UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

DETEKCIA HONEYPOTOV

2020 **Bc. Lucia KOKUĽOVÁ**

UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

DETEKCIA HONEYPOTOV

DIPLOMOVÁ PRÁCA

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

Košice 2020

Bc. Lucia KOKUĽOVÁ

Zadanie záverečnej práce

Poďakovanie

Týmto sa chcem poďakovať vedúcemu tejto diplomovej práce RNDr. JUDr. Pavlovi Sokolovi, PhD. a taktiež konzultantovi RNDr. Tomášovi Bajtošovi za ich cenné rady a pomoc s prípravou a písaním tejto práce.

Abstrakt v štátnom jazyku

Práca sa zaoberá testovaním jednej z najdôležitejších vlastností honeypotov, a to ich detekovateľnosťou. Princíp honeypotu spočíva v jeho neautorizovanom využití a je preto dôležité, aby honeypot nebolo možné v rámci počítačovej siete detegovať. V práci z tohto dôvodu rozoberieme rôzne techniky detekcie honeypotov. Našim cieľom bude navrhnúť systém, ktorý bude honeypoty detegovať, čím odhalí ich slabé stránky, ktoré následne môžu byť odstránené. Výsledkom našej práce bude nástroj, ktorý na vstupe dostane IP adresu a jeho výstupom bude vyjadrenie, či daný stroj je honeypot, alebo reálny operačný systém, spôsob detekcie a navrhnuté opatrenie proti detekcii.

Kľúčové slová: honeypot, honeynet, podvodné systémy, detekcia, informačná bezpečnosť

Abstrakt v cudzom jazyku

This paper deals with testing one of the most important properties of honeypots, namely their detectability. The principle of honeypot lies in its unauthorized use and it is therefore important that honeypot cannot be detected within the computer network. For this reason we will discuss various techniques of honeypot detection. Our goal is to design a system that detects honeypots, revealing their weaknesses, which can then be removed. The result of our paper will be a tool that gets an IP address on the input and its output will be a statement whether the machine is a honeypot or a real operating system, a method of detection and a proposed measure against detection.

Keywords: honeypot, honeynet, deception systems, detection, information security

Obsah

Zoznam ilustrácií

Zoznam tabuliek

Zoznam skratiek a značiek

ADB	Android Debug Bridge
FTP	File Transfer Protocol, protokol prenosu súborov
HTTP	Hypertext Transfer Protocol, hypertextový prenosový protokol
ICMP	Internet Control Message Protocol,
IDS	Intrusion Detection System, systém detekcie narušenia
KVM	Kernel-Based Virtual Machine, virtuálny stroj založený na jadre
LXC	Linux Containers
MAC	Media Access Control, identifikátor sieťového zariadenia
QEMU	Quick EMUlator
RTT	Round-Trip Time, obojsmerné omeškanie
SMTP	Simple Mail Transfer Protocol, jednoduchý protokol na prenos pošty
SSH	Secure Shell, zabezpečený prístup k príkazovému interpretovaču
TCP	Transmission Control Protocol, protokol riadenia prenosu
UDP	User Datagram Protocol, používateľský datagramový protokol
UML	User Mode Linux, užívateľský režim Linuxu

Úvod

V sieti internet tak ako ju poznáme dnes, sú pripojené milióny zariadení. Na tieto zariadenia každý deň smerujú útoky, či už s cieľom zneškodniť ich, alebo získať od nich rôzne informácie. Útočníci sa každým dňom vyvíjajú a ich útoky sú čím ďalej vyspelejšie. Je preto dôležité ich pozorovať, porozumieť ich technikám, a tým sa naučiť odvracať hrozby, ktoré predstavujú. Na tieto účely existuje mnoho nástrojov, ktoré majú za úlohu útočníka odhaliť, poprípade analyzovať. V tejto práci sa budeme venovať jednému z nich, konkrétne honeypotu.

Honeypot je bezpečnostný nástroj na detekciu a prípadné odvrátenie neautorizovanej manipulácie so systémom. Tento nástroj sleduje aktivitu útočníka v počítačovej sieti, čím ho dokáže efektívne identifikovať a následne monitorovať jeho aktivitu. Jeho hodnota spočíva v tom, že ho útočník použije, honeypot by sa teda mal útočníkovi javiť ako dostupný cieľ. Jednou z najdôležitejších vlastností honeypotov je však skutočnosť, že útočník nemá vedomosť o tom, že sa pripája na honeypot (nedetekovateľnosť).

Napriek tomu, že útočník má byť schopný neautorizovane použiť honeypot, nemôže byť schopný odhaliť to, že s ním komunikuje. V tomto prípade by hrozilo, že ukončí, resp. pozmení svoju aktivitu, čím sa cieľ honeypotu – identifikácia správania útočníka, nepodarí zrealizovať. Napriek skutočnosti, že nedetekovateľnosť je základná vlastnosť honeypotov, niektoré typy honeypotov je možné detegovať na základe ich charakteristických čŕt. Pri detekcii je možné sledovať napríklad služby ponúkané systémom, o ktorej chceme zistiť či je honeypotom. To nám môže napovedať viac o tom, či je systém reálny alebo komunikujeme s honeypotom. Inými vlastnosťami, ktoré pri detekcii môžeme využiť môže byť nezvyčajné správanie či používanie špecifických hardvérových zariadení.

Jednou z možností, ako vyriešiť tento problém, je pokúsiť sa obmedziť možnosti detekcie honeypotov. Cieľom našej práce je preto implementácia nástroja na detekciu honeypotov, ktorého úlohou bude odhaliť honeypot v rámci počítačovej siete, na základe čoho bude možné odstrániť vlastnosti, ktoré dopomohli k jeho odhaleniu. Súčasné riešenia detekcie honeypotov ponúkajú rôzne spôsoby ako odhaliť honeypot zapojený v sieti, pričom na ich detekciu používajú ich známe vlastnosti, či predvolené nastavenia.

Táto detekcia je teda založená na predpoklade, že používateľ, ktorý honeypot nasadil, mu ponechal jeho pôvodné nastavenia, či ho nedostatočne zabezpečil.

Cieľom tejto práce je nájsť všeobecný spôsob detekcie honeypotov. Predstavíme si preto pojem podvodné systémy a taktiež spôsoby podvodov, ktoré tieto systémy používajú. Honeypoty taktiež zaradzujeme k podvodným systémom, keďže ich cieľom je nalákať útočníka na ich zneužitie. Pomocou týchto pojmov budeme schopní klasifikovať honeypoty vo všeobecnosti do 6 základných skupín, ktoré popisujú typy podvodov používaných v honeypotoch, a to maskovanie, obalenie, zatienenie, napodobňovanie, vynájdenie a návnada. Táto klasifikácia nám pomôže viac pochopiť správanie honeypotov a tým nám uľahčí nájdenie všeobecného spôsobu, ako tieto podvodné systémy detegovať.

1 Honeypoty a honeynety

V tejto kapitole si zadefinujeme najdôležitejšie pojmy týkajúce sa tejto práce, honeypot a honeynet. Pozrieme sa taktiež na delenie honeypotov a honeynetov z rôznych hľadísk a posúdime, ktoré z týchto hľadísk je pre našu prácu významné.

1.1 Honeypot

Honeypot je systémový prostriedok, ktorý sa tvári ako služba, množina služieb alebo celý operačný systém či sieť a jeho úlohou je nalákať útočníka [6]. Cieľom honeypotu je vyzerať pred útočníkom ako ľahký, zaujímavý alebo cenný cieľ. Potom ako útočník honeypot napadne, teda skúsi ho využiť vo svoj prospech, honeypot monitoruje každú činnosť útočníka v systéme, ako je prihlasovanie, úprava súborov či spustené procesy. Na rozdiel od firewallu či systému na detekciu narušení (Intrusion Detection System, IDS) honeypot poskytuje oveľa zložitejšiu formu ochrany používateľa. Stratégiou honeypotu je nalákať útočníka, aby ho zneužil. Týmto spôsobom odtiahne jeho pozornosť od reálneho systému, v ktorom je používaný, čím chráni jeho údaje.

Pre honeypoty zvyčajne platí, že nemajú žiadnu špecifickú vlastnosť, podľa ktorej by mohli byť rozdelené do jednoznačných skupín. V rámci odbornej literatúry z tohto dôvodu nájdeme mnoho delení podľa rozličných kritérií, pričom je časté, že kategórie honeypotov sa navzájom prekrývajú a dopĺňajú. V nasledujúcich podkapitolách si predstavíme niekoľko kritérií a delení honeypotov, ktoré sú pre našu prácu dôležité.

1.1.1 Delenie podľa použitia

Podľa využitia sa honeypoty delia na dve skupiny, a to produkčné a výskumné honeypoty. **Produkčné honeypoty** [6] sú určené na ochranu organizácií. Ich cieľom je znížiť risk napadnutia organizácie. Zvyčajne sú to nízko interaktívne honeypoty, ktoré sú ľahko nasaditeľné [7]. Honeypoty sa zvyčajne používajú v systéme spolu s firewallmi, IDS systémami a inými spôsobmi zabezpečenia, keďže takýto honeypot nie je dostačujúcou ochranou pre organizáciu. Sú určené pre zbieranie informácií o tom, aké hrozby organizácii čelia.

