Detekcia honeypotov

Analýza a návrh riešenia

Bc. Lucia Kokuľová

1Im, 2018 - 2019

Abstrakt. 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 program, ktorý na vstupe dostane IP adresu a jeho výstupom bude 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

1 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).

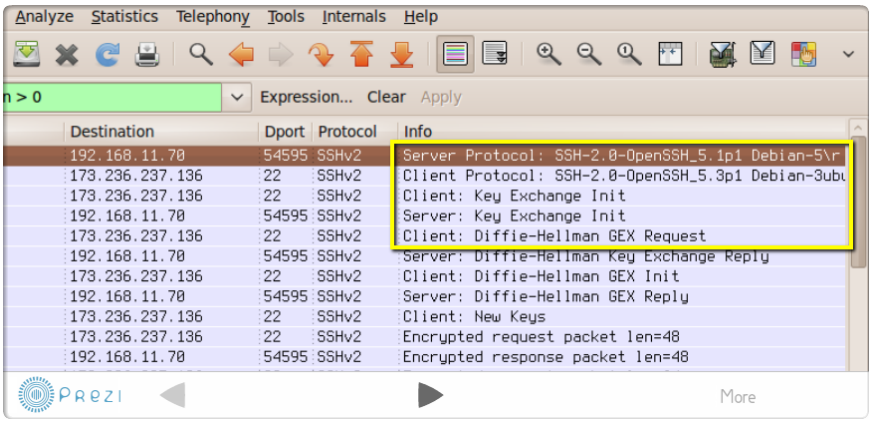
Napriek tomu, že útočník má byť schopný neautorizovane použiť honeypot, nemôže byť schopný odhaliť to, že s ním komunikuje. V tomto prípade by hrozilo, že ukončí, resp. pozmení svoju aktivitu, čím sa cieľ honeypotu – identifikácia správania útočníka, nepodarí zrealizovať. Napriek skutočnosti, že nedetekovateľnosť 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.

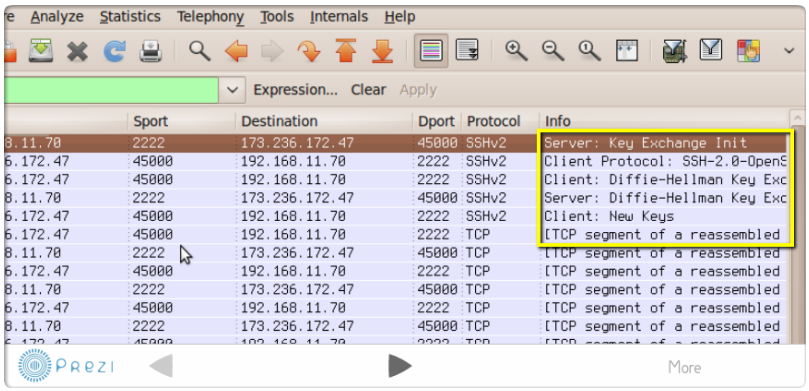
Jednou z možností, ako vyriešiť tento problém, je pokúsiť sa obmedziť možnosti detekcie honeypotov. Cieľom našej práce je preto implementácia nástroja na detekciu honeypotov, ktorého úlohou bude odhaliť honeypot v rámci počítačovej siete, na základe čoho bude možné odstrániť vlastnosti, ktoré dopomohli k jeho odhaleniu.

1.1 Ilustračný príklad

Ako príklad si uvedieme detekciu honeypotu Kippo [1][2], ktorého úlohou je simulovať službu SSH (Secure shell). Kippo sa používa na zaznamenávanie bezpečnostných útokov na SSH službu pomocou testovania prihlasovacích údajov. Pri analýze tohto honeypotu programom Wireshark [3] si môžeme všimnúť, že Kippo nedodržuje korektnú výmenu kľúča, tak ako reálny SSH server. Nižšie môžeme vidieť dva obrázky, jeden zobrazuje komunikáciu honeypotu Kippo, ktorý sa tvári ako SSH server, druhý obrázok zobrazuje komunikáciu reálneho OpenSSH [4] servera. Honeypot Kippo je detekovateľný aj pomocou iných metód, niektoré z nich budeme rozoberať v nasledujúcich kapitolách článku.

