SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

Dynamický firewall pro MikroTik

František Kramný

Obor:	18-20-M/01 INFORMAČNÍ TECHNOLOGIE se zaměřením na počítačové sítě a programování
Třída:	IT4
Školní rok:	2023/2024

Poděkování

Děkuji Ing. Petru Grussmannovi za konzultaci, pomoc a podporu v průběhu vytváření celého maturitního projektu. Zároveň bych chtěl poděkovat svému spolužákovi Václavovi Vojtovičovi s kterým jsme úspěšně i přes veškeré komplikace projekt dokončili.

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8.

V Opavě 31. 12. 2023

podpis autora práce

ANOTACE

Tato práce se zaměřuje na implementaci dynamického firewallu pro MikroTik, vycházejícího z předchozího projektu. Cílem bylo propojit kódy pro naplnění adresáře firewallu, jeho vymazání a odstranění konkrétní IP adresy. Autor podrobně zkoumá teoretická východiska, využité technologie, a postupy implementace, včetně virtualizace RouterOS a komunikace s Turris Sentinel přes MQTT.

Využité technologie zahrnují VirtualBox, Winbox, RouterOS API a ZeroMQ. Práce zdůrazňuje důležitost dokumentace MikroTiku a struktury projektu. Práce obsahuje testovací kódy pro ověření komunikace s MikroTikem, včetně mazání adres a jejich aktualizace s využitím Turris Sentinel MQTT.

Klíčová slova = Dynamický firewall, MikroTik, implementace, virtualizace, Turris Sentinel, MQTT.

ABSTRACT

This thesis focuses on the implementation of a dynamic firewall for MikroTik, based on a previous project. The goal was to link codes to populate the firewall directory, delete it and remove a specific IP address. The author examines in detail the theoretical background, technologies used, and implementation procedures, including RouterOS virtualization and communication with Turris Sentinel via MQTT.

The technologies used include VirtualBox, Winbox, RouterOS API, and ZeroMQ. The thesis emphasizes the importance of MikroTik documentation and project structure. The thesis includes tes-test codes to verify communication with MikroTik, including address deletion and update using Turris Sentinel MQTT.

Keywords = Dynamic firewall, MikroTik, implementation, virtualization, Turris Sentinel, MQTT.

Obsah

ÚVOD

Nad výběrem tématu závěrečné práce jsem moc dlouho nepřemýšlel. V třetím ročníku jsme s Václavem měli rozpracovaný projekt dynamické firewall pro MikroTik. Protože nás oba téma zaujalo, rozhodli jsme se v něm pokračovat.

Jelikož jsme oba měli pouze obecné znalosti Pythonu, rozhodli jsme se, že tento projekt bude pro nás ideální – jak pro naučení se pokročilých funkcí Pythonu, tak pro celkovou zkušenost s tímto programovacím jazykem.

Jako cíl práce jsme si stanovili spojení našich jednotlivých oddělených kódů do jednoho celku a vytvořit za jejich pomoci funkční dynamický firewall. Měli jsme kódy pro naplnění adresáře firewallu (sentynel.py), pro celkové vymazání adresáře (delete.py) a pro odstranění konkrétní IP adresy (deleteByIp.py).

Tato práce popisuje problematiku jednotlivých kódů a MikroTiku jako celku. Následně nabízí praktická řešení jednotlivých problémů, doplněná vysvětlením a ukázkou funkce kódu.