Druhou skupinou sú **výskumné honeypoty** [6]. Tento typ honeypotov sa používa na zistenie podrobnejších informácií o bezpečnostnej hrozbe, ktorej čelí

organizácia.Tento typ honeypotu je určený na to, aby sa učil. Honeypot teda okrem monitorovania útoku, získava údaje o útočníkovi, ktoré vie neskôr využiť na obranu voči nemu. Tým sa podieľa na dôležitom probléme zabezpečenia organizácií, a to na zistení identity a motivácie útočníka pri útoku na aktíva danej organizácie. Výskumné honeypoty sa tiež využívajú na odhalenie automatizovaných útokov.

1.1.2 Delenie podľa miery interakcie

V tejto podkapitole si predstavíme tri skupiny honeypotov, a to nízko, stredne a vysoko interaktívne honeypoty. Miera interakcie znamená, nakoľko je potenciálnemu útočníkovi umožnené zasahovať do honeypotu. Inými slovami, je to teda interakcia medzi nasadeným systémom a útočníkom. Interakcia ako kritérium delenia je pri honeypotoch ich najčastejšie využívaná vlastnosť [8].

Nízko interaktívne honeypoty sa vyznačujú minimálnou interakciou s útočníkom [6]. Zvyčajne sa javia len ako jednoduché sieťové služby (napr. SSH, HTTP a pod.). Na takýchto honeypotoch nebeží žiadny operačný systém. Nízko interaktívne honeypoty sú rozšírené, pretože sú veľmi ľahko nasaditeľné a udržiavateľné. Na druhej strane ale o útočníkovi vedia zistiť menej informácií (napr. IP adresu, sieťový port, použité prihlasovacie údaje). Takéto honeypoty sa používajú hlavne ak stredne a vysoko interaktívne honeypoty neprichádzajú do úvahy (napríklad nedostatočný hardvér na ich nasadenie) alebo ak chceme otestovať zraniteľnosti konkrétnej služby, ktorú nízko interaktívny honeypot môže predstavovať. Príkladom nízko interaktívnych honepotov sú Dionaea [9] a Honeyd [10].

Najdôležitejšou kategóriou sú **vysoko interaktívne honeypoty** [6]. Tie útočníkovi poskytujú reálny operačný systém, s kompletným prístupom, čo umožňuje honeypotu sledovať priebeh celého útoku. Tieto honeypoty sa teda zvyčajne využívajú na výskumné účely. Vysoko interaktívne honeypoty sú náročné na vývoj, pretože na ich spustenie potrebujeme mnoho nástrojov a znalostí. Známymi honeypotmi v tejto skupine sú napríklad Sebek [11] a Argos [12].

Na hranici medzi nízko a vysoko interaktívnymi honeypotmi sa nachádzajú **stredne interaktívne honeypoty**, ktorých cieľom je skombinovať výhody spomenutých dvoch skupín [6]. Táto kategória sa pri delení podľa miery interakcie často neuvádza a je spájaná s jednou zo zvyšných dvoch skupín honeypotov, pretože obsahuje len malé rozdiely. Najčastejšie je pripojená k nízko interaktívnym honeypotom (napr. pri

honeypote Kippo). Táto situácia nastáva, ak stredne interaktívny honeypot simuluje nejakú sieťovú službu (teda by sme ho mohli zaradiť k nízko interaktívnym honeypotom). Rozdiel bude v tom, že bude poskytovať niekoľko funkcionalít naviac. Príkladom takéhoto honeypotu je honeypot Kippo [1].

V Tab. 1. môžeme vidieť zhodnotenie výberu honeypotov podľa požiadaviek na systém. Z tabuľky si používateľ vie určiť, aký honeypot je pre neho vhodný. To je možné určiť na základe toho, aké požiadavky kladie na náročnosť inštalácie, údržbu, na riziká pri napadnutí honeypotu či množstvo informácií zbieraných o útočníkoch. Poradie faktorov zhora nadol naznačuje zvyšujúcu sa účinnosť honeypotu a zároveň zvyšuje zložitosť, riziko a ťažkosti pri jeho nasadzovaní.

Tab. 1 Výber honeypotu podľa požiadaviek na systém [6].

Faktory	**Nízko interaktívne honeypoty**	**Stredne interaktívne honepoty**	**Vysoko interaktívne honeypoty**
Stupeň zapojenia útočníka	Nízky	Stredný	Vysoký
Reálny operačný systém	Nie	Nie	Áno
Inštalácia	Ľahká	Ťažká	Veľmi ťažká
Údržba	Ľahká	Ľahká	Časovo náročná
Riziko	Nízke	Stredné	Vysoké
Cieľom je ohrozenie	Nie	Nie	Áno
Potrebná kontrola	Nie	Nie	Áno
Znalosti na nasadenie	Nízke	Nízke	Vysoké
Znalosti na vývoj	Nízke	Vysoké	Vysoké
Množstvo zbieraných dát	Limitované	Stredné	Rozsiahle
Interakcia útočníka s honeypotom	Emulované služby	Požiadavky	Plná kontrola

1.1.3 Delenie podľa typu nasadenia

Jednou z dvoch skupín v tejto kategórii sú **fyzické honeypoty** [6]. Fyzický honeypot je typicky jeden stroj pripojený k počítačovej sieti a prístupný cez jednu IP adresu. Tieto honeypoty sú stále spájané s konceptom vysoko interaktívneho honeypotu, keďže predstavujú fyzický stroj, počítač s operačným systémom. Fyzické honeypoty v praxi nie sú veľmi využiteľné z dôvodu vysokých nákladov na ich nasadenie a údržbu.

Druhou skupinou sú **virtuálne honeypoty** [6]. Oproti fyzickým honeypotom sú omnoho výhodnejšie z pohľadu nákladov na ich údržbu, pretože pomocou jedného fyzického stroja s jednou IP adresou vieme vytvoriť niekoľko virtuálnych honeypotov pomocou voľne dostupných nástrojov na virtualizáciu (Napr. XEN, LXC, QEMU/KVM).

1.1.4 Delenie podľa roly

Podľa roly delíme honeypoty na dve základné skupiny, a to klientske a serverové honeypoty. Ako vyplýva z ich názvu, každý typ predstavuje jednu stranu typickej architektúry komunikácie v počítačovej sieti, teda klienta a server.

Klientske honeypoty sa javia ako zraniteľné klientske aplikácie, ktoré komunikujú so serverom a snažia sa zistiť, či je server reálny a či sa nepokúša o útok [6]. Ich cieľom je teda odhaliť takéto podvodné servery. Klientsky honeypot sa zvyčajne skladá z troch častí. Prvou je komponent, ktorý sa nazýva žiadateľ (queuer), jeho úlohou je vytvoriť list dostupných serverov, na ktoré sa má honeypot pripájať. Druhou časťou honeypotu je samotný klient, ktorý je schopný posielať požiadavky na servery identifikované žiadateľom. Po interakcii so serverom, prichádza na rad tretí komponent klientskeho honeypotu, analytická časť, ktorá je zodpovedná za určenie toho, či na klienta bol uskutočnený útok. Príkladom klientskeho honeypotu je nízko interaktívny honeypot Thug [13].

Druhou skupinou sú **serverové honeypoty**, ktoré emulujú úlohu servera. Pasívne čakajú, kým sa k nim pripojí klient a ten má po pripojení k dispozícii celú jeho funkcionalitu. Serverovými honeypotmi sú zvyčajne nízko interaktívne honeypoty predstavujúce sieťové služby, na ktoré sa pripájajú klienti. Väčšina typov honeypotov je serverová. Typickými serverovými honeypotmi sú Kippo a Honeyd.

1.2 Honeynet

Honeynet je sieť tvorená dvomi alebo viacerými honeypotmi [6]. Honeynet je zvyčajne využívaný na rovnaký účel ako honeypot, avšak pri rozsiahlejších sieťach, kde by jeden honeypot nebol dostatočný. Na rozdiel od honeypotu je teda honeynet fyzická sieť niekoľkých systémov. Architektúru honeynetu definujú tri základné elementy [6]:

- kontrola toku údajov (data control),
- zachytávanie údajo (data capture),
- zber údajov (data collection)

Kontrola toku údajov (data control) je v honeynete činnosť, ktorá znižuje riziko [6]. Znamená to, že honeynet kontroluje útočníkovu aktivitu tým, že obmedzuje čo sa môže stať, k čomu útočník má a nemá prístup. Riziko nastáva, ak sa útočník dostane do honeynetu a môže nastať situácia, že prostredníctvom neho sa vie dostať aj do reálneho systému v ktorom je honeynet nasadený. Útočník preto musí byť kontrolovaný honeynetom a mať obmedzený prístup do niektorých častí systému.

Druhou požiadavkou pre honeynety je **zachytávanie údajov (data capture)** [6]. Dôležitým faktorom v tejto fáze je, že útočník si nesmie byť vedomý toho, že niekto sleduje jeho aktivitu a nemôže byť schopný prísť na to, že sa nachádza v honeynete. Zdroje použité na zachytávanie údajov musia byť zabezpečené tak, aby údaje nemohli byť kompromitované (aby nemohla byť narušená ich integrita). Táto práca s údajmi zahŕňa mnoho krokov, ako je napríklad uskladňovanie údajov na inom mieste než priamo na honeynete, ukladanie všetkých zozbieraných údajov, či vzdialený prístup k údajom pre administrátora.

