SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE
dokumentace

Dynamický firewall

Václav Vojtovič

Obor: 18-20-M/01 INFORMAČNÍ TECHNOLOGIE
se zaměřením na počítačové sítě a programování

Třída: IT4

Školní rok: 2023/2024

Poděkování

Děkuji panu ing. Petrovi Grussmannovi za cenné rady během naší spolupráce na projektu a Františkovi Kramnému za jeho výjimečnou spolupráci. Vaše odbornost a ochota sdílet znalosti byly neocenitelné, a díky vašim přínosům jsem dosáhl úspěšného dokončení projektu.

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. 2016

podpis autora práce

ANOTACE

Tento projekt využívá framework Flask v Pythonu k vytvoření dynamického firewallu umožňujícího uživatelům definovat a upravovat pravidla pro filtrování síťového provozu. Aplikace poskytuje webové rozhraní pro pohodlné spravování a monitorování nastavení firewallu. Uživatelé mohou dynamicky měnit pravidla během provozu, což zajišťuje flexibilitu a rychlou reakci na změny v potřebách sítě. Bezpečnost je zajištěna ověřováním identit a šifrovanou komunikací. Projekt nabízí snadné a efektivní řízení bezpečnosti sítě.

Klíčová slova

Bezpečnost; Flask; Python; webová aplikace; Api

OBSAH

ÚVOD

Moderní počítačové sítě čelí stále rostoucím výzvám v oblasti kybernetické bezpečnosti, a dynamický přístup k řízení přístupu a filtrování síťového provozu se stává klíčovým prvkem v ochraně informačních systémů. Tato závěrečná práce se zaměřuje na implementaci dynamického firewallu, který využívá framework Flask v jazyce Python a integruje API pro efektivní správu síťových pravidel. Cílem je prezentovat inovativní přístup k řízení bezpečnosti sítě prostřednictvím webového rozhraní, které umožňuje uživatelům definovat pravidla. Využití API přináší další vrstvu flexibility a interoperability, umožňující integrovat tento dynamický firewall do komplexních informačních systémů. . Tímto příspěvkem se snažím posunout diskuzi o moderních přístupech k kybernetické bezpečnosti a přispět k rozvoji efektivních nástrojů pro ochranu informačních systémů v dnešní dynamické kybernetické krajině.

1 TEORETICKÁ A METODICKÁ VÝCHODISKA

1.1 Flask App Builder

Flask App Builder (FAB) představuje vývojový framework pro webové aplikace postavený na mikroframeworku Flask v programovacím jazyce Python. Jeho hlavním cílem je zjednodušit proces vytváření webových aplikací prostřednictvím jednoduchého rozhraní a předem definovaných funkcionalit, čímž umožňuje vývojářům soustředit se na implementaci jádra aplikace a minimalizovat potřebu ručního kódování opakujících se úkolů spojených s vytvářením webových rozhraní.

Tento framework vyniká několika klíčovými vlastnostmi. Zaprvé, nabízí rychlý vývoj díky hotovým modulům pro časté úkoly, což zkracuje dobu potřebnou pro vytvoření plnohodnotné webové aplikace. Dále, modularita aplikací postavených na FAB usnadňuje integraci nových funkcí a rozšíření stávajících. Webové rozhraní FAB umožňuje pohodlnou administraci a konfiguraci aplikací přímo z prohlížeče, což usnadňuje správu bez nutnosti přímých zásahů do kódu.

Důraz je kladen i na databázovou integraci, kde FAB umožňuje snadnou propojitelnost s různými databázovými systémy, usnadňující práci s daty a jejich řízení. Systém přístupových práv a rolí je implementován pro bezpečné zabezpečení aplikací postavených na FAB. Kromě toho, podpora pro REST API usnadňuje vytváření RESTful API, což je výhodné pro komunikaci s dalšími aplikacemi a službami.

Celkově je Flask App Builder vnímán jako efektivní nástroj pro vývoj webových aplikací v prostředí Flasku, přinášející užitečné funkce pro správu a konfiguraci aplikací, což zvyšuje produktivitu vývojářů a umožňuje rychlé nasazení moderních webových řešení.