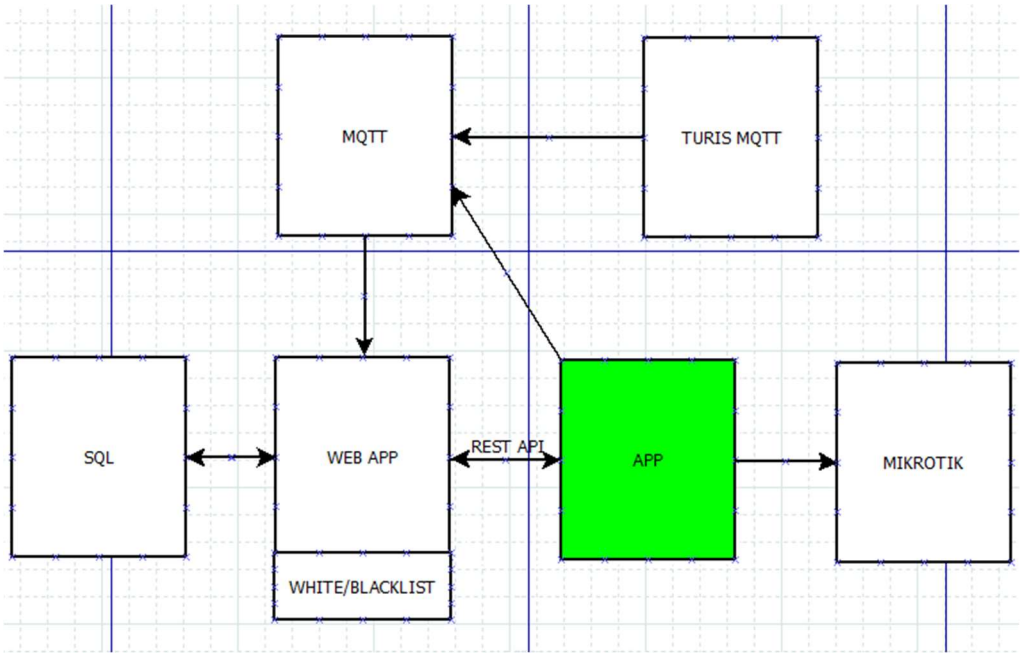
1 KONCEPCE PROJEKTU

1.1 Mikrotik RouterOS a dokumentace

Abychom mohli vůbec komunikovat s MikroTikem bylo důležité pochopit RouterOS API. Dokumentace MikroTiku se ukázala jako obsáhlá, ale za to užitečná. Díky tomu bylo jednoduché rychle pochopit RouterOS API, jak se k němu připojit a co všechno je možné konfigurovat. Dokumentace obsahuje i praktické příklady přímo pro python, které jsme několikrát využili.

1.2 Struktura projektu

Vycházeli jsme z obrázku, který jsme společně s panem učitelem Grussmannem nakreslili. Tento obrázek popisuje jednotlivé návaznosti našeho projektu a způsob jejich vzájemné komunikace.



Ukázka návaznosti jednotlivých komunikací

Tento obrázek popisuje návaznosti jednotlivých komunikací mezi námi vytvořenými kódy a struktur.

- APP(Popsaná záverečná práce) – prostřednictvím APP komunikujeme s MQTT a Turris MQTT serverem, zároveň jsme v ní skrze RouterOS API připojení na

MikroTik, což nám umožnuje zasílat jednotlivé adresy a zpětně je odebírat. Více ve funkci Handle_delta.

1.3 Turris MQTT

Potřebovali jsme někde získávat nebezpečné adresy. Pro jejich získávání jsme využili Turris Sentinel MQTT, napojený přes zmq. MQTT Turrisu jsme připojili k RouterOS API a naplnili jím firewall-addresslist. Všechny dotazy byly funkční, ale nastal problém s mazáním jednotlivých adres, protože Turris není dělaný pro MikroTik RouterOS. Kvůli tomu Turris mazal adresy z addresslistu za pomocí stringu s Ip, MikroTik tuto možnost nemá. Jediný způsob, jak v něm adresy mazat je za pomocí jejich ID. Zde jsme narazili na první problém. Který si podrobněji vysvětlíme v kapitole 3.

1.4 Virtualizace a komunikace s RouterOS

Jelikož jsme neměli k dispozici fyzický MikroTik router, bylo nezbytné stáhnout jeho ISO obraz. Pro virtualizaci jsme zvolili dobře známý VirtualBox a pomocí síťového mostu jsme se připojili k virtuálnímu routeru prostřednictvím Winboxu. Bylo klíčové také připojit náš kód, čehož jsme dosáhli díky RouterOS API, jež jsme již měli v té době studované.