Treťou požiadavkou na architektúru honeynetu je **zber údajov (data collection)** [6]. Tento krok je dôležitý najmä pri rozsiahlych honeynetoch, kedy údaje o útočníkovi zbiera viacero nasadených honeynetov, z čoho je potom potrebné získať súhrnné informácie. Zber údajov vyžaduje štandardný formát údajov (zhodný pre všetky honeynety nasadené v počítačovej sieti) a zabezpečený prenos zozbieraných údajov od jednotlivých honeynetov k zdroju, ktorý tieto údaje následne vyhodnocuje.

Okrem zachytávania útokov na počítačovú sieť a ich monitorovania, význam honeynetov spočíva aj v tom, že sa môžu použiť ako testovacie prostredia. Honeynet ako

kontrolované prostredie (administrátorom) môže byť využiteľný na analýzu zraniteľností v nových aplikáciách alebo operačných systémoch. Prínosom teda je, že bezpečnostné riziká zistené honeynetmi sa dajú vyriešiť skôr, ako sa technológie zavedú do produkčného prostredia.

2 Podvodné systémy

V oblasti informačných technológií sa často spomína pojem podvod. **Podvod** je úspešný pokus o to, aby niekto uveril niečomu, čo je nepravdivé, a to buď úmyselne alebo neúmyselne. Pod podvodom môžeme rozumieť aj činnosť, ktorá využíva cudziu nevedomosť, či dôvernosť k vlastnému prospechu.

V súvislosti s informačnou bezpečnosťou používame pojem **kybernetický podvod** [14]. Je to podvod, ktorý sa vyskytuje v kybernetickom priestore. Takýto podvod môže byť v útočnom zmysle (napadnúť niekoho) alebo v zmysle obrannom (obrana proti útokom). Ofenzívne podvody majú vo zvykoch používať obmedzenú množinu metód, ako je napríklad vydávanie sa za druhú osobu, ale vždy s drobnými zmenami. Obranné podvody môžu a mali by byť rozmanitejšie. V kybernetickom priestore zvyčajne chápeme obranný podvod ako spôsob obrany proti útoku. Inými slovamia, ide o aktívnu obranu voči útočníkovi.

Každý kybernetický podvod začína tým, že si jeho iniciátor (aj keď možno nevedome) vyberie metódu, akou bude podvod vykonávať. Podvodné metódy môžeme rozdeliť do šiestich základných kategórií [14]:

- **maskovanie (masking)** – systém skrýva nejaký proces, resp. činnosť na pozadí – napríklad monitorovanie užívateľa,
- **obalenie (repackaging)** – niečo sa skrýva ako niečo iné - napr. vloženie malvéru do bežného programu,
- **zatienenie (dazzling)** – systém niečo skrýva tým, že to „zatieni“ inou činnosťou – napr. posielanie mnohých chybových správ útočníkom, ktorí vykonávajú škodlivú činnosť,
- **napodobňovanie (mimicking)** – systém napodobňuje niečo iné – napríklad podvrhnutý súborový systém,
- **vynájdenie (inventing)** – systém vytvára stále nové objekty (často falošné) na nalákanie útočníka,
- **návnada (decoying)** – napríklad podhodenie prihlasovacích údajov.

Honeypoty sa považujú za najznámejší a najrozšírenejší spôsob podvodných technológií v oblasti informačných technológií. Podvodné metódy môžeme napasovať na

činnosti, ktoré honeypoty vykonávajú. Qassrawi a Hongli v článku [15] uvádzajú príklady využitia podvodných metód v oblasti honeypotov:

• **maskovanie** - príkladom môže byť monitorovanie používateľov, tým že modifikujú operačný systém tak, aby sa skryli jeho stopy,

• **obalenie** – príkladom môže byť vloženie softvéru, ktorý zmarí útok, do navonok bezpečne pôsobiacej časti operačného systému,

• **zatienenie** – za túto metódu môžeme považovať posielanie mnohých chybových hlášok útočníkovi,

• **napodobňovanie** - príkladom je vybudovanie falošného súborového systému, ktorý vyzerá ako súborový systém zaneprázdneného používateľa a jeho cieľom je presvedčiť útočníka, že daný systém nie je honeypot,

• **vynájdenie** - príkladom môže byť ponechanie nejakého softvéru v honeypote, ktorý si útočník stiahne, a tým mu umožní sledovať jeho osobné údaje a aktivitu,

• **návnada** - je to napríklad zámerné ponechanie hesiel v súborovom systéme, čo nabáda útočníka aby ich použil.

Podľa toho, aký typ podvodu využíva daný honeypot, vieme dopredu určiť, aké typy podvodu bude využívať. Po zistení, akú podvodnú metódu honeypot využíva, vieme určiť, akým spôsobom bolo možné honeypot detekovať.

2.1 Maskovanie

Cieľom tohto typu podvodu je zamaskovať, skryť vlastnosti reálneho objektu alebo činnosti, či už pred používateľom alebo pred samotným systémom. Maskovanie má zabezpečiť, že príslušný objekt nebude možné detegovať. Honeypoty pri svojej činnosti často využívajú túto techniku, pretože ich cieľom je aby neboli detekovateľné, teda často skrývajú, maskujú svoju činnosť. Príkladom honeypotov, ktoré využívajú maskovanie sú ADBHoney [35], ElasticPot [43], Glutton [45] a mnoho ďalších. Maskovanie je spôsob podvodu, ktorý môžeme najčastejšie pozorovať pri správaní honeypotov, pretože patrí k ich základným vlastnostiam.

2.2 Obalenie

Pri technike obalenia, resp. obaľovania je realita skrytá takým spôsobom, aby objekt vyzeral rozdielne od toho aký je v skutočnosti. Typickým príkladom je situácia, kedy sa útok javí ako priateľský e-mail s reálnou hlavičkou, aby nalákal prijímateľa na otvorenie správy či prílohy [25].

Hoci útočník môže túto techniku použiť aby oklamal používateľa, technika obalenia sa môže použiť aj ako obranný mechanizmus. Pri používaní honeypotov nie je zriedkavé, že honeypot ktorý predstavuje reálny systém, má prednastavený falošný súborový systém, či takzvané „honey files“. Sú to súbory, ktorých úlohou je vyzerať ako bežné používateľské súbory, avšak slúžia ako alarm pre systémového administrátora, pretože monitorujú ak s nimi útočník akokoľvek pracuje. Tieto súbory môžu byť rovnako vytvárané aj útočníkmi, pričom zvyčajne mávajú názvy, ktoré majú za úlohu nalákať používateľa na ich otvorenie [25]. Obalenie ako spôsob podvodu využíva napríklad honeypot Glastopf [44], tváriaci sa ako zraniteľný systém, pričom v skutočnosti zraniteľnosti neobsahuje.

2.3 Zatienenie

Zatienenie je technika, ktorej úlohou je zmiasť používateľa. Vyznačuje sa obfuskáciou a randomizáciou objektov identifikovaných v systéme [25]. Zatienenie sa snaží skryť realitu tým, že používa činnosti nesúvisiace so skrývaným objektom, s cieľom zmiasť používateľa. Pri vyššie spomínanom posielaní mnohých chybových hlášok, je ich cieľom zatienenie pravej činnosti, teda napríklad toho, čo v systéme beží na pozadí. Techniku zatienenia používajú honeypoty Dionaea [42] a takisto Glastopf.

2.4 Napodobňovanie

Rovnako ako pri zatienení, aj do tejto techniky môžeme zahrnúť podvrhnutý súborový systém, ktorý vytvárajú honeypoty. Napodobňovanie je technika, ktorá simuluje reálne črty objektu, za ktorý sa snaží vydávať. Iným príkladom sú napríklad honeypoty predstavujúce konkrétnu službu, ktoré ju napodobňujú no popritom monitorujú správanie sa útočníka ktorý ju používa (napríklad zaznamenávanie konkrétnych príkazov). Za napodobňovanie sa tiež považuje podvrhnutá webová stránka,

ktorá je v skutočnosti vytvorená útočníkom a od reálnej stránky sa líši len koncovkou, či typom šifrovania. Napodobňovanie využíva mnoho honeypotov, medzi inými aj ADBHoney, Conpot [40], Cowrie [41], či Medpot [50].

2.5 Vynájdenie

Tento typ podvodu sa vyznačuje tým, že vytvára dojem že daný objekt existuje, ale ten je pritom nereálny, falošný. Pri honeypotoch sa táto technika vyskytuje napríklad vtedy, keď honeypot predstavuje sieť so špecifickými adresami, ale v skutočnosti žiadna z nich neexistuje [25]. Ďalším príkladom môže byť podvrhnutie súboru či programu, ktorý monitoruje útočníka, ktorý si ho stiahne či skopíruje. Z honeypotov techniku vynájdenia používa napríklad Conpot a RDPY [53].