1.2 Instalace

Vytváření webových aplikací s pomocí Flask-AppBuilderu přináší jednoduchost a efektivitu do procesu vývoje. Prvním krokem je vytvoření izolovaného virtuálního prostředí, které odstíní náš projekt od ostatních Python aplikací. Následně instalujeme Flask-AppBuilder jednoduše pomocí příkazu `pip install Flask-AppBuilder`. Po vytvoření projektu a přechodu do jeho adresáře můžeme okamžitě spustit vývojový server příkazem `fabmanager run`.

Tímto krokem aktivujeme server a získáme přístup k naší aplikaci přímo v prohlížeči. Jedním z výrazných benefitů je schopnost Flask-AppBuilderu automaticky generovat uživatelské rozhraní pro CRUD operace. Tento přístup umožňuje rychlý vývoj bez zdlouhavého manuálního psaní kódu.

Díky integraci s frameworkem Flask získáváme nejen jednoduchost, ale také výhody tohoto oblíbeného nástroje. Výsledkem je webová aplikace, která je snadno spravovatelná, disponuje intuitivním rozhraním a je připravena na další rozšíření a přizpůsobení. Celý proces vytváření webových aplikací s Flask-AppBuilderem se tak stává nejen příjemným, ale také efektivním pro vývojáře.

2 VYUŽITÉ TECHNOLOGIE

2.1 GitHub

GitHub je platforma pro správu zdrojového kódu a spolupráci programátorů. Používá Git k sledování verzí kódu a umožňuje týmům sledovat úkoly. Usnadňuje sdílení kódu, podporuje různé jazyky a integrace a podporuje spolupráci na otevřených projektech. Transparentnost vývoje, sledování problémů a nástroje pro automatizaci testování jsou klíčové funkce. GitHub je nezbytný pro moderní softwarový vývoj.

2.2 Docker

Docker je open-source platforma pro vývoj a nasazování aplikací v kontejnerech. Kontejnery jsou lehké, přenosné balíčky, které obsahují veškerý potřebný software pro běh aplikace. S Dockerem můžete snadno vytvářet, sdílet a spravovat kontejnery pomocí jednoduchých příkazů a konfiguračních souborů. Tato technologie umožňuje izolaci aplikací, což zajišťuje konzistentní chování na různých systémech. Docker také nabízí automatizaci procesu nasazování a škálování aplikací, což vede k efektivnějšímu vývoji a správě softwarových projektů.

2.3 SQLALchemy

SQLAlchemy je Python knihovna pro práci s relačními databázemi, poskytující API pro různé databázové systémy. Hlavním prvkem je objektově-relační mapování (ORM), které umožňuje reprezentovat databázové záznamy jako objekty v Pythonu. SQLAlchemy zjednodušuje manipulaci s daty pomocí "table" a "mapper". Definice databázové struktury v Pythonu umožňuje snadné vytváření, čtení, aktualizaci a mazání dat. Knihovna abstrahuje od specifických SQL dialektů, což usnadňuje přenositelnost kódu mezi různými databázovými systémy. SQLAlchemy umožňuje pracovat s databází bez ručního psaní SQL dotazů a podporuje transakce, což přináší robustnost a odolnost proti chybám. Je to efektivní nástroj pro správu relačních databází v Pythonu.

2.4 Flask

Flask je mikroframework pro vývoj webových aplikací v jazyce Python, který se vyznačuje jednoduchostí a flexibilitou. Přestože je lehký, nabízí dostatek nástrojů pro efektivní vytváření webových aplikací bez nadbytečné komplexity. Hlavní myšlenkou Flasku je poskytovat základní strukturu a nástroje pro vývojáře, ale zároveň umožňovat svobodnou konfiguraci a rozšíření podle potřeby.

V rámci Flasku jsou aplikace postaveny na základě tzv. "route" (cest), což jsou URL adresy, které odpovídají konkrétním funkcím nebo metodám v kódu. Tímto způsobem vývojáři definují, jak mají být zpracovány požadavky klientů na konkrétních cestách. Flask podporuje templating pro vytváření dynamických HTML stránek a integraci s databázemi pro práci s daty.

Důležitým prvkem je také modulární rozšiřitelnost Flasku, což umožňuje integrovat dodatečné balíčky a nástroje podle potřeby projektu. Celkově je Flask vhodný pro vývoj menších a středně velkých webových aplikací, kde je kladen důraz na jednoduchost a rychlost vývoje.

