bezpečnostních incidentů je nezadržitelný. Útočníci neustále přichází s novými viry, přičemž denně se jedná o tisíce nových vzorků a jejich mutací. Abychom byli schopni se těmto hrozbám účinně bránit, je neustále potřeba vylepšovat používané bezpečnostní mechanismy a být schopni nový malware a útočné techniky identifikovat v co největším předstihu.

I když existuje celá řada programů a nástrojů, které mají ochraně přispět, je potřeba neustálého zdokonalování a hledání způsobů, jak možné útoky nejenom odrážet, ale v ideálním případě jim i předcházet. Ke známým prostředkům počítačové ochrany slouží různé firewally⁷, antiviry⁸, IDS a IPS⁹ mechanismy, které většinou slouží jako přímá ochrana fyzických produkčních systémů. Technologie honeypotů na principu účelných nástrah slouží zejména k odhalování nových typů malware, na jejichž základě jsme schopni pomocí reverzního inženýrství tyto hrozby identifikovat, odvracet a eliminovat je.

3.1 Úvod do problematiky honeypotů

Koncept honeypotů byl poprvé dokumentován Cliffordem Stolleem [6] a Williamem R. Cheswickem [7] počátkem 90. let. Od té doby se honeypoty vyvíjely a byly nasazovány jako silné bezpečnostní nástroje, i když k jejich širšímu nasazení došlo až v nedávných letech. Dle definice podle [3] je honeypot „prostředek informačního systému, jehož hodnota leží v neautorizovaném nebo nedovoleném použití tohoto prostředku“.

Koncepčně téměř všechny honeypoty fungují stejně. Jsou prostředkem, který nemá žádnou předpokládanou produkční hodnotu a autorizované užití. Jakákoliv aktivita a pokus o kontakt s honeypotem je proto považován za podezřelý a potenciálně škodlivý.

3.2 Důvody nasazení

Z pohledu důvodu nasazení v praxi jsou honeypoty děleny do dvou kategorií – jmenovitě se jedná o produkční a výzkumné honeypoty.

Produkční honeypoty, tak jak název napovídá, jsou nasazovány v produkčních systémech s účelem ochrany organizace. Jedná se zejména o prevenci, detekci a odezvu na případný útok. Jsou

⁷ Firewall je součást počítačového systému nebo prvek sítě, který je navržen s účelem blokování neoprávněných a nepovolených přístupů ze sítě do systému. Existují jak ve formě hardwarového tak softwarového řešení, kdy v obecnějším pojetí fungují jako filtry datových toků, které jsou povolovány a zakazovány podle daných pravidel.

⁸ Antiviry jsou programy pro detekci a odstraňování nežádoucích a škodlivých programů, tzv. malware.

⁹ IDS/IPS (z ang. Intrusion detection/prevention system) jsou bezpečnostní monitorovací nástroje, které analyzují prováděné aktivity v počítači a síti a snaží se odhalit nezvyklé chování a možné bezpečnostní incidenty. IDS se zejména snaží možné narušení detekovat, IPS v ideálním případě mu předcházet.

umístěny v produkční síti společně s dalšími servery. Cílem je zlepšit bezpečnostní informační systémy a snížit riziko bezpečnostních incidentů. Snaží se nasměrovat útočníka směrem pryč od kritických částí sítě a jeho úsilí vložit do návnady, zatímco reálné aktiva jsou chráněna. Většinou se jedná o nízko-interaktivní honeypoty (viz. kapitola 3.4).

Hlavním účelem výzkumných honeypotů je zachycovat informace o záškodnických aktivitách, sledovat práci útočníků, evidovat nástroje a taktiky které používají. Sběr údajů má následně různé využití – sledování trendů a způsobů útoků, k preventivním varovným účelům, odhalování nových typů útoků a podobně. Nepřináší přímý prospěch pro konkrétní firmu, jsou nejčastěji provozovány specialisty, vzdělávacími a vládními institucemi a dobrovolníky se zájmem o bezpečnost.

3.3 Přínos a význam honeypotů

Jak již bylo popsáno výše, existuje celá škála honeypotů. Tak jako se liší jejich primární určení, tak se liší i jejich přínos. Tak jako u výzkumných honeypotů je důležitý sběr dat za různým účelem následného vyhodnocení (objevování nových hrozeb, tvorba statistických metrik, právní aspekty, atd.), tak u produkčních honeypotů se jedná zejména o mechanismy zajišťující preventivní opatření vůči možnému napadení, dále pak schopnost detekovat případný výskyt útoku a provádět odpovídající odezvu, která odvrací útočníka od skutečných produkčních systémů, případně se snaží o minimalizaci možných škod.

Z definice honeypotu vychází, že neposkytují žádné skutečné služby v produkční síti a netvoří tak velký síťový provoz. Z tohoto důvodu je logování mnohem snazší a méně náročné na objem zpracovaných dat než počítače provozující legitimní služby.

Honeypoty jako součásti bezpečnostních informačních systémů mají tři hlavní úkoly - prevenci, detekci a odezvu na síťové hrozby. Mezi další rozšíření patří mechanismy, které omezují a zpomalují šíření červů po Internetu. Za tímto účelem byly vytvořeny tzv. sticky honeypoty - ty pomocí různých metod a triků zpomalují odezvu na útočnickovy požadavky a snižují rychlost šíření po síti.

Honeypoty mají značný potenciál a i do budoucnosti celou řadu možných využití. Je ovšem zřejmé, že nejsou schopny nahradit důležitost stávajících řešení a bezpečnostní technologie. Měly by být používány jako doplněk a rozšíření bezpečnostních mechanismů.

Shrnutí výhod a nevýhod honeypotů (převzato z [3]):

Výhody: Honeypoty jsou jednoduchým konceptem, který skýtá řadu silných stránek.

- *Malé množství dat velké hodnoty:* Honeypoty sbírají data relativně malého objemu. Namísto produkování gigabytů dat za den logují pouze megabajty dat denně. Namísto generování

tisíců varování generují pouze řádově desítky varování za den. Zachycují pouze podezřelou aktivitu, protože žádná předpokládaná není očekávána.

- *Nové nástroje a taktiky:* Jsou navrženy tak, aby zachycovali cokoliv, s čím přijdou do styku, včetně nových, dosud neznámých nástrojů a taktik útočníka.
- *Minimální zdroje:* Vyžadují minimální hardwarové zdroje, takže i staré PC dokáže emulovat celou B podsít¹⁰.
- *Šifrování a IPv6:* Oproti většině bezpečnostních technologií (IDS a podobně) pracují spolehlivě se šifrovanou a IPv6 komunikací. Bez ohledu na to, jaká data přijmou, jsou schopna je detekovat a zachytit.
- *Informace:* Honeypoty dokážou sbírat hloubkové detailní informace, což většina ostatních technologií obtížně zvládá.
- *Jednoduchost:* Koncepce honeypotů je jednoduchá. Nejedná se o složité vývojové algoritmy, tabulky stavů nebo signatury vyžadující neustálé aktualizace. Čím jednodušší technologie, tím je méně pravděpodobné, že se vyskytnou chyby nebo špatná konfigurace.

Nevýhody: Tak jako každá technologie, mají i svoje slabosti. Nenahrazují stávající řešení, ale koexistují s nimi.

- *Omezený pohled:* Honeypoty mohou sledovat a zachycovat pouze aktivity, které s nimi přímo interagují. Nezachytí útoky vůči jiným systémům, pokud útočník nekomunikuje i s nimi.
- *Riziko:* Všechny bezpečnostní technologie mají svá rizika. Firewally lze proniknout, šifrování lze prolomit, IDS senzory mohou selhat při detekci útoku. Honeypoty mají taktéž své slabé stránky, útočník nad nimi může převzít kontrolu a využít je k napadení dalších systémů. Riziko se liší s typem honeypotu.

3.3.1 Zero-day malware

V rámci využití maximálního potenciálu honeypotů v oblasti prevence útoků jsou využívány zejména výzkumné honeypoty. Právě díky nim bývají odhalovány a vyhodnocovány nové, tzn. zero-day hrozby. Zero-day malware tvoří doposud neznámé viry a další škodlivý software, pro který nemá antivirový software odpovídající signatury¹¹ a nedokáže je identifikovat.

Tradiční antivirové programy spoléhají na popis malware na bázi signatur. Je to poměrně spolehlivý přístup, ovšem nedokáže nás uchránit proti malware, jehož vzorky doposud nebyly zaznamenány a vytvořeny pro ně odpovídající otisky. Z toho důvodu jsou moderními bezpečnostními systémy využívány i další způsoby identifikace, takzvané heuristické analýzy:

¹⁰ B podsít neboli B třída je rozsah IP adres v rozsahu 128.0.0.0 - 191.255.255.255. V tomto rozsahu je možné adresovat 65534 stanic.

¹¹ Signatura neboli otisk je unikátní rozlišovací posloupnost instrukcí kódu, která je specifická pro daný virus. V rámci detekce může docházet k nepřesnostem, protože shoda kódu nemusí být vždy známkou viru (kód může být pouze částí větší binární posloupnosti). Může tím docházet k nekorektnímu vyhodnocení a invalidním hlášením.

- *Statická analýza kódu* - zdrojový kód je analyzován a je ověřováno, zda jeho obsah a posloupnost příkazů vypadá podezřele. Při této analýze se program nespouští, ale provede se převod do jazyka symbolických instrukcí. Pomocí následného vyhodnocení se hledají vzorky a sekvence kódu, které mohou nasvědčovat jeho škodlivost.
- *Behaviorální analýza* - analyzovaný program je spuštěn ve virtualizovaném prostředí, které je uzavřené a dobře monitorované (tzv. sandbox). Kontroluje se chování programu v rámci systému, tzn. přístupy na disk, do paměti, množství spouštěných procesů, síťová spojení a další parametry. Jsou sledovány anomálie a podezřelé chování, které rozhodne o možné škodlivosti.

3.4 Interakce honeypotů

Jeden z nejpoužívanějších způsobů dělení honeypotů je na základě stupně poskytované interakce. Interakce definuje míru činností, kterou honeypot útočníkovi nabízí. Obecně platí, že čím nižší míru interakce nabízí, tím méně nových informací z bezpečnostních incidentů lze zjistit. Ruku v ruce s mírou interakce se odvíjí míra rizika a možného narušení systému, které může záškodník způsobit.

Honeypoty s nízkou mírou interakce obecně fungují na bázi emulování služeb a dílčích částí operačního systému. Služby jsou většinou řízeny skripty, poskytující omezenou funkcionalitu, ale navozující dojem služby skutečné. Možné aktivity útočníka jsou omezeny na základě míry emulace, jsou ale vesměs malé. Tímto způsobem jsme schopni získat pouze omezené množství informací, jedná se především o statistické údaje o počtu incidentů a údaje, že někdo se pokoušel na danou službu útočit. Jejich hlavní výhodou je jejich jednoduchost, údržba a nasazení s minimálním rizikem zneužití.

Kontrast k výše uvedenému typu jsou honeypoty s vysokou mírou interakce. Jedná se nejčastěji o skutečné počítače, na kterých běží plnohodnotné operační systémy se službami a aplikacemi, které nabízejí stejné interakční možnosti jako kterýkoliv běžný počítač připojený k Internetu. V případě, že se útočníkovi podaří obelstít bezpečnostní mechanismy systému, může ovládnout celý honeypot a z něj spouštět útoky na další systémy. Hlavní úlohou je detailní zaznamenávání údajů, jakým způsobem útočník postupuje, jaké nástroje a taktiky používá a následně provést hloubkovou analýzu získaných dat.

Nízko interaktivní (Emulované OS a služby)	Vysoko interaktivní (Žádná emulace, reálný OS a služby)
Snadná instalace a nasazení, většinou jednoduchá konfigurace na PC	Komplexní instalace a nasazení (snadné u komerčních řešení)
Emulovaný TCP/IP zásobník, zranitelnosti a podobně	Skutečný OS, služby a aplikace
Nižší míra rizika zneužití	Vyšší míra rizika, poskytují útočníkovi plnohodnotný OS
Zachycují limitované množství informací, kvantitativní informace o útocích	Zachycují velké množství detailnějších informací, evidence použitých nástrojů útočníka, sledování zadávaných příkazů

Tabulka 1 - Srovnání honeypotů s vysokou a nízkou mírou interakce (převzato a upraveno z [1] a [3])

3.5 Další typologie honeypotů

3.5.1 Fyzické a virtuální honeypoty

Fyzické honeypoty běží na skutečném stroji a hardware. Často odpovídají nasazení vysoce interaktivních honeypotů. Jsou drahé na pořízení a správu. Nasazení pro honeynet (viz. kapitola 4.1) s velkým adresovým prostorem téměř nemožné.

Kontrast tvoří virtuální honeypoty, které umožňují mít tisíce honeypotů na jediném fyzickém stroji. Mezi výhody patří škálovatelnost a snadná správa. K známějším řešením patří ta využívající VMware¹² a další.

3.5.2 Bezdrátové honeypoty

Cílem těchto honeypotů je chránit bezdrátové sítě. Vytváří stovky fiktivních sítí, ke kterým se útočník zkouší připojit. Mohou produkovat falešnou komunikaci mezi klienty sítě a ztížit získání užitečných informací pro přístup k síti. V kombinaci s tradičními honeypoty dokážou předstírat tisíce komunikujících PC. Více informací lze nalézt v [8].

3.5.3 Klientské honeypoty

Na rozdíl vůči tradičním (serverovým) honeypotům, které pasivně čekají než budou napadeny, je zde i relativně nový typ klientských honeypotů, které aktivně brouzdají sítě a vyhledávají záškodnické

¹² VMware je produktem společnosti VMware Workstation sloužící pro virtualizaci jednoho nebo více počítačů na jednom hostitelském stroji. Jako emulátor poskytuje hostům kompletní sadu virtualizovaného hardware, počínaje procesorem, operační pamětí, grafickou kartou, disky a dalšími komponentami. Pomocí VMware mohou být emulované systémy s OS na bázi Windows, Linux, Solaris, FreeBSD a Novell NetWare.

servery – jedná se zejména o webové servery šířící exploity a další malware. Tak jako u tradičních serverových honeypotů zde existují nízko a vysoko-interaktivní verze.

Nízko-interaktivní využívají simulovaného klienta, interagují se servery a klasifikují je na základě výskytu daných pravidel podezřelých aktivit – většinou pomocí statické analýzy a shodnosti vzorů. Jsou rychlé, ale mohou některé servery chybně klasifikovat s ohledem na to, že se nechovají jako skutečné klientské programy a mají restriktce.

Klientské honeypoty s vysokou mírou interakce bývají realizovány jako vizualizované dedikované systémy se snadným návratem do původního bezpečného stavu po úspěšném napadení. Interagují s podezřelým serverem a monitorují systém pro nepovolené změny stavu v konfiguraci (systémových souborech, registrech a podobně). Více informací na [11]

3.6 Legální aspekty

Dosud jsme se zabývali technickou stránkou honeypotů, jejich výhodami a nevýhodami a přínosem pro bezpečnost sítě. S celou touto tematikou je spojena i druhá strana pohledu a to je legálnost použití honeypotů. Můžeme získat spoustu zajímavých informací, které nám mohou pomoci s řešením a předcházením bezpečnostních incidentů, ale je možné libovolně sbírat, využívat, případně je použít jako iniciační důkaz proti útočníkům?

Na tuto otázku neexistuje krátká jednoznačná odpověď, vyskytuje se zde totiž celá řada proměnných – právní ustanovení státu, kde je honeypot provozován, jaké typy informací sbírá a co je zamýšleno s nimi dělat a podobně. V tomto bodě vyjdeme z podstaty právního systému Spojených států Amerických, protože velká část technologií spojených s honeypoty pochází právě odtud a je právě pro tuto zemi nejlépe popsána. Základní myšlenky a právní řády zcela jistě nesou základní podobnost i pro ostatní země.

Tři hlavní aspekty týkající se daného tématu jsou ochrana soukromí (ang. privacy), účelná nástraha (ang. entrapment) a odpovědnost za provoz honeypotu (ang. liability). Účelná nástraha se vztahuje na situace, kdy vládní nebo jiná organizace se účelně zachovala tak, že přiměla osobu spáchat zločin nebo jinak překročit zákon. Tato úvaha vychází z myšlenky, že osoba by nepodnikla trestný akt, pokud by nedošlo k záměrnému oklamání a podnětu ze strany provozovatele. Tento způsob obhajoby útočníka ovšem není aplikovatelný na privátní honeypoty. Provozovatel sítě a honeypotu má odpovědnost zachovat síť bezpečnou, takže má právo monitorovat aktivity na ní. Existují zde však restriktce, kde jednou z nich je čtvrtý dodatek (ang. Fourth Amendment), který omezuje monitoring, pokud je provozovaný vládní agenturou bez soudního příkazu. Soukromé organizace mohou provozovat honeypot bez obavy z porušení tohoto zákona. Další ze zákonných složek je tzv. Wiretap Act, který zakazuje zachycování komunikace až na vyjmenované důvody a předem dané výjimky (specifické účely vládních organizací, monitoring na základě schválené vzájemné úmluvy, výjimky poskytovatele sítě atd.). Posledním z výše zmíněných bodů je

odpovědnost za provoz honeypotu. V případě, že útočník získá kontrolu nad honeypotem, může využít datové pásmo sítě a další systémové prostředky pro napadení ostatních. Může systém využít pro šíření spamu, distribuci ilegálních materiálů a mnohé další.

Ačkoliv zařazení honeypotů do právních norem států není ještě plně postihnuto a neexistuje větší množství exemplárních soudních rozhodnutí z této oblasti, tak je vřele doporučováno podle zamýšleného účelu nasazení honeypotu ověřit stavy a restrikce státních norem, kde se honeypot nachází. Důraz na ochranu osobních údajů a možné zneužití vlastních výpočetních prostředků útočníkem jako mezikrok páchání další trestné činnosti je v dnešní době hodně aktuální. (Informace o legální problematice v USA volně čerpány z [5].)

4 Honeypoty v praxi

V této kapitole se zaměříme na dostupná řešení, fungující na principech technologie honeypotů. Jak již bylo zmíněno v kapitolách 3.4 a 3.5, existuje celá škála typologií honeypotů. Vždy, než dojde ke skutečnému nasazení konkrétního honeypotu do fyzického prostředí, je potřeba ujasnit, jaké jsou na něj kladeny funkční požadavky, účel využití a výsledné výstupy. Podle typu interakce jsme schopni nabízet útočníkům řešení s omezenou nebo naopak rozšířenou škálou služeb, v produkční síti půjde zejména o ochranu aktiv podniku, v případě výzkumných honeypotů se bude jednat o sledování stavu aktuálních hrozeb.

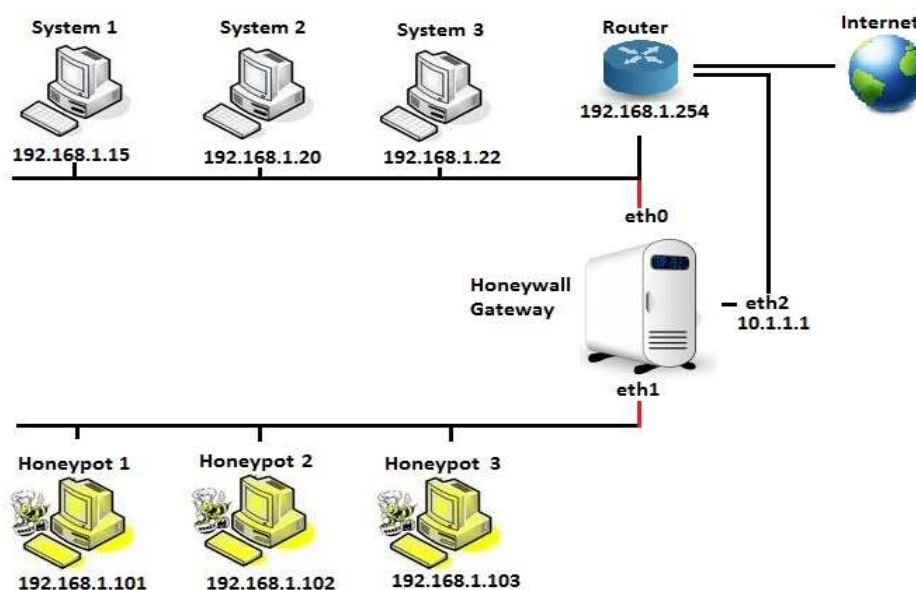
4.1 Honeynety (sítě honeypotů)

Honeynet je typ honeypotu. Jedná se o vysoce-interaktivní honeypot určen k zachytávání velkého množství informací o hrozbách.

