EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103004/I/2023/421000161848

Aplikácia pre analýzu IP adries pristupujúcich k službe SSH v operačnom systéme Linux a zistenie krajiny ich pôvodu

Diplomová práca

2023 **Bc. Tomáš Vrábel**

EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

Aplikácia pre analýzu IP adries pristupujúcich k službe SSH v operačnom systéme Linux a zistenie krajiny ich pôvodu

Diplomová práca

Študijný program:	Informačný manažment
Študijný odbor:	Ekonómia a manažment
Školiace pracovisko:	Katedra aplikovanej informatiky
Vedúci záverečnej práce:	Ing. Pavol Sojka

Bratislava 2023 **Bc. Tomáš Vrábel**

Ekonomická univerzita v Bratislave
Fakulta hospodárskej informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Bc. Tomáš Vrábel
Študijný program:	informačný manažment (Jednoodborové štúdium, inžiniersky II. st., denná forma)
Študijný odbor:	ekonómia a manažment
Typ záverečnej práce:	Inžinierska záverečná práca
Jazyk záverečnej práce:	slovenský
Sekundárny jazyk:	anglický

Názov: Aplikácia pre analýzu IP adries pristupujúcich k službe SSH v operačnom systéme Linux a zistenie krajiny ich pôvodu

Anotácia: Študent naprogramuje aplikáciu, ktorá analyzuje vybrané záznamy OS Linux (tzv. log záznamy), za určitý časový úsek vyberie z tohto záznamu IP adresy, zistí krajinu pôvodu IP adresy a taktiež počet tej ktorej IP adresy vo vybranej vzorke. Nastavením prahovej hodnoty (tzv. threshold) v aplikácii, nám táto vypíše zoznam IP adries spĺňajúcich túto prahovú hodnotu, aby používateľ vedel identifikovať, ktoré IP adresy, ako často a v akom počte kontaktujú zvolený server.

Vedúci:	Ing. Pavol Sojka, PhD.
Katedra:	KAI FHI - Katedra aplikovanej informatiky
Vedúci katedry:	Ing. Mgr. Peter Schmidt, PhD.
Dátum zadania:	18.10.2021

Dátum schválenia: 17.04.2023

prof. Ing. Ivan Brezina, CSc.
osoba zodpovedná za realizáciu študijného programu

Čestné vyhlásenie

Čestne vyhlasujem, že diplomovú prácu s názvom: Aplikácia pre analýzu IP adries pristupujúcich k službe SSH v operačnom systéme Linux a zistenie krajiny ich pôvodu som vypracoval samostatne, na základe konzultácií a štúdia odbornej literatúry. Neporušil som žiadne autorské práva a zoznam použitej literatúry som správne uviedol.

V Bratislave, dňa: 28.4.2023 .....

Bc. Tomáš Vrábel

Poďakovanie

Chcel by som sa touto cestou veľmi pekne poďakovať Ing. Pavlovi Sojkovi za poskytnutú odbornú pomoc, cenné rady a nápady pri tvorení tejto práce. Pán Ing. Sojka bol vždy ochotný so mnou konzultovať, keď som narazil na nejaký problém.

ABSTRAKT

VRÁBEL, Tomáš: *Aplikácia pre analýzu IP adries pristupujúcich k službe SSH v operačnom systéme Linux a zistenie krajiny ich pôvodu* – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra aplikovanej informatiky. – Ing. Pavol Sojka. – Bratislava: FHI, 2023, 50s.

Cieľom diplomovej práce je získanie základných teoretických vedomostí o správe systémov SSH a tiež informácií o IP adresách pripájajúcich sa k operačnému systému LINUX. Hlavným cieľom je naprogramovanie aplikácie v jazyku Python, ktorá zhromaždí zo záznamov operačného systému Linux informácie o IP adresách pristupujúcich k operačnému systému a zistí krajinu ich pôvodu. Práca je rozdelená do 3 kapitol.

Prvá kapitola je venovaná analýze problematiky súčasného stavu a definícií základných pojmov.

Druhá kapitola obsahuje návrh aplikácie.

Posledná kapitola obsahuje implementáciu a testovanie aplikácie.

Výsledkom práce je teda aplikácia, ktorá dokáže čítať informácie o IP adresách pristupujúcich k OS Linux.

Kľúčové slová: SSH, Python, IP adresa, Linux

ABSTRACT

VRÁBEL, Tomáš: *Application for analysis of IP addresses accessing the SSH service in the Linux operating system and finding out their country of origin*. – University of Economics in Bratislava. Faculty of Economic Informatics; Department of Applied Informatics. – Ing. Pavol Sojka. – Bratislava: FHI, 2023, 50p

The aim of the thesis is to gain basic theoretical knowledge about SSH systems management and to obtain information about IP addresses connecting to the LINUX operating system. The main goal is to program a Python application that collects from the Linux operating system logs information about IP addresses accessing the operating system and finds out their country of origin. The thesis is divided into 3 chapters.

The first chapter is devoted to an analysis of the current state of the subject and definitions of the basic terms.

The second chapter contains the design of the application.

The last chapter contains the implementation and testing of the application.

The result of the thesis is an application that can read information about IP addresses accessing the Linux OS.

Keywords: SSH, Python, IP address, Linux

Obsah

Zoznam použitých skratiek:

1. IETF – Internet Engineering Task Force
2. SSH – Secure Shell
3. PAM - Privileged access management
4. C&C – Command and Control

Zoznam obrázkov:

Úvod:

V súčasnej dobe je bezpečnosť IT systémov jednou z najdôležitejších oblastí, na ktorú sa organizácie musia zamerať. Hrozby internetových útokov sa neustále zvyšujú a útočníci sa snažia nájsť nové a sofistikovanejšie spôsoby, ako sa dostať do cieľových systémov a získať prístup k citlivým dátam a informáciám. Jednou z najbežnejších metód, ktoré používajú útočníci na získanie neoprávneného prístupu k cieľovým systémom, je útok cez protokol SSH. Napriek tomu, že SSH je považovaný za bezpečný protokol, útočníci využívajú rôzne techniky, aby získali prístup k cieľovým systémom. Jednou z týchto techník je útok Brute Force, ktorý spočíva v opakovanom pokuse o prihlásenie sa na SSH službu pomocou rôznych kombinácií užívateľských mien a hesiel. Tento typ útoku môže spôsobiť vážne bezpečnostné problémy, ak nie sú prijaté dostatočné opatrenia na ochranu systému. Preto je dôležité mať nástroje na monitorovanie a analýzu prístupov cez protokol SSH. V rámci analýzy IP adries, ktoré pristupujú k SSH službe, sa dá zistiť, ktoré krajiny majú najväčší počet prístupov. Na základe týchto informácií môžu byť organizácie schopné identifikovať potenciálne hrozby z rôznych krajín a prijať opatrenia na ochranu svojich systémov. To môže zahŕňať blokovanie prístupov z krajín s vysokým počtom prístupov alebo zavedenie ďalších bezpečnostných opatrení, ako je napríklad silnejšie autentifikovanie užívateľov.

Cieľom tejto diplomovej práce je vytvoriť aplikáciu, ktorá bude schopná monitorovať prístupy cez protokol SSH a analyzovať IP adresy, z ktorých sa prístup uskutočňuje. Aplikácia bude tiež schopná zistiť krajinu pôvodu IP adries a zobraziť tieto informácie užívateľovi v prehľadnej forme. Tieto informácie môžu byť použité na identifikáciu hrozieb a ochranu systému pred útokmi. V rámci práce budeme skúmať existujúce nástroje a technológie, ktoré sa používajú na monitorovanie prístupov cez protokol SSH a analýzu IP adries. Budeme tiež skúmať metódy na získavanie informácií o krajinách pôvodu IP adries. Na základe týchto poznatkov bude vytvorená aplikácia pre operačný systém Linux, ktorá bude schopná uskutočniť monitorovanie a analýzu prístupov cez protokol SSH a zobraziť informácie o krajinách pôvodu IP adries. Táto diplomová práca môže byť užitočná pre organizácie, ktoré chcú zlepšiť bezpečnosť svojich systémov a zabezpečiť, že prístupy cez protokol SSH sú monitorované a chránené pred útokmi. Práca

tiež môže slúžiť ako zdroj informácií pre vývojárov a bezpečnostných expertov, ktorí pracujú na ochrane IT systémov a snažia sa identifikovať nové metódy útokov a zabezpečiť ich ochranu.

1 Súčasný stav problematiky doma a v zahraničí

1.1 Úvod do operačného systému Linux

Linux je operačný systém, ktorý bol vytvorený v roku 1991 Linusom Torvaldsom. Je to voľne šíriteľný, open-source operačný systém, ktorý je založený na Unixovom jadre. Linux ponúka množstvo rôznych distribúcií, ktoré sa od seba líšia rôznymi balíkmi softvéru a konfiguračnými nástrojmi. Linux je známy pre svoju stabilitu, bezpečnosť a flexibilitu. Je to obľúbený operačný systém pre serverové aplikácie, webové stránky, databázové systémy, ale tiež pre osobné počítače a mobilné zariadenia.

Používatelia Linuxu majú prístup k množstvu open-source softvéru, ktorý im umožňuje prispôsobiť si svoj systém svojim potrebám. Okrem toho, Linux podporuje množstvo rôznych súborových systémov, ako aj rôzne typy sieťových protokolov, čo umožňuje používateľom pracovať s rôznymi zariadeniami a aplikáciami bez obmedzení. Linux sa tiež vyznačuje vysokou úrovňou bezpečnosti a je schopný odolávať mnohým druhom útokov a hrozieb.

Celkovo povedané, Linux je dôležitý operačný systém, ktorý ponúka mnoho výhod pre používateľov, ktorí hľadajú stabilný, bezpečný a prispôsobiteľný operačný systém. Jeho popularita stále rastie a mnoho programátorov a vývojárov venuje svoj čas a energiu na vylepšovanie a rozširovanie tohto systému.

Každá distribúcia Linuxu poskytuje užívateľské rozhranie Shell, čo je vlastne interpret príkazového riadku. Týmto rozhraním je možné ovládať celý operačný systém pomocou malých pomocných programov, ktoré sú nezávislé od danej distribúcie. Na spustenie programov stačí zadať ich názov do príkazového riadku a stlačiť klávesu Enter. Vlastnosti každého programu je možné ovplyvniť pridaním prepínačov a argumentov, napríklad takto: program –prepínač argument.

1.2 Secure Shell (SSH)

Secure Shell alebo v skratke SSH označuje protokol ale aj názov služby a klienta komunikujúceho so serverom pomocou tohto protokolu. Jedná sa o službu, na ktorú je v unixových systémoch najväčší počet útokov hrubou silou. Tato služba je využívaná na takmer všetkých serveroch. Takisto ju používa väčšina domácich routerov a iných sieťových prvkov.

Protokol SSH je bezpečnou náhradou pre staršie protokoly, ktoré sa používali na prístup k vzdialenému shellu, ako napríklad rsh alebo telnet. SSH využíva architektúru typu klient/server, pričom server sa zvyčajne nazýva sshd a klient jednoducho ssh.

Primárnym účelom SSH je umožniť používateľom spúšťať príkazy na vzdialenom systéme. Okrem toho umožňuje aj zabezpečené kopírovanie súborov cez sieť, tunelovanie TCP portov a dokonca aj tunelovanie protokolu X11.

Protokol SSH zabezpečuje autentifikáciu spojenia, čo zahŕňa overenie identity prihláseného používateľa, čo môže byť vykonané pomocou hesla alebo RSA/DSA kľúča. Taktiež poskytuje šifrovanie spojenia, čo znamená, že všetky prenášané dáta sú šifrované a nie je možné ich dešifrovať pri odposluchu komunikácie na sieti. Súčasné implementácie OpenSSH používajú šifry 3DES, Blowfish, AES alebo ARC4.

Protokol SSH tiež zabezpečuje integritu spojenia, čo znamená, že zaručuje, že prenášané dáta dorazia k cieľu nezmenené a ak boli modifikované útočníkom, táto skutočnosť bude zistená.

História SSH

SSH1 a protokol SSH-1 boli vyvinuté v roku 1995 Tatu Ylönenom na Helsinskej technickej univerzite vo Fínsku. Po tom, čo sa jeho univerzita začiatkom toho istého roka stala obeťou útoku na odcudzenie hesla, vytvoril SSH1 pre seba. Keď si však beta verzie začali získavať pozornosť, uvedomil si, že jeho bezpečnostný produkt by sa dal využiť širšie. V júli 1995 bolo SSH1 sprístupnené verejnosti ako voľný softvér so zdrojovým kódom, ktorý umožňoval ľuďom kopírovať a používať program bez nákladov. Do konca roka si SSH1 osvojilo približne 20 000 používateľov v 50 krajinách a Ylönen denne odpovedal na 150 e-mailových správ so žiadosťou o podporu. V reakcii na to Ylönen

založil v decembri 1995 spoločnosť SSH Communications Security, Ltd. (SCS), aby udržiavala, komercializovala a pokračovala vo vývoji SSH.