2.5 Flask-Migrate

Flask-Migrate, rozšíření pro Flask v Pythonu, zjednodušuje správu databázových migrací. Využívá koncept migrací, což jsou soubory popisující změny v databázovém schématu. Integrace s SQLAlchemy umožňuje definovat změny v Pythonovském kódu. Příkazový řádek umožňuje snadnou tvorbu a správu migrací, eliminující potřebu ručních SQL dotazů. Flask-Migrate generuje SQL příkazy automaticky, což usnadňuje konzistenci a přenositelnost kódu. Podpora transakcí zajišťuje atomické provedení změn v databázi a v případě chyb vrácení do původního stavu. Celkově je to nástroj pro systematickou správu databázových migrací v aplikacích postavených na Flask.

3 ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY

3.1 Problémy

Řešení, ačkoliv se zdálo, že bude jednoduché, se nakonec ukázalo jako poměrně složité. Hlavním důvodem komplikací byla dokumentace Flask-AppBuilderu, která je ve své podstatě úplně k ničemu a v ničem vám nepomůže. To znamená, že na většinu věcí musíte přijít sami nebo si brát velkou inspiraci z jiných projektů. Toto také dost protáhlo vývoj.

3.2 Modely

Model databáze byl poměrně jednoduchý. Jak má databáze vypadat jsem měl předem vymyšlené. Nejsložitější tabulka je tabulka Machine, která se skládá z id, name, comment a je ve vazbách s tabulkami Blacklist a Whitelist. Abychom mohly k jednomu Machinu přiřadit více adres s black listu a white listu musíme mezi nimi udělat vazbu many to many s čím nám pomůže tabulka machine_whitelist_association.

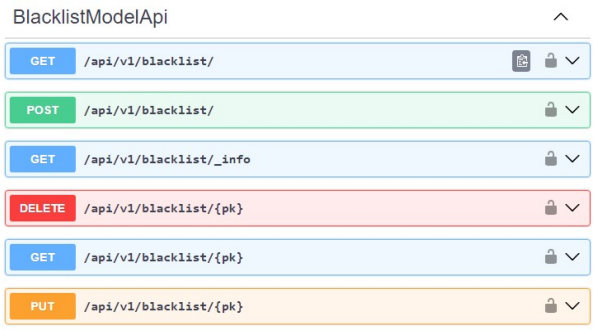
```
machine_whitelist_association = Table(
    'machine_whitelist_association',
    Model.metadata,
    Column('machine_id', Integer, ForeignKey('machine.id')),
    Column('whitelist_id', Integer, ForeignKey('whitelist.id'))
)
class Blacklist(Model):
    id = Column(Integer, primary_key=True)
    ip = Column(IPAddressType, nullable=False, unique=True)

    def __repr__(self):
        return str(self.ip)
```

3.3 Api

Api už byla větší oříšek, dokumentace pro ni je neexistující což znamená že metoda pokus omyl byla jediná, kterou jsem měl. Nakonec jsem si s ní ale poradil. První si vytvoříme class s parametrem “ModelRestApi” do classy pote přidáme resource_name a datamodel který nám říká se kterou tabulkou chceme komunikovat nakonec přidáme ještě filtr. Tohle uděláme pro všechny tabulky v tatabazi. Tyhle api modely ještě vyvoláme. Na stránce http://127.0.0.1:5000/swagger/v1 si potom můžeme všechno prohlédnout, a dokonce i vyzkoušet.

```
class MachineModelApi(ModelRestApi):
    resource_name = "machine"
    datamodel = SQLAInterface(Machine)
    allow_browser_login = True
    search_filters = {"name": [CustomFilter]}
```



Ukázka z API 1

3.4 Views

Views slouží jako deklarace tabulek které chceme zobrazovat, data také můžeme mazat a upravovat. Tato funkce se obzvlášť hodí v praxi při provozu. Pro vytvoření pohledu si uděláme class s parametrem “ModelView” do classu pote dopíšeme tabulku kterou chceme zobrazovat řádky které chceme zobrazovat a označení řádků.

```
class MachineModelView(ModelView):
    datamodel = SQLAInterface(Machine)
    label_columns = {'Name':'Name'}
    list_columns = ['name','comment','machines_ip']
```

V neposlední řadě si řadě si přidáme samotný pohled pomoci add_view.