2.6 Návnada

Posledný typ podvodu nazývaný návnada, rovnako ako aj pôvodný význam slova je spôsob ako odlákať pozornosť od relevantných objektov, či služieb. Carroll a Grosu v [26] popisujú, ako funguje interakcia medzi útočníkom a „obrancom“ počítačovej siete, pričom uvádzajú rozdiely medzi dvoma prípadmi – reálny systém tváriaci sa ako honeypot a naopak, honeypot tváriaci sa ako reálny systém. Honeypoty teda môžu slúžiť organizáciám ako návnady pre útočníkov, pretože donútia útočníka myslieť si, že jeden zo systémov danej spoločnosti je zraniteľný, čím prilákajú jeho pozornosť a zároveň ju odvrátia od systémov reálnych. Návnadu pri svojej funkcionalite používajú honeypoty Cisco ASA [37], Cowrie, HoneyPy [47], HoneyTrap [48] a mnoho ďalších.

3 Prehľad existujúcich riešení

V existujúcich prácach o detekcii honeypotov sa nachádzajú rôzne nezhody v otázke delenia honeypotov. Každý autor rozdeľuje honeypoty podľa rôznych kritérií a od toho sa zvyčajne odvíja aj technika detekcie použitá na odhalenie honeypotu. V mnohých prípadoch ide o detekciu konkrétneho honeypotu, nie typ honeypotu (napríklad všeobecne klientsky honeypot). Tieto riešenia obmedzujú možnosti detekcie honeypotov vo všeobecnej rovine. Cieľom našej práce je preto nájsť spôsob, ako detegovať honeypoty v čo najširšom zmysle, vo všeobecnej rovine a nie podľa konkrétnych kritérií.

3.1 Detekcia vysoko interaktívnych virtuálnych honeypotov

Príkladom nástroja na detekciu virtuálnych honeypotov je **VMScope** [16]. Je to monitorovací systém založený na virtualizácii, ktorý má rovnakú schopnosť hĺbkovej kontroly systému ako existujúce interné monitorovacie nástroje, pričom je rovnako transparentný a odolný voči neoprávneným zásahom ako externé monitorovacie nástroje. VMscope je odolný voči manipulácii a je transparentný pre monitorovaný systém. Okrem toho, bez potreby akejkoľvek modifikácie monitorovaného systému, VMscope je schopný pozorovať a zaznamenávať parametre a sémantiku rôznych udalostí virtuálneho systému vrátane systémových volaní. Umožňuje hĺbkovú kontrolu virtuálnych honeypotov bez toho, aby sa vo vnútri umiestnili akékoľvek senzory.

Ďalším nástrojom je **Sebek** [17], ktorý slúži na odchytávanie údajov, určený na zachytenie činností útočníka na honeypote, bez toho, aby o tom útočník vedel. Skladá sa z dvoch komponentov. Prvým z nich je klient, ktorý beží na honeypote, jeho účelom je zachytiť všetky aktivity útočníka (stlačenia klávesov, nahrávanie súborov, heslá) a následne tieto dáta odoslať na server. Ten je teda druhou zložkou a jeho úlohou je zhromažďovať údaje z jednotlivých honeypotov.

Jiang a Wang v [16] uvádzajú, že existujúce prístupy na monitorovanie virtuálnych honeypotov možno rozdeliť do dvoch hlavných kategórií, a to na interné a externé prístupy. Externé monitorovanie zostáva pre monitorovaný honeypot neviditeľné, ale za cenu straty schopnosti zachytiť interné systémové udalosti, ako sú napríklad vykonané systémové volania.

Na druhej strane, interné monitorovanie nasadzuje senzory vo vnútri monitorovaných honeypotov, a tým poskytuje rozsiahly pohľad na rôzne aspekty systému. Senzory vo vnútri honeypotov však môže útočník detegovať a deaktivovať

3.2 Detekcia nízko interaktívnych virtuálnych honeypotov

Mukkamala a spol. v článku [18] uvádzajú, že nízko interaktívne honeypoty môžeme detegovať na úrovni počítačovej siete alebo pomocou takzvaného TCP/IP fingerprintingu.

Pri detekcii na úrovni počítačovej siete, napríklad u honeypotu Honeyd [19] sa dá využiť časová analýza ICMP ECHO požiadavky. Detekčná technika je založená na jednoduchom pozorovaní, a síce že väčšina honeypotov odpovedá pomalšie na požiadavky ICMP ECHO (ping) v porovnaní s klasickými systémami.

TCP/IP fingerprinting, tiež známy ako odtlačok TCP/IP stack fingerprinting, je analýza dátových polí v pakete TCP/IP na identifikáciu rôznych konfiguračných atribútov sieťového zariadenia. Informácie, ktoré sa z tohto dajú vyčítať, zahŕňajú typ zariadenia, z ktorého paket pochádza a operačný systém, na ktorom je spustený. Pri analýze je pre každé TCP/IP pripojenie vyextrahovaných 49 rôznych kvalitatívnych a kvantitatívnych znakov (napr. sent_packets, received_packets, total_packets). Z výsledkov prezentovaných v článku [18] je zrejmé, že zatiaľ čo Honeyd implementuje základnú funkcionalitu služby, zlyháva, keď sa službu skutočne pokúsime využiť. Autori článku testovali služby HTTP, FTP a SMTP, pričom porovnávali Honeyd a reálne systémy (Linux aj Windows) s rôznymi požiadavkami na tieto služby. Výsledky, ktoré získali, môžeme vidieť v nižšie uvedenej tabuľke. Znak začiarknutia označuje funkciu (príkaz), ktorý bol prítomný, a krížik označuje, že funkcia sa nenašla, teda ju honeypot nepodporoval. Ako môžeme vidieť v Tab. 2., Honeyd zlyhal v nadpolovičnej väčšine funkcií pri testovaní služieb HTTP a SMTP a pri službe FTP podporoval presne polovicu odskúšaných funkcií.

Tab. 2 Porovnanie honeypotu Honeyd a reálneho systému

Služba	Príkaz	Reálny systém	Honeyd
HTTP	GET	✓	✓
	OPTIONS	✓	×
	HEAD	✓	×
	TRACE	✓	×
FTP	USER	✓	✓
	PASS	✓	✓
	MODE	✓	×
	RETR	✓	×
SMTP	HELO	✓	✓
	MAIL	✓	✓
	DATA	✓	×
	VRFY	✓	×
	ETRN	✓	×

3.3 Detekcia testovaním simulovaných služieb

Podobne ako v predchádzajúcej podkapitole, aj Krawetz v článku [20] predstavil myšlienku identifikovať honeypoty tým, že skúmal funkčnosť simulovaných služieb. Vo svojej práci navrhol poslať si e-maily od honeypotu, ktorý sa tvári ako SMTP server. Ak nie sú prijaté žiadne e-maily, navrhovaná funkcionalita nie je dostupná a prostredie systému je podozrivé, teda to môže byť nasadený honeypot.

Fu a spol. vo svojej práci [21] predstavili niekoľko metód na detekciu virtuálnych honeypotov, ako je Honeyd, založených na časovom správaní. Používajú Ping, TCP a UDP na určenie času od odoslania segmentu až po doručenie príslušného potvrdenia (round trip time - RTT).

Dahbul a spol. v článku [22] použili modelovanie hrozieb na identifikáciu potenciálnych hrozieb, ktoré odhaľujú existenciu honeypotu, čo ho robí neúčinným. V práci sa rozoberajú rôzne protiopatrenia, ktoré sa ukázali ako účinné na zvýšenie

neidentifikovateľnosti honeypotov. Tieto protiopatrenia aj v práci otestovali na honeypotoch Glastopf, Honeyd, Kippo a Dionaea.

3.4 Detekcia fyzickej prítomnosti honeypotu

Innes a Valli v článku [23] uvádzajú rôzne metódy, ako detegovať honeypot. Prvou z nich je postup, ako detegovať VMWare honeypot (virtuálny honeypot vytvorený pomocou nástroja VMWare). Pred verziou VMWare 4.5, veľké množstvo hardvéru nebolo konfigurovateľné. To znamená, že sa honeypot dal veľmi ľahko odhaliť pomocou predvolenej konfigurácie. Ďalšou chybou je MAC adresa sieťovej karty, pretože časť výrobcu MAC adresy (prvé tri oktety) na virtuálnom sieťovom rozhraní vo VMWare je vždy jedna z konkrétnych troch hodnôt. Teda jednoduchým nástrojom na zistenie MAC adresy (napr. ipconfig v operačnom systéme Windows alebo ip v operačnom systéme Linux) zistíme, či začiatok MAC adresy je zhodný s danými hodnotami a odhalíme tak honeypot. Poslednou slabinou VMWare honeypotov je konfigurácia virtuálneho stroja počas jeho behu. Útočník by mohol potenciálne spustiť tento kód na konfiguráciu a zistiť, že príkaz, ktorý sa pokúša vykonať bol úspešný, čím by si overil že sa nachádza vo virtuálnom prostredí VMWare (pretože v klasickom prostredí by tieto príkazy nefungovali).