V roku 1995 Ylönen zdokumentoval protokol SSH-1 ako IETF[1], ktorý v podstate opisoval fungovanie softvéru SSH1. Bol to do istej miery ad hoc protokol s množstvom problémov a obmedzení, ktoré sa objavili s rastúcou popularitou softvéru. Tieto problémy sa nedali odstrániť bez straty spätnej kompatibility, preto v roku 1996 spoločnosť SCS predstavila novú, hlavnú verziu protokolu, SSH 2.0 alebo SSH-2, ktorá obsahuje nové algoritmy a je nekompatibilná s SSH-1. V reakcii na to IETF vytvorila pracovnú skupinu s názvom SECSH (Secure Shell) s cieľom štandardizovať protokol a riadiť jeho vývoj vo verejnom záujme.

SSH-2 však nebol medzi používateľmi príliš obľúbený, kvôli obmedzujúcej komerčnej licencii a odstráneniu niektorých užitočných a praktických funkcií. Pri prechode na SSH-2 videli používatelia len málo výhod. Avšak koncom roku 2000 SCS uvoľnila licenciu SSH-2 na voľné použitie pre operačné systémy Linux, FreeBSD, NetBSD a OpenBSD[2]. Od roku 1999 sa zároveň vyvíja projekt OpenSSH, vyvíjaný pod záštitou OpenBSD, ktorý je tiež dostupný pod OpenBSD licenciou. Táto implementácia vychádza z poslednej verzie SSH-1, ktorá bola vydaná ako open source, 1.2.12. V súčasnosti OpenSSH implementuje ako SSH-1, tak aj SSH-2 protokol a od roku 2005 je najrozšírenejšou implementáciou SSH, ktorú môžeme nájsť ako predvolenú vo veľkom množstve operačných systémov. [1]

SSHD Konfigurácia

SSH daemon obsahuje veľa konfiguračných nastavení ale nižšie sú popísané tie, ktoré súvisia s obranou proti útokom hrubou silou. SSHD číta konfiguračné dáta z */etc/ssh/sshd_config* alebo zo súboru špecifikovaného s –f na príkazovom riadku. Ak nie je uvedené inak, pre každé kľúčové slovo sa použije prvá získaná hodnota. [2]

MaxAuthTries: Jedným z najjednoduchších spôsobov, ako zamedziť útokom hrubou silou, je znížiť počet pokusov o overenie povolených v rámci pripojenia. Môžeme to urobiť znížením hodnoty premennej MaxAuthTries v súbore sshd_config z predvolenej hodnoty 6. [3]

PermitRootLogin: Ďalším častým útokom je pokus o prihlásenie sa ako užívateľ root. Nastavením tohto argumentu na „no“ vieme zabrániť veľkému počtu útokov.

PasswordAuthentication: Určuje, či je povolené overovanie heslom. Predvolené nastavenie je „no“. Pri zmene nastavenia treba tiež skontrolovať **UsePAM**. Pri nastavení **PasswordAuthentication** na „no“ sa musí užívateľ prihlásiť pomocou loginu a verejného kľúča.

PermitEmptyPasswords: Ak je povolené overovanie heslom, určuje, či server povolí prihlasovanie do účtov s prázdnymi reťazcami hesiel. Nastavením hodnoty na „no“ môžeme zvýšiť bezpečnosť. [2]

LoginGraceTime: Ak sa používateľ úspešne neprihlási, server sa po tomto čase odpojí. Ak je hodnota 0, neexistuje žiadny časový limit. Predvolená hodnota je 120 sekúnd. [2]

Port: Určuje číslo portu, na ktorom SSHD počúva. Zmenou tohto portu môžeme dočasne obrániť automatizované útoky, avšak má to aj nevýhodu, každí užívateľ musí zadať hodnotu zmeneného portu. Takisto nás zmena portu neochráni pred cieleným útokom nakoľko si útočník vie prečítať port na ktorom služba beží.

Pripojenie a autentizácia na SSH

V tejto časti je vysvetlené čo sa stane, keď sa užívateľ pripojí na server a ako sa autentizuje.

O pripojenie vždy žiada užívateľský počítač. Server počúva na určenom porte (väčšinou 22, pokiaľ z bezpečnostných dôvodov nie je zmenený) a čaká na spojenie. K jednému serveru môže byť naraz pripojených viacero užívateľov.

Keď klient naviaže spojenie so serverom, server odpovie posielaním reťazca, ktorý obsahuje číslo verzie. Klient overí serverovú identitu a pošle späť svoju vlastnú identifikáciu. Toto overenie slúži na overenie toho, či sú porty správne nakonfigurované a či sa softvérové verzie na oboch stranách zhodujú. Identifikačné reťazce sú zrozumiteľné pre človeka. Ak jedna zo strán nerozumie alebo nepodporuje verziu, spojenie sa ukončí.

Po fáze identifikácie sa začne binárna komunikácia medzi oboma stranami. Server pošle svoj verejný kľúč hostiteľa a svoj vlastný verejný kľúč, ktorý má zvyčajne 768 bitov a mení sa každú hodinu. Okrem toho server oznámi, ktoré šifrovacie algoritmy pozná (napríklad 3DES, Blowfish, ArcFour, IDEA alebo CAST-128). Klient vytvorí 256-bitový kľúč pre toto spojenie, šifruje ho oboma kľúčmi RSA a odošle ho ako kľúč pre ďalšiu komunikáciu. Zvolený šifrovací algoritmus je tiež oznámený. Od tohto okamihu oba počítače šifrujú všetky údaje dohodnutým kľúčom a algoritmom. Server potvrdí, že kľúče sú správne zašifrované. Potom sa skontroluje, či je server uvedený v zozname. Ak nie, upozorní sa na to. Ak sa objaví takéto upozornenie, treba to brať vážne, pretože to môže znamenať, že sa niekto pokúša vydávať za server. Ďalšou možnosťou je, že sa na server prihlasujete prvýkrát a nemáte kľúč hostiteľa servera alebo správca servera preinštaloval SSH a vygeneroval nové kľúče. Klient sa následne autentizuje pomocou nejakej metódy overovania. [4]

Pre začiatok autentifikácie klient pošle požiadavku na spojenie pomocou správy SSH_MSG_USERAUTH_REQUEST, ktorá obsahuje používateľské meno, názov služby a názov autentifikačnej metódy a prípadné ďalšie parametre. V prípade, že server zamietne danú autentifikačnú metódu, odpovie správou SSH_MSG_USERAUTH_FAILURE a zoznamom podporovaných autentifikačných metód (v tomto zozname nie je uvedená metóda none, ktorá umožňuje prihlásenie bez autentifikácie, ale aj keď je táto metóda povolená na serveri, požiadavka na ňu je často používaná na zistenie ponúkaných autentifikačných metód). Ak server podporuje zvolenú metódu a tá je úspešná, odpovie správou SSH_MSG_USERAUTH_SUCCESS. [4]

1. **Metóda Public Key:** [5] Všetky implementácie MUSIA podporovať túto metódu; nie všetci používatelia však musia mať verejné kľúče a väčšina lokálnych nariadení pravdepodobne nebude vyžadovať autentifikáciu

verejným kľúčom pre všetkých používateľov v blízkej budúcnosti. Pri tejto metóde slúži vlastníctvo súkromného kľúča ako overenie. Na vykonanie skutočného overenia môže klient následne poslať podpis vygenerovaný pomocou súkromného kľúča. Klient MÔŽE poslať podpis priamo bez predchádzajúceho overenia, či je kľúč prijateľný. Podpis sa posiela pomocou nasledujúceho paketu:

```
byte      SSH_MSG_USERAUTH_REQUEST
string    user name
string    service name
string    "publickey"
boolean   TRUE
string    public key algorithm name
string    public key to be used for authentication
string    signature
```

Hodnota 'signature' je podpis príslušným súkromným kľúčom nad nasledujúcimi údajmi v nasledujúcom poradí:

```
string    session identifier
byte      SSH_MSG_USERAUTH_REQUEST
string    user name
string    service name
string    "publickey"
boolean   TRUE
string    public key algorithm name
string    public key to be used for authentication
```

Keď server prijme túto správu, MUSÍ skontrolovať, či dodaný kľúč je prijateľný na overenie, a ak áno, MUSÍ skontrolovať, či je podpis správny. Ak sú obe kontroly úspešné, táto metóda je úspešná. Treba si uvedomiť, že server môže vyžadovať ďalšie overenia. Server MUSÍ odpovedať správou **SSH_MSG_USERAUTH_SUCCESS** (ak už nie sú potrebné

žiadne ďalšie overenia potrebné ďalšie overenia), alebo **SSH_MSG_USERAUTH_FAILURE** (ak požiadavka zlyhala, alebo je potrebná ďalšia autentifikácia).

2. Metóda Password: [5] Pri overovaní hesla sa používajú tieto pakety. Všimnite si, že server MÔŽE požiadať používateľa o zmenu hesla. Všetky implementácie by mali podporovať overovanie hesla.

```
byte      SSH_MSG_USERAUTH_REQUEST
string    user name
string    service name
string    "password"
boolean   FALSE
string    plaintext password in ISO-10646 UTF-8 encoding
```

Za normálnych okolností server na túto správu odpovie úspešne alebo neúspešne. Ak však platnosť hesla vypršala, server by to mal oznámiť odpoveďou SSH_MSG_USERAUTH_PASSWD_CHANGEREQ. V každom prípade server NESMIE povoliť použitie hesla, ktorého platnosť vypršala na overenie.

```
byte      SSH_MSG_USERAUTH_PASSWD_CHANGEREQ
string    prompt in ISO-10646 UTF-8 encoding [RFC3629]
string    language tag [RFC3066]
```

V tomto prípade klient MÔŽE pokračovať s iným overovaním alebo si od používateľa vyžiadať nové heslo a zopakovať pokus o zadanie hesla.

```
byte      SSH_MSG_USERAUTH_REQUEST
string    user name
```

```
string    service name
string    "password"
boolean   TRUE
string    plaintext old password in ISO-10646 UTF-8
string    plaintext new password in ISO-10646 UTF-8
```

Server musí odpovedať na každú správu s požiadavkou buď
SSH_MSG_USERAUTH_SUCCESS,
SSH_MSG_USERAUTH_FAILURE alebo iným
SSH_MSG_USERAUTH_PASSWD_CHANGEREQ.
Ich význam je nasledovný:

SSH_MSG_USERAUTH_SUCCESS - Heslo bolo zmenené a overenie bolo úspešne dokončené.
SSH_MSG_USERAUTH_FAILURE s čiastočným úspechom - Heslo bolo zmenené ale sú potrebné ďalšie overenia.
SSH_MSG_USERAUTH_FAILURE bez čiastočného úspechu - Heslo nebolo zmenené. Zmena hesla nebola podporovaná alebo bolo staré heslo zlé. Všimnite si, že ak už server odoslal SSH_MSG_USERAUTH_PASSWD_CHANGEREQ, vieme, že podporuje zmenu hesla.
SSH_MSG_USERAUTH_CHANGEREQ - Heslo nebolo zmenené, pretože nové heslo nebolo prijateľné (napr. príliš ľahko uhádnuteľné).

3. Metóda Keyboard Interactive [6] Interaktívna klávesnica je metóda overovania na všeobecné účely, vhodná na interaktívne overovanie, pri ktorom sa overovacie údaje zadávajú prostredníctvom klávesnice alebo rovnocenného alfanumerického vstupného zariadenia. Hlavným cieľom tejto metódy je umožniť klientovi SSH podporovať celú triedu mechanizmov overovania bez toho, aby poznal špecifiká skutočných mechanizmov overovania.

