UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

SHELL SENZOR VO VIRTUÁLNOM HONEYNETE

BAKALÁRSKA PRÁCA

Študijný program:	Informatika
Pracovisko (katedra/ústav):	Ústav informatiky
Vedúci diplomovej práce:	RNDr. JUDr. Pavol Sokol

Košice 2015

TOMÁŠ BAJTOŠ

Univerzita P. J. Šafárika v Košiciach
Prírodovedecká fakulta

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Tomáš Bajtoš
Študijný program: Informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: Bakalárska práca
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Shell senzor vo virtuálnom honeynete

Cieľ:
1. Analyzovať a porovnať vysoko-interaktívne honeypoty z pohľadu možností odchytávania dát
2. Analyzovať využitie virtualizácie pri odchytávaní dát
3. Navrhnúť a implementovať shell senzor vo virtuálnom honeynete

Literatúra:
[1] Joshi, R. C., and Anjali Sardana. Honeypots: A New Paradigm to Information Security. Science Publishers, 2011.
[2] Provos, Niels, and Thorsten Holz. Virtual honeypots: from botnet tracking to intrusion detection. Addison-Wesley Professional, 2007.
[3] Nemeth, Evi, Garth Snyder, and Trent R. Hein. Linux: kompletní příručka administrátora: 2. aktulizované vydání. Computer Press, 2008.

Kľúčové slová: honeypot, honeynet, virtuálny honeynet, ssh, shell senzor

Vedúci: RNDr. JUDr. Pavol Sokol
Ústav: ÚINF - Ústav informatiky
Riaditeľ ústavu: prof. RNDr. Viliam Geffert, DrSc.

Spôsob sprístupnenia elektronickej verzie práce:
dočasne neprístupná, po uplynutí bez obmedzenia

Dátum zadania: 17.05.2014

Dátum schválenia: 07.05.2015

prof. RNDr. Viliam Geffert, DrSc.
riaditeľ ústavu

Poďakovanie

Rád by som sa poďakoval vedúcemu svojej bakalárskej práce RNDr. JUDr. Pavlovi Sokolovi za cenné pripomienky a za obetavosť počas tvorby tejto práce. Tiež by som chcel poďakovať mojim blízkym osobám, ktorí mi boli veľkou oporou pri písaní tejto práce a mojim priateľom, ktorí mi poskytli cenné rady a pripomienky.

Abstrakt v štátnom jazyku

Honeypoty a honeynety sú výbornými a nenahraditeľnými nástrojmi, pomocou ktorých môžeme analyzovať metódy, nástroje a ciele útočníkov. Podľa interakcie s útočníkom delíme honeypoty na nízko-interaktívne a vysoko-interaktívne. Nízko-interaktívne honeypoty emulujú určitú službu (napr. ssh, web, mysql a pod). Vysoko-interaktívne sú reálne operačné systémy poskytnuté útočníkovi, pomocou ktorých môžeme zberať komplexnejšie a cennejšie údaje. Existuje niekoľko desiatok rôznych implementácii nízko-úrovňových honeypotov (Dionaea, HoneyD), ale len niekoľko vysoko-úrovňových honeypotov. V práci sme sa zamerali na v súčasnej dobe najviac používaný a bezpečný vysoko-úrovňový honeypot – HonSSH. Tento honeypot získava údaje zo zabezpečenej komunikácie SSH. V súčasnej dobe je použitie fyzického honeynetu, sieti niekoľkých honeypotov, administratívne a ekonomicky nákladné. Z tohto dôvodu sa využívajú virtuálne honeynety, pri ktorých virtuálne stroje tvoria honeypoty. Existuje niekoľko prístupov ako virtualizovať. My sme sa zamerali na virtualizáciu na úrovni operačného systému, ktorej výhody (nízke systémové nároky, priamy prístup k všetkým zdrojom virtuálneho stroja a pod.) je možné využiť pri návrhu a implementácii virtuálneho honeynetu. V práci sa zameriavame na shell senzor ako súčasť vysoko-interaktívneho honeypotu, ktorý zbiera údaje z komunikácie s útočníkom. Rozoberáme výhody nami zvolenej virtualizácie pre splnenie základných požiadaviek kladených na správne fungujúci honeynet. V práci je kladený dôraz tiež na kontrolu celého systému a prostredia, v ktorom sa pohybujú útočníci, keďže každý honeypot môže byť zneužitý na útok voči iným systémom mimo virtuálneho honeynetu. Preto sa snažíme udržať útočníkov vo vnútri honeynetu, alebo ich od neho odpojiť. Pre uľahčenie správy virtuálneho honeynetu sme implementovali webové rozhranie, ktoré zabezpečuje jednoduché zobrazenie údajov získaných zo shell senzora a manažment jednotlivých honeypotov.

Kľúčové slová: honeypot, honeynet, virtuálny honeynet, ssh, shell senzor

Abstrakt v cudzom jazyku

Honeypots and honeynets are perfect and irreplaceable tools which can help us to analyse the methods, tools and targets of attackers. According to level of interaction with the attackers, honeypots are divided on low-interaction honeypots and high-interaction honeypots. Low-interaction honeypots emulate certain services (eg. SSH, web, MySQL, etc.). High-interaction honeypots are real operating systems through which we can collect more comprehensive and valuable data. There are several dozen different implementations of low-interaction honeypots (eg. Honeyd, Dionaea, etc.), but only a few high-interaction honeypots. In this thesis, we focused on currently the most used and secure high-interaction honeypot named HonSSH. This honeypot captures data from a secure SSH communication. Nowadays, the use of physical honeynets, the networks of several honeypots, leads to high administrative and economic cost. Therefore, there are used virtual honeynets, in which the virtual machines are considered honeypots. There are several ways how to deal with virtualization. We focused on OS-level virtualization, the benefits of which (eg. low system requirements, direct access to all the virtual machine‘s resources, etc.) can be used in the design and implementation of virtual honeynets. In the thesis, we focused on shell sensor as part of the high-interaction honeypot that captures data from communication with the attackers. We discussed the benefits of the chosen virtualization to meet the basic requirements placed on properly functioning honeynet. In this thesis we also focused on the system control and the environment control in which the attackers operate, since each honeypot can be misused for an attack against other systems outside the honeynet. Therefore we try to keep the attackers inside the honeynet or to disconnect them from it. To facilitate the management of virtual honeynet we implemented a web interface that provides easy visualization of data captured from the shell sensor and the management of the honeypots.

Keywords: honeypot, honeynet, virtual honeynet, ssh, shell sensor

Obsah

Zoznam ilustrácií

Zoznam tabuliek

Zoznam skratiek a značiek

IDS **Intrusion Detection System**, system pre odhalenie prieniku

IP **Internet Protocol**, základný protokol pracujúci na sieťovej vrstve počítačových sietí

TCP **Transmission Control Protocol**, protokol riadenie prenosu, jeden zo základných sieťových protokolov

UDP **User Datagram Protocol**, používateľský datagramový protokol, nespoľahlivý sieťový protokol

SSH **Secure Shell** zabezpečený komunikačný protokol v počítačových sieťach, taktiež označenie pre samotný program

SSL **Secure Sockets Layer**, vrstva medzi transportnou a aplikačnou vrstvou, poskytuje zabezpečenie komunikácie

MAC **Message Authentication Code**, kryptografická funkcia podobná hašovacím funkciám

RAM **Random Access Memory**, pamäť s priamym prístupom

OS **Operating System**, operačný systém

SCP **Secure Copy**, protokol alebo aj program zabezpečujúci bezpečný prenos súborov

NAT **Network Address Translation**, preklad sieťových adries

CPU **Central Processing Unit**, centrálna procesorová jednotka

LXC **Linux Containers**, kontajnerová virtualizácia na úrovni operačného systému

UML **User-Mode-Linux**, virtualizácia linuxových operačných systémov

Úvod

Vynález menom internet pomohol vytvoriť svet, v ktorom dnes žijeme a v ktorom je obrovské množstvo zariadení pripojených do jednej veľkej počítačovej siete. Tieto zariadenia si medzi sebou vymieňajú rôzne informácie. Veľká časť týchto údajov je dôverná a nemala by sa dostať do nesprávnych rúk. Preto sa snažíme jednotlivé prvky tejto veľkej počítačovej siete chrániť a tak brániť útočníkom v ich činnosti. Za týmto účelom sa používajú rôzne firewally a IDS systémy. Často sa však stáva, že sa nájde spôsob, ako tieto ochrany prekonať. V takom prípade nám môžu byť veľmi nápomocné rôzne nástroje, ktoré dokážu odhaliť činnosť útočníka a zozbierať o útoku čo najviac užitočných informácii, ktoré potom môžeme použiť na zlepšenie našej ochrany. Honeypoty a honeynety sú nástroje, pomocou ktorých môžeme analyzovať metódy, ciele a nástroje útočníkov. V tejto práci sme sa zamerali práve na tieto nenahraditeľné nástroje a na nasadenie shell senzora, ktorý získava údaje zo zabezpečenej komunikácie SSH.

Cieľom našej práce je analyzovať a porovnať vysoko-interaktívne honeypoty z pohľadu možností odchytávania dát. Analyzovať využitie virtualizácie pri odchytávaní dát. A nakoniec navrhnúť a implementovať shell senzor vo virtuálnom honeynete.

Prácu sme rozdelili do šiestich kapitol. V prvej kapitole popisujeme základné vlastnosti honeypotov a ich rozdelenie, podľa rôznych kritérií. Taktiež sa v nej venujeme významu ich nasadenia v počítačovej sieti a výhodám, ktoré takto získame. Honeynetom a ich architektúre sa venujeme v druhej kapitole našej práce. V tretej kapitole popisujeme zabezpečený protokol SSH, ktorý sa používa pre vzdialenú správu operačných systémov. Aktuálnym riešeniam SSH honeypotov sa venujeme v štvrtej kapitole, kde rozoberáme ich výhody a nevýhody a nakoniec ich porovnávame na základe nami zvolených kritérií. V piatej kapitole opisujeme základné vlastnosti a typy virtualizácií.

Posledná kapitola tvorí jadro našej práce. Venujeme sa v nej návrhu a implementácii shell senzora vo virtuálnom honeynete. Využitiu možností virtualizácie na úrovni operačného systému vo virtuálnom honeynete. A nakoniec správe a manažmentu virtuálneho honeynetu pomocou webového rozhrania.

1 Honeypoty

Pod pojmom honeypot si môžeme predstaviť množstvo nástrojov, postupov, zariadení alebo systémov, ktorých hlavnou úlohou je nalákať útočníka a zachytiť jeho aktivitu. Nie sú to len samotné lákadla, ktoré vytvárajú dojem niečoho cenného, ale aj detekčné systémy. Lance Spitzner definuje honeypot ako „systémový zdroj, ktorého hodnota spočíva v jeho neoprávnenom alebo nedovolenom použití“ [27].

Honeypot emuluje, alebo sprístupňuje reálny operačný systém alebo službu, ktorá sa chová ako pasca a láka útočníka, aby napadol systém, zatiaľ čo zaznamenáva jeho aktivitu počas útoku a takisto aj po prieniku do systému [14]. Takto získané dáta vieme použiť na odhaľovanie nových postupov útočníkov a na prevenciu pred týmito útokmi. Často sa tiež stretávame s nasadzovaním honeypotov do produkčných sieti, aby odlákali pozornosť útočníkov od hodnotných dát alebo zdrojov.

1.1 Význam honeypotov

Teraz, keď už máme zadefinované, čo to honeypot je, môžeme si rozobrať jeho význam. Na rozdiel od mechanizmov, ako sú firewall alebo systémy na detekciu útokov (itrusion detection system, IDS), sa honeypoty nezameriavajú na konkrétny problém, ale prispievajú k celkovej bezpečnosti systému. Honeypoty majú svoje výhody, ale aj nevýhody, ktoré ovplyvňujú ich hodnotu [27]. V tejto podkapitole sa pozrieme na tieto výhody a nevýhody detailnejšie.

Honeypoty v závislosti od použitej technológie majú niekoľko výhod. My si spomenieme päť z nich:

- **Menší počet false-positives** – každá akcia spojená s honeypotom sa dá považovať za útok, alebo neoprávnené použitie, keďže honeypoty nemajú žiadnu produkčnú hodnotu.
- **Skorá detekcia** – malý výskyt falošných upozornení umožňuje rýchlu detekciu skutočných hrozieb.
- **Detekcia nových hrozieb** – pretože každé spojenie s honeypotom sa dá považovať za útok, doteraz neznáme hrozby sa odhalia veľmi rýchlo.
- **Nízke nároky na výpočtové zdroje** – keďže monitorujú a zachytávajú malé množstvo aktivít, nemajú honeypoty problém so zahltením zdrojov.

- **Redukované množstvo dát** – honeypoty zbierajú malé množstvo dát, avšak ich hodnota je vyššia.

