VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

AUTOMATIZACE NASAZENÍ A SBĚRU DAT Z HONEYPOTŮ

AUTOMATIC HONEYPOTS DEPLOYMENT AND DATA GATHERING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ ĎURIŠ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. LUKÁŠ ZOBAL**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Ďuriš Tomáš**
Program: Informační technologie
Název: **Automatizace nasazení a sběru dat z honeypotů**
Automatic Honeypots Deployment and Data Gathering
Kategorie: Bezpečnost

Zadání:

1. Seznamte se s problematikou honeypotů.
2. Nastudujte technologie umožňující snadnou a škálovatelnou správu výpočetních prostředků.
3. Po dohodě s vedoucím zvolte sadu honeypotů a navrhněte systém pro jejich automatizované nasazení, monitoring a sběr dat. Zaměřte se na rozšiřitelnost tohoto řešení i pro další honeypoty v budoucnu.
4. Návrh implementujte a po dohodě s vedoucím nasaďte na vybrané servery v internetu.
5. Vytvořené řešení důkladně otestujte sadou testů i v reálném provozu.
6. Zhodnoťte svoji práci a uveďte možný další vývoj.

Literatura:

- M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil and J. Schönfelder, "A survey on honeypot software and data analysis," arXiv preprint arXiv:1608.06249, 2016.
- C. Hecker and B. Hay, "Automated Honeynet Deployment for Dynamic Network Environment," 2013 46th Hawaii International Conference on System Sciences, Wailea, Maui, HI, 2013, pp. 4880-4889.
- I. Koniaris, G. Papadimitriou, P. Nicopolitidis and M. Obaidat, "Honeypots deployment for the analysis and visualization of malware activity and malicious connections," 2014 IEEE International Conference on Communications (ICC), Sydney, NSW, 2014, pp. 1819-1824.
- Getting Started - Ansible Documentation. Ansible Documentation [online]. Dostupné z: https://docs.ansible.com/ansible/latest/user_guide/intro_getting_started.html
- How to get started with Puppet: A beginner's guide | Puppet. Powerful infrastructure automation and delivery [online]. Dostupné z: https://puppet.com/blog/how-get-started-puppet-beginners-guide/

Pro udělení zápočtu za první semestr je požadováno:

- Splnění prvních tří bodů a rozpracování bodu 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Zobal Lukáš, Ing.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 26. října 2020

Abstrakt

Táto práca sa zoberá automatizáciou nasadenia honeypotov, zberom dát z honeypotov a nasadením monitorovacieho systému s upozorňovaním. Cieľom bolo naštudovanie problematiky honeypotov, vybratie vhodných nástrojov pre ich nasadenie, údržbu a zber poskytnutých dát spoločne s vytvorením automatického systému pre nasadenie rôznych druhov honeypotov. Prvá časť práce je venovaná teórii honeypotov, ich rozdeleniu a typom. Neskôr je v práci spomenuté porovnanie jednotlivých konfiguračných nástrojov. Praktická časť sa venuje použitiu vybraného konfiguračného nástroja Ansible v spolupráci s existujúcimi voľne dostupnými aplikáciami k zostaveniu plne automatizovaného systému na nasadenie a monitorovanie honeypotov, zber poskytnutých dát a ich vizualizáciu.

Abstract

This work deals with honeypots deployment automation, data collection from honeypots and the deployment of a monitoring system with alerts. The aim was to study the issue of honeypots, choose tools for their deployment, maintenance and collection of provided data together with creation of automatic deployment system for multiple types of honeypots. The first part of the work is devoted to the theory of honeypots, their distribution and type. Furthermore, the work mentions the comparison of individual configuration tools. The practical part is devoted to the use of a selected configuration tool Ansible in cooperation with existing open-source applications to build a fully automated system for the deployment and monitoring of honeypots, collection of provided data and their visualization.

Kľúčové slová

honeypot, automatizácia, Ansible, monitorovanie, upozorňovanie, nástroje na správu konfigurácie, nástroje na monitorovanie

Keywords

honeypot, automatization, Ansible, monitoring, alerting, configuration managment tools, monitoring tools

Citácia

ĎURIŠ, Tomáš. *Automatizace nasazení a sběru dat z honeypotů.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Lukáš Zobal

Automatizace nasazení a sběru dat z honeypotů

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Lukáša Zobala s použitím vedeckej literatúry a odborných textov, ktoré sú všetky poriadne citované a uvedené v zozname použitej literatúry.

.....
Tomáš Ďuriš
9. mája 2021

Poďakovanie

Ďakujem vedúcemu bakalárskej práce Ing. Lukášovi Zobalovi a firme Avast za odbornú pomoc a množstvo cenných rád pri spracovaní mojej bakalárskej práce.

Obsah

Kapitola 1

Úvod

Kybernetická bezpečnosť patrí v poslednom období medzi najdiskutovanejšie bezpečnostné témy. Aktuálna doba je totiž charakteristická tým, že väčšina osobných či pracovných aktivít sa presunula do online prostredia. Internet sa stal globálnym nástrojom slobodného pohybu pri získavaní informácií. Priamo úmerne s nárastom jeho používania narastajú aj kybernetické hrozby a potreba exaktne riešiť kybernetickú bezpečnosť. Úlohou kybernetickej bezpečnosti je efektívne a rýchlo identifikovať zraniteľnosti systémov a aplikácií, alebo riziká a hrozby v sieti. Pri identifikácii zraniteľných miest sa musíme zaoberať zamedzením napadnutia a zneužitia systému. Je potrebná rýchla a účinná reakcia na čoraz sofistikovanejšie útoky. V úvodných fázach boja proti počítačovej kriminalite je dôležitá detekcia útokov a prevencia pred ďalšími útokmi. Z výskumu už prebehnutých útokov sa odhaľujú zraniteľné miesta a vyvíjajú účinné stratégie ako útokom zabrániť. Bezpečnostné riziko markantne narastá s každým ďalším pripojeným zariadením na internet.

Dôležitosť informačnej bezpečnosť zohráva rolu aj v každodennom živote. Rovnako ako si zabezpečíte svoj dom pred vykradnutím pri odchode na dovolenku, vypnete elektriku či odstavíte prívod vody, je na mieste zabezpečiť si aj súkromie a bezpečnosť na internete. Jedným zo spôsobov ako sa ochrániť pred rizikom je totiž predísť mu. Útočníci disponujú stále novšími a novšími zbraňami a preto je nutné s nimi udržiavať krok.

Jednou z možností ako získať informácie o aktuálnych trendoch vo svete počítačových útokov je práve využitie honeypotu. Honeypot je systém, ktorý má za úlohu prilákať rôzne typy útokov a získať čo najviac informácii o útočníkovi a prostriedkoch, ktoré použil. Úlohou tejto bakalárskej práce je navrhnúť a implementovať program, ktorý by dokázal nasadiť väčšie množstvo honeypotov rôznych druhov, ich monitorovanie a spracovanie získaných informácií. Výsledný program by mal byť ľahko udržiavateľný a rozširovateľný.

Práca je rozdelená na niekoľko častí, ktoré popisujú či už definíciu a výhody využitých prostriedkov alebo návrh a realizáciu samotného programu. Kapitola 2 sa venuje bližšiemu popisu honeypotu, stručnej histórii týchto systémov, ich klasifikácii a využitiu získaných dát. V kapitole 3 sú bližšie popísané nástroje, pomocou ktorých vieme hromadne nasadiť väčšie množstvo honeypotov a nástroje, ktoré môžeme použiť na monitorovanie či už samotných honeypotov alebo serverov, na ktorých bežia. V nasledujúcej kapitole 4 je popísaný návrh mnou zvoleného riešenia nasadenia, návrh monitorovacieho systému a popis konfigurácie honeypotov. Kapitola 5 popisuje implementáciu Ansible modulov využitých k jednotnému nasadeniu množstva honeypotov a ich monitorovania spolu s popisom konfiguračných špecifík pre každý typ honeypotu. V záverečnej kapitole 6 je otestované výsledné riešenie, analýza získaných dát a zhrnutie dosiahnutých výsledkov.

Kapitola 2

Honeypot

Počítačová bezpečnosť je koncept, ktorý je možné rozdeliť na oblasť prevencie, detekcie a reakcie. Pod prevenciou rozumieme schopnosť predísť útoku, detekcia je proces identifikácie možného ohrozenia systému a reakcia popisuje schopnosť reagovať na identifikované riziko. Každý z týchto prvkov je pritom navrhnutý tak, aby dosahoval čo najlepšie výsledky vo svojej oblasti pôsobenia. Antivírusový systém, firewall, systém na detekciu prienikov (IDS), systém na prevenciu prienikov (IPS) alebo aj honeypoty sú predstaviteľmi základných elementov počítačovej bezpečnosti. Antivírusový systém operuje na úrovni súborov a snaží sa odlíšiť infikovaný súbor od neinfikovaného. Firewall slúži na monitorovanie a spravovanie sieťovej prevádzky na základe vopred definovaných pravidiel. Kým úlohou systému na detekciu prienikov je zanalyzovať celý sieťový paket, odhaliť škodlivé signatúry a upozorniť užívateľa. Systém na prevenciu prienikov dopĺňa tento koncept o možnosť automatického pridania pravidiel k odhaleniu ďalších škodlivých signatúr [15]. Jednou z najúčinnejších metód ako reagovať na indentifikované riziko je systém honeypot. V tabuľke 2.1 sú uvedené rozdiely medzi bezpečnostnými konceptami podľa oblasti ich pôsobenia.

	Prevencia	Detekcia	Reakcia
Antivírus	++	++	++
Firewall	+++	++	+
IDS	+	+++	+
IPS	++	+++	++
Honeypot	+	++	+++

Tabuľka 2.1: Rozdiely medzi bezpečnostnými konceptami na základe oblasti ich pôsobenia [15]

Honeypot je nová a neustále vyvíjajúca sa technológia. L. Spitzner ju definoval ako bezpečnostný prvok, ktorého hodnota spočíva v tom, že je detekovaný, napadnutý a skompromitovaný [21]. Jedinečnosť tejto technológie spočíva v tom, že nerieši len jeden špecifický bezpečnostný problém. Práve naopak, honeypot sa využíva v rôznych oblastiach, pričom záleží na tom, čo chceme touto technológiou dosiahnuť. Účelom honeypotu je formou návnady nalákať útočníka na nezabezpečený systém, ktorý sa navonok javí ako bežný. Ide o vernú simuláciu reálnej aplikácie alebo systému za účelom oklamania útočníka. Cieľom je identifikovať kroky útočníka a zaznamenať informácie o jeho činnosti. Nadobudnuté informácie

sú nápomocné pri zabezpečení skutočných systémov alebo aplikácií, ktoré sú honeypotom simulované, teda môžme hovoriť o honeypote ako o istej forme prevencie voči budúcim útokom.

Medzi najpodstatnejšie ciele systému honeypot patria [16]:

- Identifikovať doposiaľ neobjavené zraniteľnosti a riziká operačného systému alebo aplikácie.
- Poskytnúť skoré varovanie pred novým typom útokov.
- Dôkladne analyzovať útočníkovu aktivitu na napadnutom systéme.
- Poskytnúť informácie o aktuálnych útokoch, trendoch a poskytnúť údaje pre neskoršiu analýzu.
- Odvrátiť pozornosť útočníka od zariadení v systéme, ktoré majú väčšiu hodnotu (nemusí byť vždy spoľahlivé).

Honeypot je systém ponúkajúci na jednej strane veľké množstvo výhod a na strane druhej aj určité množstvo nevýhod. Medzi hlavné výhody radíme jednoduchosť, hodnotu získaných dát, zdroje a návratnosť vstupných investícii.

Jednoduchosť – Nasadenie honeypotu, jeho prevádzka a údržba je v porovnaní s ostatnými bezpečnostnými konceptami jednoduchšia. Je postačujúce nainštalovať niektorú voľne dostupnú implementáciu honeypotu, pripojiť ju do siete a počkať na aktivitu útočníka. Jednoduchosť systému so sebou prináša aj zníženie pravdepodobnosti vzniku chýb, ktoré by útočník mohol využiť vo svoj prospech.

Hodnota získaných dát – Honeypot sa inštaluje na zariadenia cez ktoré nepreteká legitímna sieťová prevádzka, každá aktivita v sieti je tak považovaná za aktivitu útočníka. Honeypoty na rozdiel od iných bezpečnostných konceptov generujú menšie množstvo dát, no všetky zachytené dáta sú nesmierne cenné.

Zdroje – Investícia do hardwaru, na ktorom honeypot pobeží je rádovo nižšia oproti ostatným bezpečnostným konceptom. Ako bolo vyššie spomenuté, cez tento typ systému preteká malé množstvo dát a teda sú menej náročné na vyčerpanie zdrojov

Návratnosť vstupných investícií – Výsledky dosiahnuté nasadením honeypotov sú viditeľné krátko po nasadení. Všetky útoky na nasadený systém sú okamžite zaznamenané a finančný benefit nasadenia honeypotov je lepšie viditeľný ako u ostatných bezpečnostných konceptov.

Pri využívaní honeypotov však treba mať na pamäti aj ich nevýhody. Medzi najväčšie nevýhody patrí ich neschopnosť zaznamenávať akcie nesmerované na nich alebo riziko ich odhalenia na základe charakteristického správania. Nevýhodou je aj možné riziko ovládnutia honeypotov a ich následného zneužitia na získanie kontroly nad celým systémom, ktoré stúpa so zvyšujúcou sa mierou interakcie s útočníkom.

2.1 História honeypotov

Táto podkapitola vychádza z poznatkov diela L. Spitznera [21]. Princíp honeypotov bol využívaný rôznymi vládnymi, vojenskými a obchodnými organizáciami pred rokom 1990, avšak za pilotné dielo sa považuje román *The Cuckoo's Egg* od Clifforda Stolla [22]. V tomto

diele bol spomenutý princíp využívania honeypotov pričom autor neopisuje použitie konkrétneho honeypotu, ale len princípy tejto technológie. Autor diela po zistení, že jeho systém bol napadnutý nepodnikol žiadne kroky, aby aby útoku zamedzil. Vystopovať útočníka sa mu podarilo sledovaním útoku a jeho následnou analýzou. Jedinečnosť práce spočíva v získavaní informácii bez toho, aby si to útočník uvedomil. Útočníkovi bolo poskytnuté veľké množstvo súborov s cieľom získať čas k jeho vystopovaniu. Išlo o materiály s lákavým obsahom alebo názvom, ktoré by pritiahli útočníkovú pozornosť. Boli poskytnuté materiály s rôznym obsahom. Motív útočníka bolo možné odhadnúť na základe obsahu odcudzeného materiálu.

Ďalšia významná práca v tejto oblasti už bola viac technicky zameraná na využitie princípov honeypotov v praxi. Autor práce Bill Cheswick prvý krát opisuje vytvorenie systému, ktorého hlavným cieľom je, aby bol napadnutý a následne skompromitovaný. Ide o dielo *An Evening with Berferd in which a cracker is Lured, Endured, and Studied*. V tomto diele ide okrem opisu návrhu systému honeypot aj opis konkrétneho útoku naň. Pričom nešlo o chytenie konkrétnej osoby, ale o zistenie aké hrozby existujú v jeho sieti. Išlo o zostrojenie tzv. *"väznice"*, v ktorej bol útočník uväznený a jeho aktivita bola zaznamenávaná.

Za prvé verejné riešenie honeypotu sa považuje *Deception Toolkit*[1] vytvorený Fredom Cohenom. Išlo o bezplatný produkt, ktorý si ľubovoľný užívateľ mohol vyskúšať a zvážiť jeho nasadenie. *Deception Toolkit* umožňoval simulovať viaceré zraniteľnosti systému UNIX a jeho cieľom bolo nie len získať informácie, ale aj zmiasť útočníka.

Prvý komerčne využívaný honeypot vznikol na prelome 20. storočia. Jednalo sa o produkt *CyberCop Sting* vyvynutý spoločnosťou Secure Networks Inc., ktorý sa funčkne odlišoval od *Deception Toolkit*. Zásadnou zmenou bol výber operačného systému, na ktorom honeypot bežal. Za druhú zmenu sa považuje schopnosť emulovať 3 rozdielné systémy zároveň.

Krátko po vydaní *CyberCop Sting* bol vyvinutý *BackOfficer Friendly*. Bol jednoducho použiteľný, voľne stiahnuteľný a schopný fungovať na ľubovoľnej verzii systému Windows. Pre veľa ľudí išlo o vstupnú bránu do sveta honeypotov.

V roku 1999 vzniklo združenie tridsiatich profesionálov z oblasti počítačovej bezpečnosti zameriavajúce sa na skúmanie kybernetických útokov a zdieľanie užitočných informácií s názvom *Honeynet Project* [20]. Pri skúmaní útokov bol využívaný pokročilý typ honeypotu s názvom *honeynet*. Výsledky skúmania boli zdokumentované a vydané v roku 2001 v dokumente pod názvom *Know Your Enemy*, ktorý mal za úlohu zvýšiť povedomie odbornej verejnosti o honeypotoch [17].

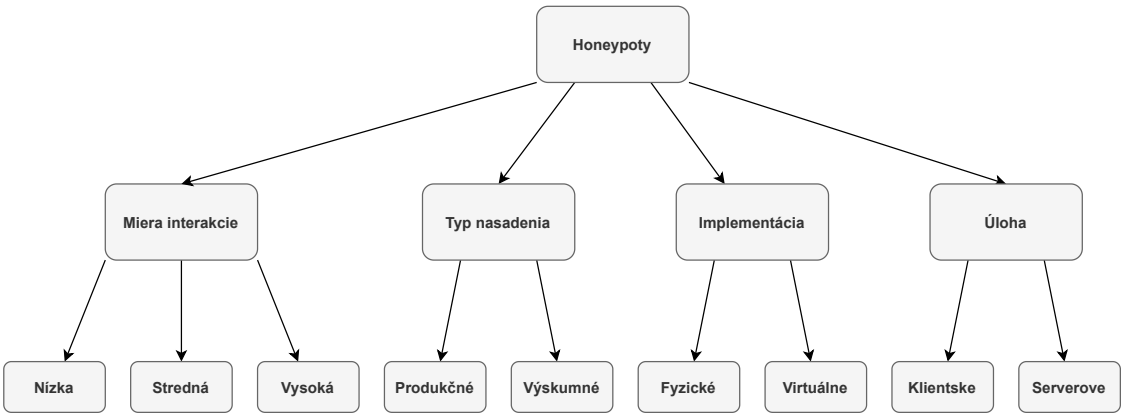
Začiatok 21. storočia charakterizuje výskyt a nárast škodlivého, efektívneho a rýchlo sa šíriaceho kódu, známeho pod názvom *"Sub7 Trojan"* [5]. Výzvou pre viaceré organizácie bolo získanie kópie tohoto škodlivého kódu a jeho následná analýza. Vzhľadom na extrémnu škodlivosť kódu bolo spätné získavanie dát z napadnutých systémov veľmi obtiažne. Takéto prípady poukázali na dôležitosť a prínos cieleného nasadzovania technológie honeypot.

V súčasnej dobe existuje veľké množstvo voľne dostupných honeypotov, ktoré sú pravidelne vyvíjané. Medzi najaktuálnejšie a najviac vyvíjané honeypoty patrí napríklad: Dionaea[2], Cowrie[3], Snare[4] a mnoho iných. Veľké množstvo bezpečnostných dier v sieti internet je odhalených a následne pokrytých práve vďaka týmto honeypotom. Spomínané honeypoty sa tešia veľkej komunite, vďaka ktorej sú pravidelne aktualizované a držia krok s útočníkom.

[1] http://all.net/contents/dtk.html
[2] https://github.com/DinoTools/dionaea
[3] https://github.com/cowrie/cowrie
[4] https://github.com/mushorg/snare

2.2 Klasifikácia honeypotov

Pre honeypoty platí, že nemajú konkrétnu špecifickú vlastnosť, podľa ktorej by mohli byť klasifikované do jednoznačných skupín. V odbornej literatúre preto nachádzame množstvo delení podľa rôznych kategórií. Jednotlivé kategórie honeypotov sa pritom navzájom prekrývajú a dopĺňajú [13]. V nasledujúcich podkapitolách uvedieme niekoľko delení honeypotov podľa špecifikovaných kritérií. Základné rozdelenie honeypotov predstavuje obrázok 2.1.



Obr. 2.1: Rozdelenie honeypotov

2.2.1 Delenie podľa miery interakcie

Honeypoty je možné kategorizovať použitím stupňa interakcie honeypotu s útočníkom, pričom interakcia stanovuje útočníkové možnosti komunikácie s honeypotom. Honeypoty podľa úrovne interakcie rozdeľujeme do troch skupín [21]:

- Nízka miera interakcie.
- Stredná miera interakcie.
- Vysoká miera interakcie.

Nízky stupeň interakcie

Pri nízkom stupni interakcie honeypot neobsahuje žiadny operačný systém, s ktorým by mohol útočník interagovať. Všetky procesy sú simulované a napodobňujú operačný systém, to znamená, že predstavujú nižšie bezpečnostné riziko v porovnaní s ostatnými honeypotmi so strednou alebo vyššou mierou interakcie. Útočník je schopný získať prístup len k obmedzenému množstvu informácii. Väčšinou sa jedná o emuláciu jednoduchých sieťových služieb ako napríklad HTTP alebo SSH. Výhodou využitia honeypotov s nízkou mierou interakcie je ich jednoduchosť nasadenia a udržovateľnosť. Medzi ich nevýhody je možné zaradiť množstvo získaných informácií o útočníkovi oproti honeypotom so strednou alebo vysokou mierou interakcie. Tieto honeypoty spravidla získajú len základné informácie o útočníkovi a spôsobe útoku (napr. jeho IP adresu, na ktorú službu zaútočil, prihlasovacie údaje, ktoré

použil a iné). Využitie týchto honeypotov sa javí ako ideálne, pokiaľ užívateľ nedisponuje dostatočným hardvérom k nasadeniu honeypotov s vyššou mierou interakcie alebo je jeho prioritou otestovať funkčnosť jednej konkrétnej služby. Do tejto kategórie radíme napríklad verejne dostupný honeypot Dionaea alebo HoneyPy[5].

Stredná miera interakcie

Honeypoty so strednou mierou interakcie sú pokročilejšie ako nízkoúrovňové honeypoty. Táto kategória tvorí akýsi kompromis medzi výhodami nízko a vysoko interaktívnych honeypotov. Tieto systémy sú síce obsiahlejšie, ale stále neposkytujú útočníkovi interakciu so skutočným operačným systémom. Rozdiel oproti honeypotom s nízkou mierou interakcie je v schopnosti emulovať hneď niekoľko služieb alebo procesov, prípadne im pridať na dôveryhodnosti a presvedčiť útočníka, že komunikuje s reálnym systémom. Príkladom takéhoto typu honeypotu môže byť napríklad honeypot Cowrie.