Z praktického hľadiska pri interaktívnom overovaní pomocou klávesnice server SSH posiela klientovi textové výzvy. Na tieto výzvy je potrebné poskytnúť textovú odpoveď. Klient posiela odpovede späť na server. V prípade akceptovania môžu byť klientovi zaslané ďalšie výzvy na odpoveď alebo môže byť autentifikácia vyhlásená za úspešnú alebo neúspešnú. Použitie tejto metódy je opäť iniciované požiadavkou SSH_MSG_USERAUTH_REQUEST v tvare:

```
byte    SSH_MSG_USERAUTH_REQUEST
string  user name (ISO-10646 UTF-8)
string  service name (US-ASCII)
string  "keyboard-interactive"   (US-ASCII)
string  language tag (as defined in [RFC-3066])
string  submethods (ISO-10646 UTF-8)
```

Server na túto správu odpovie odoslaním paketu so správou typu SSH_MSG_USERAUTH_INFO_REQUEST.

```
byte    SSH_MSG_USERAUTH_INFO_REQUEST
string  name (ISO-10646  UTF-8)
string  instruction (ISO-10646 UTF-8)
string  language tag (as defined in [RFC-3066])
int     num-prompts
string  prompt[1] (ISO-10646 UTF-8)
boolean echo[1]
string  prompt[num-prompts] (ISO-10646 UTF-8)
boolean echo[num-prompts]
```

Táto správa žiada klienta o zaslanie ďalších informácií, ktoré má možnosť špecifikovať. Typicky bude žiadať o zadanie hesla ale autentifikačný mechanizmus zvolený zo strany serveru úplne ovláda proces autentifikácie. Po prijatí tejto správy klient zobrazí položky

"name" a "instruction", ak nie sú prázdne a následne vypisuje všetky položky poľa "prompt" a číta odpovede na tieto otázky. Príslušná položka z poľa "echo" určuje, či má byť vstup používateľa zobrazovaný alebo nie (takže napríklad zabráni zobrazeniu hesla, ktoré zadáva používateľ na obrazovke). Po prijatí vstupu používateľa je serveru zaslaná správa obsahujúca odpovede na všetky jeho otázky v určenom formáte.

```
byte    SSH_MSG_USERAUTH_INFO_RESPONSE
int     num-responses
string  response[1] (ISO-10646 UTF-8
string  response[num-responses] (ISO-10646 UTF-8)
```

Na základe poskytnutých informácií server reaguje, buď zaslaním oznámenia o úspešnosti alebo neúspešnosti prihlásenia alebo môže posielať ďalšie správy s dotazmi typu SSH_MSG_USERAUTH_INFO_REQUEST.

1.3 Charakteristika útokov

V dnešnej dobe poznáme nespočetné množstvo typov kybernetických útokov, medzi ktoré patria napríklad odpočúvanie komunikácií na sieti, pretvarovanie sa za niekoho iného, využitie slabín v systémoch, DDoS a DoS útoky alebo šikovné útoky využívajúce sociálne inžinierstvo na preniknutie do systému alebo získanie citlivých dát. Popis všetkých týchto techník by bolo na samostatnú prácu, preto v tejto časti vysvetlíme najmä útoky hrubou silou.

1.3.1 Útok hrubou silou (Brute force)

Brute force útok je typ kybernetického útoku, pri ktorom útočník používa metódu pokus-omyl na uhádnutie hesla alebo šifrovacieho kľúča. To sa robí systematickým overovaním všetkých možných kombinácií znakov, kým sa nenájde správne heslo alebo kľúč. Brute force útoky sú často automatizované pomocou špecializovaných softvérových

nástrojov, ktoré môžu vyskúšať stovky až tisíce kombinácií za sekundu. Brute force útoky môžu byť cielené na rôzne typy systémov a služieb, vrátane:

- Heslom chránené účty: Brute force útoky môžu byť použité na neoprávnený prístup k používateľským účtom na webových stránkach, e-mailových účtoch a iných službách, ktoré vyžadujú prihlasovacie meno a heslo.

- Šifrované dáta: Brute force útoky môžu byť použité na dešifrovanie šifrovaných dát, ako sú súbory alebo správy, vyskúšaním všetkých možných šifrovacích kľúčov, kým sa nenájde ten správny.

- Wi-Fi siete: Brute force útoky môžu byť použité na prelomenie hesiel Wi-Fi pomocou vyskúšania všetkých možných kombinácií znakov, kým sa nenájde správne heslo.

- SSH prihlásenia: Brute force útoky môžu byť použité na neoprávnený prístup k SSH serverom vyskúšaním správneho hesla alebo autentifikačného kľúča.

- Kryptomenové peňaženky: Brute force útoky môžu byť použité na prelomenie súkromných kľúčov kryptomenových peňaženiek, čo môže umožniť útočníkovi kradnutie obsahu peňaženky.

Celkovo sú brute force útoky bežným a potenciálne nebezpečným typom kybernetického útoku, ktorý môže byť použitý na získanie neoprávneného prístupu k systémom a citlivým údajom. Je dôležité používať silné, unikátne heslá a iné bezpečnostné opatrenia na ochranu pred týmito typmi útokov.

Software

Jedným z najznámejších softvérov je Hydra. Je to nástroj na útoky hrubou silou, ktorý pomáha penetračným testerom a etickým hackerom prelomiť heslá sieťových služieb. Hydra môže vykonávať rýchle slovníkové útoky proti viac ako 50 protokolom. Medzi nimi sú telnet, FTP, HTTP, HTTPS, SMB, databázy a niekoľko ďalších služieb. Tento softvér bol vyvinutý hacker-skupinou "The Hacker's Choice" a bol prvýkrát zverejnený v roku 2000 ako nástroj na dôkaz konceptu, ktorý ukázal, ako môžete vykonávať útoky na prihlasovacie služby v sieti. Hydra je tiež paralelným "login crackerom". To znamená, že môžete mať viac ako jedno pripojenie súčasne. Na rozdiel od sekvenčného toto znižuje čas potrebný na zlomenie hesla. [7]

Ďalším takýmto softvérom je Cain a Abel. Podľa oficiálnej webovej stránky je Cain & Abel nástroj na obnovu hesiel pre operačné systémy Microsoft. Umožňuje jednoduché obnovenie rôznych druhov hesiel pomocou sniffovania siete, prelomenia šifrovaných hesiel pomocou slovníkových útokov, útokov Brute-Force a kryptoanalýzy, nahrávania konverzácií VoIP, dekódovania zakódovaných hesiel, obnovenia kľúčov bezdrôtovej siete, odhalenia schránok s heslami, odhalenia hesiel v pamäti a analýzy smerovacích protokolov. Najnovšia verzia je rýchlejšia a obsahuje množstvo nových funkcií, napríklad APR (ARP Poison Routing), ktorá umožňuje sniffovanie v prepínaných sieťach LAN a útoky typu Man-in-the-Middle. Sniffer v tejto verzii dokáže analyzovať aj šifrované protokoly, ako sú SSH-1 a HTTPS, a obsahuje filtre na zachytávanie poverení zo širokej škály overovacích mechanizmov. Nová verzia dodáva aj monitory autentizácie smerovacích protokolov a extraktory trás, slovníkové a brute-force crackery pre všetky bežné hashovacie algoritmy a pre niekoľko špecifických autentizácií, kalkulátory hesiel/hashov, kryptoanalýzy útokov, dekodéry hesiel a niektoré nie až tak bežné nástroje súvisiace so zabezpečením siete a systému.

Ďalším voľne dostupným sieťovým programom je brutessh, tento program je napísaný v jazyku python, dokáže robiť len slovníkové útoky a môže bežať vo vláknach. Na nadviazanie spojenia so serverom sa tu používa knižnica paramiko, na čo potrebuje modul crypto pre Python.

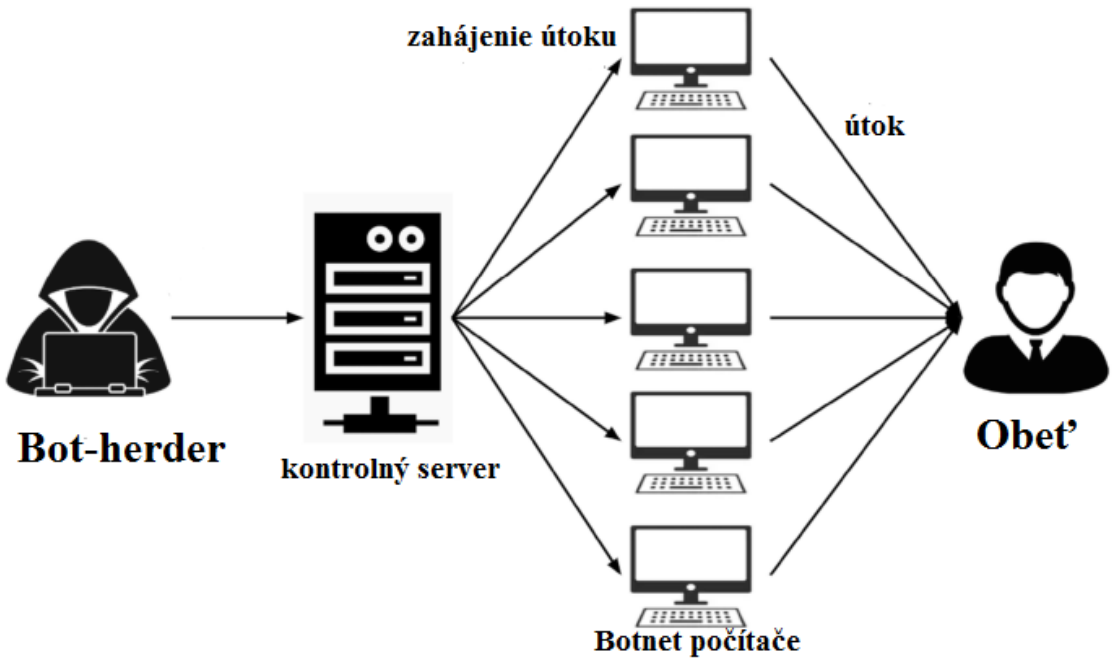
Použitie programu je veľmi jednoduché, napríklad na útok na počítač 161.188.61.190 s používateľským menom Peter a heslami zo súboru heslo.txt, stačí použiť nasledujúci príkaz.

python brutessh.py -h 161.188.61.190 -u Peter -d heslo.txt

Christian Martorella je tvorca bruteforce nástroja brutessh a organizátor konferencií FIST (First Improvised Security Testing), ktoré sa zameriavajú na testovanie bezpečnosti počítačových sietí. Tento nástroj je určený na výskumné a testovacie účely. Brutessh možno stiahnuť z webovej stránky jeho autora http://www.edge-security.com/brutessh.php.

1.3.2 Botnet

Botnet (skratka pre "sieť robotov") je sieť počítačov infikovaných škodlivým softvérom, ktoré sú pod kontrolou jednej útočiacej strany, známej ako "bot-herder". Každý jednotlivý počítač pod kontrolou bot-herdera sa nazýva bot. Z jedného centrálneho bodu môže útočiaca strana prikázať každému počítaču vo svojej botovej sieti, aby súčasne vykonal koordinovanú trestnú činnosť. Rozsah botnetu (mnohé pozostávajú z miliónov botov) umožňujú útočníkovi vykonávať rozsiahle akcie, ktoré predtým neboli možné pomocou škodlivého softvéru. Keďže botnety zostávajú pod kontrolou vzdialeného útočníka, infikované počítače môžu dostávať aktualizácie a meniť svoje správanie za chodu. V dôsledku toho sú bot-herderi často schopní prenajať prístup k segmentom svojho botnetu na čiernom trhu za značný finančný zisk. Na obrázku č. 1 je základná schéma takéhoto botnetu. [9]



Obrázok 1 Schéma Botnet siete

Šírenie botnetov

Botnety vznikajú zvyčajne vtedy, keď útočníci infikujú množstvo počítačov a iných zariadení pomocou malvéru (škodlivého softvéru). Tieto zariadenia sa potom stávajú súčasťou siete, ktorú útočníci môžu použiť na útoky na iné počítače alebo webové stránky.

Malvér môže byť distribuovaný rôznymi spôsobmi, napríklad prostredníctvom spamu, falošných reklám, neaktualizovaných softvérových aplikácií alebo využitím bezpečnostných medzier v zariadeniach. Takisto využitím útoku hrubou silou na získanie vzdialeného prístupu a následné infikovanie počítača.

Využitie botnetov

Botnety sú často využívané zlomyseľníkmi na rôzne účely, ktoré môžu spôsobiť škodu iným používateľom počítačov a internetu. Niektoré z najčastejších využití botnetov sú:

-Distribúcia spamu a phishingu: útočníci môžu použiť botnety na masové odosielanie spamu a phishingových e-mailov, ktoré sú navrhnuté tak, aby získali citlivé informácie od používateľov alebo ich presmerovali na falošné webové stránky.

-Distributed Denial of Service (DDoS) útoky: Botnety môžu byť použité na vykonávanie DDoS útokov, ktoré majú za cieľ znemožniť prístup k webovým stránkam a službám tým, že zaplavia sieť nelegitímnymi požiadavkami.

-Kryptotažba: Zlomyseľníci môžu použiť botnety na ťažbu kryptomien. Týmto spôsobom môžu získať finančné prostriedky na úkor iných používateľov.

-Škodlivý softvér: Botnety môžu byť použité na distribúciu ďalšieho škodlivého softvéru, ktorý môže spôsobiť rôzne škody, ako sú napríklad krádeže identity, šírenie vírusov a ďalšie.

-Podvod s reklamami: Zlomyseľníci môžu použiť botnety na generovanie nelegitímneho internetového obsahu alebo kliknutí na reklamy. Týmto spôsobom môžu získať finančné prostriedky na úkor iných.