Honeynety reprezentují extrém vysoko-interaktivních honeypotů. Obvykle se skládají z několika honeypotů různých typů (různých platforem a operačních systémů). To nám umožňuje sbírat velké množství podrobných data o různých typech útoků.

Honeynet je architektura – vytváří vysoce kontrolovanou síť, kde probíhá kontrola aktivit, kterých v ní probíhají. V mnoha literaturách je honeynet přirovnáván k akváriu – vytvoříte prostředí, kde můžete sledovat vše, co se děje uvnitř. Namísto korálů, kamenů a vodních řas zde zasadíme Linuxové servery, routery a stanice. Tak jako ryba viditelně interaguje s prostředím v akváriu, tak útočník interaguje s honeypoty [12].

Klíčovým prvkem architektury je honeywall. Představuje bránu (ang. gateway), která odděluje honeynet od ostatních sítí. Veškerý síťový provoz musí jít přes bránu, která je tradičně přepínacím zařízením na datové vrstvě (2. vrstva - data link layer - modelu ISO OSI).



Obrázek 3 - Schéma honeywallu : honeywall má 3 síťová rozhraní. eth0 a eth1 separují honeypoty od zbytku sítě, jsou to bridge rozhraní bez IP zásobníku. eth2 je volitelné, obsahuje IP zásobník, slouží k vzdálené administraci. (Převzato z [12])

Sdružení Honeynet Project, stanovilo soubor pravidel a standardů, které by měl honeynet splňovat [12]:

- *Kontrola dat (ang. Data control)* – definuje, jak síťové aktivity v rámci honeynetu zůstanou uvnitř sítě, bez toho, že by o tom útočník věděl. Provádí různé restrikce a filtraci dat. Účelem je minimalizovat potencionální rizika, které útočník může způsobit, když by prolomil bezpečnostní mechanismy a použil honeynet k napadení dalších systémů.
- *Zachytávání dat (ang. Data capture)* – slouží k monitorování a zachycení údajů a dat o veškerých činnostech útočníka, aniž by o tom věděl
- *Analýza dat (ang. Data analysis)* – schopnost analyzovat získaná data
- *Sběr dat (ang. Data collection)* – schopnost shromažďovat data z více honeynetů do jediného sběrného místa

4.2 Honeypoty s nízkou mírou interakce

Honeyd

- Démon provozující virtuální hosty na volných IP adresách, běží na jediném stroji poskytujícím tisíce virtuálních stanic
- Emuluje síťové služby a celou řadu různých operačních systémů
- Velká možnost nastavení – topologie sítě, routování, subsystémy, skripty služeb, logování, pluginy a podobně
- OS: BSD, Linux, Solaris, Windows
- [http://www.honeyd.org/]

Deception Toolkit

- Deception Toolkit (DTK) byl vytvořen Fredem Cohenem v roce 1998
- Jeden z nejstarších nízko-interaktivních honeypotů
- Napsán v C s použitím Perl skriptů, používá TCP wrappery ke zpracování požadavků na služby na portech, které jsou normálně blokovány
- Deception service – služba, která informuje útočníka po připojení, že se připojil k DTK – účel možného odrazení útočníka a zastavení útoku
- OS: Unix
- [<http://all.net/dtk/index.html>]

LaBrea

- Koncept tarpitu/sticky honeypotu – služba která zpomaluje činnost útočníka pomalými odezvami a nízkou konektivitou; snaží se útočníka pozdržet co nejdéle a sestavené spojení přenést do stavu, odkud nebude moci dále postupovat
- Hledá volné IP adresy a vytváří pro ně virtuální stroje, pro nevyužité IP přes ARP na ně odpovídá
- OS: FreeBSD, Linux, Solaris, Windows 98/2000
- [<http://labrea.sourceforge.net/labrea-info.html>]

HoneyC

- Klientský honeypot
- Tři moduly – queuer, visitor, analysis engine: první vytváří seznam podezřelých serverů, druhý se servery komunikuje a získává informace, třetí je vyhodnocuje
- Využívá signatury Snort
- Implementován v Ruby
- OS: nezávislé - napsáno v interpretovaném jazyce
- [<https://projects.honeynet.org/honeyc/>]

Tiny Honeypot

- Vytvořil George Bakos v r. 2002, napsán v Perlu
- Postaven na přesměrování iptables a xinetd listeneru. Naslouchá na nepoužívaných TCP portech, loguje aktivitu a poskytuje odezvy útočníkovi
- Jeví se vždy zranitelný – nezáleží na tom, jaký útok je spuštěn, vždy se bude tvářit jako úspěšným výsledkem.
- OS: Linux
- [<http://freshmeat.net/projects/thp/>]

GHH – Google Hack Honeypot

- Reakce na útoky na vyhledávací stroje
- Rozpoznává útoky založené na hackovacích nástrojích pro vyhledávače
- Implementováno v PHP
- OS: nezávislý - s podporou PHP, MySQL
- [<http://ghh.sourceforge.net/>]

PHP.HoP

- Mechanismus identifikování a sledování hrozeb na Webu
- Aplikačně založený, sloužící k matení webových útočníků, tvorbě statistik o nejpoužívanějších příkazech a zachycení nástrojů narušitelů

Kojoney

- Emuluje SSH server
- Napsán v Perlu, používá Twisted Conch knihovny
- OS: Unix
- [<http://kojoney.sourceforge.net/>]

HoneyWeb

- Emuluje různé typy web serverů, může se dynamicky měnit na základě typu požadavku
- Funguje samostatně nebo ve spojení s Honeyd
- Napsán v Perlu
- OS: Windows, Linux

KFSensor

- Komerční řešení, pouze pro platformu Windows
- Démon provozuje virtuální hosty na volných IP adresách, podporuje vzdálenou správu
- Emulace služeb – IIS, FTP, SMTP, POP3, SMB, ...
- [<http://www.keyfocus.net/kfsensor/>]

Nepenthes

- Modulární systém, primárně určen pro detekci nového malware
- Emuluje známé zranitelnosti
- Sesbírané vzorky a informace jsou dále poskytnuty open source antiviru ClamAV
- OS: Unix
- [<http://nepenthes.carnivore.it/>]

Honeytrap

- Určen k automatizovanému sběru malware
- Hlavní myšlenka je oklamat útočníka a získat kompletní data, která pak mohou být dále analyzována
- Dynamicky otevírá TCP porty v době příchozího požadavku, tímto způsobem je možné odpovídat na většinu síťových útoků
- Primárně určen k zachycení dosud neznámých (zero-day) útoků
- OS: FreeBSD, Linux, OpenBSD
- [<http://sourceforge.net/projects/honeytrap/>]

BackOfficer Friendly (BOF)

- Jednoduchá instalace a údržba na úkor menší konfigurovatelnosti a množství nastavení u jiných honeypotů
- Nabízí pouze omezené množství emulace 7 služeb a horší logování
- Systém hlášení o útocích
- OS: Běží na Unix i Windows systémech

Specter

- Komerční řešení
- GUI Specter Control poskytuje přehledný o konfiguraci a aktuálním stavu systému
- Systém okamžité notifikace o útoku
- OS: Windows XP
- [<http://www.specter.com/>]

4.3 Honeypoty s vysokou mírou interakce

Argos

- Postavený na Qemu – open-source emulátoru, který používá dynamického překladu a poskytuje dobrou rychlost; rozšiřuje ho za účelem detekce vzdálených útoků a pokusů o ohrožení emulovaného operačního systému
- Určen pro detekci neznámých hrozeb
- Využívá tzv. dynamic taint analýzu – sleduje provádění a úpravy dat a detekuje pokusy o jejich využití nedovoleným způsobem, následně vytváří obrazy paměti
- Podporuje emulaci Linuxových i Windows systémů
- [<http://www.few.vu.nl/argos/>]

Capture-HPC

- Klientský honeypot
- Klient/server architektura – umožňuje kontrolovat z jednoho místa více lokálních i síťových hostů
- Sleduje souborový systém, registry a procesy jádra
- Automatizované sbírání malware a evidence logu
- Určeno pro nasazení na VMWare Serveru
- [<https://projects.honeynet.org/capture-hpc>]

Collapsar

- Hybridní systém
- Skládá se z komponent – traffic redirector, transparentní firewall frontend, komponenta virtuálních honeypotů, kontrolní stanice, korelační mechanismus; redirector je instalován v různých produkčních sítích, zbytek je soustředěn v kontrolním bodě
- Redirector přeměrovává datový tok do centrálního místa Collapsar centra pomocí GRE tunelu, které funguje jako firewall. Zpracovaná data jsou po odchycení a analýze opět poslána zpět do produkční sítě, kde byla zachycena
- Více v [13], [1]

Potemkin

- Virtuální farma honeypotů, nasazeno až 64 000 stanic za použití pouze několika fyzických počítačů
- Využívá přidělování systémových prostředků podle aktuálního počtu připojení ve chvílích potřeby
- Vytváření paměťových snímků systému pro analýzu útoků
- Datový provoz evidován přes bránu routeru, která eliminuje možné napadení dalších počítačů při kompromitaci systému. Brána je hlavním prvkem zajišťující bezpečnost a management systému
- Více informací v [14], [1]

Podpůrné nástroje:

VMWare

- Nástroj pro široké spektrum vizualizace systémů a software, jeden z nejrozšířenějších na trhu
- Podpora většiny typů operačních systémů, řešení pro domácnosti i velké firemní prostředí
- [<http://www.vmware.com/>]

User-Mode Linux (UML)

- Vytváří virtuální systémy a honeypoty, jednoduché k nastavení, nabízeno bezplatně
- Dokáže simulovat pouze Linuxové systémy
- [<http://user-mode-linux.sourceforge.net/>]

Více informací a přehled dostupných řešení lze nalézt na: [1], [15], [3], [16],[17], [18]

4.4 Volba honeypotu pro praktickou část

Na základě předchozí analýzy a rozdělení honeypotů byl pro následnou praktickou část práce vybrán honeypot s nízkou mírou interakce Honeyd. Hlavním důvodem bylo provedení plánované analýzy, která má poskytnout podrobnější výsledky týkající se úrovně detailu informací, které nízko interaktivní honeypot dokáže poskytovat a ověření, zdali emulované služby jsou schopny připravit dostatečně zajímavý cíl pro útočníka, na jehož základě pak lze zrekonstruovat aktivity a možné útoky, o které se pokoušel.

Honeyd byl vybrán s ohledem na jednoduchost řešení, možnosti rozšíření, flexibilitu a to, že se jedná o jednoho z často využívaných zástupců honeypotů s nízkou mírou interakce. Další výhody a popis je uveden v kapitole 5, která se Honeyd věnuje podrobně.

5 Honeyd

Honeyd je zkratkou pro honeypot daemon. Honeyd [19] je open-source honeypot s nízkou mírou interakce vytvořený Dr. Nielsem Provosem v dubnu 2002, který tento nástroj vytvořil ke studiu metod a taktik používaných útočníky a hackery.

Honeyd je framework, který vytváří virtuální systémy na síti. Emulované systémy jsou konfigurovány s různými službami a jejich personalita může být upravena tak, že se jeví, jako že běží na konkrétním operačním systému. Podstatou tohoto řešení je emulace portů služeb napodobováním IP zásobníku různých OS. Honeyd může běžet na jediném počítači a emulovat mnoho systémů na nevyužitých síťových adresách (až pro síť o velikosti 65536 adres). Honeyd je open-source software vydaný pod GNU General Public Licence, existují dostupné verze pro Unix, Linux, Windows, tak i další operační systémy.

Hlavní vlastnosti a schopnosti:

- *Simuluje tisíce virtuálních systémů současně:* Hlavní důvod využívání Honeyd je jeho schopnost vytvářet celé sítě honeypotů v jednom okamžiku. Útočník může komunikovat s každým z těchto systémů po síti a u každého nalézt jiné chování a vlastnosti na základě toho, jak byl konfigurován.
- *Konfigurace služeb skrze konfigurační soubor:* Můžete snadno nastavit služby a OS, které budou připraveny pro interakci s útočníkem. Kdykoliv Honeyd obdrží požadavek o nové připojení, spustí program, který jste určili pro komunikaci na daném portu, a pošle příslušnou odpověď na útočníkův síťový dotaz. Alternativou běhu podpůrných programů a skriptů je přesměrování požadavků na jiný obsluhovaný počítač.
- *Simuluje operační systémy na úrovni TCP/IP zásobníku:* Honeyd emuluje odpovědi na ARP, ICMP, TCP a UDP pakety. Ostatní typy paketů jsou zahazovány. Tato vlastnost umožňuje oklamat Nmap a Xprobe2 tak, že se domnívají, že virtuální honeypot opravdu skýtá skutečný běžící a konfigurovaný OS. K podpoření dojmu skutečného systému slouží i podpora skládání fragmentů a FIN-skenování u paketové komunikace.
- *Simulace routovací topologie:* Routovací topologie honeypotů může poskytovat komplexní řešení, přes nastavení odezvy (ang. latency), ztrátu paketů (ang. packet loss) a datové pásmo (ang. bandwidth). Honeyd podporuje asymetrické routování, integraci fyzických stanic do virtuální topologie a distribuované operace skrze GRE tunely.
- *Virtualizace subsystémů:* Se subsystémy může Honeyd spouštět skutečné Unixové aplikace pod virtuálním jmenným prostorem honeypotu - jedná se např. o web servery, FTP servery a jiné. Tato schopnost dále nabízí dynamické svazování portů a inicializaci síťových připojení na pozadí.

- *Logování akcí:* Ačkoliv je tato schopnost omezená, umožňuje zaznamenávat aktivity prováděné s honeypotem na výstup, tak i do logovacího souboru. Zaznamenává krátký popis akce, typ protokolu, informace o zdrojové/cílové adrese a portu. Soubor logu poskytuje ještě některé další údaje.

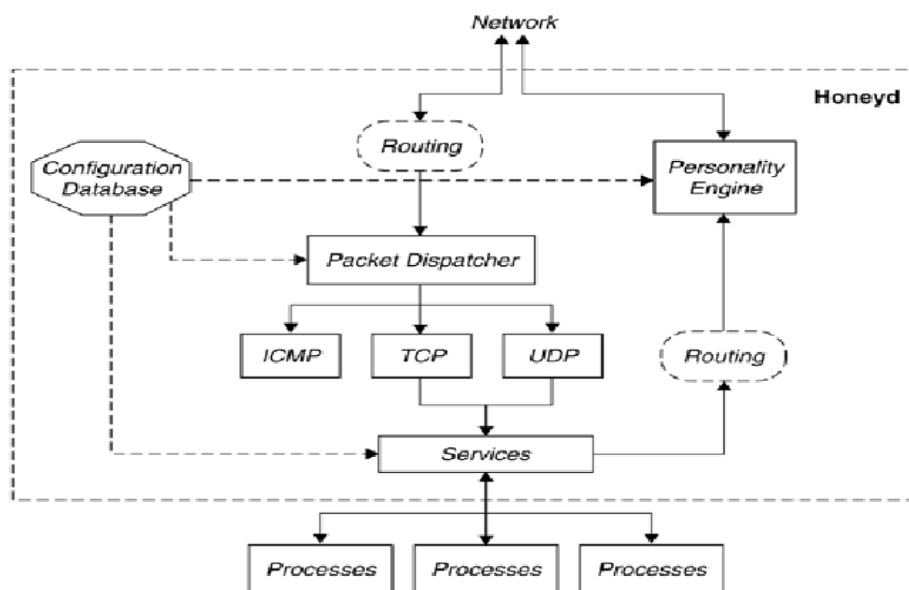
Honeyd je jedním z populárních honeypotů dnešní doby. Nabízí více funkčnosti a flexibility než většina ostatních honeypotů s nízkou mírou interakce. Největšího rozmachu dosáhl v letech 2003-2007, kdy byla vydána poslední dostupná verze 1.5c.

Honeyd se stal předlohou a jedním z prvních využitelných řešení, ovšem jeho rozvoj není v dnešní době nadále autorem a komunitou podporován, což přináší některá omezení a nemožnost plně a věrohodně simulovat současný stav dostupných operačních systémů a služeb.

5.1 Přehled konstrukčního řešení Honeyd

Abychom pochopili jako Honeyd pracuje a jak může být používán, musíme nejprve porozumět jeho základnímu návrhu. Obrázek 4 znázorňuje základní schéma architektury Honeyd, které je zodpovědné za chování honeypotu a to konkrétně:

- 1) útočníci interagují s Honeyd pouze ze sítě
- 2) Honeyd simuluje počet honeypotů podle potřeby, na základě přidělených IP a konfigurovaných služeb
- 3) klame nástroje ověřující systémové otisky (ang. fingerprints) tak, že mění každý odchozí paket, aby odpovídal charakteristice konfigurovaného OS a zachoval co největší důvěryhodnost a maskování.



Obrázek 4 – Základní architektura Honeyd (Obrázek převzat z [1])

5.2 Emulované služby

Honeyd je vytvořen tak, aby odváděl co nejlepší práci, co se týče mimikování¹³ charakteristik různých IP zásobníků a má velkou flexibilitu v emulování TCP a UDP portů. Dokáže emulovat libovolný port v rozmezí 0 - 65535, který může být použit jako jednoduchý nebo s využitím servisního skriptu, případně proxy.

Jednoduché porty (ang. simple ports) představují primitivní naslouchací proces na daném portu. Funkčně nedokážou více, než sestavení vzdálené komunikace, žádná data nejsou zasílána zpět, jde jen o zaznamenání informace o pokusu o spojení. Bez použití dalšího nástroje, který dokáže více analyzovat obsah datového toku (např. Snort), Honeyd zaloguje pouze stručné souhrnné informace. Jednoduché porty mohou být definovány různě a podle toho se odvíjí i odezvy na příchozí pokusy o spojení.

Typ protokolu portu	Nastavení	Popis chování
TCP	Open	Výchozí nastavení, odpovídá pomocí SYN/ACK paketu. Pokus o spojení je zobrazen na výstup jako "Connection established..." a údaje uloženy do logu. Útočník nemůže uplatnit svoje nástroje, které složitostí přesahují sken portů.
	Block	Bez odezvy. Definuje port jako neexistující, někde zvaný jako stealth. V produkční síti je to často zapříčiněno nastavením firewallu, ale v případě bez firewallu to není běžné chování. Žádný výstup na obrazovku není proveden, ale pokus o spojení je evidován v logu.
	Reset	Mimikuje běžné odezvy OS na neexistující port. Posílá RST/ACK paket, výstup na obrazovce je "Connection closed", událost uložena do logu.
UDP	Open	Bez odezvy, spojení logováno.
	Block	Bez odezvy, spojení přerušeno, není logováno.
	Reset	Odezva pomocí ICMP port chybové zprávy.
ICMP	Open	Výchozí nastavení, ICMP reply odezva.
	Block	Spojení přerušeno, žádná odezva neodeslána.

Tabulka 2 - Popis chování jednoduchých portů honeyd (Údaje převzaty z [2])

Emulované servisní skripty jsou jednou z nejvíce flexibilních vlastností Honeyd, které umožňují mimikovat aplikace a síťové služby. Servisní skripty je možné vytvářet téměř v libovolném

¹³ Mimikování je schopnost napodobovat co nejvěrohodněji skutečnou předlohu, tzn. v našem případě typ operačního systému, danou službu, její vlastnosti a očekávané chování a podobně.

z běžně používaných skriptovacích jazyků (Perl, Python, Bash, ...). V ideálním případě se snažíme vytvořit skript, který se chováním co nejvíce přibližuje skutečné službě, čímž pádem máme šanci zabavit útočníka na delší dobu a získat více informací. Honeyd a komunita kolem něj nabízí již řadu hotových řešení, která jsou k dispozici na [<http://www.honeyd.org/contrib.php>] - zahrnují varianty IIS, FTP, POP, SMTP, telnet. Některé další skripty lze nalézt jako součást řešení honeyd_kit, které se nachází na [<http://www.tracking-hackers.com/solutions/honeyd/>].

Omezujícím prvkem je nedostatek dalších servisních skriptů, zejména pro novější verze OS a přidružených programů. Honeyd je jedním z průkopníků v oblasti honeypotů, ovšem jeho další rozvoj a rozšiřování skriptů komunitou v poslední době značně upadl. Zmíněné skripty jsou zejména uživatelsky interaktivní, tzn. na bázi přímého zadání příkazů uživatelem do konzole a následné odezvy ze strany serveru. Pro složitější řešení automatizovaných útoků nelze poskytnout adekvátní základnu emulovaných služeb. Součástí práce je i implementace vlastního servisního skriptu bezpečnostní zranitelnosti MS08-067, kde v kapitole 0 jsou blíže popsány úskalí zjištěná při implementaci.