Ďalšou z metód, ktorú uvádzajú autori práce, je detekcia chroot prostredia v honeypote. Chroot [24] na operačných systémoch Unix je operácia, ktorá zmení zdanlivý koreňový adresár na aktuálny proces a pre procesy, pre ktorý je proces rodičom. Program, ktorý je spustený v takomto modifikovanom prostredí, nemôže pomenovať (a teda normálne nemôže pristupovať) k súborom mimo určeného adresárového stromu. Termín chroot sa môže vzťahovať na systémové volanie chroot(2) alebo na program chroot(8). Jedným zo spoločných spôsobov zabezpečenia honeypotov je používanie prostredia chroot. Najjednoduchším spôsobom, ako zistiť, či sa nachádzame v prostredí chroot, bolo spustiť príkaz ls –lia v koreňovom adresári. Treba si všimnúť, že tento príkaz nám ukáže inode (index node) koreňových adresárov „.“ a „..“. Na štandardnom systéme sme mohli vidieť, že inode-y pre tieto dva adresáre sú obe číslo 2. Ak sme ale boli v chroote a spustili by sme rovnaký príkaz, inode-y by boli úplne odlišné (čísla budú rôzne od 2).

Autori sa pokúšali detegoať aj honeypot Honeyd. Ako uvádzajú, prvý hlavný problém s Honeyd je v tom, že má tendenciu odpovedať na pakety, na ktoré by zvyčajne reagovať nemal. Ak mu niekto pošle chybne vytvorený sieťový paket (napríklad paket, ktorý okamžite otvorí a zatvorí spojenie), Honeyd odpovie ako keby prijal platný paket. Toto je však zvyčajne prehliadnuté, keďže útočník je zvyknutý neprijímať odpovede, avšak aj táto vlastnosť môže byť zneužitá na odhalenie honeypotu.

V článku autori súčasne popisujú aj niektoré všeobecné problémy, ktoré nastávajú pri detekcii honeypotov ako takých. Vzhľadom k tomu, že honeypoty zapisujú logy, keď útočník používa zariadenie, vykonávajú sa niektoré inštrukcie navyše a výsledkom je, že celkový čas vykonania procesu alebo príkazu by sa mohol rozšíriť až tak, že je zrejmé, že sa deje niečo podozrivé. Okrem toho, honeypoty často trpia výrazným zhoršením výkonu, ak sú podrobené vytrvalému útoku a v niektorých prípadoch môžu zlyhať úplne (napr. honeypot Kippo).

3.5 Detekcia UML honeypotu

Innes a Valli v článku [23] predstavujú spôsob, ako detegovať UML honeypot (User Mode Linux, teda linuxové jadro, ktoré môže byť spúšťané zo systému na ktorom beží Linux). UML, teda operačný systém Linux v používateľskom móde, má niekoľko problémov, ktoré môžu byť ľahko zneužité, ak ich útočník pozná. Napríklad, UML nepoužíva reálny disk na ukladanie údajov, ale virtuálne zariadenie, ktoré ukazuje na existujúci súborový systém, ktorý obsahuje obraz disku. Pomocou jednoduchých príkazov v príkazovom riadku Linuxu sa vieme pozrieť, či systém obsahuje takýto virtuálny disk. Týmto spôsobom vieme získať prvý náznak toho, že systém môže obsahovať UML honeypot.

V neskorších vydaniach UML bolo teda snahou vývojárov znemožniť odhalenie tohto virtuálneho disku. To však nepomohlo k zabráneniu toho, aby útočník zistil, že sa nachádza v UML prostredí. Toto prostredie obsahuje ďalšie vlastnosti, ktoré môžu byť zneužité na jeho odhalenie. Iný rýchly spôsob, ako ho odhaliť, by bolo napríklad pozrieť sa do adresára /proc/cpuinfo, ktorý ak si dáme vypísať do konzoly, nám vypíše zopár riadkov, z ktorých jeden obsahuje „model name: UML“, čím sme dosiahli náš cieľ a odhalili sme UML prostredie.

3.6 Nástroje na detekciu honeypotov

V nasledujúcich podkapitolách si predstavíme nástroje, pozostávajúce zo skriptov určených na odhalenie rôznych honeypotov. Všetky tieto nástroje sú napísané v jazyku Python a niektoré ich zdrojové kódy sú zverejnené v službe Github. Takéto projekty sú určené na odstránenie chýb, ktoré obsahujú rôzne honeypoty, zneužitím ktorých sa dajú v systéme odhaliť. Ich cieľom je otestovanie prostredia s honeypotom a následné odstránenie týchto chýb.

3.6.1 Checkpot Honeypot Checker

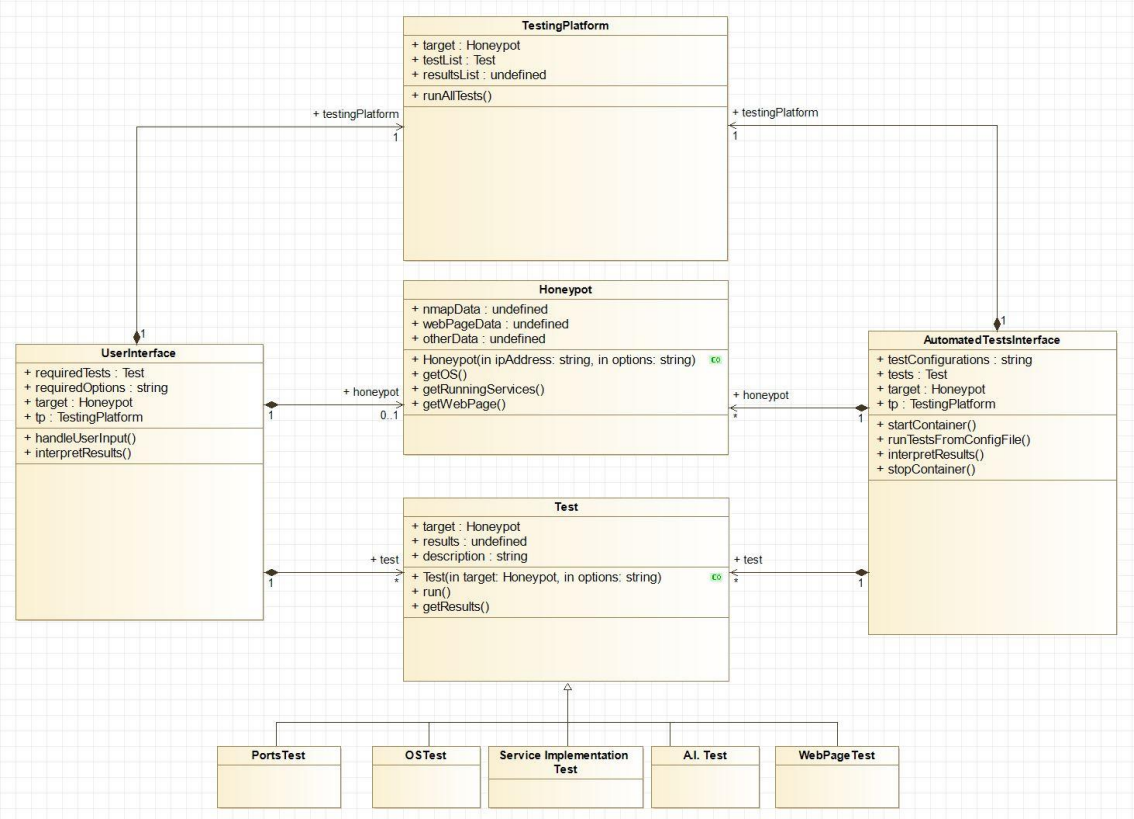
Prvým nástrojom, ktorý si priblížime je Checkpot [57]. Je to nástroj na zisťovanie chýb v konfigurácii honeypotov. Najčastejším dôvodom na odhalenie honeypotu, je ponechanie konfiguračného súboru v pôvodnom, nezmenenom stave.

Checkpot je najčastejšie využívaný na zabezpečenie honeynetu, na overenie toho či sú jednotlivé honeypoty v ňom správne nastavené a nedetekovateľné. Jeho úlohou je preskúmať systém, ktorého IP adresa je mu zadaná, vytvoriť správu so zisteniami a ich závažnosťou. Pomocou tohto nástroja môžu používatelia skontrolovať svoj systém pred uvedením do prevádzky a na základe výsledkov poprípade opraviť chyby, ktoré sa v ňom nachádzajú.

Nástroj Checkpot si po jeho spustení rozdelí vstupné parametre zadané používateľom, z ktorých najdôležitejšia informácia je IP adresa systému, ktorý má za úlohu testovať. Checkpot následne v danom systéme skenuje rôzne jeho vlastnosti, ako napríklad otvorené porty, webstránky, bežiace procesy a iné. Po úvodnom skenovaní prichádza na rad najdôležitejšia fáza, spustenie testov. Jednotlivé testy skúmajú detekovateľnosť honeypotov, pričom využívajú rôzne známe chyby v defaultných konfiguráciách, či nastaveniach. Do Chechpot-u je možné pridávať aj vlastné testy, či upravovať tie, ktoré sa v ňom už nachádzajú, je teda prispôsobiteľný pre rôzne potreby používateľa. Na Obr. 1 [58] je zobrazený UML diagram vyjadrujúci priebeh testovania v nástroji Checkpot.