Ovládanie botnetov

Ako sme videli na obrázku 1, mnohé existujúce botnety C&C (command and control) sú založené na IRC (Internet Relay Chat), ktorý poskytuje centralizovaný mechanizmus príkazov a kontroly. Botmaster môže komunikovať s botmi (napr. vydávať príkazy a prijímať odpovede) v reálnom čase pomocou správ IRC PRIVMSG. Tento jednoduchý mechanizmus C&C založený na IRC sa osvedčil byť veľmi úspešný a prijali ho mnohé botnety. Existuje aj niekoľko botnetov, ktoré používajú protokol http na C&C. C&C založený na HTTP je stále centralizovaný, ale botmaster priamo nekomunikuje s botmi pomocou mechanizmov podobných chatu. Namiesto toho boti pravidelne komunikujú so serverom(-mi) C&C, aby získali svoje príkazy. Pretože jeho účinnosti a efektívnosti očakávame, že centralizovaný C&C (napr. pomocou IRC alebo HTTP) budú stále botnety v blízkej budúcnosti vo veľkej miere používať. [10]

1.4 Obrana pred útokmi

Aby sme sa mohli proti útokom efektívne brániť je hlavné v prvom rade zistiť, či niekto na nás útočí. Po detegovaní útoku potom môžeme analyzovať o aký útok sa jedná a následne zvoliť správnu stratégiu ako tomuto útoku zabránime.

1.4.1 Detekcia útoku

Ako som spomínal na úspešnú obranu je za potreby správna detekcia útoku. V tejto časti sa budem venovať najmä detekciám útokov hrubou silou na SSH.

Monitorovanie logov

Jedným z najdôležitejších metód detekcie útokov hrubou silou je monitorovanie logov. Monitorovanie logov zahŕňa sledovanie logových súborov SSH servera, ktoré zaznamenávajú každé pripojenie, pokus o prihlásenie a aktivitu používateľa. Monitorovanie logov môže pomôcť pri identifikácii útokov na SSH v reálnom čase a zabezpečiť, aby boli identifikované všetky neobvyklé aktivity. Logovacie súbory môžu byť umiestnené v rôznych umiestneniach, závisí to na konkrétnom operačnom systéme a konfigurácii SSH servera. Napríklad, v distribúciách Linuxu, ako je Ubuntu, logovacie súbory môžu byť umiestnené v /var/log/auth.log alebo /var/log/secure. Na strane servera sa odporúča nastaviť správne konfigurácie pre logovanie SSH aktivít a presunúť logy na bezpečné miesto, kde môžu byť ďalej analyzované a spracované. Pri monitorovaní logov sa odporúča sledovať niektoré kritické informácie, ako napríklad:

- Neúspešné pokusy o prihlásenie - časté pokusy o prihlásenie s nesprávnymi údajmi môžu byť príznakom útoku hrubou silou na SSH.
- Aktivita neznámych používateľov - sledovanie aktivít, ktoré sú vykonávané pomocou účtu, ktorý nie je známy, môže pomôcť pri identifikácii potenciálneho útoku.
- Zmeny v autentifikácii - sledovanie zmien v autentifikácii, ako napríklad zmeny v nastavení kľúčov, môžu byť príznakom neoprávnenej zmeny nastavení.

p0f

Tento softvér nám priamo nepomôže s detegovaním útokov ale pri kombinácií s ostatnými opatreniami dokáže byť užitočným nástrojom pri zisťovaní zraniteľností. P0f je nástroj na detekciu a identifikáciu operačného systému a typu prehliadača používaného

klientom v sieťovej komunikácii. P0f sa zameriava na pasívne odhaľovanie týchto informácií, to znamená, že nevytvára žiadne aktívne spojenie alebo pakety, aby zistil operačný systém alebo prehliadač. P0f pracuje na základe analýzy rôznych charakteristík sieťovej komunikácie, ako napríklad okamžitej odozvy, reakčných časov a ďalších parametrov. Na základe týchto charakteristík dokáže p0f určiť operačný systém, prehliadač a ďalšie informácie o klientovi. Tieto informácie môžu byť veľmi užitočné pri identifikácii a sledovaní aktivít v sieti, ako aj pri rôznych bezpečnostných opatreniach.[11]

1.4.2 Prevencia pred útokmi

V tejto kapitole si vysvetlíme ako správne predísť úspešným útokom hrubou silou.

Kontrola a požadovanie silného hesla

K najbežnejším útokom hrubou silou patrí slovníkový útok, z tohto dôvodu je nesmierne dôležité, aby hesla obsahovali čísla, špeciálne znaky, malé a veľké písmena. Treba sa vyhnúť bežne používaným heslám ako sú: password, heslo, 123456 a pod. Na zabezpečenie a kontrolu silného hesla slúži PAM Cracklib modul. Je navrhnutý na presadzovanie zásad silných hesiel prostredníctvom kontroly sily hesiel a odmietania slabých hesiel počas overovania používateľa. Modul používa súbor preddefinovaných pravidiel na kontrolu kvality hesiel, ako je minimálna dĺžka, zložitosť a podobnosť s predchádzajúcimi heslami. Ak heslo nespĺňa zadané kritériá, modul ho odmietne a vyzve používateľa, aby si zvolil silnejšie heslo.

Zamedzenie prihlásenia root

Ďalším častým spôsobom útoku hrubou silou je pokus prihlásenie sa pomocou užívateľského mena root, ktoré slúži ako administrátor na všetkých unixových systémoch. Ako sa píše v kapitole 1.1 vypnutím možnosti prihlásenia sa pomocou root vieme zabrániť takémuto útoku. Správca si vie nastaviť na prihlásenie iné používateľské meno.

Využitie neštandardného portu

Bežné sieťové služby majú priradený štandardný port, napríklad SSH má port s číslom 22. Pri manuálnej zmene tohto čísla dokážeme predísť automatickým útokom, nakoľko tento útočník nebude vedieť, na akom porte sa nachádzame. Avšak toto nastavenie

znepríjemní užívateľom sa prihlásiť, nakoľko budú potrebovať zadávať nový port. Takisto útočník, ktorý bude chcieť cielene na nás zaútočiť nebude mať problém, pomocou skenovania všetkých portov na SSH, zistiť na ktorom porte sme.

Využitie nepredvídateľných loginov

Toto platí najmä pre firmy, ktoré na tvorbu prihlasovacích mien používajú kombináciu priezviska zamestnanca a čísiel. Pri cielenom útoku na takúto firmu môže útočník využiť verejne známe údaje o zamestnancoch a ľahšie preniknúť do systému. Aby sa tomuto firma vyhla môže vytvárať unikátne loginy pre svojich zamestnancov, čo ale môže skomplikovať spravovanie užívateľov, preto firmy radšej dbajú na silne heslá.

Povolené IP adresy

Pri niektorých špeciálnych prípadoch stačí, že sa na server môže prihlásiť iba obmedzení počet IP adries. V takom prípade sa firewall môže nakonfigurovať, že bude povoľovať prihlásenie iba IP adresám v zozname. Tým sa zabráni prihláseniu útočníka. Samozrejme vo väčšine prípadov je takýto postup nežiadúci.

Port-knocking

Port-knocking je technika, ktorá sa používa na skrytie sieťových portov pred potenciálnymi útočníkmi. Ide o metódu, pri ktorej sa používateľ snaží získať prístup k systému tým, že sa pokúsi v určitej sekvencii pripojiť k určitým portom, čo môže spôsobiť otvorenie portov a umožniť prístup na systém. Táto sekvencia otvárania portov sa označuje ako "knock". Port-knocking môže byť použité na zvýšenie bezpečnosti systému, pretože útočníkovi je ťažšie zistiť, ktoré porty sú otvorené, keď sú skryté a nie sú viditeľné v bežnej sieťovej komunikácii. Ak sa neznáma osoba snaží získať prístup na systém bez znalosti správnej sekvencie "knock", porty sa neotvoria a útočník nebude mať prístup k systému. [12]

Prihlásenie využitím DSA/RSA kľúča

Prihlasovanie pomocou asymetrických kľúčov je založené na asymetrickej kryptografii. Používateľ pomocou príkazu *ssh-keygen* vygeneruje dvojicu kľúčov, súkromný a verejný kľúč, ktoré sa potom vytvoria v adresári /.ssh/ pod názvami id_rsa

a id_rsa.pub podľa zadaných parametrov. Verejný kľúč sa vloží do súboru /.ssh/authorized keys na serveri.

Keď sa klient pripojí k serveru, overí sa pravosť súkromného kľúča používateľa a v prípade úspechu sa používateľ prihlási a už nemusí zadávať heslo. Na rozdiel od predtým uvedených metód, ktoré komplikovali prihlasovanie používateľov, táto metóda uľahčuje prihlasovanie. SSHD sa potom môže nakonfigurovať tak, aby vôbec neumožňoval prihlasovanie heslom a umožňoval len prihlasovanie kľúčom, pozri opis konfigurácie sshd v časti 1.1.

Blokovanie IP adries s neplatným prihlásením

Princíp tejto metódy je založený na automatickom prehľadávaní logov SSHD a blokovaní IP adries, ktoré sa neúspešne pokúsili o prihlásenie a tým pádom sú podozrivé z útoku hrubou silou. V kapitole 1.5 sa budeme venovať dostupným softvérom, ktoré túto funkciu majú avšak majú aj nedostatky. Moja aplikácia bude mať podobnú funkcionalitu so špecifickými vlastnosťami.

1.4.3 Blokovanie prístupov podľa IP adresy

Ako bolo spomínané v 1.4.2 jednou zo základných metód prevencie je blokovanie podozrivých IP adries. Unixové systémy môžu na blokovanie prístupu použiť niektoré z týchto nástrojov.

TCP Wrappers

Ide o filter pracujúci na aplikačnej úrovni. Balíky TCP Wrappers (tcp_wrappers a tcp_wrappers-libs) sú predvolene nainštalované a poskytujú riadenie prístupu k sieťovým službám na základe hostiteľa. Najdôležitejšou súčasťou balíka je knižnica */lib/libwrap.so* alebo */lib64/libwrap.so*. Všeobecne povedané, služba využivajúca protokol TCP je služba, ktorá bola skompilovaná proti knižnici libwrap.so. Pri pokuse o pripojenie k službe s TCP Wrapper sa služba najprv odvolá na prístupové súbory hostiteľa (/etc/hosts.allow a /etc/hosts.deny), aby určila, či je klientovi povolené pripojenie. Vo väčšine prípadov potom použije daemona syslogd na zápis názvu žiadajúceho klienta a požadovanej služby do /var/log/secure alebo /var/log/messages.

Ak je klientovi povolené pripojenie, TCP Wrappers uvoľní kontrolu nad pripojením k požadovanej službe a ďalej sa nezúčastňuje na komunikácii medzi klientom a serverom. Okrem riadenia prístupu a protokolovania môžu TCP Wrappers vykonávať príkazy na interakciu s klientom pred odmietnutím alebo uvoľnením kontroly nad pripojením k požadovanej sieťovej službe. [13]

Blackhole route

"Blackhole route" (známa aj ako null route) je sieťová trasa, ktorá sa používa na odmietnutie paketov určených pre konkrétnu adresu IP alebo rozsah adries. Keď smerovač prijme paket, ktorý zodpovedá nulovej trase, paket zahodí namiesto toho, aby ho preposlal do cieľa. V systéme Linux môžeme vytvoriť nulovú trasu pomocou príkazu "route". Ak chceme napríklad vytvoriť nulovú trasu pre IP adresu 185.134.198.175, použite nasledujúci príkaz:

sudo route add 185.134.198.175 gw 0.0.0.0 netmask 255.255.255.255

Tento príkaz pridá do smerovacej tabuľky trasu, ktorá nasmeruje všetku prevádzku pre IP adresu 185.134.198.175 na "čiernu dieru" bránu 0.0.0.0. [14]

Iptables

Nástroj Iptables, ktorý využíva operačný systém Linux je určený na konfigurovanie pravidiel IPv4 paketového filtra. Program umožňuje pridávať, odstraňovať a vypisovať jednotlivé pravidlá, ako aj získavať štatistiky o paketoch spracovaných podľa jednotlivých pravidiel. Pre blokovanie paketov z IP adries sa využívajú základné príkazy –A (pridanie nového pravidla) a –D (zmazanie predtým pridaného pravidla). Jednoduchý kód na blokovanie IP adresy by vyzeral takto:

$ iptables -A INPUT -s IP adresa -j REJECT

$ iptables -D INPUT -s IP adresa -j REJECT

Prvý príkaz vloží nové pravidlo do takzvaného reťazca INPUT, v ktorom sa najprv vyhľadajú pravidlá pre prichádzajúce pakety a prikáže, že všetky pakety prichádzajúce zo zadanej IP adresy budú odmietnuté. Akcia REJECT reaguje príslušným paketom ICMP, ktorý oznamuje, že spojenie sa nepodarilo nadviazať. Ďalšou možnosťou je použitie akcie DROP, ktorá paket zahodí a nevráti odosielateľovi žiadnu odpoveď. [15]

UFW

UFW (Uncomplicated Firewall) je front-end pre iptables, čo je vstavaný nástroj firewallu v Linuxe. UFW uľahčuje konfiguráciu iptables tým, že poskytuje užívateľsky prívetivé rozhranie a zjednodušuje proces vytvárania a spravovania pravidiel brány firewall. UFW je navrhnutý tak, aby sa ľahko používal, ale poskytuje aj pokročilé funkcie pre skúsenejších používateľov. [16]

Medzi funkcie UFW patria:

- Jednoduché rozhranie: UFW poskytuje jednoduché a intuitívne rozhranie, ktoré uľahčuje vytváranie a spravovanie pravidiel brány firewall.