Vysoká miera interakcie

Implementácia honeypotov s vysokou mierou interakcie je náročný proces, ktorý vyžaduje znalosti nielen samotného použitého honeypotu ale aj fungovania siete alebo informačnej bezpečnosti. Jedná sa o modernú technológiu s komplexným dizajnom a vysokou mierou rizika. Tento typ honeypotu poskytuje útočníkovi schopnosť komunikovať so skutočným operačným systémom, v ktorom bežia skutočné aplikácie alebo služby a nejedná sa teda len o ich simuláciu. Za najväčšiu výhodu honeypotu s vysokou mierou interakcie sa považuje množstvo informácií o útočníkovi, ktoré je možné získať. Honeypot zaznamenáva všetku útočníkovu aktivitu a útočník je neustále monitorovaný. Tento typ honeypotov prináša so sebou riziko ovládnutia systému útočníkom, kedy by dostal plnú kontrolu nad systémom a mohol by ho využiť k ďalším útokom na systémy v sieti alebo získanie nesmierne cenných informácií. Prihliadnuc k všetkým uvedeným rizikám je potrebné tento systém dôsledne zabezpečiť a monitorovať všetku útočnikovu aktivitu. Z dôvodu minimalizovania rizika je vhodné honeypoty s vysokou mierou interakcie umiestniť za systémový firewall, ktorý sa využíva aby útočník mohol preniknúť do systému, ale nemohol ho zneužiť na spustenie útokov voči ostatným systémom. Hlavnou výhodou tejto technológie je množstvo a hodnota získaných informácií. Hodnota získaných informácií je mnohonásobne väčšia v porovnaní s honeypotmi s nižšou mierou interakcie. Tieto informácie je možné využiť k zabezpečeniu aktuálneho systému alebo k ďalšiemu skúmaniu a vyhodnocovaniu rizík odborníkmi v oblasti počítačovej bezpečnosti. Medzi typických predstaviteľov tejto technológie patrí RDPy honeypot[6].

[5] https://github.com/foospidy/HoneyPy
[6] https://github.com/citronneur/rdpy

2.2.2 Delenie podľa použitia a typu nasadenia

Honeypoty môžme na základe ich využitia a prostredia, do ktorého boli nasadené rozdeliť do dvoch základných skupín na [13]:

- Produkčné honeypoty.
- Výskumné honeypoty.

Delenie honeypotov na výskumné a produkčné sa nemusí striktne dodržiavať. Môžme sa stretnúť aj s honeypotom, ktorého kategória je definovaná spôsobom jeho použitia, pretože záleží na tom ako sa v praxi používa a nie ako bol zostrojený.

Produkčné honeypoty

Produkčné honeypoty sú špecifické pre organizácie a ich úlohu je odhaliť a následne znížiť možné bezpečnostné riziká. Využívajú sa na odhalenie doposiaľ neodhalených zraniteľností a následné upozornenie príslušnej osoby. Tento typ honeypotov slúži na prvotnú detekciu útokov a na tento účel sa využívajú zväčša honeypoty s nízkou úrovňou interakcie z dôvodu ich jednoduchého nasadenia. Často sa využívajú v kombinácii s inými bezpečnostnými prvkami ako firewall alebo antivírusový systém. Ich účelom je okrem doplnenia bezpečnostnej infraštruktúry aj kontrola či sú bezpečnostné pravidlá organizácie postačujúce. Produkčné honeypoty nie sú schopné plne nahradiť všetky bezpečnostné riešenia, keďže monitorujú len aktivitu v sieti vykonávanú voči nim. Umožňujú simuláciu rôznych služieb operačného systému slúžiacich k prilákaniu útokov. Po objavení zraniteľných miest a nedostatkov je užívateľ v dostatočnom predstihu upozornený, čím sa vo vysokej miere znižuje riziko ohrozenia systému Produkčné honeypoty v porovnaní s výskumnými poskytujú menej informácií o útokoch a využívajú informácie získané výskumnými honeypotmi.

Výskumné honeypoty

Výskumné honeypoty sú využívané prevažne výskumnými (napr. univerzity), vládnymi alebo bezpečnostnými organizáciami. Ich úlohou je zistiť čo najpodrobnejšie informácie o hrozbách, ktorým organizácia čelí. Výskumné honeypoty sa využívajú k porozumeniu aktuálnym trendom a taktikám pri útokoch. Ich zámerom zvyčajne nie je ochrániť užívateľa, ale zhromaždiť čo najviac informácii o potenciálnom riziku a skúmať spôsob, akým je útok vedený na daný systém. Tento typ honeypotov je spravidla náročnejší na nasadenie a udržiavanie ako produkčné honeypoty, ale sú schopné získať podstatne väčšie množstvo informácií. Informácie získané výskumným honeypotom sú následne využité na obranu voči ďalším útokom. Útočníkovi je poskytnuté veľké množstvo priestoru a je možné sledovať jeho prácu, zistiť aké nástroje používa a dokonca odhaliť aj jeho motív a identitu. Organizácie si na základe získaných informácii vedia vypracovať bezpečnostné protokoly a zostať o krok pred útočníkom. Tento typ honeypotov je určený na to, aby sa učil a tak väčšina honeypotov na akademickej pôde je práve tohto typu.

2.2.3 Delenie podľa implementácie

Ďalším zo spôsobov ako môžme honeypoty rozdeliť je podľa implementácie. S narastajúcimi možnosťami vo svete virtualizácie je možné honeypoty rozdeliť na [13]:

- Fyzické honeypoty.
- Virtuálne honeypoty.

Fyzické honeypoty

Fyzické honeypoty predstavujú skutočné zariadenia, prípadne skutočný operačný systém so skutočnými službami, ktoré na ňom bežia. Jedná sa o stroj s vlastnou IP adresou a väčšinou ide o honeypot s vysokou mierou interakcie. Tieto typy honeypotov boli vo všeobecnosti náročné na inštaláciu a údržbu. Náklady spojené s využitím tejto technológie pri prevádzke väčšieho množstva honeypotov v systéme bývali podstatne väčšie ako pri využití virtuálnych honeypotov. Postupom času sa veľké množstvo nedostatkov spojených s fyzickými honeypotmi prekonalo a v súčasnej dobe je možné pozorovať pomerne lacné verzie týchto honeypotov. Rapídne sa znížilo aj množstvo času potrebného k inštalácii a následnej údržbe fyzických honeypotov. Jedným z príkladov súčasných fyzických honeypotov je *Thinkst Canary*[7]. Stále sa však vo väčšej miere využívajú fyzické honeypoty ak nie je možné implementovať požadovanú architektúru vo virtuálnom prostredí.

Virtuálne honeypoty

Na rozdiel od fyzických honeypotov virtuálne honeypoty predstavujú podstatnú úsporu počiatočných nákladov. Na jednom stroji je možné spravovať veľké množstvo rôznych druhov honeypotov. Ich výhoda oproti fyzickým honeypotom spočíva aj v nenáročnom nasadení, ľahkej údržbe a jednoduchej rozšíriteľnosti. Okrem virtuálnych strojov sa jedná aj o bežiace služby predstavované honeypotmi s nízkou mierou interakcie. Na vytváranie tohto typu honeypotov je možné použiť ľubovoľný komerčný alebo aj voľne dostupný virtualizačný nástroj (napr. VMware, Virtualbox apod.). V mnohých virtuálnych honeypotoch je implementovaná schopnosť zbierať potrebné informácie priamo z virtuálnych prostredí, v ktorých bežia.

2.2.4 Delenie podľa úlohy

Podľa úlohy ktorú predstavujú honeypoty v počítačovej sieti ich môžme rozdeliť na [13]:

- Klientské honeypoty.
- Serverové honeypoty.

Každý typ predstavuje jeden z koncov typickej komunikácie v počítačovej sieti.

Klientské honeypoty

Myšlienkou klientského honeypotu je napodobniť aplikáciu využívajúcu internetové služby. Ich cieľom je rozlíšiť škodlivý server od toho bezpečného. Tieto honeypoty sa využívajú na zhromaždenie informácií o zraniteľnostiach na strane klienta. Klientské honeypoty prehľadávajú sieť a snažia sa o aktívnu komunikáciu s novo-objavenými servermi. Tento typ honeypotov simuluje zraniteľné webové prehliadače, emailových klientov prípadne iné služby alebo ich doplnky. Prínos tohto typu honeypotov je nepopierateľný, keďže útoky pomocou webového prehliadača patria medzi najrozšírenejšie. Princíp klientského honeypotu sa dá rozdeliť do troch základných častí a to vytvoriť zoznam dostupných serverov, poslať požiadavky dostupným serverom, analyzovať interakciu so servermi a odhaliť prípadný útok. Získané údaje sa využívajú k vylepšeniu simulovaného softvéru a v napredovaní pred hrozbami internetu.

[7] https://canary.tools

Serverové honeypoty

Princíp serverových honeypotov spočíva v simulovaní servera. Ich úlohou je nevytvárať komunikáciu, ale vyčkávať, kým ju nenadviaže klient. Jedná sa o najrozšírenejší typ honeypotov, ktoré sú užitočné pri odhaľovaní nových útokov a zbere škodlivých vzoriek. Väčšinou sú napádané automatickými útokmi a všetka aktivita smerujúca na server sa dá považovať za škodlivú. Informácie získané zo serverových honeypotov sa využívajú k ochrane pred budúcimi útokmi. Jedná sa prevažne o honeypoty s nízkou úrovňou interakcie, ktoré simulujú jednu konkrétnu zraniteľnú službu. Príkladom takéhoto typu honeypotu je honeypot Dionaea, ktorého úlohou je odchytávať škodlivé vzorky, ktoré môžu byť následne analyzované a využité k rozpoznaniu nových alebo aktuálnych trendov medzi kybernetickými útokmi vo svete.

Kapitola 3

Nástroje na monitorovanie a správu konfigurácie

Správa konfigurácie je proces systematického spracovávania zmien a udržiavania integrity systému počas jeho behu. V celom procese zohráva zásadnú rolu automatizácia, ktorá umožní serverom nadobudnúť požadovaný stav pomocou vopred definovaných pravidiel [6].

Podstatná je schopnosť riadiť veľké množstvo serverov centrálne z jedného servera. Z pohľadu správcu systému je dôležité zohľadniť efektivitu nasadenia a úprav konfigurácie na serveroch. S pribúdajúcim počtom serverov rastie aj zložitosť konfigurácie a časová náročnosť jej nasadenia a prípadných zmien. Vykonávanie týchto zmien manuálne je mimoriadne namáhavé a prináša so sebou určitú chybovosť. Automatizácia tohto procesu pomocou nástrojov pre správu konfigurácie zvyšuje rýchlosť a redukuje úsilie pri prípadných opravách vzniknutých chýb [19].

Počiatočné náklady spojené s vytvorením pravidiel definujúcich požadovaný stav systému sú väčšie ako pri manuálnej konfigurácii serverov, avšak ich prínos v nasadzovaní alebo správe konfigurácií na veľkom množstve serverov je enormný. Pridanie nového servera do už existujúceho riešenia pri manuálnej konfigurácii zaberá podstatne väčšie množstvo času v porovnaní s vykonaním tejto činnosti za využitia automatizácie poskytnutej nástrojmi pre správu konfigurácie [8].

Medzi hlavné výhody využitia nástrojov pre správu konfigurácie patrí:

- Menšia náchylnosť na chyby – odsunutie ľudského faktoru do úzadia.
- Minimalizácia zásahov do procesu nasadzovania – nie je nutný priamy dohľad nad nasadzovaním.
- Vytvorenie riešenia, ktoré sa dá ľahko použiť na veľké množstvo rôznych serverov.
- Vytvorenie riešenia, ktoré je ľahko opakovateľné a rozšíriteľné.
- Jednoduché obnovenie systému po vzniknutých chybách.

3.1 Infraštruktúra ako kód

Infraštruktúra ako kód (anglicky *Infrastructure as Code*, ďalej IaC) je koncept vedúci k automatizácii infraštruktúry, ktorý vychádza zo skúseností pri vývoji softvéru. Dôležité je chápať infraštruktúru ako kód popisujúci konfiguráciu, ktorý je možno jednoducho testovať, verzovať a nasadzovať [14]. Tento koncept vytvára infraštruktúru podporujúcu a umožňujúcu zmeny v čase, pri ktorých sa predpokladá, že budú vykonávané opakovane. Nejedná sa o jednorázový koncept, ktorý by po prvotnom použití bol ďalej už nevyužiteľný. Pri uplatňovaní konceptu IaC je nutné zadefinovať jednotný formát popisujúci pravidlá, ktoré definujú očakávanú konfiguráciu. Dôležitou súčasťou konceptu IaC sú súbory, ktoré špecifikujú dané pravidlá. Väčšinou sa jedná o pravidlá zapísané formou slúžiacou na serializáciu dát, ktorá je pre človeka jednoducho čitateľná. Prevažne sa jedná o formáty ako YAML, JSON, CSV alebo XML. Výhodou využitia spomínaných formátov je, že výsledný súbor je strojovo čitateľný a jednotlivé pravidlá definujúce požadovanú konfiguráciu môžme interpretovať ako aplikačný kód. Medzi najznámejšie nástroje pre správu konfigurácie, ktoré využívajú koncept IaC patria Ansible[1], Chef[2], Puppet[3] a SaltStack[4] prípadne Terraform[5], ktorý sa od spomínaných odlišuje v tom, že sa jedná o nástroj spravujúci samotnú infraštruktúru. Spomínané nástroje budú popísané nižšie v kapitole 3.2.

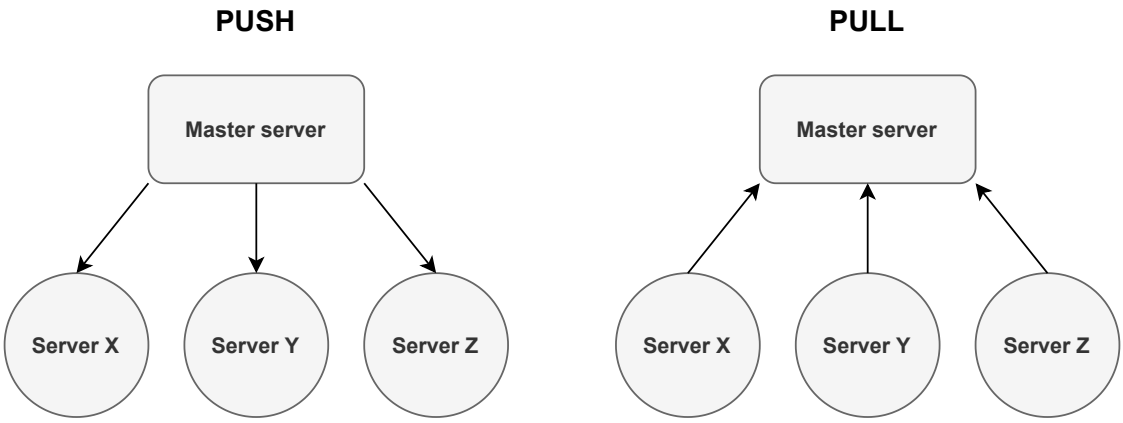
3.2 Porovnanie vybraných nástrojov pre správu konfigurácie

Existuje mnoho kritérií, podľa ktorých môžeme nástroje pre správu konfigurácie vzájomne porovnávať. Tieto kritéria nám umožňujú roztriediť dané nástroje do viacerých skupín, ktoré nám umožňujú zvoliť si nástroj čo najbližšie vyhovujúci našim počiatočným kritériám. Medzi kritéria zohľadňujúce najčastejšie požiadavky na výber nástroja pre správu konfigurácie radíme [2]:

- **Jazyk konfiguračných súborov** – jedná sa o jazyk, pomocou ktorého definujeme príkazy v konfiguračných súboroch. Väčšinou ide o formát určený na serializáciu dát ako JSON, XML alebo YAML.
- **Krivka učenia** – znázorňuje náročnosť daného nástroja. Vyjadruje časové obdobie potrebné k získaniu nevyhnutných znalostí. Priamo úmerne súvisí s náročnosťou syntaxe zvoleného jazyka.
- **Popularita nástroja** – vyjadruje mieru používania nástroja so zohľadnením aktuálnych trendov. Dôležitým faktorom je veľkosť komunity podieľajúcej sa na jeho vývoji a aj množstvo ľudí, ktorí ho aktívne využívajú. Popularita zvolených nástrojov je vyjadrená v tabuľke 3.1.
- **Prítomnosť dodatočnej aplikácie** – vyjadruje potrebu inštalácie dodatočnej aplikácie (agenta) na serveroch.
- **Prítomnosť riadiaceho servera** – predstavuje potrebu existencie tzv. Master servera, ktorý slúži na riadenie ostatných serverov. Ak sa jedná o architektúru, ktorá

[1] https://ansible.com/
[2] https://chef.io/
[3] https://puppet.com
[4] https://saltproject.io
[5] https://terraform.io/

využíva riadiaci server, delíme ju ďalej podľa spôsobu akým sú zvyšné servery riadené na *Push* model a *Pull* model. Princíp riadenia serverov je znázornený na obrázku 3.1. *Pull* model predstavuje komunikáciu medzi uzlami a riadiacim serverom kde uzly získavajú konfiguráciu od riadiaceho servera, teda komunikáciu zahajujú uzly. Naopak pri *Push* modeli komunikácie riadiaci server odošle konfiguračné dáta do uzlov, bez toho aby si o ne uzly požiadali.



Obr. 3.1: Rozdiel medzi *Push* a *Pull* modelom komunikácie

Ansible

Ansible je voľne dostupný automatizačný nástroj pre správu konfigurácií, ktorý je napísaný v jazyku Python a je jednoduchší na naučenie oproti iným nástrojom pre správu konfigurácie. Skladá sa zo súborov nazývaných aj *"Playbook"*, pričom jednotlivé súbory sú napísane vo formáte YAML [1].

Výhodou využitia formátu YAML je jeho jednoduchosť a tým pádom krivka učenia nástroja Ansible je výrazne strmšia ako u ostatných nástrojov. Jednoduchosť formátu YAML spočíva prevažne v jeho jednoduchej syntaxe a prehľadnosti. Konfiguračné súbory popisujú stav v akom sa má systém nachádzať a využívajú kombináciu procedurálneho a deklaratívneho princípu. Na druhej strane jednou z nevýhod formátu YAML je jeho slabšia výkonnosť, ktorá však nie je citeľná pri menších projektoch.

Popularitu nástroja Ansible vyjadruje frekvencia jeho používania za posledné roky znázornená v tabuľke 3.1, ktorá má rastúci trend. Komunita nástroja Ansible je pomerne aktívna a nástroj je pravidelne aktualizovaný.

Nástroj Ansible radíme do skupiny nástrojov, ktoré ku svojej činnosti nepotrebujú dodatočnú inštaláciu aplikácie (agenta) na spravovaných serveroch. K prístupu k serverom a ich následnej správe využíva Ansible protokol SSH, pomocou ktorého spúšťa príkazy a inštaluje potrebné aplikácie [9]. Samotný Ansible je rýchly a jednoduchý na inštaláciu, pričom ku konfigurácii serverov potrebuje iba prístupovú adresu a heslo na pripojenie. Je vyžadované, aby jednotlivé servery, na ktoré sa Ansible pripojuje mali nainštalovaný Python a povolený prístup cez protokol SSH.

Ansible radíme medzi nástroje, ktoré nevyžadujú potrebu riadiaceho servera. Riadenie vzdialených serverov je možné vykonávať aj z niektorého z nich, pričom jediná podmienka

je, aby bol nainštalovaný samotný Ansible. Nástroj Ansible neposkytuje užívateľské rozhranie. Jednotlivé úlohy, ktoré má nástroj vykonať na spravovaných serveroch sú definované v už spomínaných súboroch *Playbook*, ktoré je možné ďalej rozčleniť do rolí alebo modulov, kde každý z nich vykonáva určitú funkcionalitu. Spravované servery sú definované v súbore nazývanom *Inventár* umožňujúci ich kategorizáciu do skupín na základe použitia. Každý server môže mať definované premenné, ktorých hodnoty si uchovávame v súboroch s premennými viažúcimi sa na konkrétny server, prípadne je možné využívať aj globálne premenné, ktoré sa viažu ku všetkým serverom definovaným v inventári. Nástroj Ansible je možné využiť k správe serverov bežiacich na systéme Linux/Unix a Windows [11].

Chef

Chef je rozsiahly nástroj pre správu konfigurácií napísaný v špecifickom jazyku podobnom jazyku Ruby [24]. Podobne ako Ansible aj Chef je voľne dostupný a skladá sa zo súborov, ktoré sa v tomto prípade nazývajú konfiguračné súbory (*Recepty*). Recepty sú napísané v jazyku podobnom ako jazyk Ruby. Recepty sa zoskupujú do súborov receptov nazývaných *Kuchárky*, pričom každý z receptov musí do nejakej Kuchárky patriť [1].

Krivka učenia nástroja Chef je podstatne plytkejšia ako u nástroja Ansible z dôvodu využívania jazyka Ruby, ktorého syntax je všeobecne náročnejšia na pochopenie a menej prehľadná v porovnaní s formátom YAML. Využíva sa imperatívny prístup, teda konfiguračné súbory neopisujú stav v akom sa má systém nachádzať, ale kroky ako tento stav dosiahnuť.

Architektúra nástroja Chef pozostáva z troch základných prvkov:

- **Pracovná stanica** – lokálne bežiaci nástroj Chef obsahujúci všetky recepty, kuchárky a nástroj *Knife*, ktorý je využitý k ich prenosu na *Chef Server*.
- **Chef server** – server, ktorý ukladá súbory pre konfiguráciu a údaje z uzlov.
- **Uzly** – servery, ktoré komunikujú s *Chef serverom* a získavajú od neho konfiguračné súbory. Uzly využívajú *Pull* model. Na uzloch je nevyhnutné mať nainštalovanú klientskú aplikáciu *Chef Client* slúžiacu na vykonanie zmien definovaných v stiahnutých konfiguračných súboroch. Výsledok operácie je spätne odoslaný na server.

Z vyššie spomínaných informácií vyplýva, že nástroj Chef vyžaduje prítomnosť dodatočnej aplikácie na serveroch a je nevyhnutná aj prítomnosť riadiaceho servera. Nástroj Chef je podporovaný operačnými systémami ako Linux/Unix Mac OS a Windows tak na klienta ako aj na strane servera [11].

Puppet

Puppet je nástroj, ktorý je podobne ako *Chef* napísaný v jazyku Ruby. Je taktiež určený na automatizáciu a spravovanie konfigurácií na serveroch. Konfiguračné súbory zvané *"Manifesty"* sú napísané v jazyku odvodenom od jazyka Ruby obohateného o špecifiká nástroja Puppet a syntaxou sa podobá na formát JSON. Nástroj Puppet využíva deklaratívny prístup, teda konfiguračné súbory špecifikujú v akom stave má systém byť, nie ako sa do daného stavu má dostať. Jednotlivé Manifesty môžu byť združované spolu s ďalšími potrebnými súbormi do modulov [11].

Krivka učenia pri nástroji Puppet je podobná krivke učenia pri nástroji Chef, keďže sa jedná o podobné nástroje založené na jazyku Ruby, pričom je vyžadovaná jeho dobrá znalosť.

Nástroj Puppet je podobne ako Chef menej populárny v porovnaní s nástrojom Ansible, avšak stále sa jedná o dosť populárne, široko využívané nástroje s bohatou dokumentáciou.

