SŠPU STŘEDNÍ ŠKOLA PRŮMYSLOVÁ A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

CTF systém v Kubernetes

Autor: Jan Stránský

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

Třída: IT4

Školní rok: 2024/25

Poděkování

Rád bych poděkoval pánům učitelům Ing. Petru Grussmannovi a Mgr. Marku Lučnému za jejich pomoc s projektem, jelikož mi poskytovali cenné rady a připomínky.

Prohlášení

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

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

V Opavě 6. 1. 2025

. . . . . .

Podpis autora

Abstrakt

Výsledkem tohoto projektu je funkční systém pro spouštění a vytváření úloh CTF typu v systému Kubernetes běžícím na školní síti s dostatečnou mírou zabezpečení. Aplikace zahrnuje registraci a přihlašování uživatelů, zapínání nových úloh a následně jejich vypínání. Hlavní částí tohoto projektu je komunikace se systémem Kubernetes, který se využívá ve vysoce škálovaných produkčních prostředích. Uživatel s aplikací může komunikovat skrz poskytnuté webové prostředí, ale může komunikovat i přímo s poskytnutou API. Dále si tento projekt klade za cíl umožnit studentům se lépe seznámit s určitými možnostmi v oblasti IT formou hry (CTF) jako to dělají služby jako např. TryHackMe nebo HackTheBox.

Klíčová slova

CTF, Kubernetes, FastAPI, webová aplikace

Abstract

The result of this project is a functional system for running and creating CTF type tasks in the Kubernetes system running on the school network with a sufficient level of security. The application includes user registration and login, turning on new tasks and then turning them off. The main part of this project is communication with the Kubernetes system, which is used in highly scalable production environments. The user can communicate with the application through the provided web interface, but can also communicate directly with the provided API. Furthermore, this project aims to allow students to better familiarize themselves with certain IT options through a game (CTF) like services TryHackMe or HackTheBox do.

Keywords

CTF, Kubernetes, FastAPI, web application

Obsah

Úvod

Mým cílem v této práci bylo sestavit škálovatelný software, který by nad prostředím Kubernetes vytvářet a spravovat kontejnery pro soutěž typu CTF (Capture The Flag). Zároveň bylo cílem, aby se tento software dal nasadit i v prostředí s nízkým oprávněním a aby ho šlo škálovat díky architektuře mikroslužeb.

Hlavní motivací bylo pochopení funkce a komunikace v rámci aplikací s formátem typu mikroslužeb místo monolitických aplikací a zlepšení svých dovedností v oblasti prostředí Kubernetes.

Zvláštní zaměření bylo na backendovou část API a na zabezpečení celého systému.

1 Problematika efektivního hostovaní CTF systému

CTF (Capture The Flag) je forma soutěže a interaktivního vzdělávání, kde se účastníci snaží získat co nejvíce vlajek (flag) z různých úloh. Obvykle se jedná o úlohy z oblasti kybernetické bezpečnosti, ale mohou být i z jiných oblastí jako např. programování, matematika, atd. Já jsem se ve svém projektu rozhodl zkusit vytvořit funkční CTF systém spustitelný na školní instanci Kubernetes.

Obvykle se při tvorbě takových systémů brzy narazí na problémy s bezpečností samotného systému vzhledem k tomu, že se jedná o systém, který musí mít alespoň omezený přístup k jinému systému kde může spouštět kontejnery, tím pádem musí zajistit, že tento systém nelze v případě úspěšného útoku na administrátorský účet zneužít na eskalaci práv na samotný hostitelský cluster. Toto je obvykle zajištěno využitím již existujících cloudových hosting platforem (např. Amazon AWS EC2), ale v tomto projektu jsem se rozhodl pokusit o řešení tzv. on-prem nasazení, kde tento kód využívá čistě již vytvořených zdrojů bez automatického provizování nových od cloudového poskytovatele, což znatelně zvyšuje komplexitu celého systému, ale zároveň snižuje riziko nečekaných nákladů kvůli opomenutým cloudovým strojům, které jsou obvykle placeny nejen za minutu provozu, ale i za přenos dat, což se může u některých typů úloh stát velmi drahým kvůli množství paketů, které se musí přenést.

Avšak, on-prem řešení má problémy s určitými požadavky - např. může být problém zajistit přístup ke kontejnerům, který je jinak obvykle řešen buď přímým přístupem přes přiřazenou IP adresu, VPN nebo přes službou poskytnutý "Attack Box,"což je samostatný kontejner s nástroji pro účastníky, který je připojen k interní síti a může být použit pro útoky na úlohy. Přístup k tomuto boxu je obvykle řešen přes webový prohlížeč. Dalším problémem je zabezpečení samotného systému, kde je nutné zajistit, aby účastníci nemohli získat přístup k jiným účastníkům nebo k samotnému hostitelskému systému.