- Jednoduchá konfigurácia: UFW sa dodáva s preddefinovanými konfiguráciami pre bežné aplikácie, ako sú SSH, Apache a FTP.

- Rozšírené nastavenia: UFW poskytuje pokročilé možnosti pre skúsenejších používateľov, napríklad možnosť vytvárať vlastné pravidlá, konfigurovať preklad sieťových adries (NAT) a spravovať presmerovanie portov.

- Logging: UFW môže zaznamenávať všetky aktivity brány firewall, aby vám pomohol identifikovať a odstrániť problémy.

1.5 Analýza existujúcich riešení

Na ochranu SSH pred útokmi hrubou silou existuje na trhu množstvo nástrojov. Tieto nástroje majú na menšie rozdiely veľmi podobnú funkcionalitu a to, že skenujú logy SSHD a hľadajú podozrivé IP adresy a neplatné pokusy o prihlásenie. Následne postupnosťou krokov sa snažia zabrániť takýmto adresám prístup k počítaču. Nižšie sú popísané štyri takéto najviac využívané nástroje.

1.5.1 SSHGuard

SSHGuard je multiplatformný softvérový nástroj napísaný v jazyku C, ktorý je určený na ochranu serverov pred útokmi hrubou silou na rôzne služby, ako sú SSH, FTP a

SMTP. Funguje tak, že monitoruje log files na zlyhania autentifikácie a keď zistí nadmerný počet neúspešných pokusov o prihlásenie z konkrétnej IP adresy, zablokuje túto IP adresu pred ďalšími pokusmi. Výhodou je, že je vysoko konfigurovateľný a umožňuje správcom nastaviť prahové hodnoty neúspešných pokusov o overenie, konfigurovať možnosti blokovania a určiť pravidlá whitelistu a blacklistu. Možno ho tiež nakonfigurovať tak, aby v prípade zistenia útoku odosielal oznámenia prostredníctvom e-mailu alebo iných prostriedkov. [17]

Jednou zo silných stránok SSHGuard je jeho schopnosť spolupracovať s rôznymi firewallmi vrátane iptables, pf a ipfw, ako aj komunikovať s inými bezpečnostnými nástrojmi, ako sú Snort a Fail2Ban. To z neho robí flexibilný a všestranný nástroj, ktorý možno integrovať do existujúcich bezpečnostných nastavení. Nástroj aj s dokumentáciou je voľne dostupný na *https://www.sshguard.net/* .

1.5.2 Fail2Ban

Fail2Ban je softvérový nástroj s otvoreným zdrojovým kódom vytvorený v jazyku python, ktorý je navrhnutý tak, aby zabránil útokom hrubou silou monitorovaním log files a blokovaním IP adries po určitom počte neúspešných pokusov o prihlásenie. Bežne sa používa na ochranu serverov pred útokmi hrubou silou SSH a FTP, ale výhodou je, že ho možno použiť aj na ochranu iných služieb, ako je webový server Apache, poštový server Postfix a iné. [18]

Fail2Ban funguje tak, že analyzuje log files a hľadá vzory, ktoré naznačujú neúspešné pokusy o prihlásenie, napríklad neplatné používateľské mená alebo heslá. Po zistení určitého počtu neúspešných pokusov z konkrétnej IP adresy Fail2Ban zablokuje túto IP adresu pomocou pravidiel brány firewall systému. Adresu IP možno odblokovať po uplynutí určeného časového obdobia alebo manuálne správcom systému. Nástroj aj s dokumentáciou je voľne dostupný na *https://www.fail2ban.org/* .

1.5.3 Snort

Snort je bezplatný open-source systém prevencie a detekcie prienikov, ktorý sa široko používa na detekciu a prevenciu rôznych typov útokov vrátane útokov hrubou silou. Funguje tak, že analyzuje sieťovú prevádzku a porovnáva ju so súborom pravidiel na identifikáciu potenciálnych hrozieb. Ak sa zistí hrozba, Snort môže vykonať rôzne akcie,

napríklad zaznamenať udalosť, upozorniť správcu systému alebo zablokovať prevádzku. Dokáže odhaliť útoky hrubou silou prostredníctvom detekcie, ktorá zahŕňa porovnávanie sieťovej prevádzky so súborom vopred nakonfigurovaných pravidiel, ktoré opisujú správanie známych útokov. Tieto pravidlá možno prispôsobiť tak, aby zodpovedali špecifickým potrebám organizácie a možno ich pravidelne aktualizovať, aby držali krok s novými hrozbami. [19]

Jednou z funkcií je možnosť ho nakonfigurovať aj na analýzu súborov denníkov a zisťovanie útokov hrubou silou prostredníctvom detekcie založenej na správaní, ktorá zahŕňa identifikáciu vzorcov v správaní používateľov alebo systémov, ktoré naznačujú potenciálny útok. To môže byť obzvlášť užitočné pri odhaľovaní útokov, ktoré zahŕňajú viacero pokusov o prihlásenie počas dlhšieho obdobia. [19] Nástroj aj s dokumentáciou je voľne dostupný na *https://www.snort.org/*.

1.5.4 DenyHosts

Projekt denyhosts je zaujímavý tým, že na rozdiel od predchádzajúcich nástrojov udržiava centrálnu globálnu databázu identifikovaných útočníkov a jednotliví klienti do nej môžu prispievať svojimi úlovkami a čerpať z nej. Denyhosts sa môže spúšťať v určitých intervaloch pomocou cronu alebo sa môže spúšťať ako démon, kde sa v určitých intervaloch spúšťa aj spustiteľná časť programu. Program sleduje poslednú načítanú pozíciu denníka, takže vie, kde má začať pri ďalšom načítaní. Denyhosts ponúka pomerne rozsiahle možnosti konfigurácie, kde je možné nastaviť napríklad rôzne limity počtu neplatných pokusov o prihlásenie pre existujúcich používateľov počítača, pre neexistujúcich používateľov a pre používateľa root.

Už spomínaná funkcia distribúcie získaných útočných IP adries prostredníctvom centrálneho servera je nepochybne veľmi zaujímavá, ale prináša so sebou aj riziká podvrhnutých IP adries, v systéme nie je žiadny autentizačný mechanizmus, čo by pri dostupnom zdrojovom kóde aj tak nemalo veľký zmysel. Tento problém je tu čiastočne riešený použitím dvoch dodatočných konfiguračných smerníc, ktoré umožňujú klientovi nastaviť minimálny počet iných klientov, ktorí musia nahlásiť danú IP adresu, aby bola zo servera odoslaná a minimálny rozsah, v ktorom bola daná IP adresa nahlásená od rôznych

klientov [20]. Nástroj aj s dokumentáciou je voľne dostupný na https://denyhosts.sourceforge.net/index.html .

1.5.5 Porovnanie nástrojov

V tejto podkapitole si v skratke porovnáme vyššie spomínané nástroje.

Pôvodná konfigurácia

V nasledujúcom obrázku 2 je porovnanie základných konfigurácií nástrojov proti útokom na SSH. Hodnota v prvom riadku teda *findtime* určuje časový interval, v ktorom musí dôjsť k počtu neplatných pokusov o prihlásenie špecifikovaných v druhom riadku (*maxretry*), aby daná útočiaca IP adresa bola zablokovaná. Tretí riadok *bantime* potom určuje dĺžku doby, po ktorej je prístup pre zablokovanú IP adresu znova povolený.

Hodnota	SSHGuard	Fail2Ban	Snort	DenyHosts
`findtime` (sekundy)	120 - 1800	600	60	600
`maxretry`	4 - 6	5	N/A	3
`bantime` (sekundy)	3600	600	N/A	600

Obrázok 2 Defaultná konfigurácia nástrojov

Základná hodnota *findtime* pre sshguard sa líši od verzie nastroja v rozmedzí 120 až 1800 sekúnd. Pre snort je to 60 sekúnd avšak je potrebné poznamenať, že Snort používa iný parameter nazývaný *treshold* na určenie počtu udalostí, ktoré musia byť zaznamenané počas obdobia nájdenia, aby sa vykonala akcia, napríklad spustenie upozornenia. Predvolená hodnota *tresholdu* je 1. Hodnota *maxretry* sa líši od verzie sshguardu a snort využíva iné pravidlá na určenie rizika. Na rozdiel od ostatných nástrojov, o ktorých sme hovorili, Snort aktívne neblokuje ani nezakazuje IP adresy. Namiesto toho generuje výstrahy, keď na základe svojho súboru pravidiel zistí podozrivú sieťovú aktivitu a je na správcovi, aby rozhodol, aké kroky podnikne v reakcii na tieto výstrahy.

Nedostatky súčasných nástrojov

Medzi hlavné nedostatky v súčasnosti dostupných nástrojov patria:

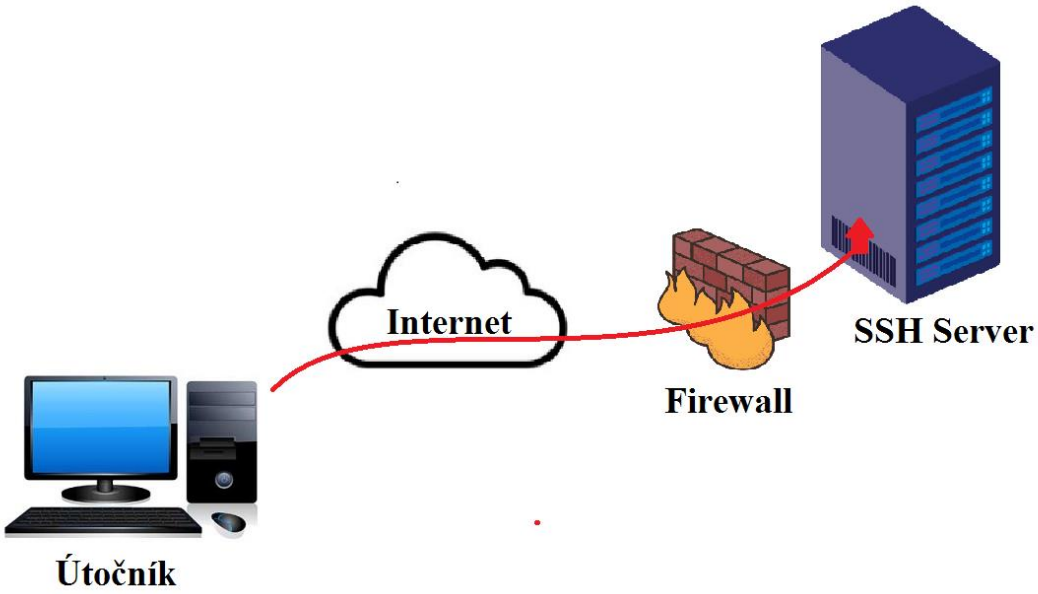
- kontrola logov po riadkoch - jeden zlý pokus môže byť identifikovaný ako niekoľko pokusov, ak služba daemon aj PAM zapíšu do denníka oznámenie o neúspešnom prihlásení.

- beží len na jednom počítači (na niektorých) - obmedzené možnosti detekcie útokov.
- alebo naopak, bežia globálne (niektoré) - existuje celosvetová výmena zistených útočníkov, ale v tomto rozsahu nie je možné účinne kontrolovať podvrhnuté IP adresy.
- neefektívne zdieľanie informácií o útokoch - hoci si niektoré systémy na detekciu a blokovanie útokov vymieňajú informácie o blokovaných adresách, vyhodnocovanie sa vždy vykonáva len na jednotlivých staniciach a preto nie je možné efektívne odhaliť všetky útoky, ktoré by sa dali odhaliť vzájomným porovnaním informácií z viacerých staníc.
- odhaľujú útoky len na základe IP adresy - často dochádza k distribuovaným útokom, ktoré sú však spojené podľa použitého používateľského mena.
- nezoskupujú útočníkov - to neumožňuje detekciu distribuovaných útokov, užitočným zlepšením by bolo inteligentné vytváranie vzťahov medzi jednotlivými IP adresami útočníkov, aby bolo možné neskôr blokovať celé skupiny na základe niekoľkých zhodných adries.
- nekonzistentné údaje o zablokovaných útočníkoch - po ukončení nástroja sa informácie o útočníkoch a ich zablokovaní zlikvidujú alebo zostanú v niektorom protokole vytvorenom nástrojom, ale už sa nikdy nepoužijú.