Posledním z trojice zástupců pokrývajících služby nabízené Honeyd jsou proxy služby. Ty zajišťují směrování příslušných spojení na jiné počítače působící v síti. Využití této vlastnosti se hodí v případech, kdy chceme přidat realistickou funkcionalitu k vytvářeným serverům a máme dostatečné bezpečnostní mechanismy zajišťující kontrolu dat a zachycení všech informací mezi honeypotem a externím počítačem. Všechny výše zmíněné prostředky nenabízely přímé ohrožení ze strany útočníka, který by mohl do systému proniknout, ovšem za pomoci proxy podsouváme k dispozici skutečné systémy, což může být nebezpečné a zvyšuje potenciální bezpečnostní rizika. Dobrým příkladem využití proxy je použití externího DNS serveru. Pokud je honeypot nastaven jako poskytovatel DNS služeb, bylo by pro útočníka matoucí a podezřelé, proč nemůže použít DNS server pro překlad doménových jmen. Vytvoření skriptu, který by pokrývalo překlad libovolných jmen, které útočník podstrčí, není dost dobře možné, takže řešení proxy je v tomto případě žádoucí.

5.3 Základní konfigurace

V průběhu testování a integrace honeypotu v prostředí domácí sítě byl Honeyd testován na virtuálním stroji s OS Ubuntu 9.10 Karmic Koala za využití VMWare Workstation. Níže uvedené postřehy zachycují některé základní funkční prvky a výstupy, které Honeyd produkuje.

Základní konfigurační soubory se nachází v adresáři `/usr/local/share/honeyd`

Příkaz pro spuštění je následující:

```
sudo honeyd -df testconfig.config -p nmap.prints -l  
/var/log/honeypot/logs.log 192.168.2.10 -x xprobe2.conf -o pf.os -s  
/var/log/honeypot/service.log -i eth0 -c 127.0.0.1:12345:pepino:pepino
```

Text 1- Ukázka spuštění Honeyd s příslušnými parametry

Jednotlivými parametry jsou odkazy na konfigurační soubory vyžadované při startu, tak logování a identifikace pluginu Honeycomb.

Základní konfigurační skript, v našem ukázkovém případě se jedná o `testconfig.conf` je ve tvaru:

```
### Windows computers
create windows
set windows personality "Microsoft Windows 2000 SP2"
set windows default tcp action reset
set windows default udp action reset
add windows tcp port 80 "perl /usr/share/honeyd/scripts/iisemulator-0.95/iisemul8.pl"
add windows tcp port 139 open
add windows udp port 139 open
add windows tcp port 137 open
add windows udp port 137 "python /usr/share/honeyd/scripts/win32/smb137.py"
add windows udp port 135 open
add windows tcp port 135 open
add windows tcp port 110 "sh /usr/share/honeyd/scripts/win32/win2k/exchange-pop3.sh"
add windows tcp port 25 "sh /usr/share/honeyd/scripts/win32/win2k/exchange-smtp.sh"
add windows tcp port 21 "sh /usr/share/honeyd/scripts/win32/win2k/msftp.sh"
add windows tcp port 143 "sh /usr/share/honeyd/scripts/win32/win2k/exchange-imap.sh"
add windows tcp port 5901 "sh /usr/share/honeyd/scripts/win32/win2k/vnc.sh"
add windows tcp port 445 "python /usr/share/honeyd/scripts/win32/smb445.py"
set windows uptime 3284460
bind 192.168.2.11 windows
```

Definice OS (odpovídá nmap fingerprintu)

Výchozí odezvy na daný typ příchozího paketu

Nastavení otevřených portů

Definice servisních skriptů včetně interpretu a umístění

Nastavení doby běhu systému (anglicky uptime)

Svázání (ang. bind) tohoto počítače s konkrétní IP adresou

Obrázek 5 - Konfigurační soubor Honeyd

Po spuštění výše uvedeným příkazem bychom měli získat výstup podobný tomuto:

```
pepino@ubuntu:/usr/local/share/honeyd$ sudo honeyd -df testconfig.config -
p nmap.prints -l /usr/local/share/honeyd/logs/log.txt 192.168.2.10 -x
xprobe2.conf -0 pf.os -s /usr/local/share/honeyd/logs/servicelog.txt -i
eth0 -c 127.0.0.1:12345:pepino:pepino
[sudo] password for pepino:
Honeyd V1.5c Copyright (c) 2002-2007 Niels Provos
honeyd[3272]: started with -df testconfig.config -p nmap.prints -l
/usr/local/share/honeyd/logs/log.txt -x xprobe2.conf -0 pf.os -s
/usr/local/share/honeyd/logs/servicelog.txt -i eth0 -c
127.0.0.1:12345:pepino:pepino 192.168.2.11
Warning: Impossible SI range in Class fingerprint "IBM OS/400 V4R2M0"
Warning: Impossible SI range in Class fingerprint "Microsoft Windows NT
4.0 SP3"
honeyd[3272]: listening promiscuously on eth0: (arp or ip proto 47 or (udp
and src port 67 and dst port 68) or (ip and (host 192.168.2.11))) and not
ether src 00:0c:29:0c:d5:f5
honeyd[3272]: registering plugin 'Honeycomb' (0.7)
honeyd[3272]: Demoting process privileges to uid 65534, gid 65534
```

Text 2 - Ukázka spuštění Honeyd

Za běhu, kdy honeypot navazuje připojení na základě dotazů, je forma výstupu následující:

```
honeyd[3277]: Sending ICMP Echo Reply: 192.168.2.11 -> 192.168.15.99
honeyd[3277]: Sending ICMP Echo Reply: 192.168.2.11 -> 192.168.15.99
honeyd[3277]: Creating new stats buffer for (192.168.15.99:55598 -
192.168.2.11:139)
honeyd[3277]: Connection request: tcp (192.168.15.99:55598 -
192.168.2.11:139)
honeyd[3277]: Creating new stats buffer for (192.168.15.99:55598 -
192.168.2.11:445)
honeyd[3277]: Connection request: tcp (192.168.15.99:55598 -
192.168.2.11:445)
```

```

honeyd[3277]: Connection dropped by reset: tcp (192.168.15.99:55598 -
192.168.2.11:139)
honeyd[3277]: Connection dropped by reset: tcp (192.168.15.99:55598 -
192.168.2.11:445)
honeyd[3277]: Creating new stats buffer for (192.168.15.99:59262 -
192.168.2.11:137)
honeyd[3277]: Connection: udp (192.168.15.99:59262 - 192.168.2.11:137)
honeyd[3277]: Connection established: udp (192.168.15.99:59262 -
192.168.2.11:137) <-> python
/usr/local/share/honeyd/scripts/win32/smb137.py
honeyd[3277]: E(192.168.15.99:59262 - 192.168.2.11:137): python: can't
open file '/usr/local/share/honeyd/scripts/win32/smb137.py': [Errno 2] No
such file or directory
honeyd[3277]: Creating new stats buffer for (192.168.15.99:5974 -
192.168.2.11:445)
honeyd[3277]: Connection request: tcp (192.168.15.99:5974 -
192.168.2.11:445)
honeyd[3277]: Connection established: tcp (192.168.15.99:5974 -
192.168.2.11:445) <-> python
/usr/local/share/honeyd/scripts/win32/smb445.py
honeyd[3277]: writing stats of length 201

```

Text 3 - Výpisy spojení za běhu Honeyd

5.4 Evidence uskutečněných aktivit

Honeypoty zaznamenávají informace o provedených činnostech a spojeních, která byla provedena v průběhu jejich běhu. Na základě těchto informací je pak možné zpětně analyzovat, které aktivity se útočník na systému snažil provádět, kterých nástrojů využíval, na které služby byl útok směřován a tímto způsobem analyzovat celý bezpečnostní incident. K tomuto účelu slouží právě evidence aktivit, neboli logování a sběr dat s ním spojeným.

Logování Honeyd je značně omezené, což částečně vychází z myšlenky, že se jedná pouze o honeypot s nízkou mírou interakce, tak i použitého implementačního řešení. Výstupy a logy, které Honeyd poskytuje jsou v podstatě trojího typu. Prvním z nich jsou hlášení, která jsou produkována na standardní výstup, nesoucí informace týkající se spuštění honeyd, informace o síťových připojeních, která jsou honeypotem zaznamenána, informace o spouštění obslužných servisních skriptech pro definované porty, případně některé chybové hlášení. Ukázka tohoto typu výstupu je znázorněna v Text 5 - Ukázka honeyd servisního logu .

Druhým a stěžejním logem je log spojení, který je aktivován a specifikován parametrem - l. Ve výchozím nastavení je vytvářen soubor honeyd.log, ale je možné specifikovat jméno dle vlastních požadavků. Žádný z logů ovšem neposkytuje bližší informace o zachycených paketech ze spojení, případně jejich obsah. Log obsahuje pouze strohé údaje o uskutečněném spojení.

```

2010-03-28-17:03:35.5572 honeyd log started -----
2010-03-28-17:03:43.2610 icmp(1) - 192.168.15.99 192.168.2.20: 8(0): 28
2010-03-28-17:03:43.2638 tcp(6) S 192.168.15.99 38096 192.168.2.20 443
[SunOS 4.1 ]
2010-03-28-17:03:43.2639 icmp(1) - 192.168.15.99 192.168.2.20: 13(0): 40
[SunOS 4.1 ]

```

```

2010-03-28-17:03:43.2649 tcp(6) E 192.168.15.99 38096 192.168.2.20 443: 0
0
2010-03-28-17:03:43.2791 tcp(6) S 192.168.15.99 38096 192.168.2.20 445
[SunOS 4.1 ]
2010-03-28-17:03:43.2796 tcp(6) E 192.168.15.99 38096 192.168.2.20 445: 0
0
2010-03-28-17:03:52.5882 icmp(1) - 192.168.15.99 192.168.2.20: 8(0): 60
[SunOS 4.1 ]
2010-03-28-17:04:53.4326 tcp(6) - 192.168.15.99 56661 192.168.2.11 443: 44
S
2010-03-28-17:22:05.2326 udp(17) - 192.168.15.1 67 255.255.255.255 68: 576
2010-03-28-17:22:05.2336 udp(17) - 192.168.15.1 67 255.255.255.255 68: 576
2010-03-28-18:24:22.9605 honeyd log stopped -----

```

Text 4 - Ukázka honeyd logu

Soubor s logy obsahuje tyto informace (v pořadí dle výskytu), s ohledem na dostupnost informace pro daný typ protokolu:

- Datum a čas události
- Typ protokolu - ICMP, TCP, UDP
- Stav spojení - zahájení (S), konec (E), nemožno určit (-)
- IP adresa zdrojové stanice
- Číslo portu zdrojové stanice
- IP adresa cílové stanice
- Číslo portu cílové stanice
- Celková velikost paketu (payload plus hlavičky)
- Dodatkové informace (TCP flags - S = SYN, A = ACK; odhad zdrojového OS, ...)

Posledním z trojice uvedených logů je servisní log. Zobrazuje pouze informace o spuštění/zastavení honeyd, případně o výskytech chybových stavů, jako je nedostupnost některého ze servisních skriptů a podobně.

```

2010-03-27-21:03:13.4632 honeyd log started -----
2010-03-27-21:04:48.6858 udp(17) 192.168.15.99 52983 192.168.2.11 137:
|python: can't open file
|'/usr/local/share/honeyd/scripts/win32/smb137.py': [Errno 2] No such file
or directory|
2010-03-27-21:04:56.5276 udp(17) 192.168.15.99 58467 192.168.2.11 137:
|python: can't open file
|'/usr/local/share/honeyd/scripts/win32/smb137.py': [Errno 2] No such file
or directory|
2010-03-27-21:23:05.5641 honeyd log stopped -----

```

Text 5 - Ukázka honeyd servisního logu

5.5 Rozšiřující a podpůrné nástroje Honeyd

5.5.1 Honeydstats

Honeyd sbírá údaje o provedených pokusech útoků na služby a eviduje je do logů. Jelikož těchto logů může být obsáhlé množství a v rámci výzkumného nasazení se nemusí jednat jen o jednu instanci Honeyd, tudíž jeden zdroj logů, je získání přehledu o množství útoků a informací o nich pro člověka poměrně komplikované.

Z tohoto důvodu byl vyvinut nástroj Honeydstats, který centralizuje zpracování těchto evidenčních souborů a poskytuje výsledky ve formě, která je pro uživatele přijatelná. Statistiky jsou evidovány ve 4 kategoriích:

- *Verze operačních systémů:* Použitím pasivního sledování otisků Honeydstats eviduje počty výskytů jednotlivých OS, které se připojovaly k Honeyd v poslední minutě, hodině, dni
- *Cílové porty:* Uvádí celkový počet připojení na jednotlivé porty. Na základě těchto statistik jsme schopni odhadnout, které typy útoků byly prováděny vůči honeypotu.
- *Zeměpisný report:* Za použití reverzního DNS získá vrcholovou doménu a určí lokaci, ze které byl kontakt s honeypotem proveden. Tyto údaje nabývají značné nepřesnosti, ale poskytují alespoň přibližnou představu.
- *Spam report:* Analyzuje datový provoz SMTP na portu 25 a eviduje nejčastěji používané IP adresy. Pro korektní funkčnost je potřeba mít nastavenou službu pro odezvy emailových služeb.

Ukázkový příkaz spuštění:

```
honeydstats -p 41000 -f /usr/local/share/honeyd/honeydstats/stats.config  
--os_report /var/log/honeypot/honeydstats/os.txt --port_report  
/var/log/honeypot/honeydstats/port.txt --spammer_report  
/var/log/honeypot/honeydstats/spammer.txt --country_report  
/var/log/honeypot/honeydstats/country.txt -c  
/var/log/honeypot/honeydstats/checkpoint.txt -l localhost -p 12345 -d
```

Text 6 - Parametry spuštění honeydstats

Operating System Statistics			
SunOS 4.1 :	0	1	1
Destination Port Statistics			
0:	0	1	1
256:	0	1	1
512:	0	1	1
768:	0	1	1
1024:	0	1	1
2048:	0	1	1
5120:	0	1	1
8192:	0	1	1
32768:	0	1	1
49152:	0	1	1
1:	0	1	1
257:	0	1	1
513:	0	1	1
769:	0	1	1
1025:	0	1	1
Spammer Address Statistics			
Country Activity Statistics			
unknown:	0	1	1

Text 7 - Ukázka výpisu z honeydstats

5.5.2 Honeydctl

Jedná se o zkratku zastupující Honeyd control, který je utajeným vstupem do běžícího Honeyd. Tímto způsobem lze za běhu měnit konfiguraci a vytvářet nové honeypoty pro přizpůsobení aktuální situaci. Lze přidávat/odebírat honeypoty, měnit obslužné servisní skripty a služby a podobně. To vše je řešeno pomocí konzole.

5.5.3 Honeycomb

Honeycomb je dalším z Honeyd pluginů. Aplikuje analýzu datového toku, protokolů a detekci vzorů a automaticky generuje signatury pro IDS systém jako je Snort a podobně. Pokud vezmeme v potaz to, že se jedná o analýzu datového toku z honeypotu, která je již z definice podezřelá, tak můžeme s poměrně vysokou pravděpodobností očekávat, že bude obsahovat škodlivý kód. Jednou z výhod je rozpoznání signatur červů.

5.5.4 Honeydsum

Honeydsum je analyzátor Honeyd logů, který dokáže generovat sumarizační výstupy v textové podobě. Výstupy mohou být filtrovány podle specifické IP adresy, portu, protokolu nebo sítě. Zobrazuje zdrojové IP adresy s nejčastějším výskytem připojení, porty a počty připojení za hodinu. Podporuje taktéž současné zpracování několika vstupních log souborů, které mohou pocházet z různých zdrojů.

Spuštění honeydsum:

```
sudo ./honeydsum.pl -c honeydsum.conf /var/log/honeypot/logs.log > out.txt
```

Text 8 - Parametry spuštění honeydsum

```
|
### Honeydsum's Configuration ###

-----
Connection Counter
-----
Total:      23
  TCP:      23
  UDP:       0
  ICMP:     0
-----

Honeydsum: 192.168.2.10

-----
Source IP      Resource  Connections
192.168.2.99   21/tcp    3
192.168.2.105  21/tcp    7
192.168.2.106  21/tcp   13
-----
IPs            Resources  Connections
3              1         23
-----

Top 10 Source Hosts

Rank Source IP      Connections
1    192.168.2.106   13
2    192.168.2.105   7
3    192.168.2.99    3

Top 10 Accessed Resources

Rank Resource      Connections
1    21/tcp         23

Connections per Hour

Hour  Connections
00:00 0
01:00 0
02:00 0
03:00 0
04:00 0
05:00 0
06:00 0
```

Obrázek 6 - Ukázka výpisu z honeydsum (přeformátováno)

6 Emulace Confickeru v Honeyd

Servisní skripty tvoří základ flexibility honeypotu Honeyd. Díky nim je konfigurováno spektrum služeb, na které může být prováděn útok. Jednou z komponent, která byla využita při výsledném nasazení Honeyd honeypotu do sítě pro získávání a analýzu aktuálních síťových hrozeb je servisní skript pro Honeyd, který dokáže emulovat zranitelnost MS08-067 Windows systémů, kterou zneužívá červ Conficker blíže popsány v kapitole 2.3 Cílem vytvoření a nasazení skriptu je snaha o zvýšení atraktivity cíle útočníkovi. V této kapitole se budeme zabývat způsobem a úskalími spojenými s implementací tohoto skriptu.

6.1 Detekce zranitelnosti nástrojem nmap

Program nmap (název je zkratkou Network mapper) [26] je nástrojem určeným k získávání informací o počítačích v síti a bezpečnostnímu auditu. Patří k předním nástrojům užívaným systémovými administrátory k monitorování sítě, ověřování dostupnosti stanic, určování typu použitého operačního systému, analýze dostupných portů a dalších údajů. Nmap je open-source řešení dostupné pro systémy Windows, Linux, Mac OS a další. Jedná se o konzolové řešení, kde mezi jeho hlavní přednosti patří jednoduchost, značná flexibilita, dobrá dokumentace a rozvoj podpůrných skriptů.

Mezi zmíněné skripty bezesporu patří testovací skript označený jako smb-check-vulns [27], který slouží k odhalování zranitelnosti MS08-067 na systémech Windows a umožňuje i hlubší analýzu a odhalení možného infikování červem Conficker. Právě tento skript se stal předlohou pro následně implementovaný servisní skript Honeyd, který dokáže zasílat pozitivní odpovědi na dotazy programu nmap, zda systém obsahuje zranitelnost, případně je již infikován.

Skript smb-check-vulns se spouští ve tvaru podobném tomuto :

```
nmap -PN -T4 -p139,445 -n -v --script smb-check-vulns 192.168.2.10
```

Text 9 - Parametry spuštění nmap skriptu

Skript samotný, pokud není rozšířen o další vstupní parametry, probíhá ve dvou fázích. V první fázi je detekována přítomnost zranitelnosti MS08-067, v druhé pak detekce Confickeru.

Raw packets sent: 2 (88B) | Rcvd: 2 (120B)

Skript funguje na tradičním principu klientského dotazování a odpovědi ze strany serveru. Pro bližší představu byl datový tok komunikujících stran zachycen nástrojem Wireshark [<http://www.wireshark.org/>], kde část zachycení komunikace je zobrazena na obrázku Obrázek 7.