S honeypotmi sú však spojené aj niektoré nevýhody a riziká, ako sú napríklad tieto:

- **Potreba útokov** – honeypoty sú bezcenné, ak na nich nik nezaútočí.
- **Obmedzený pohľad na vec** – honeypot vidí a zaznamenáva iba aktivity vedené priamo voči nemu.
- **Riziko zneužitia** – honeypot môže byť použitý pre napadnutie iného stroja.
- **Zanechanie stopy** – v niektorých prípadoch môže byť honeypot identifikovaný na základe očakávaných charakteristík správania.

1.2 Rozdelenie honeypotov

Existuje množstvo rôznych honeypotov, ktoré však vieme rozdeliť do niekoľkých hlavných kategórií, čo nám umožní sa v nich ľahšie zorientovať. Často sa však stáva, že sa nejaký honeypot objaví vo viacerých prelínajúcich sa kategóriách.

1.2.1 Rozdelenie na základe účelu

Na základe účelu alebo použitia sa honeypoty delia na produkčné honeypoty a výskumné honeypoty.

Produkčné honeypoty sú určené na zvýšenie bezpečnosti a ochrany informačných systémov. Sú používané organizáciami na zníženie možnosti prieniku do produkčnej siete pomocou skrytých alebo neobjavených zraniteľností a taktiež sa používajú na upozornenie administrátora na útok [14]. Organizácie potrebujú dbať na bezpečnostnú politiku, ktorú používajú, aby udržali nežiadaných útočníkov mimo ich informačný systém. Potrebujú zakázať nepoužívané služby, zavádzať rôzne záplaty systému, nasadzovať rôzne bezpečnostné prvky, ako je firewall alebo antivírusový systém. Produkčné honeypoty sú v takýchto prípadoch jednoducho použiteľné a slúžia ako prvotné detektory útokov, alebo ako kontrola zavedených bezpečnostných pravidiel. Taktiež sú schopné odhaľovať útoky z vnútra systému, ktoré samotný firewall nemusí odhaliť. Ich najväčšia hodnota je v detekcii a prevencii útokov, tým, že emulujú bezpečnostné zraniteľnosti, na ktoré sa útočník zameria. Keďže sú však nasadené v reálnych produkčných sieťach, nesmú sa stať zraniteľným miestom v sieti. Produkčné honeypoty nenahradia existujúce bezpečnostné riešenia, pretože zachytávajú iba aktivitu

vykonávanú na nich a nie na celej produkčnej sieti, ale vytvárajú veľmi silnú vrstvu ochrany, ktorá tieto riešenia dopĺňa.

Výskumné honeypoty poskytujú útočníkom oveľa viac priestoru, ako je to pri produkčných honeypotoch, a teda vystavujú počítačovú sieť väčšiemu riziku. Vďaka väčšiemu priestoru vieme sledovať prácu útočníkov. Vieme zistiť, ako postupujú, aké nástroje používajú, dokonca aj kto sú. Po získaní týchto informácii sa vieme proti ich útokom brániť. Tento typ honeypotov používajú prevažne univerzity, vlády, bezpečnostné zložky a bezpečnostné organizácie [14]. Jednou takouto organizáciou je nezisková organizácia **The Honeynet Project**, ktorá sa venuje výskumu v oblasti bezpečnosti. Honeypoty určené pre výskum sú vhodné pre odchytávanie automatizovaných útokov alebo analýzu malwaru. Vďaka väčšej miere zraniteľnosti vyžadujú viacej údržby a času. Na druhej strane však zozbierajú viac užitočných údajov.

1.2.2 Rozdelenie podľa úlohy

Podľa role, ktorú vykonávajú honeypoty, ich môžeme rozdeliť na klientské alebo serverové.

Serverové honeypoty sú najrozšírenejším typom honeypotov. Sú naprogramované tak, aby nevytvárali žiadne spojenia, až kým nie sú napadnuté. Ticho vyčkávajú na útočníka a stávajú sa aktívnymi až po začatí útoku [14]. Sú užitočné pri detekcii nových typov útokov, zberu škodlivého kódu, odchytávanie červov a podobne [10]. Tvoria značnú časť získavaných dát o útočníkoch. Väčšinou sú napádané pomocou automatických útokov. Útočníkov lákajú hodnotným obsahom, alebo nejakými zraniteľnosťami.

Klientské honeypoty sú na druhej strane veľmi aktívne. Používajú sa na odhaľovanie útokov voči klientskym staniciam. Tieto útoky využívajú hlavne zraniteľnosti v klientskych aplikáciách ako napríklad webový prehliadač, ktorý je po navštívení zákernej stránky napadnutý škodlivým kódom [10]. Klientsky honeypot je schopný odhaliť takéto servery. Alebo môže spúšťať škodlivý kód v chránenom prostredí a pozorovať, čo vykonáva so systémom.

1.2.3 Rozdelenie podľa miery interakcie

Podľa miery interakcie vieme honeypoty rozdeliť na nízko interaktívne a vysoko interaktívne.

Nízko interaktívne honeypoty sú, ako už názov napovedá, charakteristické nízkou mierou interakcie s útočníkom a tiež emuláciou falošných služieb. Útočníkovi neponúkajú reálny operačný systém ani reálne služby. Namiesto toho sú emulované nad vrstvou reálneho operačného systému. Tieto honeypoty sa dajú ľahko nasadiť a spravovať, taktiež nevyžadujú veľa systémových zdrojov [10]. Zberajú však limitované množstvo informácií a útočník ich môže ľahko odhaliť. No aj po odhalení, nemá útočník možnosť preniknúť do reálneho systému, čo zvyšuje jeho bezpečnosť. Poskytujú mu len obmedzené prostredie a aj po prieniku môže útočník zhodiť iba samotnú emuláciu služby. Používajú sa na odhaľovanie IP adries útočníkov, alebo k zberu rôznych štatistík o útokoch [14]. Keďže neponúkajú reálne služby, nevedia odhaliť nové útoky. Vedia však zaznamenať štatistiky o útokoch na zraniteľnosti, ktoré boli dopredu známe a ktoré ponúknu útočníkovi. Napríklad emulovaním SSH služby vieme získať informácie o počte pokusov o útok alebo použité prihlasovacie mená a heslá. Taktiež sa dajú použiť na odvedenie pozornosti útočníka od reálneho produkčného systému.

Vysoko interaktívne honeypoty poskytujú útočníkom prístup k reálnemu operačnému systému a jeho službám. Vďaka tomu dokážu zberať cennejšie informácie o útočníkoch a samotných útokoch. Ich detekcia je oveľa ťažšia, ale aj ich nasadenie a správa sú zložitejšie. Avšak po úspešnom nasadení sú veľmi prospešné v odchytávaní nových typov útokov, zraniteľností, malwaru, vírusov a podobne. Pre útočníkov sa správajú ako reálne produkčné systémy, teda aj ich atraktivita je vyššia [14]. Ideálnym prostredím pre takýto typ honeypotu je stav operačného systému po inštalácii, ktorý môže takto obsahovať množstvo neopravených zraniteľností. Na ich monitorovanie a kontrolu treba veľa nástrojov a prostriedkov systému. Taktiež vystavujú počítačovú sieť väčšiemu riziku, pretože po prieniku môže útočník s honeypotom pracovať ako s reálnym systémom. Aby sa to nestávalo, musia sa nasadzovať rôzne bezpečnostné nástroje ako napríklad firewall, ktorý bráni útočníkovi preniknúť ďalej do počítačovej siete, alebo útočiť na stroje mimo túto počítačovú sieť.

Ďalším typom honeypotov sú **stredne interaktívne honeypoty**. Tieto spájajú výhody oboch typov spomenutých vyššie, aj nízko interaktívnych, aj vysoko interaktívnych honeypotov. Sú viac pokročilé ako tie nízko interaktívne ale menej

pokročilé ako vysoko interaktívne. Taktiež neponúkajú reálny operačný systém, ani neponúkajú reálne služby, avšak implementujú väčšinu detailov aplikačného protokolu. Na každú požiadavku dostane útočník očakávanú odpoveď, ale s reálnym systémom sa nič nevykoná [27].

1.2.4 Rozdelenie podľa hardvérového nasadenia

Podľa toho na akom hardvéry sú nasadené, vieme honeypoty rozdeliť na fyzické a virtuálne.

Fyzický honeypot je reálny stroj, na ktorom bežia reálny operačný systém a reálne služby a samotný honeypot je pripojený k zvyšku sveta pomocou fyzického sieťového rozhrania. Keďže takýto typ honeypotu beží vždy práve na jednom fyzickom stroji, sú tieto honeypoty často spájané s vysoko interaktívnymi honeypotmi [14]. Pri nasadení sú málo praktické, pretože jeden honeypot potrebuje jeden fyzický stroj a tak stúpa cena samotnej realizácie aj prevádzky.

Na druhej strane sú **virtuálne honeypoty**. V tomto prípade beží na jednom fyzickom stroji viacero honeypotov [23]. Sú schopné monitorovať väčší rozsah IP adries a služieb. Je to možné vďaka možnosti emulovať sieťové rozhrania, rôzne služby a aj samotné stroje. Ich prevádzka je lacnejšia a aj samotná realizácia je lacnejšia, pretože nám stačí jeden výkonnejší fyzický stroj, namiesto mnohých malých strojov. Prenos takýchto honeypotov na iný stroj je jednoduchší, ako keby sme museli každý jeden prenášať osobitne.

2 Honeynety

Dva a viac honeypotov v rámci počítačovej sieti tvorí **honeynet**. Typické využitie honeynetu je na monitorovanie väčšej a oveľa rozmanitejšej siete, v ktorej jeden honeypot nemusí byť dostačujúci. Koncept honeynetu je jednoduchý. Požaduje vytvorenie počítačovej siete so štandardnými produkčnými systémami a umiestnenie tejto počítačovej siete za zariadenie kontrolujúce prístup a zaznamenávajúce prehľad o komunikácii, ako napríklad firewall [27]. Útočník môže napádať ktorýkoľvek systém vo vnútri počítačovej siete, ale nemal by mať možnosť útočiť na stroje mimo túto sieť. Honeynety sú veľmi užitočné a flexibilné v prevencii a detekcii útokov, takže môžu nahradiť každú rolu honeypotov [14]. Honeynety zachytávajú každú útočníkovu aktivitu na reálnom a komplexnom systéme. Zachytávajú všetko od úderov na klávesnicu až po každý paket, ktorý vstúpi alebo opustí honeynet. Všetky tieto údaje môžu byť potom použité na odpovedanie základných otázok o útočníkovi a jeho metódach.

Aj keď sú honeynety postavené na koncepte honeypotov, posúvajú sa o krok ďalej. Namiesto toho, aby boli tvorené iba jedným jediným systémom, sú honeynety reálnou sieťou mnohých systémov. Honeynet nie je produkt, ktorý si stačí nainštalovať, alebo zariadenie, ktoré stačí zapojiť do siete. Je to koncept architektúry, ktorá vytvára vysoko kontrolovanú sieť, do ktorej môžeme nasadiť akýkoľvek systém alebo aplikáciu. Pri vytváraní honeynetu, vytvárame prostriedok, ktorý má nulovú alebo takmer nulovú produkčnú sieťovú prevádzku. Všetko poslané smerom k honeynetu je potenciálny pokus o útok, alebo už samotný útok. A všetka komunikácia von z honeynetu napovedá o aktivite útočníka, alebo aplikácie vo vnútri siete. Teda o tom, že systém bol kompromitovaný. Do honeynetu môžeme umiestniť akýkoľvek systém alebo aplikáciu. Teda systém umiestnený v honeynete môže kopírovať skutočný produkčný systém.

2.1 Architektúra honeynetu

Každý honeynet má tri alebo aj štyri základné kritické body, ktorými sa musíme pri jeho tvorbe zaoberať. Ide o tieto body:

- kontrola údajov (data control),
- zber údajov (data capture),
- zhromaždenie údajov (data collection) a
- analýza údajov (data analysis).

Prvé dva body sú najdôležitejšie a musíme sa nimi zaoberať pri každom nasadení honeynetu. Tretiemu bodu sa venujeme, ak nasadzujeme viac honeynetov v distribuovanom prostredí [27]. Posledným bodom sa zaoberáme, ak už máme údaje zozbierané a chceme z nich vyťažiť cenné informácie o útokoch.

2.1.1 Kontrola údajov

Kontrola údajov je časť honeynetu, ktorá znižuje riziká pri útoku. Teda je to veľmi dôležitá časť, na ktorú sa nesmie zabúdať pri budovaní honeynetu. Spomínané riziká sú v tom, že ak sa už útočník dostane do vnútra honeynetu, môže odtiaľ útočiť na ďalšie systémy mimo neho. Túto vlastnosť sa snaží eliminovať práve kontrola údajov. Je dôležité, aby útoky boli vedené voči honeynetu a nie mimo neho, pretože nezabezpečením tejto vlastnosti podporujeme útočníkov v ich činnosti a navyše nemusíme z toho získať žiadne relevantné dáta, keďže útok sa odohráva mimo kontrolovaný systém [27].