2 VYUŽITÉ TECHNOLOGIE

2.1 VirtualBox

VirtualBox poskytuje prostředí pro virtualizaci, což je užitečné pro testování, vývoj a nasazování softwaru v izolovaném prostředí. V našem případě byl VirtualBox využit k vytvoření virtuálního prostředí, ve kterém běží MikroTik router a je možné se k němu připojit skrze síťový most s využitím nástroje Winbox. VirtualBox jsme vybrali, jelikož už jsme s ním měli zkušenosti z předchozích ročníků.

2.2 Winbox

Winbox je proprietární správcovský software od společnosti MikroTik, navržený pro konfiguraci a správu MikroTik routerů. S výrazným grafickým uživatelským rozhraním poskytuje uživatelům intuitivní prostředí pro nastavování síťových parametrů, firewallu, bezpečnostních opatření a dalších funkcí. Winbox zajišťuje bezpečné šifrované spojení pro ochranu citlivých dat a je klíčovým prvkem pro efektivní správu sítí s použitím produktů MikroTik.

2.3 RouterOS API

RouterOS API je rozhraní pro komunikaci s MikroTik RouterOS, což je operační systém pro směrovače a síťová zařízení od společnosti MikroTik. Toto API umožňuje automatizovanou správu a konfiguraci MikroTik zařízení pomocí programování. Používá se pro vzdálenou správu, konfiguraci firewallu, monitorování a další síťové operace.

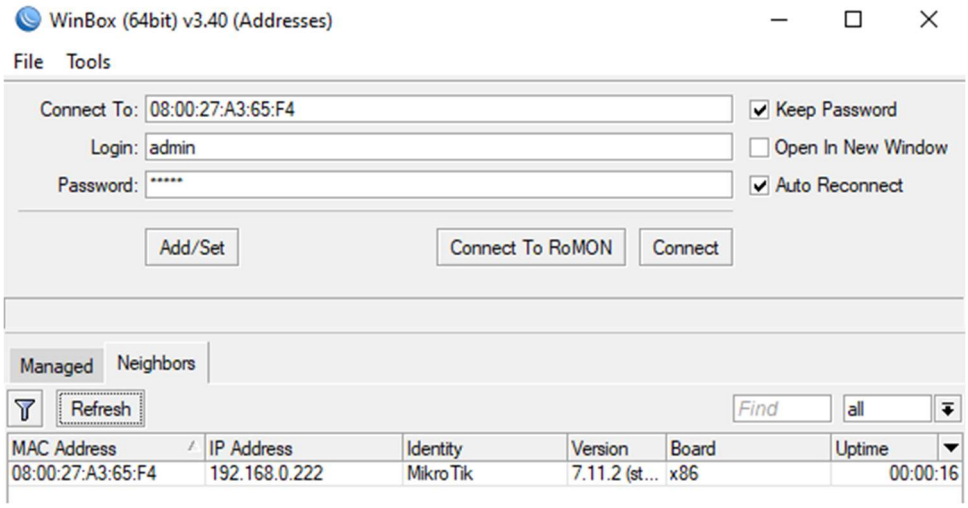
2.4 ZeroMQ

ZeroMQ je v kódu použito pro komunikaci mezi Turris Sentinel Dynamic Firewall Client a serverem. Poskytuje efektivní rozhraní pro přenos zpráv přes síť, což umožňuje aktualizace v síťových pravidlech na základě příchozích informací od serveru.