Pri využívaní nástroja Puppet je vyžadovaná prítomnosť dodatočnej klientskej aplikácie (Puppet Agent) na riadených serveroch. Táto aplikácia komunikuje s riadiacim serverom, od ktorého si v periodických intervaloch žiada katalóg prípadne katalógy s Manifestami definujúcimi požadovaný stav servera (*Pull* model). Po uvedení servera do požadovaného stavu sa výsledok odošle naspäť riadiacemu serveru. Je vyžadovaná prítomnosť minimálne jedného riadiaceho servera [24].

Na rozdiel od nástroja Chef je riadiaci server nástroja Puppet možné nainštalovať len na serveroch založených na operačnom systéme Linux. Klientské aplikácie môžu byť inštalované na operačné systémy ako Linux/Unix, Mac OS tak aj Windows.

SaltStack

Podobne ako nástroj Ansible aj nástroj SaltStack je napísaný v jazyku Python. Jedná sa o pomerne nový nástroj, ktorého najväčšia devíza sú rýchlosť a rozširovateľnosť. Konfiguračné súbory sa nazývajú *"states"* a sú napísané vo formáte YAML. Je podporovaný deklaratívny a aj imperatívny prístup k nasadeniu konfigurácie. Jednotlivé konfiguračné súbory sú združované do modulov, ktoré sú zodpovedné za uvedenie systému do požadovaného stavu. Formát YAML je predvolený, ale podporovaný je aj napríklad formát JSON alebo mnoho iných [11].

Nástroj SaltStack podporuje konfiguračné súbory napísané vo formáte YAML, ktoré sa vyznačujú jednoduchou syntaxou a ľahkou čitateľnosťou. Ide o nástroj s pomerne nízkou krivkou učenia vzhľadom na veľké množstvo vopred definovaných modulov. Nevýhodou nástroja SaltStack je absencia modulov tretích strán a užívateľ si musí vystačiť iba s vopred definovanými modulmi.

SaltStack podporuje princíp riadenia kedy je na riadených serveroch (nazývane *minions*) nainštalovaná klientská aplikácia, ale takisto je možné využiť aj princíp fungovania bez dodatočnej aplikácie na riadených serveroch. Pri využití princípu bez dodatočnej aplikácie sú požadované servery obsluhované cez protokol SSH, ktorý je ale výrazne pomalší ako pri využití klientskej aplikácie. Využitie princípu s inštaláciou klientskej aplikácie sa okrem rýchlosti vyznačuje aj nutnosťou existenciu riadiaceho servera alebo serverov, ktoré môžu byť usporiadané do viacerých úrovní. Servery na najvyššej úrovni riadia tie na nižšej a tie im prislúchajúce servery s klientskou aplikáciou. Pri takejto architektúre nie je nutné ukladať konfiguráciu riadených serverov lokálne. Je podporovaná správa viacerých uzlov zároveň [24].

Podobne ako nástroj Puppet aj SaltStack podporuje inštaláciu riadiaceho servera len na systémy založených na operačnom systéme Linux. Klientské aplikácie môžu bežať na všetkých bežných operačných systémoch ako Linux/Unix, Mac OS a aj Windows.

Terraform

Jedná sa o mierne odlišný nástroj v porovnaní s vyššie spomínanými. Nástroj Terraform sa radí medzi nástroje spravujúce infraštruktúru a nie konfiguráciu. Nie je potrebná dodatočná inštalácia a ani prítomnosť ďalších aplikácii na spravovaných serveroch. Hlavnou výhodou nástroja Terraform je jeho rozšírenosť medzi najznámejšími poskytovateľmi Cloud služieb ako AWS, Azure a mnoho iných. Je využitý deklaratívny princíp konfigurácie serverov. V jednotlivých moduloch je popísaný stav, v ktorom sa má systém nachádzať a následne je možné celú infraštruktúru nasadiť použitím jedného príkazu [2].

Jednotlivé moduly sú napísané vo vlastnom formáte pripomínajúcom formát JSON. Ide o pomerne jednoduchý nástroj s nízkou krivkou učenia.

Nástroj Terraform je často využívaný v spojení s nástrojom Ansible. V konfiguračnom súbore nástroja Ansible nazývanom *Playbook* je možné okrem konfigurácie servera využiť aj modul Terraform na nasadenie samotnej infraštruktúry [2].

Tabuľka 3.1 znázorňuje popularitu vybraných nástrojov pre správu konfigurácie. Je možné pozorovať, že vo väčšine ukazovateľov si vedie nástroj Ansible. Jednotlivé hodnoty sú farebne odlíšené, pričom zelená znamená, že vybraný nástroj patril k trom najpopulárnejším nástrojom a naopak červená farba znamená, že patril k dvom najhorším nástrojom v danom ukazovateli. Čím tmavší je odtieň farby, tým lepšie prípadne horšie sa daný nástroj umiestnil.

	Prispievatelia	Hviezdičky	Príspevky za mesiac	Nahlásené chyby za mesiac	Knižnice	Príspevky na SO	Pracovné ponuky
Ansible	4 386	37 161	506	523	20 677	11 746	8 787
Chef	562	5 794	435	86	3 832	5 982	4 378
Puppet	515	5 299	94	314	6 110	3 585	4 200
SaltStack	2 237	9 901	608	441	318	1 062	1 622
Terraform	1 261	16 387	173	204	1 462	2 730	3 641

Tabuľka 3.1: Popularita vybraných nástrojov pre správu konfigurácie k máju 2019 [6]

3.3 Monitorovanie infraštruktúry

Pri nasadzovaní honeypotov je dôležité zabezpečiť aj monitorovací proces. Jeho úlohou je kontinuálne a celoplošne získavať informácie o vyťaženosti servera ale aj o nasadených procesoch. Monitorovaním odhalíme zlyhania a je to spôsob ako diagnostikovať problémy na serveroch za behu. Dôležitou funkciou monitorovania je signalizácia chýb administrátorovi, ktorý sa následne môže prihlásiť do systému a analyzovať vzniknuté závady. Prípadne riešenie chýb môže byť zabezpečené aj automaticky bez manuálneho zásahu administrátora. V tomto prípade monitorovanie slúži len k agregovaniu monitorovaných metrík a údajov o procesoch a následnému upozorneniu ak nastala chyba, teda zachytené hodnoty sa líšia od očakávaných. Vzhľadom na praktickú časť predkladanej práce v následujúcej podkapitole priblížim voľne dostupný nástroj Prometheus[7] (viz 3.3), ktorý bol zvolený pre monitorovanie mnou nasadenej infraštruktúry.

Prometheus

Prometheus patrí aktuálne k najpopulárnejším voľne dostupným nástrojom určeným k monitorovaniu systémov. Úlohou nástroja Prometheus je zbierať informácie a kvantitatívne údaje o bežiacich procesoch alebo o samotnom serveri. Nadobudnuté údaje v sebe obsahujú časové razítko identifikujúce čas vzniku. Prometheus okrem zbierania a vyhodnocovania dát

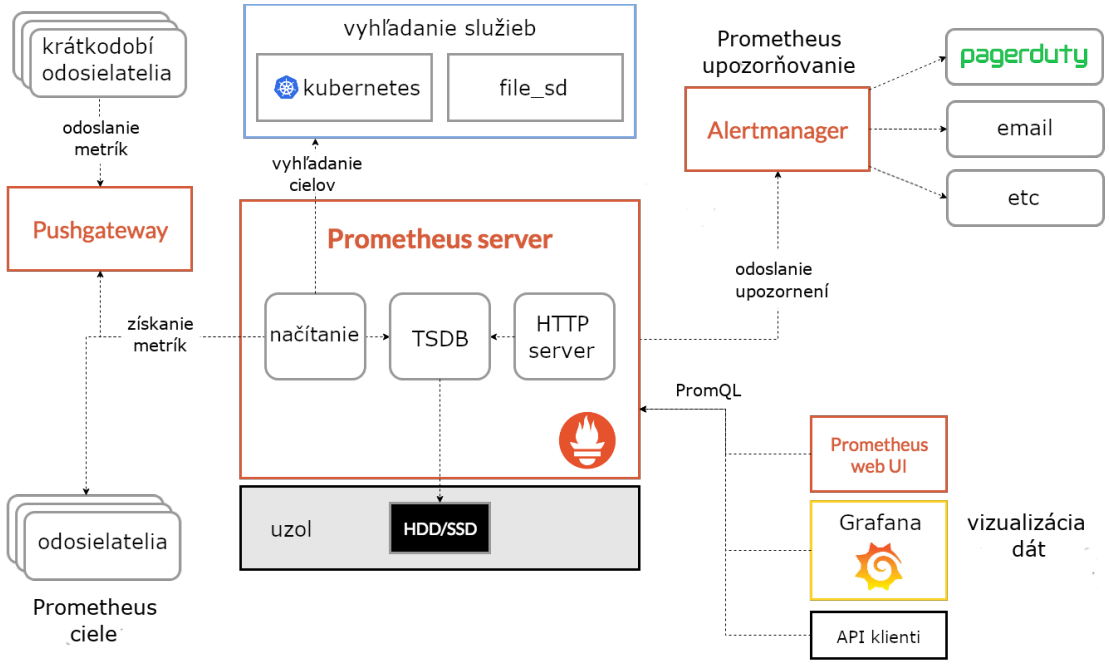
[6] https://blog.gruntwork.io/why-we-use-terraform-and-not-chef-puppet-ansible-saltstack-or-cloudformation-7989dad2865c

[7] https://prometheus.io/

umožňuje aj upozorňovanie administrátora pomocou konfigurovateľných výstražných správ. Súčasťou nástroja Prometheus je aj webové rozhranie umožňujúce jednoduchú vizualizáciu monitorovaných serverov, nadobudnutých dát či pravidiel pre upozorňovanie [23].

Prometheus pracuje na princípe *Pull* modelu, kedy vopred definované metriky zbiera z koncových serverov. Na koncových serveroch musí byť spustený program tzv. *node exporter*, ktorý exportuje monitorované metriky koreňovému serveru, na ktorom Prometheus beží. Údaje sú odosielané za využitia protokolu HTTP. Údaje sú ukladané lokálne pod jednoznačným identifikátorom skladajúcim sa z časového razítka a názvu zberaného údaju. Jednou z výhod nástroja Prometheus je možnosť vykresliť graf popisujúci vývoj hodnôt danej metriky v čase, avšak vo väčšine prípadov je vhodné využiť programy tretích strán na vizualizáciu dát (napr. *Grafana*[8]). V prípade porušenia administrátorom definovaného pravidla určujúceho povolený rozsah metrík je možné využiť komponent nazývaný *Alertmanager*[9]. Alertmanager slúži k spracovaniu vzniknutého upozornenia a následnému odoslaniu varovnej správy užívateľovi. Koncové zariadenie, na ktoré je upozornenie zaslané je možné definovať v konfiguračnom súbore. Užívateľ si môže nastaviť, aby upozornenia boli zasielané napríklad na email, Slack[10], pager a mnoho iných technológií [23].

Schéma 3.2 popisuje princíp fungovania nástroja Prometheus jeho komponentov.



Obr. 3.2: Prometheus architektúra[11]

[8] https://grafana.com/
[9] https://github.com/prometheus/alertmanager
[10] https://slack.com/
[11] https://prometheus.io/docs/introduction/overview/

Kapitola 4

Návrh

Cieľom tejto práce je vypracovať riešenie umožňujúce efektívne a predovšetkým hromadné nasadenie viacerých druhov honeypotov na cieľových serveroch z jedného počiatočného bodu. Výsledné riešenie si kladie za cieľ jednoduchú škálovateľnosť spočívajúcu prevažne v doplnení jednej krátkej časti pravidiel na vopred špecifikované miesto pre nasadenie nového druhu honeypotu. V počiatočnej fáze návrhu sa počíta s nasadzovaním troch rôznych honeypotov za využitia jazyku Ansible. Jazyk Ansible je využívaný na špecifikáciu výslednej infraštruktúry pomocou kódu. Kód pozostáva z pravidiel definujúcich požadovaný výsledný stav na koncových zariadeniach. Spolu s nasadzovaním honeypotov sa nasadzuje aj monitorovanie koncového servera a procesov na ňom bežiacich. K monitorovaniu zdrojov sa využije softvér Prometheus (viz 3.3), ktorý umožňuje okrem monitorovania aj jednoduché upozorňovanie užívateľov na rôznych komunikačných médiách. Na vizualizáciu je využitý nástroj Grafana a aplikácia Slack na upozorňovanie. V prípade zlyhania niektorého z procesov je realizovaná automatická obnova tohoto procesu. Pri návrhu aplikácie je braný ohľad na maximálnu užívateľskú prívetivosť a jednoduchosť používania.

4.1 Zvolené technológie

V rámci mojej bakalárskej práce som sa rozhodol pre využitie nástroja Ansible. Ansible je aktuálne najpoužívanejší a najpopulárnejší nástroj pre správu konfigurácie. Vzhľadom k jednorázovému nasadeniu a konfigurácii serverov je výhodný jeho *Push* model, kde nie je potrebný žiaden riadiaci server ani nie je potrebné mať nainštalovanú žiadnu ďalšiu aplikáciu na uzloch. Honeypoty budú prevažne nasadzované na množstvo serverov so slabším výkonom. Dôležité je, aby bežalo okrem samotného honeypotu čo najmenej dodatočných aplikácii.

K monitorovaniu serverov bola zvolená architektúra Prometheus, ktorá poskytuje všetky nevyhnutné súčasti ako monitorovanie metrík, procesov, definovanie vlastných pravidiel a upozorňovanie užívateľa pri ich porušení. Prometheus vďaka svojej jednoduchosti disponuje obrovskou komunitou, ktorá prináša množstvo podporných aplikácií a integrácií uľahčujúcich jeho nasadenie a následné využívanie.

Rozhodol som sa pre využitie aplikácie Slack k upozorneniu užívateľa. Táto aplikácia je denno-denne používaná v rámci firmy Avast, ktorá je zadávateľom tejto bakalárskej práce. Upozornenia je možné priamo napojiť na vybraný kanál a upozorniť tým želaného užívateľa alebo skupinu užívateľov.

Dôležitou súčasťou monitorovania výsledného riešenia je aj vizualizácia získaných dát. K vizualizácii dát som sa rozhodol využiť nástroj Grafana, ktorý je takisto rozsiahlo využívaný vo firme Avast. Získané dáta môžu byť napojené priamo na ich interný systém, ktorý sa stará o ich klasifikáciu a následnú analýzu.

4.2 Návrh architektúry

Celková architektúra riešenia sa skladá z počiatočného zariadenia vykonávajúceho nasadzovanie honeypotov. Po spustení programu je užívateľovi umožnené si vybrať, ktorý z ponúknutých honeypotov bude nasadený na koncové zariadenia bližšie definované v konfiguračnom súbore (viz 4.3). Okrem špecifikácie nasadzovaného honeypotu je možné povoliť aj nasadenie monitorovania na vopred špecifikovaných serveroch. Samotné nasadenie honeypotu je ďalej realizované pomocou krokov zapísaných jazykom YAML. Ansible skript je štrukturovaný do tzv. rolí, ktoré sa vykonajú na základe užívateľského vstupu. Role môžme rozdeliť do samostatných kategórii a to na:

- Dionaea – rola predstavujúca nasadenie Dionaea honeypotu.
- Hermes – rola predstavujúca nasadenie Hermes honeypotu.
- Cowrie – rola predstavujúca nasadenie honeypotu Cowrie.
- Prometheus – rola pre nasadenie softvéru Prometheus na počiatočné zariadenie (zariadenie na ktorom je spustený skript).
- Node Exporter – rola predstavujúca nasadenie aplikácie monitorujúcej základné metriky na koncových zariadeniach, medzi ktoré patrí napríklad vyťaženosť procesora, aktuálne voľné miesto na disku a iné.
- Process Exporter – rola predstavujúca nasadenie aplikácie monitorujúcej údaje o vopred definovaných procesoch bežiacich na koncových zariadeniach.

Spomínané typy honeypotov a ich vlastnosti sú popísané v sekcii 4.5. Pri nasadení honeypotov je možné interagovať s užívateľom, ktorý pomocou konzoly dokáže upraviť konfiguráciu už samotného honeypotu. Každý z honeypotov má špecifické hodnoty ktoré je možné upraviť. Medzi niektoré z nich patria:

- Úprava užívateľského profilu honeypotu Dionaea (meno, doména, názvy diskov).
- Úprava IP adresy, portu a užívateľských údajov pre mailový honeypot Hermes.
- Konfigurácia úvodnej obrazovky, prihlasovacích údajov a súborov pri honeypote Cowrie.

Každá z konfigurácií ma za úlohu zaistiť čo najväčšiu autenticitu daného honeypotu a rôzne variácie konfigurácii v rámci jedného druhu. Pri nasadzovaní na viacero serverov je umožnená individuálna konfigurácia každého z nich. Užívateľské otázky boli umiestnené na začiatok konfigurácie, hneď po zvolení typu honeypotu, ktorý sa bude nasadzovať. Užívateľ má tak možnosť na začiatku zvoliť požadovanú konfiguráciu a následne nechať bežať program bez ďalšieho zásahu. Po dokončení nasadenia a inštalácie na vybraných serveroch je nasadzovaný honeypot spustený. Údaje získané z honeypotu sú pomocou protokolu MQTT zaslané do centrálneho servera realizujúceho ich spracovanie a vizualizáciu pomocou nástroja Grafana.

4.3 Konfigurácia

Typ honeypotu, ktorý bude nasadený spolu s jeho unikátnou konfiguráciou je volený pomocou užívateľského vstupu pri spustení skriptu. K nastaveniu serverov, na ktoré budú jednotlivé druhy honeypotov nasadzované sa využíva konfiguračný súbor `inventory.yml`, v ktorom je možné špecifikovať tzv. značku (ďalej ako tag). Tag sa zapisuje do hranatých zátvoriek a definuje aká rola sa použije na IP adresy alebo doménové mená pod ním špecifikované. Príkladom validného obsahu súboru `inventory.yml` môže byť kód vo výpise 4.1.

```
[dionaea]
1.1.1.1
2.2.2.2

[hermes]
3.3.3.3

[cowrie]
4.4.4.4
5.5.5.5

[executors]
1.1.1.1
2.2.2.2
3.3.3.3
5.5.5.5

[monitoring]
127.0.0.1
```

Výpis 4.1: príklad nastavenia súboru `inventory.yml`

Vyššie uvedená konfigurácia značí nasadenie honeypotu typu Dionaea na serveroch s IP adresami 1.1.1.1 a 2.2.2.2, nasadenie honeypotu Hermes na server s IP adresou 3.3.3.3 a nasadenie honeypotu Cowrie na serveroch s IP adresami 4.4.4.4 a 5.5.5.5. Odosielatelia metrík a informácii o bežiacich procesoch sú nasadení na serveroch s IP adresami 1.1.1.1, 2.2.2.2, 3.3.3.3 a 5.5.5.5. Infraštruktúra na monitorovanie je nasadená na lokálny stroj (IP adresa 127.0.0.1). Pre každé z uvedených koncových zariadení musí existovať konfiguračný súbor, ktorý popisuje prístupové práva a prihlasovacie údaje pre Ansible skript. Z dôvodu bezpečnosti a zamedzenia úniku prístupových údajov sa využíva šifrovanie spomínaných konfiguračných súborov. K šifrovaniu a dešifrovaniu sa využíva modul Ansible Vault, ktorý je súčasťou softvéru Ansible. Ďalšia konfigurácia už konkrétnych rolí prebieha v konfiguračných súboroch umiestnených v priečinku *templates* prípadne *vars* pod každou rolou. Konfiguračný súbor nachádzajúci sa v priečinku *vars* umožňuje priradenie hodnoty premenným využívaných počas behu Ansible skriptu, užívateľ si tu môže nakonfigurovať napríklad verziu nasadzovaného honeypotu či cestu k domovskému priečinku kde honeypot bude nainštalovaný. Priečinok *templates* umožňuje pokročilejšie nastavenia konfiguračných súborov nasadzovaných spolu s honeypotom k zaisteniu jeho správneho behu. Keďže o väčšinu konfigurácií samotných honeypotov sa stará užívateľský vstup, je zasahovanie do súborov v priečinku *templates* neodporúčané. Dodatočnú konfiguráciu je vhodné vykonať podľa inštrukcií v dokumentácii ku konkrétnemu honeypotu a v konfiguračných súboroch po nainštalovaní.

4.4 Monitorovanie

K monitorovaniu jednotlivých serverov a na nich bežiacich procesov súvisiacich s nasadenými honeypotmi sa využíva voľne dostupný nástroj Prometheus (viz 3.3). Monitorovací proces sa skladá z dvoch častí - Prometheus klient a Node/Process odosielateľ tzv. exporter. Serverová časť aplikácie Prometheus je nainštalovaná na server, z ktorého inicializujeme komunikáciu a zahajujeme nasadzovanie honeypotov. Na koncových serveroch je nasadená klientská časť, ktorá spočíva v zbieraní podstatných údajov a metrík. Serverová strana si tieto metriky v pravidelný intervaloch žiada pomocou *Pull* modelu popísaného v sekcii 3.3. Údaje sú následne spracované, uložené a vyhodnocované. Súčasťou balíka Prometheus je aj modul AlertManager, ktorý slúži k zasielaniu upozornení užívateľovi, ak boli porušené definované pravidlá. Užívateľ je schopný upraviť už existujúce pravidlá, či už v súbore `alert_rules.yml.j2` ešte pred samotným nasadením alebo aj v po nasadení v súbore `alert_rules.yml` nachádzajúcom sa v priečinku, kde bol Prometheus nasadený. Východzie nastavenie počíta s cestou `/root/Prometheus/prometheus-<v>.linux-amd64`, kde `<v>` značí aktuálnu verziu softvéru Prometheus. K základným pravidlám pre monitorovanie metrík na serveroch patrí:

- Výpadok samotnej aplikácie zbierajúcej metriky (Node/Prometheus exporter), prípadne neschopnosť s ňou komunikovať.
- Výpadok procesu súvisiaceho s nasadeným honeypotom.
- Nedostatok voľného miesta na monitorovanom serveri.
- Nadmerné vyťaženie procesoru po dlhšiu dobu na monitorovanom serveri.
- Nedostatok pamäte RAM na monitorovanom serveri.