Pri budovaní honeynetu máme pomerne veľa voľnosti, avšak je kriticky dôležité dodržať týchto osem vlastností (funkcionalít) [14]:

1. Implementácia aj automatickej aj manuálnej kontroly údajov.
2. Najmenej dve úrovne kontroly.
3. Udržiavanie stavu všetkých prichádzajúcich aj odchádzajúcich spojení.
4. Kontrola každej neautorizovanej aktivity.
5. Kontrolné pravidlá kedykoľvek konfigurovateľné administrátorom.
6. Neodhaliteľnosť kontrolných spojení útočníkom.
7. Aspoň dve metódy upozornení na aktivity v honeynete.
8. Vzdialená správa kontroly údajov.

2.1.2 Zber údajov

Ďalšou časťou honeynetu je zber údajov. Má na starosti monitorovanie a logovanie podozrivých aktivít v honeynete. Vďaka tomu máme tie údaje, ktoré sú neskôr analyzované a z ktorých získavame potrebné informácie o útočníkoch. Dôležité je zozbierať čo najviac údajov, ale na druhej strane nedovoliť útočníkom odhaliť túto činnosť systému [19]. Do úvahy tiež musíme brať, že väčšina spojení s honeynetom bude

šifrovaná, ako napríklad SSH alebo SSL spojenia, a teda honeynet sa s tým musí vedieť vysporiadať [19].

Pre efektívny zber údajov bolo definovaných niekoľko požiadaviek na honeynety [14]. Nižšie spomeniem niekoľko z nich:

- Údaje zachytené honeynetom nebudú uchovávané lokálne na honeypotoch.
- Znečistenie údajov v honeynete napríklad testovaním je neakceptovateľné.
- Aktivity siete, systému, aplikácii a užívateľov by mali byť uchovávane po dobu aspoň jedného roka.
- Administrátor má mať možnosť vzdialene pristúpiť k zozbieraným údajom a najlepšie v reálnom čase.
- Automatická archivácia zozbieraných údajov pre budúcu analýzu.
- Zavedenie štandardizovaných logov pre každý nasadený honeypot.
- Zdroje používane na zber údajov musia byť bezpečné a neodhaliteľné.

2.1.3 Zhromažďovanie údajov

Zhromažďovanie údajov je špecifická časť honeynetu, ktorá sa využíva pri nasadení viacerých honeynetov do distribuovaného prostredia. Pri využívaní distribuovaných honeynetov, ktoré sa môžu nachádzať kdekoľvek na svete, je dôležité všetky údaje z nich zozbierať a ukladať na jednom centrálnom mieste. Takto môže hodnota zozbieraných údajov exponenciálne vzrásť [19]. Takáto distribuovaná architektúra sa však používa menej často a častejšie sa stretneme s jednoduchými honeynetmi. Keď sa už ale rozhodneme pre vybudovanie takéhoto distribuovaného honeynetu, mali by sme dodržať tieto štyri zásady [14]:

- Zavedenie menných konvencií – každý honeynet má svoj identifikátor.
- Zabezpečený presun údajov bezpečnou cestou.
- Možnosť zhromažďovať údaje anonymne.
- Správna synchronizácia zhromaždených údajov.

2.1.4 Analýza údajov

Ak už máme údaje z útokov, je prirodzené, že chceme mať možnosť tieto údaje analyzovať a získať z nich cenné informácie. Analýza údajov sa zaoberá hľadaním nových typov útokov, vyšetrovaniu útokov po prieniku útočníka do systému

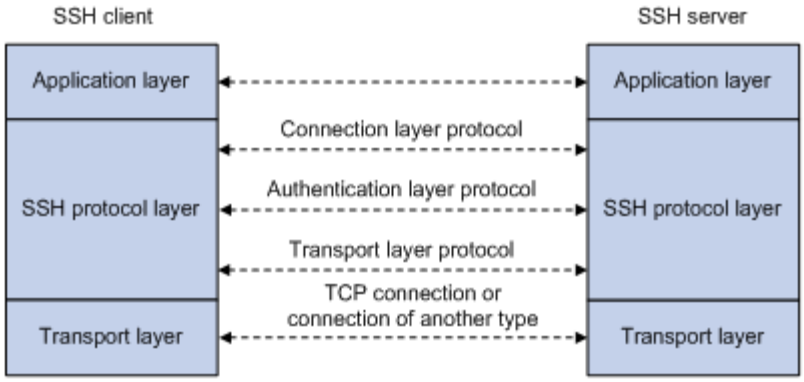
a novými trendmi medzi útočníkmi. Väčšina honeypotov však priamo nepodporuje analýzu zozbieraných dát [10]. Taktiež nie je veľa analytických nástrojov, ktoré by sa dali použiť pri analýze údajov z honeypotov a honeynetov. Teda aj keď zozbierame dostatok údajov, bez analytického modulu ich informačná hodnota klesá.

3 Secure Shell

Secure Shell alebo v skratke **SSH** je protokol pre bezpečný prístup k vzdialenému počítaču a tiež pre používanie iných bezpečných služieb siete prostredníctvom nezabezpečenej siete [30]. Tento protokol pozostáva z troch hlavných komponentov. Prvým je **protokol transportnej vrstvy**, ktorá zabezpečuje autentifikáciu servera, utajenie a integritu komunikácie. Taktiež ponúka voliteľnú možnosť kompresie, takže prenášané dáta sú menšie. Transportná vrstva beží typicky cez TCP/IP spojenie, ale môže byť tiež použitá s iným spoľahlivým dátovým spojením.

Ďalším z hlavných komponentov je **protokol autentifikácie užívateľa**. Zabezpečuje overenie užívateľa na strane klienta voči serveru. Beží nad transportnou vrstvou.

A tretím je **protokol spojenia**, ktorý zabezpečuje zlúčenie šifrovaných tunelov do niekoľkých logických kanálov. Tieto kanály môžu mať mnohoraké využitie. K dispozícii sú štandardné metódy na nastavenie zabezpečeného interaktívneho shell sedenia. Taktiež na preposielanie alebo inak tunelovanie voliteľných TCP/IP portov a X11 spojení. Beží nad autentifikačným protokolom.



Obr. 1 Protokol SSH [11]

Protokol SSH ponúka dve základné spôsoby autentifikácie užívateľa do systému. Jednou z nich je autentifikácia pomocou hesla. Pri tomto spôsobe autentifikácie je klient požiadaný o zadanie hesla, ktoré sa však v nejakej podobe musí nachádzať aj na serveri. Teda pri takomto overení klienta musíme predpokladať, že server nebol kompromitovaný, inak môže útočník odhaliť správnu kombináciu užívateľského mena a hesla, ktoré môže neskôr použiť na ďalšie útoky. Týmto rizikám sa dá predísť, ak používame autentifikáciu pomocou verejného kľúča. V takom prípade nemusí klient

zadávať žiadne heslá a jeho totožnosť je overená na základe vlastností asymetrickej kryptografie. Pri tejto autentifikácii musíme však predpokladať, že klient nebol kompromitovaný. Server tiež môže požadovať autentifikáciu klienta oboma týmito spôsobmi a vtedy sa bezpečnosť zvyšuje.

SSH protokol zabezpečuje, ako už bolo spomenuté utajenie a integritu komunikácie. **Utajenie** je zabezpečené pomocou symetrického šifrovania a podporuje mnohé známe a dostatočne silné šifry ako napríklad AES, 3DES alebo blowfish. **Integrita** je zabezpečená pomocou MAC funkcií. Keďže MAC používajú 32 bitové sekvenčné čísla, po odoslaní 2 na 32 paketov môže vzniknúť riziko úniku informácii. Preto sa po prenesení približne jedného gigabajtu údajov zmení kľúč šifrovania.

Pri tomto protokole sa často stretávame s útokmi typu **Man-in-the-middle**, kedy útočník nastraží stroj medzi server a klienta a snaží sa komunikáciu buď odpočúvať, alebo ju aj pozmeniť. Jedným z najznámejších spôsobov tohto útoku je, keď nie je zabezpečené bezpečné zdieľanie verejného kľúča servera [30]. Vtedy klient nevie jednoznačne povedať, že komunikuje so správnym serverom. Útočník nastraží medzi klienta a server stroj, ktorý vytvorí spojenie útočníka so zamýšľaným serverom a klient vytvorí spojenie s útočníkom. Takto môže útočník odchytávať celú komunikáciu a taktiež ju aj pozmeniť.

Protokol SSH sa v súčasnosti považuje za bezpečný, samozrejme pri správnej konfigurácii a je tiež najpoužívanejším protokolom pre vzdialený prístup k počítaču. Najpoužívanejšou implementáciou je open-source projekt pod názvom **OpenSSH** [22]. Ďalšie známe implementácie sú **Apache MINA** [3], **Copssh** [7], **Dropbear** [9] a ďalšie.

4 Dostupné riešenia

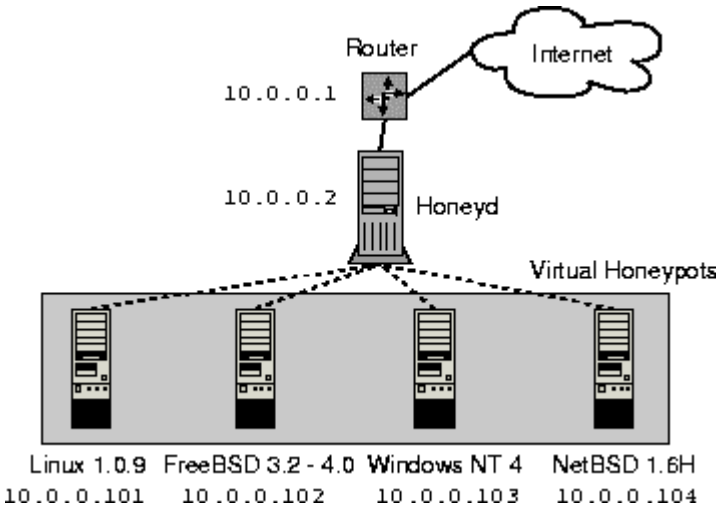
V tejto kapitole sa zameriame na porovnanie už existujúcich riešení. Postupne sa budeme venovať nízko interaktívnym, stredne interaktívnym a vysoko interaktívnym honeypotom. V závere kapitoly zhrnieme zistené poznatky.

4.1 Nízko interaktívne implementácie

Existuje mnoho honeypotov, ktoré riešia detekciu útokov na SSH služby. Väčšina je však zameraná len na samotné nadväzovanie spojenia a nie na komunikáciu po prihlásení. Zachytáva len pokusy o uhádnutie užívateľských mien a hesiel. Medzi takéto honeypoty patria práve nízko interaktívne honeypoty.

4.1.1 Honeyd

Prvým príkladom je honeypot **Honeyd** [12], ktorý dokáže emulovať rôzne služby na strane servera. Pri útoku však zachytáva len čas, IP adresy a porty. Tieto informácie sa hodia len, ak potrebujeme vedieť, ktoré služby sú napádané a odkiaľ. Podporuje však možnosť emulovať na jednom stroji mnoho rôznych virtuálnych strojov (Obr. 2). Vďaka tomu nie je problém vytvoriť počítačovú sieť tak, aby sa útočník domnieval, že máme oveľa viac strojov so spustenou službou SSH, alebo aj inou službou, ako reálne máme. Aktuálne už nie je aktívne vyvíjaný. Podobne funguje aj **KFsensor** [15], ktorý je však výlučne určený len pre operačný systém Windows.



Obr. 2 Virtuálne honeypoty v Honeyd [1]

4.1.2 Kojoney

Ďalšou možnosťou je honeypot **Kojoney** [20]. Ten je už priamo určený na detekciu útokov na službu SSH, ktorú ako jedinú dokáže emulovať. Slúži na pozorovanie brute-force prihlasovacích útokov. Po nadviazaní spojenia emuluje falošný terminál s jednoduchými príkazmi. Je však ľahko detekovateľný už na základe samotnej sieťovej komunikácie. Zaznamenáva pokusy o prihlásenie, IP adresy útočníkov, použité prihlasovacie mená a ak bolo prihlásenie úspešné aj zoznam príkazov z falošného terminálu. Jeho vývoj sa však zastavil v roku 2010.