2 Cieľ práce, metodika práce a metódy skúmania

2.1 Návrh aplikácie

Náš systém bude tvorený z python skriptu, ktorý obsahuje celý mozog aplikácie. Tento systém bude bežať na servery, ku ktorému sa budú pomocou SSH pripájať rôzni používatelia. Systém bežiaci na servery potom deteguje nepodarené pokusy o prihlásenie a blokuje IP adresy týchto potencionálnych útočníkov za pomoci nástrojov *iptables*. Tieto zablokované IP adresy budú následne zapísané do textového súboru, ktorý bude obsahovať IP adresu, čas pokusu o prihlásenie a krajinu, z ktorej sa tento užívateľ snažil pripojiť.



Obrázok 3 Schéma systému

2.1.1 Spôsob detegovania a blokácie

Detegovanie útokov bude prebiehať viacerými spôsobmi

- Viacero nepodarených pokusov o prihlásenie z jednej IP adresy.
- Viacero pokusov o prihlásenie z jednej IP adresy v krátkom časovom intervale.

Pre správne detegovanie potencionálnych útočníkov je dôležité zhromažďovať rôzne informácie, vďaka ktorým systém dokáže pracovať efektívnejšie a s lepšou presnosťou. Medzi tieto informácie patria:

- Najdôležitejšou informáciou je samozrejme IP adresa útočníka.
- Použitý login môže takisto pomôcť pri vylepšení aplikácie.
- Dátum a čas prístupu je tiež veľmi dôležitá informácia.
- V neposlednom rade chceme zistiť krajinu pôvodu IP adresy.

2.1.2 Prínos systému

Implementácia tohto systému môže pomôcť menším firmám a bežným užívateľom chrániť svoj server pred útokmi hrubou silou. Kvôli jednoduchosti systému by mala umožňovať administrátorovi siete jednoduchšiu konfiguráciu celého systému. Tento systém by mal byť teda prínosom najmä pre malé a stredné siete.

2.1.3 Ciele

- V prvom rade by mal systém implementovať funkcionalitu dostupných nástrojov na blokovanie útokov hrubou silou.
- Ďalej by mal systém ukladať záznamy o týchto útokov v zrozumiteľnej forme.
- Systém by mal taktiež využiť službu GeoIP2 na identifikáciu krajiny pôvodu útoku.

2.1.4 Riziká

Pri každom návrhu aplikácie vznikajú taktiež riziká, ktoré môžu spôsobiť nežiadúce efekty v našej sieti preto treba na tieto riziká dbať ohľad pri implementácií.

- Hlavný problém systému môže vzniknúť pri nesprávnej konfigurácií alebo pri nesprávnom použití. V tomto prípade môže dôjsť k blokovaniu legitímnych užívateľov a spôsobiť tak problémy k prístupu našej siete.
- Pri príliš prísnom blokovaní podozrivých IP adries môžeme stratiť dôležité informácie, ktoré by sme mohli získať v ďalších krokoch a tým pádom znížime úspešnosť obrany pred budúcimi útokmi. Na tento problém by sa dali využiť honeypots, ktorí nebudú blokovať žiadne IP adresy ale iba pozorovať proces útoku.

- Samotný systém nás neochráni pred všetkými typmi útokov. Napríklad útoky typu DDoS (distributed denial of service) môžu znefunkčniť správny chod servera a tým nám zabrániť v práci. Avšak tento problém sa dá vyriešiť doplnením kódu o ochranu pred DDoS.

3 Výsledky práce a diskusia

3.1 Implementácia systému

V tejto kapitole sa budeme venovať implementácií jednotlivých skriptov. Taktiež si priblížime popis všetkých technológií (zariadení a softvérov), ktoré boli použité na tvorbu tejto práce. Odôvodníme si aj výber programovacieho jazyka, v ktorom bol skript napísaný.

Naša aplikácia sa volá TOVR (čítaj: TOWER). V ďalších podkapitolách budeme pomocou tohto názvu odkazovať na aplikáciu.

3.1.1 Voľba programovacieho jazyka

Pri výbere programovacieho jazyka sme sa rozhodovali medzi jazykmi Python a C, nakoľko podobné dostupné nástroje ako sme mohli vidieť v podkapitole 1.5 využívajú tieto dva jazyky. Nastroj Fail2Ban využíva jednoduchší a prehľadnejší jazyk Python, zatiaľ čo SSHGuard využíva komplexnejší jazyk C. Našim cieľom bolo najmä, aby naša aplikácia bola spustiteľná na akejkoľvek distribúcii operačného systému GNU/Linux. Preto sme potrebovali jazyk, ktorý je týmito operačnými systémami podporovaný. Jedným z hlavných faktorov pri výbere programovacieho jazyka bola jednoduchosť kódu, aby sme aj pri dlhšom kóde nestrácali prehľadnosť. Keďže náš kód patrí medzi jednoduchšie, rozhodli sme sa využiť programovací jazyk Python, ktorý je aj pri plánovanom budúcom vylepšení ľahšie modifikovateľný. Jazyk Python je dostupný vo viacerých verziách, avšak my sme sa rozhodli využiť verziu Python 3.11.2 , z toho dôvodu, že už bola nainštalovaná na servery.

3.1.2 Technológie

V tejto podkapitole si povieme o tých nástrojoch/softvéroch, ktoré boli potrebné pri tvorbe tejto práce.

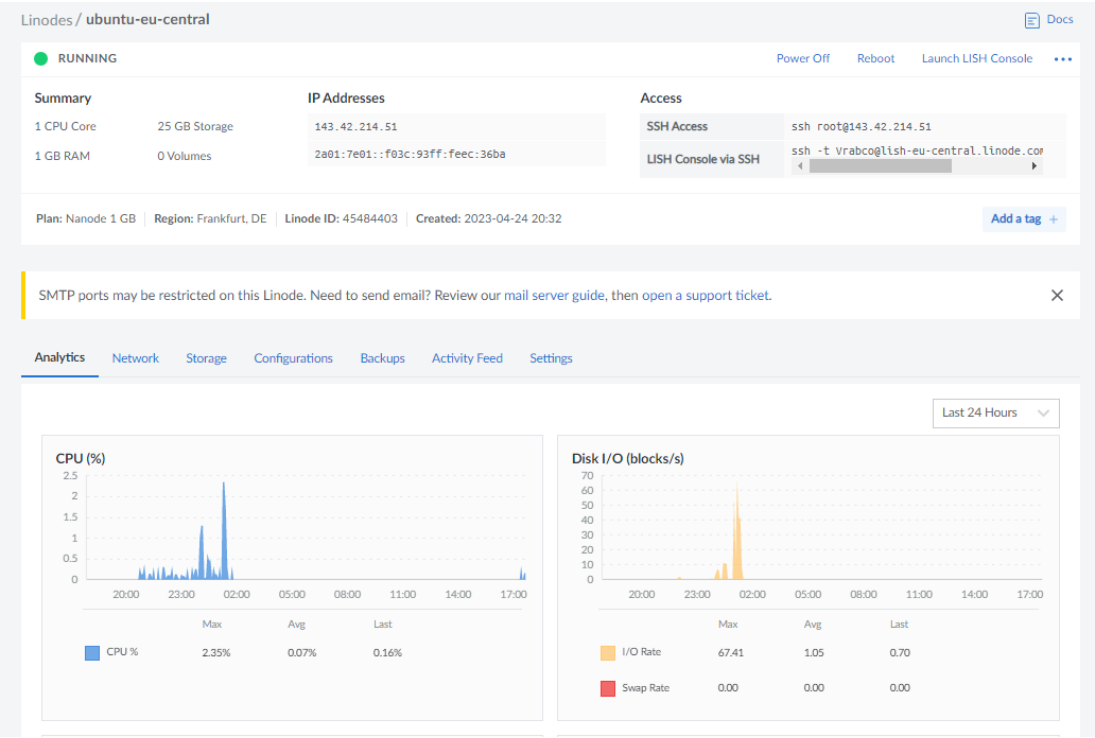
Server

Ako sme už spomínali pri návrhu aplikácie, aplikácia TOVR je spustená na servery, kde číta logy pripojení a hľadá IP adresy s neplatným prihlásením, ktoré potom zablokuje. Na to aby mohla byť spustená, je za potreby server. Na trhu je veľa dostupných spoločností, ktoré ponúkajú cloudové linuxové servery. Z počiatku sme pracovali s *google cloud* , ktorý pri registrácií ponúka zadarmo služby v hodnote 300 dolárov. Avšak lepšie sa nám pracovalo s cloudovým serverom od spoločnosti Linode.

Linode je poskytovateľ cloudového hostingu, ktorý ponúka virtuálne privátne servery (VPS) pre jednotlivcov a firmy. Bola založená v roku 2003 a sídli vo Filadelfii v Pensylvánii v USA. Spoločnosť Linode poskytuje celý rad cloudových hostingových služieb vrátane hostingu VPS, dedikovaných inštancií CPU, hostingu Kubernetes, objektového úložiska a spravovaných databáz. Ponúka používateľsky prívetivý ovládací panel, rozhranie API a nástroje CLI na správu a nasadzovanie serverov, ako aj celý rad vopred pripravených obrazov a inštalácií populárnych aplikácií a nástrojov jedným kliknutím.

Plány VPS spoločnosti Linode sú škálovateľné a prispôsobiteľné, takže si zákazníci môžu vybrať z radu plánov s rôznymi konfiguráciami procesora, pamäte, úložiska a šírky pásma. Linode ponúka aj pokročilé funkcie, ako je automatické zálohovanie, vyrovnávanie záťaže a správa DNS. Spoločnosť má dátové centrá vo viacerých regiónoch po celom svete vrátane USA, Kanady, Európy, Ázie a Austrálie, čo zákazníkom umožňuje vybrať si lokalitu, ktorá je najbližšie k ich používateľom, aby dosiahli optimálny výkon. [21]

Po registrácií a prihlásení môžeme vytvoriť náš server. Môžeme si nastaviť rôzne atribúty napríklad RAM, CPU avšak s lepšími parametrami stúpa aj cena. Dôležitá je taktiež voľba regiónu, pre ktorú sme si zvolili Frankfurt, aby bola čo najmenšia odozva. Po vytvorení a spustení servera sa nám ukáže takáto stránka so všetkými potrebnými údajmi. Najdôležitejší údaj je IP adresa nášho servera.



Obrázok 4 Linode interface

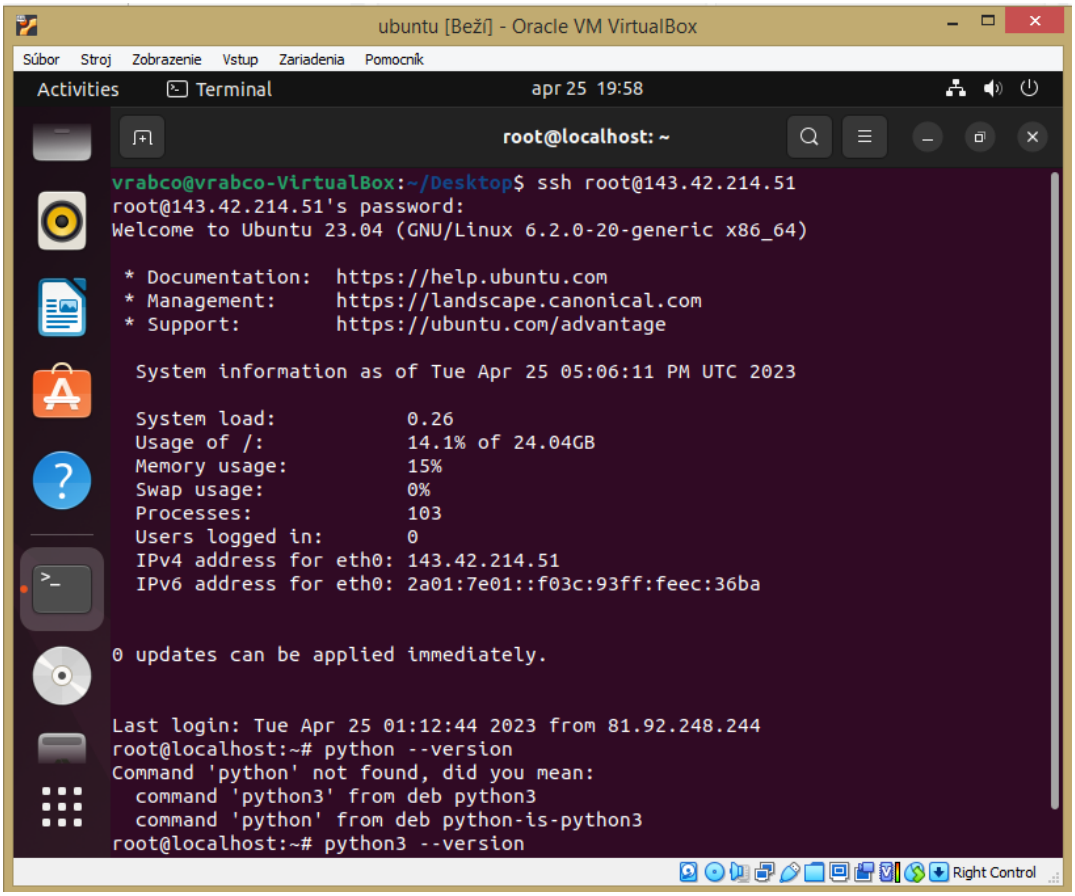
Virtual Machine

Aby sme mohli spravovať administratívu Linode servera potrebovali sme sa prihlásiť ako SSH správca. Na prihlásenie sme potrebovali zariadenie s operačným systémom Linux. Nakoľko osobný počítač má operačný systém Windows, rozhodli sme sa využiť virtual machine. Na trhu je takisto množstvo aplikácií na tvorbu virtuálneho stroja ale medzi hlavné patri OracleVMVirtualBox a VMware. Po naštudovaní manuálov a dokumentácií sme sa priklonili k využitiu OracleVMVirtualBox. Po stiahnutí tohto softvéru bolo potrebné vytvoriť nový virtuálny stroj a nainštalovať naňho operačný systém Linux. To sme spravili vďaka inštalačnému súboru ISO. Nainštalovali sme distribúciu operačného systému Linux a to UBUNTU verziu 22.04.2, ktorá je založená na linuxovej distribúcii Debian.