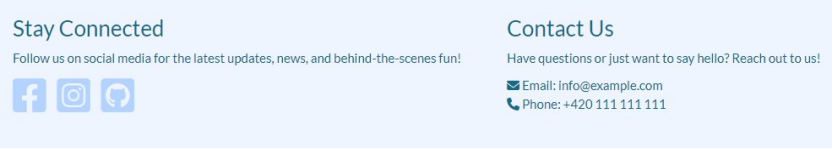
```
appbuilder.add_view(
    WhiteListModelView, "White list", icon="fa-folder-open-o", category="Lists"
)
```

3.5 Úprava vzhledu aplikace

3.5.1 Footer

Patička má dvě části: jednu zaměřenou na udržení spojení s uživateli a druhou na kontaktování. První část obsahuje nadpis "Stay Connected" s popisem a odkazy na sociální média (Facebook, Instagram, GitHub). Druhá část obsahuje nadpis "Contact Us" s informacemi o možnosti kontaktování, včetně e-mailu a telefonního čísla.

Dále jsou v kódu zahrnuty i skripty pro načítání knihoven, jako je jQuery a Bootstrap, a také skripty pro načítání ikon ze sady Font Awesome.



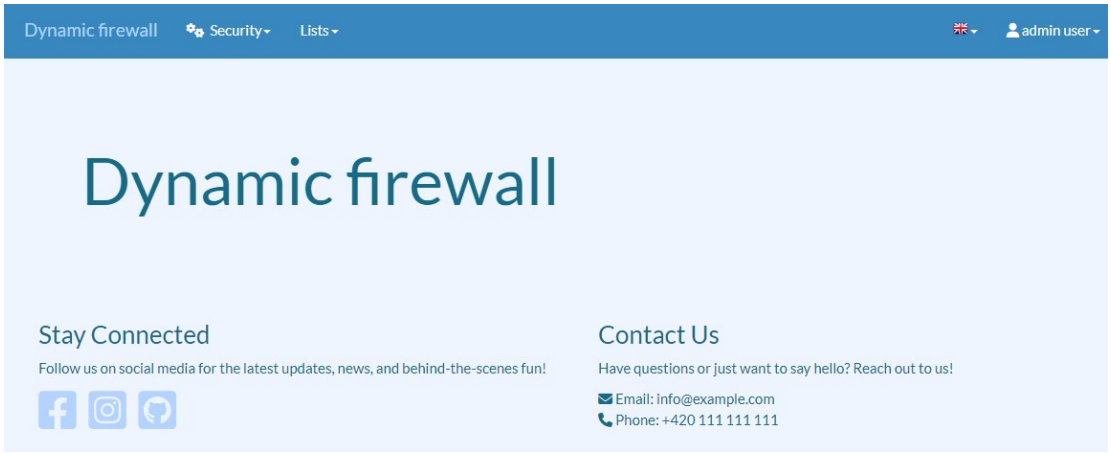
Footer

3.5.2 Hlavní strana

Celkový design této webové stránky zdůrazňuje minimalistický a moderní přístup s použitím odstínů modré a šedé barvy. Hlavní pozadí hlavní stránky je světle modré, což vytváří svěží a příjemnou atmosféru pro návštěvníky. Text na stránce je kontrastně zobrazen tmavě modrým odstínem, což zajistí snadnou čitelnost a vizuální jasnost.

Jumbotron, velký vizuální prvek na stránce, má světle šedé pozadí, což dodává subtilní odlišnost a zvýrazňuje obsah v tomto bloku. Navigační panel (navbar) je definován temně modrým pozadím, čímž vytváří jasný kontrast k hlavnímu obsahu stránky.

Interaktivní prvky, jako jsou tlačítka v navigačním panelu, získávají světle modré podbarvení při najetí myši, což přidává dojem interaktivity. Celkově je design stránky koncipován tak, aby byl moderní, přívětivý a snadno navigovatelný. Použití ikon ve stylu Font Awesome přispívá k modernímu vzhledu a zvyšuje vizuální apel stránky. Odkazy jsou zpracovány v modrém odstínu, což přispívá k jednotnému designu a intuitivní navigaci pro uživatele.