4.1.3 Artillery

Projekt **Artillery** [4] je ďalšou z možností, ako odchytávať brute-force útoky na SSH server. Je to open-source nástroj napísaný v pythone, ktorý vznikol ako doplnok k bezpečnostnej ochrane servera a sťažuje útočníkovi prienik do systému. Jeho podstata je jednoduchá. Otvorí rôzne porty na serveri a keď sa niekto na tieto porty napojí, automaticky pridá jeho IP adresu do blacklistu a teda zablokuje ho až kým ho niekto odtiaľ nevymaže. Taktiež monitoruje SSH logy a zisťuje, či sa niekto nepokúša o brute-force útok. Ak nájde takýto útok, pridá IP adresu útočníka do blacklistu a zablokuje ho. Tiež umožňuje nakonfigurovať adresáre, v ktorých kontroluje zmeny. Pri každom z útokov ponúka možnosť informovať o tom pomocou emailu. Jeho hlavnou výhodou sú minimálne požiadavky na pamäť RAM a procesor. Je určený hlavne pre linuxové operačné systémy, aj keď niektoré funkcie sú podporované aj pod OS Windows. Informácie, ktoré loguje sú dosť chabé, je to len IP adresa útočníka, port, na ktorý sa útočilo a čas útoku. Avšak je ešte stále v aktívnom vývoji.

4.1.4 Ďalšie implementácie

Spomenieme ešte dva veľmi podobné riešenia a to **SSHpot** [28] a **Secure Honey** [26]. Sú to malé projekty, napísané v programovacom jazyku C, ktoré využívajú knižnicu libssh. Sú zamerané na zber IP adries útočníkov a užívateľských mien a hesiel, ktoré útočník pri útoku použil. Získané údaje ukladajú do súborov a pridávajú k nim časovú pečiatku. Teda nemajú žiadnu špeciálnu podporu pre logovanie. Hneď po autentifikácii útočníka, sa spojenie uzavrie. Ich hlavnou výhodou je, že sú ľahko konfigurovateľné, úplne jednoduché a nerobia nič navyše, čo by zaťažovalo systém.

4.2 Stredne interaktívne implementácie

Aby sme mohli zbierať aj údaje z priebehu útoku, začali vznikať viac pokročilé implementácie honeypotov. Implementujú aplikačný protokol, na ktorom sú postavené a emulujú takto celú službu s takmer všetkými detailmi. Niekedy dokonca emulujú aj celý operačný systém. Keďže sú o trochu viac pokročilejšie ako nízko interaktívne honeypoty, nazývajú sa stredne interaktívne honeypoty.

4.2.1 Kippo

Takýmto honeypotom a hlavne aktívnym projektom je honeypot **Kippo** [16]. Taktiež emuluje SSH službu na strane servera. Zaznamenáva brute-force prihlasovacie útoky na SSH službu a navyše aj komunikáciu útočníka so serverom, teda kroky, ktoré vykonal útočník na systéme po úspešnom prihlásení. Emuluje falošný súborový systém linuxovej distribúcie (napríklad Debian 5.0) a dáva možnosť pridať falošné súbory. Útočník si môže čítať obsah súborov ako napríklad /etc/passwd a podobne. Útočník taktiež môže pridávať nové súbory a aj ich mazať. Ukladá všetky súbory, ktoré boli počas SSH spojenia stiahnuté. Celá komunikácia útočníka so serverom je uložená vo forme binárneho súboru, ktorý je možné pomocou priloženej aplikácie prehrať. Taktiež podporuje ukladanie všetkých dát do databázy MySQL a tak uľahčuje nasledujúcu analýzu. Pri väčšej záťaži nastávajú problémy, pretože Kippo nebol na to dizajnovaný . Pri prihlásení približne 100 útočníkov, majú ďalší útočníci problém s nadviazaním spojenia [10]. Tento honeypot je veľmi užitočný pre zber údajov o útočníkoch, a preto je aj často nasadzovaný.

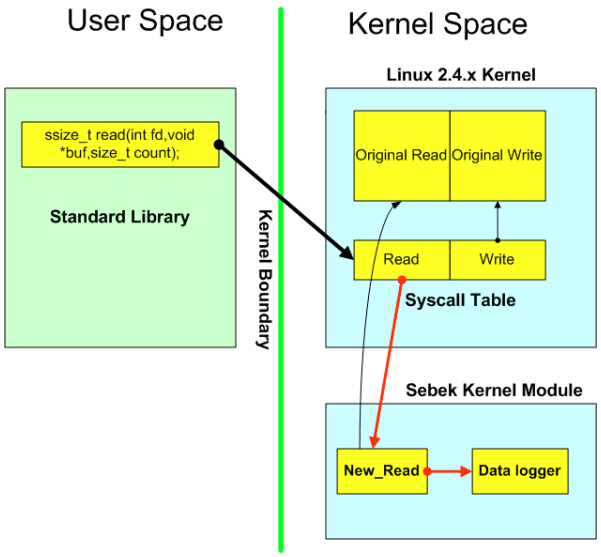
4.3 Vysoko interaktívne implementácie

Všetky doteraz spomenuté honeypoty boli nízko alebo stredne interaktívne, teda služby maximálne iba emulovali, resp. čiastočné implementovali. Takto sa dajú zbierať informácie o útokoch o ktorých sme už pred nasadením vedeli, ale nevieme zaznamenať takzvané zero day útoky, teda také, ktoré ešte neboli zaznamenané. Tieto cennejšie informácie vieme získavať vďaka vysoko interaktívnym honeypotom, pretože ponúkajú útočníkovi interakciu s reálnym systémom.

4.3.1 Sebek

Medzi takéto honeypoty patrí aj honeypot **Sebek** [25]. Zaznamenáva kedy sa útočník dostal do systému, ako sa tam dostal a čo robil po získaní prístupu. Na

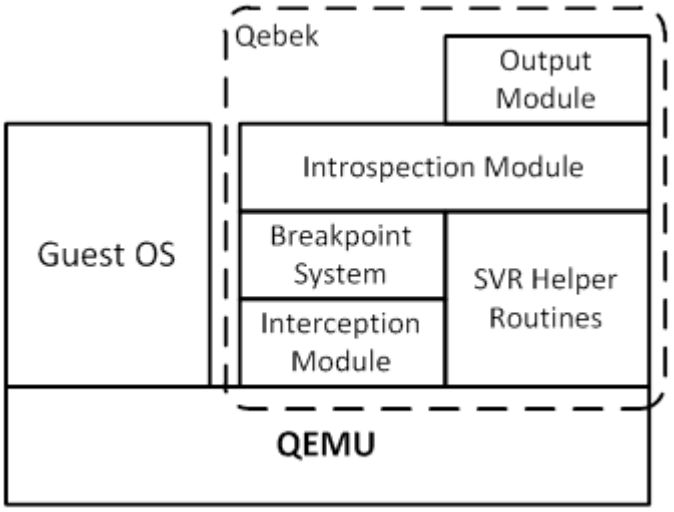
monitorovanie aktivít používa architektúru klient-server a je možné ho nasadiť ako na Linux tak aj na Windows. Klientská časť zachytáva informácie z honeypotu a posiela ich serverovskej časti. V jadre systému je nahradené systémové volanie read(), vďaka čomu je tento honeypot schopný zachytávať všetky údaje prístupné cez toto volanie read() (Obr. 3). Nová funkcia zavolá pôvodnú funkciu, skopíruje jej obsah do zásobníka, pridá hlavičku a pošle ako UDP paket serveru. Sebek monitoruje taktiež sieťovú komunikáciu. Server potom zbiera všetky tieto pakety a spracuje ich. Honeypot je postavený tak, že server môže zberať pakety od všetkých klientov v sieti, pričom nezáleží na operačnom systéme na ktorom sú nasadené. Vďaka tomu ako je Sebek postavený vieme napríklad logovať SSH komunikáciu, obnovovať súbory poslané pomocou SCP, alebo zaznamenať všetky heslá zadané útočníkom. Aby útočník nezistil, že na serveri beží Sebek, musí sa vedieť zamaskovať. Na skrytie klientského modulu sa nainštaluje ďalší modul, ktorý manipuluje s listom nainštalovaných modulov tak, že Sebek je vymazaný z tohto listu. Taktiež zabraňuje útočníkovi vidieť pakety, ktoré posiela serveru a to nielen od daného honeypotu, ale od všetkých v sieti. Napriek tomu všetkému je Sebek stále detekovateľný v systéme, keďže sa v ňom priamo nachádza. Informácie získane z tohto honeypotu sú veľmi užitočné, pretože vieme monitorovať každú aplikáciu nainštalovanú v operačnom systéme. Aj napriek užitočnosti takýchto dát, Sebek nie je aktívne vyvíjaný už dlhšiu dobu. Linuxový modul jadra a nástroje serverovej časti neboli aktualizované od roku 2009.



Obr. 3 Systémové volania v honeypote Sebek [25]

4.3.2 Qebek

Ďalším z rodiny vysoko interaktívnych honeypotov je **Qebek** [24]. Je to honeypot založený na QEMU emulácii a zameraný na nahradenie honeypotu Sebek. Hlavným dôvodom vývoja tohto honeypotu je viditeľnosť Sebeku. Presunutím monitorovacieho senzoru z jadra operačného systému do pozadia virtualizačnej vrstvy sa stáva ťažšie pre útočníka zistiť, že je monitorovaný. Taktiež sleduje systémové volania, podobne ako Sebek, aby získal informácie o aktivitách útočníka vykonaných v operačnom systéme, ale na rozdiel od Sebeku, útočník vie zistiť len to, že je na virtuálnom stroji. Po naštartovaní operačného systému Qebek pridá na začiatok funkcie systémového volania breakpoint. Po zavolaní funkcie monitorovacie senzory interpretujú údaje z pamäte RAM a získajú požadované informácie. Momentálne však podporuje len operačný systém Windows XP Service Pack 2. Qebek používa rovnaký formát dát ako Sebek, takže všetky monitorovacie aplikácie sú kompatibilné. Nepodporuje posielanie získaných informácii pomocou siete, čo však nebráni vytvoreniu takéhoto modulu. Takisto ako Sebek ani Qebek nie je už dlhšiu dobu aktívne vyvíjaný.

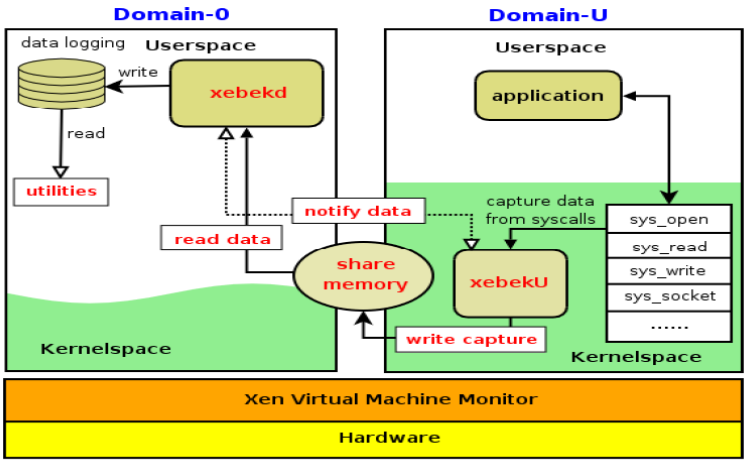


Obr. 4 Moduly honeypotu Qebek v qemu virtualizácii [24]

4.3.3 Xebek

Existuje taktiež verzia tohto honeypotu, ktorý je určený pre virtualizáciu Xen a volá sa **Xebek** [33]. Tak ako Qebek aj Xebek sa snaží znížiť počet možností, ako môže útočník zistiť, že ide o honeypot. Na rozdiel od Sebeku nepotrebuje skrytý modul v jadre operačného systému. Odchytáva údaje pomocou upravených systémových volaní. Kvôli

tejto úprave je jadro o trošku väčšie. Tiež nepoužíva sieť na prenos týchto údajov. Prenáša ich pomocou zdieľanej pamäte medzi doménou-0 a doménou-U. Na doméne-0 beží centrálny logovací server, kompatibilný so Sebekom a na doméne-U beží samotný honeypot. Vďaka tomuto riešeniu, sa odstránilo mnoho problémov, ktoré mal Sebek. Podobne ako pri predošlých dvoch, vývoj aj tohto honeypotu už nie je aktívny.