No.	Time	Source	Destination	Protocol	Info
15204	26.450777	192.168.15.99	192.168.15.1	DNS	Standard query PTR 120.15.168.192.in-addr.arpa
15207	26.452996	192.168.15.1	192.168.15.99	DNS	Standard query response, No such name
15212	26.459722	192.168.15.99	192.168.15.120	TCP	33182 > microsoft-ds [SYN] Seq=0 win=2048 Len=0 MSS=1460
15215	26.460404	192.168.15.120	192.168.15.99	TCP	microsoft-ds > 33182 [ACK] Seq=0 Ack=1 win=65535 Len=0 MSS=1460
15222	26.470142	192.168.15.99	192.168.15.120	NBNS	Name query NBSTAT <00<<00<<00<<00<<00<<00<<00<<00<<00<
15224	26.470992	192.168.15.120	192.168.15.99	NBNS	Name query response NBSTAT
15225	26.472928	192.168.15.99	192.168.15.120	TCP	brcd > microsoft-ds [SYN] Seq=0 win=8192 Len=0 MSS=1460 WS=2
15227	26.473997	192.168.15.99	192.168.15.99	TCP	microsoft-ds > brcd [SYN, ACK] Seq=0 Ack=1 win=65535 Len=0 MSS=1460 WS=2
15228	26.474063	192.168.15.99	192.168.15.120	TCP	brcd > microsoft-ds [ACK] Seq=1 Ack=1 win=65700 Len=0
15230	26.475836	192.168.15.99	192.168.15.120	SMB	Negotiate Protocol Request
15232	26.477009	192.168.15.120	192.168.15.99	SMB	Negotiate Protocol Response
15235	26.479035	192.168.15.99	192.168.15.120	SMB	Session Setup AndX Request, NTLMSSP_NEGOTIATE
15237	26.480930	192.168.15.120	192.168.15.99	TCP	microsoft-ds > 33182 [RST, ACK] Seq=0 Ack=1 win=0 Len=0
15238	26.480992	192.168.15.120	192.168.15.99	SMB	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCESSING_REQUIRED
15240	26.482934	192.168.15.99	192.168.15.99	SMB	Session Setup AndX Request, NTLMSSP_AUTH, User: \guest
15244	26.485990	192.168.15.120	192.168.15.99	SMB	Session Setup AndX Response, Error: STATUS_LOGON_FAILURE
15247	26.488198	192.168.15.99	192.168.15.120	SMB	Session Setup AndX Request, NTLMSSP_NEGOTIATE
15249	26.488993	192.168.15.120	192.168.15.99	SMB	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCESSING_REQUIRED
15251	26.491146	192.168.15.99	192.168.15.120	SMB	Session Setup AndX Request, NTLMSSP_AUTH, User: \
15253	26.492099	192.168.15.120	192.168.15.99	SMB	Session Setup AndX Response
15257	26.495233	192.168.15.99	192.168.15.120	SMB	Tree Connect AndX Request, Path: IPC\$
15258	26.495982	192.168.15.120	192.168.15.99	SMB	Tree Connect AndX Response
15261	26.498300	192.168.15.99	192.168.15.99	SMB	NT Create AndX Request, Path: \\BROWSER
15263	26.499930	192.168.15.120	192.168.15.99	SMB	NT Create AndX Response, FID: 0x0400
15265	26.500758	192.168.15.99	192.168.15.120	DCERPC	Bind: call id: 109479558 srvsvc V3.1
15268	26.502987	192.168.15.120	192.168.15.99	SMB	Write AndX Response, FID: 0x0400, 72 bytes
15269	26.504626	192.168.15.99	192.168.15.120	SMB	Read AndX Request, FID: 0x0400, 2048 bytes at offset 0
15271	26.504986	192.168.15.120	192.168.15.99	DCERPC	Bind_ack: call id: 109479558 accept max_xmit: 2048 max_recv: 2048
15275	26.522445	192.168.15.99	192.168.15.120	SRVSVCSVC	NetPathCompare request
15288	26.528198	192.168.15.120	192.168.15.99	SMB	Write AndX Response, FID: 0x0400, 204 bytes
15291	26.526059	192.168.15.99	192.168.15.120	SMB	Read AndX Request, FID: 0x0400, 4097 bytes at offset 0
15292	26.527006	192.168.15.120	192.168.15.99	SRVSVCSVC	NetPathCompare response
15296	26.530573	192.168.15.99	192.168.15.120	SMB	Tree Disconnect Request
15297	26.531998	192.168.15.120	192.168.15.99	SMB	Tree Disconnect Response
15300	26.533547	192.168.15.99	192.168.15.120	SMB	Logoff AndX Request
15302	26.534090	192.168.15.120	192.168.15.99	SMB	Logoff AndX Response
15303	26.535847	192.168.15.99	192.168.15.120	TCP	brcd > microsoft-ds [FIN, ACK] Seq=1442 Ack=1090 len=6408 Len=0
15305	26.536589	192.168.15.99	192.168.15.120	TCP	delta-mcp > microsoft-ds [SYN] Seq=0 win=8192 Len=0 MSS=1460 WS=2
15306	26.536994	192.168.15.120	192.168.15.99	TCP	microsoft-ds > brcd [FIN, ACK] Seq=1090 Ack=1443 win=64094 Len=0
15308	26.537037	192.168.15.99	192.168.15.120	TCP	brcd > microsoft-ds [ACK] Seq=1443 Ack=1091 win=6408 Len=0
15310	26.538198	192.168.15.120	192.168.15.99	TCP	microsoft-ds > delta-mcp [ACK] Seq=0 Ack=1 win=65535 Len=0 MSS=1460 WS=0
15311	26.538092	192.168.15.99	192.168.15.120	TCP	delta-mcp > microsoft-ds [ACK] Seq=1 Ack=1 win=65700 Len=0

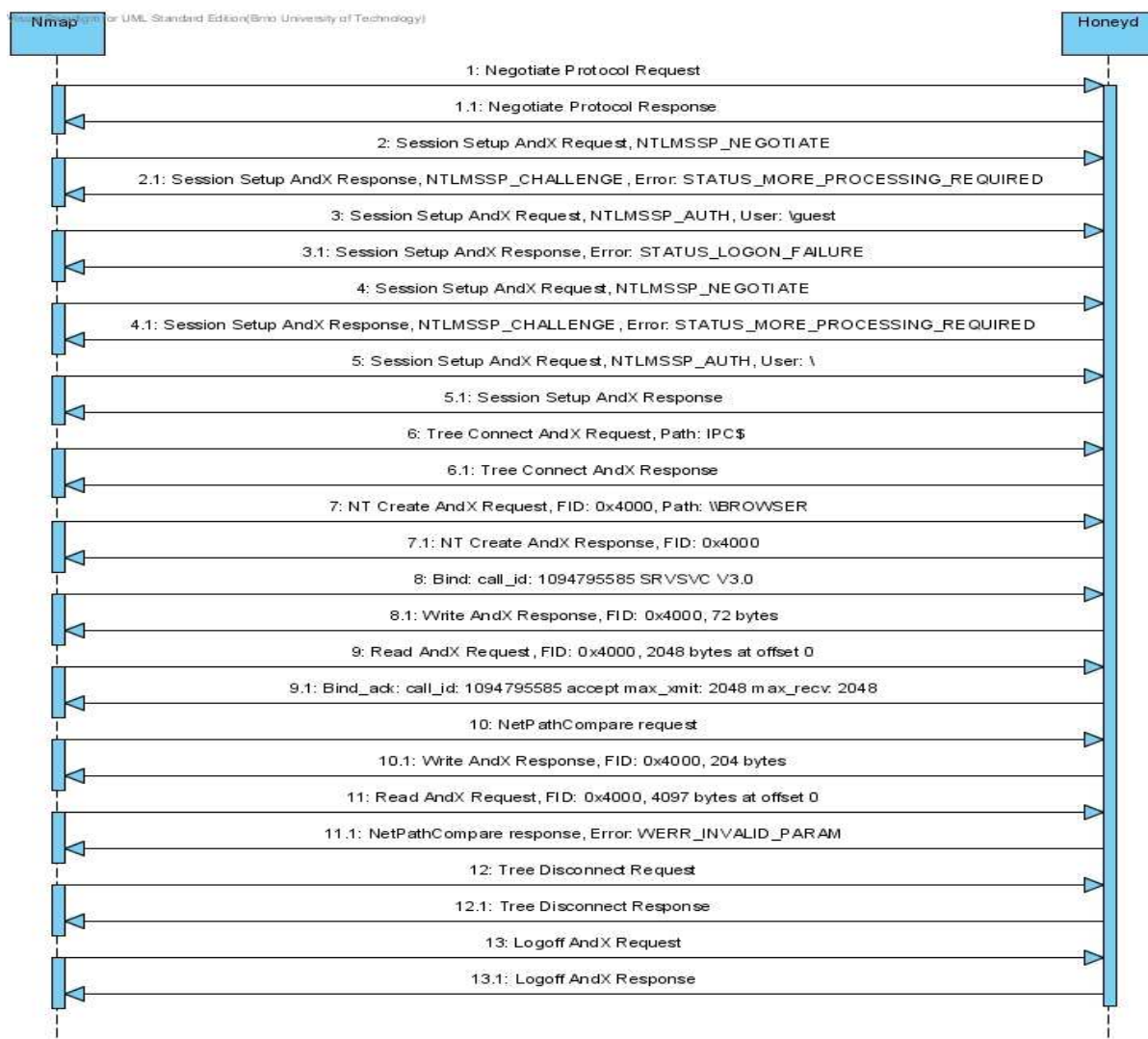
6.2 Postup tvorby servisního skriptu

Jako vhodný jazyk pro implementaci servisního skriptu Honeyd simulujícího zranitelnost MS08-067 byl vybrán Python. Zranitelnost využívá bezpečnostní chyby v Microsoft Server service (SAMBA), který je značně rozsáhlý, obsahuje velké množství subsystémů a provázaností s dalšími komponentami Windows. To je jeden z hlavních důvodů, proč není možné implementovat kompletní emulaci tohoto systému. Dostupné servisní skripty Honeyd poskytují pouze částečné emulace služeb

(FTP, telnet, IIS...) nebo poskytují emulaci služeb specifického účelu. Analýzou dostupných zdrojů na Internetu nebyly nalezeny žádné směrodatné podklady vedoucí k možnosti implementovat simulaci složitějších systémů a rutin.

Většina ze současně dostupných implementovaných servisních skriptů je interaktivní (FTP, telnet), což znamená, že na uživatelem zadaný terminálový vstup poskytnou odpověď. Modul Honeyd, který zprostředkovává most mezi jádrem Honeyd a vstupy/výstupy servisních skriptů umožňuje přístup pouze k datové části (ang. payload) paketu. O správu hlaviček a korektní znovu seskládání paketu se stará jádro samotné. Tento postup byl navržen s ohledem na zachování vlastnosti fingerprintingu a mimikování OS, přináší tak, co se psaní rozšiřujících skriptů týče, značné výhody, tak i nevýhody. Mezi výhody patří zjednodušení pro vývojáře, který se nemusí zabývat úpravou paketových hlaviček, o to vše zajišťují mechanismy Honeyd samostatně. Nevýhoda vzniká v okamžiku, kdy potřebujeme implementovat skript, který nebude interaktivní – tzn. uživatel nebude zadávat dotazy na server konzolovým vstupem. Podrobnější analýzou bylo zjištěno, že vzniká problém identifikace konce znakové posloupnosti na vstupu (EOL, EOF). S pomocí standardních technik nejsme schopni určit délku vstupní znakové posloupnosti a její konec.

Implementace skriptu simulující MS08-067 zranitelnost probíhala na bázi rozboru datového obsahu paketů, které kladl jako dotazy nástroj nmap. Byla zachycena kompletní síťová komunikace mezi stanicí, kde byl nainstalován nástroj nmap a počítačem, který byl infikován Confickerem. Jelikož struktura a pořadí dotazů pocházející se smb-check-vulns skriptu je pořád stejná, liší se jen v závislosti na odpovědích dotazovaného systému, bylo možné vyhledat podobnosti v podřetězcích a zpětně rekonstruovat odpovědi, které musí simulační skript poskytovat. Posloupnost vzájemné komunikace je zachycena na obrázku Obrázek 8.



Obrázek 8 - Sekvence dotazů první části nmap skriptu a Honeyd

Na základě získaného rozboru datového toku na úrovni paketů, bylo možné zpětně zrekonstruovat odpovědi, které zasílá infikovaný stroj. Jak již bylo řečeno výše, jádro Honeyd poskytuje komponentě zajišťující zpracování servisních skriptů pouze datový obsah (payload) paketů. Analýzou celkové komunikace byly nalezeny unikátní sekvence v rámci payloadu, které dokážou rozlišit, o kterou část komunikace se jedná. Situaci znázorňuje Obrázek 9.

No.	Time	Source	Destination	Protocol	Info
1368	2010-05-12 21:37:02.207619	192.168.15.99	192.168.15.102	TCP	jt400 > microsoft-ds [SYN] Seq=0 win=8192 Len=0 MSS=1460 WS=2
1369	2010-05-12 21:37:02.207809	192.168.15.102	192.168.15.99	TCP	microsoft-ds > jt400 [SYN, ACK] Seq=0 Ack=1 win=65535 Len=0 MSS=1460
1370	2010-05-12 21:37:02.208017	192.168.15.99	192.168.15.102	TCP	jt400 > microsoft-ds [ACK] Seq=1 Ack=1 win=65700 Len=0
1372	2010-05-12 21:37:02.210205	192.168.15.99	192.168.15.102	SMB	Negotiate Protocol Request
1373	2010-05-12 21:37:02.210555	192.168.15.102	192.168.15.99	SMB	Negotiate Protocol Response
1375	2010-05-12 21:37:02.213677	192.168.15.99	192.168.15.102	SMB	Session Setup AndX Request, NTLMSSP_NEGOTIATE

Frame 1372 (107 bytes on wire, 107 bytes captured)

Ethernet II, Src: Micro-St-50:74:75 (00:19:db:60:74:75), Dst: Dfi-d2:ea:38 (00:01:29:d2:ea:38)

Internet Protocol, Src: 192.168.15.99 (192.168.15.99), Dst: 192.168.15.102 (192.168.15.102)

Transmission Control Protocol, Src Port: jt400 (3470), Dst Port: microsoft-ds (445), Seq: 1, Ack: 1, Len: 53

NetBIOS Session Service

SMB (Server Message Block Protocol)

```

0000  00 01 29 d2 ea 38 00 19 db 60 74 75 08 00 45 00  ..).8..`tu..E.
0010  00 5d 42 f0 40 00 80 06 17 91 c0 a8 0f 63 c0 a8  ..]B.@.....C.
0020  0f 66 0d 8e 01 bd d4 4f 72 d0 ee ab fe 1a 50 18  .f.....O"....P.
0030  40 29 ba a7 00 00 00 00 00 00 31 ff 53 4d 42 72 00  @).....[.].SMBR.
0040  00 00 00 18 45 68 00 00 00 00 00 00 00 00 00 00  ....Eh.....
0050  00 00 00 00 00 00 00 00 00 00 00 00 0e 00 02 4e 54  .........NT
0060  20 4c 4d 20 30 2e 31 32 00 02 00  ..LM 0.12 ...

```

Unikátní podřetězec v rámci všech paketů spojení

Obsah paketu bez hlaviček filtrovaný Honeyd

V Text 11 je prezentována upravená ukázka zdrojového kódu servisního skriptu. Úvodní část obsahuje posloupnost příchozích požadavků, unikátních podřetězců a odpovídajících odpovědí. Samotné zpracování funguje na bázi srovnávání příchozí posloupnosti, ověření, zda odpovídá požadovanému podřetězci a předpokládanému pořadí. Pokud ano, tak je jako odpověď na dotaz programu nmap zaslán zrekonstruovaný paket s obdobným payloadem jako je ten, který byl zachycen v úvodní části od skutečně infikovaného počítače.

```
#1 - Negotiate Protocol Request
x1 =
"00000031ff534d427200000000184568000000000000000000000000000000e00024e
54204c4d20302e3132000200"
x1_match = "184568"

#2 - Negotiate Protocol Response
x2 =
"00000055ff534d42720000000098456800000000000000000000000000000000000110000030a
000100041100000000010000000000fde300802e83d7cd0e9cca01c4fff00100059f163facdf33845847
d947a72e90c22"

#3 - Session Setup AndX Request, NTLMSSP_NEGOTIATE
x3 =
"0000006fff534d4273000000001845680000537b733454c0745a00000000000000000000cff000000
ffff01000000000000000200000000005000008034004e544c4d535350000100000011820000000000
00000000000000000000000004e6d6170004e6174697665204c616e6d616e0000"
x3_match = "6fff53"
```

```

#4 - Session Setup AndX Response, NTLMSSP_CHALLENGE, Error:
STATUS_MORE_PROCESSING_REQUIRED
x4 =
"00000088ff534d4273160000c09845680000537b733454c0745a00000000000000008000004ff008800
000038005d004e544c4d53535000020000000000000000380000001182000248ed7de276fb593d0000000
0000000000000000380000000501280a0000000f57696e646f777320352e310057696e646f77732032
303030204c414e204d616e6167657200"

...
...
...
request = ''
step = 1
run = 1
counter = 0

while 1:
    next = sys.stdin.read(1)
    counter += 1
    request += str(next.encode("hex"))
    if (string.find(request, x1_match) > -1) and (step == 1):
        sys.stdout.write(x2.decode("hex"))
        sys.stdout.flush()
        request = ''
        step = 3
        continue
    elif (string.find(request, x3_match) > -1) and (step == 3):
        sys.stdout.write(x4.decode("hex"))
        sys.stdout.flush()
        request = ''
        step = 5
        continue
    ...

```

Text 11 - Ukázka kódu servisního skriptu (upraveno, zkráceno)

7 Automatizovaný nástroj detekce útoků

Hlavní tvůrčí myšlenkou diplomové práce bylo vytvoření automatizovaného nástroje pro analýzu a sběr informací o síťových útocích provedených na honeypot. Tento nástroj by měl rozšířit škálu dostupných prostředků a výstupů, které dokážeme za použití stávajících řešení získat. V dřívějších částech bylo již zmíněno, že existuje velká řada typů honeypotů a s tím spojená možnost jejich interakce, spektrum nabízených služeb a výstupních informací, které produkují. Honeyd jako představitel honeypotů s nízkou mírou interakce dokáže emulovat některé ze služeb, ovšem jejich počet, věrnost chování vůči neemulovaným řešením a množství výstupů je značně omezené. Zaměříme se zde na možnost využití dalších současně dostupných nástrojů, které jsou využívány pro potřebu analýzy a získávání dat týkajících se síťových útoků a jejich propojení do zjednodušeného automatizovaného procesu.

7.1 Požadavky automatizace a výstupů

Stěžejním požadavkem je implementace skriptu, který bude běžet a koexistovat s honeypotem Honeyd a dokáže sledovat podezřelou činnost na síti, zaznamenávat a analyzovat spojení, která byly uskutečněna s nasazeným systémem. Skript bude vytvořen v libovolném skriptovacím jazyce, je možné využít libovolných volně dostupných a volně šiřitelných nástrojů. Bude schopen pracovat na reálné bázi a výsledky ukládat do vhodného datového úložiště, pro další možné zpracování a analytické vyhodnocení. Důraz je kladen i na možnost získání co největší úrovně detailu o jednotlivých připojeních a datových tocích (ang. streamech), které byly s honeypotem navázány.

Očekávaným výsledkem budou vytvořené metriky a výstupní hodnoty popisující zachycený bezpečnostní incident. Metriky budou sloužit k detailní analýze provedeného útoku, zjištění o jaký útok se jedná, přičemž výsledná data budou uložena pro následné automatizované vyhodnocení.

7.2 Podpůrné implementační nástroje

7.2.1 Perl

Perl je interpretovaný programovací jazyk, koncem 80. let 20. století. Perl poskytuje výkonné zpracovávání textu, je určen pro systémovou administraci, programování sítí, grafické výstupy, tak

jako práci s databázovými systémy, CGI a web technologiemi. K jeho hlavním výhodám patří velká uživatelská základna a velká řada volně dostupných modulů třetích stran v archivu CPAN.

Perl je svobodný software pod GNU licencí pracující na naprosté většině současně dostupných platform. Snadné a jednoduché použitelnosti přispívá existence hotových modulů, snadná práce s textem a regulárními výrazy a jeho rychlost a celková jednoduchost. Domovská stránka perlu se nachází na [<http://www.perl.org/>].

7.2.2 Tcpcdump

Tcpcdump je známý nástroj pro analýzu sítí a síťového provozu (paketů), fungující v textovém prostředí příkazové řádky. Jako nejstarší analyzátor a sledovací nástroj sítě (ang. sniffer) byl vytvořen v roce 1987 Van Jacobsonem, Craigem Leresem a Stevem McCannem ve výzkumné skupině počítačových sítí na v Lawrence Berkley laboratořích. Jedná se o volně šiřitelný software distribuovaný pod BSD licencí. Je rozšířen na unixové platformy, tak i na OS Windows. Tcpcdump je starší alternativou dnes rozšířeného nástroje Wireshark [<http://www.wireshark.org/>], který skýtá v základu prakticky totožné možnosti využití, ovšem s rozšířením grafického uživatelského prostředí. Domovská stránka projektu tcpcdump je nachází na [<http://www.tcpcdump.org/>].

Hlavní využitelnost tcpcdump spočívá v jeho schopnosti zachycovat síťový provoz na síťovém rozhraní počítače. Dokáže zachytávat pakety v podobě hrubých (ang. raw) dat, případně je transformovat do uživatelsky přívětivější a čitelnější textové formy v ASCII nebo hexadecimálně, ukládat data do souborů, pracovat s celou řadou známých a používaných protokolů, tak i využití velké škály filtrů a výstupních informací o trafficu. Tcpcdump je postaven na knihovně pcap¹⁴ a formát parametrů a zpracovaných dat se stal standardem pro celou řadu nástrojů. Manuálové stránky projektu a informace týkající se možných filtrů, úrovně detailu výstupů a mnoho dalších se nachází na [http://www.tcpcdump.org/tcpcdump_man.html].