Front page

```
body {
        background-color: #EEF5FF !important;
        font-family: "Lato", "Helvetica Neue", Helvetica, Arial, sans-serif;
        font-size: 15px;
        line-height: 1.42857143;
        color: #176B87;
    }
.jumbotron {
        background-color: #ecf0f100;
    }
```

3.6 Přihlášení

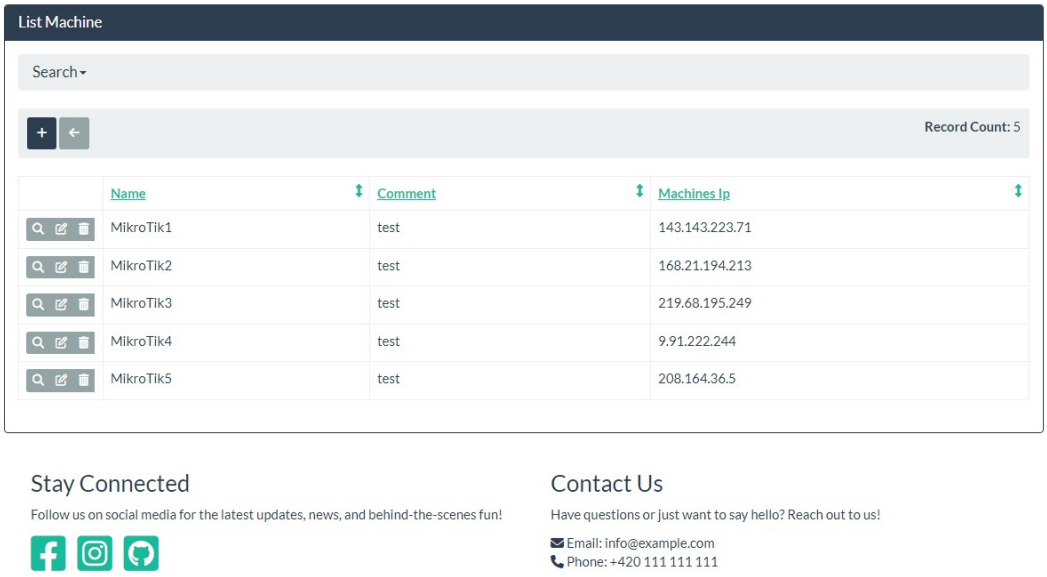
Na přihlášení používám již implementované přihlašovaní ve Flask App Builderu. K prvotnímu přihlášení zadáme příkaz ” flask fab create-admin” kde si vytvoříme admina kterým se přihlásíme do flasku. Poté si můžeme pomocí grafického rozhraní přidat další uživatele.

Sign In tabulka

4 VÝSLEDKY ŘEŠENÍ

4.1 Finální aplikace

Cílem projektu bylo vytvořit funkční webovou aplikaci a api pro záznam ip adres. Během vývoje jsem se přiučil o různých technologiích a naučil jsem se řešit problémy nebo je, popřípadě obejít. Aplikace funguje tak jak jsem si na začátku vývoje představoval. Je zde ale stále prostor na zlepšení.



Machine list view

ZÁVĚR

Úspěšně jsem dosáhl svého cíle vytvořit tuto aplikaci, která v současné době slouží jako funkční základ pro naše budoucí plány s projektem. Přestože projekt splnil svůj primární účel, stále vidím prostor pro vylepšení, které by posunulo aplikaci k praktičtějšímu využití. Jednou z klíčových oblastí pro budoucí zlepšení je bezpečnost aplikace. Svůj projekt hodnotím pozitivně. Získal jsem cenné zkušenosti a dovednosti, které budou prospěšné v mé budoucí kariéře v oblasti vývoje webových aplikací. Projekt je pro mě nejen úspěšným zakončením ročníkového úkolu, ale i přínosem pro můj osobní a profesní růst.

SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

[1] GitHub. Flask-AppBuilder. [Online]. Dostupné z: https://github.com/dpgaspar/Flask-AppBuilder

[2] Flask-AppBuilder Documentation. [Online]. Dostupné z: https://flask-appbuilder.readthedocs.io/en/latest/index.html.

[3] Introduction to Flask-AppBuilder: Building a Simple Web Service. [Online]. Dostupné z: https://python.plainenglish.io/introduction-to-flask-appbuilder-buildinga-simple-web-service-16ad26876ef6

[4] Docker. [Online]. Dostupné z: https://www.docker.com/

[5] Flask-Migrate na PyPI. [Online]. Dostupné z: https://pypi.org/project/Flask-Migrate/