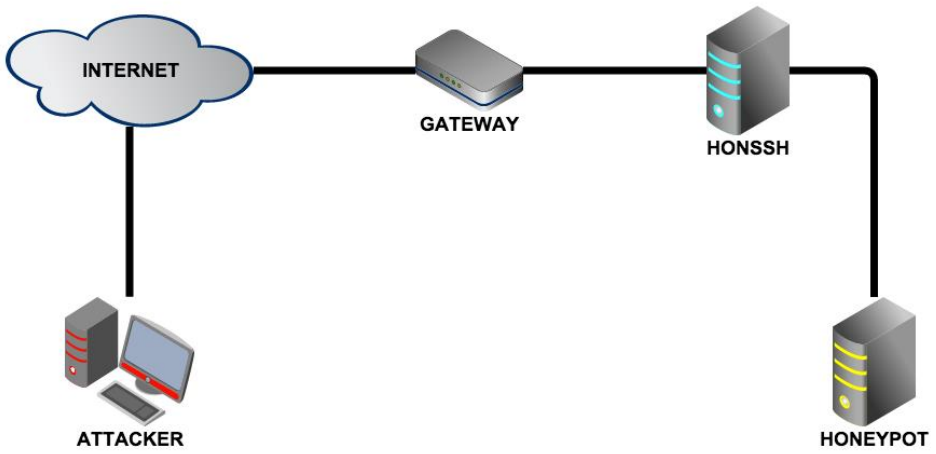


Obr. 5 Honeypot Xebek a výmena dát medzi virtuálnymi strojmi [33]

4.3.4 HonSSH

Najnovším projektom je honeypot **HonSSH** [13]. Je to vysoko interaktívny honeypot, ktorý je však postavený na inom princípe ako predošlé tri. Nemonitoruje priamo operačný systém, ale komunikáciu útočníka so serverom. Je vložený medzi útočníka a honeypot a vytvára medzi nimi dve SSH spojenia (Obr. 6). Záznamy zo všetkých spojení ukladá buď do súborov, databázy alebo sa dajú poslať emailom. Ponúka možnosť prihlásenia útočníka aj keď neuhádne správne heslo, avšak s týmto heslom nebude mať šancu použiť sudo a teda získať privilegované oprávnenia. Po úspešnom prihlásení je celá komunikácia uložená do TTY logu, ktorý je prevzatý z Kippo, a teda sa dá prehrať pomocou malého programu z Kippo honeypotu. Takisto sťahuje všetky prenášané súbory, sťahované napríklad pomocou príkazu wget. Ponúka tiež možnosť integrovať vlastné skripty pre logovací výstup. Vďaka tomu, ako je HonSSH postavený, je možné server, na ktorom beží použiť zároveň ako NAT smerovač alebo firewall. Ak sa útočník prihlási na iný stroj cez honeypot, celá táto komunikácia ostane zaznamenaná vďaka záznamu spojenia s honeypotom. Nepodporuje však privátne kľúče, teda autentifikácia pomocou nich musí byť na honeypote vypnutá. Keďže pracuje len s komunikáciou na ceste, je jedno, aký operačný systém je použitý na honeypote.

Nevytvára ani žiadne procesy, ani neinštaluje žiaden softvér na stroj, na ktorý sa útočník pripája. Prakticky je teda nemožné zistiť, že je to honeypot. Je to ešte len veľmi čerstvý projekt, preto má zopár chýb, ale momentálne asi jedno z najlepších riešení, aké sa dá jednoducho nasadiť.



Obr. 6 Nasadenie HonSSH v sieti ako router [17]

4.4 Zhodnotenie

Každý z honeypotov spomenutý v tejto kapitole vznikol z nejakého dôvodu. Niektoré preto, že niečo podobné neexistovalo, iné zase opravovali chyby a zraniteľnosti predchodcov. Preto sa ťažko určuje, ktoré z týchto riešení použiť. Každý honeypot pracuje odlišným spôsobom a zbiera iné typy údajov. Záleží aj na operačnom systéme, na ktorom bude honeypot spustený. Pre ďalšie spracovanie údajov je vhodné, ak honeypot podporuje ukladanie do databázy. Ak potrebujeme zbierať podrobné údaje o priebehu útoku, je vhodné použiť vysoko-interaktívny honeypot, ale ak nám stačia iba údaje odkiaľ a koľko útokov nastalo, postačia aj tie najjednoduchšie honeypoty. Taktiež, ak honeypot vyžaduje nejaký druh virtualizácie, tak ju musíme použiť. No najdôležitejšími parametrami sú ťažká odhaliteľnosť a aktívny vývoj. Bez nich sa aj tie najpodrobnejšie údaje môžu stať bezcennými, napríklad ak útočník odhalí honeypot, tak sa začne správať úplne inak, alebo ho zneužije pre ďalšie útoky bez nášho vedomia. Vyššie spomínané honeypoty môžeme rozdeliť podľa niekoľkých základných vlastností. Najdôležitejšou vlastnosťou je miera interakcie, pretože tá určuje možností zberu údajov a ich informačnú

cenu. Dôležitý je aj aktívny vývoj. Každý program je nedokonalý a teda musíme opravovať chyby, ktoré sa v ňom nájdu. Ďalšími vlastnosťami, v ktorých sa líšia, sú operačný systém, ktorý je potrebný pre ich beh. Taktiež potreba virtualizácie pre nasadenie týchto honeypotov. Dôležitým parametrom je aj možnosť použitia databázy pre ukladanie zozbieraných údajov. Ak dokáže útočník ľahko odhaliť honeypot, nie je táto implementácia vhodná pre široké nasadenie v počítačových sieťach. Preto je aj táto vlastnosť vhodná pre popísanie honeypotov. Honeypoty sme podľa nej rozdelili do troch skupín. V prvej sú honeypoty, ktoré je ľahké odhaliť. Už na prvý pohľad je jasné, že môže ísť o honeypot. V druhej skupine sú honeypoty so strednou možnosťou odhaliteľnosti. Teda ide o tie honeypoty, ktoré sa útočníkovi začnú zdať podozrivé až po vykonaní určitých činností. A v poslednej skupine sú honeypoty, ktoré by mali byť na nerozoznanie od skutočných systémov. V tabuľke nižšie (Tab. 1) popisujeme tieto vlastnosti honeypotov, spomenutých v tejto kapitole.

Tab. 1 Vlastnosti honeypotov

Názov honeypotu	Miera interakcie	Aktívny vývoj	OS	Potreba virtualizácie	Databáza	Záznam o priebehu	Odhaliteľnosť
Honeyd	Nízka	Nie	Linux / Windows	Nie	Áno	Nie	Ľahká
KFsensor	Nízka	Áno	Windows	Nie	Nie	Nie	Ľahká
Kojoney	Nízka	Nie	Linux	Nie	Nie	Áno	Stredná
Kippo	Stredná	Áno	Linux	Nie	Áno	Áno	Stredná
Artillery	Nízka	Áno	Linux / Windows	Nie	Nie	Nie	Ľahká
SSHpot	Nízka	Nie	Linux	Nie	Nie	Nie	Ľahká
Secure Honey	Nízka	Nie	Linux	Nie	Nie	Nie	Ľahká
Sebek	Vysoká	Nie	Linux / Windows	Nie	Áno	Áno	Ľahká
Qebek	Vysoká	Nie	Linux	Áno	Áno	Áno	Stredná
Xebek	Vysoká	Nie	Linux	Áno	Áno	Áno	Ťažká
HonSSH	Vysoká	Áno	Linux	Nie	Áno	Áno	Ťažká

5 Virtualizácia

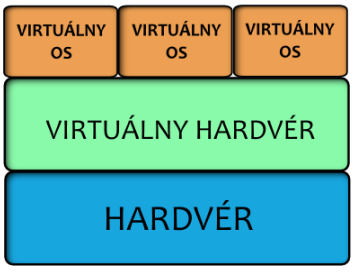
Virtualizáciu môžeme definovať ako technológiu, ktorá nám dovoľuje súčasný beh viacerých virtuálnych strojov na jednom fyzickom stroji [18]. Každý z týchto virtuálnych strojov môže mať vlastný operačný systém, a tie potom bežia paralelne. Takýto stav vieme dosiahnuť vďaka zdieľaniu fyzických zdrojov, medzi ktoré patrí napríklad CPU, pamäť, pevný disk a podobne.[2] Softvér, ktorý riadi tieto zdroje sa nazýva hypervisor. Je to medzivrstva medzi fyzickými zdrojmi a virtuálnymi strojmi. Pri virtualizácii sa môžeme stretnúť ešte s pojmami **hostiteľský operačný systém**, teda operačný systém nad ktorým beží virtualizácia a **virtualizovaný operačný systém**, ktorý beží vo vnútri virtuálneho stroja. Virtualizačných nástrojov je mnoho a dajú sa rozdeliť do týchto štyroch skupín:

- plná virtualizácia
- para-virtualizácia
- emulácia
- virtualizácia na úrovni operačného systému

Ďalej si rozoberieme tieto štyri typy virtualizácii a predstavíme si tiež niektoré ich odlišnosti.

5.1 Emulácia

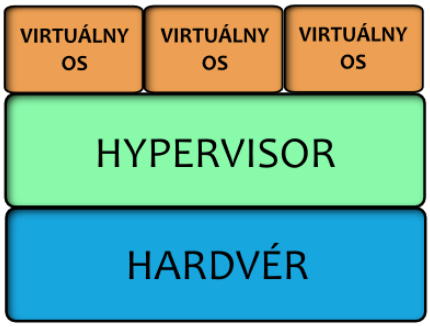
Emulácia je najkomplexnejšia z virtualizácií, pretože emuluje aj hardvér, na ktorom bežia virtuálne stroje. Je najpomalšou z virtualizácií. Jej hlavnou výhodou je, že dokáže emulovať rôzne druhy hardvéru. Táto vlastnosť je výhodná, ak potrebujeme virtuálne stroje so špecifickým hardvérom (napríklad ARM procesorom). Na virtuálnych strojoch beží nemodifikovaný operačný systém. To znamená, že pomocou emulácie môžeme vo virtuálnom stroji rozbehať akýkoľvek operačný systém. Ako príklad najznámejšieho emulátora sa dá uviesť QEMU.



Obr. 7 Emulácia

5.2 Plná virtualizácia

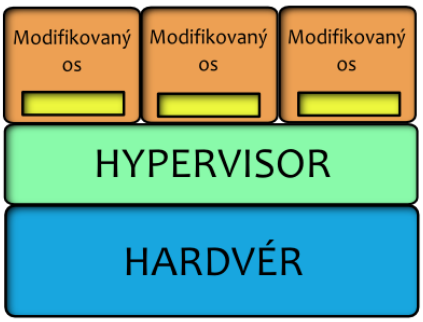
Plná virtualizácia, tiež známa ako **natívna virtualizácia** umožňuje beh nemodifikovaného operačného systému na simulovanom hardvéri. Simulovaný procesor je však v tomto prípade rovnaký ako fyzický. Táto virtualizácia vyžaduje podporu hardvéru, napríklad pri procesoroch od Intelu je táto technológia nazvaná Intel VT-x [6]. Plná virtualizácia je rýchlejšia ako emulácia, no aj tak nedosahuje výsledky behu priamo na hardvéri, pretože tam stále vystupuje hypervisor. Túto technológiu využíva mnoho nástrojov ako napríklad VirtualBox, VMware a podobne.



Obr. 8 Plná virtualizácia

5.3 Para-virtualizácia

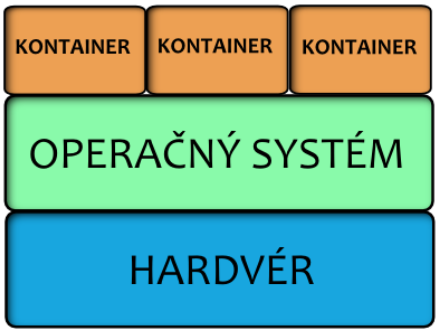
Para-virtualizácia je ďalším typom virtualizácie. V tomto prípade je potrebné upraviť jadro virtualizovaného operačného systému. Musia byť nahradené niektoré inštrukcie, aby komunikácia prebiehala priamo s hypervisorom [5]. Takto môžeme fungovať s rôznymi operačnými systémami, ale strácame flexibilitu, pretože takto upravený a rekompilovaný operačný systém nemôže bežať priamo na hardvéry. Na druhej strane je para-virtualizácia výkonnostne na tom lepšie ako plná virtualizácia. Najznámejšie para-virtualizácie sú Xen a UML.



Obr. 9 Para-virtualizácia

5.4 Virtualizácia na úrovni operačného systému

Ďalším typom virtualizácie je virtualizácia na úrovni operačného systému (**kontajnerová virtualizácia**). Na rozdiel od predošlých, táto virtualizácia nie je postavená na hypervízore. Celá virtualizácia je založená na vrstve operačného systému, a teda všetky virtuálne stroje zdieľajú spoločné jadro hosťujúceho systému [32]. V tomto prípade sa virtuálne stroje nazývajú kontajnery. Aj napriek zdieľaniu operačného systému sa všetky procesy v kontajneroch správajú, ako keby bežali v samostatnom systéme. Výhoda tejto virtualizácie je v minimálnom zaťažení celého systému, a teda dosahuje vysoký výkon. Všetky virtuálne operačné systémy však musia mať rovnaké jadro. Teda neumožňuje beh dvoch rôznych operačných systémov vo virtuálnych strojoch. Taktiež izolácia virtuálnych strojov je slabšia, preto sa musia nasadzovať rôzne ochranné systémy. Najznámejšími implementáciami sú OpenVZ, Linux-VServer a LXC. V našej práci sa zameriame práve na tento typ virtualizácie, konkrétnejšie použijeme virtualizáciu LXC.