1.1 Využití v praxi

Nejčastěji se CTF systémy využívají pro vzdělávání a trénování v oblasti kybernetické bezpečnosti, kde se účastníci snaží získat co nejvíce vlajek (flag) z různých úloh. Toto je obvykle

prováděno v rámci škol nebo na veřejných soutěžích, kde se účastníci snaží získat co nejvíce bodů za úspěšné řešení úloh, čímž lze přitáhnout nové talenty do oblasti kybernetické bezpečnosti. Zároveň jsou CTF soutěže ideální formou vzdělávání hrou, jelikož účastníci se učí nové věci formou hry, což je obvykle mnohem efektivnější než tradiční vzdělávání.

Zároveň tyto platformy umožňují jednoduše vytvářet nové úlohy a zároveň je možné je jednoduše škálovat, což je důležité pro velké soutěže, kde se může zúčastnit až několik tisíc účastníků. Toto je zajištěno samotnou architekturou těchto platforem, které v případě on-prem řešení nejčastěji využívají Kubernetes nebo jeho nadstavbou jako je třeba OpenShift od společnosti RedHat.

1.2 Cíle práce

2 VYUŽITÉ TECHNOLOGIE

2.1 BACKEND

2.1.1 Kubernetes

Tento projekt využíval na zprovoznění a správu samotných úloh Kubernetes. Kubernetes je open-source platforma pro orchestraci kontejnerů, která umožňuje automatizovat nasazování, škálování a správu aplikací běžících v kontejnerech.

Tato platforma má zároveň plnohodnotnou REST API a oficiální knihovnu pro jazyk Python, což např. systém Docker standardně neobsahuje. Zároveň má možnost vzdálené správy, díky čemuž se daly samotné mikroslužby jednoduše restartovat a aktualizovat bez nutnosti přímého přístupu na server pomocí SSH nebo webového rozhraní (např. Portainer). Toto mi později pomohlo při debugování a vývoji.

Tento systém se dá v případě potřeby škálovat a to jak na lokálních serverech, tak také na cloudu v případě, že by bylo potřeba zvýšit výkon nebo dostupnost systému.

Abych nemusel vytvářet nový systém, tak jsem se rozhodl využít Kubernetes, který běží na školní síti a je dostupný pro všechny studenty.

vCluster

Vzhledem k požadavku, aby byl systém spustitelný na školní instanci Kubernetes, jsem se rozhodl využít Kubernetes vcluster, který je dostupný pro všechny studenty a zároveň je možné vytvářet nové namespace a pody, čímž se dají mnohem jednodušeji zařizovat případná práva přes RBAC systém ke kterému žáci standardně přístup nemají.

VCluster je program, který umožňuje vytvářet virtuální clustery na již běžícím Kubernetes clusteru, čímž umožňuje získat uživateli "zvýšená"práva oproti standardnímu clusteru. Tento program mi umožnil vytvořit vlastní cluster, který byl dostupný pouze mně a zároveň mi umožnil vytvářet nové namespace a pody, což bylo důležité pro zajištění bezpečnosti a izolace jednotlivých úloh.

2.1.2 FastAPI

FastAPI je moderní webový framework napsaný v jazyce Python, který je založen na standardu ASGI (Asynchronous Server Gateway Interface). Tento framework je velmi rychlý a zároveň je velmi jednoduchý na použití a má velmi dobrou dokumentaci. Díky tomu jsem se rozhodl použít tento framework pro vytvoření API části tohoto projektu.

Tento framework mi jednoduše umožnil psát API backend v jazyce Python a zároveň mi umožnil využívat asynchronní programování, což bylo důležité pro rychlou odezvu API.

Zároveň mi tento framework velmi zjednodušil zajištění čitelnosti kódu díky vynucenému využití typování a automatickému generování dokumentace.

2.1.3 SQLAlchemy

SQLAlchemy je knihovna pro jazyk Python, která umožňuje komunikaci s relačními databázemi pomocí SQL. Tato knihovna mi umožnila vytvořit ORM (Object-Relational Mapping) pro databázi PostgreSQL, díky čemuž jsem mohl jednoduše vytvářet a upravovat databázové tabulky pomocí Pythonu bez nutnosti psát SQL dotazy a kontrolovat bezpečnost uživatelských vstupů.