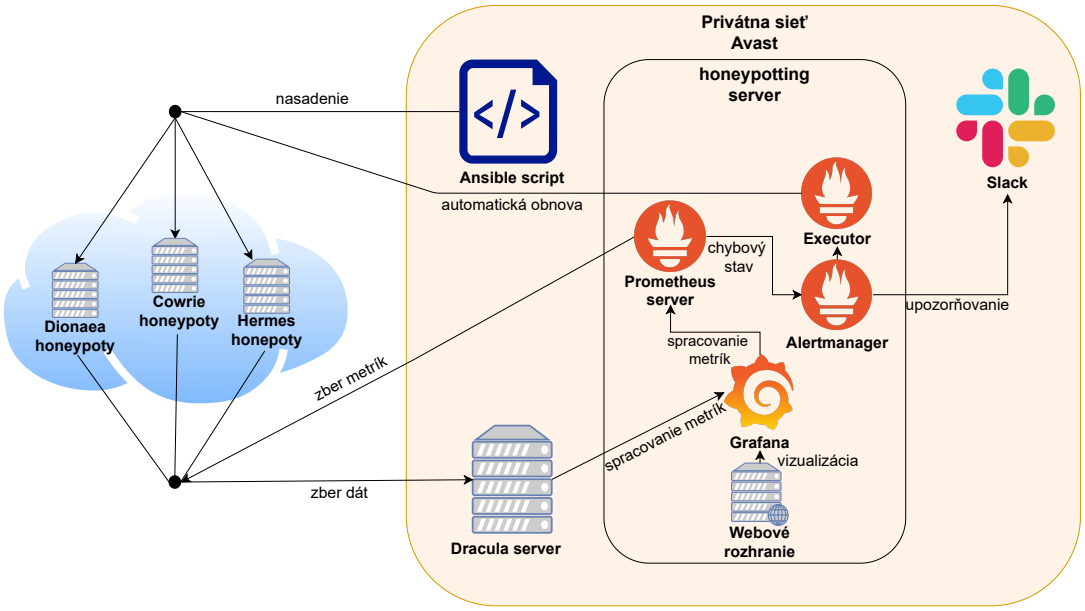
Pravidlá je možné ľubovoľne meniť a dopĺňať, je však nutné dodržiavať oficiálnu syntax[1].

Upozornenia a zotavenie

K upozorneniu užívateľa dochádza pri porušení jedného z definovaných pravidiel. Modul AlertManager umožňuje viaceré spôsoby upozorňovania užívateľa. K tým najvyužívanejším patrí napríklad email, Slack alebo pager. V mojej práci som sa rozhodol pre využitie upozorňovania pomocou aplikácie Slack, ktorá je najvyužívanejším komunikačným prostriedkom vo firme. Výhodou upozorňovania pomocou aplikácie Slack je schopnosť prispôsobovať upozornenia a zahrnúť v nich všetky podstatné detaily užívateľsky prívetivou formou.

Jednou zo súčastí riešenia je možnosť automatického zotavenia sa pri prípadných chybách. Aktuálne je možné automatické spustenie honeypotu Dionaea pri jeho výpadku na ľubovoľnom serveri. Zotavenie sa je možné realizovať za pomocou modulu `prometheus-am-executor`[2], ktorý reaguje na prípadne porušenie pravidla kontrolujúceho proces na serveri (viz 4.4) a spustí skript starajúci sa o obnovu procesu pri výpadku. Architektúra navrhnutého systému je znázornená na obrázku 4.1.

[1] https://prometheus.io/docs/prometheus/latest/configuration/alerting_rules/
[2] https://github.com/imgix/prometheus-am-executor



Obr. 4.1: Návrh architektúry nasadenia, monitorovania a vizualizácie dát

4.5 Využité honeypoty

V tejto práci som sa rozhodol pre nasadenie a konfiguráciu troch honeypotov doplnených o všeobecnú kostru nasadenia ľubovoľného honeypotu v budúcnosti. Jedná sa o dva voľne dostupné a rozšírené honeypoty Dionaea a Cowrie. Tretím zvoleným honeypotom je Hermes[3], ktorý bol navrhnutý a implementovaný v rámci minuloročnej bakalárskej práce[4] v spolupráci s firmou Avast. Pre nasadenie spomenutých honeypotov som sa rozhodol prevažne na základe ich rozšírenosti, možnosti konfigurácie a miery pokrytia útokov vo svete. Do úvahy boli takisto vzaté požiadavky zadávateľa a firemné ciele.

Dionaea

Honeypot Dionaea patrí k honeypotom s nízkou mierou interakcie [18]. Vznikol v roku 2009 v rámci projektu *The Honeynet Project*[5] prebiehajúceho počas programu *Google Summer of Code*[6].

Dionaea emuluje viacero zraniteľností pomocou rôznych protokolov ako napríklad SMB, FTP, HTTP, VOIP a rôzne ďalšie. Pri napadnutí jednej z emulovaných slabín Dionaea dokáže zachytiť použitú škodlivú vzorku (malware) bez toho, aby nejakým spôsobom poškodila systém na ktorom honeypot beží. Pojem malware môžme charakterizovať ako softvér, ktorý sa snaží získať nepovolený prístup k zariadeniu, poškodiť zariadenie alebo znepríjemniť život užívateľovi [7]. Vzhľadom na užívateľskú bezpečnosť honeypot Dionaea beží v uzavretom prostredí a nie je nutné ho spustiť s administrátorskými právami. Okrem možnosti zachytávania škodlivých vzoriek honeypot Dionaea takisto ponúka rozšírený logovací

[3] https://github.com/avast/hermes
[4] https://fit.vut.cz/study/thesis/22243/
[5] https://honeynet.org
[6] https://summerofcode.withgoogle.com

systém. Nad logovacím systémom bežia ďalšie moduly, ktoré ho monitorujú a odosielajú podrobné informácie o vzniknutých incidentoch, čo je pre užívateľa nesmierne cenné. Medzi najvyužívanejšie moduly starajúce sa o prenos metadát o incidentoch k užívateľovi patria napríklad *hpfeeds*, *log_sqlite*, *log_json submit_http* alebo *VirusTotal*. Je na užívateľovi, ktorý z nich si zvolí a nakonfiguruje pre vlastné potreby.

Nasadenie pomocou Ansible skriptu bude prebiehať v troch fázach. V prvej užívateľ špecifikuje doménové meno a názov emulovaného disku pre každý server, na ktorý bude honeypot Dionaea nasadený. V druhej fáze prebehnú automatické inštalácie bez nutnosti ďalšieho zásahu užívateľa pozostávajúce zo stiahnutia vopred definovanej verzie Dionaea honeypotu, doinštalovanie potrebných balíčkov a jeho následná inštalácia na všetkych špecifikovaných serveroch 4.3. V tretej fáze prebehne úprava pôvodnej konfigurácie honeypotu, zvolenie protokolu, ktorý bude emulovaný, presun podporných skriptov pre zachytávanie nových vzoriek a ich odosielanie na neskoršiu analýzu a samotné spustenie honeypotu. V tejto práci som sa rozhodol pre emulovanie protokolu SMB, ktorý je pre honeypot Dionaea špecifický a jeho pokrytie inými dostupnými honeypotmi je veľmi zriedkavé. Takisto útoky zneužívajúce zraniteľnosti protokolu SMB sú veľmi rozšírené. K odosielaniu škodlivých vzoriek som sa rozhodol využiť jednoduché skripty napísané v jazyku Python, ktoré novovzniknuté škodlivé vzorky odošlú pomocou protokolu MQTT k následnej analýze do internej siete. Správy obsahujúce či už škodlivé vzorky alebo metadáta o útoku sú odoslané na vopred špecifikovanú tému (topic) prostredníctvom centrálneho bodu tzv. *broker*, ktorý ich ukladá. Na druhej strane komunikácie je potrebný prijímateľ tzv. *subscriber*, ktorý je pripojený na takisto vopred zvolenú tému a pri príchode správy na túto tému do centrálneho bodu je mu táto správa preposlaná [10].

Cowrie

Cowrie sa považuje za voľne dostupný honeypot so strednou až vysokou mierou interakcie. Jedná sa o jeden z najrozšírenejších a najaktívnejšie rozvíjaných honeypotov. Umožňuje emulovanie protokolu SSH a Telnet. Zaznamenáva údaje o útokoch hrubou silou ale aj údaje o jednotlivých príkazoch a interakciách prevedených útočníkom na honeypote. Honeypot cowrie dokáže fungovať v dvoch režimoch. V prvom alebo tzv. interaktívnom režime emuluje UNIX systém pomocou jazyka Python. V druhom tzv. proxy režime, ktorý umožňuje vysokú mieru interakcie spočívajúcu v pozorovaní útočníkovho správania na inom systéme. Honeypot Cowrie po inštalácii obsahuje štandardnú konfiguráciu, ktorá mu umožňuje bezproblémové fungovanie a zaznamenávanie veľkého množstva útokov. Avšak pre náročnejšie používanie je žiadúca úprava, ktorá zamedzí alebo minimálne sťaží identifikáciu honeypotu útočníkom, prípadne automatickým skriptom. Hodnota honeypotu významne narastá pokiaľ je nedetekovaný a útočník si myslí, že pracuje s reálnym systémom [3]. Honeypot Cowrie podobne ako Dionaea umožnuje logovanie informácii o incidentoch v čase a ich následné triedenie do jednotlivých útokov na základe identifikácie naviazaného spojenia. Metadáta o útokoch ako IP adresu útočníka, príkazy zadané do konzoly pri útoku, dĺžku trvania spojenia a mnoho iných je možné pravidelne odosielať pomocou vstavaných modulov ako *hpfeeds*, *mysql*, *kafka* alebo *virustotal*. Je na užívateľovi či chce metadáta odosielať a následne spracovať a ktorý modul k tomu využije. Okrem ukladania metadát o incidentoch dokáže Cowrie ukladať aj škodlivé vzorky, ku ktorých stiahnutiu došlo počas útoku.

Pri nasadzovaní honeypotu Cowrie v rámci bakalárskej práce som sa rozhodol honeypot vopred predkonfigurovať a tak zaistiť jeho unikátnosť v sieti a zamedziť jeho jednoduchej identifikácii útočníkom. Nasadenie môžme podobne ako pri honeypote Dionaea rozdeliť do

troch častí, kde v prvej časti nainštalujeme potrebné balíčky a stiahneme vopred zvolenú verziu Cowrie. V druhej časti dochádza k inštalácii honeypotu a úprave konfiguračných súborov, prípadne rozšírenie konfigurácie. V poslednej časti sa nastaví odosielanie výstupných logov a honeypot sa spustí. Rozhodol som sa takisto pre vlastné odosielanie metadát pomocou protokolu MQTT, ktorý sprostredkúva skript napísaný v jazyku Python. Odosielanie vzoriek nie je potrebné, keďže informácie o stiahnutí vzorky a adresa, z ktorej bola stiahnutá sú obsiahnuté v logu a pre ďalšiu analýzu sú pre nás tieto informácie postačujúce. Forma prihlásenia útočníka k honeypotu pri SSH útoku je možná v dvoch variantách. Prvá dovoľuje útočníkovi prístup pod prístupovými údajmi, ktoré sú špecifikované v konfigurácii. Druhá metóda je pokročilejšia a umožňuje útočníkovi prístup po zadaní náhodných prístupových údajov a po náhodnom počte pokusov. Pri druhej variante dôjde k uloženiu prístupových údajov pre danú IP adresu a následne sa útočníkovi z tejto IP adresy povolí prístup len pod rovnakými prístupovými údajmi. Rozhodol som sa pre využitie druhej varianty, ktorá sa javí z hľadiska bezpečnosti a sťaženia identifikácie honeypotu ako lepšia.

Hermes

Honeypot Hermes je nový honeypot, ktorý bol vyvinutý minulý rok v rámci bakalárskej práce [4]. Ide o SMTP[7] honeypot, ktorý bol postavený na základe predlohy honeypotu SHIVA[8] napísaného v jazyku Python 2. Hlavným cieľom bolo prepísať tento honeypot do jazyku Python 3, ktorý je podporovaný aj v dnešnej dobe. Hermes je honeypot s vysokou mierou interakcie a jeho hlavným cieľom je zber nevyžiadanej pošty. Architektúra honeypotu Hermes sa skladá z dvoch častí. Prvá časť (receiver) slúži na príjem a uloženie elektronickej pošty. Druhá časť (relay) sa stará o spracovanie pošty a jej preposielanie skutočným príjemcom, ak je táto možnosť povolená v konfigurácii. Pre prenos správ k ďalšej analýze v internej sieti sa podobne ako pri prvých dvoch predstavených honeypotoch využíva protokol MQTT. Medzi hlavné výhody honeypotu Hermes oproti konkurenčným SMTP honeypotom patrí [4]:

- Ukladanie nevyžiadanej pošty a jej preposielanie pomocou protokolu MQTT.
- Podpora SMTP autentizácie a možnosť definovania prihlasovacích údajov v konfiguračnom súbore.
- Schopnosť ukladať škodlivé prílohy.
- Možnosť poškodenia obsahu nevyžiadanej pošty ako napríklad škodlivých príloh, elektronických odkazov alebo e-mailových adries, na ktoré môže príjemca poslať odpoveď z dôvodu zaistenia maximálnej bezpečnosti príjemcu.
- Inteligentný výber a možnosť rozsiahlej konfigurácie preposielanej pošty.

K automatickému nasadeniu honeypotu Hermes v rámci mojej bakalárskej práce je využité už existujúce riešenie, ktoré je jeho súčasťou. Pre správnosť fungovania a jeho celkového začlenenia do výsledného skriptu pre hromadné nasadzovanie honeypotov v sieti je nutná úprava modulov a interakcie s užívateľom. Vo veľkej miere sa však modul *Hermes* zhoduje s riešením navrhnutým ako súčasť honeypotu Hermes. Nasadenie honeypotu Hermes je štrukturované do častí, ktoré sa postupne starajú o konfiguráciu honeypotu na

[7] https://tools.ietf.org/html/rfc5321
[8] https://github.com/shiva-spampot/shiva

základe užívateľského vstupu, inštaláciu potrebných balíčkov, stiahnutie a inštaláciu zvolenej verzie honeypotu Hermes, inštaláciu nevyhnutných modulov pre správne fungovanie honeypotu, uloženie konfigurácie a v neposlednom rade o spustenie honeypotu. Keďže honeypot Hermes už bol navrhnutý aby správne fungoval v internej sieti firmy Avast, nie je potrebné ďalšie rozširovanie už existujúceho modulu pre odosielanie škodlivej elektronickej pošty spolu so škodlivými prílohami a metadátami o útokoch. Správy sú odosielané na vopred špecifikovanú tému (topic), ktorú si je užívateľ schopný nastaviť v konfiguračnom súbore `salmon.yaml`.

Kapitola 5

Implementácia

Obsahom tejto kapitoly je popis implementácie skriptu pre hromadné nasadenie honeypotov spolu s ich monitorovaním. V jednotlivých častiach bude popísaná architektúra modulov skriptu, konfigurácia skriptu, spôsob monitorovania serverov a posledná podkapitola bude venovaná konfigurácii jednotlivých honeypotov a jej prínosu pre zisk čo najlepších výsledkov.

5.1 Architektúra skriptu

Hlavný skript je možné rozdeliť do dvoch modulov. Prvý modul sa skladá z tzv. kuchárky, ktorej cieľom je popísať želaný výsledný stav systému. Úlohou tejto kuchárky po jej spustení je získať od užívateľa informácie o tom, ktoré honeypoty chce nasadiť a ktoré servery si želá monitorovať. Následne na základe užívateľského vstupu zvolí jednu z vopred predkonfigurovaných rolí, ktorá má za úlohu tento typ honeypotu nakonfigurovať a nasadiť. Ako som už spomínal v návrhu 4, jedná sa o tri role pre nasadenie honeypotu Dionaea, Cowrie a Hermes a dve role pre nasadenie centrálneho monitorovacieho systému a odosielateľov alebo tzv. exporterov na koncové stanice, ktorí slúžia k zasielaniu ľubovoľných metrík o nasadených serveroch do centrálneho monitorovacieho systému. Detailnejší popis modulu nasadenia honeypotov je v sekcii 5.2.

Druhý modul výsledného riešenia sa venuje vylepšeniu, respektíve aktualizácii už nasadeného honeypotu. Skladá sa z troch kuchárok pričom každá z nich má za úlohu nasadenie novšej verzie konkrétneho honeypotu a zachovanie užívateľských nastavení. Potrebné balíčky a návod na ich doinštalovanie je obsiahnutý v pribalenej dokumentácii `README.md`. Schéma skriptu pre nasadenie honeypov je uvedená v prílohe B. Bližší popis modulu aktualizácie honeypotov sa nachádza v sekcii 5.3

5.1.1 Prístup k serverom

Je nutné špecifikovať servery, na ktoré budú zvolené honeypoty nasadené (viz 4.3) a poskytnúť skriptu ich prihlasovacie údaje. Z hľadiska bezpečnosti som sa v práci rozhodol pre využitie technológie Ansible Vault[1], ktorá už je súčasťou použitého jazyka Ansible. Táto technológia slúži k bezpečnému uskladneniu citlivých informácii. Užívateľ špecifikuje prístupové údaje pre protokol SSH, ktorý Ansible využíva k prístupu k serverom do súboru `<host>.yml` umiestného v priečinku `host_vars` v koreňovom adresári skriptu, kde `host` je nahradené IP adresou servera. Jedná sa o predkonfigurovanú cestu k premenným pre

[1] https://docs.ansible.com/ansible/latest/user_guide/vault.html

jednotlivé servery, ktorá je prednastavená jazykom Ansible a je dobrým zvykom ju využívať. Obsah súboru `<host>.yml` pozostáva zo špecifikovania prístupových údajov ako meno a heslo pre prístup s dostatočným oprávnením. Odporúča sa prístup ako správca servera. V tomto súbore je možné špecifikovať aj iné premenné, ku ktorým je následne možné pristupovať priamo z Ansible skriptu pri práci s daným serverom. Príklad súboru `<host>.yml` je znázornený vo výpise 5.1.

```
ansible_user: root
ansible_ssh_pass: rootpass
ansible_become_pass: rootpass
```

Výpis 5.1: Príklad rozšifrovaného súboru `<host>.yml`

Pre zašifrovanie vytvoreného súboru pomocou technológie Ansible Vault je nutné vytvoriť súbor s heslom, ktorý následne predáme ako argument prepínača `-vault-password-file`. Z hľadiska bezpečnosti by sa nemalo jednať o jednoduché užívateľské heslo. Rozhodol som sa pre vytvorenie hesla pozostávajúceho z dlhého a predovšetkým náhodného reťazca zakódovaného pomocou kódovania base64 [12]. Pre vytvorenie base64 reťazca je možné použiť príkaz `openssl rand -base64 2048 > <ansible heslo>.pass`[2]. Celý príkaz pre zašifrovanie súboru s prístupovými údajmi k serveru `1.1.1.1` pomocou súboru s názvom `ansible-vault.pass` obsahujúceho potrebné prístupové údaje by vyzeral ako:

```
ansible-vault encrypt --vault-password-file ansible-vault.pass ./host_vars
    /1.1.1.1.yml
```

Obsah súboru je možné následne zobraziť pomocou argumentu `view`, prípadne spätne rozšifrovať pomocou argumentu `decrypt`. V oboch prípadoch je takisto potrebné predať príkazu cestu k súboru s heslom použitým k zašifrovaniu. V priečinku `host_vars` je možné uchovať ľubovoľné množstvo konfiguračných súborov pre prístup k rôznym serverom zašifrovaných pomocou jediného Ansible Vault hesla.

5.2 Modul nasadenia honeypotov

Nasadenie honeypotov je vykonané pomocou kuchárky `hp_deployment.yml`. Po spustení tejto kuchárky pomocou príkazu

```
ansible-playbook hp_deployment.yml --vault-password-file=ansible-vault.
    pass
```

si užívateľ prostredníctvom interaktívneho vstupu zvolí, ktoré honeypoty si želá nasadiť. Následne na základe vstupu užívateľa skrip spustí príslušné role, ktoré sa starajú už o nasadenie konkrétneho druhu honeypotu, prípadne nasadenie monitorovacieho systému. Pri nasadzovaní honeypotov Dionaea a Hermes je vyžadovaný ďalší dodatočný interaktívny vstup, ktorý umožní zmenu údajov v konfigurácii daného typu honeypotu. Konfigurácia jednotlivých honeypotov pomocou užívateľského vstupu, ale aj vopred predkonfigurované nastavenia pre zaistenie unikátnosti a znemožnenie odhalenia honeypotu sú bližšie popísané v sekcii 5.2.3.

[2] https://gist.github.com/hvanderlaan/ae5d7f62d42c927fdad42309d25c9693

5.2.1 Role pre nasadenie honeypotov

Role slúžiace k nasadeniu konkrétnych druhov honeypotov sú umiestnené v priečinku `roles` a štruktúrované tak, že každá rola sa stará o nasadenie jedného druhu honeypotu a monitorovacieho systému. Každá rola obsahuje zoznam úloh, ktoré sa musia splniť, respektíve doviesť systém do požadovaného stavu. Tieto úlohy sú umiestnené v súbore `roles/<názov role>/tasks/main.yml` a tvoria telo celého programu. Pri každom honeypote sú mierne odlišnosti vyplývajúce z rôzneho postupu nasadzovania, rôznych počiatočných predpokladov a balíčkov nutných k inštalácii a predovšetkých rôznej konfigurácie samotného honeypotu.

Všeobecná kostra pre nasadenie ľubovoľného honeypotu je popísaná pomocou role `generic_deployment`, ktorá neslúži k spusteniu, ale k akejsi pomyselnej inšpirácii pri tvorbe nových rolí a rozširovaní skriptu v budúcnosti. Pre popis toho, ako vo všeobecnosti funguje nasadenie honeypotu teda použijem túto rolu. Úlohy nutné k nasadeniu honeypotu je možné rozdeliť do 5 pomyselných častí, pričom každá z nich sa venuje jednej všeobecnej úlohe pri nasadzovaní honeypotov.

1. **Získanie užívateľského vstupu slúžiaceho k dodatočnej konfigurácii honeypotu:** Voliteľná časť, ktorá slúži k získaniu vstupu od užívateľa pomocou zaregistrovania odpovede zadanej na štandardný vstup na vopred špecifikovanú otázku. Následne je táto hodnota priradená do premennej unikátnej pre každý server, na ktorý je honeypot nasadzovaný. Sme teda schopní dosiahnuť unikátnu konfiguráciu viacerých serverov.

2. **Aktualizácia a nainštalovanie všetkých potrebných balíčkov, vytvorenie systémového užívateľa a skupiny pre spustenie a konfiguráciu honeypotu:** Časť, ktorá je takmer totožná pre ľubovoľný honeypot a líši sa jedine v zozname balíčkov nutných pre správne fungovanie honeypotu. Tento zoznam je nutné získať z oficálnej dokumentácie zvoleného honeypotu a následne doplniť do skriptu do časti inštalácie balíčkov znázornenej vo výpise 5.2.

```
- name: Install missing packages
  package:
    name: "{{ item }}"
  with_items:
    - !package
    - !package
```

Výpis 5.2: Inštalácia balíčkov vo všeobecnej kostre pre nasadenie honeypotov

   Je nutné nahradiť `!package` za požadovaný balíček. Poslednou úlohou tejto časti je vytvorenie užívateľa a skupiny, pod ktorou bude honeypot bežať. Zamedzíme tak administrátorskému prístupu k honeypotu a tým pádom zvýšime bezpečnosť nášho riešenia.