Checkpot v jeho najnovšej verzii je schopný odhaliť 13 konkrétnych honeypotov (medzi inými aj najčastejšie používané Conpot, Cowrie, Dionaea, Glastopf a iné) pomocou 15 prednastavených testov.

Obr.1 Testovanie honeypotu v nástroji Checkpot [58]



3.6.2 Honeybee

Ďalším nástrojom na detekciu honeypotov je Honeybee [59]. Je o niečo menej rozsiahly ako Checkpot a je rovnako napísaný v jazyku Python. Pozostáva z 5 skriptov, ktorých úloh je odhalenie honeypotov Amun [60], Glastopf [44] a Kippo [1].

Jeho prvotnou funkcionalitou je podobne ako u nástroja Checkpot skenovanie siete zadanej ako vstupný parameter, teda prehľadanie otvorených portov, bežiacich procesov a podobne. Následne pomocou jednotlivých skriptov Honeybee hľadá v systéme prítomnosť spomínaných 3 honeypotov na základe ich vlastností, pomocou ktorých sú tieto honeypoty detekovateľné.

3.6.3 Honeyscore

Honeyscore [61] je webová služba respektíve nástroj, ktorý je súčasťou bezpečnostnej služby Shodan [62]. Shodan používateľovi umožňuje nájsť rôzne typy systémov pripojených k internetu (napríklad webové kamery, servery, smerovače a

podobne). Shodan na rozdiel od bežných vyhľadávačov neprehľadáva len webové stránky ale skenuje celú počítačovú sieť, kontroluje teda otvorené porty, známe služby či bežiace procesy.

Honeyscore je časť Shodanu, ktorej úlohou je identifikácia honeypotu na danej IP adrese, ktorú jej zadáme. Tento nástroj bol vytvorený na základe známych charakteristík rôznych honeypotov. Honeyscore po preskúmaní IP adresy podá vyhlásenie o tom, či sa podľa neho na danej adrese nachádza honeypot alebo reálny systém.

4 Návrh a implementácia nástroja

V predchádzajúcich kapitolách sme popísali honeypoty a podvodné systémy. Ako sme si ukázali, tieto dve témy sú úzko prepojené, pretože podvodné metódy, ktoré sa využívajú kybernetickom priestore, sú podstatou aktivít vykonávaných honeypotmi. Cieľom našej práce je návrh a implementácia prostredia, resp. nástroja na detekciu honeypotov. V podobných prácach, ktoré sme rozobrali, je vidieť, že neexistuje žiadna všeobecná metóda na detekciu honeypotov. V našom riešení je cieľom nájsť spôsob, ako detegovať honeypoty podľa typov podvodu, ktoré používajú.

Pri detekcii honeypotov je potrebné, aby nástroj, ktorý vytvoríme bol schopný detegovať, resp. identifikovať v počítačovej sieti ľubovoľný (vopred neznámy) honeypot. Cieľom je vytvoriť program v jazyku Python, ktorý na vstup dostane IP adresu a na základe naprogramovaných vlastností a metód, určí či táto IP adresa je adresou honeypotu, alebo ide o štandardnú sieťovú službu, resp. operačný systém. V tomto programe budeme jednotlivé systémy (teda vstupné IP adresy) určené na analýzu rozdeľovať do rôznych podvodných metód, ktoré sme uviedli v článku. Predtým, ako budeme testovať reálne systémy, sme sa pozreli na zoznam najčastejšie sa vyskytujúcich honeypotov. Tieto honeypoty si rozdelíme do jednotlivých podvodných metód a na základe tohto delenia budeme vedieť pracovať so systémami, o ktorých dopredu nebudeme vedieť, či sú honeypotmi alebo ide o reálny systém.

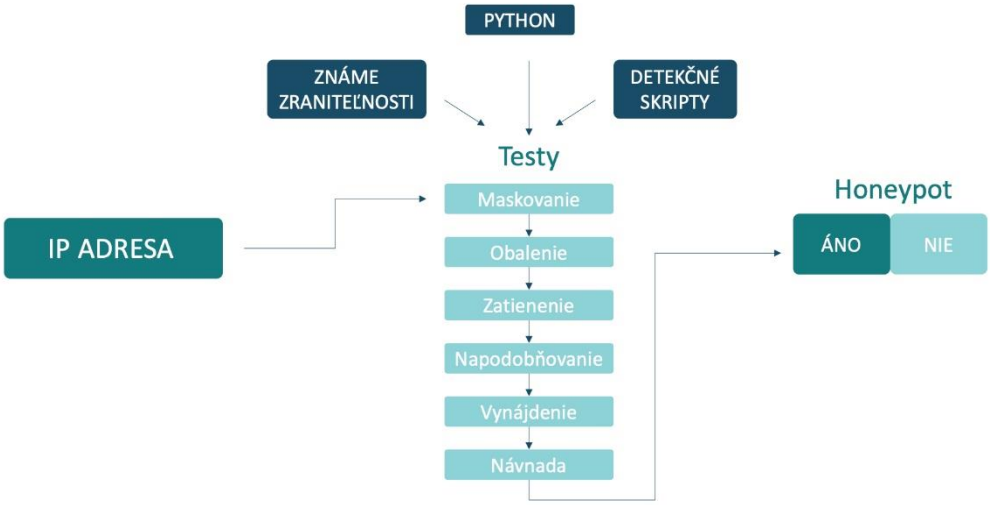
Pri prideľovaní honeypotov k šiestim základným podvodným metódam, môže nastať situácia, že jeden honeypot bude spĺňať popis niekoľkých podvodných metód. To nám však nevadí, keďže pri analýze systému, ktorý dostane program na vstup, pre nás nebude dôležité, akú konkrétnu podvodnú metódu systém využíva. Dôležité pre nás bude, že využíva ľubovoľnú z týchto metód, čím budeme vedieť jednoznačne povedať, že sa jedná o podvodný systém.

Po rozdelení honeypotov bude nasledovať tvorba nástroja. Tento program musíme navrhnúť tak, aby bez znalosti vstupu (teda pridelená IP adresa môže ale nemusí byť honeypot) bol schopný rozoznať tento systém. Pritom platí, že nás pri výstupe programu bude zaujímať, aký typ honeypotu daný systém predstavuje. Dôležité bude aj to, že program správne označí systém za podvodný.

4.1 Návrh nástroja na detekciu honeypotov

Nástroj na detekciu honeypotov, ktorý bude výsledkom našej práce bude detekovať honeypoty na základe ich vlastností ako podvodných systémov. Tento nástroj bude napísaný v jazyku Python a na detekciu bude využívať vlastnosti typov podvodov. Na Obr. 2 je znázornená štruktúra nástroja, ktorý budeme implementovať.

Obr. 2 Návrh nástroja na detekciu honeypotov



Po zadaní IP adresy do nášho nástroja, bude nasledovať prvotný sken daného systému, teda kontrola spustených služieb, otvorených portov a podobne. Následne táto adresa bude otestovaná skriptami v jazyku Python, ktoré budú navrhnuté podľa vlastností jednotlivých spôsobov podvodov. Tieto vlastnosti popíšeme a implementujeme na základe už známych zraniteľností jednotlivých honeypotov a známych detekčných skriptov a nástrojov. Po analýze vlastností systému zo vstupu, nástroj rozhodne o tom, či na danej IP adrese sa nachádza podvodný systém, alebo systém reálny. Pri príprave na implementáciu návrhu sme preskúmali výhody a nevýhody existujúcich riešení na detekciu honeypotov a zozbierali sme rôzne detekčné skripty, ktoré nám pomôžu pri návrhu jednotlivých modelov pre každý typ podvodu.

4.2 T-Pot

V tejto podkapitole si predstavíme prostredie, na ktorom budeme testovať náš nástroj na detekciu honeypotov. T-Pot [27] je nástroj pochádzajúci z projektu Deutsche Telekom AG Community Honeypot Project [28]. T-Pot je platforma obsahujúca viacero honeypotov, prepojených pomocou Dockeru, čo je open source projekt poskytujúci rozhranie využívajúce izoláciu jednotlivých aplikácií (v tomto prípade honeypotov) do takzvaných kontajnerov [29][30]. V nasledujúcich podkapitolách si priblížime koncept nástroja T-Pot, popíšeme aké honeypoty obsahuje a pozrieme sa na ich znaky podvodných systémov.

4.2.1 Štruktúra nástroja

Ako sme uviedli v úvode kapitoly, T-Pot využíva Docker na správu jednotlivých honeypotov, ktoré obsahuje. Jeho výhodou je, že honeypoty sú uložené v kontajneroch, ktoré zabezpečujú izoláciu týchto systémov. Jednotlivé kontajnery obsahujú honeypot a potrebné súbory pre jeho chod, ale neobsahujú žiadny operačný systém, čo zabezpečuje nižsie náklady na prevádzku a taktiež úspora miesta.