Po spustení virtuálneho stroja sme sa jednoducho cez terminál pripojili k SSH serveru pomocou kódu:

ssh root@143.42.214.51

Po zadaní hesla zvoleného pri tvorbe servera sme sa úspešne prihlásili ako je vidieť na obrázku 5.



Obrázok 5 Pripojenie sa na SSH server cez Ubuntu terminál

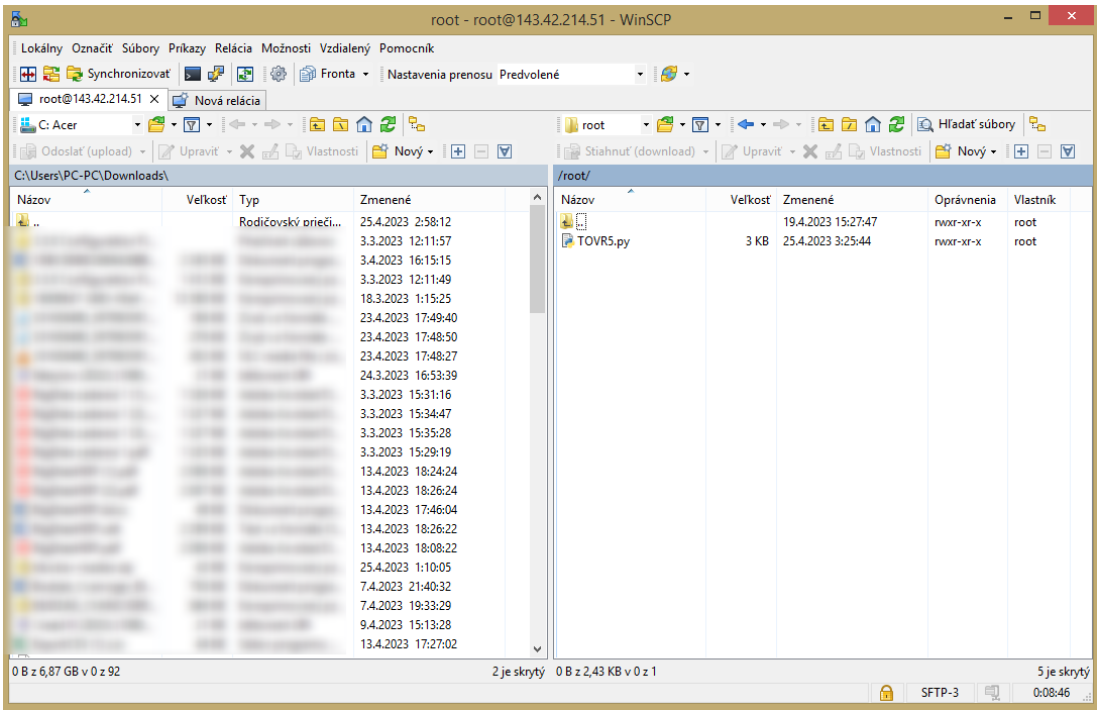
Po prihlásení ako správca sme mohli voľne zadávať ďalšie príkazy, ktoré sa mali vykonať na servery.

Prenos súborov

Pre správne fungovanie python skriptu bolo za potreby tento skript vložiť do adresára Linode servera. To sa ukázalo ťažšie ako sme očakávali. Po testovaní viacero metód sme sa rozhodli využiť softvér WinSCP.

WinSCP je bezplatný open-source klient pre prenos súborov medzi lokálnym a vzdialeným počítačom, ktorý beží na operačných systémoch Windows. Je určený pre použitie s protokolmi FTP, SFTP, SCP, WebDAV a S3. Tento nástroj umožňuje jednoduchý prenos súborov medzi lokálnou a vzdialenou zložkou pomocou prehľadného užívateľského rozhrania. Obsahuje funkcie ako napríklad synchronizáciu zložiek, správu obľúbených zložiek a súborov, editáciu súborov priamo na vzdialenom serveri a podporu pre zložky a súbory chránené heslom. Taktiež umožňuje použitie vlastných skriptov pre

automatizáciu procesov prenosu súborov. Na prihlásenie sme použili prenosový protokol SFTP. Po zadaní IP adresy nášho servera a hesla sa nám otvorí používateľské rozhranie vďaka ktorému už bolo jednoduché python skript nahrať do adresára servera.



Obrázok 6 WinSCP transfer súborov

Viac o tomto nástroji sa môžete dozvedieť na stránke: https://winscp.net/eng/index.php

GeoIP2

Ďalším nástrojom, ktorý bol tento krát použitý priamo v skripte je GeoIP2. Ide o databázu viac ako 99% IP adries a k nim priradených krajín. Túto databázu spravuje spoločnosť Maxmind. Táto služba obsahuje balíky, ktoré môžeme priamo v našom skripte využiť na priradenie krajiny k IP adrese. Pre viac informácií môžete navštíviť ich webovú stránku: https://www.maxmind.com/en/home.

Hydra

Hydra je nástroj pre útok hrubou silou, ktorý používa automatizovaný proces na testovanie rôznych kombinácií prihlasovacích údajov, ako sú užívateľské mená a heslá, až kým nenájde správnu kombináciu. Tento proces môže byť veľmi časovo náročný, ale ak sa

použije správne, môže byť veľmi účinný pri získavaní neoprávnených prístupov k cieľovým účtom alebo zariadeniam. Tento nástroj sme použili spolu s VPN pri testovaní.

3.1.3 Spustenie skriptu

Po nahraní python súboru za pomoci WinSCP môžeme tento skript spustiť. V prvom rade sa musíme uistiť, že náš skript má spustiteľné oprávnenia. To skontrolujeme použitím nasledujúceho príkazu v termináli: *chmod +x TOVR.py*.

Po zadaní oprávnenia spustíme skript pomocou príkazu: *python3 TOVR.py &* . Apersand na konci príkazu nám dovolí chod skriptu na pozadí. Príkazom: *ps aux | grep TOVR.py* , môžeme skontrolovať či skript beží.

Pozn.: Musíme sa nachádzať v priečinku, kde sa nachádza náš skript.

3.1.4 TOVR.py

V tejto podkapitole si vysvetlíme jednotlivé funkcie samotného skriptu. Celý kód môžete nájsť v prílohe A.

- *import time* - Tento riadok importuje vstavaný časový modul, ktorý sa používa na uspanie skriptu na určitý čas.
- *import subprocess* - Tento riadok importuje vstavaný modul podprocesu, ktorý sa používa na spúšťanie a správu externých príkazov shellu.
- *import re* - Tento riadok importuje vstavaný modul re, ktorý sa používa na porovnávanie regulárnych výrazov.
- *import os* - Tento riadok importuje vstavaný modul os, ktorý sa používa na operácie so súbormi a adresármi.
- *import geoip2.database* - Tento riadok importuje modul geoip2.database, ktorý sa používa na prácu s databázami GeoIP2 spoločnosti MaxMind.
- *max_failed_attempts = 3* - Tento riadok nastavuje maximálny počet neúspešných pokusov pred zakázaním IP adresy. Zmenou tejto hodnoty môžeme meniť dovolený počet pred zablokovaním.
- *access_log_file = '/var/log/apache2/access.log* - Tento riadok nastavuje cestu k súboru protokolu prístupu Apache.

- *banned_ips_file = '/etc/banned_ips.txt* - Tento riadok nastavuje cestu k súboru, do ktorého sa budú zaznamenávať zakázané IP adresy.
- *geoip_db_file = '/usr/share/GeoIP/GeoLite2-Country.mmdb* - Tento riadok nastavuje cestu k súboru databázy MaxMind GeoIP2.
- *geoip_reader = geoip2.database.Reader(geoip_db_file)* - Tento riadok vytvorí objekt Reader z modulu geoip2.database, ktorý načíta súbor databázy MaxMind GeoIP2.
- *banned_ips = []* - Tento riadok inicializuje prázdny zoznam, ktorý obsahuje zakázané IP adresy.
- *if os.path.isfile(banned_ips_file):* - Tento riadok kontroluje, či existuje súbor so zakázanými IP adresami.
- *with open(banned_ips_file, 'r') as f:* - Tento riadok otvorí súbor so zakázanými IP adresami na čítanie.
- *banned_ips = [line.strip() for line in f*] - Tento riadok načíta každý riadok súboru so zakázanými IP adresami a pridá ho do zoznamu banned_ips po odstránení všetkých bielych znakov.
- *tail_command = ['tail', '-f', access_log_file]* - Tento riadok vytvorí príkaz shellu, ktorý priebežne vypíše nové riadky pridané do súboru denníka prístupu.
- *tail_process = subprocess.Popen(tail_command, stdout=subprocess.PIPE, stderr=subprocess.PIPE)* - Tento riadok otvorí podproces na spustenie príkazu tail a prečíta jeho výstup.
- *for line in iter(tail_process.stdout.readline, b''):* - Tento riadok prečíta každý riadok outputu príkazom tail z podprocesu.
- *match = re.search(r'^(\d+\.\d+\.\d+\.\d+)', line.decode('utf-8'))* - Tento riadok sa pokúša porovnať IP adresu na začiatku každého riadku access logu pomocou regulárneho výrazu.
- *if match:* - Tento riadok kontroluje, či bola nájdená zhoda.
- *ip_address = match.group(1)* - V tomto riadku sa zo zhody regulárneho výrazu vyberie zodpovedajúca IP adresa.

- *if ip_address in banned_ips:* - Tento riadok kontroluje, či už bola IP adresa zakázaná.
- *failed_attempts = 0* - Tento riadok inicializuje počítadlo počtu neúspešných pokusov o prihlásenie podľa IP adresy.
- *for line2 in open(access_log_file):* - Tento riadok opäť otvorí súbor denníka prístupu a prečíta každý riadok.
- *for line in iter(tail_process.stdout.readline, b''):* - Spustí sa slučka, ktorá číta riadok po riadku súbor protokolu prístupu Apache pomocou príkazu tail. Cyklus bude pokračovať donekonečna, pretože do súboru sa pridávajú nové riadky.
- *match = re.search(r'^(\d+\.\d+\.\d+\.\d+)', line.decode('utf-8'))* - Na extrakciu IP adresy z aktuálneho riadku súboru protokolu sa použije regulárny výraz. Hľadá reťazec, ktorý začína sériou číslic a bodiek, čo by mala byť IP adresa. Ak nájde zhodu, uloží IP adresu do premennej match.
- *if ip_address in banned_ips: continue* - Táto funkcia skontroluje, či aktuálna IP adresa už bola zakázaná a ak áno, preskočí na ďalší riadok cyklu.
- *if ip_address in line2 and ' 401 ' in line2:* - Táto funkcia kontroluje, či aktuálny riadok súboru protokolu prístupu obsahuje aktuálnu IP adresu a stavový kód 401, ktorý označuje neúspešný pokus o prihlásenie.
- *failed_attempts += 1* - Ak sa aktuálny riadok súboru protokolu prístupu zhoduje, zvýši sa počet neúspešných pokusov o prihlásenie pre aktuálnu IP adresu.
- *if failed_attempts >= max_failed_attempts:* break - Ak počet neúspešných pokusov o prihlásenie pre aktuálnu IP adresu prekročil povolené maximum (max_failed_attempts), dôjde k prerušeniu vnorenej slučky.
- *banned_ips.append(ip_address)* - Ak bola IP adresa zakázaná, pridá sa do zoznamu zakázaných IP adries.
- *with open(banned_ips_file, 'a') as f: f.write('{} {}\n'.format(ip_address, country_code))* - Otvorí sa súbor so zakázanými IP adresami v režime pridávania a na koniec súboru sa zapíše aktuálna IP adresa a kód krajiny.