3 ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY

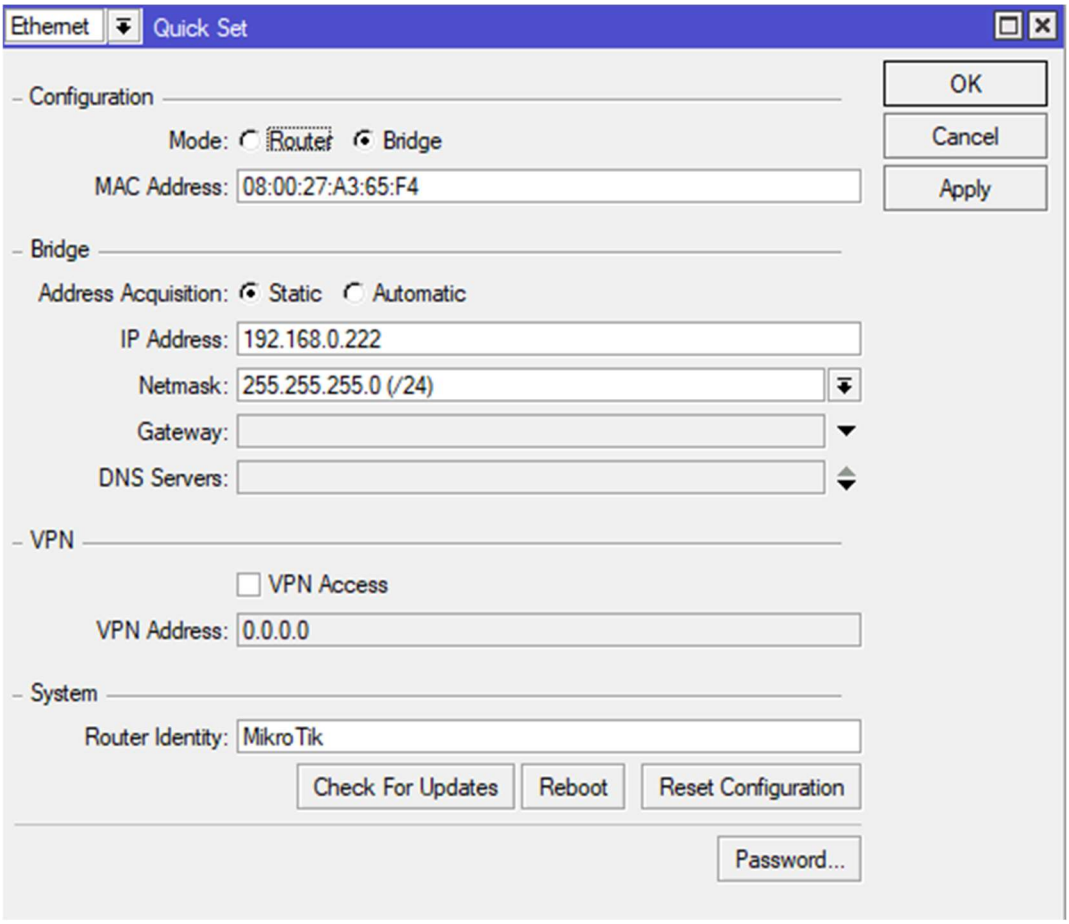
3.1 Virtualizace RouterOS

Před začátkem programování bylo důležité získat ISO od MikroTiku a následně si ho virtualizovat. Naštěstí je ISO RouterOS k dispozici zdarma na webových stránkách MikroTiku. ISO jsme si stáhli a následně jej importovali do VirtualBoxu. Poté jsme jednoduše nastavili síťový most, přiřadili IP adresu MikroTiku a připojili se pomocí Winboxu.