3. **Stiahnutie špecifikovanej verzie honeypotu a jej nainštalovanie:** Povinná časť, ktorá je takisto takmer rovnaká pri všetkých druhoch honeypotov a skladá sa spravidla zo stiahnutia honeypotu. Jedná sa prevažne o naklonovanie zdrojového kódu z verejného repozitára na stránke Github[3]. Tento repozitár by mal vo väčšine prípadov obsahovať aj inštrukcie k nainštalovaniu dokumentácie riešenia a zoznam potrebných balíčkov nutných k vykonaniu druhého kroku. Po stiahnutí honeypotu sa vykoná

[3] https://github.com

jeho inštalácia, ktorá je však už špecifická pre každý druh. A táto kostra teda slúži k ukázaniu len jedného prístupu pozostávajúceho z vytvorenia adresára, preloženiu zdrojového kódu a inštalácie honeypotu a nastavenia príslušných oprávnení novovzniknutého súboru s honeypotom pre užívateľa definovaného v kroku 2. Pokiaľ je potrebné doinštalovať balíčky do virtuálneho prostredia jazyka Python, kostra obsahuje ukážkový príkaz, ako takéto prostredie vytvoriť a akým spôsobom je možné doinštalovať potrebné balíčky.

4. **Nastavenie nainštalovaného honeypotu:** Voliteľný krok pozostávajúci z úpravy konfiguračných súborov na základe vstupu v počiatočnom kroku a nahradenie konfiguračných súborov pomocou vopred definovaných šablón obsahujúcich predkonfigurované hodnoty líšiace sa od pôvodných s cieľom zaistiť unikátnosť honeypotu. Šablóny sú umiestnené v priečinku `templates` a je využitý šablónovací jazyk Jinja[4]. Po presunutí šablóny je možné ju aj upraviť pomocou vstavaného modulu jazyka Ansible – `lineinfile`[5], ktorý umožňuje pridať, odstrániť alebo aj upraviť riadok v súbore na vzdialenom serveri.

5. **Spustenie honeypotu a nastaveniu sieťových pravidiel pre obmedzenie nežiadúceho vstupu na porty, ktoré nie sú využívané honeypotom:** Dôležitým krokom je vytvorenie systémovej služby, ktorá umožní jednoduché zapnutie honeypotu pod zvoleným užívateľom a povolenie behu honeypotu po štarte systému. Je nutné vytvoriť šablónu pre túto službu a následne ju presunúť na cieľový server. Príklad takejto šablóny je umiestnený v súbore `templates/honeypot.service.j2`. Po vykonaní tohto kroku sú upozornené služby tzv. handlery, ktoré načúvajú a reagujú na notifikácie pomocou nasledujúceho príkazu:

```
notify:
  - reload <name>
  - restart <honeypot>
```

Táto notifikácia upozorní službu s názvom `<name>`, ktorá vykoná definované úkony. Tieto služby sú definované v súbore `handlers/main.yml` a prevažne sa jedná o interakciu so systémovými službami. Následným krokom po spustení služby starajúcej sa o beh honeypotu je možné, ak to užívateľ vyžaduje, nastaviť pravidlá pre obmedzenie sieťového prenosu na zvolenom porte. Napríklad, ak nám honeypot beží na porte 80, ktorý predstavuje protokol HTTP[6] a chceme zablokovať prístup na port 22, ktorý predstavuje protokol SSH mimo našu IP adresu použijeme program `iptables`[7], konkrétne príkaz:

```
iptables -C INPUT \! --src 1.1.1.1 -m tcp -p tcp --dport 22 -j
    DROP
```

K spusteniu tohto príkazu na vzdialenom serveri je možné využiť vstavaný modul command[8].

[4] https://jinja.palletsprojects.com/en/2.11.x/
[5] https://docs.ansible.com/ansible/latest/collections/ansible/builtin/lineinfile_module.html
[6] https://tools.ietf.org/html/rfc2616
[7] https://linux.die.net/man/8/iptables
[8] https://docs.ansible.com/ansible/latest/collections/ansible/builtin/command_module.html

Vyššie spomenuté kroky k všeobecnému nasadeniu ľubovolného druhu honeypotu, je nutné rozšíriť, prípadne mierne upraviť na základe požiadaviek nasadzovaného honeypotu obsiahnutých v oficiálnej dokumentácii. Premenné použité počas behu skriptu je možné špecifikovať v súbore `/vars/main.yml` vo formáte `name: value`. Napríklad:

```
version: master
```

Následne v rámci tejto role, ľubovoľne v kóde alebo v šablónach vieme použiť referenciu na túto premennú v tvare `"{{version}}"`, namiesto ktorej bude doplnená hodnota `master`. Príkladom použitia môže byť možnosť jednoduchej zmeny požadovanej verzie nasadzovaného honeypotu bez nutnosti zásahu do samotného kódu. Bližší popis pre nasadenie konkrétnych druhov honeypotov (Dionaea, Cowrie, Hermes) je spolu s popisom ich konfigurácie spomenutý v kapitole 5.2.3 alebo priamo v zdrojovom kóde.

5.2.2 Role pre nasadenie monitorovania

Spolu s nasadením honeypotov je užívateľ schopný aj nasadiť monitorovací systém, ktorý sa stará o zber a kontrolu špecifikovaných metrík. Architektúra monitorovacieho systému pozostáva z centrálneho servera, ktorého úlohou je pravidelný zber dát z koncových staníc tzv. odosielateľov. Celková architektúra je bližšie popísaná a znázornená v sekcii 3.3.

Spôsob nasadenia monitorovacieho systému pozostáva z časti nasadenia centrálneho servera a nasadenia odosielateľov na koncové stanice. V predvolených nastaveniach sa monitorovací server nasadí na počítač, z ktorého je skript pre nasadenie spustený. Užívateľ je však schopný toto nastavenie zmeniť v súbore `inventory` pod značkou `[prometheus]`. Servery, na ktoré budú nasadení odosielatelia je možné špecifikovať pod značkou `[exporters]`.

Nasadenie centrálneho Prometheus servera

K nasadeniu centrálneho Prometheus servera je využitá rola `prometheus` pozostávajúca z nasadenia samotného Prometheus servera, nasadenia nástroja k upozorňovaniu užívateľa - *AlertManager* a nástroja pre spúšťanie externých skriptov pri vyvolaní upozornenia - *prometheus-am-executor*. Verzie týchto nástrojov je možné špecifikovať v súbore `/vars/main.yml`

Súčasťou nasadenia Prometheus servera je jeho stiahnutie a konfigurácia, ktoré pozostáva z nahradenia súborov `prometheus.yml` a `alert_rules.yml`.

1. **`prometheus.yml`** – Pozostáva z nastavenia hodnôt časového intervalu zberu dát, špecifikácie servera a portu, na ktorom poslúcha `AlertManager`, nastavenia cesty k súboru s pravidlami upozornení (`alert_rules.yml`) a špecifikácie serverov s odosielateľmi metrík.

   Nastavenie intervalu zberu dát z koncových staníc je možné špecifikovať pomocou nastavenia hodnoty premennej `scrape_interval` a nastavenie intervalu vyhodnotenia pravidiel pre upozorňovanie je možné pomocou premennej `evaluation_interval`. Obe hodnoty sú prednastavené na pätnásť sekúnd.

   K upozorňovaniu je využitý komponent *AlertManager*. Potrebné je špecifikovať server a port, na ktorom tento komponent beží. V predvolených nastaveniach je využitý lokálny počítač, z ktorého bol skript spustený a predvolený port - `9093`. K definícii pravidiel pre upozorňovanie je využitý súbor `alert_rules.yml` popísaný v sekcii 2.

   Poslednou dôležitou časťou konfiguračného súboru `prometheus.yml` je definícia serverov, z ktorých je potrebné zbierať požadované metriky. Súčasťou definície týchto

serverov je aj ich pomenovanie z dôvodu ich ľahšej identifikácie. Zber metrík je rozdelený na zber systémových metrík a zber informácii o bežiacom procese. Obe časti sú bližšie popísané v sekcii 5.2.2. Je možné špecifikovať ľubovoľné množstvo serverov, pričom každý z nich by mal obsahovať popisné meno servera (napríklad doménové meno) a IP adresu servera. Odosielateľ metrík v predvolených nastaveniach beží na porte 9100 a odosielateľ informácii o procesoch na porte 9256. Vo výpise 5.3 je znázornený príklad definície dvoch serverov z ktorých budú zberané obidva druhy metrík.

```
scrape_configs:
  - job_name: 'node_exporter'
    static_configs:
    - targets: ['1.1.1.1:9100']
      labels:
        instance: 'First-server'
        ip: '1.1.1.1'
    - targets: ['2.2.2.2:9100']
      labels:
        instance: 'Second-server'
        ip: '2.2.2.2'
  - job_name: 'process_exporter'
    - targets: ['1.1.1.1:9256']
      labels:
        instance: 'First-server'
        ip: '1.1.1.1'
    - targets: ['2.2.2.2:9256']
      labels:
        instance: 'Second-server'
        ip: '2.2.2.2'
```

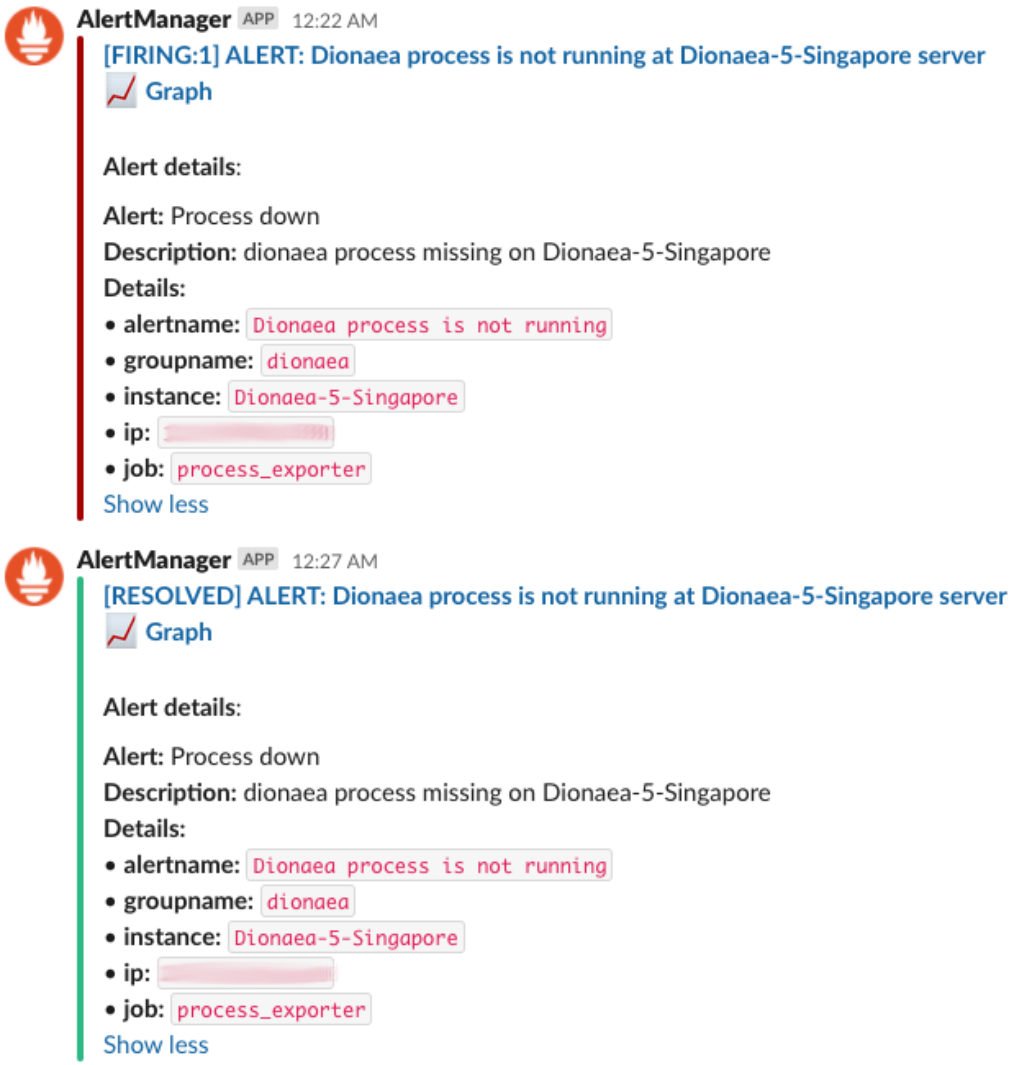
Výpis 5.3: Príklad definície monitorovaných serverov v súbore `prometheus.yml`

2. **`alert_rules.yml`** – Súbor, ktorý obsahuje zoznam pravidiel, ktorých nedodržanie vedie k upozorneniu užívateľa. Súbor obsahuje päť základných pravidiel, ktoré je možné ľubovoľne zmeniť alebo rozšíriť (viz 4.4). Každé pravidlo by malo obsahovať jeho unikátny názov, podmienku, ktorej nesplnenie vedie k vyvolaniu upozornenia, doba po ktorú dané pravidlo nesmie platiť, aby došlo k vyvolaniu upozornenia a zoznam voliteľných anotácii bližšie špecifikujúcich upozornenie. Zoznam anotácii by mal vždy obsahovať IP adresu stroja, na ktorom došlo k porušeniu pravidla, ktorá je prístupná pomocou premennej `{{$labels.ip}}` špecifikovanej v časti 1, jednoznačný názov upozornenia a jeho krátky popis. Tieto anotácie sú následne zobrazené užívateľovi pri vyvolaní upozornenia.

Konfigurácia komponentu `AlertManager` pozostáva z jeho stiahnutia a následného nahradenia konfiguračného súboru `alertmanager.yml`, ktorého úlohou je špecifikácia koncových aplikácii (ďalej ako receiver), na ktorých je užívateľ upozornený po vyvolaní príslušného upozornenia. V predvolených nastaveniach je implementované upozorňovanie pomocou aplikácie *Slack* pri porušení každého pravidla priamo do kanála `#p-hp-monitoring`, ktorý musí byť vopred vytvorený. Ďalším z implementovaných spôsobov upozornenia užívateľa je využitie aplikácie *Splunk On-Call*, predtým známej ako *VictorOps*[9] pri porušení kritic-

[9] https://splunk.com/en_us/investor-relations/acquisitions/splunk-on-call.html

kých pravidiel. Aplikácia *Splunk On-Call* slúži k upozorneniu užívateľa prostredníctvom SMS správy alebo telefonátu na základe jeho nastavení na danej platforme. Upozorňovanie prostredníctvom tejto aplikácie teda prebieha len v kritických situáciach a to napríklad pri výpadku produkčného servera. Posledným zo spôsobov upozorňovania je napojenie na komponent `prometheus-am-executor`, ktorý slúži k spusteniu určitého príkazu alebo skriptu pri obdržaní upozornenia. Vzhľad upozornení v aplikácii Slack bol upravený a rozšírený o graf monitorovanej metriky alebo o graf procesu, ktorý vyvolal príslušné upozornenie a krátky popis vyvolaného upozornenia. K vytváraniu vzhľadu upozornenia a testovania som využil nástroj *Alertmanager Slack Playground*[10]. Vzhľad upozornenia je definovaný v súbore `templates/slack.tmpl` a jeho výsledná podoba v aplikácii Slack je znázornená na obrázku 5.1.



Obr. 5.1: Vzhľad upozornenia v aplikácii Slack pre výskyt aj vyriešenie problému

[10] https://juliusv.com/promslack/

Každý receiver musí mať špecifikovanú cestu, kam sa má dané upozornenie presmerovať. Pri aplikácii *Slack* je to adresa rozhrania API, ktorú je možné získať po vytvorení kanálu a nainštalovaní potrebných rozšírení[11]. Upozornenie prostredníctvom aplikácie *Splunk On-Call* vyžaduje API kľúč spolu s kľúčom pre nasmerovanie upozornenia do konkrétnej skupiny tzv `routing_key`[12]. Pre presmerovanie upozornenia do komponentu `prometheus-am-executor` stačí nastaviť IP adresu a port, na ktorom komponent beží. Príklad nastavenia presmerovania všetkých upozornení na *Slack* je znázornený vo výpise 5.4.

```
route:
  group_by: [...]
  receiver: slack_hp_monitoring

  -
    receiver: slack_hp_monitoring
    continue: true

  receivers:
  - name: 'slack_hp_monitoring'
    slack_configs:
    - api_url: <API URL>
    channel: '#p-hp-monitoring'
    icon_url: https://avatars3.githubusercontent.com/u/3380462
    send_resolved: true
    title: '{{ template "custom_title" . }}'
    text: '{{ template "custom_slack_message" . }}'

templates:
- ./slack.tmpl
```

Výpis 5.4: Príklad nastavenia súboru `alertmanager.yml`

Nasadenie komponentu *prometheus-am-executor* sa skladá zo stiahnutia príslušnej verzie, preložení zdrojového kódu pomocou jazyka `go`[13] a nastavenia tohoto komponentu pomocou súboru `executor.yml`. Súbor `executor.yml` znázornený vo výpise 5.5 obsahuje číslo portu, na ktorom bude *prometheus-am-executor* počúvať a reagovať na vzniknuté upozornenia a zoznam príkazov ktoré budu vykonané.

```
listen_address: ":8080"
verbose: false
commands:
  - cmd: ./start_dionaea.sh
    match_labels:
      "groupname": "dionaea"
```

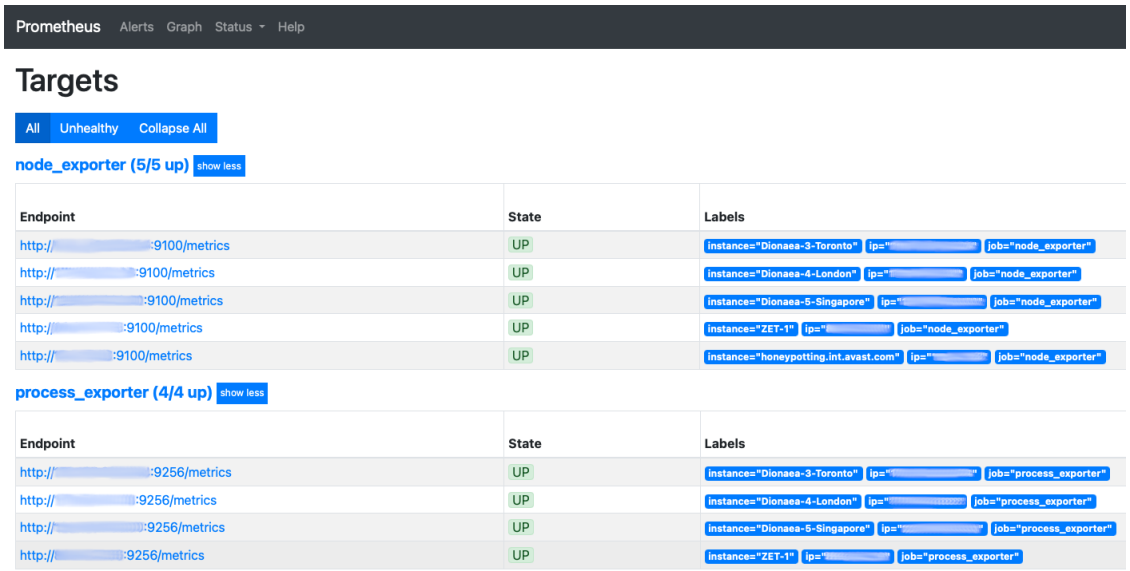
Výpis 5.5: Príklad nastavenia súboru `executor.yml`

V aktuálnej implementácii sa jedná o spustenie skriptu `start_dionaea.sh`, ktorého úlohou je obnoviť process `dionaea` po výpadku na postihnutom serveri. Pre implementáciu

[11] https://grafana.com/blog/2020/02/25/step-by-step-guide-to-setting-up-prometheus-alertmanager-with-slack-pagerduty-and-gmail/
[12] https://help.victorops.com/knowledge-base/victorops-prometheus-integration/
[13] https://golang.org

automatického obnovenia procesu `dionaea` som sa rozhodol vzhľadom na nižšiu stabilitu novšej verzie Dionaea honeypotu. K výpadkom procesu dochádzalo priemerne 1-2x za mesiac. Proces je reštartovaný automaticky a nie je nutný zásah užívateľa. Pri písaní tohto skriptu a nastavovaní komponentu `prometheus-am-executor` som sa inšpiroval oficiálnou dokumentáciou a poskytnutými príkladmi spomenutého nástroja[14].

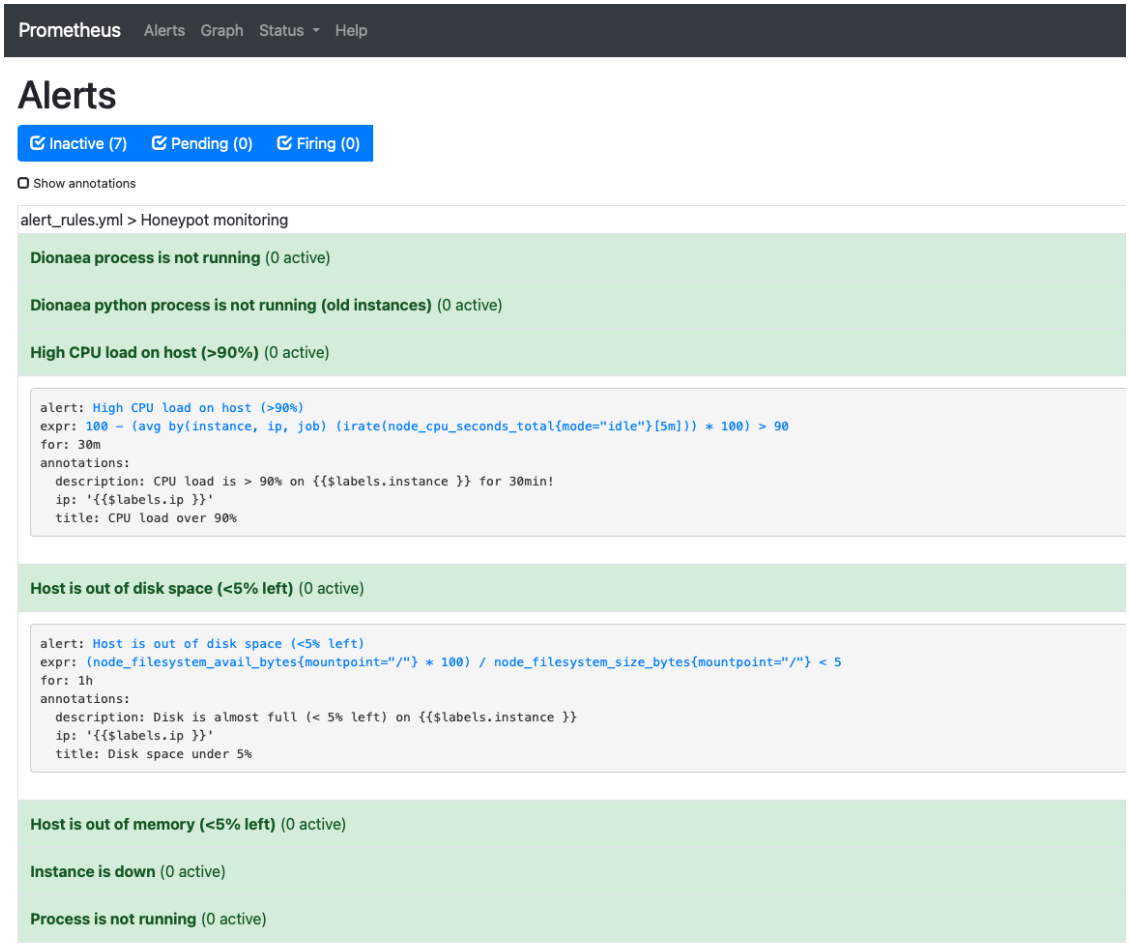
Popri nasadzovaní centrálneho monitorovacieho servera stiahneme a nasadíme aj odosielateľa metrík na tento stroj, aby užívateľ bol upozornený aj na prípadné výpadky centrálneho Prometheus servera. Po nainštalovaní a konfigurácii všetkých komponentov dôjde k ich zapnutiu.