2.1.4 Redis

Na uchování již nastartovaných úloh jsem použil databázi Redis. Redis je open-source in-memory databáze, která je používána pro ukládání klíč-hodnota párů. Tato databáze mi umožnila ukládat informace o běžících úlohách a zároveň mi umožnila jednoduše získávat informace o těchto úlohách.

Redis se standardně využívá např. na ukládání cache, ale v tomto projektu jsem ho využil na ukládání informací o běžících úlohách, jelikož je velmi rychlý a jednoduchý na použití.

2.1.5 Databáze

Jakožto databázi jsem zvolil PostgreSQL, která je open-source relační databáze. Tato databáze mi umožnila ukládat informace o uživatelích, úlohách a dalších informacích, které byly potřeba pro správné fungování tohoto projektu.

Tuto databázi jsem zvolil z důvodu, že je velmi rychlá, je velmi dobře podporována knihovnou SQLAlchemy a je vhodná pro produkční nasazení.

2.1.6 JWT

Na autentizaci jsem použil JWT (JSON Web Token), který mi umožnil vytvářet a ověřovat tokeny, které byly použity pro ověření uživatele v ostatních částech systému. Tento token byl podepsán pomocí privátního klíče (algoritmus RSA) a ověřován pomocí veřejného klíče. Tyto tokeny byly ukládány do cookie a zároveň byly přenášeny v hlavičce požadavku.

Tento algoritmus mi umožnil zajištění bezpečnosti uživatelů a zároveň mi umožnil jednoduché ověřování uživatelů v ostatních částech systému díky jeho bezestavové povaze.

Samotné klíče jsem ukládal do Kubernetes Secret, což mi umožnilo jednoduše měnit klíče na jednom místě bez nutnosti změn v samotných aplikacích.

2.1.7 Přístup k systémům

Kvůli nastazení jsem se rozhodl pro připojování k databázím a ke Kubernetesu používat na míru vytvořené třídy, které mi umožnily připojit se k těmto systémům a využívat funkce pro komunikaci s nimi bez nutnosti se obtěžovat s vytvářením nových připojení a zároveň mi umožnilo jednoduše měnit přístupové údaje k těmto systémům.

```
from kubernetes import client, config
from kubernetes.utils import create_from_dict

class Kubernetes:
    def __init__(self, key, url):
        self.configuration = client.Configuration()
        self.configuration.api_key['authorization'] = key
        self.configuration.api_key_prefix['authorization'] = 'Bearer'
        self.configuration.host = url
        self.configuration.verify_ssl = False
        self.client = client.ApiClient(self.configuration)
        self.v1 = client.CoreV1Api(self.client)
    def get_k8s_config(self):
        return self.v1
    def create_namespace(self, name: str):
        body = client.V1Namespace(metadata=client.V1ObjectMeta(name=name))
        return self.v1.create_namespace(body)

    def namespace_exists(self, name: str):
        namespaces = self.v1.list_namespace()
        for ns in namespaces.items:
```

```
22              if ns.metadata.name == name:
23                  return True
24          return False
25  ...
```

Kód 2.1: Ukázka třídy pro připojení ke Kubernetes

2.2 Frontend

2.2.1 React

Na frontend jsem využil toolkit Vite a knihovnu React. React je knihovna pro tvorbu uživatelských rozhraní, která je vyvíjena a udržována společností Facebook. Tato knihovna mi umožnila vytvořit jednoduché a rychlé webové rozhraní, které bylo zároveň jednoduché na údržbu a rozšíření.

Využití Reactu mi umožnilo zajištění rychlé odezvy a zároveň jednoduchého vytváření komponent, což mi zjednodušilo vytváření webového rozhraní vzhledem k tomu, že frontend nebyl primární částí tohoto projektu.

Na kód jsem využil JavaScript jelikož jsem s ním již měl zkušenosti, které jsem postrádal u TypeScriptu.

2.3 Ostatní části

2.3.1 Git submodules

Pro sdílení modelů mezi mikroslužbami jsem využil Git Submodules, které mi umožnily sdílet kód mezi jednotlivými mikroslužbami a zároveň mi umožnily jednoduše aktualizovat kód v jednotlivých mikroslužbách bez nutnosti hlídat aktuálnost zkopírovaného kódu.

Toto mi navíc umožnilo jednoduše sdílet kód mezi jednotlivými částmi systému a zároveň mi umožnilo jednoduše měnit kód v jednotlivých částech systému.