Ukázka připojení na MikroTik pomocí MAC adresy

Při instalaci MikroTiku dbejte na několik zásad. Jako první si uvědomte, že ISO se stahuje na VDI v podobě CD. Při nastavování bootovacího zařízení je tedy důležité zvolit ISO místo prázdného harddisku. Po instalaci je také klíčové nastavit síťový most, což lze udělat v uživatelském rozhraní VirtualBoxu pod kartou síť. Dalším krokem je přiřazení IP adresy MikroTiku, což není úplně jednoduché. Z nějakého důvodu nelze funkčně přidat adresu v rozhraní MikroTiku spolu s networkem. Jediná funkční metoda, kterou jsme objevili, byla pomocí WinBoxu. Winbox má tu výhodu, že se dá připojit pomocí MAC adresy. Poté už stačí kliknout na záložku Quick Set a přidat adresu. Důležité je přiřadit adresu z rozsahu sítě, kterou máte na počítači. Tuto informaci zjistíte ve Windows příkazovém řádku pomcí příkazu ipconfig, v Linuxu pak příkazem ifconfig.



Ukázka funkce Quick Set ve Winboxu

3.2 Počáteční psaní kódů

Potřebovali jsme otestovat, zda se můžeme připojit k MikroTiku skrze API a provádět zápisy do firewall-addresslistu. K tomu jsme napsali první testovací kódy. Protože jsme nevěděli, zda to bude hardwarově náročné, rozhodli jsme se provést řadu testů. V posledních 3 řádcích ukázky kódu jdou vidět tyto body.

```
# -*- coding: utf-8 -*-
#!/usr/bin/python

import routeros_api
from random import randint


connection = routeros_api.RouterOsApiPool('10.57.10.111', username='admin',
password='admin', port=8728, plaintext_login=True )
api = connection.get_api()
# vypsani aktualniho adresslistu
list_queues = api.get_resource('/ip/firewall/address-list')
# pridani adresy
for i in range(1,10001):
    def generate_random_ip():
        return '.'.join(
            str(randint(0, 255)) for _ in range(4)
        )
    random_ip = generate_random_ip()
    list_queues.add(address=random_ip, comment='TEST',  list='sentinel')
    print(id)

connection.disconnect()

# otestovat rychlost zápisu cca 10K ip adress
# mazání přes ID - otestovat 10K
# Otestovat rychlost výpisu 10k záznamů
```

3.3 Delete.py

Po úspěšném připojení k MikroTiku a otestování zápisu 10 000 IP adres jsme zjistili, že jsme hardwarově připraveni. Měli jsme plný seznam adres ve firewallu a potřebovali jsme ho vyprázdnit. Z dokumentace MikroTiku jsme věděli, že nelze adresy odstranit prostřednictvím samotného řetězce IP adresy, ale je nutné použít ID adresy. V další ukázce

kódu jde vidět, jak hledání a mazání adres funguje.

```
import routeros_api
ip4='192.168.0.222'
connection = routeros_api.RouterOsApiPool(ip4, username='admin',
password='admin', port=8728, plaintext_login=True )
api = connection.get_api()
list_queues = api.get_resource('/ip/firewall/address-list')

pole=list_queues.get()
for prvek in pole:
    print(prvek)
    list_queues.remove(id=prvek['id'])

    print(id)
connection.disconnect()
```

3.4 DeleteByIp.py

Když už jsme měli hotové naplnění a mazání, potřebovali jsme mazat i konkrétní adresy kvůli tomu jsme napsali funkci DeleteByIp.py.

```
# -*- coding: utf-8 -*-
#!/usr/bin/python
import routeros_api

connection = routeros_api.RouterOsApiPool('192.168.0.222',
username='admin', password='admin', port=8728, plaintext_login=True )
api = connection.get_api()
# vypsani aktualniho adresslistu
list_queues = api.get_resource('/ip/firewall/address-list')
print("zadej ip: ")
ipToDelete = input()
i = 0
pole=list_queues.get()
for prvek in pole:
    if prvek['address'] == ipToDelete:
        break
    i+= 1
print(list_queues.get()[i]['id'])
print(list_queues.get()[i]['address'])
idToDelete = list_queues.get()[i]['id']
list_queues.remove(id=idToDelete)
connection.disconnect()
```

3.5 Napojení na Turris Sentinel

Pro připojení k MQTT Sentinelu jsme využili kód, který nám poskytla oficiální stránka Turris Sentinelu, Kód obsahoval serverový certifikát, zmq klient pro MQTT, základní parsování a class IpSet. Nějakou dobu nám trvalo, než jsme se v kódu zorientovali. Zjistili, co chybí, jaké knihovny využívá a jak změnit jeho fungování. Kód, již řešil přidávání adres, ale neřešil update funkci a delete funkci. Ty bylo třeba přidat a upravit návaznosti, např. aby se adresy nepřidávaly v loopu stále do kola. Ale právě díky němu byla půlka práce za námi.