Ďalšou dôležitou vlastnosťou T-Potu je, že je založený na takzvanom ELK Stack [31]. ELK je skratka pre 3 open source projekty, a to Elasticsearch [32], Logstash [33] a Kibana [34]. Elasticsearch je distribuovaný, open source, vyhľadávací a analytický nástroj pre všetky typy údajov, v našom prípade údajov zozbieraných z jednotlivých honeypotov. Nástroj Logstash slúži na spracovanie dát na strane servera, pričom zozbierava dáta z viacerých zdrojov (teda kontajnerov obsahujúcich honeypoty) a následne ich odosiela ďalej do Elasticsearch. Kibana umožňuje vizualizáciu dát zozbieraných pomocou Elasticsearch a taktiež jednoduchú navigáciu vo webovom rozhraní nástroja T-Pot.

4.2.2 Honeypoty v nástroji T-Pot

T-Pot v jeho najnovšej verzii 19.03 obsahuje spolu 16 honeypotov rozdelených do jednotlivých kontajnerov v Dockeri. V tejto kapitole si uvedieme, na čo tieto honeypoty slúžia a tiež do ktorých typov podvodných systémov ich zaraďujeme.

ADBHoney [35] je nízko interaktívny honeypot navrhnutý pre protokol Android Debug Bridge (ADB) [36]. ADB je protokol určený na sledovanie emulovaných či

skutočných zariadení pripojených k danému používateľovi. Je to nástroj fungujúci v príkazovom riadku, honeypot ADBHoney teda simuluje tento nástroj. O ADBHoney môžeme teda povedať, že využíva napodobňovanie. Tento honeypot môžeme rovnako zahrnúť aj do maskovania, keďže maskuje to že je honeypot tým. že je schopný vykonať niektoré konkrétne príkazy reálneho nástroja ADB (napríklad príkaz `adb connect`) [35].

Ďalším z honeypotov v kolekcii je nízko interaktívny **Cisco ASA honeypot** [37]. Cisco Adaptive Security Appliance (alebo skrátene Cisco ASA) je rodina bezpečnostných nástrojov chrániacich siete a dátové centrá rôznych spoločností [38]. Honeypot Cisco ASA je schopný detegovať zraniteľnosť CVE-2018-0101 [39], ktorá umožňuje útočníkovi bez autentifikácie vzdialene vykonať ľubovoľný kód a taktiež deteguje záplavový útok DoS (odmietnutie služby). Z hľadiska typov podvodných systémov ho môžeme zaradiť k návnade, pretože tváriac sa ako zraniteľný nástroj, láka útočníka na jeho zneužitie.

Tretím v poradí je serverovský **Conpot** [40], ktorý je nízko interaktívny, ako predchádzajúce dva honeypoty. Conpot zahrňuje rôzne bežne používané protokoly využívané v priemysle, pomocou ktorých pred útočníkom simuluje štruktúru priemyselného kontrolného systému. Tento honeypot z pohľadu podvodných systémov využíva napodobňovanie lebo simuluje reálny systém, vynájdenie, pretože predstavuje systém ktorý nie je reálny a taktiež návnadu, keďže jeho cieľom je tváriť sa ako systém, ktorý sa dá zneužiť.

Na rozdiel od prechádzajúcich, ďalší z honeypotov **Cowrie** [41], je stredne až vysoko interaktívny honeypot, simulujúci služby SSH a telnet. Je určený na zaznamenávanie konzolovej interakcie útočníka a tiež útokov hrubou silou na prelomenie hesla. Cowrie zaraďujeme pod typy podvodu napodobňovanie (reálnej služby) a taktiež návnadu (prihlasovanie pod heslom).

Nasledujúcim honeypotom je **Dionaea** [42], honeypot ktorý zachytáva malvér využívajúci zraniteľné miesta služieb ponúkaných cez sieť a snaží sa pritom získať kópiu tohto malvéru. Dionaea využíva zatienenie, keďže schválne útočníkovi poskytuje známe typy zraniteľností, s cieľom aby ich zneužil. Tento honeypot môžeme tiež priradiť k vynájdeniu, pretože sa tvári že obsahuje zraniteľnosť, ktorá sa však reálne v sieti nenachádza.

Súčasťou kolekcie je tiež nástroj **ElasticPot** [43], čo je jednoduchý honeypot zameraný na zaznamenávanie útokov na vyhľadávací a analytický nástroj Elasticsearch,

ktorý je taktiež súčasťou kolekcie T-Pot. ElasticPot využíva ako podvod hlavne maskovanie, keďže jeho úlohou je bez interakcie sledovať komunikáciu útočníka s nástrojom Elasticsearch.

Ďalším honeypotom v T-Pote je **Glastopf** [44], honeypot pre webové aplikácie napísaný v Pythone. Namiesto napodobňovania konkrétnej zraniteľnosti vo webovej aplikácii, napodobňuje typ zraniteľnosti, je teda viac všeobecný, čo zvyšuje jeho šancu zachytiť útočníka. Glastopf vo veľkej miere využíva napodobňovanie (zraniteľností) ale tiež zatienenie (zraniteľnosti simuluje všeobecne, nie konkrétne) a obalenie (webovú aplikáciu predstavuje ako zraniteľnú, pričom opak je pravdou).

Nasledujúcim honeypotom v zbierke je **Glutton** [45], poskytujúci SSH a TCP pripojenie k serveru. SSH proxy v tomto prípade funguje ako prostredník medzi útočníkom a serverom a navyše slúži na zaznamenávanie celej komunikácie, TCP proxy ktoré predstavuje Glutton zatiaľ logovanie komunikácie neposkytuje. Tento honeypot sa vyznačuje návnadou, pretože láka útočníka na prelomenie prihlasovacích údajov a maskovaním, pretože sa tvári ako reálna služba.

Ďalším jednoduchým honeypotom je **Heralding** [46], nízko interaktívny honeypot, ktorý používateľovi umožňuje simulovať množstvo protokolov (v súčasnej dobre je ich 14) využívajúcich prihlásenie do systému, či služby. Heralding následne zaznamenáva všetky prihlasovacie údaje zadané útočníkom pri prihlasovaní sa do jednotlivých služieb. Z rovnakých dôvodov ako Glutton, zaraďujeme Heralding k návnade a maskovaniu.

O niečo komplexnejší je honeypot **HoneyPy** [47], ktorý takisto simuluje systém zahŕňajúci viacero služieb a zaznamenáva útoky na jednotlivé z nich. HoneyPy je nízko až stredne interaktívny honeypot emulujúci služby založené na internetových protokoloch TCP a UDP. Podobne ako honeypoty simulujúce pripojenie, aj HoneyPy využíva návnadu vo forme využívania prihlasovacích údajov.

Podobným nástrojom ako HoneyPy je **HoneyTrap** [48], bezpečnostný nástroj slúžiaci na sledovanie útokov na služby využívajúce protokoly TCP a UDP. Server emuluje známu službu jednoduchým odoslaním zachytenej sieťovej prevádzky pripojenému hostiteľovi, teda potenciálnemu útočníkovi. Rovnako ako HoneyPy, HoneyTrap predstavuje návnadu pre útočníka, umožnením prihlásenia sa do jednotlivých služieb.

Ďalším honeypotom, ktorý obsahuje T-Pot je **Mailoney** [49], SMTP honeypot napísaný v Pythone. Zaznamenáva pokusy o prihlásenie a taktiež e-maily, ktoré sa

útočník pokúša odoslať. Mailoney využíva maskovanie a z rovnakých dôvodov ako iné honeypoty sledujúce prihlásenie, využíva návnadu.

Funkcionálne najviac odlišným z týchto honeypotov je **Medpot** [50], takzvaný HL7/FHIR honeypot. HL7 alebo tiež Health Level Seven je spoločnosť a zároveň súbor medzinárodných štandardov na prenos a zdieľanie klinických a administratívnych dát medzi aplikáciami, ktoré používajú rôzni poskytovatelia zdravotnej starostlivosti [51]. FHIR je štandard pre výmenu údajov o zdravotnej starostlivosti publikovaný spoločnosťou HL7 [52]. Medpot využíva návnadu, keďže predstavuje systém pracujúci s citlivými dátami a tiež napodobňovanie (prenosu údajov).

Nasledujúcim honeypotom je **RDPY** [53], implementácia protokolu RDP (Remote Desktop Protocol), čo je sieťový protokol umožňujúci používateľovi vzdialene ovládať počítač prostredníctvom počítačovej siete [54]. RDPY ako podvod používa napodobňovanie, keďže simuluje reálny protokol a taktiež vynájdenie, keďže reálne pripojenie k vzdialenému počítaču nie je zrealizované.

Poslednou dvojicou z kolekcie T-Pot sú honeypoty **Snare** [55] a **Tanner** [56]. Snare je honeypot pre webové aplikácie, ktorý funguje ako senzor a je lákadlom pre škodlivý softvér. Tanner funguje ako jeho druhá polovica, je teda službou, ktorá analyzuje a klasifikuje údaje z udalostí zaznamenaných honeypotom Snare a rozhoduje o tom, ako má Snare na dané udalosti reagovať. Snare aj Tanner oba využívajú maskovanie a návnadu, keďže Snare funguje ako lákadlo a Tanner ako logovacia služba.