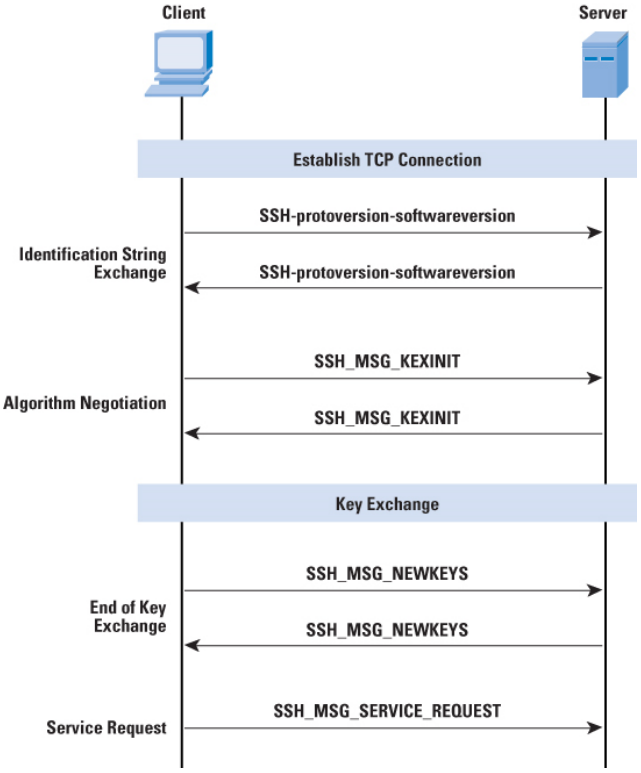


Obr. 1. Komunikácia reálneho OpenSSH servera [2]



Obr. 2. Komunikácia honeypotu Kippo s útočníkom [2]

Na Obr. 2, ktorý znázorňuje komunikáciu honeypotu Kippo s útočníkom je vidieť, že honeypot (server) predčasne poslal paket potrebný na výmenu kľúčov, čím porušil správnu sekvenciu nadviazanie komunikácie. Táto chyba je viditeľná napríklad programom Wireshark. Týmto spôsobom je honeypot Kippo rozoznateľný od reálneho systému (respektíve od reálnej služby SSH).



Obr. 3. Priebeh komunikácie v protokole SSH [5]

Ako môžeme vidieť na Obr. 3, pri normálnom pokuse o nadviazanie komunikácie klient aj server posielajú číslo svojej verzie. Druhá strana odpovie rovnako svojim číslom a na rad príde žiadosť o výmenu kľúčov. Na túto výzvu server odpovie a následne sa kľúče vymenia pomocou Diffie-Hellmanovho protokolu a nadviaže sa šifrované spojenie.

2 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.

2.1 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.

2.1.1 Delenie honeypotov

Honeypoty 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 zvyčajne platí, že kategórie honeypotov sa navzájom prekrývajú a dopĺňajú. V nasledujúcich podkapitolách si predstavíme niekoľko kritérií a delení honeypotov, ktoré sú pre našu prácu dôležité.

2.1.2 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.

2.1.3 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 málo 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). Napríklad 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 napríklad honeypot Kippo [1], ktorý sme spomínali v úvode tejto práce ako ilustračný príklad.

V Tab. 1. [6] 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

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

2.1.4 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).

2.1.5 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.

2.2 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) a
- analýza údajov (data analysis).

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.

3 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 vyberie metódu, akou bude podvod vykonávať. Podvodné metódy môžeme rozdeliť do šiestich základných kategórií [14]:

- **maskovanie (masking)** – systém skrýva nejaký process, 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 to 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 napríkladzámerné ponechanie hesiel v súborovom systéme, čo nabáda útočníka aby ich použil.

Podľa toho, aký typ podvodu využíva daný honeypot, vieme dopredu určiť, aké typy maskovania bude využívať. Po zaradení honeypotu do podvodných metód teda vieme určiť, akým spôsobom by sa v systéme dal vypátrať, resp. odhaliť.

4 Prehľad existujúcich riešení

V existujúcich prácach o detekcii honeypotov sa nachádzajú rôzne nezhody v otázke delenia honeypotov. Každý autor rozdeľoval honeypoty podľa rôznych kritérií a od toho sa zvyčajne odvíjala 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 a zmysle, vo všeobecnej rovine a nie podľa konkrétnych kritérií.

4.1 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 VM 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.

4.2 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	✓	×

4.3 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.

4.4 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).

4.5 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, 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.

5 Návrh riešenia

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, sa pozrieme 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.

6 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.

Našim ďalším krokom v práci bude výber a inštalácia vhodných honeypotov pre účely prideľovania jednotlivých podvodných metód ku konkrétnym honeypotom. 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.

Poďakovanie

Týmto sa chcem poďakovať vedúcemu mojej diplomovej práce JUDr. RNDr. Pavlovi Sokolovi, PhD. za cenné rady a pomoc s prípravou a písaním tohto článku.

Literatúra

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://wiki.debian.org/chroot