3.5.1 Handle_list

Funkce, která načítá adresy do firewall-addresslistu se schovává pod názvem Handle_list. Ta ovšem nebyla úplně dokonalá, jak již jsem zmiňoval nefungoval delete, čili ikdyž se adresy načítaly poměrně rychle, přicházely z MQTT dotazy na mazání z aktuálního listu. To mělo za výsledek, nesprávný počet škodlivých adres a celkovou asynchronnost s Turris Sentinelem. Po delším uvažování jsme zjistili, že chyba není v samotném kódu, ale z důvodu absence funkce delete ve funkci Handle_delta. Jedinou změnu, kterou jsme v tomto kódu provedli byla implementace výše zmiňovaného kódu delete.py, který zajišťoval, že po každém restartu programu budou vymazány všechny adresy z firewall-addresslistu.

3.5.2 Handle_delta

Funkce je updatovací, to znamená, že po nahrání adres z Handle_list, přicházejí dodatečné zprávy ze serveru, které uvádějí, jestli jsou to adresy na přidání nebo na smazání. Opět bylo přidávání funkční, ale mazání nebylo vyřešené vůbec. Na tento problém jsme měli opět předem nachystaný kód DeleteByIp, který jsme implementovali a tím vyřešili mazání. Kód fungoval, ale ukázalo se, že je příliš pomalí a kvůli tomu se stávalo, že přestal fungovat. Důvodem bylo, že když přišla 1 adresa na smazání během 20 sekund, vše fungovalo, ale jakmile jich přišlo např. 6 kód se zastavil a přestal fungovat. Pro urychlení kódu jsme využili ChatGPT, který funkci upravil na novou podobu.

```
def remove_ips(self, ips_to_remove):
    list_resource = self.api.get_resource('/ip/firewall/address-list')
    ip_to_delete = ips_to_remove
    address_list = list_resource.get()
    index_to_delete = next((i for i, item in enumerate(address_list) if
item['address'] == ip_to_delete), None)
    if index_to_delete is not None:
        id_to_delete = address_list[index_to_delete]['id']
        list_resource.remove(id=id_to_delete)
        print(f"Removed IP {ip_to_delete} from the address list")
    else:
        logger.warning("IP address not found in the list: %s",
ip_to_delete)
```

3.6 Úpravy kódu

Další úpravy kódu, zahrnovaly mazání Serial loopu a dalších podobných věcí, které znemožňovali správnou funkci kódu. Bylo to více méně další hlubší pochopení classy Serial, která běžela v loopu a po 100 zaslání dotazu MQTT došlo ke špatnému resetu počítání MQTT dotazu.

4 VÝSLEDKY ŘEŠENÍ, VÝSTUPY, UŽIVATELSKÝ MANUÁL

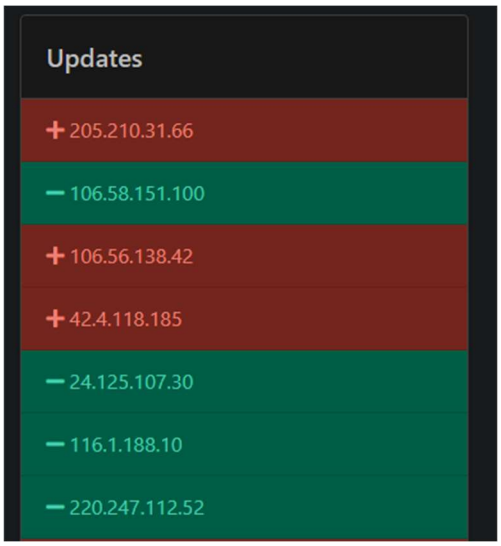
4.1 Výsledek řešení

Výsledkem řešení, je plně funkční kód, který spravuje dynamickou firewall pro MikroTik Router. Stačí zadat IP4 adresu, password a username MikroTiku, které jsou uložené v proměnných.