- *subprocess.run(['iptables', '-A', 'INPUT', '-s', ip_address, '-j', 'DROP']* - Na blokovanie prichádzajúcej prevádzky zo zakázanej IP adresy sa použije príkaz iptables.
- *print('Banned IP address {} from {}'.format(ip_address, country_code))* - Na konzolu sa vypíše správa, že aktuálna IP adresa bola zakázaná spolu s krajinou pôvodu (ak je k dispozícii).

3.2 Testovanie

Ako bolo spomínané v podkapitole 3.1.2 na testovanie našej aplikácie sme používali nástroj Hydra, ktorý využíva databázu hesiel k útoku hrubou silou. Taktiež sme použili nástroj PuTTY na pokus o pripojenie sa na SSH server z operačného systému Windows. A taktiež sme použili VPN na testovanie funkcie GeoIP2.

Keďže naša aplikácia nevyužíva MySQL databázu ale iba jednoduchý textový súbor na zápis zablokovaných IP adries a kódov krajiny pôvodu, výstup takého súboru bol nasledovný.

```
root@localhost:~# cat banned_ips.txt
162.203.22.29 SK
11.248.105.168 US
35.252.235.87 CN
141.33.243.255 Unknown
181.89.203.222 RU
108.187.214.109 SK
136.185.152.38 CZ
73.0.101.28 US
51.212.93.14 US
104.221.143.228 FR
155.182.239.199 FR
133.66.176.54 SK
15.118.45.227 CN
147.116.129.80 US
```

Obrázok 7 Obsah súboru banned_ips.txt

3.2.1 Vylepšenia

Aplikácia TOVR poskytuje len základnú ochranu pred útokmi hrubou silou. Tým, že je programovaná v jazyku python je ľahké ju v budúcnosti vylepšiť o funkcionalitu.

Jedným z možných vylepšení je vytvorenie MySQL databázy, do ktorej sa budú zapisovať údaje o zablokovaných IP adresách, použitých loginoch a heslách. Takisto sa dá pridať kontrola pokusov o pripojenie z rôznych IP adries z rovnakej siete s C maskou alebo možná detekcia opakovaných vzoriek útočníkov zo staršieho útoku a zablokovanie celej skupiny identifikovaných v predošlom útoku.

Záver

Téma SSH je veľmi aktuálnou problematikou, nakoľko využitie tejto technológie v súčasnosti tvorí obrovskú časť skoro všetkých sietí. V tejto diplomovej práci sme sa venovali analýze zabezpečenia SSH protokolu pred útokmi hrubou silou, nástrojmi na detekciu a ochranu pred takýmito útokmi a implementáciou takého nástroja použitím python skriptu.

V prvej kapitole sme analyzovali, čo je SSH protokol a prečo je dôležité ho zabezpečiť. Následne sme preskúmali najbežnejšie útoky hrubou silou a ako sa môžeme pred takýmito útokmi brániť.

V druhej časti sme sa zameriavali na existujúce nástroje na obranu pred útokmi hrubou silou. Konkrétne sme sa zaoberali nástrojmi fail2ban, sshguard, denyhosts a snort, ktoré sme porovnávali a zhodnotili ich silné a slabé stránky. V nasledujúcej časti sme navrhli vlastnú aplikáciu na ochranu pred útokmi hrubou silou. Aplikácia bola navrhnutá tak, aby bola schopná detegovať útoky hrubou silou a automaticky blokovať IP adresy, ktoré sa pokúšajú o takéto útoky. Takisto naša aplikácia zistila krajinu, z ktorej sa daná IP adresa snažila pripojiť.

V poslednej časti sme otestovali našu aplikáciu a vyhodnotili jej účinnosť v ochrane proti útokom hrubou silou. Výsledky ukázali, že naša aplikácia bola schopná účinne detegovať útoky hrubou silou a zamedziť im.

Výsledkom tejto diplomovej práce je teda funkčná aplikácia TOVR, ktorá dokáže detegovať útok a zabrániť mu avšak táto aplikácia nie je bezchybná. Aplikácia v predvolenej konfigurácii by mala byť vhodná na bežné používanie, ale aj tak by bolo vhodné v budúcnosti vytvoriť pre aplikáciu nejaké pohodlné administračné rozhranie, ktoré by napríklad poskytovalo ďalšie možnosti získavania informácií zo zozbieraných údajov a vizualizácie útokov. Medzi ďalšie možnosti zlepšenia aplikácie by mohlo patriť monitorovanie počítadiel použitého firewallu (ak firewall poskytuje počítadlá) a v prípade ich zvýšenia adekvátne predĺženie času blokovania príslušných IP adries. Na strane servera je stále obrovský priestor na vytvorenie nových algoritmov na vyhodnocovanie útokov a aj tu by bolo možné systém ďalej vylepšovať; medzi prvými kandidátmi by mohol byť

napríklad nejaký pokročilejší zhlukovací algoritmus na prideľovanie útočníkov do skupín a algoritmov, ktorý by priebežne odstraňoval skupiny, o ktorých rozhodne, že ich nemá zmysel ponechať v databáze, čo by mohli byť napríklad skupiny obsahujúce menej útočníkov, ako je dolná hranica pre blokovanie skupiny alebo skupiny obsahujúce len útočníkov, ktorí sú aj v iných skupinách. Je veľa možností ako tento nástroj zdokonaliť. Celkovo sme v tejto práci ukázali, že existujúce nástroje ako fail2ban a sshguard sú účinné pri obrane proti útokom hrubou silou, ale aj vlastná aplikácia môže byť efektívnym riešením. V prípade, že by sme chceli implementovať vlastné riešenie, mali by sme zvážiť viaceré faktory, ako napríklad dostupnosť a výkon nástroja, jeho jednoduchosť použitia a efektivitu v ochrane proti útokom hrubou silou.

Zoznam použitej literatúry

[1] D.J. Barrett and R.E. Silverman: *SSH, the Secure Shell, The Definitive Guide*, 2001, ISBN: 0-596-00011-1

[2]FreeBSD:FreeBSDManualPages,3.3.2023,https://man.freebsd.org/cgi/man.cgi?sshd_config(5) ,dátum prístupu: 10.4.2023

[3] Robert Watson: *SSH Hardening Tips to Prevent Brute-Force Attacks*, 14.1.2022 https://goteleport.com/blog/ssh-hardening-to-prevent-brute-force-attacks/, dátum prístupu: 10.4.2023

[4]Ledvina, J.: *SSH*, 15.5.2010. http://www.kiv.zcu.cz/~ledvina/vyuka/PDS/PDS-tut/ssh/ssh.doc , dátum prístupu: 10.4.2023

[5]Ylonen, T.; Lonvick, C.: *The Secure Shell (SSH) Authentication Protocol*. 2006.

http://www.ietf.org/rfc/rfc4252.txt , datum prístupu: 10.4.2023

[6] Xceed Software Inc, *Keyboard Interactive Authentication*, https://doc.xceed.com/xceed-filesystem-for-net/topic260.html , datum prístupu: 10.4.2023

[7] Manish Shivanandhan, *How to Use Hydra to Hack Passwords – Penetration Testing Tutorial*, 18.11.2022. https://www.freecodecamp.org/news/how-to-use-hydra-pentesting-tutorial/ , datum prístupu: 11.4.2023

[8] Ahmed Mohamed, *Password cracking using Cain & Abel*, 25.1.2018. https://resources.infosecinstitute.com/topic/password-cracking-using-cain-abel/ , dátum prístupu: 11.4.2023

[9] Palo Alto networks, *What is a Botnet? ,* https://www.paloaltonetworks.com/cyberpedia/what-is-botnet , dátum prístupu: 11.4.2023

[10] Guofei Gu, Junjie Zhang, Wenke Lee, *BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic,* https://people.engr.tamu.edu/guofei/paper/Gu_NDSS08_botSniffer.pdf , dátum prístupu: 11.4.2023

[11] Michal Zalewski, *p0f v3 (3.09b)*, https://lcamtuf.coredump.cx/p0f3/ , dátum prístupu: 11.4.2023

[12] Sandra Henry-Stocker, 3.3.2010, *Port Knocking*, https://www.networkworld.com/article/2762517/port-knocking.html , dátum prístupu: 11.4.2023

[13] Red Hat Inc., Security Guide, https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security_guide/sect-security_guide-tcp_wrappers_and_xinetd#sect-Security_Guide-TCP_Wrappers_and_xinetd-TCP_Wrappers , dátum prístupu: 12.4.2023

[14] Martin A. Brown, *Guide to IP Layer Network Administration with Linux*, http://linux-ip.net/html/linux-ip.html#ex-list-route-blackhole , dátum prístupu: 12.4.2023

[15] B. Csaba: *Vše o iptables*, http://www.root.cz/serialy/vse-o-iptables/ , dátum prístupu: 12.4.2023

[16] Ubuntu, *UFW*, https://help.ubuntu.com/community/UFW , dátum prístupu: 12.4.2023

[17] F. Maggi, *SSHGuard*, https://www.sshguard.net/ , dátum prístupu: 13.4.2023

[18] Community, *Fail2Ban*, https://www.fail2ban.org/wiki/index.php/Main_Page , dátum prístupu: 13.4.2023

[19] Sourcefire Inc., *Snort*, https://www.snort.org/ , dátum prístupu: 13.4.2023

[20] P. Schwartz, *DenyHosts*, http://denyhosts.sourceforge.net/index.html , dátum prístupu: 13.4.2023

[21] Akamai, Linode, https://www.linode.com/cloud-computing-community/ , dátum prístupu: 15.4.2023

Prílohy

Príloha A: Python skript

```
import time

import subprocess

import re

import os

import geoip2.database


# Nastavenie počtu neúspešných pokusov pred zakázaním IP
adresy

max_failed_attempts = 3


# Nastavenie časového okna (v sekundách), ktoré sa má
zohľadniť pri neúspešných pokusoch o prihlásenie

time_window = 60


# Nastavenie cesty k Apache access log file

access_log_file = '/var/log/apache2/access.log'


# Nastavenie cesty k súboru zakázaných IP adries

banned_ips_file = '/etc/banned_ips.txt'


# Nastavenie cesty k súboru databázy GeoIP2

geoip_db_file = '/usr/share/GeoIP/GeoLite2-Country.mmdb'
```

```
# Otvorenie databázy GeoIP2
geoip_reader = geoip2.database.Reader(geoip_db_file)

# Načítanie zoznamu už zakázaných IP adries zo súboru
banned_ips = []
if os.path.isfile(banned_ips_file):
    with open(banned_ips_file, 'r') as f:
        banned_ips = [line.strip() for line in f]

# Spustenie príkazu na sledovanie Apache access log file
tail_command = ['tail', '-f', access_log_file]
tail_process = subprocess.Popen(tail_command,
stdout=subprocess.PIPE, stderr=subprocess.PIPE)

# Spustenie spracovania súboru denníka riadok po riadku
for line in iter(tail_process.stdout.readline, b''):
    # Rozbor IP adresy z access log riadku
    match = re.search(r'^(\d+\.\d+\.\d+\.\d+)',
line.decode('utf-8'))
    if match:
        ip_address = match.group(1)

        # Skontroluje, či je IP adresa už zakázaná
        if ip_address in banned_ips:
```

```
            continue

        # Skontroluje, či adresa IP zlyhala viackrát ako
max_failed_attempts
        failed_attempts = 0
        for line2 in open(access_log_file):
            if ip_address in line2 and ' 401 ' in line2:
                # Získanie časovej značky riadku access
logu
                match2 = re.search(r'^\S+ \S+ \S+
\[([\w:/]+\s[+\-]\d{4})\]', line2)
                if match2:
timestamp = time.mktime(time.strptime(match2.group(1),
'%d/%b/%Y:%H:%M:%S %z'))
# Kontrola, či sa časová značka nachádza v časovom okne
if time.time() - timestamp <= time_window:
failed_attempts += 1
if failed_attempts >= max_failed_attempts:
break

# Zakázať IP adresu, ak zlyhala viac ako
max_failed_attempts krát
if failed_attempts >= max_failed_attempts:
# Získanie krajiny pôvodu pre IP adresu
try:
response = geoip_reader.country(ip_address)
```

```
country_code = response.country.iso_code
except geoip2.errors.AddressNotFoundError:
country_code = 'Unknown'

# Pridanie IP adresy do zoznamu zakázaných IP a súboru
banned_ips.append(ip_address)
with open(banned_ips_file, 'a') as f:
f.write('{} {}\n'.format(ip_address, country_code))

# Zablokovanie IP adresy pomocou iptables
subprocess.run(['iptables', '-A', 'INPUT', '-s',
ip_address, '-j', 'DROP'])

print('Banned IP address {} from {}'.format(ip_address,
country_code))
```