7.2.3 Tcpslice

Tcpslice je nástrojem systémové administrace tvořící nadstavbu nad knihovnou pcap. Čte a zpracovává zachycený hrubý paketový tok ze souborů dříve získaných pomocí tcpcdump -w. Jeho primárním užitím je extrahování specifických sekvencí na základě časových značek, případně pak vytváření souvislých souborů vytvořených spojením několika požadovaných souborů, vytvořených pomocí tcpcdump (dále jen dump souborů). Manuálové stránky projektu lze nalézt na [<http://linux.die.net/man/8/tcpslice>].

¹⁴ Knihovna pcap (packet capture) se skládá z aplikačního programového interface (API), určeného k zachycování síťové komunikace. Jedná se o funkce pro vytváření síťových socketů, filtrování toku, rozbor paketů, parsování hlaviček paketů, ukládání a další zpracovávání zachycených dat. U Unixových systémů je knihovna dostupná v rámci libpcap, na Windows pak portovaná a známá jako WinPcap.

7.2.4 Tshark

Tshark je konzolovou verzí programu Wireshark. Jedná se taktéž o síťový analyzátor se schopností zachytávat hrubý síťový provoz, číst pakety z dříve zachycených souborů, dekodování dat a využití filtrů a různých analytických nástrojů. Využívá formátu libpcap, který je využit v tcpdump. K jeho výhodám ve srovnání s tcpdump je dostupnost novějších verzí programu a užitečná rozšíření zejména v oblasti filtrovacích procedur. Je vydáván jako součást distribučního balíčku Wireshark, manuálové stránky jsou dostupné na [<http://www.wireshark.org/docs/man-pages/tshark.html>].

7.2.5 Tcptrace

Tcptrace je nástroj vytvořený Shawnem Ostermannem na Univerzitě Ohio. Je primárně určený k analýze TCP dump souborů. Produkuje informace o jednotlivých připojeních, jako je čas události, trvání, komunikující adresy a porty, RTT hodnoty, propustnost a další. Kromě výstupů v textové formě dokáže zpracovávat i grafické reporty pro následnou analýzu. Restrikcí je zaměření na TCP datové streamy, informace o UDP jsou v omezené míře. Jiné typy protokolu nejsou podporovány. Domovská stránka projektu se nachází na [<http://www.tcptrace.org/>].

7.2.6 Snort

Snort je zástupce open-source řešení z oblastí IDS (network intrusion detection system) a IPS (network intrusion prevention system). Patří do kategorie IDS založených na pravidlech. Hledá vzorky známých útoků a v případě nalezení je schopen provádět různé akce bez přerušení sledování probíhajícího síťového provozu. Snort je široce používanou technologií s miliony uživatelů, je udáván jako jeden ze standardů v oblasti IPS. Je vyvíjen jak pro Unix/Linuxová řešení, tak i pro OS Windows. Domovská stránka se nachází na [<http://www.snort.org/>].

Snort je schopen pracovat v několika režimech a plnit částečně odlišnou funkci - dokáže pracovat zjednodušeně ve sniffer módu a pouze zachytávat provoz na síťovém rozhraní. Jeho hlavní síla spočívá však v IDS funkcionalitě. K tomu využívá řadu zásuvných modulů, preprocesorů a systému logování a výstrah. Využívá základnu definovaných pravidel, která může být dle potřeb libovolně upravována daným požadavkům.

7.2.7 MySQL

MySQL je databázový systém vytvořený švédskou firmou MYSQL AB. Jedná se o multiplatformní databázi komunikující pomocí jazyka SQL. MySQL nyní spadá pod společnost Oracle. MySQL je relační databázový systém. Jedná se o open-source řešení, kde zdrojový kód je dostupný pro GNU licenci.

MySQL je relační databázový systém typu DBMS (database management system). V současné době je k dispozici ve verzi 5. Za jeho oblíbeností stojí častá koexistence s webovým serverem Apache, jazykem PHP a dalšími webovými technologiemi. Výhodou je i značná provázanost a podpora rozhraní ze strany dalších jazyků, jako je v našem případě Perl. Domovská stránka a více informací se nachází na[<http://www.mysql.com/>].

7.3 Omezující prvky použitých nástrojů

Jak již bylo zmíněné v kapitole 5.4, logování Honeyd je omezené a výstupní informace pouze hrubé a pro podrobnější analýzu nepostačující. Hlavní log událostí nese pouze informace ve tvaru datum spojení, typ protokolu, stav spojení, zdrojová adresa, zdrojový port, cílová adresa, cílový port a některé dodatečné poznámky. Množství obsažené informace se liší podle typu protokolu. Zjednodušeně lze říci, že eviduje pouze prvotní interval spojení, ovšem detailnější popis a metriky není možné získat.

Druhým omezujícím elementem je evidovaný čas události. Dle informací autora se jedná o čas, kdy spojení je evidováno a zpracováno Honeyd. Problém je v tom, že tato hodnota se liší od času, který poskytují nástroje typu tcpdump, který odvozuje čas od příchodu paketu na síťové rozhraní. Čas uvedený v logu honeyd je mírně posunut dopředu (ve smyslu, že časová hodnota je blíže času současnému). Tato nepřesnost se pohybuje v řádech setin sekundy, ale pro další analytické a statistické zpracování přináší některé problémy.

Primární určení programu tcpdump ve vytvořeném automatizovaném skriptu je pouze zachytávání hrubého datového toku (ang. traffic) a jeho ukládání do souborů. Z důvodu popsaného výše, týkajícího se způsobené časové odchylky, není možné přímo srovnávat počátek času spojení s údajem z Honeyd. Jelikož spojení je identifikovatelné i na základě dalších parametrů (zdrojová/cílová adresa, porty, typ protokolu), je možné za použití uživatelem stanovené časové odchylky stanovit rozmezí výskytu události. Za pomoci filtru a vstupních parametru programu tcpdump pak lze určit, který paket je úvodním paketem spojení s velmi vysokou přesností.

Při sledování kompletního datového toku na daném síťovém rozhraní může docházet k velké vytíženosti linky a následně neschopnosti zpracování všech požadovaných paketů. Z tohoto důvodu může docházet ke ztrátě paketů, které mohou být zahazovány a vzniká pak jejich absence v dump souborech. Z tohoto důvodu není možné dohledat a přiřadit vždy kompletní informace a příslušnost paketů daného streamu, čímž mohou vznikat určité nepřesnosti.

Využití tcpslice je v našem případě za účelem spojování více dump souborů do jediného. Tcpdump dokáže rotovat výstupní soubory na základě určené časové periody nebo dosažení určitého množství zachycených dat. V našem případě je to na základě časového intervalu. Jedním z hlavních problémů, se kterým se automatizační skript potýká, je nemožnost předem určit celkovou délku

spojení, které bylo navázáno. Jelikož skript má fungovat na bázi realtime řešení, přináší to určitá úskalí a možné nepřesnosti, způsobené tímto problémem.

Toto omezení se snaží částečně vyřešit spojování více dump souboru, pomocí něhož jsme schopni seskupit datový tok, který byl evidován v určitém, předem definovaném rozmezí od výskytu prvního paketu spojení. Vždy dochází ke spojování tří logů. Řekneme, že R je rotační perioda dump souboru, X je počátek časového rozmezí, které zachycuje daný log, Y je konec tohoto rozmezí. Platí $Y = X + R$. Událost E odpovídá počátku daného síťového spojení. Platí, že $X < E < Y$. Zmiňované spojované logy pak tvoří časový výsek pokrývající rozmezí $(X-R, Y+R)$. Tímto sestavíme datový tok s danou dobou trvání, kde předpokládáme, že skutečná doba trvání požadovaného streamu padne do tohoto rozsahu. Hodnota R by neměla být velká, s ohledem na poskytování rychlých odezev, objemu zpracovaných dat a požadavku na realtime řešení.

Analytická fáze spočívá v identifikaci těch paketu, které přísluší hledanému streamu. Toho dosáhneme pomocí filtru nástroje tshark. Pro TCP jsme schopni zjistit příslušnost přímo, pro UDP na základě porovnání adres a portu. Účelem je vyfiltrovat datový soubor obsahující pouze hrubý datový tok spojení. Ten je zpracován pomocí tcptrace, který ze vstupních dat zpracuje odpovídající metriky spojení.

Snort provádí analýzu nad hrubým paketovým streamem, takže svým způsobem vstup ve formě již vyfiltrovaného spojení, které nás zajímá, je ulehčením. Snort nepřináší v našem užití žádné omezující elementy, ovšem za použitého způsobu, kdy jsou mu předkládány pouze jednotlivé datové streamy bez možnosti ověřit okolní komunikaci nelze postihnout útoky typu sken portů a podobně.

MySQL databáze byla použita s ohledem na možnost bezplatného využití, velkou podporou komunity a implementovaných řešení a modulů pro programovací jazyky. Omezujícím prvkem, který se přímo vztahuje na implementaci automatizačního skriptu je nemožnost ukládat milisekundy jako součást použitého datového typu `datetime`. Z toho důvodu je časový údaj v databázi ukládán jako složenina, tzn. jedno pole tvaru `YYYY-MM-DD HH:MM:SS` a druhé nesoucí údaj o mikrosekundách tvaru `UUUUUU`.

7.4 Popis implementace skriptu

Předpoklady pro korektní a očekávanou funkčnost automatizovaného nástroje analýzy je spuštěný a korektně nastavený honeypot Honeyd a dostupnost souborů nesoucích kompletní data zachyceného datového toku tak, jak je zpracovává a ukládá nástroj tcpdump.

Samotné spuštění skriptu odpovídá příkazu ve tvaru:

```
perl script.pl /home/xgalet02/honeyd/logs/log.txt
```

Text 12 - Příkaz spuštění automatizovaného skriptu

Vstup tvoří logovací soubor Honeyd nesoucí informace o provedených spojeních. Skript pracuje tím způsobem, že postupně čte jednotlivé záznamy, rozlišuje typ spojení podle daného

protokolu, čte obsah zachycených dump souborů a z celkového objemu komunikace filtruje pouze pakety náležící danému spojení. Filtrovaný datový stream je uložen do separátního souboru, nad kterým je následně prováděna analýza nástrojem tcptrace a snort. Získané výstupy jsou zpracovány a výstupní metriky uloženy do databáze. Bližší popis činnosti a obsahu skriptu je naznačen v Text 13 - Zdrojový kód skriptu, pseudokód (upraveno, zkráceno).

```
open(HDLOG, "honeyd.log"); # otevři honeyd log
for (;;) { # nekonečná smyčka
    while ($line = <HDLOG>) { # čti jednotlivé řádky logu
        if ($line =~ /(tcp|udp)/) { # nalezen záznam o TCP/UDP spojení
            proveď parsování informací - čas, ip adresy, porty
        }
        elsif ($line =~ /icmp/) { # nalezen záznam ICMP spojení
            proveď parsování informací - čas, ip adresy, porty
        }
        if (($line !~ /honey/) { # honeyd start/stop řádky nas nezajímají
            ulož rozparsované hodnoty do proměnných
            if ($line =~ /(tcp|udp)/) {
                přesuň staré dump soubory do archivní složky
                vytvoř souvislý dump soubor z posledních 3 souborů
                filtruj pouze spojení, které nás zajímá
                proveď rozbor datového streamu a vytvoř metriky
                proveď analýzu streamu pomocí nástroje snort
                ulož získané výstupy do databáze
            }
            if ($line =~ /icmp/) {
                přesuň staré dump soubory do archivní složky
                vytvoř souvislý dump soubor z posledních 3 souborů
                filtruj pouze spojení, které nás zajímá
                ulož informace o spojení do databáze
            }
        }
    }
    čekej než se objeví nové záznamy v logu
}
```

Text 13 - Zdrojový kód skriptu, pseudokód (upraveno, zkráceno)

7.5 Výstupní metriky

Jak již bylo popsáno výše, výstupní hodnotou skriptu jsou informace a metriky získané analýzou provedenou nad jednotlivými spojeními s honeypotem, které jsou ukládány do databáze. Metriky jsou popisné vlastnosti, které specifikují dané spojení a podávají informace o parametrech útoku. Jejich analýzou jsme schopni blíže určit, jak byl útok prováděn, co bylo cílem útoku, o jakou službu projevil útočník zájem, případně možnost identifikace samotného původce útoku.

Koncepce skriptu je zaměřena na protokol TCP, proto i většina výstupních metrik je dostupná pouze pro tento protokol. Toto omezení je dáno i faktem, že některé hodnoty protokoly UDP a ICMP vůbec neobsahují nebo nemají adekvátní vypovídací hodnotu. Seznam metrik, které jsou vytvářeny a jejich dostupnost pro daný typ protokolu je vyjádřena v Tabulka 3 - Přehled výstupních metrik skriptu, jejich dostupnost podle typu protokolu.

Vybrané metriky a zjištěné odpovídající hodnoty parametrů odpovídají výstupům získaným programem tcptrace. Tento nástroj je hlavním zdrojem analyzovaných informací nad zachycenými datovými toky. Popis metrik, případně výpočet jejich hodnot vychází z manuálové stránky programu dostupné na [31].

Označení metriky	Protokol	Základní/odvozená	Stručný popis
id	TCP, UDP, ICMP	základní	Pořadové číslo
timeStart	TCP, UDP, ICMP	základní	Začátek spojení
timeStartMsecs	TCP, UDP, ICMP	základní	Začátek spojení mikrosekundy
timeEnd	TCP, UDP, ICMP	základní	Konec spojení
timeEndMsecs	TCP, UDP, ICMP	základní	Konec spojení mikrosekundy
duration	TCP, UDP, ICMP	odvozená	Délka spojení
durationMsecs	TCP, UDP, ICMP	odvozená	Délka spojení mikrosekundy
proto	TCP, UDP, ICMP	základní	Typ protokolu
srcIp	TCP, UDP, ICMP	základní	Zdrojová IP adresa
srcPort	TCP, UDP	základní	Zdrojový port
dstIp	TCP, UDP, ICMP	základní	Cílová IP adresa
dstPort	TCP, UDP	základní	Cílový port
connComplete	TCP	základní	Spojení kompletní
pckCountTotal	TCP, UDP	základní	Celkový počet paketů spojení
pckBytesUnique	TCP	základní	Počet unikátních bytů spojení
pckCountData	TCP	základní	Počet datových paketů spojení
pckBytesData	TCP, UDP	základní	Objem bytů datových paketů
pckCountRexmt	TCP	základní	Počet znovu zaslaných paketů
pckBytesRexmt	TCP	základní	Objem bytů znovu zaslaných paketů
mssReq	TCP	základní	Požadovaná maximální velikost segmentu
pckCountOutorder	TCP	základní	Počet paketů mimo pořadí
segmSizeMin	TCP	základní	Minimální velikost segmentu
segmSizeMax	TCP	základní	Maximální velikost segmentu
segmSizeAvg	TCP	odvozená	Průměrná velikost segmentu
idletimeMax	TCP	základní	Max. doba nečinnosti mezi 2 pakety
rttSamplesCount	TCP	základní	Počet zachycených RTT vzorků
rttMin	TCP	základní	Minimální hodnota RTT
rttMax	TCP	základní	Maximální hodnota RTT
rttAvg	TCP	odvozená	Průměrná hodnota RTT
rttStdev	TCP	odvozená	Standardní odchylka RTT
rtt3whs	TCP	základní	RTT z 3-Way Handshake
rttFullSamplesCount	TCP	základní	Počet RTT vzorků s plným segmentem
rttFullMin	TCP	základní	Min. RTT full segment size vzorků

rttFullMax	TCP	základní	Max.. RTT full segment size vzorků
rttFullAvg	TCP	odvozená	Průměrné RTT full segment size vzorků
rttFullStddev	TCP	odvozená	Std odchylka RTT full segment size vzorků
throughput	TCP, UDP	odvozená	Průměrná přenosová rychlost
os	TCP, UDP, ICMP	základní	Operační systém zdrojového uzlu
data	TCP, UDP	základní	Datový tok
snort	TCP, UDP	základní	Analýza nástrojem snort

Tabulka 3 - Přehled výstupních metrik skriptu, jejich dostupnost podle typu protokolu a stručný popis

7.5.1 Čas počátku spojení (*timeStart*, *timeStartMsecs*)

Čas počátku spojení udává bod na časové ose, kdy došlo k zaevidování příchozího požadavku o vytvoření datové komunikace s honeypotem. Tento parametr je vytvořen jako složenina dvou hodnot, a to *timeStart* a *timeStartMsecs*. *timeStart* nese časový údaj databázového datového typu *datetime* ve tvaru YYYY-MM-DD HH:MM:SS. Dodatkovou složkou je hodnota udávající doplněk ve formě mikrosekund. Tento způsob zápisu je ovlivněn implementací použité databáze MySQL verze 5, která sice dokáže s časovými hodnotami s přesností na mikrosekundy pracovat, nedokáže je ovšem v rámci datového typu uchovat. Tento poznatek vychází z oficiální manuálové stránky dostupné na [30].

Čas počátku spojení je jednou se stěžejních metrik sloužící k identifikaci konkrétního datového toku. Na základě této hodnoty je pak odvozována celková délka spojení, lze určit a statisticky vyhodnocovat vliv dané části dne (tzn. jestli se jedná o ráno, odpoledne, večer) a množství výskytu evidovaných spojení v daném rozmezí.

7.5.2 Čas konce spojení (*timeEnd*, *timeEndMsecs*)

Je bodem na časové ose označující, kdy došlo k ukončení vzájemné komunikace útočníka a honeypotu. Parametricky je totožné jako čas počátku spojení, kde v tomto případě se jedná o složeninu hodnot *timeEnd* a *timeEndMsecs*.

7.5.3 Délka trvání spojení (*duration*, *durationMsecs*)

Je odvozenou hodnotou, vzniklou jako rozdíl konce a počátku času spojení. Je reprezentována opět jako složenina hodnot *duration* a *durationMsecs*. Vzorec pro výpočet odpovídá $duration.durationMsecs = timeEnd.timeEndMsecs - timeStart.timeStartMsecs$

Délka spojení je určující metrikou popisující, jak dlouho útočník s honeypotem komunikovat a jak dlouho pro něj přidružená obslužná služba, ke které se připojoval, zajímavá, případně jestli zde objevil zranitelnosti a odezvy, které očekával. Délka spojení se často odvíjí od typu komunikačního protokolu. V případě ICMP se jedná nejčastěji o datové pakety na bázi ping, kdy celá komunikace

může být tvořena pouze jediným paketem. V tom případě je čas příchodu prvního paketu totožný s časem posledního paketu a tudíž je výsledná celková doba spojení nulová.

7.5.4 Komunikační protokol (*proto*)

Jak již bylo zmíněno dříve, užití honeypotu je zaměřeno na využití v síti Internet a protokolech rodiny TCP/IP. Komunikační protokol je množina pravidel, která určují syntaxi a význam jednotlivých zpráv při komunikaci. Architektura TCP/IP je rozdělena do 4 vrstev:

- aplikační vrstva (ang. application layer) - jedná se o vrstvu aplikací, což jsou procesy, které využívají přenosu dat po síti ke konkrétním službám pro uživatele. Mezi zástupce protokolů zde patří FTP, HTTP, telnet, DHCP, DNS.
- transportní (ang. transport) - zodpovědná za zapouzdřování datových bloků aplikací do datových jednotek (datagramů, TCP segmentů). Provádí doručování dat odpovídajícím procesům na koncovém počítači. Poskytuje spojované (TCP, spolehlivý) nebo nespojované (UDP, nespolehlivý) transportní služby.
- síťová (ang. network) - zajišťuje především síťovou adresaci, směrování a předávání datagramů. Mezi zástupce protokolů zde patří IP, ARP, ICMP, IGMP a další.
- vrstva síťového rozhraní (ang. network interface) - nejnižší vrstva, umožňuje přístup k fyzickému přenosovému médium, specifická pro každou síť a použitou implementaci - Ethernet, Token ring, X.25 atd.