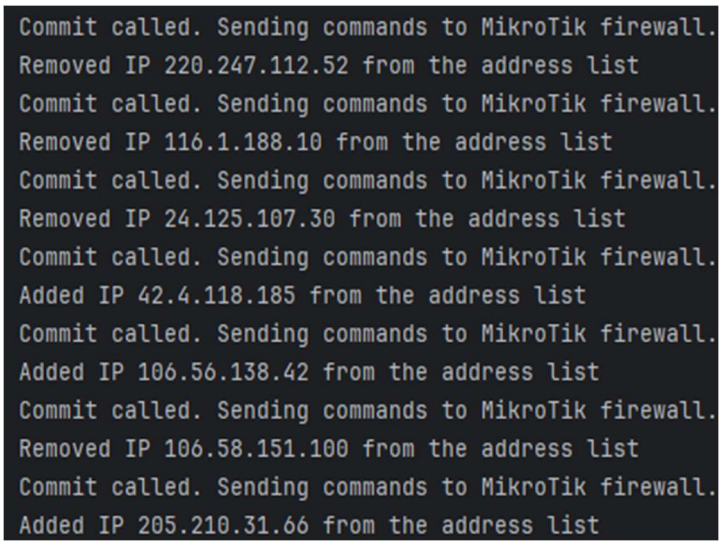
```
#nastavte IP MikroTiku
ip4 = '192.168.0.222'
#nastavte username MikroTiku
username = 'admin'
#nastavte heslo MikroTiku
password = 'admin'
#nastavte počet přijmutých pokynu MQTT, před reset routeru
MISSING_UPDATE_CNT_LIMIT = 10000
```

Ověření funkčnosti.

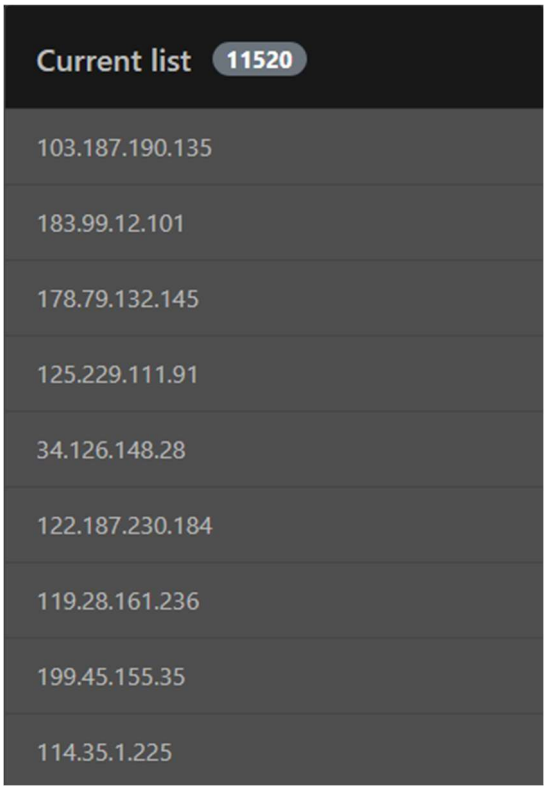
Porovnání Turris Sentinelu s naším výstupem kódu v cmd.