Obr. 10 Virtualizácia na úrovni operačného systému

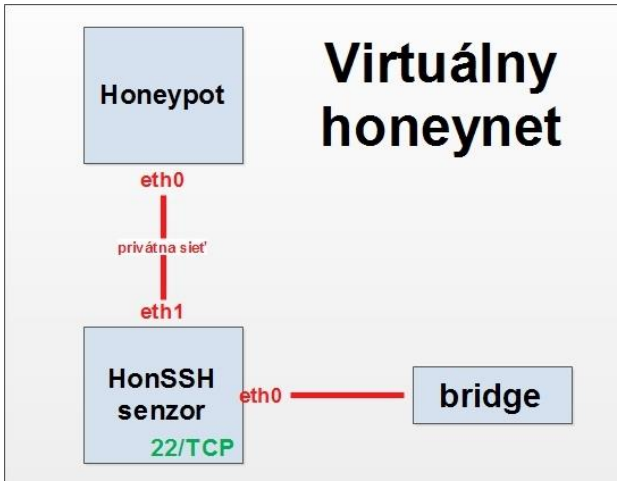
6 Návrh a implementácia shell senzora

V tejto časti si predstavíme samotný návrh shell senzora v honeynete a implementáciu niektorých zásadných častí tohto honeynetu. V predošlých kapitolách sme si predstavili základne pojmy a tiež implementácie SSH honeypotov. Taktiež sme sa oboznámili s pojmom virtualizácia. Pri návrhu potrebných častí rátame s virtuálnym honeynetom postavenom na virtualizácii na úrovni operačného systému.

6.1 HonSSH ako shell senzor v honeynete

V našej práci sme sa zamerali na HonSSH ako na senzor monitorujúci SSH komunikáciu. Aj keď existuje viacero spôsobov, ako pristupovať k správe operačného systému, v našom honeynete nám postačuje monitorovať SSH komunikáciu, pretože je jedinou, okrem priameho fyzického prístupu, ako môže útočník pristupovať a kontrolovať operačný systém v honeypote. Po prihlásení monitorujeme celú činnosť útočníka prebiehajúcu v systéme.

Ako sme už spomenuli v kapitole (4.3.4), HonSSH potrebuje pre svoje fungovanie dva stroje. Jedným je samotný honeypot, kde nie je podstatné, aký operačný systém si zvolíme a na druhom stroji je spustený senzor HonSSH. Pre oba tieto stroje sme si zvolili linuxovú distribúciu Ubuntu 14.04 [31]. Na honeypote beží štandardná inštalácia Ubuntu, a o SSH spojenia sa stará OpenSSH server [openssh]. OpenSSH služba beží na porte 22, čo je štandardný port pre túto službu. Honeypot má jedno sieťové rozhranie (eth0) pripojené priamo k druhému stroju na ktorom beží HonSSH. Medzi nimi je nastavená privátna sieť.



Obr. 11 Návrh honeypotu

HonSSH stroj je spustený tiež so štandardnou inštaláciou Ubuntu. O SSH službu sa však nestará openSSH, ale samotné HonSSH, ktoré je nakonfigurované, aby bežalo na porte 22, tak ako v honeypote. Tento stroj má dve sieťové rozhrania. Prvé rozhranie (eth0) je pripojené k internetu a má pridelenú verejnú IP adresu. Druhé rozhranie (eth1) je pripojené k privátnej sieti a prepojené s honeypotom. V operačnom systéme je povolené smerovanie medzi týmito dvoma rozhraniami, a taktiež je nastavená NAT tabuľka pre odchádzajúce pakety. Takto sme zabezpečili prístup honeypotu k internetu. Všetky SSH spojenia prechádzajú cez HonSSH, ktoré ich potom preposiela na honeypot. Nastavené máme taktiež takzvané „**advanced networking**“**,** ktoré zabezpečí, že všetky pakety prichádzajúce na honeypot budú vyzerať, že prišli od útočníka. Nastavili sme logovanie do súborov a tiež do MySQL databázy. MySQL databáza beží lokálne na HonSSH stroji a prístup k nej je povolený len z localhostu. V našom prípade sme nepoužili odporúčané rozmiestnenie súborov HonSSH, ale upravili sme implementáciu tak, aby bolo možné mať konfiguračné súbory v adresári „/etc“, a aby sa dal HonSSH spúšťať kdekoľvek v celom systéme a nie v danom adresári.

6.2 Honeynet založený na virtualizácii na úrovni operačného systému

Ako je vyššie spomenuté, HonSSH vyžaduje dva stroje, aby sme ho mohli nasadiť. Keďže sme sa ho rozhodli použiť vo virtuálnom honeynete, teda v honeynete postavenom na virtualizácii, ušetrili sme tieto zdroje a všetko môže bežať na jednom fyzickom stroji. Z rôznych typov virtualizácie sme si vybrali virtualizáciu na úrovni operačného systému. Tá najmenej zaťažuje fyzický stroj. Ako hosťujúci operačný systém sme zvolili linuxovú distribúciu Ubuntu 14.04. Prvotne sme však chceli použiť linuxovú distribúciu Debian [8], tá však v stabilnej verzii používa staršie balíčky a aj samotné jadro je neaktuálne. Keďže sme zvolili LXC (LinuxContainers) vo verzii 1.0 [21] ako konkrétnu implementáciu virtualizácie, potrebovali sme linuxové jadro vo verzii aspoň 3.12 a niekoľko balíčkov, tiež vo vyššej verzii, ako nám ponúkala distribúcia Debian. LXC nám ponúka dva typy virtuálnych strojov. Prvým sú **privilegované kontajnery**, ktoré sa spúšťajú pod superpoužívateľom (root user) na hosťujúcom operačnom systéme a ktorý je totožný so superpoužívateľom vo virtuálnom stroji. Táto vlastnosť pre nás znamenala riziko, pretože chceme poskytnúť celý virtualizovaný operačný systém útočníkom a takto by mali možnosť preniknúť do vyššej vrstvy, teda do hosťujúceho operačného systému

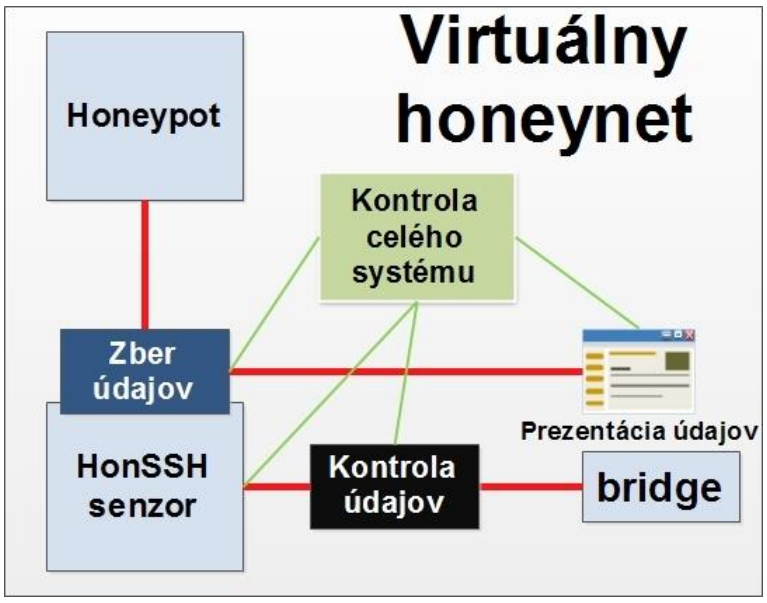
a mať práva ako superpoužívateľ. Preto používame **neprivilegované kontajnery**, ktoré sú spúšťané pod neprivilegovaným užívateľom, teda iným ako superpoužívateľ. Taktiež pre užívateľov v kontajneroch namapuje uid a gid na iný rozsah, ako sa používa na hosťujúcom operačnom systéme a tak aj keď prenikne útočník o úroveň vyššie, bude mať práva ako „nobody“ užívateľ.

6.2.1 Návrh honeynetu

Pretože sme sa v našej práci zamerali na shell senzor, implementovali sme len niektoré základné časti honeynetu. Významnou časťou je práve senzor zaznamenávajúci SSH komunikáciu. Ďalším dôležitým komponentom je kontrola údajov a manažment celého honeynetu. Teda základné prvky honeynetu sa dajú zhrnúť do týchto troch častí:

- **Zber údajov**
- **Kontrola údajov a celého systému**
- **Prezentačný modul**

Posledné dva body sa dajú spoločne nazvať vzdialený manažment honeynetu alebo taktiež správa honeynetu. Konkrétna architektúra tohto honeynetu je znázornená na obrázku nižšie. (Obr. 12).



Obr. 12 Návrh honeynet systému

6.2.2 Inštalácia shell senzora a honeypotu

Pre možnosť nasadenia HonSSH sme vytvorili dva neprivilegované kontajnery. Oba s operačným systémom Ubuntu 14.04. V kontajnery, na ktorom bude bežať honeypot sme nainštalovali OpenSSH server a spustili ho na porte 22. Ako autentifikáciu sme zvolili klasickú možnosť, teda autentifikáciu pomocou hesla. Taktiež sme vytvorili niekoľko užívateľov a nastavili im slabé heslá. Takto budú mať útočníci možnosť ľahšie preniknúť do systému a my môžeme zozbierať viac údajov. Autentifikáciu privátnym kľúčom sme museli zamietnuť, pretože HonSSH ju nepodporuje.

Kontajner s HonSSH taktiež obsahuje inštaláciu OpenSSH servera, avšak táto služba nie je v kontajneri spustená. Jej inštalácia je potrebná pre HonSSH, ktorý si pomocou utilít tohto softvéru generuje privátne a verejné kľúče. Taktiež sme nainštalovali databázu MySQL a sprístupnili len pre localhost. Do tejto databázy si HonSSH bude lokálne ukladať údaje. Pre túto databázu sme zapli binárne logy, ktoré MySQL natívne podporuje. Do týchto logov sa ukladajú operácie, ktoré vykonávajú zmeny v databáze (INSERT, UPDATE, DELETE, CREATE, a podobne).

Ďalším krokom bol návrh počítačovej siete. HonSSH kontajner musíme pripojiť k internetu a prideliť mu verejnú IP adresu. Potom musíme prepojiť HonSSH kontajner s honeypot kontajnerom privátnou sieťou. Na hostiteľskom operačnom systéme sme vytvorili pre tento účel dva sieťové mosty. Pre kontajnery sme povolili vytvorenie troch virtuálnych sieťových rozhraní. Keďže používame neprivilegované kontajnery, musíme tak urobiť, lebo inak by sme nemali práva na takéto akcie. Zo všetkých možných druhov sme si vybrali veth rozhranie pre všetky tri sieťové rozhrania. Veth rozhrania fungujú na druhej sieťovej vrstve a teda dovoľujú broadcast. Prvotný návrh bol použiť macvlan rozhrania medzi kontajnermi, avšak tieto sú položené až na tretej sieťovej vrstve a nefungujú pri neprivilegovaných kontajneroch. Macvlan rozhrania sú však rýchlejšie ako veth rozhrania. Jedno z týchto rozhraní sme pridelili honeypot kontajneru ako eth0 rozhranie a druhý koniec tohto rozhrania sme pridelili jednému zo sieťových mostov. Ďalšie dva veth rozhrania sme pridelili HonSSH kontajneru ako rozhranie eth0 a eth1. Druhý koniec rozhrania eth1 sme pridelili do toho istého sieťového mosta ako predošlé z honeypotu a eth0 do druhého sieťového mosta. Do tohto mosta sme pripojili aj rozhranie eth0 z hostiteľského operačného systému, a teda sme umožnili pripojenie sa HonSSH kontajnera k internetu. Pre HonSSH kontajner sme nastavili staticky sieť pre obe rozhrania a pridelili sme mu verejnú IP adresu. Taktiež sme povolili smerovanie

a nastavili NAT tabuľku. Pre honeypot kontajner sme nastavili staticky sieť tak, aby bránou bol práve HonSSH kontajner, a teda aby sa pripájal k internetu pomocou neho. Ešte sme museli pomocou iptables nastaviť pravidlá pre sieťový most spájajúci kontajnery tak, ako sú štandardne nastavené pre sieťový most vytvorený po inštalácii LXC 1.0. Jednoduchšou možnosťou je použiť pre toto spojenie práve tento štandardný sieťový most. Ďalším krokom bola inštalácia a konfigurácia HonSSH senzora a honeypot bol hotový.

6.3 Zber údajov

Dôležitým prvkom honeynetu je práve zber údajov. Jeho súčasťou je aj samotný HonSSH, pretože zabezpečuje zber údajov z SSH komunikácie. Tieto údaje si HonSSH ukladá lokálne v kontajnery, v ktorom beží. Otázkou ale je, ako dostať tieto údaje na vyššiu úroveň, čiže uložiť ich na hosťujúcom stroji. Navyše chceme zachovať izoláciu virtuálnych strojov od hosťujúceho systému. HonSSH nám zabezpečí dva typy ukladania dát. Ukladá ich do MySQL databázy a taktiež do súborov. Pre nás sú výhodnejšie dáta z databázy, pretože sa s nimi lepšie pracuje, na druhej strane súborové dáta sa získavajú ľahšie pri použití virtualizácie. V tejto podkapitole si rozoberieme možnosti presunu údajov z virtuálnej vrstvi na vrstvu operačného systému.