3 Řešení backendu - mikroslužby

3.1 Úvod

V této kapitole se seznámíme s tím, co mikroslužby jsou a s jednotlivými mikroslužbami použitými v API částí tohoto projektu. Všechny tyto mikroslužby jsou napsány v jazyce Python s použitím

Na autentizace byly použity JWT tokeny, které byly podepsány pomocí privátního klíče (algoritmus RSA) a ověřovány pomocí veřejného klíče. Tyto tokeny byly ukládány do cookie a zároveň byly přenášeny v hlavičce požadavku.

Tyto mikroslužby jsou:

- Router
- Auth
- Lister
- Deployer
- Deleter
- Flag-submitter

3.2 Router

Tato mikroslužba je zodpovědná za směrování požadavků na správné mikroslužby a veškeré požadavky na API putují skrz ni, díky čemuž se dá využít globální modifikace, monitorování a logování požadavků. Kvůli tomuto účelu tato služba nepotřebuje žádné privilegované přístupy do ostatních částí systému. Jednou z částí této mikroslužby je i zajištění přesunu JWT tokenu z cookie do hlavičky požadavku, aby se dala API používat jak z webového frontendu, tak i z jiných aplikací.

Tato mikroslužba zároveň funguje jakožto filtr nevalidních typů požadavků (dále posílá pouze požadavky typu GET, POST, PUT a DELETE, ostatní jsou zahozeny s chybovou hláškou)

3.3 AUTH

Tato mikroslužba je zodpovědná za registraci uživatele a vytvářením jeho záznamu v databázi PostgreSQL.

Tato služba je jediná, která má přístup k privátnímu klíči používaného k podepisování tokenů algoritmem RS256. Dále je také zodpovědná za ověření přihlašovacích údajů uživatele a vytvoření JWT tokenu, který se následně používá pro ověření uživatele v ostatních částech systému. Tato služba má přístup k databázi PostgreSQL.

Tato služba má tři API endpointy:

- POST /register
- POST /login
- GET /health

kde první dva slouží k registraci a přihlášení uživatele a třetí slouží k zjištění stavu služby, primárně kvůli liveness a readiness HTTP checku v Kubernetes při chybě nebo při čekání na databázi.

3.4 LISTER

Účel mikroslužby Lister je umožnění uživatelům získat informace o všech dostupných úlohách a jejich stavech. Dále tato služba umožňuje získat data o právě aktivních úlohách uživatele a získání detailních informací o těchto úlohách.

Tato mikroslužba potřebuje přístup k Redis a PostgreSQL databázím.

Tato služba má čtyři API endpointy:

- GET /
- GET /running
- GET /running/id
- GET /health

kde první endpoint vrací veškeré dostupné úlohy a nepotřebuje žádné přihlášení, zatímco druhý a třetí endpoint vrací informace o právě běžících úkolech uživatele, tudíž vyžadují token, s tím, že třetí vrací i detailní informace o tomto úkolu.

3.5 Deployer

Tato mikroslužba zajišťuje zapínání úkolů uživatele v systému Kubernetes a zapsání informací o této běžící službě do databáze Redis, čímž zpřístupní tato data službě Lister.

Jednotlivé úkoly jsou v Kubernetes spuštěné jakožto pody v namespace daným uživatelem, což je také jeden z důvodů užívání samostatného Kubernetes clusteru (ať už opravdového nebo vcluster) pro tyto studentské stroje - ServiceAccount spojený s tímto projektem musí mít jak práva na vytváření nových podů, tak vytváření nových namespace.

Tato služba vyžaduje přístup k Redis a PostgreSQL databázím a ke Kubernetes API.

Tato služba má dva API endpointy:

- POST /
- GET /health

kde základní endpoint vyžaduje JSON data s `challenge_id` klíčem. Dále tento endpoint potřebuje přístup k tokenu.

3.6 Deleter

Tato mikroslužba umožňuje vypínat (mazat) již vytvořené úkoly uživatele a to jak v Redis databázi, tak jejich instance běžící v systému Kubernetes.

Tato služba vyžaduje přístup k Redis databázi a ke Kubernetes API.

Tato služba má dva API endpointy:

- DELETE /id
- GET /health

kde endpoint `/{id}` vyžaduje id úkolu, který uživatel chce vypnout a JWT token uživatele.

3.7 Flag-submitter

Tato mikroslužba umožňuje odevzdávat řešení jednotlivých úkolů (vlajky).

Tato služba vyžaduje přístup k PostgreSQL databázi.

Tato služba má dva API endpointy:

- POST /flag_id
- GET /health

kde endpoint `/{flag_id}` vyžaduje v těle požadavku string `flag` a token uživatele.