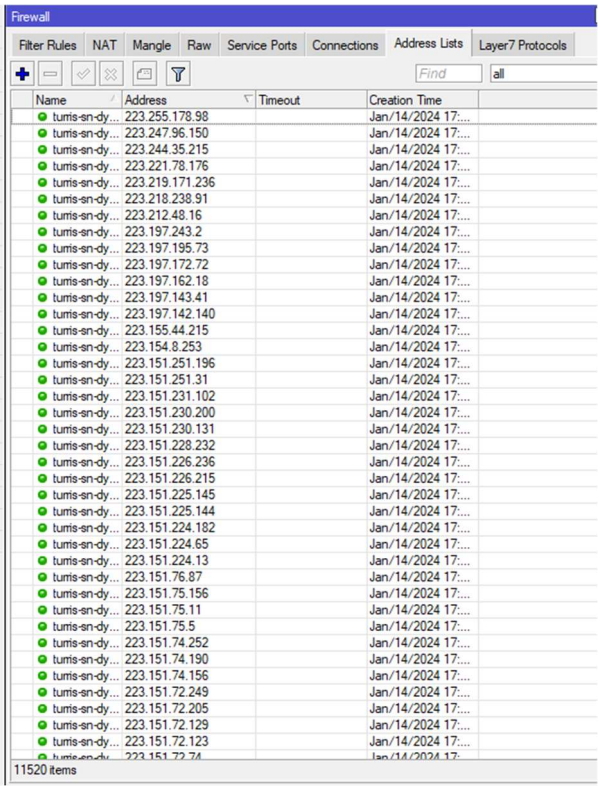
Real-time Handle_delta



Real-time Handle_delta posílá update MikroTiku



Real-time Handle_list s 11520 adresy



Real-time firewall-addresslist se stejnou hodnotou

4.2 Splněné cíle

Cílem bylo úspěšné zrealizování dynamické firewally, která by fungovala na principu Turris Sentinel pro MikroTik router. Dalším cílem bylo využít naše předchozí kódy a implementovat je tak, aby vše fungovalo. Všechny výše zmíněné cíle se nám podařilo zrealizovat a vytvořit tak funkční dynamickou firewall pro MikroTik router.

4.3 Plány do budoucna

Jelikož pro nás tato problematika nekončí, stanovili jsme si určité cíle do budoucna. Jelikož náš kód není úplně přehledný a se správnou návazností, chtěli bychom jej posunout o krůček víš, zbavit se přebytečných funkcí, které kód stále obsahuje. Dalším krokem je propojit náš program s flask aplikací na kterém momentálně, již Vašek pracuje a která přinese uživatelské rozhraní uživatelům.

ZÁVĚR

Celkově lze shrnout, že implementace dynamického firewallu pro MikroTik Router byla úspěšná, splňující stanovené cíle projektu. Vytvořený kód poskytuje efektivní nástroj pro automatickou aktualizaci síťových pravidel na základě informací od Turris Sentinel. Možné uplatnění řešení v praxi spočívá v posílení bezpečnosti sítí blokováním nebezpečných IP adres.

Další možnosti vylepšení zahrnují optimalizaci kódu, odstranění přebytečných funkcí a implementaci uživatelského rozhraní. Tyto kroky by přispěly k lepší efektivitě a uživatelské přívětivosti. Projekt představuje solidní základ pro rozvoj v oblasti kybernetické bezpečnosti s možností dalšího rozšíření o nové bezpečnostní prvky či integraci s dalšími síťovými zařízeními.

https://github.com/frantisek-218/MIKROTIK

SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

[1] Gitlab. Turris-Sentinel-dynfw. [Online]. Dostupné z https://gitlab.nic.cz/turris/sentinel/dynfw-client

[2] RouterOS Documentation. [Online]. Dostupné z https://help.mikrotik.com/docs/display/ROS/API

[3] Python documentation. [Online]. Dostupné z https://docs.python.org/3/

[4] ZeroMQ Python. [Online]. Dostupné z https://zeromq.org/get-started/?language=python#

[5] How to install MikroTik OS. [Online]. Dostupné https://www.youtube.com/watch?v=7kfusph7Unw&t=174s&ab_channel=TutorialSchools