7.5.4.1 ICMP

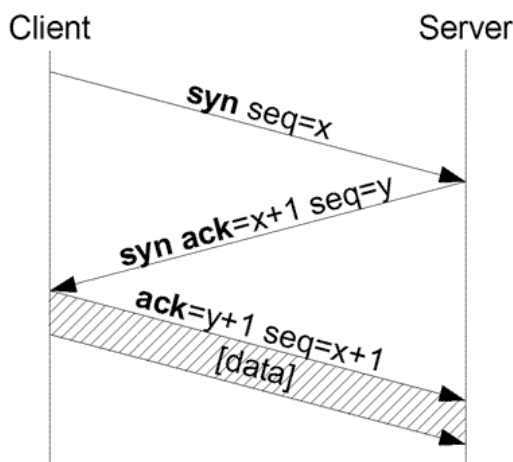
ICMP (Internet control message protocol) je jeden ze základních protokolů rodiny TCP/IP. Je nejčastěji používán operačním systémem k zasílání chybových zpráv indikující nedostupnost služeb, neschopnost nalezení cílového počítače a podobně. Bývá používán nejčastěji nástroji typu ping a traceroute pro testování dostupnosti počítače, případně sledování cesty paketů mezi komunikačními uzly.

V případě zaznamenaných příchozích ICMP paketů se většinou jedná právě o zjištění ze strany útočníka, že daná stanice je dostupná a bude schopen zde provádět další možné pokusy o komunikaci. Celková komunikace se pak skládá z jednoho paketu, přičemž celková délka spojení je nulová.

7.5.4.2 TCP

TCP (Transmission control protocol) je nejpoužívanějším protokolem v síti Internet a společně s IP je jedním z původních komponent protokolové sady. IP zpracovává požadavky na nízké úrovni, kdežto TCP operuje na vyšší úrovni a zprostředkovává spojení dvou systémů, např. webového prohlížeče a webového serveru. TCP poskytuje spolehlivé uspořádané doručování datových proudů. K využívaným mechanismům patří správa velikosti přenášeného segmentu, data flow, prevence zahlcení a další.

Pro pochopení metrik a pojmů vztahujícím se k protokolu TCP stručně popíšeme základní princip sestavení spojení, který se nazývá Three-Way Handshake nebo též TCP Handshake. Ukázkový příklad je zobrazen na obrázku Obrázek 10.



Obrázek 10 - Princip TCP handshake

Nový TCP proces otevírá socket a používá při tom lokální a vzdálené číslo portu. Po vytvoření lokálního socketu zasílá klient inicializační paket SYN, nastavuje časovač pro opětovné poslání (ang. retransmission timer) a čeká návrat odpovědi od vzdáleného hosta, nejdéle však po dobu danou časovačem. SYN paket obsahuje prázdný TCP segment, má nastavený příznak SYN v hlavičce a hodnotu sekvenčního čísla X . Když dorazí SYN paket na serverovou stranu procesu, dojde k ověření, že je možné spojení na daném socketu přijmout. Je-li vše v pořádku, je paket postoupen naslouchajícímu procesu, který verifikuje nastavení příznaku SYN a kontrolní součet. Nově vytvořený proces odesílá aktivní straně SYN-ACK paket, přičemž nastavuje časovač, tak jako v prvním případě. Generuje vlastní sekvenční číslo Y a dává na vědomí aktivní straně, že další paket očekává se sekvenčním číslem $X+1$. Když TCP proces, který vyvolal spojení obdrží ze serverové strany paket potvrzení synchronizace SYN-ACK, posílá zpět paket potvrzující přijetí. Tento paket nese příznak ACK s hodnotou $Y+1$ a sekvenčním číslem $X+1$. Pokud serverová strana obdrží ACK paket, je spojení ustaveno a může být zahájen přenos dat.

TCP vytváří virtuální okruh mezi koncovými aplikacemi zajišťující spolehlivý přenos dat. Mechanismy protokolu zajistí, že adresátovi budou doručena všechna data bez ztráty a ve správném pořadí. K dalším řídicím a opravným mechanismům slouží opětovné zaslání ztracených paketů, zahazování duplicitních paketů, kontrolní součty pro bezchybný datový přenos, řízení toku a zahlcení.

7.5.4.3 UDP

UDP (User datagram protocol) využívá jednoduchý přenosový model bez implicitního handshake pro sestavení spolehlivého spojení. UDP předpokládá, že možné vzniklé chyby nejsou pro potřeby aplikace kritické a možné ověřování a opravy jsou plýtvání časem a systémovými prostředky. UDP je

využíván zejména pro aplikace citlivé na čas, kde možné zahazování paketů je preferovanější než čekání na doručení těch zpožděných. Jedná se zejména o real-time systémy, broadcast, multicast, DNS, media streaming, IPTV, VoIP, online hry a podobně. Hlavní výhodou UDP je jednoduchost implementace a lepší poměr mezi užitečnými daty a režii.

7.5.5 Zdrojová IP adresa (*srcIp*)

IP adresa je numerickým identifikátorem, který je přiřazen komunikujícím zařízením v počítačové síti. Slouží k jednoznačné identifikaci a adresování. V současně používaném standardu IPv4 je definována jako 32 bitové číslo, nejčastěji reprezentované ve formě čitelné a srozumitelné uživatelům ve tvaru XXX.XXX.XXX.XXX, např. 192.168.1.1 Adresa IPv4 se dělí na 3 základní části - adresa sítě, adresa podsítě a adresa počítače.

Rozdělování IP na globální úrovni má na starosti IANA (Internet Assigned Numbers Authority). Eviduje, komu daný rozsah adres náleží a spravuje kořenové DNS servery. IP adresy jsou rozdělovány v rámci geografických regionů, takže zpětným trasováním cílové IP adresy lze zjistit přibližnou lokalitu útočníka. Tento údaj ovšem nemusí být zcela přesný, protože útočníci využívají nástrojů a technik, které jim umožňují vlastní IP maskovat a podvrhnout cizí.

V rámci databáze je IP adresa ukládána ve formátu unsigned int konvertovaná funkcí INET_ATON, pro zpětné sestavení nutné použít funkci INET_NTOA. Tyto funkce jsou právě za účelem snadného manipulace s daty tohoto typu v databázi implementovány, tzn. jednoduché srovnávání, řazení a podobně.

7.5.6 Zdrojový port (*srcPort*)

Protokoly transportní vrstvy, jako je TCP, UDP specifikují zdrojový a cílový port v hlavičce paketu. Číslo portu je reprezentován jako 16 bitový integer, což odpovídá rozsahu 0-65535. Proces vyžadující komunikaci po síti je svázán s konkrétním portem (tzv. binding), což znamená, že bude naslouchat příchozím požadavkům, kde jejich zdrojový port odpovídá právě tomuto portu. Aplikace využívající běžné a často využívané služby tradičně naslouchají na portech specifických čísel, která jsou stanovena obecnými úmluvami.

7.5.7 Cílová IP adresa (*dstIp*)

Cílová IP adresa odpovídá v našem případě adrese honeypotu. Je určujícím prvkem, který identifikuje, na který z vystavených honeypotů je spojení vedeno. Způsob zápisu a určující prvky jsou totožné jako u zdrojové adresy.

7.5.8 Cílový port (*dstPort*)

Cílový port je určující metrikou, která udává, ke které službě se příchozí spojení vztahuje. Tak jak bylo uvedeno výše, dle dohodnutých standardů mají nejčastěji používané služby ve výchozím nastavení určeno číslo portu, na kterém komunikují, např. protokol HTTP a webové služby na portu 80, FTP port 21 a tak dále.

7.5.9 Stav spojení (*connComplete*)

Stav spojení je metrikou vztahující se pouze k datovým tokům na protokolu TCP. Za kompletní spojení je považováno takové, kde byl zachycen a evidován jak inicializační SYN segment, tak ukončující FIN segment. Možnými hodnotami jsou yes, no, reset. "Yes" odpovídá informaci, že spojení bylo evidováno jak kompletní včetně SYN a FIN segmentu. Hodnota "no" pak stavu, kdy pakety sestavovací/ukončovací fáze nebyly všechny zaznamenány. Reset je ekvivalentem TCP paketu s příznakem RST, který je zaslán zdrojové straně v případě, že na příslušném portu cílové strany není žádná běžící služba, port je uzavřený a značí, že má být spojení ukončeno.

7.5.10 Celkový počet paketů (*pckCountTotal*)

Celkový počet paketů nese číselné vyjádření odpovídající součtu všech paketů, které byly v rámci daného spojení zaznamenány. Co se týče metrik týkající se množství zachycených paketů, vždy se jedná pouze o vyjádření množství paketů zaslaných směrem od útočníka k honeypotu. Statistiky opačného směru pro nás nejsou zajímavé, protože odpovědi, které honeypot může poskytovat jsou známé a neobsahují žádnou novou informaci, případně využití pro statistické vyhodnocení. Tato metrika je aktivní pouze pro spojení na protokolu TCP, UDP.

Do celkového množství paketů v případě TCP jsou zaznamenávány i pakety pro sestavení spojení (SYN, ACK). Z hodnoty celkového počtu se pak odvíjí i odvozené metriky popisující počet přenesených bytů, počet pouze datových paketů a podobně. Tyto jsou popsány níže. Vždy ale platí, že hodnota celkového počtu je větší, maximálně rovna součtu uvedených odvozených metrik vztahující se k množství přenesených paketů.

Z hlediska analýzy a vyhodnocení platí, že čím více paketů je přeneseno a zachyceno, tím zajímavější komunikace proběhla a může být podrobena hlubšímu zkoumání. V opačném případě, kdy v případě TCP se jedná o součet přibližně 5 a méně paketů, můžeme říci, že byly zachycena pouze sestavovací sekvence a samotný užitečný datový obsah je prakticky nulový.

7.5.11 Množství unikátních bytů (*pckBytesUnique*)

Hodnota unikátních bytů udává objem zaslaných dat v bytech, což odpovídá datovému objemu bez znovu zaslaných bytů a bytů použitých k window probingu, což je technika používaná při žádostech zaslaných serveru o aktuální velikost přenosového okna.

7.5.12 Počet datových paketů (*pckCountData*)

Počet paketů nesoucí neprázdnou hodnotu v hlavičce protokolu v poli data (payload).

7.5.12.1 Objem bytů datových paketů (*pckBytesData*)

Vyjadřuje množstevní součet dat v těle paketu ze všech zaznamenaných datových paketů spojení.

7.5.13 Počet opakovaně zaslaných paketů (*pckCountRexmt*)

Opakovaně zaslané neboli retransmissioned pakety vznikají nejčastěji v důsledku poškození nebo ztráty původního paketu. V důsledku nekvalitního spojení, velkého zahlcení komunikační linky a dalších elementů může docházet k tomuto jevu. Opětovné zaslání je automatický opravný mechanismus poskytující spolehlivá spojení, jako např. TCP

7.5.13.1 Objem bytů opakovaně zaslaných paketů (*pckBytesRexmt*)

Vyjadřuje součet bytů datové části paketu z opakovaně zaslaných paketů.

7.5.14 Počet out-of-order paketů (*pckCountOutorder*)

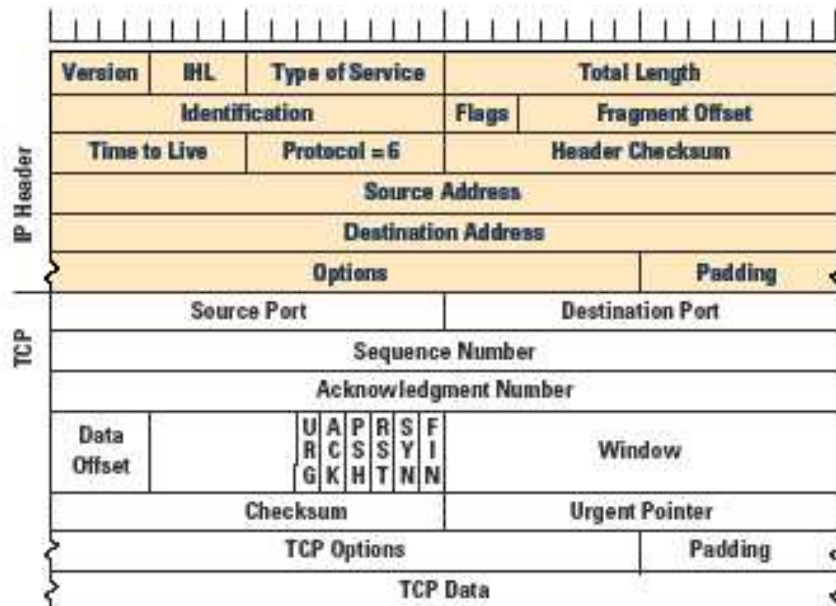
Jedná se o pakety, které z nějakého důvodu dorazily do koncového uzlu v jiném pořadí, než byly vyslány. Důvodem proč se tak děje může být jiná trasa, kudy došlo k transferu, případně vliv nekvalitního spojení.

7.5.15 Maximální velikost segmentu (*mssReq*)

Maximální velikost paketu (Maximum segment size - MSS) značí velikost prostoru, který je v rámci každého paketu určen pro samotnou datovou část. Je to největší možné množství dat v rámci paketu uváděné v bytech, které počítač dokáže zpracovat bez toho, že by ho musel dělit na menší části a fragmentovat. V rámci rozboru TCP paketu je to pole data. Každý paket který je přijímán nebo odesílán je rozdělen na dvě základní části - hlavičku a data. Hlavička nese údaje o zdrojovém a cílovém portu, sekvenčním čísle a další, tvoří zapouzdření samotných dat. Datová část nese samotný obsah paketu, což je například text emailu, obrázek ikony a podobně. Pro každý standardní TCP/IP paket existují dvě hlavičky - TCP hlavička a IP hlavička, každá o velikosti 20 bytů. Maximální velikost Ethernetového paketu je 1500 bytů, přičemž maximální velikost datového pole je

1460 bytů. Z důvodu efektivnosti přenosu na síti je nejvýhodnější posílat co nejvíce dat v rámci jednoho paketu. Protokol TCP poskytuje mechanismy pro obě strany komunikace, aby vyžádaly MSS na začátku ustavení spojení.

V rámci uvedené metriky *mssReq* se jedná o maximální velikost segmentu obsaženou v úvodním SYN paketu při sestavování spojení v TCP options.



Obrázek 11 - Schéma TCP/IP paketu, zdroj [<http://www.potaroo.net/>]

Name	Kind	Length
End of Option List	0	N/A
No-Operation	1	N/A
Max Segment Size	2	4
Window Scale	3	3
Selective ACK	4	2
Timestamp	8	10

Obrázek 12 - Pole TCP Options, zdroj [<http://www.chrisbrenton.org/>]

7.5.15.1 Minimální velikost segmentu v průběhu spojení (*segmSizeMin*)

Metrika udávající minimální velikost segmentu zaznamenanou v průběhu života spojení.

7.5.15.2 Maximální velikost segmentu v průběhu spojení (*segmSizeMax*)

Metrika udávající maximální velikost segmentu zaznamenanou v průběhu života spojení.

7.5.15.3 Průměrná velikost segmentu v průběhu spojení (*segmSizeAvg*)

Metrika udávající průměrnou velikost segmentu zaznamenanou v průběhu života spojení, kalkulovaná

$$\text{jako } segmSizeAvg = \frac{pckBytesData}{pckCountData}$$

7.5.16 Maximální doba nečinnosti mezi pakety (*idleTimeMax*)

Metrika udává maximální dobu nečinnosti (ang. idle time), která byla zaznamenána mezi dvěma po sobě následujícími pakety v rámci spojení ve stejném směru komunikace.

7.5.17 Počet zachycených RTT vzorků (*rttSamplesCount*)

Round trip time (RTT) mezi dvěma uzly sítě je čas, který uběhne mezi vysláním paketu od odesílatele, odpovědí vzdáleného uzlu a přijetím této odpovědi. RTT je měřeno za účelem monitorování výkonnosti sítě a její diagnostiky. RTT je vyjadřován v milisekundách a jako nejznámější určení této hodnoty slouží program ping.

V rámci udávané metriky se udává celkový počet nalezených RTT vzorků. Ty jsou vyhodnoceny na základě programu tcptrace, přičemž za validní RTT vzorek je považován pouze takový, kde ACK paket je přijat od zdrojového uzlu pro předchozí vyslaný paket, takže hodnota ACK je o 1 vyšší než sekvenční číslo posledního paketu.

Výsledná hodnota RTT se odvíjí od různých faktorů, ke kterým patří zejména přenosová rychlost spojení (konektivita), typ přenosového média, fyzická vzdálenost komunikujícího zdroje a cíle, počet uzlů na trase spojení, zatížení linky a další.

7.5.17.1 Minimální hodnota RTT (*rttMin*)

Odvozená metrika udávající minimální evidovanou hodnotu RTT v průběhu spojení.

7.5.17.2 Maximální hodnota RTT (*rttMax*)

Odvozená metrika udávající maximální evidovanou hodnotu RTT v průběhu spojení.

7.5.17.3 Průměrná hodnota RTT (*rttAvg*)

Odvozená metrika udávající průměrnou hodnotu RTT v průběhu spojení, počítaná jako podíl celkové hodnoty RTT zachycených vzorků vůči počtu zachycených vzorků.

7.5.17.4 Standardní odchylka RTT (*rttStdev*)

Standardní odchylka (ang. standard deviation) z počtu zachycených RTT vzorků (*rttSamplesCount*).

7.5.18 Hodnota RTT z TCP 3-Way Hand-Shake (*rtt3whs*)

Hodnota RTT vypočtená z času přenosu paketů v úvodní sestavovací sekvenci spojení (3-Way Handshake). Předpokladem je, že SYN pakety byly korektně zaznamenány.

7.5.19 Počet full size segment RTT vzorků (*rttFullSamplesCount*)

Celkový počet vzorků, které byly vypočteny pro hodnotu RTT a přenášely datový segment maximální velikosti (full size segment). Vzorky plné velikosti jsou definovány jako segmenty největší velikosti nalezené v průběhu spojení.

7.5.19.1 Minimální hodnota RTT full size segment vzorku (*rttFullMin*)

Odvozená metrika udávající minimální hodnotu RTT v rámci všech vzorků s full size segmentem.

7.5.19.2 Maximální hodnota RTT full size segment vzorku (*rttFullMax*)

Odvozená metrika udávající maximální hodnotu RTT v rámci všech vzorků s full size segmentem.

7.5.19.3 Průměrná hodnota RTT full size segment vzorku (*rttFullAvg*)

Odvozená metrika udávající průměrnou hodnotu RTT v rámci všech vzorků s full size segmentem.

7.5.19.4 Standardní odchylka RTT full size segment vzorku (*rttFullStdev*)

Odvozená metrika udávající standardní odchylku RTT v rámci všech vzorků s full size segmentem.

7.5.20 Průměrná přenosová rychlost (*throughput*)

Přenosová rychlost udává objem dat, který lze přenést za časovou jednotku. Nejčastěji se udává v jednotkách kB/s nebo Mbit/s, v našem případě kB/s. Je kalkulovaná podílem unikátních přenesených bytů k délce trvání spojení, tedy $throughput = \frac{pckBytesUnique}{duration.durationMsecs}$

7.5.21 Procentuální vyjádření ztrátovosti paketů (*packetloss*)

Ztráta paketu nastává v případě, že jeden nebo více paketů přenášených komunikační linkou zdárně nedorazí na místo určení, tzn. na síťové rozhraní cílového počítače. Ztráta paketu může nastat řadou vlivů, jako je výpadek sítě, poškození paketu v průběhu přenosu, porucha hardware nebo chyba v routování. Ztrátovost paketu je spjata s hodnotou znovu zaslaných paketů, které jsou pak opakovaně zaslány v rámci opravných mechanismů síťového protokolu. Ztrátovost zjednodušeně udává kvalitu spojení komunikujících uzlů. Ideální hodnotou je nulová ztrátovost. Výsledná hodnota je počítána

jako $packetloss = \frac{pckCountRexmt}{pckCountTotal}$

7.5.22 Odhadovaná verze OS zdrojové stanice (*os*)

Odhadovaná verze operačního systému zdrojového počítače je v našem případě spíše informativní, jde zejména o stanovení přibližných statistik uskutečněných spojení. S ohledem, že současná

komunita útočníků využívá řadu nástrojů a technik, jak podvrhnout tuto hodnotu, nelze brát výsledky jak příliš určující. Odhad OS vychází z hodnoty dostupné v logu Honeyd.

7.5.23 Hrubý datový tok spojení (*data*)

V tomto poli je uložen kompletní hrubý datový tok příslušného spojení ve stejném tvaru a formě, jakou produkuje nástroj `tcpdump -w`. Jedná se o veškeré pakety spojení, filtrované pouze pro příchozí směr dané komunikace, tzn. data, která přišla od odesílatele na honeypot. Z těchto dat lze zrekonstruovat a získat prakticky všechny z výše uvedených metrik. Data jsou ukládána i pro potřebu případného rozšíření metrik a požadovaných údajů, tak i další možné budoucí analýzy.