4 FRONTEND

Frontend část tohoto projektu je napsána v Reactu a jakožto nástroje je využíván projekt Vite. Samotná stránka funguje na bázi CSR (Client Side Rendering) a komunikuje s API popsáným v předchozí kapitole. Díky tomuto je tato stránka zároveň kódově oddělená od backendové části a může být nasazena na jiném serveru než backend a může být psána v jiném jazyce než backendová část. Frontend je napsán v JavaScriptu díky jeho jednoduchosti a rychlosti vývoje.

První stránka, kterou člověk vidí, je přihlašovací formulář a navigační lišta. Po přihlášení se zobrazí stránka s úlohami, které může uživatel zapínat a vypínat. Dále je zde možnost zobrazit si informace o právě běžících úlohách a o všech dostupných úlohách.

Využití Reactu umožňuje stránce být tzv. SPA (Single Page Application) a tím pádem se nemusí stránka znovu načítat při každém přechodu mezi stránkami, což zvyšuje rychlost a plynulost stránky. Zároveň díky tomuto přístupu může webový server frontendu pouze posílat statické soubory a nemusí se starat o žádnou logiku, což zvyšuje bezpečnost a snižuje nároky na server.

4.1 ADMINISTRÁTORSKÁ SEKCE

Sekce pro správce v tuto chvíli obsahuje tři části - vytváření nových úloh, vytváření vlajek, aktualizace uživatelů. Prostředí v administrátorské sekci je děláno tak, aby bylo intuitivní a nebyl problém s tímhle prostředím pracovat.

Vytváření nových úloh je děláno tak, že je nutné zadat pouze název úlohy, image úlohy a kategorii úlohy - veškeré ostatní části JSON manifestu úlohy jsou generovány na straně serveru automaticky, čímž se minimalizuje prostor na bezpečnostní chyby - není potřeba kontrolovat validitu odevzdaného JSON souboru, ale pouze těchto tří částí.

Vytváření vlajek pouze požaduje identifikační číslo ulohy, ke které se vlajka váže, a vlajku samotnou.

Aktualizace uživatelů umožňuje změnit uživateli práva nebo změnit heslo uživatele.

Pro bezpečnost této části je využíváno parametru "admin"u JWT tokenu, který je vytvořen při přihlášení uživatele s právy administrátora. Díky tomuto je možné jednoduše ověřit, zda je uživatel oprávněn k použití této části vzhledem k tomu, že s JSON web tokeny nelze manipulo-

vat bez privátního klíče.

4.2 Access box

Jakožto přístup k samotným úlohám je uživateli standardně poskytnut na žádost tzv. Access box, což je kontejner s image kalilinux/kali-last-release, který se nachází ve stejném namespace a tudíž ve stejné síti jakožto úlohy uživatele.

Toto řešení má oproti řešení pomocí VPN výhodu v tom, že není nutno instalovat žádný software na straně uživatele a není nutno vytvářet certifikáty, popř. uživatele, tudíž je mnohem jednodušší na implementaci a jednodušší na škálování jelikož lze tento kontejner spustit na každém nodu v Kubernetes clusteru.

Použití Kali Linuxu je z důvodu, že je to jedna z nejznámějších distribucí pro pentesting a je také jedna z nejvíce používaných distribucí pro tento účel, což umožňuje uživateli si toto prostředí vyzkoušet.

Ke kontejneru lze přistoupit z webového prostředí, kde lze vidět terminál realizovaný pomocí REST API.

- POST /exec
- POST /create
- GET /logs
- GET /health

kde první endpoint slouží k odeslání příkazu do kontejneru, druhý k vytvoření nového kontejneru, třetí k získání logů z kontejneru a čtvrtý k zjištění stavu služby.

5 ZÁVĚR

Celkově bych řekl, že se mi podařilo splnit veškeré cíle, které jsem si stanovil na začátku tohoto projektu. Vytvořil jsem funkční systém pro vytváření a správu úloh CTF typu v systému Kubernetes, který je dostupný na školní síti a je dostatečně zabezpečený. Zároveň jsem se naučil pracovat s mikroslužbami a s prostředím Kubernetes, což byly mé hlavní cíle tohoto projektu.

V budoucnu bych chtěl do aplikace přidat možnost vytváření vlastních API tokenů, rozšířit frontend a zjednodušit administraci systému pro uživatele. Zároveň bych chtěl do aplikace přidat žebříček uživatelů, možnost vytváření týmů a možnost dynamického skórování vlajek, jako to poskytuje např. systém CTFd.

Zároveň mi tato aplikace v budoucnu může posloužit jako základ pro další projekty, kde by bylo potřeba vytvářet a spravovat úlohy CTF typu a naučila mě jak pracovat v prostředí s nízkými opravněními i přes projekt na první projekt vyžadující práva vyšší.