Obr. 5.2: Webové rozhranie aplikácie Prometheus

Zoznam monitorovaných serverov a ich aktuálny stav je možné vidieť vo webovom rozhraní aplikácie Prometheus, ktorá okrem zoznamu serverov obsahuje aj zoznam pravidiel. Vzhľad webového rozhrania je znázornený na obrázku 5.2 a zoznam pravidiel na obrázku 5.3.

[14] https://github.com/imgix/prometheus-am-executor/tree/master/examples



Obr. 5.3: Zoznam užívateľských pravidiel pre monitorovanie

Nasadenie odosielateľov

Nasadenie odosielateľov je rozdelené do dvoch rolí. Rola `prometheus_node_exporter` nasadí odosielateľa metrík a rola `prometheus_process_exporter` nasadí odosielateľa informácií o procesoch na príslušných serveroch. Obe role sú spustené súčasne v prípade kladnej odpovede na otázku, či si užívateľ želá nasadiť monitorovanie na špecifikovaných serveroch umiestnených pod značkou `[exporters]` v súbore `inventory`. Obe role sú takmer identické a líšia sa len v tom, že rola `prometheus_process_exporter` obsahuje konfiguračný súbor, ktorý je špecifický pre servery, na ktorých beží rovnaký typ honeypotu. Nasadenie odosielateľov sa skladá z viacerých pomyselných častí:

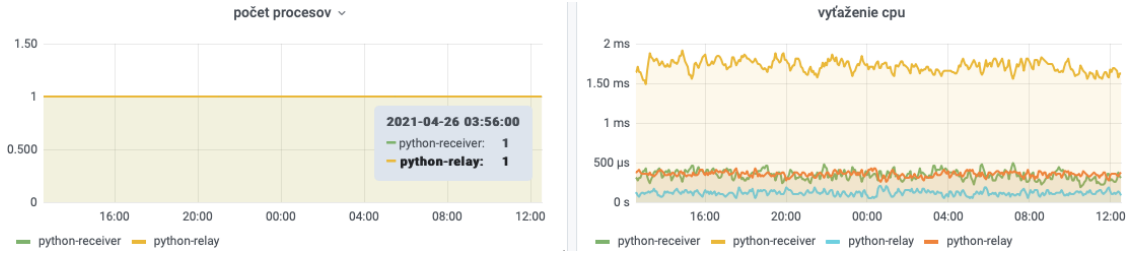
1. Vytvorenie užívateľa a skupiny, pod ktorou bude daný odosielateľ bežať.
2. Stiahnutie príslušnej verzie odosielateľa.
3. Kontrola aký typ honeypotu beží na danom serveri a následné prekopírovanie príslušnej konfigurácie - krok špecifický pre rolu `prometheus_process_exporter`.

4. Vytvorenie systémovej služby, ktorá sa bude starať o správny beh odosielateľa a jej spustenie.

5. Zablokovanie prístupu na porty, ktoré poskytujú zberané metriky alebo procesy mimo internej sieti firmy Avast.

Konfiguračný súbor odosielateľa `prometheus_process_exporter` obsahuje názov procesov, ktoré sú špecifické pre daný honeypot a informujú užívateľa, že honeypot nebeží, alebo nebeží jedna zo služieb spojených so zberom dát. Procesy je možné identifikovať a monitorovať prostredníctvom názvu ich spustiteľného súboru (`comm`) alebo časti príkazu použitého k ich spusteniu (`cmdline`). Nástroj `process-exporter` podporuje aj iné možnosti identifikácie procesu[15], ktoré neboli použité v tejto bakalárskej práci.

K vizualizácii monitorovaných metrík a procesov je využitý nástroj *Grafana*. Dáta z centrálneho Prometheus servera sú pridané ako nový zdroj dát a k vizualizácii sú využité mierne upravené komunitné riešenia pre odosielateľa metrík[16] a odosielateľa procesov[17]. Vizualizácia bežiacich Hermes procesov na serveri je znázornená na obrázku 5.4.



Obr. 5.4: Vizualizácia monitorovania procesov honeypotu Hermes pomocou nástroja Grafana

5.2.3 Konfigurácia honeypotov

Výhodou tejto práce je okrem jednoduchého nasadenia veľkého množstva honeypotov aj ich predkonfigurovanie, ktoré sťaží útočníkovi možnosť zistiť, že sa nenachádza na reálnom systéme. Konfigurácia honeypotov bola prispôsobená pre nasadzovanie viacerých inštancií honeypotov zároveň. V nasledujúcich podkapitolách budú zhrnuté vlastnosti konfigurácie jednotlivých honeypotov a výhody s nimi spojené.

Dionaea

Pri spustení skriptu pre automatické nasadenie (viz 5.2) a zvolení nasadenia Dionaea honeypotov na príslušných serveroch je na začiatku skriptu umožnené užívateľovi zvoliť názov doménového mena honeypotu a názov zdieľaného disku, ktoré budú viditeľné útočníkovi po napojení sa na honeypot. Táto konfigurácia je unikátna pre každú inštanciu honeypotu a užívateľ je schopný ju nastaviť pre každý honeypot zvlášť. Honeypoty Dionaea sú nasadené s podporou protokolu SMB s úlohou zaistenia potenciálne škodlivých vzoriek

[15] https://github.com/ncabatoff/process-exporter/blob/master/README.md
[16] https://grafana.com/grafana/dashboards/1860
[17] https://grafana.com/grafana/dashboards/249

získaných počas útoku. Konfiguračný súbor `dionaea.cfg` je pozmenený počas inštalácie, aby umožnil odposluch na sieti, k čomu bola využitá globálna premenná jazyka Ansible `ansible_default_ipv4.address` umožňujúca dynamické vloženie IP adresy servera, na ktorom honeypot Dionaea beží. Spolu s nastavením IP adresy odposluchu sa nastaví aj podporované rozhranie Ethernet a povolia sa moduly `python`, `curl`, `nfq`, `emu` a `pcap`. Namiesto ukladania všetkých správ bolo zvolené ukladanie len výstražných a chybových správ do textového súboru unikátneho pre každý deň. Súbory s uloženými správami sú dostupné po dobu siedmich dní a následne sú vymazané. Súbory obsahujúce škodlivé vzorky sú komprimované pokiaľ sú staršie viac ako hodinu a odstránené, pokiaľ sú staršie viac ako šesť hodín[18]. Dôležitou súčasťou honeypotu Dionaea je zber škodlivých vzoriek. Pre účel pravidelného odosielania novovzniknutých vzoriek sú využité tri mierne upravené interné skripty firmy Avast:

- `binaries_listener.py` – Skript slúžiaci k odoslaniu novovzniknutého škodlivého súboru pomocou protokolu MQTT. Téma, na ktorú sa súbor odošle je vypočítaná pomocou zahashovania doménového mena servera za využitia hashovacej funkcie `sha1`. Posledné tri číslice vzniknutého reťazca sú MQTT téma.
- `notify-bistreams.py` – Skript slúžiaci k upozorneniu pomocou MQTT na vznik nového útoku a odoslaniu metadát o danom spojení.
- `meta-notify-bistreams.py` – Podporný skript pre skript `notify-bistreams.py` slúžiaci k upozorneniu na vznik nového priečinku obsahujúceho informácie o spojeniach pre daný deň.

Prístup k službe SSH po úspešnom nasadení honeypotu je zablokovaný mimo internú sieť firmy Avast. Honeypot emuluje službu SMB na porte 445, nie je teda potrebné otvorenie ďalších portov pre potenciálnych útočníkov.

Cowrie

Pri zvolení nasadenia honeypotu Cowrie je honeypot nasadený bez ďalšej nutnej konfigurácie zo strany užívateľa. Je nasadená rovnaká verzia honeypotu na všetkých zvolených serveroch vzhľadom na nutnosť vygenerovania celého súborového systému, čo by bolo počas nasadzovania obtiažne. Všetky nasadené inštancie sa však budú líšiť od tých nasadených inými autormi a teda budú unikátne a ťažko odhaliteľné voči ostatným na internete.

Honeypot Cowrie je predkonfigurovaný, aby podporoval prístup pomocou protokolu SSH a Telnet. Prístup je umožnený útočníkovi po zadaní aspoň dvoch a maximálne piatich prístupových údajov. Údaje k prihláseniu sú vyberané náhodne a ich kombinácia pre danú IP adresu útočníka je uložená. Využíva sa trieda `AuthRandom`. Súborový systém emulovaný honeypotom je Linux, konkrétne distribúcia Debian 10, ktorý bol predgenerovaný. Súborový systém obsahuje užívateľa Daan. Spolu s vytvoreným súborovým systémom bol aj upravený konfiguračný súbor, aby reflektoval vzniknutého užívateľa. Upravené sú aj reťazce, pomocou ktorých je možné odhaliť, že daný systém je honeypot Cowrie. Predvolené verzie reťazca popisujúceho verziu SSH, verziu jadra operačného systému a mnohé iné sú zmenené a nahradené hodnotami z reálneho servera.

[18] https://web.archive.org/web/20180818104322/https://kangaroosecurity.com/setting-up-a-dionaea-honeypot/

Výhodou honeypotu Cowrie je možnosť sprístupnenia niektorých súborov útočníkovi a možnosť simulovania príkazov. Nasadzovaná verzia honeypotu Cowrie upravuje napríklad súbory ako:

- `apt` – simulovanie príkazu `apt`. Cowrie v pôvodnej verzii podporuje len príkaz apt-get, ktorý je nástupcom príkazu `apt`. Príkaz `apt` vypíše vlastnú chybovú hlášku, ktorá napovie útočníkovi, že pravdepodobne myslel príkaz `apt-get`.
- `cmdoutput.json` – zoznam bežiacich procesov upravený, aby hodnoty zabranej pamäte, doby spustenia alebo aj dĺžky behu neboli pôvodné. Výstup tohto súboru je často porovnávaný s predvolenými hodnotami pri snahe útočníka odhaliť, že sa nachádza na honeypote.
- `cpuinfo` – informácie o procesore na danom serveri boli upravené aby reflektovali reálny stroj a líšili sa od pôvodných hodnôt.
- `fs.pickle` – súbor obsahujúci súborový systém emulovaný honeypotom.
- `motd` – vlastná uvítacia hláška zobrazená útočníkovi po prihlásení sa na honeypot.
- `passwd`, `group`, `shadow` a `hostname` – súbory z vygenerovaného súborového systému s užívateľom Daan reflektujúce všetkých užívateľov a procesy. Jedným z častých trikov ako odhaliť honeypot Cowrie je preskúmanie napríklad súboru `passwd` s cieľom nájsť užívateľa `phil`. Nasadzovaná verzia honeypotu neobsahuje žiadnych z pôvodných užívateľov.
- `meminfo` – súbor obsahujúci aktuálne využitie pamäti RAM reálneho stroja.

Súčasťou konfigurácie honeypotu Cowrie je nainštalovanie balíčku `iottl.zip` vyvinutého firmou Avast, ktorý slúži na spojenie sa s ich interným serverom k prenosu správ pomocou protokolu MQTT. Honeypot Cowrie v pôvodnej verzii nepodporuje odosielanie metadát prostredníctom protokolu MQTT. Rozhodol som sa teda pre úpravu už implementovaného modulu `output/hpfeeds` a nahradiť odosielanie na hpfeeds server[19] odosielaním na MQTT server na téma `honeyfeed/cowrie`. Ide o jednoduché roztriedenie súboru s údajmi o spojeniach a získanie potrebných informácii ako URL adresy slúžiace k stiahnutiu súboru, prihlasovacie údaje útočníka, IP adresa útočníka, čas pripojenia a dĺžka trvania a mnohé iné. Stiahnutie škodlivého binárneho súboru z danej url adresy je vzhľadom na bezpečnosť riešené až v internej sieti firmy Avast nezávisle na tomto skripte.

Dôležitou časťou nasadzovania honeypotu Cowrie je zmena portu pre SSH pripojenie na port **35535** a presmerovanie spojení z portu **22** na port **2222** pre SSH respektíve portu **23** na port **2223** pre Telnet, na ktorých beží honeypot a sú zaznamenávané útoky.

Hermes

Honeypot Hermes je nasadený v pôvodnej konfigurácii a bez akýchkoľvek zmien, ktoré by mali vplyv na jeho fungovanie. Súčasťou nasadzovania honeypotu Hermes je možnosť špecifikovať jeho konfiguráciu prostredníctvom interaktívneho užívateľského vstupu. Nastavenie jednotlivých údajov je podrobne vysvetlené v dokumentácii[20]. Jedinou zmenou pôvodnej

[19] https://hpfeeds.org
[20] https://github.com/avast/hermes

konfigurácie bolo premenovanie bežiacich procesov k zjednodušeniu ich monitorovania a úprava štýlu Ansible príkazov aby reflektovali koncept využitý aj pri iných roliach a kuchárkach skriptu pre hromadné nasadzovanie honeypotov.

5.3 Modul aktualizácie honeypotov

Ako som už spomínal v kapitole 5 , modul aktualizácie honeypotu je zložený z troch kuchárriek, pričom každá z nich sa stará o aktualizovanie honeypotov jedného typu. Jedným zo spôsobov ako byť pred útočníkom a zvýšiť šance, že nasadený honeypot nebude odhalený, je mať nasadenú čo najaktuálnejšiu verziu. Všetky typy honeypotov nasadzované v rámci tejto bakalárskej práce sú aktívne vyvíjané a prispôsobované zachytávať aktuálne trendy vo svete útokov. Súčasťou mojej implementácie je aj udržiavanie práve najnovších verzií honeypotov.

Kuchárky pre aktualizovanie honeypotov sa nachádzajú v priečinku `upgrade` a slúžia k nasadeniu vždy najnovšej dostupnej verzie honeypotu. Kuchárky sa nespúšťajú pomocou hlasného skriptu `hp_deployment`, ale je potrebné spustiť ich samostatne pomocou príkazu

```
ansible-playbook upgrade/<meno-honeypotu> --vault-password-file=<ansible
    heslo>.pass
```

a nahradiť `<meno-honeypotu>` za názov honeypotu (**cowrie**, **dionaea** alebo **hermes**), ktorý bude aktualizovaný na špecifikovaných serveroch. Rovnako ako skript pre nasadenie honeypotu aj skript pre jeho aktualizáciu vykoná všetky definované kroky vedúce k aktualizácii honeypotu na serveroch špecifikovaných pod príslušnou značkou honeypotu (viz 4.3). Všetky tri skripty fungujú na rovnakom princípe, ktorý je možné rozdeliť do štyroch častí:

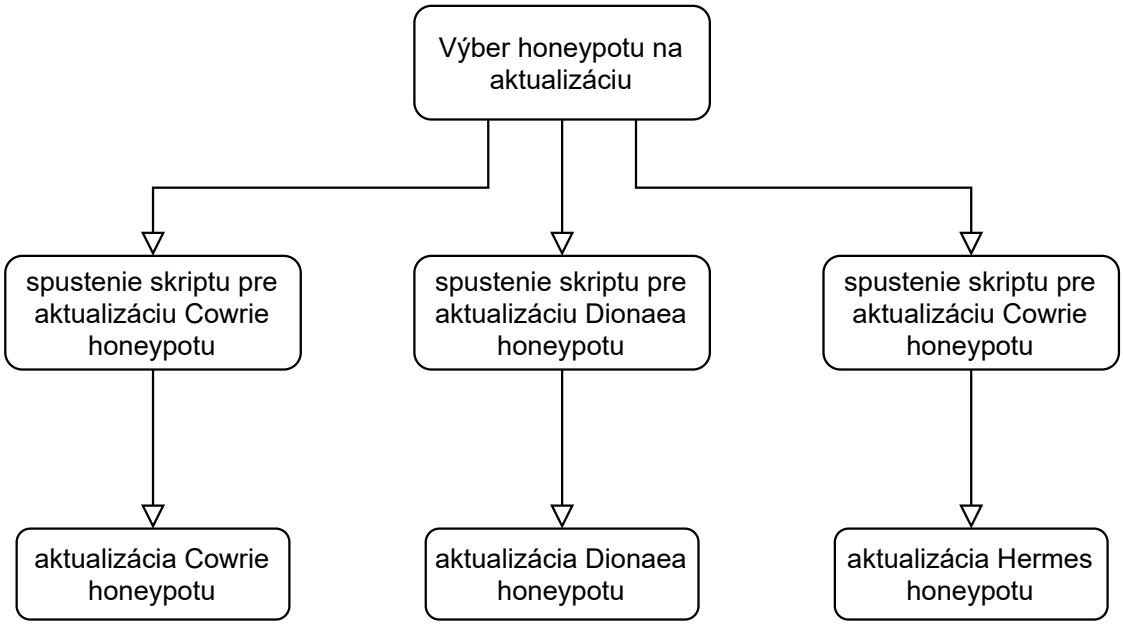
1. **Zastavenie behu honeypotu a uloženie pôvodnej konfigurácie** – krok dôležitý k zachovaniu špecifickej konfigurácie honeypotu. Nová verzia honeypotu je nainštalovaná bez nutnosti zásahu užívateľa a všetka pôvodná konfigurácia je zachovaná.

2. **Stiahnutie a inštalácia novej verzie honeypotu** – dôjde k stiahnutí najnovšej verzie honeypotu, jeho inštalácii a k prípadnému doinštalovaniu dodatočných balíčkov.

3. **Nahradenie pôvodnej konfigurácie za zálohovanú** – vrátenie zálohovaných konfiguračných súborov naspäť a prepísanie pôvodnej konfigurácie v novej verzii honeypotu.

4. **Zmazanie dočasnej zálohy a spustenie honeypotu** – odstránenie vzniknutých záložných súborov s konfiguráciou a spustenie novej verzie honeypotu. Ak sa jedná o honeypot Dionaea dôjde k spusteniu a zálohe aj podporných skriptov pre zber dát a metadát z honeypotu.

Schéma znázorňujúca proces aktualizácie nasadených honeypotov je znázornená na obrázku 5.5.



Obr. 5.5: Diagram modulu pre aktualizáciu honeypotu

Kuchárky pre aktualizáciu honeypotov je možné rozšíriť pri pridaní podpory nového typu honeypotu. Je potrebné vytvoriť novú kuchárku, ktorá zabezpečí vykonanie štyroch vyššie spomenutých krokov a spustiť ju. Nevyhnutnými krokmi pri pridávaní podpory nových typov honeypotov sú aj pridanie značky do súboru `inventory` (viz 4.3) špecifikujúcej servery na ktorých bude honeypot nasadený alebo aktualizovaný. Pokiaľ sa jedná o rozšírenie modulu pre hromadné nasadenie honeypotov 5.2 je potrebné vytvoriť novú rolu nasadenia v `/roles`, doplniť v nej potrebné súbory (je možné vychádzať z poskytnutej kostry nasadenia 5.2.1) a upraviť hlavnú kuchárku pre nasadenie honeypotov `hp_deployment.yml`. Súčasťou úpravy súboru `hp_deployment.yml` je doplnenie počiatočných otázok o otázku, či si užívateľ želá nasadiť nový typ honeypotu, uložiť jeho odpoveď a na základe hodnoty tejto odpovede spustiť rolu pre nasadenie nového typu honeypotu.

Kapitola 6

Testovanie a anlýza dát

V tejto kapitole je popísané testovanie pomocou sady testov a simulácie nasadenia alebo testovanie nasadených honeypotov na serveroch v sieti internet. Vytvorené Ansible kuchárky a role som sa rozhodol testovať pomocou rozhrania Molecule[1]. Molecule umožňuje testovanie každej role na rôznych operačných systémoch bežiacich v Docker kontajneroch[2]. Je umožnená simulácia nasadenia bez toho, aby si užívateľ musel zaobstarať servery v sieti.

6.1 Testovanie pomocou Molecule

V mojej práci som sa rozhodol pre testovanie nasadenia honeypotov na stroje bežiace pod operačným systémom Linux, konkrétne distribúcia Debian 10 a Ubuntu 20.04. Spomínané distribúcie som zvolil v dôsledku ich používanosti a toho, že všetky poskytnuté servery na ktorých boli honeypoty nasadené obsahovali distribúciu Debian 10. Použil som mierne upravenú verziu, ktorá okrem základnej funkcionality v kontajneri Docker poskytuje aj emulácie systémových služieb, ktoré sú využívané k nasadeniu honeypotov. Testovanie honeypotu Cowrie je špecifické v tom, že virtualizovaný operačný systém musí obsahovať podporu služby SSH spolu s jej konfiguráciou, aby bolo možné správne otestovať jej zamaskovanie.

Každá rola je otestovaná samostatne a okrem otestovania nasadenia, je otestovaná kvalita kódu a to či výsledný systém obsahuje nasadzovanú konfiguráciu. Testovanie sa primárne skladá z častí:

- Test kvality kódu a analýza potenciálnych chýb pomocou linterov (flake8[3], ansible-lint[4], a yamllint[5]). Linter je program, ktorý slúži k analýze kódu a upozorneniu užívateľa na konštrukcie vedúce k potenciálnym chybám prípadne odlišujúce sa od noriem.
- Test syntaktickej správnosti zdrojového kódu
- Vytvorenie obrazu operačného systému, na ktorom bude otestované nasadenie honeypotu a doinštalovanie potrebných súčastí systému.
- Test nasadenia zvolenej role pomocou Ansible kuchárky na simulovaný systém.
- Overenie výsledného systému

[1] https://molecule.readthedocs.io/en/latest/
[2] https://docker.com
[3] https://flake8.pycqa.org/en/latest/
[4] https://ansible-lint.readthedocs.io/en/latest/
[5] https://yamllint.readthedocs.io/en/stable/

K overeniu výsledného systému je použité rozhranie testinfra[6], ktoré umožňuje písanie testov v jazyku Python a overenie stavu výsledného systému po využití nástrojov pre správu konfigurácie ako napríklad *Ansible*. Ako som vyššie spomínal každá rola sa testuje samostatne a obsahuje aj rozdielne testy výsledného systému. Testy výsledného systému sú umiestnené v súbore `molecule/tests/test_default.py`. Medzi základné testy však patrí:

- Test balíčkov – skontroluje sa, či výsledný systém obsahuje balíčky, ktoré mali byť doinštalované pri nasadzovaní honeypotu.
- Test konfigurácie – skontroluje sa existencia konfiguračných súborov a porovná sa ich obsah s očakávaným obsahom po nasadení. Rovnako sa aj overia potrebné práva konfiguračných alebo spustiteľných súborov
- Test behu honeypotu – overenie existencie systémovej služby starajúcej sa o beh honeypotu a overenie, či je honeypot spustený a pripravený prijímať dáta.