6.3.1 Databázové údaje

HonSSH ukladá zozbierané dáta do MySQL databázy. Ukladá ich do niekoľkých tabuliek. Všetko sa však deje lokálne na stroji, na ktorom beží. Ak teda chceme pristupovať k týmto dátam, musíme pristupovať priamo k HonSSH stroju. Bez použitia virtualizácie bolo možné pristúpiť k údajom iba po počítačovej sieti alebo priamo na stroji. Pri priamom prístupe k stroju stráca administrátor možnosť vzdialeného prístupu k dátam. Ak pristupujeme k zozbieraným dátam po počítačovej sieti, vytvárame nadbytočnú sieťovú komunikáciu s HonSSH strojom. Tá sa síce nezahrnie do logov, ktoré vytvára, ale môže znamenať riziko odhalenia, alebo dokonca napadnutia HonSSH stroja, pretože musíme mať spustenú ešte aspoň jednu službu, ktorá je prístupná z internetu a ktorá nám umožní prístup k údajom.

Naším riešením je využitie virtualizácie na rovni operačného systému. Na HonSSH stroji má MySQL databáza povolené binárne logy. Do nich sa ukladajú operácie, ktoré

akokoľvek menia databázu. Keďže používame virtualizáciu, máme priamy prístup k súborovému systému kontajnerov. Teda získanie týchto logov znamená iteratívne presúvať zmenené súbory logov do adresára na hosťujúcom operačnom systéme. Tieto logy prevedieme pomocou utility **mysqlbinlog**, ktorá je obsiahnutá v inštalácii MySQL, do textového formátu. Taktiež pri konverzii vyberáme iba záznamy od istého dátumu a času, teda od poslednej konverzie. Takto môžeme volať konverziu dát iteratívne. Z textového formátu vyberieme iba tie riadky, ktoré sa týkajú HonSSH údajov, a ktoré obsahujú operácie „INSERT“ a aj niektoré riadky obsahujúce „UPDATE“. Tieto údaje potom vložíme do databázy, ktorá sa už nachádza na hosťujúcom systéme.

Výhody takéhoto prístupu sú zjavné. Získane dáta sa nenachádzajú iba na lokálnom honeypote, ale **aj** v **databáze honeynetu**. Takýto spôsob zberu dát je tiež bezpečný a hlavne **neodhaliteľný.** Administrátor takto získava možnosť pristupovať k údajom odkiaľkoľvek a nemusí sa vôbec pripájať k samotným honeypotom. Na druhej strane existuje okrem všeobecných nevýhod virtualizácie na úrovni operačného systému malá nevýhoda. HonSSH senzor ukladá údaje v reálnom čase, avšak my ich získavame v cykloch. Tento fakt sa dá čiastočne eliminovať zmenšením intervalu, v ktorom sú údaje iteratívne presúvané, alebo pri potrebe aktuálnych dát môžeme spustiť tento proces manuálne.

6.3.2 Súborové údaje

Druhá možnosť ako HonSSH ukladá údaje je do súborov. Pretože používame virtualizáciu, bez problémov vieme tieto súbory skopírovať na hosťujúci systém. Údaje zo súboru však musíme upravovať do nejakej rozumnej formy. Na rozdiel od databázových údajov sa v nich ťažšie vyhľadáva. V našej práci sme sa na tento typ údajov nezamerali. Nie je však problém v budúcnosti dopracovať podporu pre prácu s nimi. Výhody oproti klasickému prístupu sú rovnaké ako pri databázových údajoch. Avšak pri iteratívnom prístupe ku kopírovaniu týchto súborov je potrebné vo väčšine prípadov kopírovať všetky súbory, teda presúvať vždy všetky údaje.

6.3.3 Možnosti ďalších senzorov s využitím virtualizácie

Aj keď sme sa v tejto práci zamerali práve na shell senzor v honeynete, využitie virtualizácie na úrovni operačného systému nám ponúka ďalšie možnosti ako zberať dáta z honeynetu bez toho, aby to útočník zistil. Niekoľko z nich si teraz rozoberieme.

Základom každého honeynetu je senzor zberajúci údaje zo sieťovej komunikácie (**sieťový senzor**). Vďaka architektúre našej počítačovej siete máme prístup k všetkej komunikácii pomocou sieťových mostov. Nie je teda problém zberať a ukladať hlavičky všetkých prichádzajúcich a odchádzajúcich paketov z honeynetu. Jednoducho len určíme, ktoré sieťové mosty sú pre nás dôležité a zberáme dáta z komunikácie na nich. Je dôležité, aby cez tieto spojenia prechádzala iba komunikácia z honeynetu a nie z hosťujúceho systému. Ak sa tam nejaká vyskytne, musíme ju potom dodatočne odfiltrovať, aby sme neznehodnotili údaje. Príkladom takéhoto zberu údajov môže byť program tcpdump [29] spustený na niektorom zo sieťových mostov, ktorý bude zberať hlavičky paketov.

Ďalším dôležitým senzorom je **senzor súborového systému**. Môžeme zberať údaje o zmenách v súborovom systéme honeypotov. Prístup k týmto údajom nám zabezpečí virtualizácia. V ktoromkoľvek okamihu môžeme ukladať útočníkom upravené súbory tak, že o tom útočník nemusí vedieť. Taktiež môžeme vykonávať zmeny v súborovom systéme honeypotov, no takáto činnosť sa už môže útočníkovi javiť ako podozrivá.

Ďalej máme možnosť zberať údaje z bežiacich procesov, z pamäte virtuálnych strojov a podobne. Dokonca môžeme v nejakom dôležitom okamihu stroj zmraziť (tzv. „Freeze stav“) a vytvoriť kópiu tohto stroja v stave, v akom sme ho pozastavili a znova ho spustiť. Takto vytvorená kópia stroja, inak nazývaná aj ako „**SNAPSHOT**“ a môže obsahovať dôležité informácie o stroji v stave, v ktorom sme ho pozastavili.

6.4 Kontrola údajov

Ďalším dôležitým prvkom honeynetu je kontrola údajov. Útočníka chceme udržať v honeynete a kontrolovať jeho činnosť, no on to nesmie zistiť. V našom návrhu sa tento komponent nachádza na viacerých miestach a úrovniach honeynetu. Vďaka virtualizácii máme možnosť nasadiť oveľa viac kontrolných prvkov, ako je tomu pri klasických honeynetoch a možnosť ich odhalenia sa znižuje. Základnou kontrolou je riadenie virtuálnych kontajnerov. Akonáhle sa útočník snaží o nekalú činnosť, ktorú mu už nechceme dovoliť, jednoducho ho odrežeme od honeynetu. Možností je veľa, no

najjednoduchšou je zastaviť virtuálny stroj. Takto však zastavíme všetky senzory, ktoré boli na ňom závislé. Ďalšou možnosťou je útočníka odstaviť pomocou firewall pravidiel. Takto ostane virtuálny stroj v prevádzke, no daný útočník už k nemu nebude mať prístup. V oboch spomenutých prípadoch však útočník jednoducho zistí, že niečo nie je v poriadku a náš systém sa mu môže zdať podozrivý. Takáto kontrola môže byť manuálna, keď administrátor spozoruje podozrivú činnosť zakročí, alebo automatická, na základe nejakých pravidiel. V našom honeynete používame práve tú prvú možnosť kontroly, teda administrátor má možnosť stroj pozastaviť. Druhú možnosť nijako nevylučujeme, ale administrátor si musí pravidlá pre firewall zadávať sám.

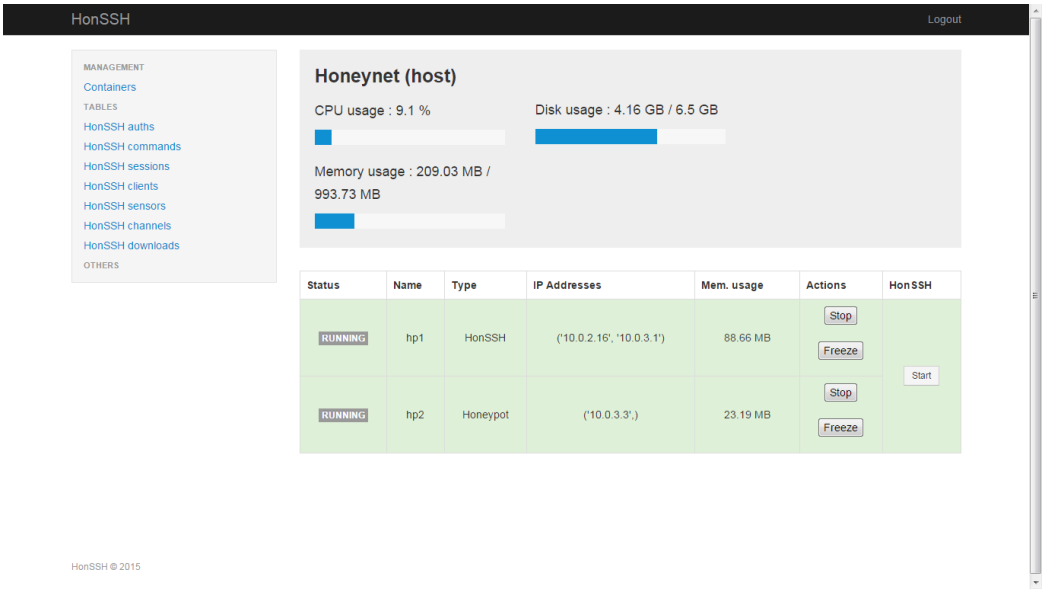
Lepším riešením je útočníka iba obmedziť v činnosti. Môžeme ho napríklad obmedziť počtom spojení, ktoré môže pre danú službu vytvoriť. Taktiež v počte paketov, ktoré môže útočník preniesť za nejaký čas alebo mu zakázať určité služby. Takáto kontrola je pre útočníka menej podozrivá. Riešením pre takýto typ kontroly je nastavenie firewall pravidiel. Tieto sa dajú v čase meniť, teda administrátor ich môže prispôsobovať aktuálnym požiadavkám. Celá kontrola potom však už bude automatická na základe daných pravidiel. V našom návrhu používame firewall pravidlá, ktoré sú nasadené priamo na HonSSH stroji. Celá sieťová komunikácia s honeypotom je teda kontrolovaná týmito pravidlami.

Kontrola údajov nie je zameraná iba na sieťovú komunikáciu. Dôležité sú aj iné možnosti ako napríklad obmedzenia pri využívaní systémových zdrojov. Útočníka môžeme obmedziť v počte procesov, v množstve pamäte, ktorú smie použiť, v kapacite diskov a podobne. Takto máme možnosť kontrolovať činnosť útočníkov v honeynete. Pri použití LXC virtualizácie, ale aj pri iných virtualizáciách, má administrátor možnosť meniť rôzne obmedzenia pre virtuálne stroje. V našom prípade sa tieto nastavenia vykonávajú pomocou nástroja **cgroups.**

Dôležité je tiež, aby bola **kontrola údajov** aspoň **dvojúrovňová**. Náš návrh spĺňa túto podmienku. Kontrola prebieha priamo na virtuálnom stroji, kde je zabezpečená pomocou firewall pravidiel na HonSSH stroji. Druhá úroveň je na fyzickom stroji, kde máme možnosť kontrolovať beh virtuálnych kontajnerov a taktiež máme prístup k sieťovej kontrole, ktorú môžeme vykonávať vďaka využitiu sieťových mostov, cez ktoré celá komunikácia v honeynete prebieha.

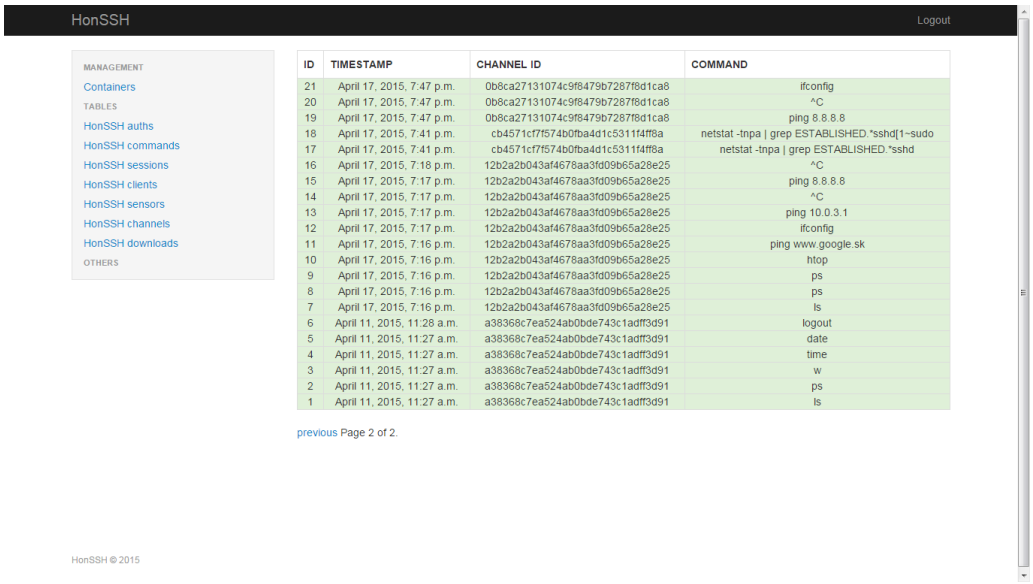
6.5 Manažment a správa honeynetu

K správe honeynetu pristupujeme dvojakým spôsobom. Administrátor môže riadiť honeynet pomocou terminálu, ale keďže táto možnosť nie každému vyhovuje, vytvorili sme webové rozhranie, pomocou ktorého môže vykonávať základne operácie v honeynete aj človek, čo začína s honeynetmi. Je jasné, že s využitím príkazového riadku má administrátor viac možností pri správe honeynetu, ale postupne pridávame viac riadiacich prvkov aj do webového rozhrania, čím uľahčujeme prácu s honeynetom. Taktiež sme vytvorili RESTful webové rozhranie, ktoré umožňuje integráciu vzdialenej správy honeynetu do iných honeynet nástrojov.