Tab. 3 Prehľad spôsobov podvodu, ktoré využívajú jednotlivé honeypoty

Honeypot	**Masko-vanie**	**Obale-nie**	**Zatiene-nie**	**Napodob-ňovanie**	**Vynájde-nie**	**Návna-da**
ADBHoney	X			X		
Cisco ASA						X
Conpot				X	X	X
Cowrie				X		X
Dionaea			X		X	
ElasticPot	X					
Glastopf		X	X	X		

Glutton	X					X
Heralding	X					X
HoneyPy						X
HoneyTrap						X
Mailoney	X					X
Medpot				X		X
RDPY				X	X	
Snare	X					X
Tanner	X					X

Záver

V článku sme popísali definície honeypotov, honeynetov a ich delenia. Ďalej sme objasnili pojem podvodu a podvodných systémov, pričom sme popísali, ako sa tieto systémy dajú prirovnať k honeypotom. Toto porovnanie má veľký význam pre našu prácu, pretože ho plánujeme využiť pri implementácii nástroja na detekciu honeypotov.

V ďalších kapitolách a podkapitolách sme popísali podobné práce, ktoré sa venujú detekcii honeypotov. Popísali sme rôzne pohľady autorov na detekciu honeypotov a taktiež sme zhodnotili, že podľa týchto prác neexistuje jednotná metóda na detekciu honeypotov. Cieľom našej práce je z tohto dôvodu nájsť model na detekciu honeypotov a pozrieť sa na detekciu honeypotov z iného, všeobecného pohľadu.

Ďalej sme v práci popísali jednotlivé honeypoty, ktoré budú testovacou sadou pre náš nástroj. Oboznámili sme sa s ich vlastnosťami, funkcionalitou a každému z nich sme priradili typ podvodu, ktorý využíva. O väčšine honeypotov z nástroja, ktorý sme si vybrali na testovanie sa dá povedať, že využívajú viac ako jednu techniku podvodu. Popísali sme preto ktoré metódy jednotlivé honeypoty ako podvodné systémy používajú a taktiež ako.

Našim ďalším krokom v práci bude zostavenie všeobecných modelov pre jednotlivé spôsoby podvodov využívaných u honeypotov. Následne ostáva implementácia nástroja a jeho vyhodnotenie. Výsledkom našej práce bude program v jazyku Python, ktorý na vstupe dostane IP adresu stroja a jeho výstupom bude vyjadrenie, či daný stroj je honeypot, alebo reálny operačný systém, spôsob detekcie a navrhnuté opatrenie proti detekcii**.**

Zoznam použitej literatúry

1. Kippo - SSH Honeypot, 2016 [online] Dostupné z: https://github.com/desaster/kippo

2. Kippo is being detected by Metasploit, 2012 [online] Dostupné z: https://bruteforcelab.com/kippo-is-being-detected-by-metasploit.html

3. Wireshark, 2019 [online] Dostupné z: https://www.wireshark.org/

4. OpenSSH, 2019 [online] Dostupné z: https://www.openssh.com/

5. Protocol Basics: Secure Shell Protocol, 2019 [online] Dostupné z: https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-46/124-ssh.html

6. JOSHI, R. C.; SARDANA, Anjali, 2011. Honeypots: a new paradigm to information security. CRC Press.

7. AKKAYA, D.; THALHOTT, F. 2010. Honeypots in network security.

8. SPITZNER, L., 2002. Honeypots: Tracking Hackers.

9. Dionaea, 2011 [online] Dostupné z: http://honeynet.org/project/dionaea

10. Honeyd, 2008 [online] Dostupné z: http://www.honeyd.org/

11. The Honeynet Project: Know Your Enemy: Sebek2 A kernel based data capture tool, 2003.

12. Argos, 2014 [online] Dostupné z: https://www.few.vu.nl/argos/

13. Thug, 2019 [online] Dostupné z: https://buffer.github.io/thug/

14. ROWE, N. C.; RRUSHI, J., 2016. Introduction to Cyberdeception

15. QASSRAWI, M. T.; HONGLI, Z., 2010. Deception Methodology in Virtual Honeypots

16. JIANG, X.; WANG, X., 2007. Monitoring of VM-based High-Interaction Honeypots

17. Sebek, 2006 [online] Dostupné z: http://old.honeynet.org/tools/sebek/

18. MUKKAMALA, S., 2007. Detection of Virtual Environments and Low Interaction Honeypots

19. Honeyd, 2007 [online] Dostupné z: http://www.honeyd.org/

20. KRAWETZ, N., 2004. Anti-Honeypot Technology

21. FU, X., 2006. On Recognizing Virtual Honeypots and Countermeasures

22. DAHNUL, R. N.; LIM, C.; PURNAMA, J., 2017. Enhancing Honeypot Deception Capability Through Network Service Fingerprinting

23. INNES, S.; VALLI, C., 2006. Honeypots: How do you know when you are inside one?

24. chroot, 2019 [online] Dostupné z. https://en.wikipedia.org/wiki/Chroot

25. ZUHRI, F. A., 2019. The Illusion Of The Cyber Intelligence Era

26. CARROLL, T. E.; GROSU, D., 2009. A Game Theoretic Investigation of Deception in Network Security

27. Release T-Pot 19.03, 2019 [online] Dostupné z: https://dtag-dev-sec.github.io/mediator/feature/2019/04/01/tpot-1903.html

28. DTAG Community Honeypot Project, 2019, [online] Dostupné z: https://dtag-dev-sec.github.io/

29. Docker, 2019, [online] Dostupné z: https://www.docker.com/

30. Docker: What is a Container?, 2019, [online] Dostupné z: https://www.docker.com/resources/what-container

31. ELK Stack: Elasticsearch, Logstash, Kibana, 2019 [online] Dostupné z: https://www.elastic.co/what-is/elk-stack

32. Elasticsearch, 2019 [online] Dostupné z: https://www.elastic.co/what-is/elasticsearch

33. Logstash: Collect, Parse, Transform Logs, 2019 [online] Dostupné z: https://www.elastic.co/products/logstash

34. Kibana: Explore, Visualize, Discover Data, 2019 [online] Dostupné z: https://www.elastic.co/products/kibana

35. Honeypot ADB Honey, 2019 [online] Dostupné z: https://github.com/huuck/ADBHoney

36. Android Debug Bridge, 2019 [online] Dostupné z: https://developer.android.com/studio/command-line/adb

37. Cisco ASA Honeypot, 2019 [online] Dostupné z: https://github.com/Cymmetria/ciscoasa_honeypot

38. Cisco Adaptive Security Appliance (ASA) Software, 2019 [online] Dostupné z: https://www.cisco.com/c/en/us/products/security/adaptive-security-appliance-asa-software/index.html

39. Zraniteľnosť CVE-2018-0101, 2019 [online] Dostupné z: https://nvd.nist.gov/vuln/detail/CVE-2018-0101

40. Honeypot Conpot, 2019 [online] Dostupné z: http://conpot.org/

41. Honeypot Cowrie, 2019 [online] Dostupné z: https://github.com/cowrie/cowrie

42. Honeypot Dionaea, 2019 [online] Dostupné z: https://github.com/DinoTools/dionaea

43. Honeypot ElasticPot, 2019 [online] Dostupné z: https://github.com/schmalle/ElasticpotPY

44. Honeypot Glastopf, 2019 [online] Dostupné z: https://github.com/mushorg/glastopf

45. Honeypot Glutton, 2019 [online] Dostupné z: https://github.com/mushorg/glutton

46. Honeypot Heralding, 2019 [online] Dostupné z: https://github.com/johnnykv/heralding

47. Honeypot HoneyPy, 2019 [online] Dostupné z: https://github.com/foospidy/HoneyPy

48. Honeypot HoneyTrap, 2019 [online] Dostupné z: https://github.com/armedpot/honeytrap/

49. Honeypot Mailoney, 2019 [online] Dostupné z: https://github.com/awhitehatter/mailoney

50. Honeypot Medpot, 2019 [online] Dostupné z: https://github.com/schmalle/medpot

51. Health Level Seven International, 2019 [online] Dostupné z: https://www.hl7.org/

52. Štandard FHIR, 2019 [online] Dostupné z: https://www.hl7.org/fhir/

53. Honeypot RDPY, 2019 [online] Dostupné z: https://github.com/citronneur/rdpy

54. Understanding the Remote Desktop Protocol, 2019 [online] Dostupné z: (RDP) https://support.microsoft.com/en-us/help/186607/understanding-the-remote-desktop-protocol-rdp

55. Honeypot Snare, 2019 [online] Dostupné z: https://github.com/mushorg/snare

56. Honeypot Tanner, 2019 [online] Dostupné z: https://github.com/mushorg/tanner

57. Checkpot Honeypot Checker, 2019 [online] Dostupné z: https://github.com/vladalexgit/checkpot

58. Checkpot: Tests and TestPlatform framework, 2019 [online] Dostupné z: https://checkpot.readthedocs.io/en/master/test_platform_description.html

59. Honeybee, 2019 [online] Dostupné z: https://github.com/mohitrajain/honeybee

60. Honeypot Amun, 2019 [online] Dostupné z: https://github.com/zeroq/amun

61. Honeyscor, 2019 [online] Dostupné z: https://honeyscore.shodan.io/

62. Shodan, 2019 [online] Dostupné z: https://www.shodan.io/