Testy je možné spustiť pomocou príkazu `molecule test`, avšak je potrebné mať aktivované virtuálne prostredie, v ktorom sú nainštalované potrebné balíčky. Podrobný návod na spustenie je v dokumentácii `README.md` v sekcii `Testing`. Testy je potrebné spustiť z priečinku, v ktorom sa daná rola nachádza. Napríklad pre otestovanie funkčnosti nasadenia honeypotu Cowrie, je nutné pred zadaním príkazu `molecule test` sa presunúť do priečinku `/roles/cowrie_deployment`. Počas testovania nie je potrebný žiadny dodatočný zásah užívateľa a pri úspešnom vykonaní testov je skript ukončený s návratovým kódom **0**. Súhrn výsledkov testov je možné pozorovať na štandardnom výstupe, ktorý obsahuje výstup z nasadzovania honeypotu, ale aj výstup testov výsledného systému. Niektoré časti vyžadujúce pokročilú prácu so systémovými službami nie sú testované vzhľadom na problematickú prácu so systémovými službami v Docker kontajneroch. Výsledné služby sú otestované pomocou vyhľadania bežiaceho honeypot procesu, čo je ekvivalentné k overeniu, či služba beží v prípade reálneho nasadzovania. Dôležitou súčasťou testovania je testovanie honeypotov nasadených na reálnych serveroch na internete. Analýza získaných dát je popísaná v nasledujúcej kapitole.

6.2 Analýza dát

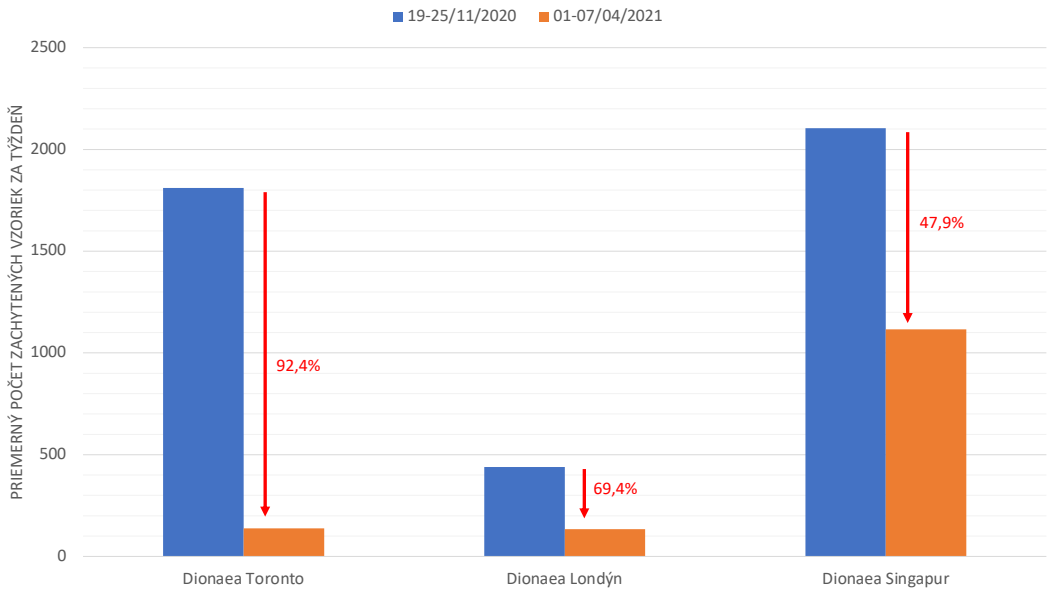
V rámci tejto bakalárskej práce boli honeypoty Dionaea nasadené na troch rôznych serveroch na internete. Servery sa nachádzajú v Toronte, Londýne a Singapure, každý na inom kontinente. Nasadená konfigurácia medzi servermi bola totožná až na doménové meno a názov zdieľaného disku užívateľa. Získané vzorky a metadáta boli odoslané pomocou protokolu MQTT do internej sieti firmy Avast k následnej analýze. Okrem nasadených honeypotov Dionaea boli rovnako nasadené aj tri honeypoty Cowrie na serveroch nachádzajúcich sa v mestách Soul, Singapur a Tel Aviv. Každý z nasadených honeypotov Cowrie sa líši v konfigurácii.

Honeypot Cowrie nasadený na server v meste Soul obsahuje vlastnú konfiguráciu s vlastným súborovým systémom a s umožnením prihlásenia po zadaní náhodných prístupových údajov s funkciou zapamätania si zadaných údajov pre ďalší prístup. Druhý nasadený Cowrie honeypot, konkrétne v meste Singapur, obsahuje pôvodnú konfiguráciu a takisto prihlásenie po zadaní náhodných prihlasovacích údajov. Posledný nasadený Cowrie honeypot

[6] https://testinfra.readthedocs.io/en/latest/

lokalizovaný v meste Tel Aviv obsahuje pôvodnú konfiguráciu a prihlásenie po zadaní predkonfigurovaných prihlasovacích údajov, ktoré ako užívateľské meno obsahujú jedno z trojice `root`, `tomcat` alebo `oracle` a ako heslo obsahujú ľubovoľný reťazec okrem reťazca `root`, `123456` alebo reťazec obsahujúci podreťazec `honeypot`. Jedná sa o predvolenú konfiguráciu bez žiadnych dodatočných zmien.

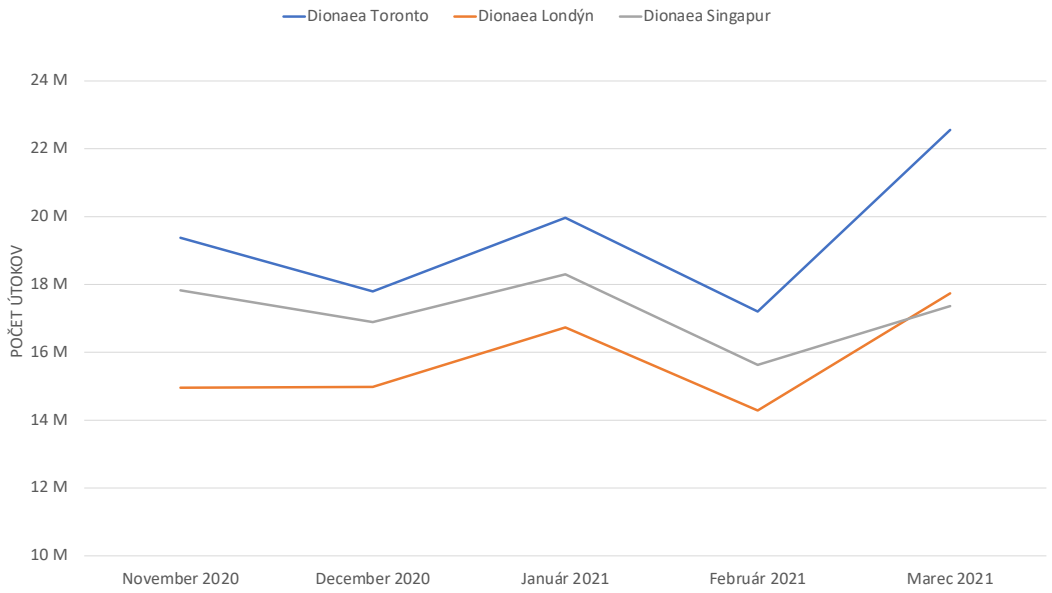
Honeypoty typu Dionaea boli nasadené v novembri roku 2020 a získané údaje boli analyzované za obdobie do apríla 2021. Je možné pozorovať úbytok zachytených vzoriek medzi prvým týždňom po nasadení a po devätnástich týždňoch od nasadenia. Je pravdepodobné, že tieto honeypoty aj napriek ich vlastnej konfigurácii boli pokročilejšími útočníkmi odhalené. Obrázok 6.1 predstavuje porovnanie priemerného počtu zachytených vzoriek honeypotom Dionaea medzi prvým a devätnástym týždňom od nasadenia.



Obr. 6.1: Dionaea – Priemerný počet zachytených vzoriek prvý týždeň po nasadení vs. devätnásty týždeň po nasadení

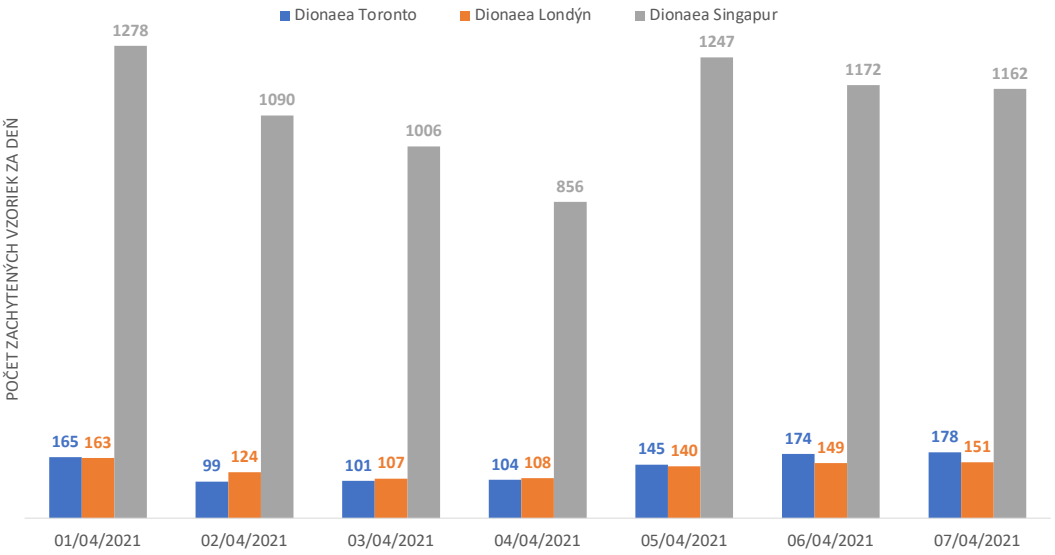
Zaujímavým faktom je, že úbytok zachytených vzoriek v meste Toronto bol takmer 93% a pri honeypote v meste Singapur sa jednalo len o úbytok takmer 50%. Vzhľadom na to, že konfigurácia honeypotov medzi mestami sa líšila len v použitom doménovom mene odvodeného z najpoužívanejších mien v danej krajine a názve zdieľaného disku predpokladám, že najväčší vplyv na úbytok vzoriek mala daná lokácia.

Aj napriek tomu, že postupom času upadal počet zachytených vzoriek, počet útokov na honeypoty typu Dionaea sa nijako zásadne nelíšil. Najpravdepodobnejším vysvetlením tohto javu je, že automatické skripty predstavujúce väčšinu vykonaných útokov neboli ovplyvnené časom. Ich cieľom je zistiť, či daný stroj je vhodný na útok, prípadne, či sa jedná o honeypot. Odhalenie a označenie daného servera za honeypot mohlo mať vplyv na následné sofistikovanejšie útoky predstavujúce snahu nakaziť systém vírusom. Takéto útoky boli značne zredukované. Graf 6.2 znázorňuje počet útokov počas jednotlivých mesiacov od nasadenia.



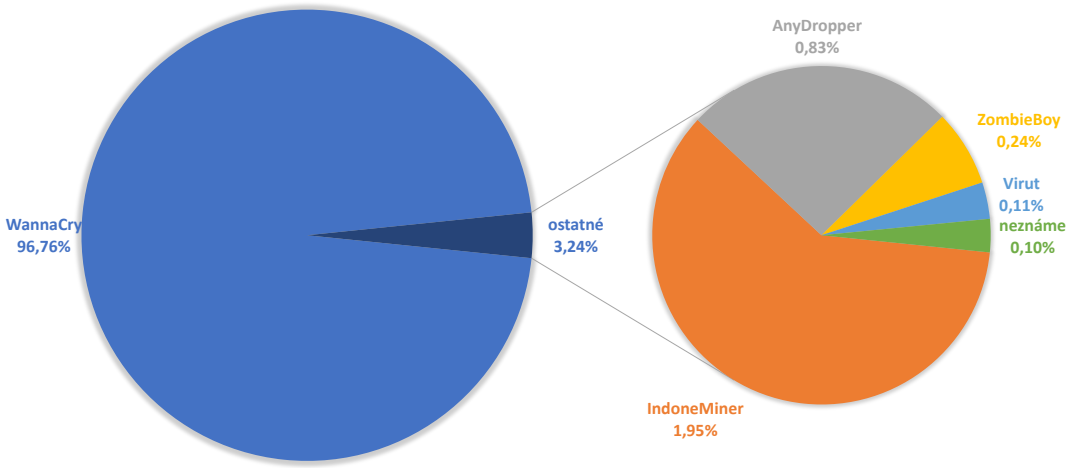
Obr. 6.2: Dionaea – Počet zachytených útokov počas jednotlivých mesiacov

Analýza posledného skúmaného týždňa ukazuje, ktorým Dionaea honeypotom bolo zachytené aké celkové množstvo vzoriek. Táto analýza je zobrazená v grafe 6.3. Je možné pozorovať menší úbytok 3. apríla a 4. apríla 2021. Tieto dni predstavujú sobotu a nedeľu. Veľká väčšina zachytených vzoriek pochádza z honeypotu v meste Singapur, ktorý aj po takmer piatich mesiacoch zachytáva veľké množstvo získaných vzoriek za deň. Pri nasadzovaní ďalších honeypotov by som vybral práve podobné lokácie.



Obr. 6.3: Dionaea – Počet zachytených vzoriek za jednotlivé dni v týždni

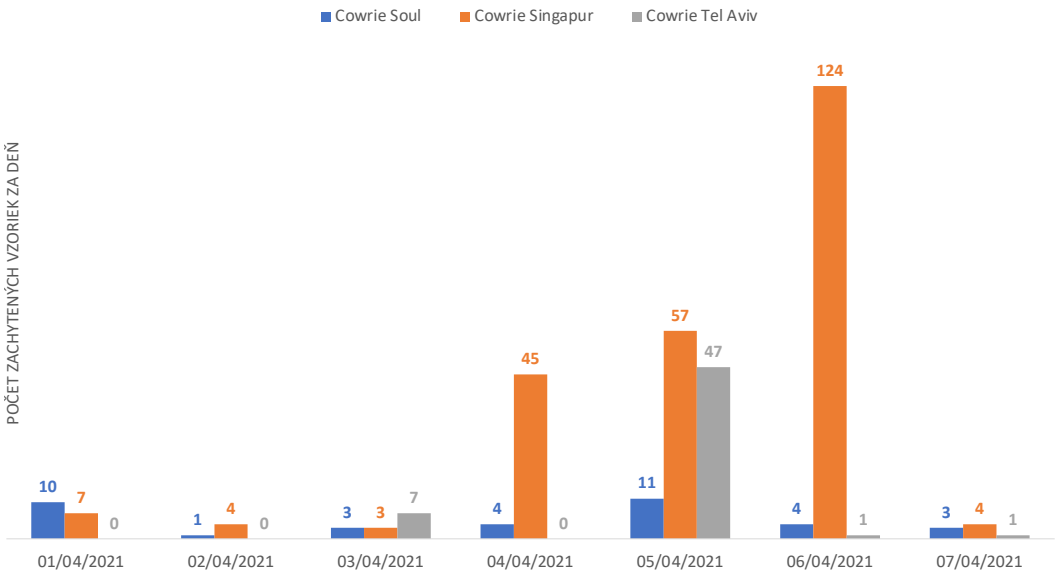
Zachytené vzorky boli odoslané do internej sieti firmy Avast, kde bola následne vykonaná ich analýza a vzorky boli roztriedené do rodín, kam daný malware spadá. Obrázok 6.4 predstavuje percentuálny pomer zachytených malware vzoriek. Väčšina zachytených vzoriek patrila pod rodinu *WannaCry*[7], ale je možné vidieť aj zastúpenie iných rodín. Vírusy spadajúce pod Rodinu *WannaCry* majú za úlohu vydierať užívateľa formou zašifrovania jeho pevného disku a požadovania výkupného za jeho odšifrovanie. Ostatné typy rodín boli zastúpené v mnohonásobne menšom počte. Jednalo sa prevažne o rodiny slúžiace k ťaženiu kryptomien alebo ako zadné dvierka pre stiahnutie iných škodlivých súborov na cieľový počítač. Údaje sú zozbierané za týždeň od 1. do 7. apríla 2021.



Obr. 6.4: Dionaea – Pomer zachytených malware rodín

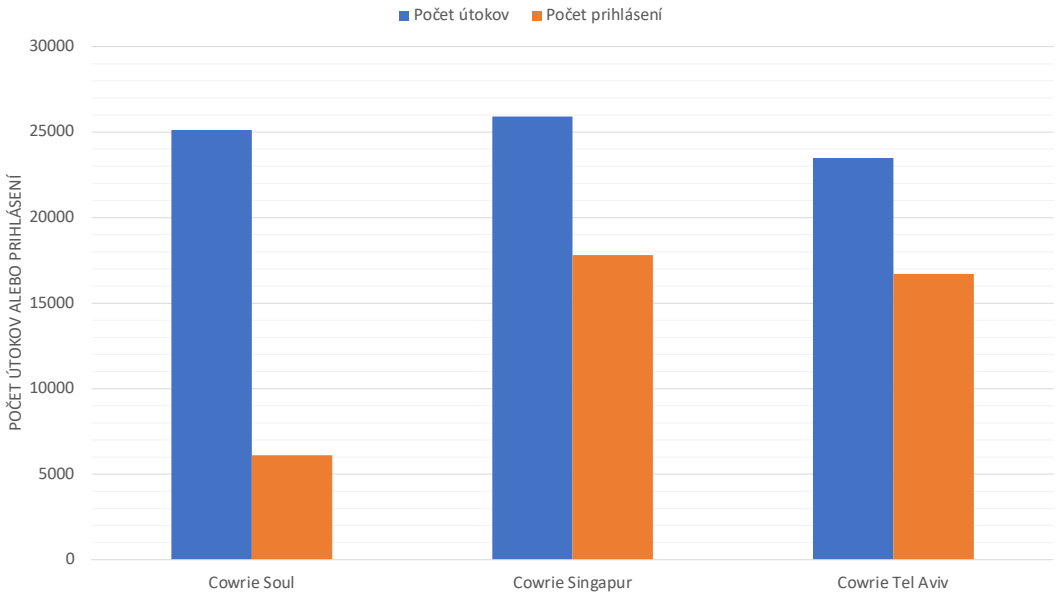
Honeypoty typu Cowrie boli nasadené koncom marca 2021 a zozbierané údaje sú za obdobie od 1. do 7. apríla 2021. Súčasťou analýzy získaných dát znázornenej na obrázku 6.5 bolo porovnanie jednotlivých nasadených konfigurácii, ktoré sa od seba zásadne líšia. Pri počte zachytených vzoriek za prvý týždeň zberu od nasadenia je možné pozorovať obmedzenia konfigurácie prihlasovania pomocou náhodných prihlasovacích údajov s funkciou zapamätania si týchto údajov a priradenia k IP adrese útočníka. Táto kombinácia ostane uložená a následné prihlásenia z rovnakej IP adresy budú povolené len s rovnakou kombináciou. Tento spôsob zabezpečenia je nastavený tak, aby nikdy nepovolil prihlásenie po zadaní prvého hesla. Minimálny počet pokusov musí byť dva a maximálny päť. Myslím si, že toto je hlavný dôvod, prečo počet zachytených vzoriek za týždeň na honeypote v meste Soul, ktorý obsahuje tento spôsob prihlasovania je nižší oproti ostatným honeypotom. Zvyšné dva honeypoty obsahujú pôvodný systém prihlasovania, ktorý umožňuje prihlásenie po zadaní vopred špecifikovaných povolených kombinácii prihlasovacích údajov.

[7] https://avast.com/c-wannacry



Obr. 6.5: Cowrie – Počet zachytených vzoriek za jednotlivé dni v týždni

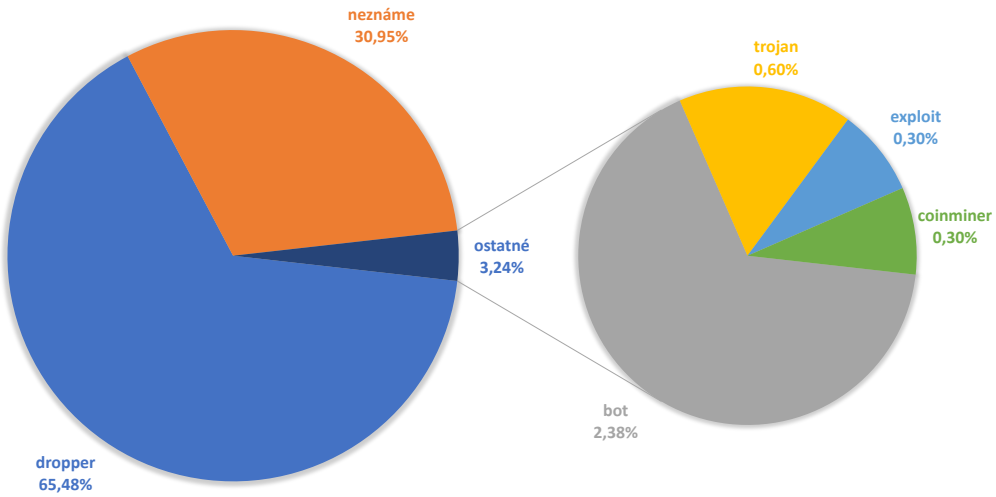
Najlepšie výsledky z hľadiska počtu zachytených vzoriek dosiahol honeypot nasadený v meste Singapur. Tento honeypot obsahuje mnou vytvorenú konfiguráciu a pôvodný systém prihlasovania. Graf 6.6 zobrazuje počet útokov a počet úspešných útokov. Za úspešný útok predpokladám, že útočníkovi sa podarilo prihlásiť na stroj a bol schopný ho ďalej využívať. Počty zaznamenaných útokov na jednotlivé verzie honeypotov boli minimálne odlišné. Je možné však pozorovať, že počet úspešných prihlásení, ktorý je priamo ovplyvnený spôsobom prihlasovania je výrazne nižší na honeypote v meste Soul. Na dané výsledky môže mať vplyv aj lokácia, podobne ako pri honeypotoch Dionaea.



Obr. 6.6: Cowrie – Pomer úspešných útokov

Získané vzorky boli rovnako ako pri vzorkách z honeypotu Dionaea analyzované a roztriedené. Zatiaľ čo honeypot Dionaea zbieral škodlivé súbory určené pre operačný systém Windows, honeypot Cowrie zbiera škodlivé súbory určené pre operačný systém Linux. Vo väčšine prípadov jedná o textové skripty, ktoré vyžadujú dynamickú analýzu. Vzorky z honeypotu Cowrie boli roztriedené do typov, keďže firma Avast sa zameriava na klasifikáciu vzoriek určených pre operačný systém Linux. Podstatné množstvo zachytených vírusov nebolo možné roztriediť ani do typov. Po ručnom preskúmaní veľkého množstva práve týchto nezaradených vírusov som dospel k názoru, že ich nebolo možné zaradiť vzhľadom na ich obsah. Väčšinou sa jednalo o textové súbory obsahujúce SSH kľúč útočníka nasadený na honeypot alebo zoznam príkazov k stiahnutiu ďalších potenciálne škodlivých súborov.