7.5.24 Výstupní detekce nástroje snort (*snort*)

Pole obsahující textové údaje o možném typu bezpečnostního incidentu tak, jak byl vyhodnocen na základě pravidel a znalostní databáze programu snort. Jedná se o výstupní popis incidentu, který odpovídá známým typům útoků a podezřelého chování, které jsou konfigurována v rámci snortu.

7.6 Struktura uložení metrik v databázi

Jednotlivé metriky a účel použití byly popsány v kapitole 7.5. Zde se podíváme na jednotlivé metriky z pohledu datového typu a uložení v databázi. Obrázek 13 - Datová struktura uložení metrik v databázi zobrazuje označení jednotlivých parametrů, tak jak jsou uloženy v databázi a jejich odpovídající datový typ.

Stream		
+id	integer	Nullable = false
timeStart	timestamp	Nullable = false
timeStartMsecs	integer	Nullable = true
timeEnd	timestamp	Nullable = true
timeEndMsecs	integer	Nullable = true
duration	time	Nullable = false
durationMsecs	integer	Nullable = false
proto	varchar	Nullable = false
srcIp	integer	Nullable = false
srcPort	integer	Nullable = true
dstIp	integer	Nullable = false
dstPort	integer	Nullable = true
connComplete	varchar	Nullable = true
pckCountTotal	integer	Nullable = true
pckBytesUnique	integer	Nullable = true
pckBytesData	integer	Nullable = true
pckCountRexmt	integer	Nullable = true
pckBytesRexmt	integer	Nullable = true
pckCountOutorder	integer	Nullable = true
mssReq	integer	Nullable = true
segmSizeMin	integer	Nullable = true
segmSizeMax	integer	Nullable = true
segmSizeAvg	integer	Nullable = true
idletimeMax	float	Nullable = true
rttSamplesCount	integer	Nullable = true
rttMin	float	Nullable = true
rttMax	float	Nullable = true
rttAvg	float	Nullable = true
rttStdev	float	Nullable = true
rtt3whs	float	Nullable = true
rttFullSamplesCount	integer	Nullable = true
rttFullMin	float	Nullable = true
rttFullMax	float	Nullable = true
rttFullAvg	float	Nullable = true
rttFullStdev	float	Nullable = true
throughput	float	Nullable = true
packetloss	float	Nullable = true
os	varchar	Nullable = true
data	blob	Nullable = true
snort	varchar	Nullable = true

Obrázek 13 - Datová struktura uložení metrik v databázi

7.7 Zhodnocení implementace

Vytvořený nástroj automatizovaného zpracování údajů o spojeních a potenciálních útocích na honeypot potvrdil schopnost vyhodnocovat a vytvářet výstupní metriky vhodné pro další analýzu. Bylo vytvořeno 40 metrik popisující charakter spojení s orientací na protokol TCP. Použité nástroje byly schopny zajistit odpovídající předzpracování a získání mezivýsledků.

V rámci případného budoucího rozšiřování funkcionality je možné přidat další metriky dle potřeby, které lze například získat z dosud nevyužitých výstupů nástroje tcptrace. Implementace je provedena jednoduchou formou a umožňuje flexibilní úpravy podle potřeby. Pro komplexnější budoucí využití s požadavky na vyšší efektivitu by bylo možné vytvořit další pomocné nástroje, které by nahradily funkcionalitu programů tcpdump a tshark, kde by mohlo být efektivněji s ohledem na vyžadovaný čas prováděno parsování a filtrování datových toků. To ovšem přesahuje možnosti této práce. Po ostatních stránkách byly splněny předpokládané vstupní požadavky a funkcionalita.

8 Sběr a vyhodnocení získaných dat

V této kapitole provedeme zhodnocení dosažených výsledků získaných v průběhu sběru dat z honeypotů a jejich zpracování vytvořeným automatizovaným nástrojem. Ověříme, jaká je kvalita a podrobnost dat o útocích získaných honeypotem s nízkou mírou interakce a jak lákavým cílem dokáže honeypot pro útočníka být.

8.1 Konfigurace a požadavky systému

K nasazení honeypotu Honeyd došlo ve spolupráci se společností AEC, která patří k nejvýznamnější domácímu poskytovateli software a služeb pro bezpečnost dat a antivirovou ochranu. Díky prostředkům školy a zprostředkování spolupráce se společností AEC vedoucím práce Ing. Michalem Drozdem, byl poskytnut server a dodatečné potřebné prostředky nutné pro nasazení v prostředí firemní sítě AEC. V průběhu nasazení a konfigurace serveru byla využita asistence pana Tomáše Vobraby, který svou výraznou pomocí přispěl ke zprovoznění honeypotu, za což mu tímto velice děkuji.

Na základě požadavků na obslužný software automatizovaného skriptu popsaného v kapitole 7, byl server konfigurován následovně:

- Operační systém CentOS 5.3 32bit
- Honeyd 1.5b
- Tcpdump 4.0.0
- TShark 1.2.7
- Tcpslice 1.2a3
- Tcptrace 6.6.7
- Perl 5.8.8
- Snort 2.8.5.3
- MySQL 5.0.77 + phpMyAdmin 2.11.10

Honeypoty byly nasazeny na servery DELL s procesorem Xeon E3110 @ 3Ghz, 4GB RAM, 500GB HDD. Pro provoz honeypotů byly vyhrazeny 4 veřejné IP adresy v rozsahu 95.80.208.133 - 95.80.208.136. Server s Honeyd běžel na adrese 95.80.208.130, kde byl nainstalován veškerý software, nastavena konfigurace honeypotů a zajištěno odpovídající routování. K tomuto účelu byl využit program farpd (Fake ARP user space daemon). Jedná se o ARP¹⁵ démon službu, která

¹⁵ ARP (Adress resolution protocol) je síťový protokol k určení hardwarové adresy cíle, přičemž je známa pouze jeho IP adresa. Jedná se o využití v lokální síti tak pro routování skrze gateway, kdy je potřeba určit další uzel komunikace.

odpovídá na všechny ARP dotazy nastavené pro daný rozsah IP adres a směřuje požadavky na jedno ze síťových rozhraní serveru, poté co je ověřeno, že tato IP adresa není využívána jiným uzlem v síti. Tímto způsobem je umožněno, aby jediná stanice tvrdila vlastnictví volného rozsahu IP adres a zpracovávala požadavky směřované na ně. V našem případě to znamená, že fyzický server adresy 95.80.208.130 bude zpracovávat veškeré příchozí požadavky určené pro volný rozsah veřejných adres 95.80.208.133-136, kde budou emulovány honeypoty.

V rámci nasazení byly vystaveny 4 honeypoty na bázi emulace systému Microsoft Windows XP Professional SP1 s následujícími službami a příslušnými porty:

- 21 - FTP server
- 23 - Telnet
- 25 - Exchange SMTP server
- 80 - IIS Web server
- 110 - Exchange POP3 server
- 139 otevřený port
- 143 Exchange IMAP server
- 389 LDAP
- 445 Skript simulující infekci Confickerem pro nmap
- 5901 VNC
- Výchozí akce pro ostatní TCP/UDP porty RESET

Přehled spuštěných služeb s ukázkou jejich parametrů znázorňuje Text 14 - Spouštění dávkky vyžadované pro funkčnost sběru a automatizovaného vyhodnocení. Nejprve je nutné zajistit požadované routování, nástroj tcpdump zajistí zachytávání a ukládání veškeré síťové komunikace do souborů, následně je spuštěn samotný démon honeyd a automatizovaný skript s určením cesty log souboru. Obsah konfiguračního souboru honeyd.config je uveden jako příloha 2 a je součástí i na přiloženém CD.

```
/usr/sbin/farpd 95.80.208.133
/usr/sbin/farpd 95.80.208.134
/usr/sbin/farpd 95.80.208.135
/usr/sbin/farpd 95.80.208.136

/usr/local/sbin/tcpdump -i eth0 -n -vv -XX -s 0 -tttt -w
/home/xgalet02/honeyd/pcap/%Y-%m-%d_%H-%M-%S.pcap -G 60

/usr/bin/honeyd -i eth0 -df /mnt/data/home/xgalet02/honeyd/honeyd.config -p
/mnt/data/honeyd/nmap.prints -l /mnt/data/honeyd/logs/log.txt -x
/mnt/data/honeyd/xprobe2.conf -0 /mnt/data/honeyd/pf.os -s
/mnt/data/honeyd/logs/servicelog.txt -u 0 95.80.208.133-95.80.208.136

perl script.pl /home/xgalet02/honeyd/logs/log.txt
```

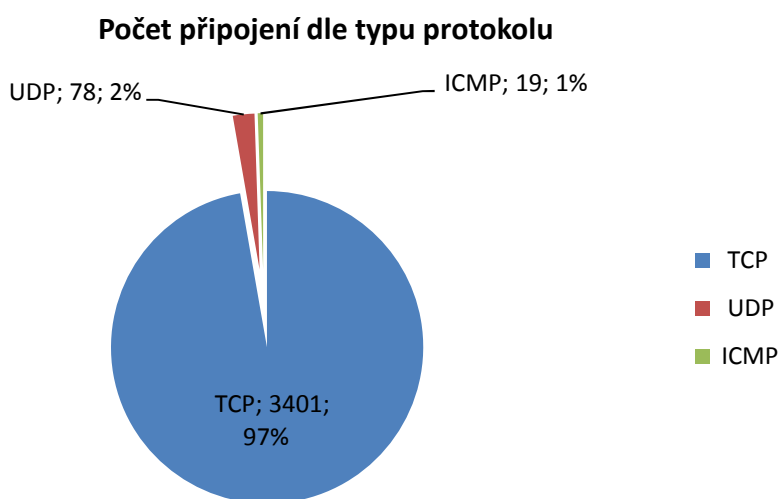
Text 14 - Spouštění dávkky vyžadované pro funkčnost sběru a automatizovaného vyhodnocení

8.2 Statistické výsledky sběru

Analyzované výsledky zachycují data zachycená v rozmezí 27.4.2010 - 2.5.2010. Za sledované období došlo k celkem 3498 evidovaným spojením. Výsledný počet spojení, který byl zpracován vytvořeným automatizovaným skriptem, se mírně liší od údajů dostupných ze sumarizačního nástroje honeydsum. Z důvodu většího množství dostupných metrik a přesnějších údajích o realizovaných spojeních budeme nadále používat data získaná vytvořeným skriptem. Pro možné srovnání je sumarizační výstup z honeydsum je uveden jako součást příloh v elektronické podobě.

8.2.1 Počty spojení dle typu protokolu

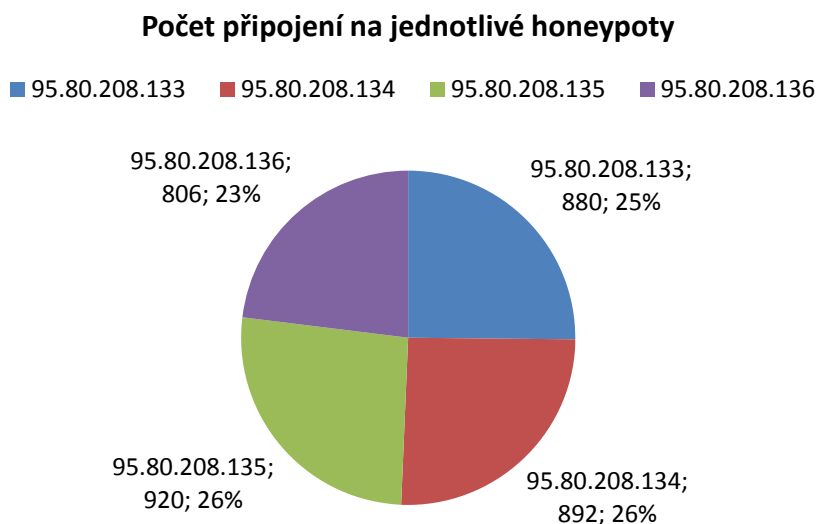
Honeypoty byly konfigurovány tak, aby mohly simulovat chování OS Windows XP s přístupnými serverovými a klientskými službami, kde bylo předpokládáno, že většina komunikace bude probíhat na protokolu TCP. Tento předpoklad byl prakticky ověřen, čemuž odpovídají výsledky, které zobrazuje Obrázek 14.



Obrázek 14 - Počet uskutečněných spojení s honeypoty podle typu protokolu

8.2.2 Počet spojení na jednotlivé honeypoty

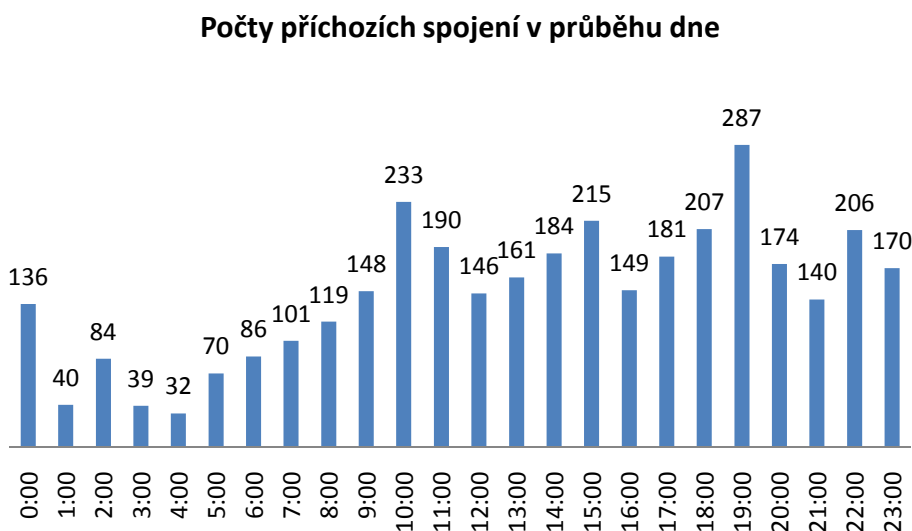
Co se týče zátěže a počtu spojení s jednotlivými honeypoty, jsou výsledné hodnoty vyrovnané, každý z honeypotů uskutečnil přibližně stejné množství spojení.



Obrázek 15 - Počet připojení na jednotlivé honeypoty

8.2.3 Rozložení spojení v průběhu dne

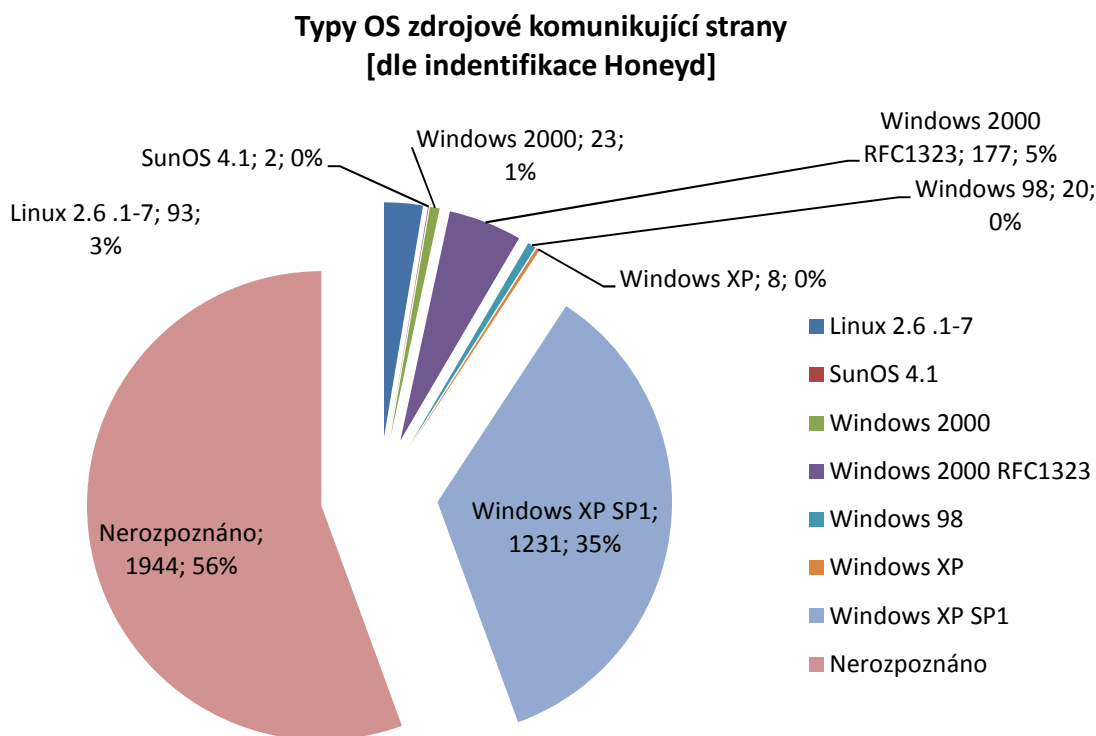
K dalším zajímavým statistikám zjištěných z analýzy výsledků patří i rozložení výskytu spojení v čase, to znamená, zdali se počet připojení odvíjí toho, jestli se jedná o ranní, odpolední, či večerní hodinu. Tento výstup nemusí být zcela určující, jelikož je nutné kalkulovat s možností přístupů z lokalit v jiných časových pásmech.



Obrázek 16 - Počty příchozích spojení na honeypoty v průběhu dne

8.2.4 Operační systémy útočníků

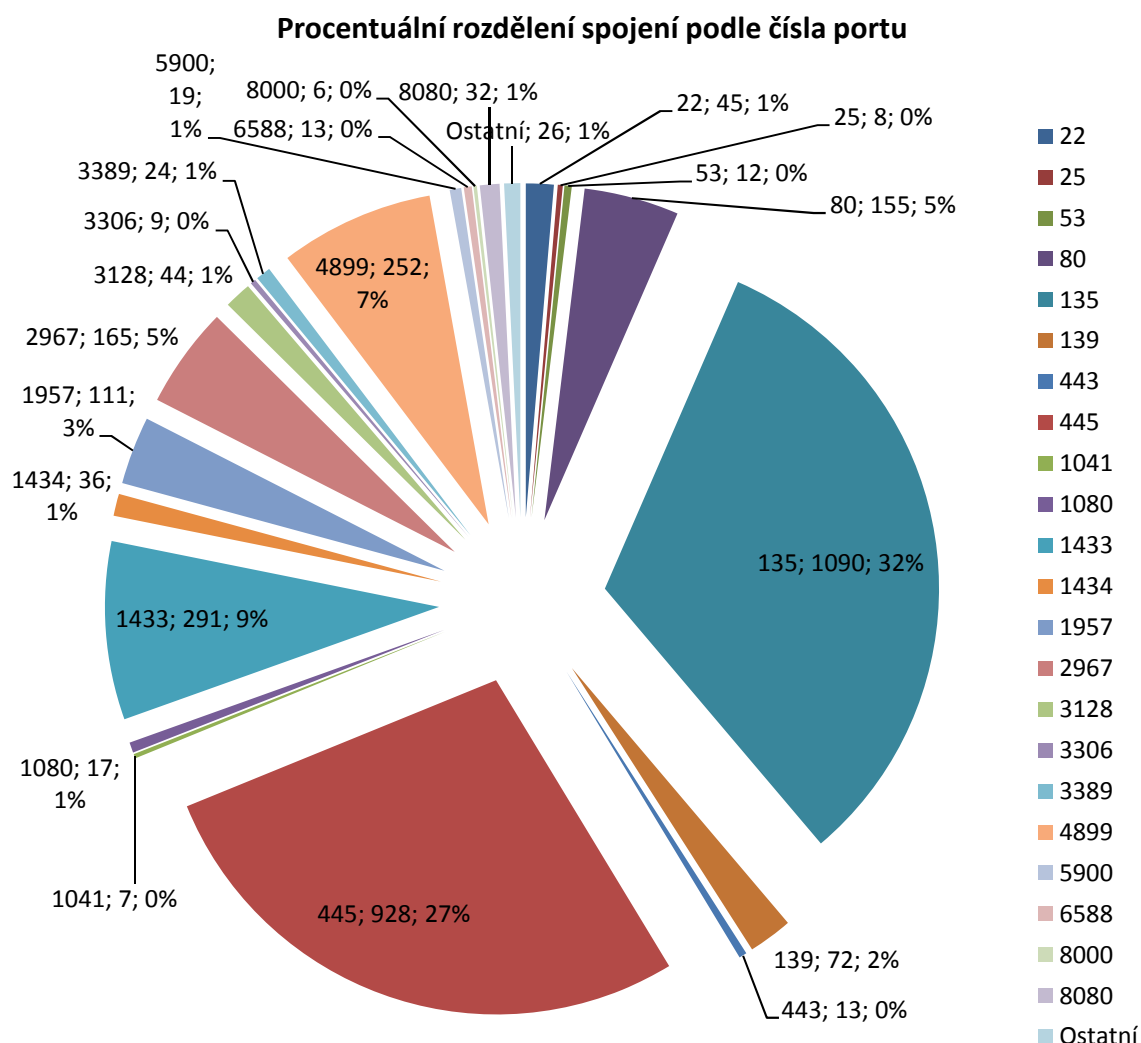
Jako další informativní výstup analýzy získaných dat je přehled operačních systémů, které útočníci využívají. Tak jak bylo zmíněno dříve, tento údaj bývá záměrně zkreslován a znalostní databáze Honeyd nepostihuje novější verze dostupných OS. S ohledem na to, že Windows XP je stále využíván na přibližně 60 procentech počítačů a útočníci se často rádi takto maskují, tak i získané výsledky tomuto stavu odpovídají.