Obr. 13 Web rozhranie

Ďalšou časťou webového rozhrania je **prezentačný modul**. Zozbierané údaje z HonSSH honeypotu znázorňujeme v jednoduchých tabuľkách. Takto má možnosť administrátor získať prehľad o aktivitách na shell senzore. Napríklad môže sledovať príkazy, ktoré útočníci zadali, tak ako je to znázornené na obrázku nižšie (Obr. 14). Tieto údaje je však potrebné ďalej analyzovať pomocou iných nástrojov, ktoré nám poskytnú komplexnejší pohľad na postupy útočníkov.



ID	TIMESTAMP	CHANNEL ID	COMMAND
21	April 17, 2015, 7:47 p.m.	0b8ca27131074c9f8479b7287f8d1ca8	ifconfig
20	April 17, 2015, 7:47 p.m.	0b8ca27131074c9f8479b7287f8d1ca8	^C
19	April 17, 2015, 7:47 p.m.	0b8ca27131074c9f8479b7287f8d1ca8	ping 8.8.8.8
18	April 17, 2015, 7:41 p.m.	cb4571cf7f574b0fba4d1c5311f4ff8a	netstat -tnpa | grep ESTABLISHED.*sshd[1~sudo
17	April 17, 2015, 7:41 p.m.	cb4571cf7f574b0fba4d1c5311f4ff8a	netstat -tnpa | grep ESTABLISHED.*sshd
16	April 17, 2015, 7:18 p.m.	12b2a2b043af4678aa3fd09b65a28e25	^C
15	April 17, 2015, 7:17 p.m.	12b2a2b043af4678aa3fd09b65a28e25	ping 8.8.8.8
14	April 17, 2015, 7:17 p.m.	12b2a2b043af4678aa3fd09b65a28e25	^C
13	April 17, 2015, 7:17 p.m.	12b2a2b043af4678aa3fd09b65a28e25	ping 10.0.3.1
12	April 17, 2015, 7:17 p.m.	12b2a2b043af4678aa3fd09b65a28e25	ifconfig
11	April 17, 2015, 7:16 p.m.	12b2a2b043af4678aa3fd09b65a28e25	ping www.google.sk
10	April 17, 2015, 7:16 p.m.	12b2a2b043af4678aa3fd09b65a28e25	htop
9	April 17, 2015, 7:16 p.m.	12b2a2b043af4678aa3fd09b65a28e25	ps
8	April 17, 2015, 7:16 p.m.	12b2a2b043af4678aa3fd09b65a28e25	ps
7	April 17, 2015, 7:16 p.m.	12b2a2b043af4678aa3fd09b65a28e25	ls
6	April 11, 2015, 11:28 a.m.	a38368c7ea524ab0bde743c1adff3d91	logout
5	April 11, 2015, 11:27 a.m.	a38368c7ea524ab0bde743c1adff3d91	date
4	April 11, 2015, 11:27 a.m.	a38368c7ea524ab0bde743c1adff3d91	time
3	April 11, 2015, 11:27 a.m.	a38368c7ea524ab0bde743c1adff3d91	w
2	April 11, 2015, 11:27 a.m.	a38368c7ea524ab0bde743c1adff3d91	ps
1	April 11, 2015, 11:27 a.m.	a38368c7ea524ab0bde743c1adff3d91	ls

Obr. 14 Web prezentácia údajov

Záver

Útoky na počítačové siete čím ďalej tým viac pribúdajú, stávajú sa čoraz častejšie a sú agresívnejšie. Proti týmto útokom sa musíme chrániť. Bez informácií o útočníkoch a ich nástrojoch by sme však mali len malú šancu. V tejto práci sme sa zamerali na zber takýchto údajov o útočníkoch pomocou honeypotov a honeynetov. Jedným zo spôsobov, ako môže útočník ovládnuť cudzí stroj je pomocou SSH protokolu, ktorý sa využíva pre vzdialenú správu operačných systémov. Zamerali sme sa na použitie shell senzora ako nástroja na zber údajov z SSH komunikácie v honeynete.

Prvým cieľom tejto práce bolo analyzovať a porovnať vysoko-interaktívne honeypoty z pohľadu možností odchytávania údajov. V prvej kapitole sme sa zamerali na vysvetlenie pojmu honeypot. Taktiež sme sa zaoberali využitím honeypotov a ich základným rozdelením. V druhej kapitole sme sa zaoberali honeynetmi a ich architektúrov. Spomíname ich kľúčové vlastnosti, ktoré sú potrebné pre implementáciu honeynetu.

V kapitole o dostupných riešeniach sme rozobrali možné implementácie SSH honeypotov. Kapitola je členená do troch podkapitol, z ktorých každá reprezentuje určitý druh honeypotov. Toto rozdelenie sme urobili podľa miery interakcie, ktorú ponúkajú jednotlivé honeypoty. Rozhodli sme sa pre takéto rozdelenie, pretože miera interakcie najlepšie popisuje možnosti odchytávania údajov. Nízko-interaktívne honeypoty umožňujú zber najmenej užitočných informácií, naopak vysoko-interaktívne umožňujú zber najcennejších a najužitočnejších informácií o útočníkoch a ich praktikách.

Druhým cieľom bolo analyzovať využitie virtualizácie pri odchytávaní údajov. V piatej kapitole sa zaoberáme pojmom virtualizácie. Opísali sme štyri druhy virtualizácie, ktoré môžeme využiť pri výstavbe honeynetu. Celá práca sa zameriava na využitie virtualizácie na úrovni operačného systému ako virtualizácie k vytvoreniu honeynetu. V šiestej kapitole sa zaoberáme honeynetom založenom na virtualizácii na úrovni operačného systému. Využívame virtualizáciu LXC a honeypot HonSSH ako shell senzor pre vybudovanie virtuálneho honeynetu.

Posledným cieľom bolo navrhnúť a implementovať shell senzor vo virtuálnom honeynete. Ako spomíname vyššie, v šiestej kapitole sa zaoberáme návrhom tohto honeynetu. Rozoberáme základné body dôležité pri jeho výstavbe. Týmito bodmi sú zber údajov a kontrola údajov. Zber údajov je zabezpečený honeypotom HonSSH. Taktiež popisujeme iné možnosti zberu údajov pri využití virtualizácie. Kontrolu údajov

zabezpečujeme pomocou kontroly virtuálnych kontajnerov a počítačovej sieti, ktorú používajú na komunikáciu.

Pre uľahčenie manažmentu celého systému sme vytvorili webové rozhranie, ktoré je schopné aj jednoduchej prezentácie zozbieraných údajov. V ďalšej práci sa chceme zamerať na prenesenie čo najväčšej časti kontroly honeynetu do webového rozhrania, pretože takto dokážeme sprístupniť honeynety aj ľuďom, ktorí s honeynetmi začínajú.

Zoznam použitej literatúry

[1] A Virtual Honeypot Framework, https://www.usenix.org/legacy/event/sec04/tech/full_papers/provos/provos_html/honeyd.html, [Naposledy navštívené: 16.5.2015]

[2] ABBASI, F. H., HARRIS, R. J. Experiences with a Generation III virtual Honeynet. In: Telecommunication Networks and Applications Conference (ATNAC), 2009 Australasian. IEEE, 2009. p. 1-6.

[3] Apache MINA, https://mina.apache.org, [Naposledy navštívené: 19.4.2015]

[4] Artillery, https://www.trustedsec.com/downloads/artillery/, [Naposledy navštívené: 19.4.2015]

[5] BANSTOLA, B. Analysing the Scalability of Virtualized Environment. 2012.

[6] CAMPBELL, S., JERONIMO, M. An Introduction to Virtualization. In Applied Virtualization. Intel, 2006.

[7] Copssh, https://www.itefix.net/copssh, [Naposledy navštívené: 19.4.2015]

[8] Debian, https://www.debian.org, [Naposledy navštívené: 19.4.2015]

[9] Dropbear, https://matt.ucc.asn.au/dropbear/dropbear.html, [Naposledy navštívené: 19.4.2015]

[10] European Network and Information Security Agency (ENISA). Proactive Detection of Security Incidents II: Honeypots. https://www.enisa.europa.eu/activities/cert/support/proactive-detection/proactive-detection-of-security-incidents-II-honeypots/, 2012. [Naposledy navštívené: 19.4.2015]

[11] H3C, SSH Technology White Paper, http://www.h3c.com/portal/Products__Solutions/Products/Switches/Campus_Network_Switches/H3C_S5500-SI_Series_Switches/White_Paper/200906/689006_57_0.htm, [Naposledy navštívené: 16.5.2015]

[12] Honeyd, http://www.honeyd.org/, [Naposledy navštívené: 19.4.2015]

[13] HonSSH, https://github.com/tnich/honssh, [Naposledy navštívené: 19.4.2015]

[14] JOSHI, R. C., SARDANA, A. Honeypots: A New Paradigm to Information Security. Science Publishers, 2011.

[15] KFsensor, http://www.keyfocus.net/kfsensor/, [Naposledy navštívené: 19.4.2015]

[16] Kippo, https://github.com/desaster/kippo, [Naposledy navštívené: 19.4.2015]

[17] KitPloit: HonSSH, http://www.kitploit.com/2014/04/honssh-log-all-ssh-communications.html, [Naposledy navštívené: 16.5.2015]

[18] Know your enemy: Defining Virtual Honeynets, http://old.honeynet.org/papers/virtual/, [Naposledy navštívené: 19.4.2015]

[19] Know your enemy: Honeynets. http://old.honeynet.org/papers/honeynet/, [Naposledy navštívené: 19.4.2015]

[20] Kojoney, http://kojoney.sourceforge.net, [Naposledy navštívené: 19.4.2015]

[21] Linux containers, https://linuxcontainers.org, [Naposledy navštívené: 19.4.2015]

[22] OpenSSH, http://www.openssh.com, [Naposledy navštívené: 19.4.2015]

[23] PROVOS, N. HOLZ, T. Virtual honeypots: from botnet tracking to intrusion detection. Pearson Education, 2007.

[24] Qebek, http://honeynet.org/sites/default/files/files/KYT-Qebek-final_v1.pdf, [Naposledy navštívené: 19.4.2015]

[25] Sebek, http://old.honeynet.org/papers/sebek.pdf, [Naposledy navštívené: 19.4.2015]

[26] Secure Honey, http://securehoney.net, [Naposledy navštívené: 16.5.2015]

[27] SPITZNER, L. Honeypots: tracking hackers. Addison-Wesley, 2002.

[28] SSHpot, http://www.threadstates.com/projects/sshpot.html, [Naposledy navštívené: 16.5.2015]

[29] Tcpdump, http://www.tcpdump.org, [Naposledy navštívené: 19.4.2015]

[30] The Secure Shell (SSH) Protocol Architecture, https://tools.ietf.org/html/rfc4251, [Naposledy navštívené: 19.4.2015]

[31] Ubuntu, http://www.ubuntu.com/, [Naposledy navštívené: 19.4.2015]

[32] XAVIER, M. G., et al. Performance evaluation of container-based virtualization for high performance computing environments. In: Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on. IEEE, 2013. p. 233-240.

[33] Xebek, http://archive.hack.lu/2006/Xebek-HackDotLu06.pdf, [Naposledy navštívené: 19.4.2015]

Príloha A

Obsah CD

- Zdrojový kód webového rozhrania v podobe Django projektu (./stranka)
- Skripty pre zber údajov z kontajnerov (./data_collect)
- Upravené zdrojové kódy HonSSH (./container_honssh)
- Požiadavky pre správny beh aplikácií (./requirements.txt)
- Elektronická podoba tejto práce vo formáte PDF (./bakalarka.pdf)