Pomer zachytených typov vírusov na Cowrie honeypotoch je znázornený na obrázku 6.7. Najpočetnejšie zastúpenie má typ *dropper*[8], ktorý slúži k stiahnutiu iného typu vírusu. Ďalšie rozpoznané typy vírusov sú napríklad *trojan*[9], *bot*[10], *coinminer*[11] alebo *exploit*[12]. *Trojan* je typ vírusov, ktorý sa navonok tvári ako bezpečný software, ale často rovnako ako *dropper* slúži k stiahnutiu iného škodlivého vírusu. Typ *bot* slúži k nadobudnutiu kontroly nad infikovaným počítačom a vykonávaniu často opakujúcich sa činností. *Coinminer* je typ vírusu určený na ťaženie kryptomien bez vedomosti vlastníka počítača a *exploit* je program, ktorý slúži na vyhľadanie bezpečnostných slabín aplikácie a umožňuje útočníkovi ich zneužiť vo svoj prospech.

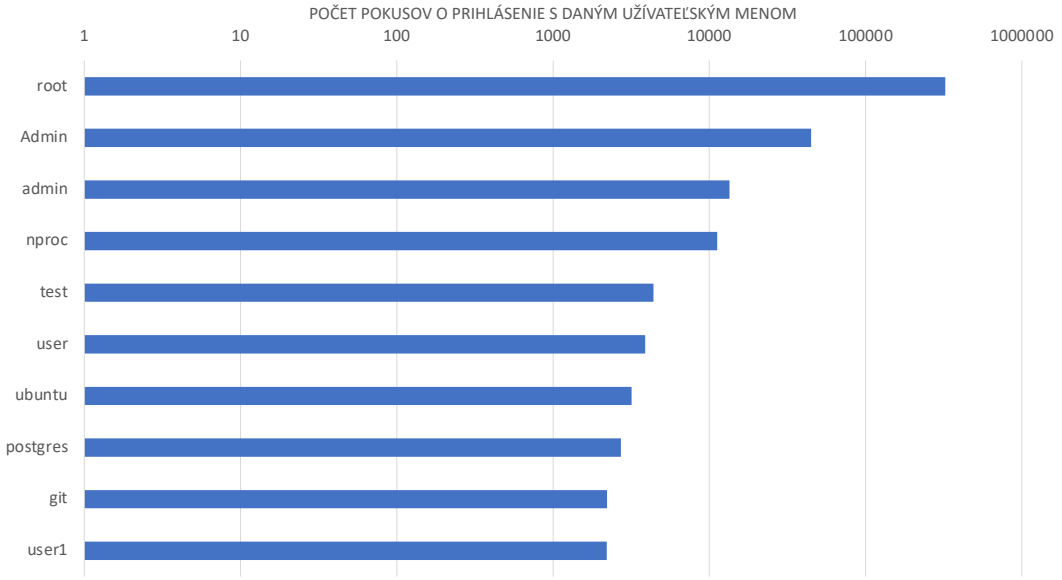


Obr. 6.7: Cowrie – Pomer zachytených typov vírusov

[8] https://blog.malwarebytes.com/detections/trojan-dropper/
[9] https://avast.com/c-trojan
[10] https://us.norton.com/internetsecurity-malware-what-are-bots.html
[11] https://avast.com/c-protect-yourself-from-cryptojacking
[12] https://avast.com/c-exploits

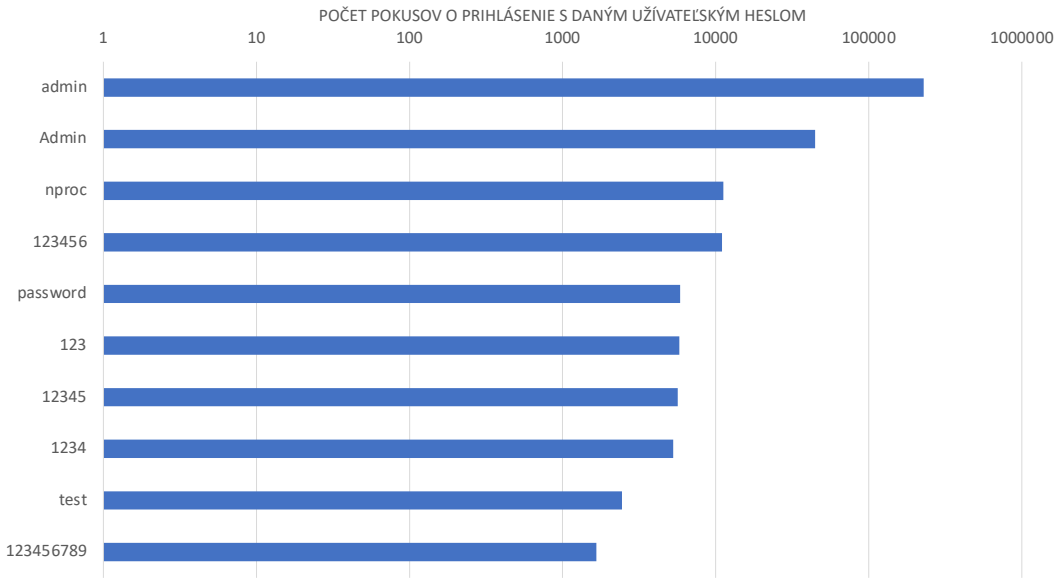
Vzhľadom na skutočnosť, že najlepšie výsledky dosiahol honeypot s vlastnou konfiguráciou a prihlasovaním cez vopred špecifikované prihlasovacie údaje som sa rozhodol pre analýzu použitých prihlasovacích údajov a zobrazenie desiatich najpoužívanejších prihlasovacích údajov na obrázku 6.8 a hesiel na obrázku 6.9. Vzhľadom na veľký rozdiel medzi prvým a druhým najpoužívanejším prihlasovacím menom alebo heslom oba grafy obsahujú x-ovú os v logaritmickom meradle. Do tejto štatistiky sú okrem úspešných zarátané aj neúspešné pokusy na všetkých troch honeypotoch. Všetky konfigurácie majú teda na výsledok rovnaký vplyv.

Najpoužívanejšie prihlasovacie meno je s vysokým náskokom meno `root`, za ním nasledujú varianty mena `admin` s veľkým aj malým začiatočným písmenom.



Obr. 6.8: Cowrie - Najpoužívanejšie užívateľské mená

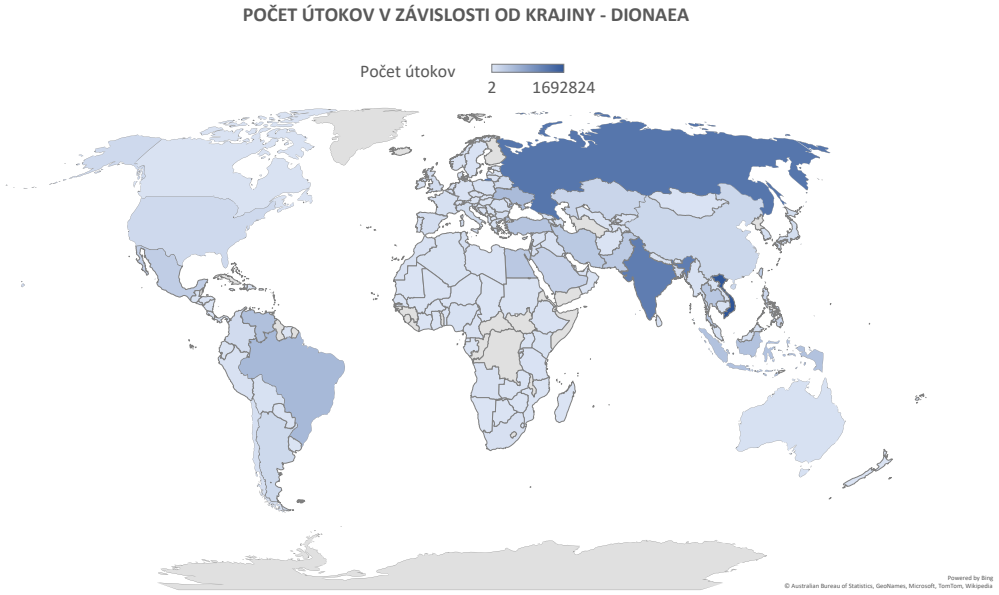
Najpoužívanejším heslom sú podobne ako pri užívateľských menách varianty slova `admin` s veľkým a malým začiatočným písmenom. Veľké zastúpenie medzi použitými heslami majú aj tradičné číselné postupnosti rôznej dĺžky.



Obr. 6.9: Cowrie – Najpoužívanejšie užívateľské heslá

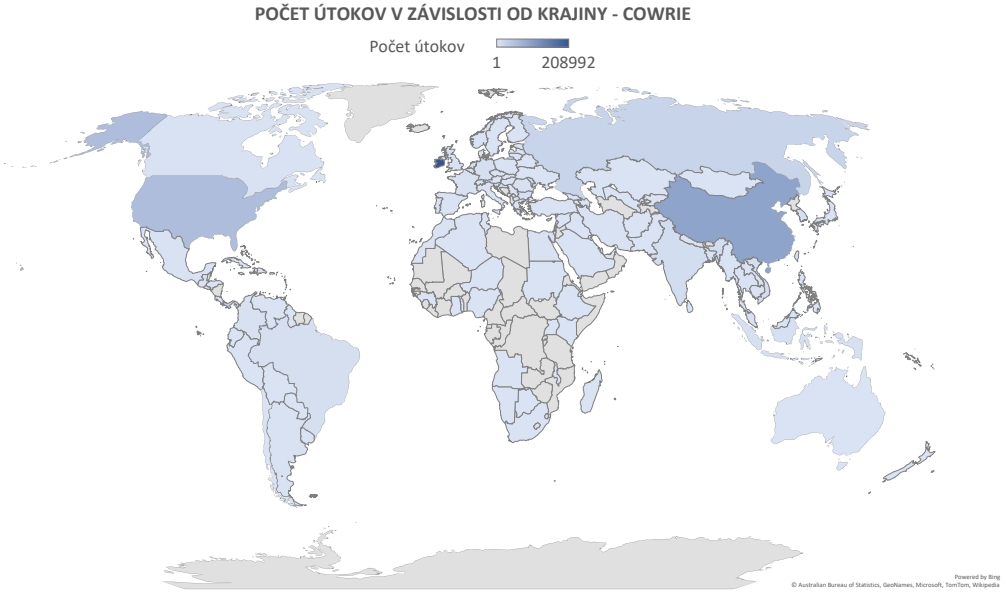
Poslednou spracovanou štatistikou spoločnou pre honeypoty Cowrie a Dionaea je počet útokov v závislosti od ich lokácie. Do úvahy boli pre obidva typy honeypotov brané dáta za týždeň od 1. do 7. apríla a výsledky sú vyobrazené na mape sveta. Čím tmavšia je farba krajiny, tým väčšie množstvo útokov z danej krajiny prišlo. Počet útokov na honeypoty Dionaea znázornený na obrázku 6.10 bol za spomínaný týždeň podstatne väčší a teda aj zastúpenie útočiacich krajín je rôznorodejšie ako pri honeypote Cowrie. Výsledky skúmania môžu byť ovplyvnené pomerne krátkou dobou merania, pri dlhšej analýze by sa spriemerovali možné jednorázové kampane, ktoré v rámci jedného dňa mohli vykonať veľké množstvo útokov z jednej lokácie a tým pádom skresliť meranie. Počet útokov na honeypoty typu Cowrie v závislosti od lokácie je znázornený na obrázku 6.11.

Pri honeypotoch Dionaea bol vykonaný najväčší počet útokov z krajín ako Rusko, India, Vietnam či Brazília. Naopak možno prekvapivo Čína a Spojené štáty americké patria ku krajinám s nižším počtom útokov.



Obr. 6.10: Dionaea – Počet útokov v závislosti od krajiny

Pri honeypotoch Cowrie bol suverénne najväčší počet útokov vykonaný z Írska čo je určite takisto prekvapivý výsledok. Veľký počet útokov bol vykonaný aj z krajín ako Čína alebo Spojené štáty americké.



Obr. 6.11: Cowrie – Počet útokov v závislosti od krajiny

Kapitola 7

Záver

Cieľom tejto práce bolo navrhnúť a implementovať nástroj slúžiaci k automatizácii nasadenia, monitorovania a zberu dát z honeypotov.

Práca skúmala možnosti nasadenia honeypotov a ich následné monitorovanie s cieľom získať množstvo užitočných dát k ďalšiemu spracovaniu. Riešenie bolo navrhnuté v spolupráci so spoločnosťou Avast, ktorá bola zadávateľom práce. Boli preskúmané možnosti nasadenia s cieľom vytvoriť užívateľsky prívetivý koncept jednoduchého nasadenia voľne dostupných honeypotov. Na základe podrobnej analýzy bol následne vybratý softvér určený k monitorovaniu výsledného riešenia a upozorňovaniu na prípadné chyby či zotaveniu sa z nich. Získané dáta boli spracované a vizualizované.

Výsledné riešenie je koncipované tak, aby sa umožnilo jednoduché nasadenie ľubovoľného honeypotu bez nutnosti výrazného rozšírenia. V aktuálnom štádiu je možné nasadiť tri rôzne druhy honeypotov (Dionaea, Hermes a Cowrie). Pri riešení som sa zoznámil s rôznymi nástrojmi pre správu konfigurácie, ktoré boli využité práve k nasadeniu a počiatočnej konfigurácii zvolených honeypotov. Pri vytváraní konceptu bolo nutné v prvom rade brať ohľad na bezpečnosť. Práca ma naučila, aké dôležité je zabezpečiť honeypoty pred ich odhalením a prípadnou kompromitáciou či zneužitím samotného servera. Návrh počíta s nasadzovaním veľkého množstva rôznych druhov honeypotov na serveroch v sieti, pričom cieľom je zaistiť unikátnosť každého takto nasadeného honeypotu. Unikátnosť je zaistená formou vlastnej konfigurácie obohatenej o interaktívny prístup pri nasadzovaní. V rámci nasadzovania je možnosť zvoliť typ honeypotu, rozšíriť jeho počiatočnú konfiguráciu a zároveň minimalizovať čas strávený pri inštalácii. Riešenie je doplnené všeobecnou schémou nasadzovania, ktorá slúži k jednoduchému návodu pre užívateľa ako v budúcnosti toto riešenie rozšíriť. Okrem možnosti nasadiť honeypoty je implementovaná aj možnosť honeypoty kedykoľvek aktualizovať.

Práca bola otestovaná na serveroch v sieti internet a po analýze výsledkov môžem konštatovať, že práca spĺňa predpoklady a nadobudnuté výsledky obsahujú množstvo užitočných informácii. Získané informácie o útokoch na nasadené honeypoty a informácie o porovnaní nasadených konfigurácii je v budúcnosti v pláne využiť na vylepšenie konfigurácie honeypotov s cieľom dosiahnúť ešte lepšie výsledky. Aktuálne riešenie sa zameriava primárne na nasadenie honeypotov a ich unikátna konfigurácia je pomerne jednoduchá a ľahko odhaliteľná. V pláne je vylepšiť túto konfiguráciu tak, aby zaistila čo najspoľahlivejšie výsledky a doplniť ďalšie druhy honeypotov. Ďalším možným pokračovaním vývoja je zapracovanie poznatkov z monitorovania a doplnenie skriptu pre automatickú obnovu o ďalšie procesy.

Literatúra

[1] Benson, J. O., Prevost, J. J. a Rad, P. Survey of Automated Software Deployment for Computational and Engineering Research. In: *2016 Annual IEEE Systems Conference (SysCon)*. 2016, s. 1–6. DOI: 10.1109/SYSCON.2016.7490666. ISBN 9781467395199.

[2] Brikman, Y. *Why We Use Terraform and Not Chef, Puppet, Ansible, Saltstack, or Cloudformation*. Gruntwork, Júl 2019 [cit. 2020-12-16]. Dostupné z: https://blog.gruntwork.io/why-we-use-terraform-and-not-chef-puppet-ansible-saltstack-or-cloudformation-7989dad2865c.

[3] Cabral, W., Valli, C., Sikos, L. a Wakeling, S. Review and Analysis of Cowrie Artefacts and Their Potential to be Used Deceptively. In: *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. 2019, s. 166–171. DOI: 10.1109/CSCI49370.2019.00035. ISBN 9781728155845.

[4] Chlupová, S. *Software pro zachytávání a inteligentní zpracování spamu*. Brno, CZ, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/22243/.

[5] Crapanzano, J. Deconstructing subseven, the trojan horse of choice. *SANS Institute*. 2003.

[6] DigitalOcean. *An Introduction to Configuration Management*. DigitalOcean, December 2020 [cit. 2020-1-10]. Dostupné z: https://www.digitalocean.com/community/tutorials/an-introduction-to-configuration-management.

[7] Felt, A. P., Finifter, M., Chin, E., Hanna, S. a Wagner, D. A Survey of Mobile Malware in the Wild. In: *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*. New York, NY, USA: Association for Computing Machinery, 2011, s. 3–14. DOI: 10.1145/2046614.2046618. ISBN 9781450310000. Dostupné z: https://doi.org/10.1145/2046614.2046618.

[8] Hardion, V., Spruce, D., Lindberg, M., Otero, A., Lidon Simon, J. et al. Configuration Management of the Control System. In: *ICALEPCS'13*. Október 2013. ISBN 9783954501397.

[9] Hochstein, L. a Moser, R. *Ansible: Up and Running: Automating Configuration Management and Deployment the Easy Way*. 2. vyd. O'Reilly Media, Inc., 2017. ISBN 9781491979754.

[10] Hunkeler, U., Truong, H. L. a Stanford-Clark, A. MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks. In: *2008 3rd*

International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08). 2008, s. 791–798. DOI: 10.1109/COMSWA.2008.4554519. ISBN 9781424417964.

[11] Johari, A. *Chef vs Puppet vs Ansible vs Saltstack: Which One to Choose*. November 2019 [cit. 2020-12-16]. Dostupné z: https://www.edureka.co/blog/chef-vs-puppet-vs-ansible-vs-saltstack/.

[12] Josefsson, S. *The Base16, Base32, and Base64 Data Encodings* [Internet Requests for Comments]. RFC 4648. RFC Editor, October 2006. Dostupné z: http://www.rfc-editor.org/rfc/rfc4648.txt.

[13] Joshi, R. a Sardana, A. *Honeypots: A New Paradigm to Information Security*. 1. vyd. CRC Press, 2011. ISBN 9781578087082.

[14] Morris, K. *Infrastructure as Code*. 1. vyd. O'Reilly Media, 2016. ISBN 9781491924358.

[15] Nawrocki, M., Wählisch, M., Schmidt, T. C., Keil, C. a Schönfelder, J. A Survey on Honeypot Software and Data Analysis. *ArXiv preprint arXiv:1608.06249*. 2016.

[16] Obied, A. Honeypots and spam. *University of Calgary*. 2006.

[17] Project, H. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. 1. vyd. Addison-Wesley, 2001. ISBN 9780201746136.

[18] Sethia, V. a Jeyasekar, A. Malware Capturing and Analysis using Dionaea Honeypot. In: *2019 International Carnahan Conference on Security Technology (ICCST)*. 2019, s. 1–4. DOI: 10.1109/CCST.2019.8888409. ISBN 9781728115764.

[19] Sissel, J. *Day 19 - Why Use Configuration Management?* 2011 [cit. 2020-12-21]. Dostupné z: https://sysadvent.blogspot.com/2011/12/day-19-why-use-configuration-management.html.

[20] Spitzner, L. The Honeynet Project: Trapping the Hackers. *IEEE Security Privacy*. 2003, zv. 1, č. 2, s. 15–23.

[21] Spitzner, L. *Honeypots: Tracking Hackers*. 1. vyd. Addison-Wesley Reading, 2003. ISBN 9780321108951.

[22] Stoll, C. *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*. 1. vyd. Simon and Schuster, 2005. ISBN 9781416507789.

[23] Turnbull, J. *Monitoring With Prometheus*. 1. vyd. Turnbull Press, 2018. ISBN 9780988820289.

[24] Venezia, P. *Puppet vs. Chef vs. Ansible vs. Salt*. InfoWorld, November 2013 [cit. 2020-12-16]. Dostupné z: https://www.networkworld.com/article/2172097/puppet-vs--chef-vs--ansible-vs--salt.html.

Príloha A

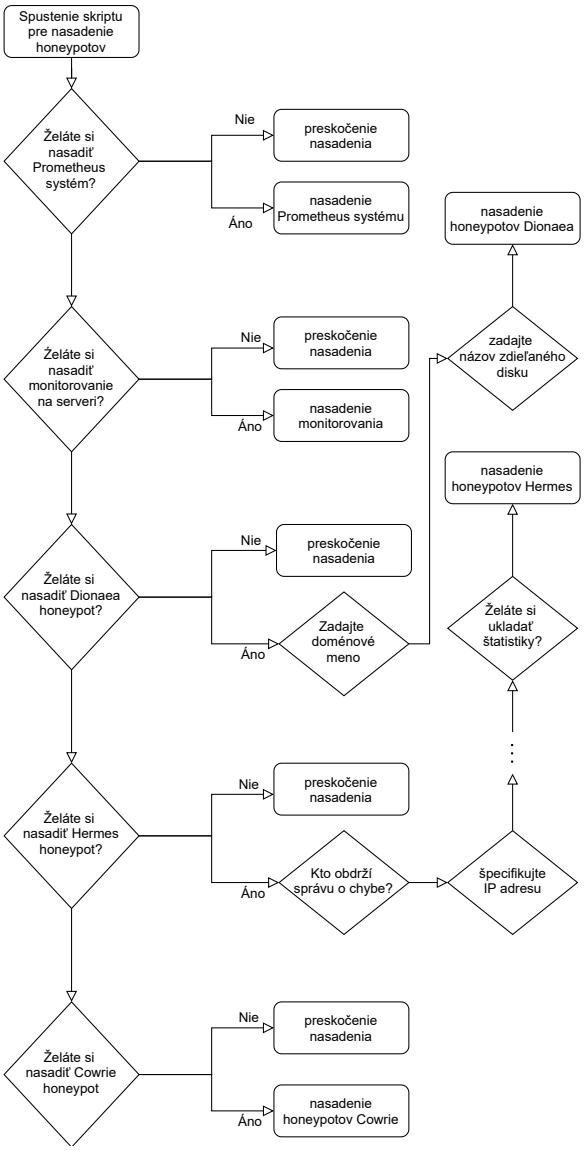
Obsah CD

Priložené CD obsahuje:

- `/latex` - priečinok so zdrojovými súbormi textu bakalárskej práce v Latexe
- `/src` - priečinok so zdrojovými súbormi programu
- `/src/README.md` - priložená dokumentácia k programu v anglickom jazyku
- `thesis.pdf` - bakalárska práca vo formáte PDF
- `obsah.txt` - stručný popis obsahu CD

Príloha B

Nasadenie honeypotov



Obr. B.1: Diagram skriptu pre automatické nasadenie honeypotov