Obrázek 17 - Rozdělení OS zdrojových stran spojení

8.2.5 Cílové porty zájmu útočníků

Mezi nejzajímavější statistické údaje patří analýza uskutečněných spojení na základě cílových portů. Jak bylo uvedeno v kapitole 8.1, honeypoty byly konfigurovány s omezeným množstvím služeb využívaných na systémech Microsoft Windows. Výsledný graf a procentuální vyjádření portů, které tvořily střed zájmu útočníků, popisuje Obrázek 18.



Obrázek 18 - Graf procentuálního rozdělení spojení dle čísla portu

Z grafu je zřetelné, že zájmu se těšily z velké míry i porty, které nebyly konfigurovány s žádnou aktivní službou a obsluhou. Největší zastoupení tvoří port 135, který je běžně využíván jako RPC (remote procedure call) end-point mapper. RPC je používán řadou protokolů vyšších tříd jako jejich transportní vrstva, jako je DCOM¹⁶. Port 135 je považován za jeden z nejvíce napadaných portů na Internetu. Port 135 využívá ke svému šíření kupříkladu červ MS Blaster. Druhé místo ve statistikách zabírá port 445, který patří SMB/CIFS a slouží k sdílenému přístupu k souborům, tiskárnám a dalším systémovým zdrojům počítače. Poskytuje také autentizovaný mechanismus pro meziprocovou komunikaci. Zranitelnosti MS08-067, která byla popsána v kapitole 2.3.1 využívá právě Conficker, který uskutečňuje spojení právě na portu 445.

Port 1433 bývá využíván Microsoft SQL Serverem pro připojování k databázi. Pro MS SQL Server existuje řada bezpečnostních děr, které bývají útočníky využívány pro provádění kódu

¹⁶ DCOM (Distributed Component Object Model) je proprietární technologie Microsoftu, zabudovaná do operačních systémů rodiny Windows, která slouží k povolování různým softwarovým komponentám komunikovat napříč sítí.

a získání přístupu k systému. Port 4899 je v největším množství případů využíván pro Vzdálenou správu (ang. Remote administration), která umožňuje vzdálené přihlášení k systému a jeho správu a interakci s dostupnými prostředky. Z ostatních zastoupených stojí za zmínku port 2967, který je známý bezpečnostní chybou, která umožňuje uživatelům vzdálený přístup do systému skrze Symantec Antivirus. Cílem pokusů o útok byl i webový server, běžící na portu 80.

8.2.6 Ostatní statistiky

Analýzou zachycených dat byly získány i níže uvedené metriky, které znázorňují, jak zajímavým cílem byly nasazené honeypoty útočníkovi:

- Průměrná hodnota¹⁷ délky TCP spojení byla 3.4s, modus¹⁸ 0s, medián¹⁹ 0.4s
- Průměrná hodnota počtu paketů TCP spojení byly 3 pakety, modus 3, medián 3
- Průměrný počet bytů TCP spojení bylo 93 bytů, modus 0, medián 0
- Bylo evidováno 396 spojení skládající se pouze z 1 paketu, 875 ze 2 paketů, 1203 ze 3, 497 ze 4, 200 z 5 paketů; tzn. 90 procent všech spojení neslo zanedbatelný datový obsah, tudíž minimální možnost podrobnější analýzy obsahu datového přenosu a vyhodnocení technik a užívaných nástrojů
- Analýza nástrojem Snort nezachytila jediný výskyt útoku dle znalostní báze (do tohoto počtu nelze kalkulovat útoky typu skenování portů)

V případě sesumarování výsledných statistik nedošlo k zachycení výrazně zajímavějších spojení, které by šlo detailněji analyzovat. Část uskutečněných spojení byla vedena s účelem zjištění otevřených portů a dostupných služeb na základě skenování definovaných portů. V ostatních případech byla s velkou pravděpodobností spojení vedena s využitím automatizovaných nástrojů, které mají vést k inicializaci bezpečnostního incidentu, ovšem rozsah a kvalita úrovně služeb nabízených Honeyd nedokázala poskytnout dostatečně zajímavý cíl a útočníci svou snahu o přístup k systému dále nerozvíjeli.

8.3 Vyhodnocení získaných výsledků

S ohledem na možnosti Honeyd a služeb, které byly vystaveny jako návnada útočníkům, v celkovém pohledu nebylo možné provést podrobnější analýzy nejčastěji napadaných portů a příslušných služeb.

¹⁷ Průměrná hodnota : $\bar{x} = \frac{x_1+x_2+x_3+x_4+\dots+x_n}{n} = \frac{\sum_{i=1}^n x_i}{n}$

¹⁸ Modus je nejčastěji se vyskytující číslo ve skupině čísel

¹⁹ Medián, $\tilde{x}$, souboru výsledků je hodnota ležící uprostřed intervalu hodnot výsledků seřazených podle velikosti. Pro lichý počet výsledků se medián rovná prostřednímu z výsledků a pro sudý počet se rovná aritmetickému průměru dvou prostředních výsledků.

Toto konstatování vychází z předpokladu, že komunita útočníků využívá řadu automatizovaných nástrojů a technik, které mají dopomoci narušení bezpečnosti, ovšem nabízené cíle v podobě nízko interaktivních honeypotů nebyly schopny poskytnout dostatečnou věrohodnost skutečného systému, na základě čehož by bylo možné provádět hlubší analýzu konkrétních útoků, rozbor odezvy systému a rozpoznání použitých nástrojů útočníků. Dosažené výsledky odpovídají použité technologii honeypotu s nízkou mírou interakce.

Vytvořený automatizovaný nástroj analýzy počítačových útoků potvrdil své možné využití i pro další použití v projektech orientovaných na počítačovou bezpečnost. I přes to, že získané statistiky nepřinesly nové poznatky o nových vzorcích malware a útocích, tak potvrdily předpoklady současných trendů a ukázaly, že množství útoků na nechráněné systémy není pouze banalitou, ale probíhá ve velké míře.

9 Závěr

V úvodních části diplomové práce jsme se seznámili s hrozbami současných síťových útoků a šíření škodlivého kódu v podobě malware. Jak dokládají uvedená čísla, nelze tuto problematiku v žádném případě podceňovat a je potřeba jí věnovat zvýšenou pozornost. Je zapotřebí neustálého zdokonalování bezpečnostních mechanismů a schopnosti včas detekovat a eliminovat nové hrozby.

Kapitoly 3, 4 a 5 popisují technologii honeypotů, které slouží zejména k odhalování a analýze nových vzorků škodlivého kódu, tak jako i sledování současných trendů v oblasti počítačové kriminality. Uvedli jsme, v čem spočívá síla těchto nástrojů, jaké inovace přináší a jaká úskalí mohou obsahovat. Byl proveden přehled dostupných řešení s ohledem na možnou míru interakce, která je útočníkům nabízena. Hlavní zájem je věnován honeypotu s nízkou mírou interakce Honeyd. Honeyd nabízí funkcionalitu emulace velkého rozsahu systémů, kde jeho nejsilnější stránkou je flexibilita v oblasti konfigurace nabízených služeb. Honeyd vytvořil stěžejní základ systému sběru dat a jejich vyhodnocení vytvořeným automatizovaným skriptem. Bylo zjištěno, že Honeyd není ideálním řešením v možnostech využití detekce nového typu malware, jelikož jeho vývoj a podpora ze strany vývojářské komunity v posledních letech upadl.

V šesté kapitole tvořící jednu z praktických částí práce jsme se věnovali tvorbě servisního skriptu simulující chování infikované stanice červem Conficker. I přes nalezená úskalí byl implementován skript, který dokáže poskytovat očekávané odpovědi pro nástroj nmap a jeho interní skript detekce zranitelnosti MS08-067 na systémech Microsoft Windows. Bylo zjištěno, že možnost vytváření vlastních rozšiřujících skriptů pro Honeyd simulujících komplexnější subsystémy a služby je značně omezená a prakticky nerealizovatelná.

Kapitola sedmá je věnována popisu implementace automatizovaného nástroje analýzy a vyhodnocování údajů z útoků prováděných na honeypot. Za využití podpůrných nástrojů, které tvoří zejména tcpdump a tcptrace bylo možné provést hlubší analýzu datových toků a produkovat výsledky popisující spojení ve formě odpovídajících metrik. Vytvořený nástroj potvrdil svoji schopnost analyzovat uskutečněná datová spojení a generovat odpovídající metriky. Vytvořený nástroj lze použít i v rámci dalších projektů, týkající se počítačové bezpečnosti a analýzy datových spojení.

Vyhodnocení zaznamenaných dat o útocích na honeypot, který byl jako součást praktické části nasazen v síti Internet je popsáno v kapitole 8. Za sledované období trvající přibližně týden bylo zaznamenáno 3500 unikátních pokusů o spojení se 4 konfigurovanými honeypoty na bázi Windows XP. Vytvořený automatizovaný nástroj zajistil odpovídající zpracování dat a generování metrik spojení, ovšem s ohledem na použitý honeypot s nízkou mírou interakce a nedostatečnou škálu nabízených služeb a jejich věrohodnost, nedošlo k zachycení žádného nového malwarového vzorku, případně zajímavějšího pokusu o útok.

V rámci možného budúciho rozvoje projektu by bolo vhodné použitie schopnejšieho honeypotu než je Honeyd. Ideálnym riešením by bol honeypot s vysokou mierou interakcie, kde by bolo zapotrebie rozšíriť i sledované parametre chovania útočníkov a odpovídající metriky automatizovaného nástroje analýzy. Počítačová bezpečnosť je jedným z hlavných odvetví IT, kde je a do budúcnosti bude potreba neustále sledovania možných hrozieb a zlepšovania stávajících bezpečnostních řešení. Dle mého názoru se technologie honeypotů stane v brzké době základem a součástí většího množství komerčních bezpečnostních řešení, protože nabízí značný potenciál odhalování nových hrozeb. I z toho důvodu se domnívám, že tato diplomová práce postihuje dostatek informací pro pokračování v dalších projektech týkající se bezpečnosti.

Literatura

- [1] PROVOS, Niels, HOLZ, Thorsten. *Virtual Honeypots : From Botnet Tracking to Intrusion Detection*. [s.l.] : Addison Wesley Professional, 2007. 480 s. ISBN 978-0-321-33632-3.
- [2] GRIMES, Roger A. *Honeypots for Windows*. [s.l.] : [s.n.], 2005. 361 s. ISBN 1-59059-335-9.
- [3] SPITZNER, Lance. *Honeypots: Tracking Hackers*. [s.l.] : Addison Wesley, 2002. 480 s. ISBN 0-321-10895-7.
- [4] PROVOS, Niels. A Virtual Honeypot Framework. In *In Proceedings of the 13th USENIX Security Symposium*. [s.l.] : [s.n.], 2003. s. 1-14. Dostupný z WWW: <<http://www.citi.umich.edu/u/provos/papers/honeyd.pdf>>.
- [5] MOKUBE, Iyatiti, ADAMS, Michele. Honeypots: concepts, approaches, and challenges. In *ACM-SE 45: Proceedings of the 45th annual southeast regional conference*. [s.l.] : [s.n.], 2007. s. 321-326. Dostupný z WWW: <<http://delivery.acm.org/10.1145/1240000/1233399/p321-mokube.pdf?key1=1233399&key2=7341370621&coll=GUIDE&dl=GUIDE&CFID=68145946&CFTOKEN=73845660>>. ISBN 978-1-59593-6.
- [6] STOLL, Clifford. *The cuckoo's egg: tracking a spy through the maze of computer espionage*. [s.l.] : Doubleday, 1989. 326 s. ISBN 0-385-24946-2.
- [7] CHESWICK, Bill. An Evening with Berferd In Which a Cracker is Lured, Endured, and Studied. In *In Proc. Winter USENIX Conference*. [s.l.] : [s.n.], 1990. s. 163-174.
- [8] OUDOT, Laurent. *Wireless Honeypot Countermeasures* [online]. 2004-02-13 [cit. 2009-10-31]. Dostupný z WWW: <<http://www.securityfocus.com/infocus/1761>>.
- [9] RIDEN, Jamie, SEIFERT, Christian. *A Guide to Different Kinds of Honeypots* [online]. 2008-02-14 [cit. 2009-10-31]. Dostupný z WWW: <<http://www.securityfocus.com/infocus/1897/1>>.
- [10] RIDEN, Jamie. *Server honeypots vs. client honeypots* [online]. 08/16/2008 [cit. 2009-10-31]. Dostupný z WWW: <<http://www.honeynet.org/node/158>>.
- [11] RIDEN, Jamie. *Know Your Enemy: Malicious Web Servers* [online]. 08/16/2008 [cit. 2009-10-31]. Dostupný z WWW: <<http://www.honeynet.org/papers/mws>>.
- [12] *Know Your Enemy: Honeynets* [online]. 31 May, 2006 [cit. 2009-11-02]. Dostupný z WWW: <<http://old.honeynet.org/papers/honeynet/>>.
- [13] JIANG, Xuxian. Collapsar: A VM-Based Architecture for Network Attack Detention Center. In *In Proceedings of the 13th USENIX Security Symposium*. [s.l.] : [s.n.], 2004. s. 15-28. Dostupný z WWW: <<http://www.cs.purdue.edu/homes/dxu/pubs/USENIXSecurity04.pdf>>.
- [14] VRABLE, Michael, et al. *Scalability, Fidelity, and Containment in the Potemkin Virtual Honeyfarm*. [s.l.] : [s.n.], 2005. 15 s. Dostupný z WWW: <<http://cseweb.ucsd.edu/~savage/papers/Sosp05.pdf>>.
- [15] *Honeypot Software, Honeypot Products, Deception Software* [online]. 23 October 2009 , 23 October 2009 [cit. 2009-11-02]. Dostupný z WWW: <<http://www.honeypots.net/honeypots/products>>.
- [16] GARCIA, Jess. *Honeypots Tools* [online]. January 28, 2007 , January 28, 2007 [cit. 2009-11-02]. Dostupný z WWW: <<http://www.jessland.net/JISK/Honeypots/Tools.php>>.
- [17] *The Honeynet Project* [online]. 31.10.2009 [cit. 2009-10-31]. Dostupný z WWW: <<http://www.honeynet.org/project>>.

- [18] BARTEL, Petr. *Analýza dat získaných honeynety*. [s.l.], 2009. 43 s. Bakalářská práce.
- [19] PROVOS, Niels. *Developments of the Honeyd Virtual Honeypot* [online]. 1999-2004 , July 15 2008 06:48:04 PM [cit. 2009-12-10]. Dostupný z WWW: <<http://www.honeyd.org/>>.
- [20] *Microsoft Security Bulletin MS08-067 – Critical* [online]. October 23, 2008 , October 23, 2008 [cit. 2009-12-21]. Dostupný z WWW: <<http://www.microsoft.com/technet/security/Bulletin/MS08-067.msp>>.
- [21] HOUSER , Pavel. *Rok červa Conficker: 7 milionů nakažených počítačů* [online]. 03.11.09 [cit. 2009-12-21]. Dostupný z WWW: <<http://securityworld.cz/securityworld/rok-cerva-conficker-7-milionu-nakazenych-pocitacu-2059>>.
- [22] *Conficker Eye Chart* [online]. [2009] [cit. 2009-12-21]. Dostupný z WWW: <http://www.confickerworkinggroup.org/infection_test/cfeyechart.html>.
- [23] PORRAS, Phillip, SAIDI, Hassen, YEGNESWARAN, Vinod. *An Analysis of Conficker's Logic and Rendezvous Points* [online]. 4 February 2009 , 19 March 2009 [cit. 2009-12-21]. Dostupný z WWW: <<http://mtc.sri.com/Conficker/>>.
- [24] Červ Win32/Conficker letí světem!. *Viry.cz* [online]. 16.01.2009 [cit. 2009-12-21]. Dostupný z WWW: <<http://www.viry.cz/go.php?p=viry&t=novinka&id=2597>>.
- [25] *Conficker Working Group* [online]. [2009] [cit. 2009-12-21]. Dostupný z WWW: <<http://www.confickerworkinggroup.org/wiki/>>.
- [26] *Nmap - Free Security Scanner For Network Exploration & Security Audits* [online]. [1997] [cit. 2009-12-21]. Dostupný z WWW: <<http://nmap.org/>>.
- [27] *Smb-check-vulns.nse NSE Script* [online]. [2009] [cit. 2009-12-21]. Dostupný z WWW: <<http://nmap.org/nsedoc/scripts/smb-check-vulns.html>>.
- [28] HOUSER, Pavel. Desetina počítačů stále zůstává zranitelná červem Conficker. *Security World* [online]. 20.04.10, 2010, [cit. 2010-05-11]. Dostupný z WWW: <<http://securityworld.cz/securityworld/desetina-pocitacu-stale-zustava-zranitelna-cervem-conficker-2556>>.
- [29] Kaspersky Security Bulletin 2009. Malware Evolution 2009. *Securelist* [online]. 17 Feb 2010, [cit. 2010-05-14]. Dostupný z WWW: <http://www.securelist.com/en/analysis/204792100/Kaspersky_Security_Bulletin_2009_Malware_Evolution_2009>.
- [30] *MySQL 5.0 Reference Manual* [online]. 2010 [cit. 2010-05-16]. The DATETIME, DATE, and TIMESTAMP Types. Dostupné z WWW: <<http://dev.mysql.com/doc/refman/5.0/en/datetime.html>>.
- [31] RAMADAS, Manikantan. *TCPTRACE Manual* [online]. 24 August 2003 [cit. 2010-05-16]. TCPTRACE Manual. Dostupné z WWW: <<http://www.tcptrace.org/manual/index.html>>. *MySQL 5.0 Reference Manual* [online]. 2010 [cit. 2010-05-16]. The DATETIME,

Seznam příloh

Příloha 1. Použitá konfigurace Honeyd

Příloha 2. CD se zdrojovými kódy skriptu a výstupními výsledky analýzy

Příloha 1 - Použitá konfigurace Honeyd

```
create windows
set windows personality "Microsoft Windows XP Professional SP1"
set windows default tcp action reset
set windows default udp action reset
add windows tcp port 21 "sh
/mnt/data/home/xgalet02/honeyd/services/win32/win2k/msftp.sh $ipsrc
$sport $ipdst $dport"
add windows tcp port 23 "sh
/mnt/data/home/xgalet02/honeyd/services/unix/linux/suse8.0/telnetd.s
h $ipsrc $sport $ipdst $dport"
add windows tcp port 25 "sh
/mnt/data/home/xgalet02/honeyd/services/win32/win2k/exchange-smtp.sh
$ipsrc $sport $ipdst $dport"
add windows tcp port 80 "perl
/mnt/data/home/xgalet02/honeyd/services/win32/iis-0.95/iisemul8.pl
$ipsrc $sport $ipdst $dport"
add windows tcp port 110 "sh
/mnt/data/home/xgalet02/honeyd/services/win32/win2k/exchange-pop3.sh
$ipsrc $sport $ipdst $dport"
add windows tcp port 139 open
add windows tcp port 143 "sh
/mnt/data/home/xgalet02/honeyd/services/win32/win2k/exchange-imap.sh
$ipsrc $sport $ipdst $dport"
add windows tcp port 389 "sh
/mnt/data/home/xgalet02/honeyd/services/win32/win2k/ldap.sh $ipsrc
$sport $ipdst $dport"
add windows tcp port 445 "python
/mnt/data/home/xgalet02/honeyd/services/win32/i-smb445.py"
add windows tcp port 5901 "sh
/mnt/data/home/xgalet02/honeyd/services/win32/win2k/vnc.sh $ipsrc
$sport $ipdst $dport"
set windows uptime 3284460

bind 95.80.208.133 windows
bind 95.80.208.134 windows
bind 95.80.208.135 windows
bind 95.80.208.136 windows
```